BAB 2
TINJAUAN PUSTAKA

2.1 Profil Perusahaan

New Cianjur (Aqobah) yang beralamat di Jalan Karanggetas No 87 Cirebon ini merupakan sebuah perusahaan yang bergerak dalam bidang retail busana muslim dan muslimah seperti menyediakan gamis, mukena, peci, jilbab, sarung, sajadah dan perlengkapan haji. *New* Cianjur (Aqobah) ini mulai berdiri pada tahun 1997 sampai saat ini.

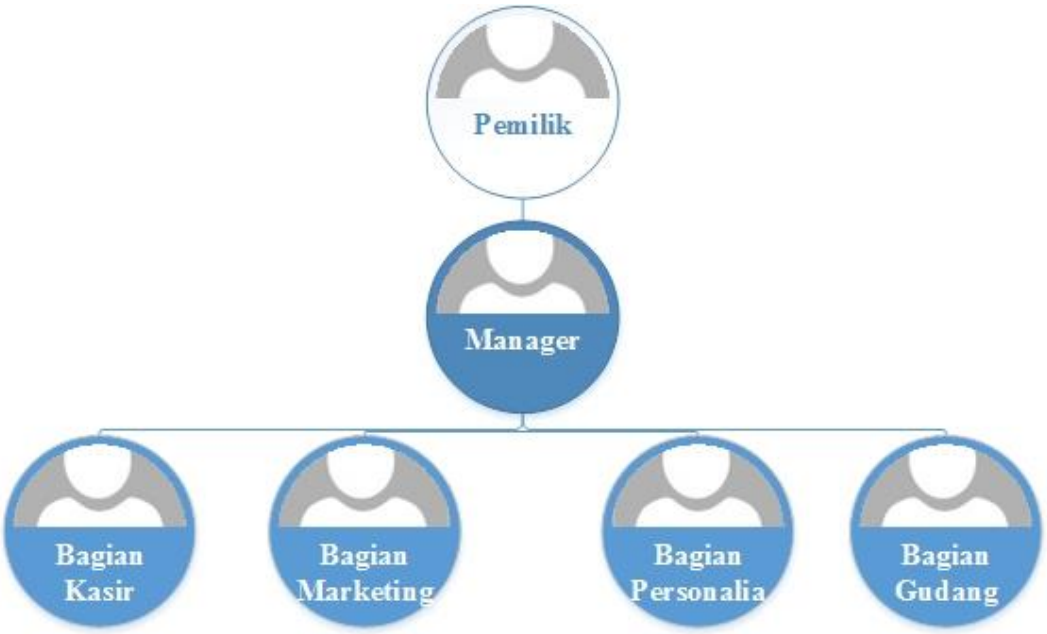
2.1.1 Sejarah Perusahaan

Sejarah singkat berdirinya *New* Cianjur (Aqobah) adalah sebuah perusahaan perseorangan yang bergerak dibidang retail busana muslim dan muslimah, perusahaan ini dibangun pada tahun 1997 mulanya usaha berawal dari modal yang tidak besar, dengan berjalan waktu usahanya mengalami perkembangan cukup signifikan hingga saat ini menjadikan *New* Cianjur (Aqobah) salah satu toko retail busana muslim dan muslimah terbesar di Cirebon.

2.1.2 Struktur Organisasi Perusahaan

Struktur organisasi merupakan gambaran formal organisasi yang menunjukkan adanya pemisahan fungsi, uraian tugas, wewenang dan tanggung jawab yang disusun untuk membantu pencapaian hasil usaha perusahaan serta pengendaliannya secara efektif. Bentuk organisasi *New* Cianjur (Aqobah) didasarkan tinjauan dari segi wewenang, tanggung jawab dan hubungan kerja dalam fungsi organisasi.

New Cianjur (Aqobah) yang termasuk ke dalam kategori perusahaan perseorangan memiliki struktur organisasi sederhana yang dapat dilihat pada gambar 2.1.



Gambar 2. 1 Struktur Organisasi Perusahaan

2.1.3 Deskripsi Tugas

Berikut ini akan diuraikan tugas-tugas dari beberapa bagian pada struktur organisasi di *New* Cianjur (Aqobah).

1 Pemilik

Pemilik adalah seorang pimpinan tertinggi dalam perusahaan.

Tugas:

Mengambil keputusan, jika perusahaan mengalami kesulitan permodalan.

2 Manager

Manager adalah seorang yang bertanggung jawab atas segala aktifitas yang terjadi di perusahaan.

Tugas :

Memimpin dan mengawasi segala aktifitas yang berjalan di tiap bagian.

3 Bagian Kasir

Kasir adalah seorang pegawai yang menangani berbagai transakasi dan mengelola keuangan.

Tugas :

a. Melayani dan mencatat transaksi penjualan.
b. Managemen piutang.

c. Mencatat penerimaan pembayaran piutang.
d. Mencatat pengiriman barang.
e. Membuat laporan keuangan.
f. Membuat pertanggungjawaban dan bertanggung jawab terhadap kebenaran uang yang diterima dan dikeluarkan.

4 Bagian Marketing

Marketing adalah bagian dimana aktifitas pemasaran di perusahaan.

Tugas :

a. Memperkenalkan profile perusahaan ke masyarakat.
b. Menjalin hubungan baik dengan pelanggan.
c. Membuat strategi pemasaran.

5 Bagian Personalia

Personalia adalah bagian pengelolaan sumber daya manusia (SDM).

Tugas :

a. Rekruitmen karyawan.
b. Perencanaan karir karyawan.
c. Kontrol karyawan.

6 Bagian Gudang

Gudang adalah bagian dimana aktifitas penyimpanan barang.

Tugas :

a. Cek penerimaan dan pengeluaran barang.
b. Mengeluarkan barang.
c. Membuat laporan penerimaan dan pengeluaran barang.

2.2 Landasan Teori

Landasan teori merupakan definisi, konsep yang telah disusun secara sistematis dan dasar yang kuat dalam sebuah penelitian. Landasan teori yang digunakan dalam Pembangunan Aplikasi Android Dengan Memanfaatkan Tag NFC (Near Field Communication) Untuk Keanggotaan Konsumen Di New Cianjur (Aqobah) meliputi Kartu Anggota, Kartu Tanda Penduduk Elektronik, Android, Basis Data, MySQL, Pemrograman Berorientasi Objek, Konsep Dasar Berorientasi Objek, *Near Field Communication*, JSON, PHP, Mysql, *Unified Modeling Language* (UML), Web Service.

2.2.1 Kartu Anggota

Kartu anggota adalah kartu yang hanya dimiliki oleh anggota perusahaan atau perkumpulan. Kartu Anggota biasanya menawarkan keuntungan-keuntungan tertentu terhadap pemegangnya.

Kartu Anggota membuat anggota dari perusahaan atau organisasi anda akan merasa lebih eksklusif dan dapat keuntungan tertentu. Kartu Member dapat menunjukkan identitas perusahaan atau organisasi anda dan membuat anggota / staff merasa lebih dihargai.

2.2.2 Android

Android merupakan sistem operasi mobile yang tumbuh di tengah sistem operasi lainnya yang berkembang dewasa ini. Sistem operasi lainnya seperti *Windows Mobile, i-Phone OS,* Symbian dan masih banyak lagi juga menawarkan kekayaan isi dan keoptimalan berjalan di atas perangkat hardware yang ada. Akan tetapi, sistem operasi yang ada ini berjalan dengan memprioritaskan aplikasi inti yang dibangun sendiri tanpa melihat potensi yang cukup besar dari aplikasi pihak ketiga. Oleh karena itu, adanya keterbatasan distribusi aplikasi pihak ketiga untuk platform mereka.

Android menawarkan sebuah lingkungan yang berbeda untuk pengembang. Setiap aplikasi memiliki tingkatan yang sama. Android tidak membedakan antara aplikasi inti dengan aplikasi pihak ketiga. *Application Programming Interface (API)* yang disediakan menawarkan akses ke *hardware*, maupun data-data ponsel

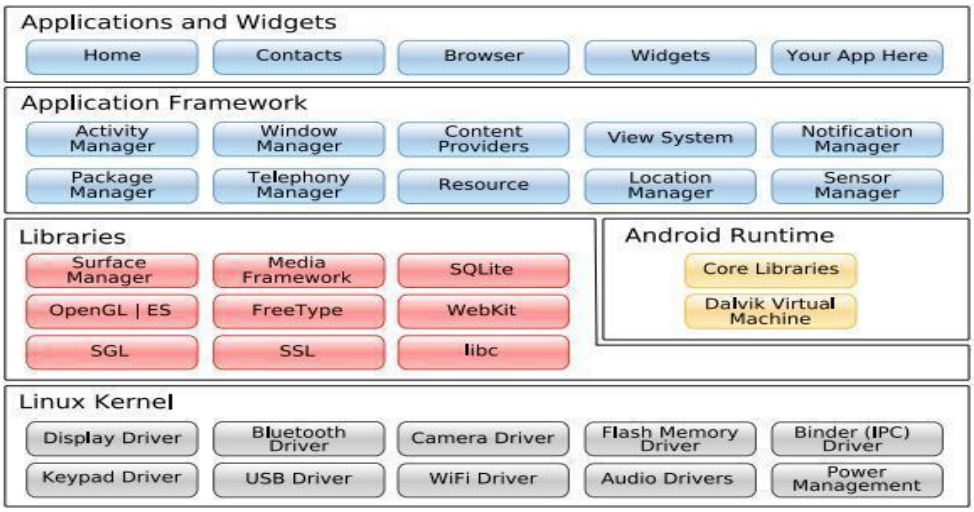
sekalipun, atau data sistem sendiri. Bahkan pengguna dapat menghapus aplikasi inti dan menggantikannya dengan aplikasi pihak ketiga [4].

2.2.3 Arsitektur Android

Arsitektur Android dapat dijelaskan sebagai berikut:

a. *Application* dan *Widgets*

Application dan *Widgets* ini adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di layer terdapat aplikasi inti termasuk klien *email*, program *SMS*, kalender, peta, *browser*, kontak, dan lain-lain. Hampir semua aplikasi ditulis menggunakan bahasa pemrograman Java.



Sumber Gambar : N. Safaat H [4]

Gambar 2. 2 Arsitektur Android

b. Aplication Framework

Android adalah *"Open Development Platform"* yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resource*, menjalankan *service background*, mengatur alarm, dan menambah status *notifications*, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat

menggunakan kembali komponen yang sudah digunakan (*reuse*). Sehingga bisa kita simpulkan *Application Frameworks* ini adalah layer dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti *content providers* yang berupa sms dan panggilan telepon. Komponen-komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut :

1. *Views*
2. *Content Provider*
3. *Resource Manager*
4. *Notification Manager*
5. *Activity Manager* [4]

c. Libraries

Libraries ini adalah layer dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Berjalan di atas Kernel, layer ini meliputi berbagai library C/C++ inti seperti Libc SSL, serta :

a. Libraries media untuk pemutaran media audio dan video
b. Libraries untuk manajemen tampilan
c. Libraries *Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
d. Libraries SQLite untuk dukungan database
e. Libraries SSL dan *WebKit* terintegrasi dengan web *browser* dan *security*
f. Libraries *LiveWebcore* mencakup modern *web browser* dengan *engine embedded webview*
g. Libraries 3D yang mencakup implementasi Open GL ES1.0 API's. [4]

d. Android *Run Time*

Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. Dalvik Virtual Machine (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android *Run Time* dibagi menjadi dua bagian yaitu:

a. *Core* Libraries: Aplikasi Android dibangun dalam bahasa Java, sementara Dalvik sebagai virtual mesinnya bukan Virtual *Machine* Java, sehingga

diperlukan sebuah libraries yangberfungsi untuk menterjemahkan bahasa Java/C yang ditangani oleh Core Libraries.

b. Dalvik Virtual Machine: Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat Linux Kernel untuk melakukan threading dan manajemen tingkat rendah [4].

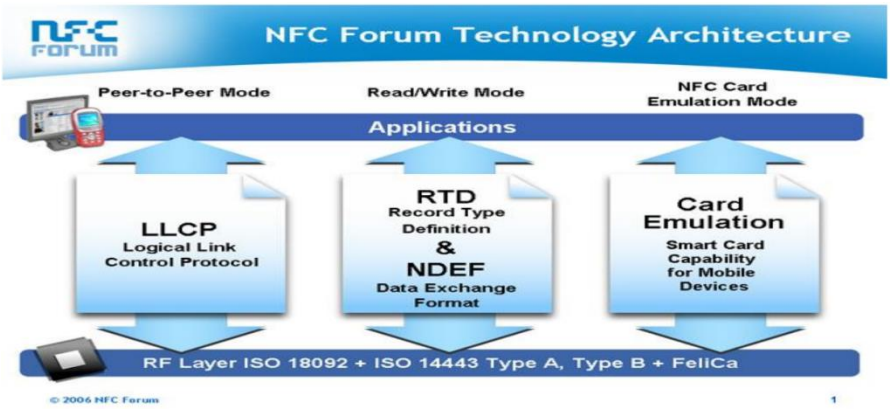
e. Linux Kernel

Linux Kernel adalah layer dimana inti dari system operasi Android itu berada. Berisi file-file *system* yang mengatur sistem *processing, memory, resource, drivers,* dan sistem-sistem operasi Android lainnya. Linux Kernel yang digunakan Android adalah Linux Kernel release 2.6.

2.2.5 Near Field Communication

Near Field Communication (NFC) merupakan teknologi komunikasi baru dengan menggunakan induksi magnet berbasis teknologi Radio *Frequency Identification (RFID). NFC* mulai dikembangkan pada tahun 2002 oleh NXP dan Sony. Mereka berhasil membangun Forum *NFC* pertama bersama dengan Nokia. Sejak saat itu Forum *NFC* memiliki lebih dari 150 anggota dan peneliti yang terlibat dari beberapa perusahaan dan organisasi. Pada forum ini juga *NFC* dipromosikan dan secara resmi mengeluarkan sertifikasi untuk perangkatperangkat yang mengadopsi teknologi *NFC*. *NFC* bisa juga disebut sebagai generasi kedua dari teknologi *RFID*.

NFC beroperasi pada frekuensi 13,56 MHz dengan kecepatan transmisi pengiriman mencapai 424 kbit/s. Jarak transmisi *NFC* sekitar 4-10 cm. Perbedaan antara *NFC* dan teknologi komunikasi *contactless* lainnya yaitu perangkat *NFC* dapat bersifat aktif – aktif (*peer to peer*) dan aktif – pasif. Oleh karena itu *NFC* selalu melibatkan inisiator (*reader*) dan target. Inisiator aktif menghasilkan medan *RF* (*Radio Frequency*) yang dapat memberikan kekuatan ke target yang pasif (tidak memiliki sumber daya). Hal ini memungkinkan target *NFC* untuk memiliki bentuk yang sangat sederhana seperti stiker, gantungan kunci, atau kartu yang tidak memerlukan energy khusus [5].



Sumber Gambar : NFC Forum [5]

Gambar 2. 3 Arsitektur NFC

Banyak manfaat kegiatan-kegiatan yang diperoleh dengan memanfaatkan tekonolgi *NFC*. Seiiring perkembangan jaman, inovasi-inovasi terhadap pemanfaatan *NFC* ini juga ikut berkembang. Perkembangan inovasi-inovasi tersebut terbagi ke dalam 3 kategori:

1. *Service Initiation*

Pada kategori ini *NFC* berguna untuk melakukan layanan sebagai media untuk mendapatkan informasi tertentu atau dapat juga untuk membuka layanan yang mengharuskan autentikasi data terlebih dahulu. Hal ini melibatkan *NFV Tag* sebgai media penyimpanan maupun penyampaian sementara. Pada *Tag* tersebit dapat berisi informasi tejs, alamat sebuah *website* (*url*), nomor telepon atau data lain dengan ukuran yang kecil.

Salah satu contoh penggunaannya yaitu pada poster pintar (*smart poster*). Dengan mendekatkan ponsel kepada *NF Tag* pada poster, maka pengguna akan langsung mendapatkan informasi lebih detail pada yang belum tertera pada poster tersebut tanpa harus menginputkan apapun pada ponselnya.

2. *Peer-to-peer*

Pada kategori ini *NFC* berperan dalam mengkomunikasikan antara dua perangkat aktif dengan ukuran data yang relative kecil. Namun untuk kebutuhan data yagn berukuran besar, *NFC* dapat digunakan sebagai sambungan nirkael seperti *Bluetooth* dan *WiFi*.

Contoh dari penggunaannya yaitu pada proses pencetakan foto langsung dari ponsel atau kamera. Pengguna tinggal memilih foto yang akan dicetak, kemudian mendekatkan ponselnya pada printer yang sdah terintregrasi dengan teknologi *NFC* untuk mebuat sambungan *Bluetooth* untuk mengirimkan gambar dan mencetaknya.

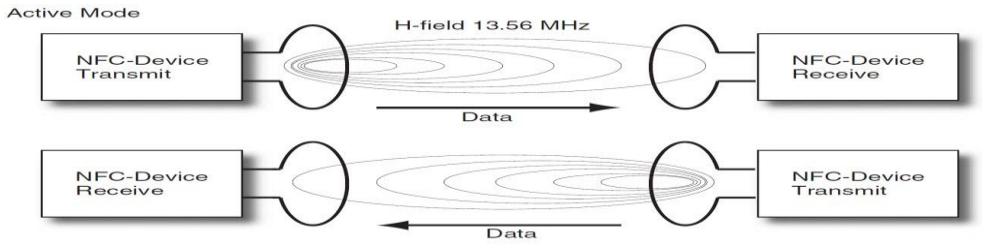
3. *Payment*

Pada kategori ini *NFC* berperan dalam proses pembayaran suatu barang yang sudah terintegrasi sebelumnya dengan kartu kredit, bank dan operator jaringan lainnya. Dengan memanfaatkan teknologi *NFC* dalam pembayaran ini dapat mempermudah dan mempercepat proses transaksi pembayaran karena tidak menggunakan uang tunai maupun metode pembayaran konvensional lainnya. Contoh dari penggunaanya yaitu pada proses transaksi pembayaran di restoran, pusat perbelanjaan dan area parkir.

Komunikasi *NFC* dapat dilakukan dalam 2 mode yaitu mode aktif dan mode pasif.

1. Mode Aktif

Pada mode aktif, *NFC* yang berperan sebagai initiator mengaktifkan transmiter yang dimiliki sehingga arus dengan frekuensi tinggi yang mengalir pada antena menimbulkan medan magnetik H yang menyebar ke sekitarnya seperti terlihat pada Gambar 2.17.



Sumber Gambar : NFC Forum [5]

Gambar 2. 4 Mode Aktif

Sebagian akan bergerak dan tertangkap oleh antena interface *NFC* yang terletak pada lokasi berdekatan. Kemudian tegangan U dihasilkan di antena yang dapat dideteksi dan diterima oleh *receiver* pada *NFC* target.

Cara kerja tersebut mirip dengan konsep *RFID Reader* dan *transponder*. Namun, terdapat perbedaan dimana pada transponder *RFID* reader harus

memberikan suplai energi kepada *transponder* untuk mengoperasikan microchip untuk komunikasi data. Sementara pada *NFC*, perangkat elektronik yang memiliki fitur *NFC* yang memberikan suplai energi.

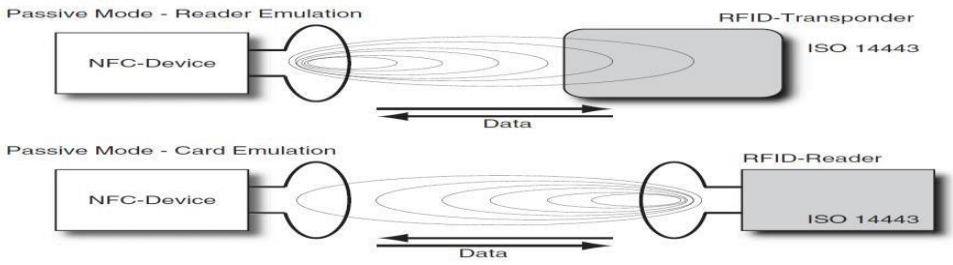
Pada mode aktif, pengiriman data dilakukan dari *NFC* target kepada *NFC* initiator dengan *NFC* target mengaktifkan transmitter dan *NFC* initiator berpindah mengaktifkan receiver. Sehingga pada mode ini kedua interface *NFC* bergantian menghasilkan medan magnetik.

2. Mode Pasif

Pada mode pasif, *NFC* inisiator juga menghasilkan medan magnetik untuk memulai komunikasi dengan *NFC* target. Namun setelah paket data sampai ke target, *NFC* inisiator tidak mengganti mode menjadi penerima seperti *NFC* mode aktif, melainkan tetap mengaktifkan transmitter. Sehingga trasmitter hanya diaktifkan pada *NFC* inisiator.

Kemudian data pada *NFC* target akan "ditumpangkan" pada gelombang tersebut dengan menghasilkan yang disebut sebagai *load modulation*. Hal ini memberikan keuntungan dalam pengoperasiannya pada kondisi daya yang dimiliki oleh *NFC* target sedang lemah.

Hal ini juga memungkinkan *interface NFC* berfungsi seperti *RFID Reader* untuk berkomunikasi dengan transponder pasif dengan memberikan suplai daya pada transponder dan menggunakan *load modulation* data ditransimisikan pada *interface NFC*. Misalnya, *smartphone* dengan fitur *NFC* dapat digunakan untuk membaca dan menulis pada transponder seperti *smart label* atau *e-ticket*. Aktivitas *interface NFC* yang menyerupai *RFID Reader* disebut juga '*reader-emulation mode*'.



Sumber Gambar : NFC Forum [5]

Gambar 2. 5 Mode Pasif

2.2.6 *NFC Tag*

NFC Tag merupakan media pendukung teknologi *NFC*. *NFC* Tag biasanya bersifat pasif, sehingga membutuhkan inisiator (*NFC* bersifat aktif) untuk proses komunikasi. *NFC Tag* dapat berisi data dengan ukuran kecil. NFC *Tag* saat ini memiliki *memory* berkisar antara 96 dan 4.096 byte. NFC *Tag* berisfat *read-only* tetapi memungkinkan juga bersifat *re-writeable* [6].



Sumber Gambar : D. Coleman [6]

Gambar 2. 6 NFC Tag

Produsen *NFC* telah menentukan standar dari *NFC Tag* untuk dapat dioperasikan dengan perangkat *NFC*. Spesifikasi yang ditentukan terbagi menjadi 4 tipe. Keempat tipe *Tag* NFC tersebut didasarkan pada produk dan ketersediaan komersil. Berikut adalah 4 tipe dari *NFC Tag*:

1. Tipe 1 : didasari pada ISO/IEC 14443A. *Tags* mampu membaca dan menulis ulang. Pengguna dapat mengatur tag menjadi *read-only*. Memori yang tersedia adalah 96 bytes dan dapat di-*upgrade* hingga 2 Kbyte.
2. Tipe 2 : didasari pada ISO/IEC 14443A. *Tags* mampu membaca dan menulis ulang. Pengguna dapat mengatur tag menjadi *read-only*. Memori yang tersedia adalah 48 *bytes* dan dapat di upgrade hingga 2 *Kbyte*.
3. Tipe 3 : didasari pada standar industri Jepang atau *Japanese industrial standard* (JIS) X 6319-4, atau disebut *Felica*. *Tag* diatur pada pembuatannya untuk mendukung baik dalam *read* dan *re-writeable*, ataupun *read-only*. Memori yang tersedia dapat diubah, secara teori batas memorinya adalah 1 *Mbyte* per *service*.

4. Tipe 4 : kompatibel sepenuhnya dengan ISO/IEC 14443 standard *series*, memiliki memori 32 KB dan lebih cepat daripada tipe *tag* lainnya, memiliki proteksi terhadap data *collision*, dapat menggunakan komunikasi NFC-A maupun *NFC*-B. Tipe 4 dapat ditulis ulang *(rewriteabl*e) atau *read-only*. Tetapi pengaturan tersebut dilakukan saat tipe 4 dalam tahap pembuatan. Saat sudah selesai dibuat, pengaturannya tidak dapat diubah [6].

2.2.7 *NFC Reader*

NFC reader adalah perangkat yang menerima atau mengirimkan perintah dari *NFC. NFC reader* akan sangat berguna bagi pedagang yang ingin menyediakan layanan pembayaran menggunakan *NFC* [7].

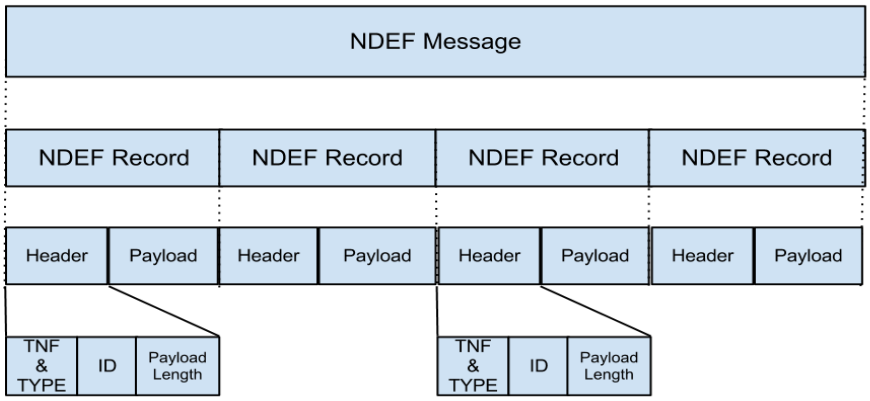


Sumber Gambar : Made Krisnanda [7]

Gambar 2. 7 Reader (Near Field Communication)

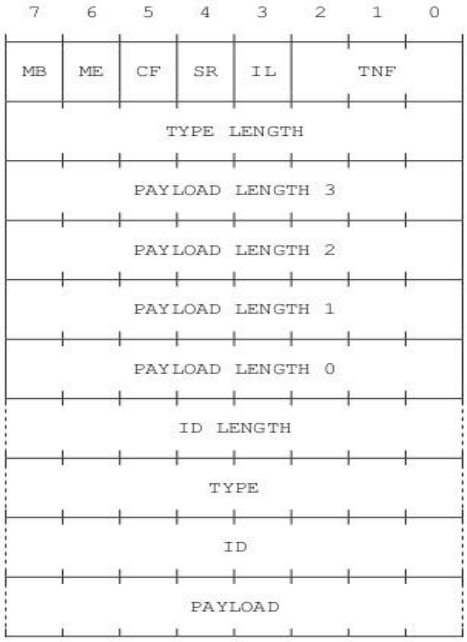
2.2.8 *NFC* Data *Exchange* Format (*NDEF*)

NFC Data Exchange Format (*NDEF*) adalah format pesan standar pada proses pertukaran informasi antara perangkat NFC ke perangkat NFC dan perangkat *NFC* ke *Tag NFC*. Di dalam *NDEF* terdapat *NDEF Message* yang terdiri dari beberapa *record* seperti pada gambar 2.8.



Gambar 2. 8 Struktur NDEF Message

NDEF Record merupakan format umum dari panjang variabel *record.* Berikut adalah struktur *NDEF Record.*



Gambar 2. 9 Struktur NDEF Record

Berikut adalah keterangan dari struktur *NDEF Record*:

1. MB (*Message Begin*) : Area 1 bit. Ketika diset maka akan menunjukan awal *NDEF Message.*
2. ME (*Message End*) : Area 1 bit. Ketika diset maka akan menunjukan akhir *NDEF Message*.

3. CF (*Chunk Flag*) : Area 1 bit. Untuk mengidentifikasi apakah potongan *record* pertama atau potongan tengah *record*.
4. SR *(Short Record)* : Area 1 bit. Jika diset maka PAYOAD_LENGTH merupaan oktet tunggal. Short Record ini dimaksudakn untuk mengkapsulasi *Payloadi,* dengan kisaran ukuran 0 sampai 255
5. IL (*Identification Length*) : Area 1 bit. Jika diset maka ID_LENGTH akan muncul di bagian *header* sebagai oktet tunggal. Jika ID_LENGTH bernilai 0. ID_LENGTH dihilangkan dari *header record* dan area ID dihilangkan dari *record.*
6. TNF (*Type Name Format*) : Area bernilai 3 bit. Untuk mengidentifikasi struktur dari *value* TYPE.

Tabel 2. 1 Value Type Name Format

Type Name Format	***Value***
Empty	0x00
NFC Forum well-known type [NFC RTD]	0x01
Media-type as defined in RFC 2046 [RFC 2046]	0x02
Absolute URI as defined in RFC 3986 [RFC 3986]	0x03
NFC Forum external type [NFC RTD]	0x04
Unkown	0x05
Unchanged	0x06
Reserved	0x07

Keterangan tabel 2.1 *Value Type Name Format*:

1. 0x00 *(Empty)*

   Menunjukan bahwa tidak ada jenis atau payload yang terkait dengan *record*. Ketika digunakan TYPE_LENGTH, ID_LENGTH dan bidang

PAYLOAD_LENGTH harus bernilai nol dan TYPE_ID dan PAYLOAD dihilangkan dari *record*. Nilai TNF ini dapat digunakan setiap kali *record* kosong diperlukan, misalnya untuk mengakhiri NDEF *message* ketika tidak ada PAYLOAD yang diinputkan oleh pengguna aplikasi.

2. 0x01 (NFC Forum wel-known type [NFC RTD])
   Menunjukan bahwa TYPE berisi nilai yang mengikuti *Type Name Format* RTD sesuai spesifikasi pada NFC Forum.
3. 0x02 (Absolute URI as defined in RFC 3986 [RFC 3986])
   Menunjukan bahwa TYPE berisi nilai yang sesuai dengan yang dibuat oleh BNF dan didefinisikan dalam RFC 2046.
4. 0x03 (Absolute URI as defined in RFC 3986 [RFC 3986])
   Menunjukan bahwa TYPE berisi nilai yang mutlak sesuai dengan URI BNF dan didefinisikan dalam RFC 3986.
5. 0x04 (NFC Forum external type [NFC RTD])
   Menunjukkan bahwa bidang TYPE berisi nilai yang mengikuti *Type Name Format* RTD yang didefinisikan dalam NFC Forum RTD spesifikasi [NFC RTD] untuk nama jenis eksternal.
6. 0x05 (Unkown)
   Menunjukan jika tipe PAYLOAD tidak diketahui. Hal ini serupa dengan “application/octet-stream” jenis media yang didefinisikan oleh MIME RFC 2046. Ketika digunakan TYPE_LENGTH harus bernilai nol. Maka dengan demikian TYPE dihilangkan dari NDEF *record*.
7. 0x06 (Unchanged)
   Hanya untuk *record* tengah dan mengakhiri *record* dari Chunked Payload. TYPE_LENGTH harus bernilai nol.
8. 0x07 (Reserved)
   Milik NFC Forum untuk digunakan di masa depan.
9. TYPE_LENGTH
   TYPE_LENGTH adalah 8bit integer yang menentukan panjang oktet dari TYPE

10. ID_LENGTH

    ID_LENGTH adalah 8bit integer yang menentukan panjang oktet dalam ID. ID_LENGTH muncul apabilan IL diset menjadi 1 di dalam *record header*. ID_LENGTH boleh bernilai 0 oktet namun ID harus dihilangkan dari NDEF *record.*

11. PAYLOAD_LENGTH

    PAYLOAD_LENGTH adalah yang menentukan panjang oktet dari PAYLOAD.

12. TYPE

    Nilai TYPE adalah identifikator yang menjelaskan jenis nilai PAYLOAD. Nilai dari TYPE harus sesuai dengan struktur, encoding dan format pada TNF.

13. ID

    Nilai dari ID adalah identifier dalam bentuk resferensi URI yang dijelaskan dalam RFC 3986. Referensi URI dapat bersifat relative dan absolut karena NDEF tidak mendefiniskan basis URI.

14. PAYLOAD

    Konten yang dapat berisi apa saja dalam ukuran beberapa bytes.

2.2.9 JSON

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

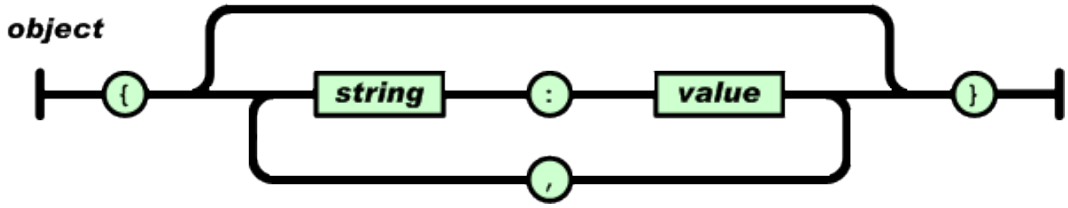
1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), daftar berkunci (*keyed list*), atau *associative* array.

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 sruktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut:

1. Objek

Objek adalah sepasang nama / nilai yang tidak terurutkan. Seperti yang diilustrasikan pada Gambar 2.8 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). Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON.

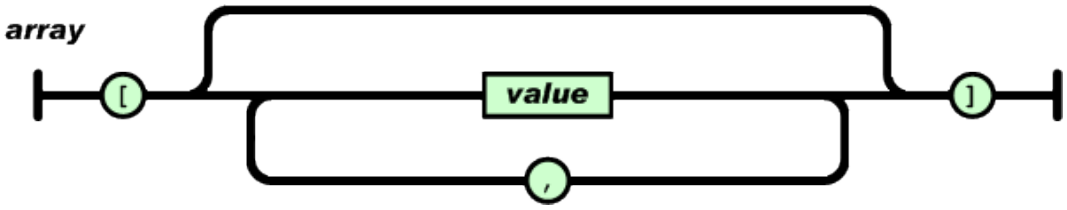


Sumber Gambar : Introducing JSON [8]

Gambar 2. 10 Objek JSON

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma) seperti yang diilustrasikan pada Gambar 2.9. Larik dalam JSON dapat digunakan sebagai value dari JSON object hal ini dapat berguna jika JSON menyimpan data bertingkat [8].



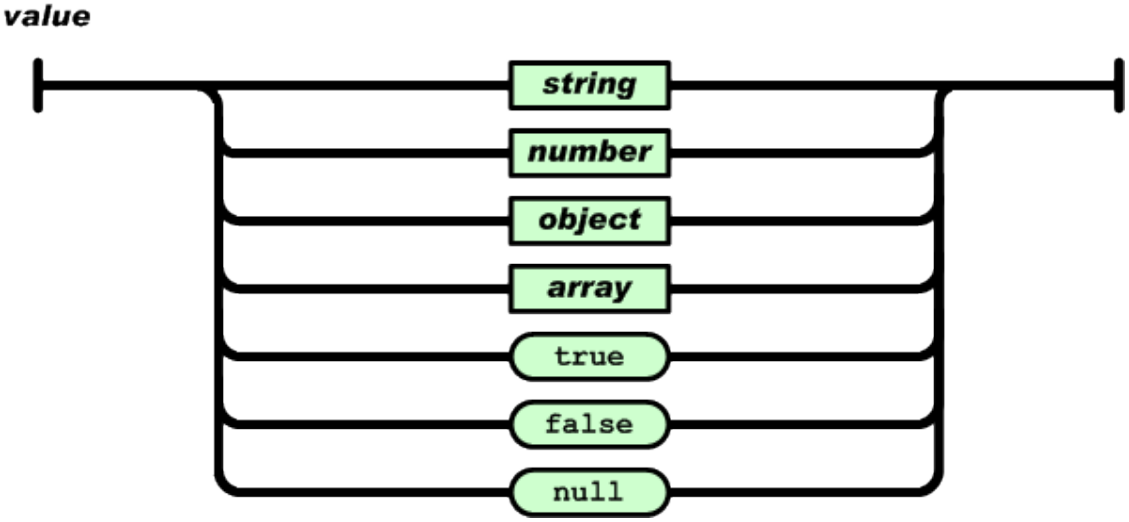
Sumber Gambar : Introducing JSON [8]

Gambar 2. 11 Array JSON

Bentuk data JSON objek dan larik dapat saling dikombinasikan untuk mendukung struktur data yang lebih kompleks. JSON mendukung beberapa tipe data untuk menjadi value seperti Angka, String, Bollean dan Nilai NULL [8].

3. Nilai

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 bertingka.



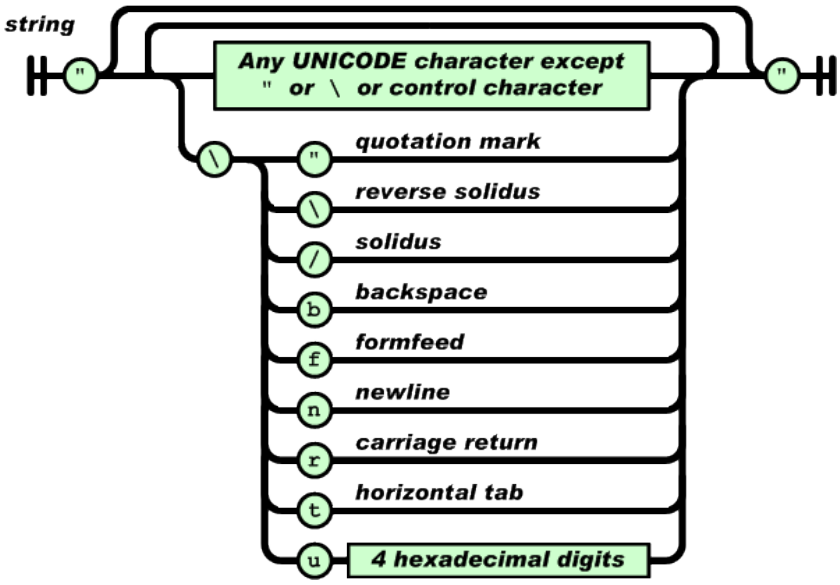
Sumber Gambar : Introducing JSON [8]

Gambar 2. 12 Value JSON

4. String

String adalah kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash

escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java

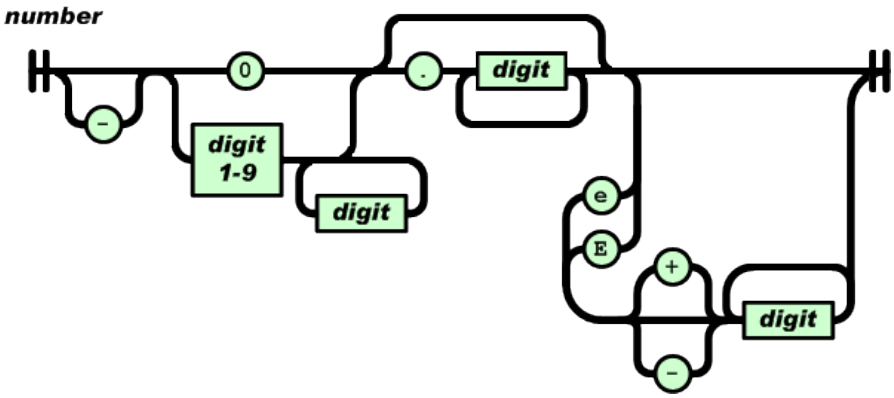


Sumber Gambar : Introducing JSON [8]

Gambar 2. 13 String JSON

5. Angka

   Angka, Format oktal dan heksadesimal tidak digunakan.



Sumber Gambar : Introducing JSON [8]

Gambar 2. 14 Number JSON

2.2.10 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientas objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek [9].

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekata terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini beragam dengan platform yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut. Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (reusable).
2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengodean.
3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Karena pendekatan pengembangan lebih dekat dengan dunai nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.2.9.1 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi, atau sistem lainnya). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata. Ada banyak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dan abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dipunyai oleh objek akan dienkapsulasi (dibungkus) menjadi suatu kesatuan [9].

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancanggan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus (dienkapsulasi) menjadi kelompok daata dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat dan komponen lainnya, dan dapat berinteraksi satu sama lain.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

1. Kelas (*Class*)

   Kelas adalah sekumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (metode/operasi), hubungan *(relationship)* dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara tekniks kelas adalah sebuah struktur dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

2. Objek (*object*)

   Objek adalah abstraksi dari sesuatu yang mewakili dunia nyata benda, manusia, satua organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstra. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan. Secara teknis, sebuah kelas saat program dieksekusi makan akan dibuat sebuah objek. Objek dilihat darisegi teknis adalah elemen pada saat runtime yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehinga sebuah objek hanya ada saat sebuah program dieksekusi. Jika masih dalam bentuk kode, disebut sebagai kelas jadi pada saat runtime (saat sebuah program dieksekusi), yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas.

3. Metode (*method*)

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode atau operasi dapat berasal dari *event*, aktifitas atau aksi

keadaan, fungsi, atau kelakuan dunnia nyata. Contoh metode atau operasi misalnya *Read*, *Write*, *Move*, *Copy*, dan sebagainya.

4. Atribut (*attribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat beruoa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkusa atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja-nya.

7. Pewarisan (*inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

8. Antarmuka (*interface*)

   Antarmuka sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antarobjek

    Komunikasi antarobjek dilakukan lewat pesan *(message)* yang dikirim dari satu objek ke objek lainnya.

12. Polimorfisme (*polymorphism*)

    Kemampuan suatu objek digunakan di banyak tujuan yang berbeda dengan nama yang sehingga menghemat baris program.

13. *Package*

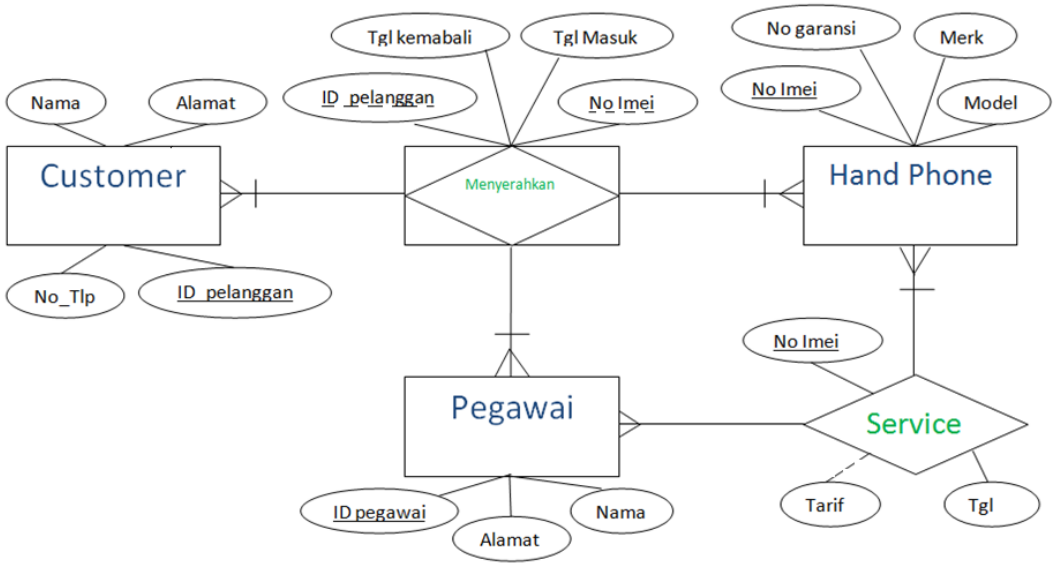
    Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda.

2.2.11 *Structured Analysis and Design*

Structured analysis and *design* adalah pendekatan sistem analisis dan desain sistem informasi secara hierarki. SADM diproduksi untuk CCTA, kantor pemerintah Inggris yang berkaitan dengan penggunaan teknologi di pemerintahan dari tahun 1980 dan seterusnya [14].

1. Entity Relationship Diagram (ERD)

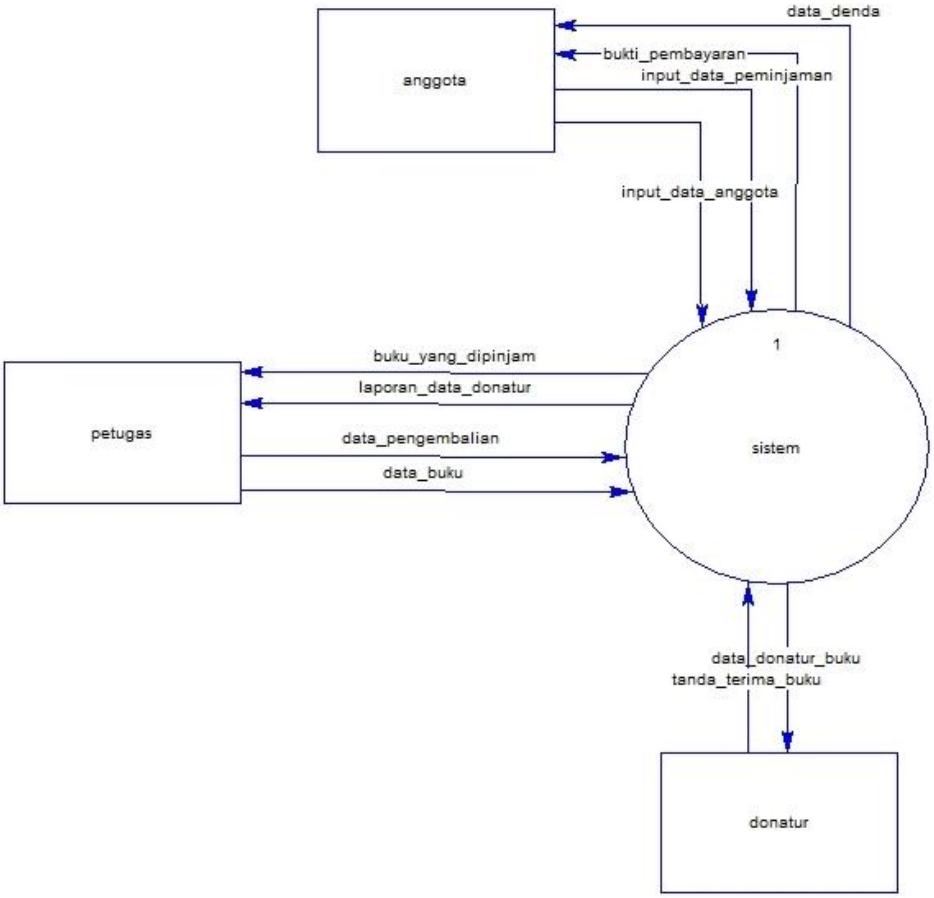
ERD *(Entity Relationship Diagram)* adalah satu model jaringan yang menggunakan susunan data yang disimpan dalam sistem. ERD merupakan model jaringan data yang menekankan pada struktur dan hubungan antardata.



Gambar 2. 15 Contoh ERD

2. Diagram Konteks

Diagram Konteks, adalah data *flow* diagram tingkat atas (DFD *Top Level*), yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar entitas-entitas eksternal. (CD menggambarkan sistem dalam satu lingkaran dan hubungan dengan entitas luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem).



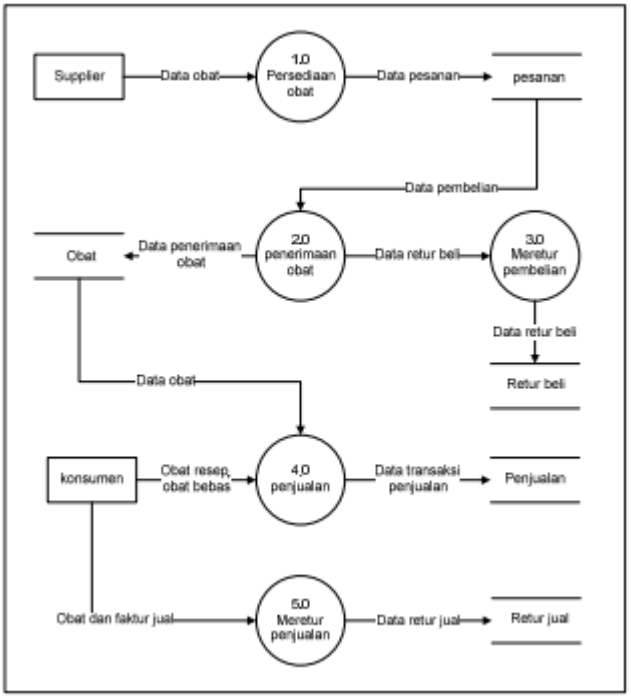
Gambar 2. 16 Contoh Diagram Konteks

3. Data *Flow Diagram* (DFD)

Data Flow Diagram (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data dimana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut. DFD terdiri dari empat komponen utama diantaranya adalah :

1. Entitas.
2. Proses.
3. *Data Store.*
4. *Data Flow*.

DFD dapat dikatakan merupakan *breakdown* dari *diagram konteks*. Analisis mengenai sistem yang lebih detail dapat dilakukan dengan melakukan *breakdown* pada DFD level 1, 2 dan seterusnya.



Gambar 2. 17 Contoh Data Flow Diagram

4. Spesifikasi Proses

Spesifikasi Proses adalah tabel yang berisi keterangan atau deskripsi dari semua proses yang terdapat pada DFD. Logika proses yang ditulis pada spesifikasi proses harus dituliskan secara jelas baik menggunakan bahasa deskriptif atau *pseudecode* tetapi tidak boleh mengkombinasikan kedua bahasa tersebut. Berikut adalah Tabel 2.1 Contoh Spesifikasi Proses:

Tabel 2. 2 Contoh Spesifikasi Proses

No	Proses	Keterangan
1.	No Proses	Menyatakan no proses
	Nama Proses	Menyatakan nama proses
	Source (Sumber)	(Menyatakan sumber data input menuju proses)
	Input	(menyatakan isi data yang masuk ke proses)
	Output	(menyatakan informasi yang keluar dari proses)
	Destination	(menatakan tujuan informasi output dari proses)
	Logika Proses	(menyatakan algoritma dari proses)

5. Kamus Data

Kamus data adalah tabel yang berisi deskripsi dari data yang mengalir pada DFD.

Tabel 2. 3 Contoh Kamus Data

Nama aliran data	Data Siswa Nama dari data yang digunakan
Where used/How used	Daftar pada poin proses apa data digunakan
Keterangan	Uraian singkat dari data yang digunakan
Struktur Data	Daftar komponen data yang ada pada data yang digunakan
Deskripsi	Jenis data dalam representasi komputer untuk masing-masing data.

2.2.12 *Unified Modeling Language* (UML)

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. Pengertian berorientasi objek berarti bahwa cara mengorganisasikan perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data perilakunya.

Konsep OOAD (*Object Oriented Analysis Design*) mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu:

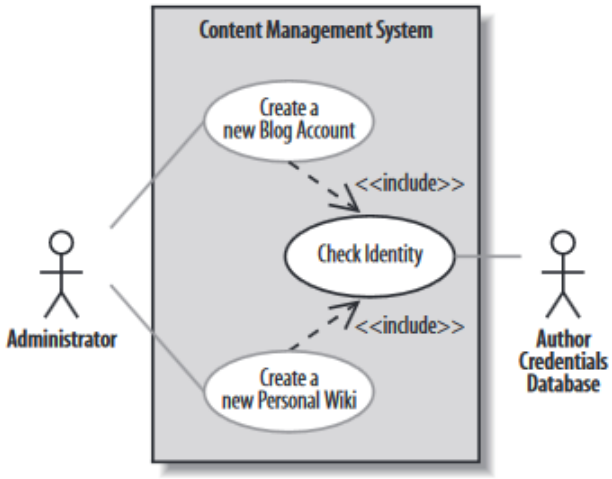
1. Analisis berorientasi objek (OOA)

Metode analisis yang memerika *requirement* (syarat/keperluan) yang harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. Unified Modeling Language (UML) adalah bahasa untuk mengspesifikasi, memvisualisasi, membangun dan mendokumentasi artefacts (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak), seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya. UML merupakan bahasa standar untuk penulisan blueprint software yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian alat-alat dari sistem perangkat lunak.

UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat. Berikut ini merupakan beberapa bagian dari UML adalah sebagai berikut :

a. *Use Case Diagram*

   Use case diagram menggambarkan sejumlah eksternal aktor dan hubungannya ke use case yang diberikan oleh sistem.

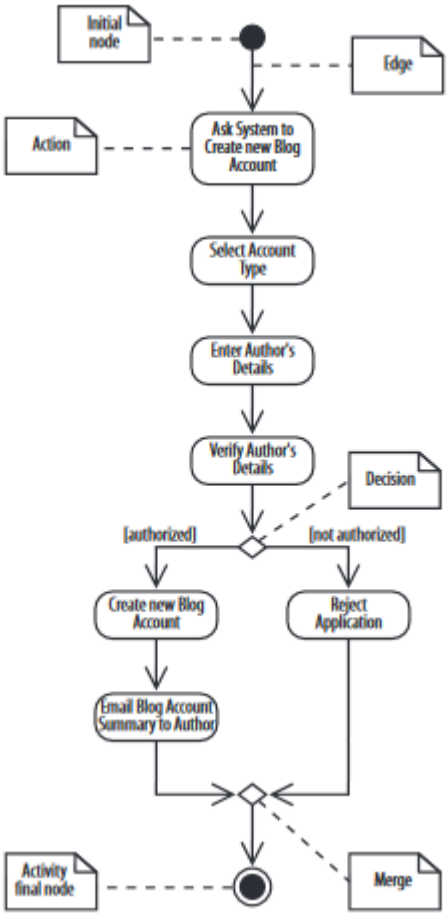


Sumber Gambar : Learning UML 2.0 [10]

Gambar 2. 18 Contoh Use Case Diagram

b. Activity Diagram

Activity diagram menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainya seperti *use case*.

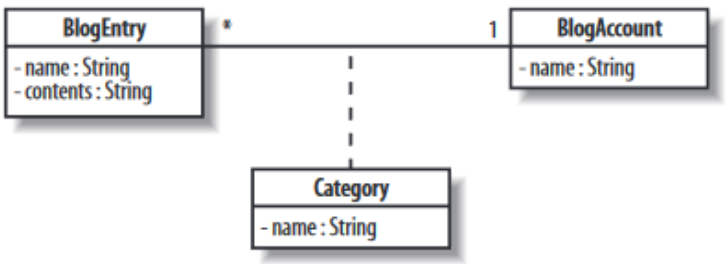


Sumber Gambar : Learning UML 2.0 [10]

Gambar 2. 19 Contoh Activity Diagram

c. *Class Diagram*

Class diagram menggambarkan struktur dan deskripsi class, package (paket) dan objek beserta hubungan satu sama lain seperti containment (penahanan), pewarisan, asosiasi dan lain-lain.

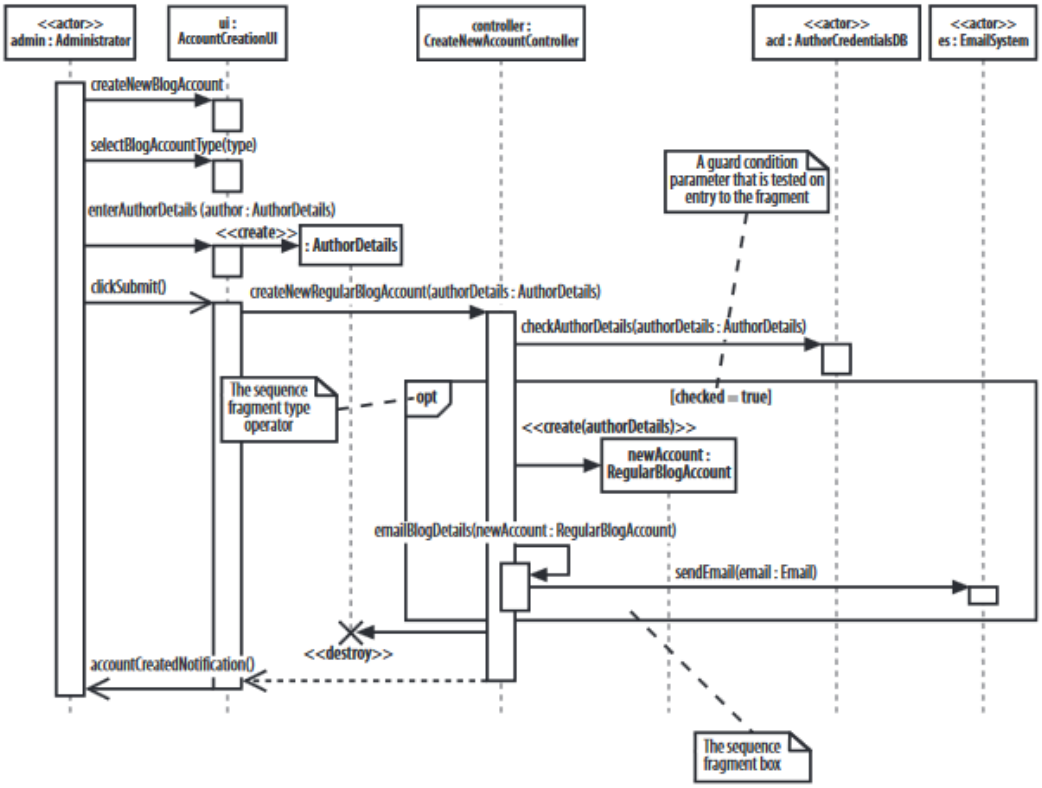


Sumber Gambar : Learning UML 2.0 [10]

Gambar 2. 20 Contoh Class Diagram

d. Sequence Diagram

Sequence Diagram menggambarkan kolaborasi dinamis antara sejumlah objek dan untuk menunjukan rangkaian pesan yang dikirim antara objek juga interaksi. Sesuatu yang terjadi pada titik tertentu dalam eksekusi system.



Sumber Gambar : Learning UML 2.0 [10]

Gambar 2. 21 Contoh Sequence Diagram

2.2.13 Basis Data

Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat [11].

Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apa pun bentuknya, entah berupa *file* teks ataupun Database Management *System* (DBMS). Kebutuhan basis data dalam sistem informasi meliputi :

1. Masukkan, menyimpan, dan mengambil data
2. Membuat laporan berdasarkan data yang telah disimpan

2.2.14 Database MySQL

MySQL adalah sebuah perangkat lunak system manajemen basis data SQL (DBMS) yang multithread, dan multi-user. MySQL adalah implementasi dari system manajemen basisdata relasional (RDBMS). MySQL dibuah oleh TcX dan telah dipercaya mengelola system dengan 40 buah database berisi 10.000 tabel dan 500 di antaranya memiliki 7 juta baris. MySQL AB merupakan perusahaan komersial Swedia yang mensponsori dan yang memiliki MySQL. Pendiri

MySQL AB adalah dua orang Swedia yang bernama David Axmark, Allan Larsson dan satu orang Finlandia bernama Michael "Monty". Setiap pengguna MySQL dapat menggunakannya secara bebas yang didistribusikan gratis dibawah lisensi GPL (General Public License) namun tidak boleh menjadikan produk turunan yang bersifat komersial.

Pada saat ini MySQL merupakan database *server* yang sangat terkenal di dunia, semua itu tak lain karena bahasa dasar yang digunakan untuk mengakses database yaitu SQL. SQL (Structured Query Language) pertama kali diterapkan pada sebuah proyek riset pada laboratorium riset San Jose, IBM yang bernama *system* R. Kemudian SQL juga dikembangan oleh Oracle, Informix dan Sybase. Dengan menggunakan SQL, proses pengaksesan database lebih user-friendly dibandingan dengan yang lain, misalnya dBase atau Clipper karena mereka masih menggunakan perintah-perintah pemrograman murni [11].

SQL dapat digunakan secara berdiri sendiri maupun di lekatkan pada bahasa pemograman seperti C, dan Delphi.

Elemen dari SQL yang paling dasar antara lain pernyataan, nama, tipe data, ekspresi, konstanta dan fungsi bawaan.

Dalam MYSQL ada beberapa operasi dasar yang lebih dikenal dengan operasi CRUD yaitu:

1. *Create*

   Create adalah operasi penambahan data baru kedalam table. Terdapat 2 *Query* untuk menambah data, yang pertama adalah penambahan yang tidak menspesifikasikan nama kolom yang akan ditambahkan dan hanya memberikan isi dari tabelnya. Querynya adalah sebagai berikut:

INSERT INTO table_name VALUES (value1,value2,value3,...);

Untuk yang kedua adalah menginputkan data dengan mencantumkan nama kolom yang akan diisikan dengan isi di dalamnya. Querynya 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. Querynya adalah sebagai berikut:

   SELECT column_name,column_name
   FROM table_name;

3. *Update*

   Update adalah operasi untuk mengubah data yang ada di dalam database. Querynya 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 table. Querynya adalah sebagai berikut:

   DELETE FROM table_name
   WHERE some_column=some_value;

Sistem database menyediakan bahasa untuk mendefinisikan database Data Definition Language (DDL) serta bahasa untuk memanipulasi database Data Manipulation Language (DML) untuk melakukan operasi-operasi tertentu pada database dan juga Data Control Language (DCL).

a. Data Definition Language (DDL)

   Kumpulan perintah SQL yang digunakan untuk membangun database, bisa juga memiliki arti lain perintah untuk membuat struktur dasar database dan tabel. Perintah-perintah yang termasuk DDL adalah statment connect, create, show, describe, drop, rename, dan alter.

b. Data Manipulation Language (DML)

Kumpulan statment yang digunakan sebagai perintah untuk mengelola data dalam sebuah tabel. Perintah DML digunakan setelah menggunakan DDL anatar lain seperti *statment insert, update, delete*, dan *select*.

c. Data Control Language (DCL)

Perintah yang digunakan untuk keperluan keamanan (security) database dengan membuat hak akses tertentu bagi setiap user. Statment pada DCL antara lain grant, revoke, set, dan lock table.

2.2.15 PHP

PHP merupakan akronim dari PHP: *Hypertext Prepocessor* adalah bahasa pemrograman *script* berbasis *web* yang paling banyak dipakai saat ini. PHP banyak dipakai untuk memrogram 30 situs *web* dinamis, walaupun tidak tertutup kemungkinan digunakan untuk pemakaian lain.

Untuk menjalankan sistem PHP dibutuhkan tiga komponen:

1. *Web server*, karena PHP termasuk bahasa pemrograman *server side*.
2. Program PHP, program yang memproses script PHP.
3. Database *server*, yang berfungsi untuk mengelola database.

Kelebihan PHP dari bahasa pemrogaraman lain adalah:

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaannya.
2. *Web server* yang mendukung PHP dapat ditemukan dimana-mana dari mulai IIS sampai dengan apache, dengan konfigurasi yang relatif mudah.
3. Dalam isi pengembangan lebih mudah, karena banyaknya milis-milis dan *developer* yang siap membantu dalam pengembangan.
4. Dalam isi pemahaman, PHP adalah bahasa *scripting* yang paling mudah karena referensi yang banyak.
5. PHP adalah bahasa *open source* yang dapat digunakan diberbagai mesin *(linux, unix, windows)* dan dapat dijalankan secara runtime melalui *console* serta juga dapat menjalankan perintah-perintah *system* [12].

Contoh script PHP sederhana adalah sebagai berikut:

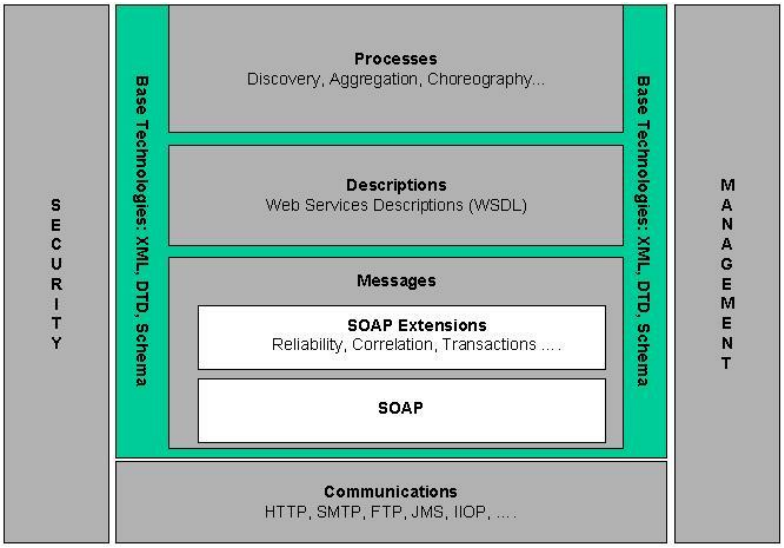
```
<?php
//pembuatan array
$nama = array(
        1=>"Aldot",
        2=>"Jaka",
        3=>"Putri");
//cara akses array
echo $nama[1]; //Aldot
echo "<br />";
echo $nama[2]; //Jaka
echo "<br />";
echo $nama[3]; //Putri
?>
```

2.2.16 *Web Service*

Web service adalah sebuah entitas komputasi yang dapat diakses melalui jaringan internet maupun intranet dengan standar protokol tertentu dalam platform dan antarmuka bahasa pemograman yang independen. Tujuan pengembangannya adalah untuk menjembatani komunikasi antar program, sehingga aplikasi yang satu dan aplikasi yang lain yang terdapat pada suatu jaringan yang sama atau pada jaringan berbeda dapat saling berkomunikasi asalkan menggunakan standart protokol yang ditetapkan oleh *web service*. Hal ini bisa terjadi, karena standart protokol itu tidaklah terikat pada suatu platform atau bahasa pemograman. Protokol itu sendiri dibangun oleh *Extensible Markup Language* (XML) yang memang kenyataannya telah didukung oleh banyak paltform, bahasa pemograman, dan oleh *developer* di seluruh dunia.

Arsitektur *web service* dibangun oleh beberapa layer dan teknologi yang saling berhubungan. Banyak cara untuk memvisualisaskan *service*, sama

banyaknya dengan cara untuk membangun dan menggunakan web service gambar berikut hanyalah salah satu cara untuk menggambarkan arsitekturnya.



Sumber Gambar : Purba dan Siregar [13]

Gambar 2. 22 Arsitektur Web Service

Web service disusun oleh tiga komponen standart, yaitu :

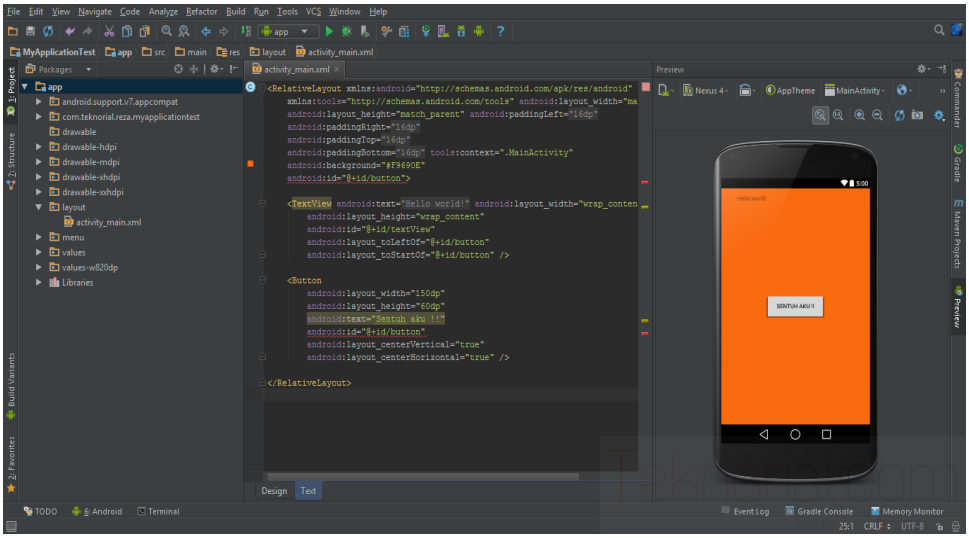
1. *Simple Object Access Protocol* (SOAP), yaitu protokol yang bertanggungjawab dalam pertukuran informasi dalam lingkungan jaringan terdistribusi.
2. *Web service definition Language* (WDSL), dokumen standart yang dituliskan pada format XML, dan mendifinikasan kehadiran web service dalam suatu jaringan.
3. *Universal description, Discovery and Inttegration* (UDDI), yaitu suatu lokasi direktori yang beriskan *service* (layanan) dan bersifat bebas platform (platform independent), dituliskan berbasis XML dan dapat diakses oleh entittas yang berada di dalam dan luar jaringan.

Adanya standart tersebut membuat web service mudah di akses melalui berbagai antarmuka dan juga memberi peluang dimungkinkannya berbagai sistem yang dibangun pada platform yang berbeda dan bahasa yang berbeda untuk berkolaborasi pada dalam suatu pekerjaan [13].

2.2.17 Android Studio

Android studio adalah lingkungan pengembangan terpadu (*IDE)* resmi untuk pengembangan platform android, hal itu disampaikan pada tanggal 16 mei 2013 Google I/O, android studio sudah tersedia secara bebas dibawah lisensi Apache 2.0.

Android studio pada awalnya tahap preview versi 0.1 yang dipakai pada tanggal 1 mei 2013 dan memasuki tahap beta pada bulan juni 2014 dan mulai stabil dirilis pada Desember 2014 dengan versi 1.0, berdasarkan jetBrains 'IDEA Intellij Software, android studio dirancang khusus untuk pengembangan android yang tersedia untuk Windows,Mac OS X dan Linux sebagai pengganti Eclipse.



Gambar 2. 23 Layout Android Studio

1. Struktur Android Studio

   Rincian dari struktur Android Studio adalah sebagai berikut:

   1. App : berfungsi untuk menyimpan file file pendukung aplikasi yang akan diolah.
      1. *Manifest*

         tempat dimana konfigurasi konfigurasi pendukung project serta izin (permision) untuk mengakses hardware.

2. Java

   tempat berdiamnya code/scripts java untuk menjalankan aplikasi yang akan dibuat

3. *Res* :

   a. Drawable : tempat untuk menyimpan gambar atau icon pendukung aplikasi.

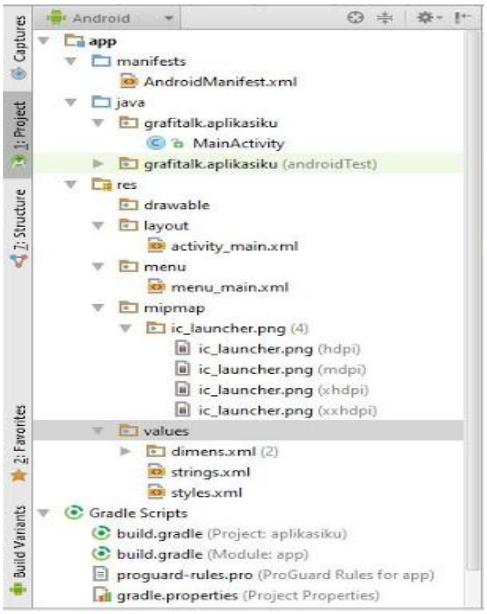
   b. *Layout* : tempat berkumpulnya design tampilan yang akan tampak pada tampilan aplikasi anda.

   c. *Menu* : berisi code berupa file xml yang didalamnya berisi array tentang menu yang akan tampil dalam project anda, misalkan file, new,open,close,dsb.

   d. *Mipmap* : tempat berkumpulnya icon launcher (icon utama aplikasi) yang ukurannya menyesuaikan device yang digunakan.

   e. *Values* : untuk menyimpan file .xml yang menggambarkan sumber dari berbagai jenis. Sebagai contoh folder berisi file strings.xml dengan definisi sumber string. Seperti yang akan anda lihat dari menambahkan bagian color , folder tata letak juga dapat berisi, misalnya, deskripsi warna. Di dalam file dimens.xml, terdapat string-string untuk konfigurasi dimensi aplikasi. Tidak jauh berbeda dengan dimens.xml maupun strings.xml, pada file styles.xml terdapat deklarasi variable untuk theme.

4. *Gradle scripts* : berfungsi sebagai configurasi aplikasi anda dengan device, serta menghubungkan library pendukung aplikasi yang anda buat.

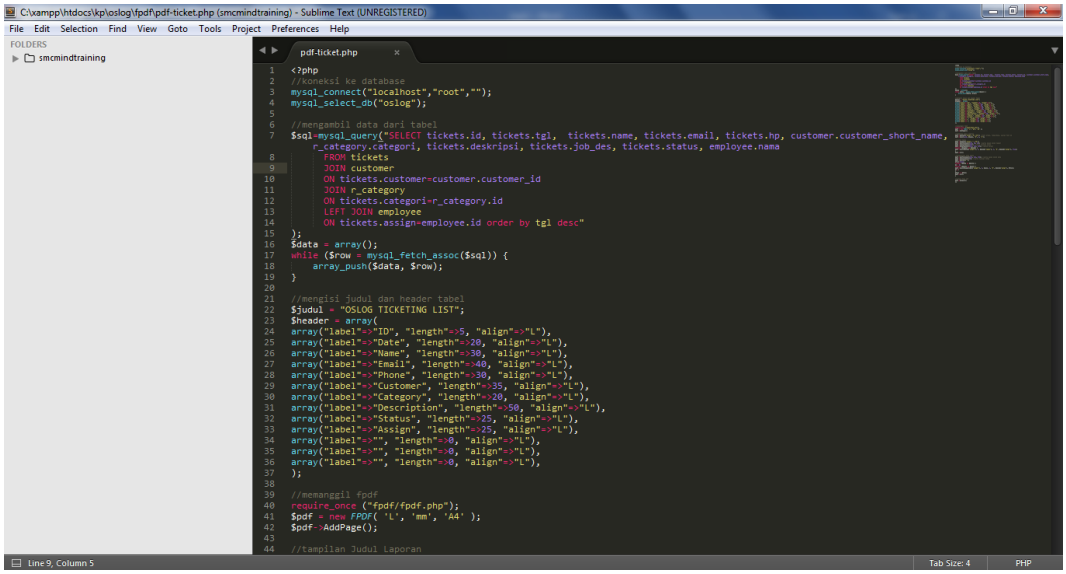


Gambar 2. 24 Struktur Android Studio

2.2.18 *Sublime Text*

Sublime Text adalah aplikasi editor untuk kode dan teks yang dapat berjalan diberbagai platform *operating system* dengan menggunakan teknologi *Pyhton* API. Terciptanya aplikasi ini terinspirasi dari aplikasi *Vim*, Aplikasi ini sangatlah fleksibel dan *powerfull*. Fungsionalitas dari aplikasi ini dapat dikembangkan dengan menggunakan *sublime-packages*. Sublime Text bukanlah aplikasi *open source* dan juga aplikasi yang dapat digunakan dan didapatkan secara gratis, akan tetapi beberapa fitur pengembangan fungsionalitas (*packages*) dari aplikasi ini merupakan hasil dari temuan dan mendapat dukungan penuh dari komunitas serta memiliki linsensi aplikasi gratis.

Sublime Text mendukung berbagai bahasa pemrograman dan mampu menyajikan fitur *syntax highlight* hampir di semua bahasa pemrogramman yang didukung ataupun dikembangkan oleh komunitas seperti; C, C++, C#, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL, Textile and XML. Biasanya bagi bahasa pemrograman yang didukung ataupun belum terdukung secara *default* dapat lebih dimaksimalkan atau didukung dengan menggunakan *add-ons* yang bisa didownload sesuai kebutuhan *user*.



Gambar 2.24 Tampilan Sublime Text

Berikut beberapa fitur yang diunggulkan dari aplikasi Sublime Text:

a. *Goto Anything*

Fitur yang sangat membantu dalam membuka file ataupun menjelajahi isi dari file hanya dengan beberapa *keystrokes*.

b. *Multiple Selections*

Fitur ini memungkinkan *user* untuk mengubah secara interaktif banyak baris sekaligus, mengubah nama variabel dengan mudah, dan memanipulasi file lebih cepat dari sebelumnya.

c. *Command Pallete*

Dengan hanya beberapa keystorkes, user dapat dengan cepat mencari fungsi yang diinginkan, tanpa harus menavigasi melalu menu.

d. *Distraction Free Mode*

Bila user memerlukan fokus penuh pada aplikasi ini, fitur ini dapat membantu *user* dengan memberikan tampilan layar penuh.

e. *Split Editing*

Dapatkan hasil yang maksimal dari monitor layar lebar dengan dukungan *editing* perpecahan. Mengedit sisi file dengan sisi, atau mengedit dua lokasi

di satu file. Anda dapat mengedit dengan banyak baris dan kolom yang *user* inginkan.

f. *Instant Project Switch*

Menangkap semua file yang dimasukkan kedalam *project* pada aplikasi ini. Terintegrasi dengan fitur *Goto Anything* untuk menjelajahi semua file yang ada ataupun untuk beralih ke file dalam *project* lainnya dengan cepat.

g. *Plugin API*

Dilengkapi dengan plugin API berbasis Phyton sehingga membuat aplikasi ini sangat tangguh.

h. *Customize Anything*

Aplikasi ini memberikan *user* fleksibilitas dalam hal pengaturan fungsional dalam aplkasi ini.

i. *Cross Platform*

Aplikasi ini dapat berjalan hampir disemua *operating system* modern seperti Windows, OS X, dan Linux *based operating system*.

2.2.19 Intranet

Intranet adalah konsep LAN yang mengadopsi teknologi Internet dan mulai diperkenalkan pada akhir tahun 1995. Dengan kata lain intranet adalah LAN yang menggunakan standar komunikasi dan segala fasilitas internet. Intranet dapat diibaratkan berinternet dalam lingkungan local dan pada umumnya terkoneksi ke internet sehingga memungkinkan pertukaran informasi dan data dengan jaringan intranet lainnya melalui *backbone* Internet.

Intranet adalah sebuah jaringan privat (*private network*) yang menggunakan protokol-protokol Internet (TCP/IP) untuk membagi informasi rahasia perusahaanatau operasi dalam perusahaan tersebut kepada karyawannya. Kadang-kadang, istilah intranet hanya merujuk kepada layanan yang terlihat, yakni situs web internal perusahaan. Untuk membangun sebuah intranet maka sebuah jaringan harus memiliki beberapa komponen yang membangun Internet yaitu protokol Internet (Protokol TCP/IP, alamat IP, dan protokol lainnya), *client* dan juga *server*. Protokol HTTP dan beberapa protokol Internet lainnya (FTP, POP3, atau SMTP)

umumnya merupakan komponen protokol yang sering digunakan. Umumnya, sebuah intranet dapat dipahami sebagai sebuah versi pribadi dari jaringan internet atau sebagai sebuah versi dari Internet yang dimiliki oleh sebuah organisasi. Adapun manfaat dari Intranet adalah :

1. Membuat perusahaan atau institusi menjadi lebih hemat dan efisien, karena sistem informasi berubah dari cara konvensional ke cara yang memanfaatkan web dan *database*.
2. Perolehan informasi yang cepat dan detail serta sistem manajemen yang canggih akan membuat perusahaan atau intsitusi tersebut memimpin dalam bidangnya.
3. Pertukaran data antar lokasi secara *point to point*, seperti *video conference*, transfer data atau file dan telepon intranet [14].