BAB II
LANDASAN TEORI

2.1. Profil Dinas Kependudukan dan Catatan Sipil Kabupaten Cianjur

Dinas Kependudukan dan Catatan Sipil Kabupaten Cianjur adalah salah satu dinas pemerintahan yang mempunyai tugas dalam mengkoordinasi kependudukan dan pencatatan sipil di Kabupaten Cianjur. Dinas ini mengelola, membina, mengawasi, dan mengendalikan data dan catatan kependudukan. Dinas Kependudukan dan Catatan Sipil juga bertugas dalam menerbitkan dokumen-dokumen mengenai kependudukan dan catatann sipil, yaitu, kartu keluarga dan akta-akta sipil.

2.1.1. Sejarah

Pembangunan kependudukan merupakan isu strategis dan bersifat lintas sektor, sehingga pengintegrasian berbagai aspek kependudukan ke dalam perencanaan pembangunan perlu diwujudkan. Upaya-upaya mewujudkan keterkaitan perkembangan kependudukan, dengan berbagai kebijakan pembangunan menjadi prioritas penting agar pengelolaan perkembangan kependudukan dapat mewujudkan keseimbangan yang serasi antara kuantitas, kualitas dan mobilitas penduduk.

Pemerintah Kabupaten Cianjur sudah menyelenggarakan pendaftaran penduduk dan pencatatan sipil dengan menggunakan Sistem Administrasi Kependudukan (SAK) yang didukung dengan Sistem Informasi Administrasi Kependudukan (SIAK). Sistem ini sudah mulai dilaksanakan sejak tahun 2006 dan sudah menghasilkan basis data kependudukan untuk Kabupaten Cianjur. Basis data kependudukan ini dapat dimanfaatkan untuk memberikan gambaran bagaimana kondisi dan karakteristik penduduk Kabupaten Cianjur dan dapat menjadi alternatif untuk memenuhi kebutuhan data kependudukan bagi Pemerintah Kabupaten Cianjur. Selama ini pemerintah Kabupaten Cianjur hanya menggunakan data yang dihasilkan dari Badan Pusat Statistik (BPS) maupun pendataan yang dilakukan oleh instansi terkait lainnya. Kelemahan data statistik yang disajikan adalah bahwa data

tersebut hanya dikumpulkan dalam jangka waktu tertentu (10 tahunan), sehingga untuk memperoleh data tahunan digunakan data proyeksi atau data perkiraan yang dihitung dari dua atau tiga titik tahun pendataan penduduk.

Berkenaan dengan penyajian data dan informasi perkembangan kependudukan terutama untuk perencanaan pembangunan manusia, baik itu pembangunan ekonomi, sosial, politik, lingkungan, dan lain-lain yang terkait dengan peningkatan kesejahteraan manusia, maka data dan informasi perlu menggunakan data yang valid dan dapat dipercaya baik dari sisi jumlah maupun kualitas data dan dikemas secara baik, sederhana, informatif dan tepat waktu dalam bentuk Profil Perkembangan Kependudukan yang disajikan secara berkelanjutan. Profil perkembangan kependudukan tersebut diharapkan dapat memberikan gambaran kondisi kependudukan di Kabupaten Cianjur serta prediksi prospek kependudukan dimasa yang akan datang.

2.1.2. Visi dan Misi

Visi

Menunjang terwujudnya masyarakat Cianjur lebih sejahtera dan berakhlakuk karimah melalui peningkatan administrasi Kependudukan dan catatan sipil dalam memberikan pelayanan kepada masyarakat.

Misi

1. Meningkatnya pelayanan administrasi Kependudukan dan Catatan Sipil di Kabupaten Cianjur.
2. Meningkatnya tertib administrasi Kependudukan dan Catatan Sipil di Kabupaten Cianjur.
3. Meningkatnya akurasi data Kependudukan dan Catatan Sipil di Kabupaten Cianjur.
4. Meningkatnya pelayanan informasi Kependudukan dan Catatan Sipil.

2.1.3. Logo

Berikut adalah logo dari instansi kedinasan di Kabupaten Cianjur yang dapat dilihat pada Gambar 2.1:



Sumber gambar: http://*jabarprov.go.id*

Gambar 2.1 Logo instansi kedinasan Kabupaten Cianjur

2.2. Kartu Pencari Kerja

Kartu Pencari Kerja atau Kartu Kuning adalah kartu yang dikeluarkan oleh Dinas Ketenagakerjaan dan Transmigrasi kepada masyarakat yang sedang mencari pekerjaan dan berstatus sebagai tuna karya atau pengangguran. Masyarakat yang memiliki kartu pencari kerja, maka masyarakat tersebut terdaftar untuk mendapatkan rekomendasi ke perusahaan dan pelatihan-pelatihan dari dinas setempat untuk mendapatkan pekerjaan. Kartu ini berwarna kuning dan berisi data diri, keterampilan yang dia miliki, serta tanda tangan pejabat dan cap dari dinas setempat. Setiap pemilik Kartu Pencari Kerja harus melaporkan balik ke Dinas Ketenagakerjaan dan Transmigrasi apabila dia sudah mendapat pekerjaan. Apabila belum, maka pemilik kartu harus melaporkan ke kantor Dinas Ketenagakerjaan dan Transmigrasi sebanyak satu kali dalam enam bulan.

Berikut adalah syarat-syarat dalam pembuatan Kartu Pencari Kerja:

1. Membawa fotokopi KTP
2. Membawa fotokopi Akta Kelahiran
3. Membawa fotokopi Kartu Keluarga
4. Membawa fotokopi ijazah pendidikan terakhir

5. Membawa pasfoto sebanyak dua buah, masing-masing wilayah memiliki syarat ukuran yang berbeda-beda

Di beberapa wilayah mewajibkan pula untuk membawa Surat Keterangan Cakap Kepolisian yang dapat dibuat di kantor polisi (polsek, polres, atau polda) setempat. Dan berikut adalah syarat-syarat dalam pembuatan Surat Keterangan Cakap Kepolisian:

1. Membawa fotokopi KTP
2. Membawa fotokopi Akta Kelahiran
3. Membawa fotokopi Kartu Keluarga
4. Membawa pasfoto ukuran 4x6 sebanyak 10 buah
5. Membawa surat pengantar dari RT, Kecamatan, dan Polsek

Untuk pembuatan kartu tersebut sendiri hanya membutuhkan waktu beberapa jam. Cukup datang ke kantor Dinas Ketenagakerjaan dan Transmigrasi dan menyerahkan seluruh berkas yang dibawa. Berikut adalah contoh Kartu Pencari Kerja yang dapat dilihat pada Gambar 2.2:


PEMERINTAH KABUPATEN BANYUMAS
DINAS SOSIAL, TENAGA KERJA DAN TRANSMIGRASI
Jalan Pemuda Nomor 24 Telp. (0281) 636198 , Fax. (0281) 633047 Purwokerto

KARTU TANDA PENCARI KERJA

NO. PENCAKER : 3374 190609 - 473
NO. PENDUDUK : 3302032304860001

Ketentuan :
1. Kartu ini berlaku untuk melamar pekerjaan.
2. Bila ada perubahan data/keterangan lainnya atau telah mendapat pekerjaan harap segera melapor.
3. Apabila pencari kerja yang bersangkutan telah diterima bekerja maka instansi/perusahaan yang menerima agar mengembalikan AK/I ini ke-pada Dinas.
4. Kartu ini berlaku selama 2 tahun dengan keharusan melapor setiap 6 bulan sekali terhitung sejak tanggal pendaftaran.
5. Apabila dikemudian hari terbukti memberikan keterangan yang tidak benar dan atau memiliki lebih dari satu kartu AK/I maka akan ditindak sesuai dengan peraturan perundang-undangan yang berlaku

Laporan	Tgl-Bl-Th	Ttd. Petugas (cantumkan NIP)
Pertama		
Kedua		
Ketiga		

Tanda Tangan
Pencaker

Diterima di .....
Tgl. M.T .....

Sumber gambar: http://informasibumncpns.com

Gambar 2.2 Contoh Kartu Pencari Kerja

2.3. Aplikasi

Aplikasi dalam konteks ini adalah aplikasi perangkat lunak adalah sebuah program atau perangkat lunak yang ditujukan untuk tujuan tertentu yang dapat membantu pengguna dalam menyelesaikan atau mempercepat penyelesaian masalah dengan memanfaatkan kemampuan komputer. Menurut buku Pengantar Basis Data oleh Sri Widianti (2000), aplikasi adalah sebuah perangkat lunak yang menjadi *front end* dalam sebuah sistem yang digunakan untuk mengolah data menjadi suatu informasi yang berguna untuk orang-orang dan sistem yang bersangkutan [4]. Contoh aplikasi dapat membantu mempercepat sebuah pekerjaan yang biasa dilakukan manusia adalah aplikasi pengolah kata, sebelum munculnya sebuah aplikasi perangkat lunak pengolah kata, manusia menggunakan sebuah perangkat mesin ketik yang di mana apabila melakukan sebuah kesalahan maka harus mengulang proses pengetikan, dengan aplikasi pengolah kata hal tersebut dengan mudah cukup dengan menghapus bagian yang perlu diubah. Contoh lain adalah aplikasi pembuatan tagihan berbasis *web*, aplikasi ini berguna untuk membuat tagihan secara otomatis setiap bulan yang dikirimkan melalui email, tanpa perlu seseorang membuatkan tagihan kepada sekumpulan pelanggan yang dikirim melalui surat.

2.3.1. Aplikasi Berdasarkan Fungsi

Aplikasi perangkat lunak dapat dibagi menjadi beberapa macam berdasarkan fungsinya, yaitu:

1. *Word Processing*

Word Processing adalah aplikasi perangkat lunak yang berfungsi untuk melakukan pengolahan kata seperti menulis dokumen atau surat-menyurat.

Contoh: Microsoft Word, Google Docs

2. *Desktop Publishing*

Desktop Publishing adalah aplikasi perangkat lunak yang berfungsi untuk membantu dalam penulisan naskah.

Conth: Final Draft, Celtx, Trelby

3. Program *Speadsheet*

Program Speadsheet adalah aplikasi perangkat lunak yang berfungsi untuk melakukan perhitungan matematika dengan layout baris dan kolom.

Contoh: Microsoft Excel, Google Sheets

4. Database Management System

Database Management System adalah aplikasi perangkat lunak yang berfungsi untuk melakukan pengolah basis data. Pengolahan ini dapat berupa penambahan, pengubahan, pembacaan, dan penghapusan data.

Contoh: Microsoft Access, phpMyAdmin

5. Graphics

Graphics adalah aplikasi perangkat lunak yang berfungsi untuk melakukan pengolahan sebuah citra.

Contoh: Adobe Photoshop, Paint, Corel Draw

6. Program Akuntasi

Aplikasi perangkat lunak yang berfungsi untuk pengolahan keuangan dan akuntansi.

Contoh: Zoho Books, ZipBooks

7. Program Statistik

Aplikasi perangkat lunak yang berfungsi untuk pengolahan data yang mampu menghasilkan informasi yang berguna.

Contoh: SPSS, Stata, GraphPad

8. Communication

Aplikasi perangkat lunak yang berfungsi untuk membantu dalam hubungan komunikasi jarak jauh antar dua atau lebih orang dalam waktu yang bersamaan atau berbeda.

Contoh: Skype, WhatsApp, Gmail

9. Multi media

Aplikasi multi media adalah aplikasi perangkat lunak yang berfungsi untuk melakukan pekerjaan multimedia yang mencakup beberapa aspek seperti suara, video, teks, dan gambar.

Contoh: Adobe Premiere Pro, Blender

10. Game

Aplikasi *game* adalah aplikasi perangkat lunak yang berfungsi untuk melakukan kegiatan permainan yang menghibur dengan media video.

Contoh: Pinball, Solitaire

11. Anti virus

    Aplikasi anti virus adalah aplikasi yang berfungsi untuk mendeteksi virus yang terdapat pada komputer dan membantu membersihkannya.

    Contoh: AVG, McAfee, Smadav

2.4. Web service

Web service adalah salah satu komponen dari sistem terdistribusi yang terdapat sekumpulan program untuk memproses masukan dan mengembalikan pesan melalui perantara komunikasi *World Wide Web* (WWW) HTTP [5]. Menurut W3C—lembaga internasional yang bergerak dalam membangun standarisasi *web—web service* adalah sistem perangkat lunak yang didesain dapat dioperasikan mesin ke mesin melalui jaringan. *Web service* adalah seperangkat standar untuk membentuk *service-oriented architecture* (SOA). Arsitektur ini memodelkan interaksi antara tiga peran: penyedia layanan, konsumen layanan, dan pendaftar layanan [6]. Penggunaan *web service* adalah untuk mengkoneksikan secara dinamis dari perangkat-perangkat yang diketahui maupun tidak diketahui di dalam satu jaringan [7]. *Web service* memungkinkan sarana operasi antar perangkat lunak menjadi standar pada berbagai *platform* yang berbeda. Pesan yang diterima dari proses program yang ada pada *service* dapat berupa *XML* atau *JSON*. Adanya *web service* memungkinkan komunikasi antar klien pengguna *web service* (aplikasi) berkomunikasi dengan bertukar informasi berupa teks melalui jaringan yang tersedia pada alamat URL yang tersimpan pada sebuah *server* yang selalu aktif.

2.4.1. Keuntungan *Web service*

Terdapat keuntungan-keuntungan yang dapat dirasakan apabila membangun sebuah sistem menggunakan metode *web service*, yaitu:

1. Mudah

Apabila sistem memiliki lebih dari satu aplikasi, bahkan masing-masing aplikasi menggunakan bahasa pemrograman yang berbeda, maka pembuat aplikasi tidak perlu membuat masing-masing program atau layanan untuk menghubungkan atau berkomunikasi dengan *server* maupun *database*. Klien dapat mengakses sumber daya yang disediakan oleh *web service* melalui URL yang terstandar.

2. Aman

   Dengan terpusatnya program atau sumber daya yang terdapat di *server*, maka pemrogram *web service* dapat mengatur keamanan dan tingkat hak akses dari permintaan yang masuk. Misal pada sebuah sistem terdapat sebuah basis data, maka yang berhak berhubungan langsung dengan basis data tersebut hanya *web service* yang dibuat, sehingga aplikasi-aplikasi klien tidak berhubungan langsung dengan basis data. *Web service* bisa kita sebut sebagai perantara atau makelar dari sebuah pertukaran data.

3. Terstandar

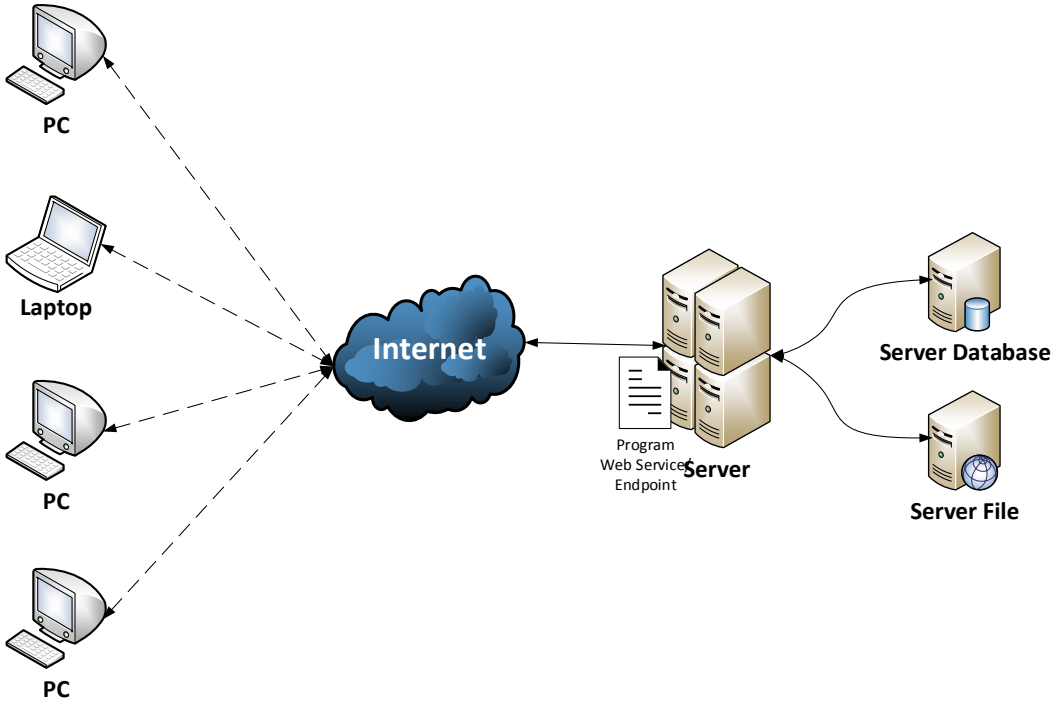
   Web service memungkinkan untuk melakukan pertukaran data di *server* dari beberapa klien dengan bahasa pemrograman yang berbeda. Misal, sebuah aplikasi di *website* menggunakan bahasa pemrograman PHP dan aplikasi lain menggunakan bahasa pemrograman Python dapat menggunakan sebuah *web service* yang sama yang ditulis dalam bahasa pemrograman JavaScript. Pemrogram hanya perlu membuat satu layanan *server* dengan menggunakan bahasa pemrograman JavaScript. Pertukaran informasi menggunakan format *file* yang terstandar yaitu *XML* atau *JSON*.

4. *Low-cost*

   Pertukaran informasi layanan *web service* menggunakan format *file* teks, memungkinkan untuk bertransaksi antar perangkat lebih cepat menggunakan sumber daya yang ringan.

2.4.2. Cara Kerja *Web service*

Berikut adalah cara kerja dari sebuah *web service* yang dapat dilihat pada Gambar 2.3:



Gambar 2.3 Arsitektur *Web service*

1. Klien (PC dan laptop) menggunakan aplikasi *brower* terhubung melalui jaringan *internet* mengakses sebuah program di *server* melalui endpoint (URL).
2. *Server* mengeksekusi perintah dari potongan program yang dimaksud dari *endpoint* yang dituju.
3. *Server* mengakses *server database* dan *server file* apabila perlu.
4. *Server* mengirim permintaan dari klien melalui jaringan internet berupa format *file* teks, dapat berupa *XML* atau *JSON*.

2.5. Access token

Access token merupakan sebuah *string unique* dan acak yang berisi angka dan huruf. *Access token* adalah sebuah kunci yang dibutuhkan oleh pemohon layanan

yang harus digunakan untuk mengidentifikasi dan memverifikasi apakah pemohon memiliki hak dalam mengakses *web service* [8]. Masing-masing pemilik akun pada layanan *web service* memiliki sebuah *access token*.

Access token ini berfungsi untuk menambah tingkat keamanan pada *web service*, sehingga *web service* hanya dapat diakses oleh pihak yang berwenang saja. *Access token* tersebut dapat digenerasi ulang apabila pemilik token hendak memperbaharui token yang dirasa sudah terlalu lama. *Access token* baiknya selalu digenerasi ulang tiap periode yang berkala untuk mencegah terjadinya penggunaan *access token* oleh pihak yang tidak berwenang.

Berikut adalah beberapa contoh dari *string access token*:

```
String masukan: email@domain.com1492035660

SHA-1
ffd50176c3b72e2b8b78598dcfab50759929af64

MD5
9730f528b042c8d624f8ac5831309871
```

a. email@domain.com adalah email dari pengguna layanan *web service*.

b. 1492035660 adalah waktu *timestamp* waktu generasi kunci baru.

Tidak ada aturan dalam format bentukan *string access token*. Pada kasus ini peneliti menggunakan fungsi hash dari *string* kombinasi email pengguna dengan waktu pembentukan *access token*. Misal *access token* digenerasi menggunakan metode acak, hal tersebut diperbolehkan.

2.5.1. Penggunaan *Access token*

Dalam penggunaan *access token*, terdapat dua metode yang bisa digunakan, yaitu [9]:

1. Menggunakan *access token* di dalam *header*

```
$ curl -H "Authorization: Bearer ACCESS_TOKEN"
http://domainapi.com/endpoint
```

2. Menggunakan *access token* di dalam parameter URL

```
$ curl http://domainapi.com/endpoint?access_token=ACCESS_TOKEN
```

Contoh di atas adalah penggunaan *access token* menggunakan perintah curl.

2.6. REST

REST merupakan gaya arsitektur dalam mendesain sebuah *web service* di mana desain REST memiliki resource yang dapat diakses melalui sebuah alamat HTTP URL yang *unique* [10] [11]. Masing-masing alamat URL mengacu kepada kumpulan program yang akan dieksekusi dan akan mengembalikan pesan kepada pengirim perintah. Perantara/gerbang antara klien dan *server* untuk dapat mengirim perintah dan mendapatkan pesan ini dinamakan API (*Application Programming Interface*).

Peneliti menggunakan gaya arsitektur *REST* ketimbang *SOAP* karena beberapa pertimbangan yang di mana REST lebih unggul. Beberapa keunggulan *REST* daripada *SOAP* adalah sebagai berikut [12]:

1. *REST* merupakan teknologi yang lebih baru
2. *REST* lebih ringan daripada *SOAP*
3. *REST* menggunakan lebih sedikit data, *SOAP* kurang lebih menggunakan sepuluh kali data lebih banyak ketimbang *REST*
4. *REST* lebih mudah dibangun

REST mengirimkan perintah yang akan dikerjakan oleh *server* menggunakan metode-metode HTTP *request method* yang disebut *verb*. Terdapat delapan HTTP *request method*, yaitu *GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE*, dan *CONNECT* [13] [14] [15]. Dalam penggunaan *API REST* hanya menggunakan empat dari metode-metode tersebut, yaitu: *GET, POST, PUT,* dan *DELETE*. Berikut adalah penjelasan dari masing-masing metode tersebut [16]:

a. ***GET*:** Cocok digunakan untuk mengambil data dari basis data
b. ***POST*:** Cocok digunakan untuk menambah data ke basis data
c. ***PUT*:** Cocok digunakan untuk mengubah data di basis data
d. ***DELETE*:** Cocok digunakan untuk menghapus data di basis data

Pesan yang diterima dari *server* berupa kode HTTP berhasil atau gagal di dalam *header* dan isi pesan hasil pengolahan program itu sendiri. Berikut adalah kode-kode HTTP yang sering digunakan dalam penggunakan API REST [17] [18]:

a. 200 *OK*

Perintah yang dikirim ke *server* benar dan berhasil dijalankan.

b. 400 *Bad Request*

Perintah yang dikirim ke *server* berisi isian yang salah.

c. 401 *Unauthorized*

Pengirim perintah mengirimkan kode kunci yang salah.

d. 403 *Forbidden*

Pengirim perintah tidak memiliki hak akses ke dalam resource yang dituju.

e. 404 *Not Found*

Resource yang dituju tidak ditemukan dalam *server*.

f. 429 *Too Many Requests*

Pengirim perintah mengakses mencapai/melebihi dari limit yang telah ditentukan dari batas waktu tertentu.

g. 500 *Internal Server Error*

Server atau potongan program dalam resource mengalami kesalahan.

2.7. JSON

JSON adalah sebuah format pesan balikan yang berukuran kecil yang mudah dibaca dan ditulis oleh manusia, dan juga mudah ditulis dan diurai oleh mesin [19]. *JSON* merupakan salah satu pesan balikan yang dapat digunakan dalam sebuah *REST API*. Format lain yang sering digunakan adalah *XML*.

JSON terbagi dalam dua struktur, yaitu [19]:

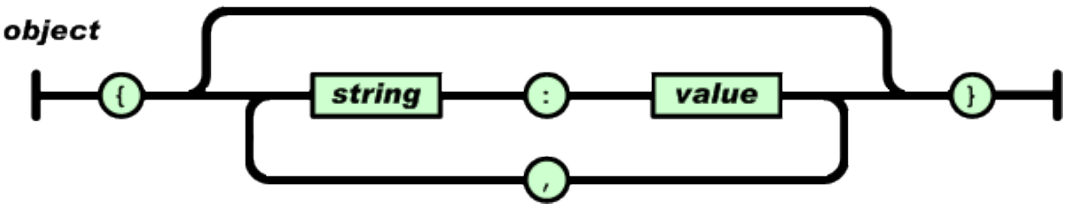
1. Gabungan *name/value*, atau yang biasa dikenal *object* atau *record* dalam bahasa pemrograman.
2. *List value*, atau yang biasa dikenal larik dalam bahasa pemrograman.

2.7.1. Format Bentukan *JSON*

JSON memiliki format-format tipe data berikut yang dapat digunakan [19]: objek, larik, *value*, *string*, dan angka. Berikut adalah penjelasan dari masing-masing bentukan format tersebut:

2.7.1.1. object

Berikut adalah format *JSON object* yang dapat dilihat pada Gambar 2.4:



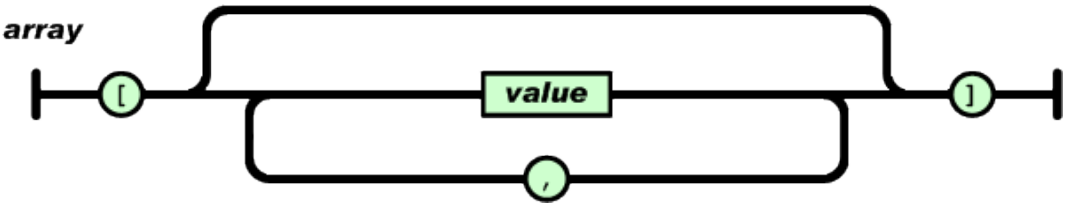
Sumber gambar: http://json.org

Gambar 2.4 Format *JSON Object*

Object adalah sekumpulan dari pengenal dan nilai. Sebuah *object* diawali dengan karakter "{" (kurung kurawal kiri) dan diakhiri dengan karakter "}" (kurung kurawal kanan). Masing-masing pengenal diikuti dengan karakter ":" (titik dua) dan masing-masing nilai diakhiri dengan karakter "," (koma).

2.7.1.2. *array*

Berikut adalah format *JSON array* yang dapat dilihat pada Gambar 2.5:



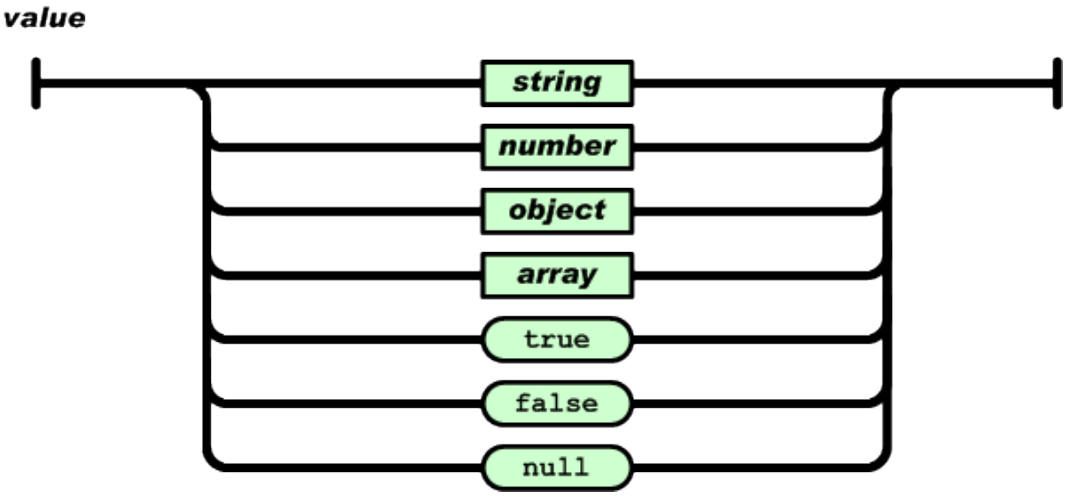
Sumber gambar: http://json.org

Gambar 2.5 Format *JSON Array*

Array adalah sekumpulan nilai. Sebuah *array* diawali dengan karakter "[" (kurung siku kiri) dan diakhiri dengan karakter "]" (kurung siku kanan). Masing-masing nilai dipisahkan dengan karakter "," (koma).

2.7.1.3. *value*

Berikut adalah format *JSON value* yang dapat dilihat pada Gambar 2.6:



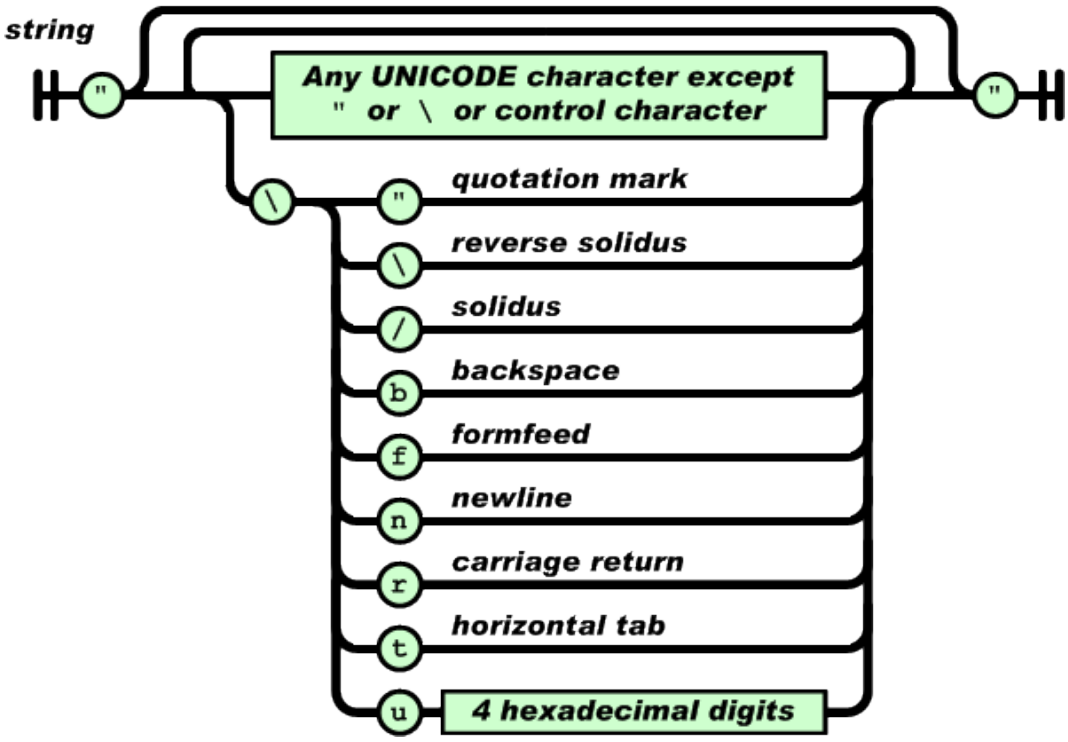
Sumber gambar: http://json.org

Gambar 2.6 Format *JSON value*

Sebuah *value* dapat bernilai *string* diapit dengan karakter petik dua, angka, *boolean* atau null, *object*, atau *array*.

2.7.1.4. *string*

Berikut adalah format *JSON string* yang dapat dilihat pada Gambar 2.7:



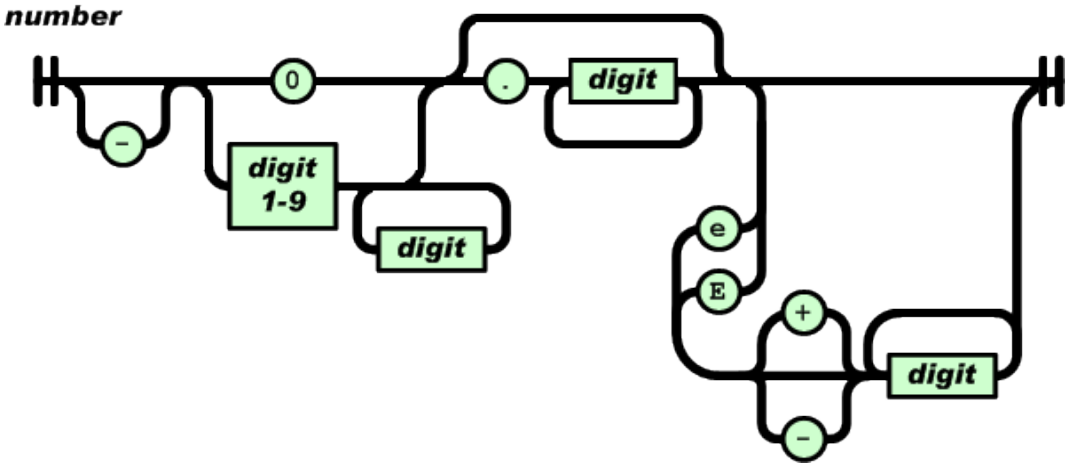
Sumber gambar: http://json.org

Gambar 2.7 Format *JSON String*

String adalah sekumpulan dari karakter kosong atau karakter Unicode yang diapit oleh karakter petik dua atau karakter khusus diawali dengan karakter "\" (garis miring terbalik). Sebuah karakter dianggap sebagai sebuah *string* dengan panjang satu karakter. *String* mirip dengan tipe data *string* pada bahasa pemrograman C atau Java.

2.7.1.5. *number*

Berikut adalah format *JSON number* yang dapat dilihat pada Gambar 2.8:



Sumber gambar: http://json.org

Gambar 2.8 Format *JSON Number*

Number memiliki kemiripan dengan tipe data *number* pada bahasa pemrograman C atau Java, namun yang membedakan adalah pada *string JSON* tidak menggunakan format oktal dan hexadesimal.

2.8. CORS

CORS adalah singkatan dari *Cross-origin Resource Sharing*. Penggunaan praktik keamanan pada *CORS* memungkinkan *server* untuk memvalidasi datangnya sumber perintah yang dilihat dari *header* pengirim perintah [20]. Apabila sumber perintah datang dari URL yang terdaftar/valid, maka *server* akan mengirim balasan *header* 200 OK dan pesan yang terproses, namun bila sumber datang dari URL yang tidak terdaftar, maka *server* akan mengirim balasan *header* 403 *Forbidden* dan pesan kesalahan. *CORS* berfungsi untuk mencegah penyalahgunaan oleh pengirim perintah dari sumber yang tidak terdaftar yang mampu menyebabkan pencurian data. Biasanya *CORS* berguna pada praktik pertukaran data menggunakan AJAX JavaScript.

Sumber daya pada *web* saat ini banyak yang mengambil rujukan lain di luar *web*site itu sendiri. Entah itu gambar, *file* CSS, atau *file* JavaScript. Contoh pada *web*site http://abc.com ingin menampilkan gambar <img> dari *web*site http://cde.com, maka hal tersebut dinamakan *CORS*.

Untuk masalah keamanan, browser melarang kegiatan *cross-origin* (keluar dari domain asal) yang dilakukan dari *file* JavaScript. Contoh, *XML*HttpRequest dan Fetch akan menampilkan pesan error jika meminta sumber daya dari *web*site lain secara *default*. Maka dari itu, pada sisi *server* harus dilakukan validasi. Apakah permintaan yang datang dari domain yang diizinkan? Jika ya, maka lanjutkan permintaan. Pada sisi *server*, harus menambahkan aturan di bawah ini untuk dapat melakukan *cross-origin*:

```
Access-Control-Allow-Origin: http://cde.com
```

Dengan kode di atas, segala permintaan yang selain dari domain cde.com maka akan ditolah dan diberikan status kembali 403 Forbidden. Kita bisa saja menuliskan dengan aturan seperti di bawah ini untuk mengizinkan semua permintaan yang masuk:

```
Access-Control-Allow-Origin: *
```

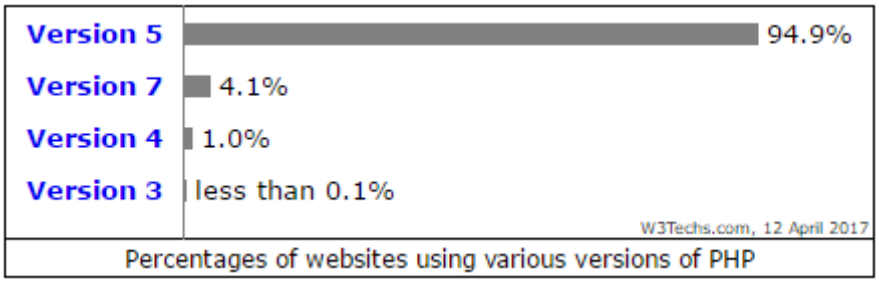
Namun, dengan praktik seperti akan membuka tingkat kerentanan sebuah sistem *web service*. Jadi, dengan kombinasi *access token* dan keamanan *CORS*, terdapat dua verifikasi untuk menggunakan sumber daya *web service*. Pertama, apakah permintaan yang datang, berasal dari domain yang diizinkan. Kedua, apakah permintaan yang datang membawa key atau *access token* yang benar.

2.9. PHP

PHP adalah sebuah bahasa pemrograman sumber terbuka yang ditujukan untuk pemrograman *web* dan dapat diaplikasikan ke *HTML*. *PHP* merupakan bahasa skrip yang ditanam dalam *HTML*. Ini berarti bahwa Anda dapat menggabungkan kode *PHP* dan *HTML* dalam berkas yang sama [21]. *PHP* merupakan bahasa pemrograman *server-side* yang di mana program akan

dijalankan di *server* dan hasilnya akan diintegrasikan ke dalam kode sumber *HTML*. *PHP* diciptakan untuk membuat halaman *web* yang interaktif.

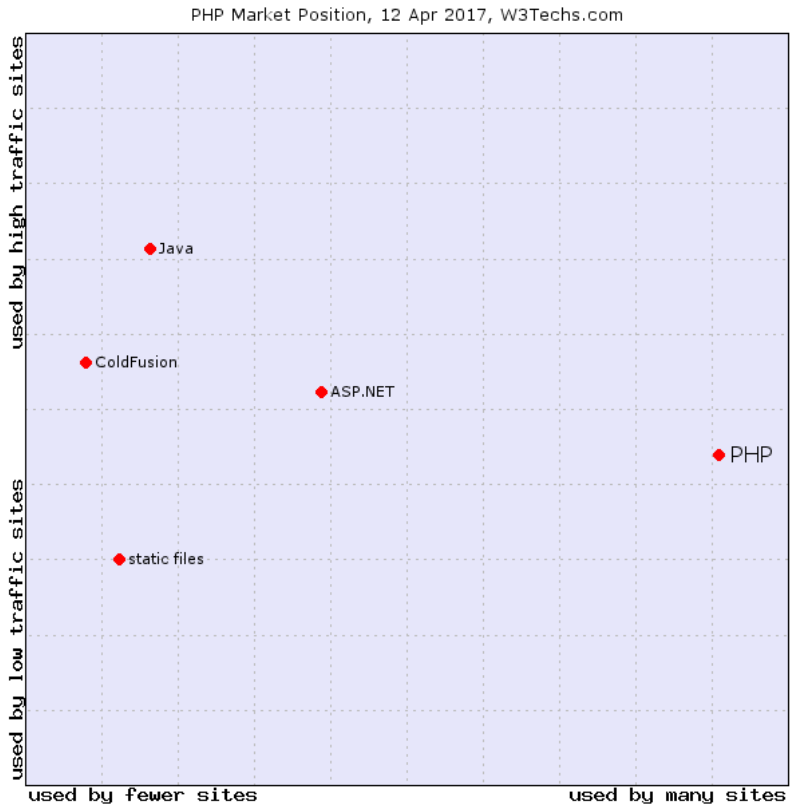
Dengan tujuan utama untuk membuat halaman *web* yang interaktif tersebut, *PHP* cukup dikenal oleh banyak kalangan pemrogram *web*. Menurut W3Techs—lembaga survei teknolgi *web*—menyatakan 82.7% *web*site yang menggunakan pemrograman *server*-side, menggunakan bahasa pemrograman PHP [22]. Versi PHP yang paling digunakan saat ini adalah PHP versi 5 dan diikuti dengan versi 7. Berikut adalah perbandingan penggunaan versi PHP yang dapat dilihat pada gambar 2.9:



Sumber gambar: http://w3techs.com

Gambar 2.9 Perbandingan penggunaan versi PHP

Dan gambar berikut menyatakan perbandingan bahasa pemrograman PHP dengan bahasa pemrograman populer yang lain yang dapat dilihat pada gambar 2.10:



Sumber gambar: http://w3techs.com

Gambar 2.10 Perbandingan PHP dengan bahasa pemrograman lain

PHP juga memiliki fitur yang cukup mumpuni untuk mengakomodasi dalam kebutuhan basis data. Terdapat modul untuk menggunakan basis data yang otomatis terinstal pada saat instalasi PHP pada komputer berhasil. Basis data yang didukung oleh PHP cukup beragam, misal, MySQL, Oracle DB, PostgreSQL, MongoDB, SQLite, dan lain-lain.

Program PHP diapit dengan sintak `<?php ?>` dan ekstensi format *file* php adalah .php. Program PHP dapat digabung dengan sintak HTML, namun hanya akan membaca kode yang berada di antara sintak `<?php ?>`, hasil pemrosesan akan ditampilkan kembali ke dalam kode HTML. Contoh sebuah program PHP:

```
1.  <!doctype html>
2.  <html>
3.  <head>
4.      <title>Contoh Program PHP dalam HTML</title>
5.  </head>
6.  <body>
```

```
7.      <p>
8.          <?php
9.              echo 'Sekarang adalah tahun ' . date('Y') . '.';
10.         ?>
11.     </p>
12. </body>
13. </html>
```

Program di atas, setelah diproses oleh kompiler PHP akan menghasilkan:

```
1.  <!doctype html>
2.  <html>
3.  <head>
4.      <title>Contoh Program PHP dalam HTML</title>
5.  </head>
6.  <body>
7.      <p>
8.          Sekarang adalah tahun 2017.
9.      </p>
10. </body>
11. </html>
```

2.10. MySQL

MySQL adalah sebuah aplikasi *server* basis data yang dijalankan di *server*, MySQL menggunakan sintaks SQL standar dalam penggunaannya. MySQL bersifat relasional yang memungkinkan satu tabel dapat berelasi dengan tabel lainnya, sehingga memungkinkan dilakukannya normalisasi untuk mencegah redundansi dalam pemakaian data. MySQL mampu menampung data yang sangat besar. MySQL menggunakan InnoDB mampu menampung sebanyak 1017 kolom dan besar data yang mencapai 256TB [23].

MySQL memiliki tipe-tipe bidang data yang dapat dituliskan ke dalam tabelnya. Tipe-tipe tersebut adalah format yang dapat diterima untuk setiap isiannya. Beberapa tipe bidang akan dijelaskan pada Tabel 2.1 [21].

Tabel 2.1 Tipe-tipe bidang pada MySQL

Tipe Bidang	Penjelasan Singkat
char	Sembarang data berbasis karakter dapat disimpan pada bidang ini, tetapi bidang tersebut akan memiliki panjang yang dibatasi dengan nilai di dalam kurung.
varchar	Sembarang data berbasis karakter dapat ditempatkan pada bidang ini, dan data tersebut dapat memiliki panjang yang bervariasi dari 0 sampai 255 karakter. Panjang maksimum bidang diberikan pada nilai di dalam kurung.

int	*Integer* yang memiliki rentang dari -2,147,483,648 sampai +2,147,483,647 dapat disimpan pada bidang ini. Parameter length membatasi jumlah dijit yang dapat ditampilkan, bukan nilainya.
text	Sembarang data berbasis karakter dapat ditempatkan pada bidang ini, dengan ukuran maksimum sebesar 65,536 karakter.
decimal	Bidang numerik yang dapat menyimpan nilai desimal. Parameter length membatasi jumlah dijit yang ditampilkan, dan parameter dec membatasi jumlah dijit di belakang titik desimal yang dapat disimpan.
enum	Dipakai hanya untuk nilai-nilai tertentu, seperti "*true*" dan "*false*".
datetime	Bidang multiguna yang dapat menyimpan tanggal dan waktu secara bersamaan sebagai YYYY-MM-DD' hh:mm:ss.

Berikut adalah beberapa contoh sintak dasar yang perlu diketahui dalam mengelola sebuah basis data MySQL:

1. Membuat *Database*

```
1.  create table db1;
```

2. Membuat Tabel

```
1.  create table mahasiswa (
2.    mahasiswa_id int(6) auto_increment primary key,
3.    nim varchar(12),
4.    nama_awal varchar(30) not null,
5.    nama_akhir varchar(30) not null,
6.    date_added datetime default CURRENT_TIMESTAMP
7.  )
```

3. Menambahkan Data

```
1.  insert into mahasiswa (nim, nama_awal, nama_akhir)
2.  values ('10113081', 'Iqbal', 'Perkasa');
```

4. Mengambil Data

```
1.  select * from mahasiswa;
```

5. Mengubah Data

```
1.  update mahasiswa set nama_akhir = 'Perdana' where nim = '10113081';
```

6. Menghapus Data

```
1.  delete from mahasiswa where nim = '10113081';
```

7. Menghapus Tabel

```
1.  drop table mahasiswa;
```

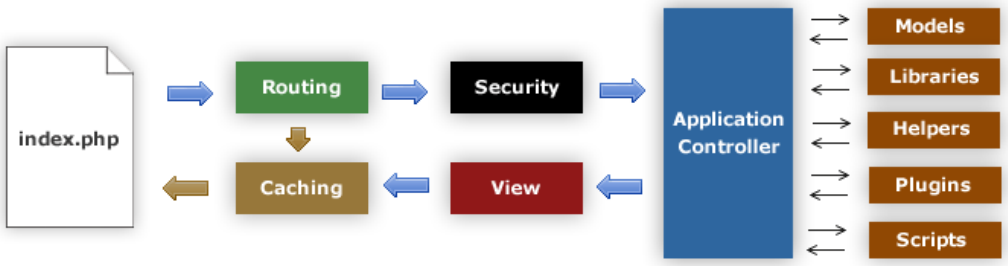
2.11. CodeIgniter

CodeIgniter adalah sebuah kerangka kerja sumber terbuka yang ditulis dalam bahasa pemrograman PHP. CodeIgniter menggunakan gaya arsitektur pemrograman MVC, yaitu *Model-View-Controller*, yang di mana masing-masing bagian memiliki peran yang berbeda. CodeIgniter memberikan kemudahan bagi pengembang *web* dalam membuat aplikasi berbasis *web* dengan lebih cepat karena kerangka kerja ini sudah dibekali dengan program-program yang lengkap. Program-program ini termasuk dalam penanganan basis data, routing, session, keamanan, dan lain-lain.

CodeIgniter sebelumnya dikembangkan dan diurus oleh perusahaan *internet* bernama EllibLab, namun pada tahun 2014, proyek ini dipindahalihkan ke British Columbia Institute of Technology (BCIT), Vancouver, Canada [24]. Pengembangan proyek dan perawatan kini ada pada tanggung jawab universitas tersebut.

2.11.1. Alur CodeIgniter

Berikut adalah alur dari data dari sistem CodeIgniter yang dapat dilihat pada Gambar 2.11:



Sumber gambar: http://www.codeigniter.com

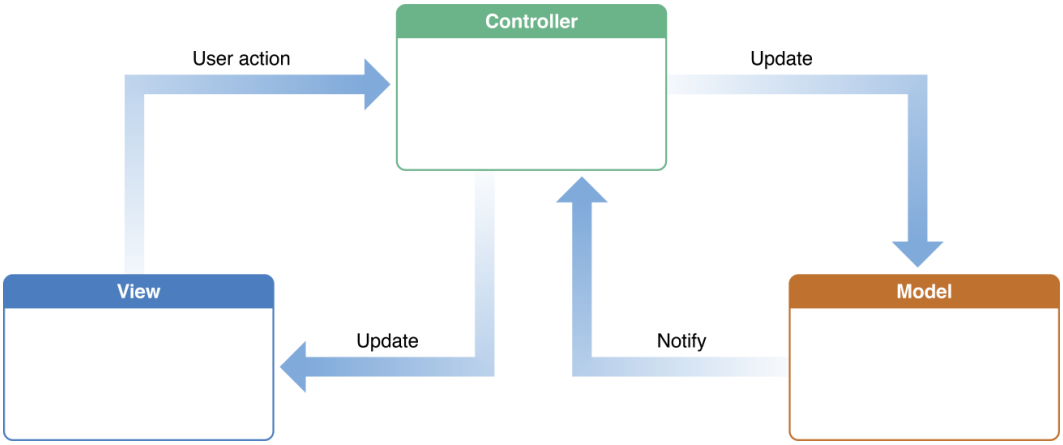
Gambar 2.11 Alur Data CodeIgniter

1. *File* index.php pasti akan dieksekusi terlebih dahulu dan sebagai gerbang utama sistem
2. *Router* akan mengarahkan request HTTP ke sumber daya atau program yang dimaksud
3. Jika *cache* tersedia, maka kirim nilai *cache* ke browser
4. Sebelum masuk ke kontroler sistem, program sekuriti akan mengecek dan memfilter masukan dari pengguna
5. Kontroler menjalankan program dan memuat sumber daya yang dibutuhkan
6. *View* akan menampilkan ke *browser,* jika *cache* diaktifkan, maka *view* akan memasukkan nilai ke *cache*.

2.11.2. MVC

MVC merupakan singkatan dari *Model-View-Controller*. MVC merupakan praktik pemrograman yang memisahkan antara ketiga komponen utama dari sebuah sistem. Tidak seperti praktik pemrograman bukan MVC, yang di mana ketiga hal yang sering dilakukan dalam pemrograman, yaitu program dengan basis data, penulisan logika, dan menampilkan ke browser dalam satu *file* atau kelas yang sama. MVC membagi ketiga komponen tersebut agar mudah dalam pengkodean dan pemeliharaan kode.

Dalam CodeIgniter, konsep MVC tidak terlalu ketat yang di mana pemrogram dapat menghilangkan bagian Model dan menyatukan dengan kelas controller. Jadi, view tetap menjadi bagian untuk menampilkan data, dan kelas controller merangkap menjadi logika program, dan program dengan basis data. Tentu saja program seperti akan lebih sulit dalam hal pemeliharaannya. Berikut adalah hubungan keterkaitan antar kelas dalam MVC yang dapat dilihat pada Gambar 2.12:



Sumber gambar: https://developer.apple.com

Gambar 2.12 *Model-View-Controller*

2.11.2.1. Model

Model menangani segala aktivitas dengan data. Sebagai contoh objek model akan merepresentasikan sebuah karakter dalam sebuah game atau kontak dalam buku kontak. Idealnya, objek model tidak boleh berurusan langsung dengan objek *view*, objek model tidak mempunyai kewajiban untuk berurusan dengan tampilan antarmuka [25].

2.11.2.2. View

View menangani segala aktivitas yang dapat dilihat oleh pengguna. Tugas dari objek *view* adalah menampilkan nilai-nilai yang sudah diolah dalam objek controller dan mengirimkan nilai-nilai masukan dari pengguna ke objek *controller* [25].

2.11.2.3. Controller

Controller adalah objek utama dan penengah dari ketiga kelompok ini. Objek controller mengkoordinasi alur kerja dari program dan mengatur hubungan antar kelas yang berbeda. Objek controller adalah berisi program-program dan logika dari sebuah aplikasi tersebut dan menjadi perantara antara objek model dan objek *view* [25].

2.12. NativeScript

NativeScript adalah sebuah kerangka kerja sumber terbuka untuk membuat sebuah aplikasi Android menggunakan bahasa pemrograman JavaScript. NativeScript bertugas mengubah bahasa pemrograman JavaScript yang ditulis oleh pengembang, menjadi bahasa *native* Java yang dimengerti oleh *smartphone*. NativeScript adalah bersifat terbuka dan bisa diunduh gratis. Keunggulan yang lain adalah NativeScript bisa mendukung sistem operasi iOS apabila pengembang menggunakan komputer macOS. NativeScript pertama kali diluncurkan pada bulan Maret 2015 dan dikembangkan oleh Telerik hingga saat ini.

Hal-hal yang harus dimiliki sebelum memulai menggunakan NativeScript adalah sebagai berikut:

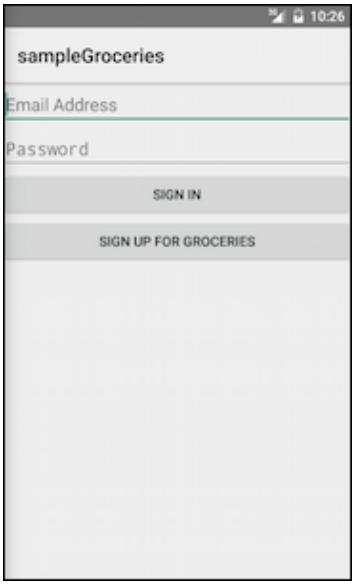
1. Kemampuan JavaScript dasar dan konsep-konsep dasar JavaScript seperti fungsi, operator percabangan, dan *looping*.
2. Kemampuan CSS tentang selector dan mengaplikasikan nilai CSS ke dalam propery yang tepat.
3. Aplikasi *terminal*, untuk menjalankan perintah-perintah yang disediakan oleh NativeScript, seperti perintah untuk *compile* dan menjalankan aplikasi di perangkat *smartphone*.
4. Aplikasi editor teks, untuk menulis program aplikasi. Dapat menggunakan editor teks apa saja, editor teks yang sederhana maupun IDE yang memiliki banyak fitur.

Berbeda dengan halnya pemrograman *web* yang menggunakan HTML. NativeScript menggunakan kode *XML* untuk membuat tampilan aplikasi. Namun untuk mengubah-ubah tampilannya, NativeScript sama dengan program *web* yaitu menggunakan CSS. Berikut adalah contoh membuat tampilan form login:

```
1.  <Page>
2.      <StackLayout orientation="vertical">
3.
4.          <TextField hint="Email Address" keyboardType="email" autocorrect
    ="false" autocapitalizationType="none" />
5.          <TextField hint="Password" secure="true" />
6.
7.          <Button text="Sign in" />
8.          <Button text="Sign up for Groceries" />
9.
```

```
10.     </StackLayout>
11. </Page>
```

Setelah dijalankan dan di-*compile* dengan kompiler dari NativeScript, maka akan menghasilkan tampilan yang dapat dilihat pada Gambar 2.13:



Sumber gambar: https://docs.nativescript.org

Gambar 2.13 Tampilan *form* login

2.13. Android

Android adalah sistem operasi untuk *smartphone* yang dikembangkan oleh Google yang ditujukan untuk perangkat layar sentuh. Sebagai sistem operasi, Android bertugas sebagai perantara antara pengguna dan perangkat tersebut. Aplikasi-aplikasi yang dibangun untuk sistem operasi Android, dibangun menggunakan bahasa pemrograman Java.

Android adalah sistem operasi yang dikembangkan untuk perangkat mobile berbasis Linux. API yang disediakan menawarkan akses ke *hardware*, data-data ponsel sekalipun, maupun data sistem sendiri. Setiap aplikasi pada Android memiliki tingkatan yang sama baik aplikasi inti maupun aplikasi pihak ketiga [26].

Android SDK dirilis pertama kali pada 12 November 2007 dan para pengembang memiliki kesempatan untuk memberikan umpan balik dari

pengembangan SDK tersebut. Pada bulan September 2008, T-Mobile memperkenalkan ketersediaan T-Mobile G1 yang merupakan smart phone pertama berbasiskan platform Android. Beberapa hari kemudian, Google merilis Android SDK 1.0. Google membuat *source code* dari platform Android menjadi tersedia di bawah lisensi Apache's *open source*.

Saat ini, Android sudah mengeluarkan versi ke 7.1 yang terus berkembang hingga saat ini, dan Android memiliki fitur-fitur yang mampu membantu pengembang dalam menggunakan fitur yang ada pada *smartphone*, antara lain: Dalvik *virtual machine* yang membagi masing-masing aplikasi Android memiliki sumber dayanya masing-masing, GSM yang memungkinkan pengguna dapat melakukan sambungan telepon, SQLite untuk membuat basis data relasional yang ringan, dan fitur-fitur lain seperti kamera, Bluetooth, WiFi, GPS, kompas, *accelerometer*, dan lain-lain.

2.13.1. Arsitektur Android

Sistem operasi Android memiliki komponen-komponen arsitektur sebagai berikut:

1. Aplikasi

   Ini adalah bagian teratas dari arsitektur. Komponen ini yang digunakan oleh pengguna akhir dan tampilan yang terlihat di layar *smartphone*. Terdapat banyak aplikasi di dalam perangkat Android, seperti, kalendar, kalkulator, SMS, kamera, dan banyak lagi.

2. Kerangka Kerja Aplikasi

   Android menawarkan para pemrogram untuk membangun aplikasi yang tidak terbatas kemampuannya. Pemrogram bebas untuk mengambil fitur-fitur dari perangkat keras, seperti akses lokasi informasi, menjalankan background services, mengatur alarm, menambahkan peringatan ke status bar, dan masih banyak lagi. Pemrogram memiliki akses yang penuh ke dalam kerangka kerja API yang sama yang digunakan oleh aplikasi utama.

3. Libraries

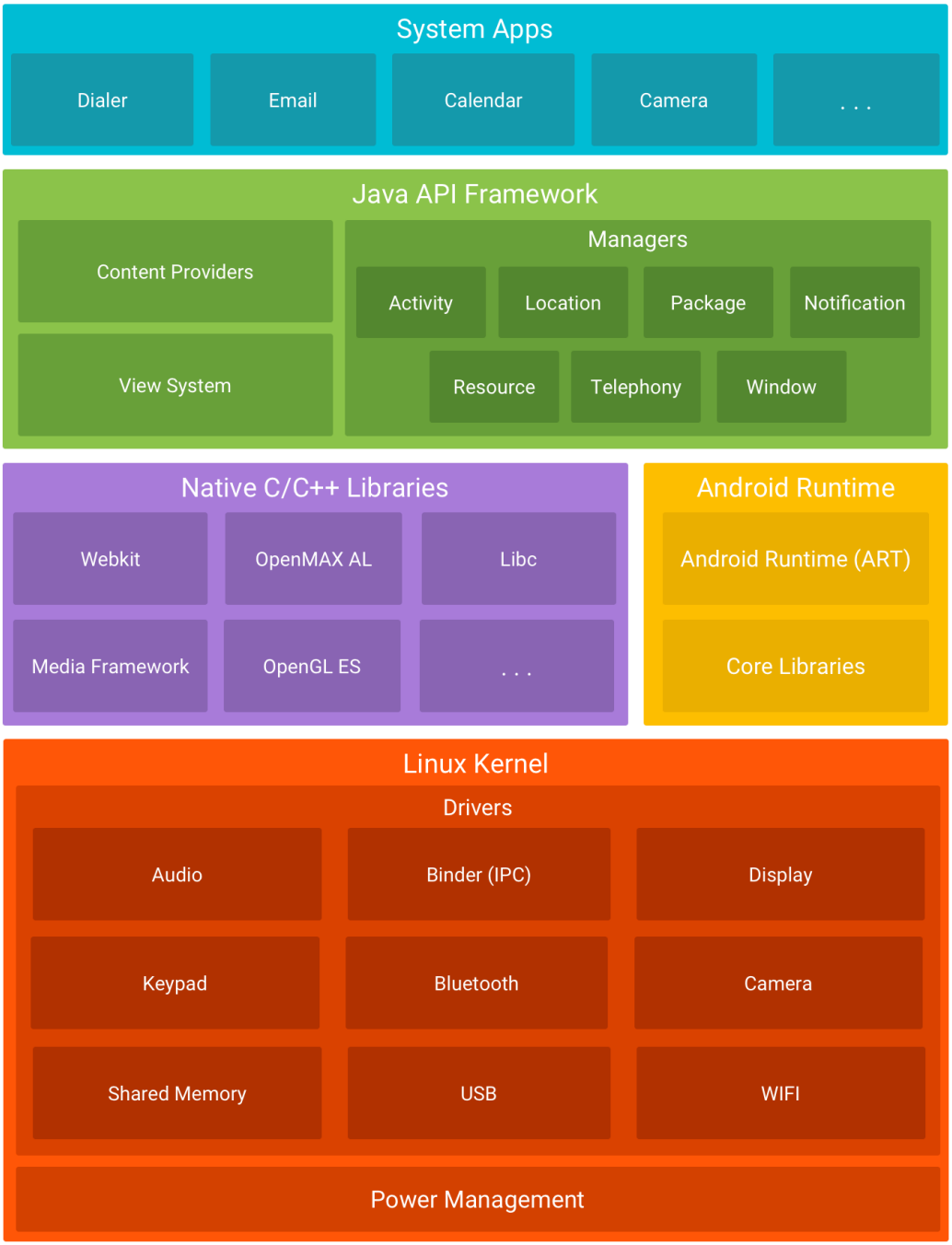
   Libraries merupakan bagian lebih dalam dari kerangka kerja dan merupakan kumpulan program untuk digunakan pada kerangka kerja.

4. *Android Runtime*

Tempat pembagian sumber daya (DVM) untuk masing-masing aplikasi di *smartphone*. DVM dirancang secara khusus untuk Android pada saat dijalankan pada lingkungan yang terbatas, dimana baterai yang terbatas, CPU, memori, dan penyimpanan data.

5. Linux *Kernel*

Kernel juga bertindak sebagai lapisan abstrak antara perangkat keras dan seluruh *software stack*. Merupakan bagian terdalam dan sebagai perantara akhir antara perangkat lunak dan perangkat keras.



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

Gambar 2.14 Arsitektur Sistem Operasi Android

2.13.2. Versi Android

Android versi pertama yaitu versi 1.0 keluar pada tanggal 23 September 2008 dan hingga versi paling mutakhir saat ini adalah versi 7.1.2 yang keluar pada

tanggal 22 Agustus 2016. Berikut adalah daftar versi Android yang telha dirilis yang dapat dilihat pada Tabel 2.2:

Tabel 2.2 Daftar Versi Android yang Telah Rilis

Nama Versi	Versi	Tanggal Rilis
Android (Alpha)	1.0	September 23, 2008
Beta (Internally known as "Petit Four")	1.1	February 9, 2009
Cupcake	1.5	April 27, 2009
Donut	1.6	September 15, 2009
Eclair	2.0 - 2.1	October 26, 2009
Froyo	2.2 - 2.2.3	May 20, 2010
Gingerbread	2.3 - 2.3.7	December 6, 2010
Honeycomb	3.0 - 3.2.6	February 22, 2011
Ice Cream Sandwich	4.0 - 4.0.4	October 18, 2011
Jelly Bean	4.1 - 4.3.1	July 9, 2012
KitKat	4.4 - 4.4.4	October 31, 2013
Lollipop	5.0 - 5.1.1	November 12, 2014
Marshmallow	6.0 - 6.0.1	October 5, 2015
Nougat	7.0 - 7.1.2	August 22, 2016

2.14. UML

UML merupakan singkatan dari *Unified Modelling Language* yaitu suatu metode permodelan secara visual untuk sarana perancangan sistem berorientasi objek, atau definisi UML yaitu sebagai suatu bahasa yang sudah menjadi standar pada visualisasi, perancangan dan juga pendokumentasian sistem perangkat lunak.

Beberapa tujuan atau fungsi dari penggunaan UML, yang diantaranya:

1. Dapat memberikan bahasa permodelan visual kepada pengguna dari berbagai macam pemerograman maupun proses rekayasa.
2. Dapat menyatukan praktek-praktek terbaik yang ada dalam permodelan.
3. Dapat memberikan model yang siap untuk digunakan, merupakan bahasa permodelan visual yang ekspresif untuk mengembangkan sistem dan untuk saling menukar model secara mudah.
4. Dapat berguna sebagai *blueprint*, sebab sangat lengkap dan detail dalam perancangannya yang nantinya akan diketahui informasi yang detail mengenai koding suatu program.

Dapat memodelkan sistem yang berkonsep berorientasi objek, jadi tidak hanya digunakan untuk memodelkan perangkat lunak saja. Dapat menciptakan suatu bahasa permodelan yang nantinya dapat dipergunakan oleh manusia maupun oleh mesin.

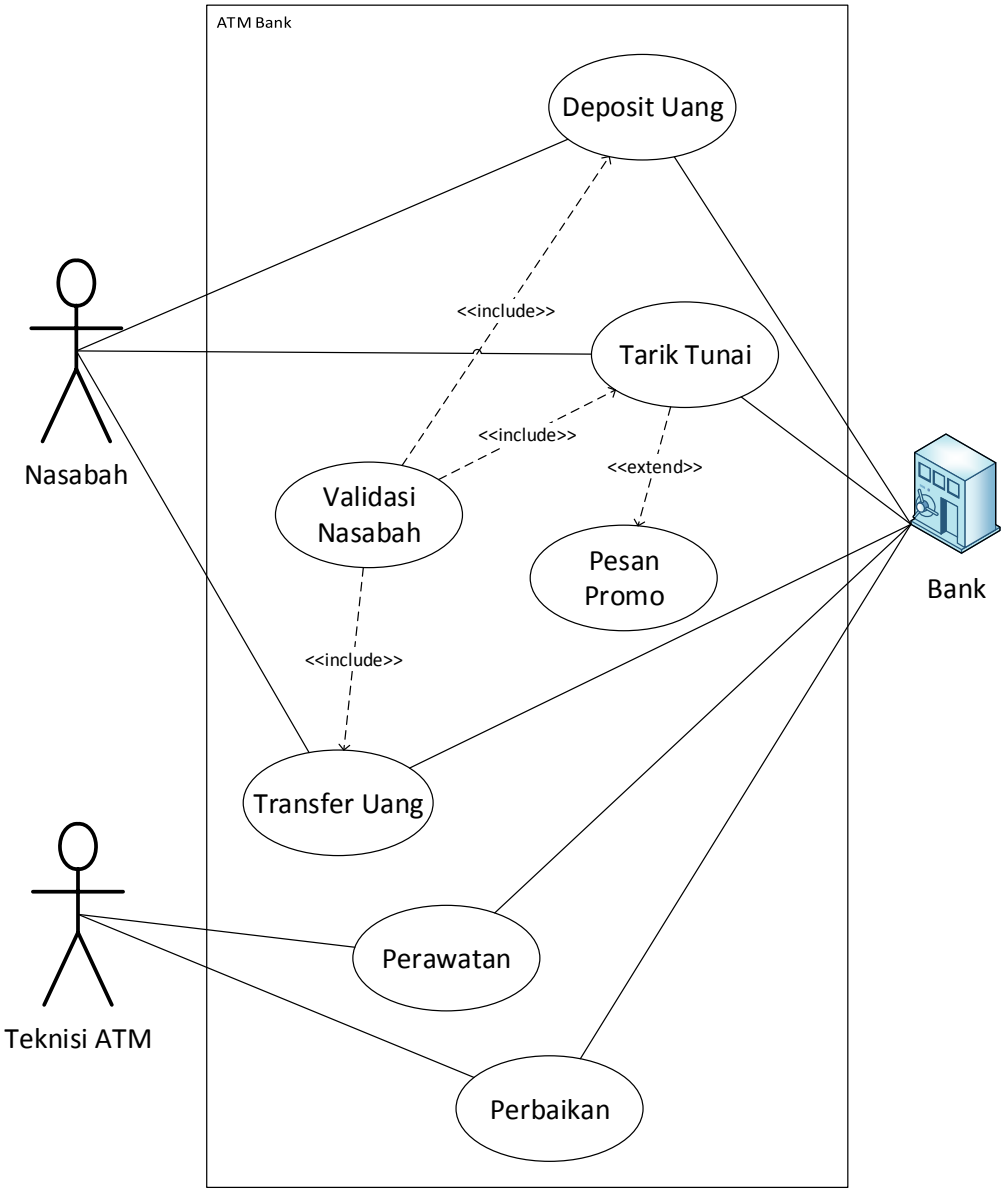
2.14.1. Use Case Diagram

Use Case Diagram adalah gambaran *graphical* dari beberapa atau semua actor, *use case*, dan interaksi di antaranya yang memperkenalkan suatu sistem. *Use case diagram* tidak menjelaskan secara detil tentang penggunaan *use case*, tetapi hanya memberi gambaran singkat hubungan antara *use case*, aktor, dan sistem. Di dalam *use case* ini akan diketahui fungsi-fungsi apa saja yang berada pada sistem yang dibuat.

Di dalam diagram *use case*, terdapat tiga garis dan panah yang menghubungkan antar komponen, baik aktor dengan use case atau use case dengan *use case* yang lainnya. Ketiga garis dan panah tersebut adalah sebagai berikut yang dapat dilihat pada Tabel 2.3 dan contoh *use case diagram* pada Gambar 2.15:

Tabel 2.3 Garis dan panah dalam diagram *use case*

Bentuk garis	**Nama**	**Keterangan**
	Asosiasi	Hubungan dan ketergantungan antar aktor dengan *use case* yang di jalani.
<include>	*Include*	Digunakan untuk menambahkan use case ke *use case* yang lain. Opsional, bisa dijalankan atau tidak.
<extend>	*Extend*	Digunakan untuk menyederhanakan satu *use case* yang digunakan dan harus dijalankan pada use yang lain.



Gambar 2.15 Contoh *Use Case Diagram* ATM Bank

2.14.2. Skenario *Use Case*

Skenario *Use Case* merupakan hasil instansiasi dan penjelasan dari setiap *Use Case*. Skenario *Use Case* menceritakan kegiatan-kegiatan dan detail yang terjadi pada sebuah pekerjaan. Format skenario *Use Case* adalah berbentuk tabel.

Komponen-komponen Skenario *Use Case*:

1. *Name*: Memberikan penjelasan singkat tentang nama dari use case.
2. *Actors*: Daftar aktor yang dapat mengakses use case.

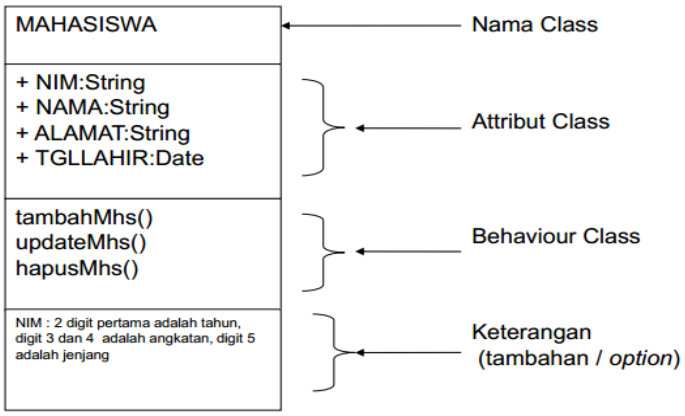
3. *Goals*: Menjelaskan apa yang aktor coba untuk dapatkan dari use case.
4. *Preconditions*: Kondisi sistem sebelum *use case* dijalankan.
5. *Summary*: Memberikan penjelasan singkat tentang deskripsi informal dari sesuai *use case*.
6. *Related use cases*: Daftar *use case* yang berhubungan dengan *use case* tersebut.
7. *Steps*: Menjelaskan setiap langkah yang dijalankan pada *use case* tersebut.
8. *Post conditions*: Kondisi sistem setelah *use case* dijalankan

Tabel 2.4 Contoh *Use Case Scenario* pada sistem login

<table>
<tr><td colspan="5">Use Case Login</td></tr>
<tr><td colspan="2">Nama Use Case:</td><td colspan="3">Login</td></tr>
<tr><td colspan="2">Aktor:</td><td colspan="3">Admin, Operator, dan Kepala Dinas</td></tr>
<tr><td colspan="5">Skenario:</td></tr>
<tr><td colspan="5">Tabel 3.12 Skenario Use Case Login</td></tr>
<tr><td colspan="2">Aksi Aktor</td><td colspan="3">Reaksi Sistem</td></tr>
<tr><td colspan="5">Skenario Normal</td></tr>
<tr><td>1.</td><td>Membukan halaman login</td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td>2.</td><td colspan="2">Menampilkan halaman login</td></tr>
<tr><td>3.</td><td>Mengisi username dan password</td><td></td><td colspan="2"></td></tr>
<tr><td>4.</td><td>Klik tombol “Masuk”</td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td>5.</td><td colspan="2">Mengarahkan pengguna ke halaman dashboard</td></tr>
<tr><td colspan="5">Skenario Alternatif (Username atau password salah)</td></tr>
<tr><td>1.</td><td>Membukan halaman login</td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td>2.</td><td colspan="2">Menampilkan halaman login</td></tr>
<tr><td>3.</td><td>Mengisi username dan password</td><td></td><td colspan="2"></td></tr>
<tr><td>4.</td><td>Klik tombol “Masuk”</td><td></td><td colspan="2"></td></tr>
<tr><td></td><td></td><td>5.</td><td colspan="2">Menampilkan pesan kesalahan</td></tr>
</table>

2.14.3. Class Diagram

Class diagram menggambarkan relasi antar kelas yang membangun sebuah sistem. Masing-masing bagan kelas menjelaskan identitas dan fungsi dari kelas tersebut. Terdapat empat bagian pada masing-masing bagan kelas, yaitu: nama, atribut, perilaku, dan keterangan kelas.



Gambar 2.16 Contoh bagan *Class Diagram* kelas mahasiswa

2.14.4. Activity Diagram

Activity diagram digunakan untuk menganalisis proses, *activity diagram* biasanya dibuat untuk untuk sebuah *use case* dan menampilkan beberapa skenario [27]. Sebuah *Activity Diagram* bukan sebuah *tool* yang sempurna untuk menganalisis masalah dari sistem. Setiap *Activity Diagram* selalu mempunyai satu *initial state*. *Initial node* yang digambarkan dengan simbol lingkaran padat, merupakan titik yang mengawali *Activity Diagram*. *Activity diagram* dapat diakhiri dengan memberikan *activity final diagram* yang digambarkan dengan lingkaran padat dengan mempunyai cincin dibagian luarnya.

a. Activity Node

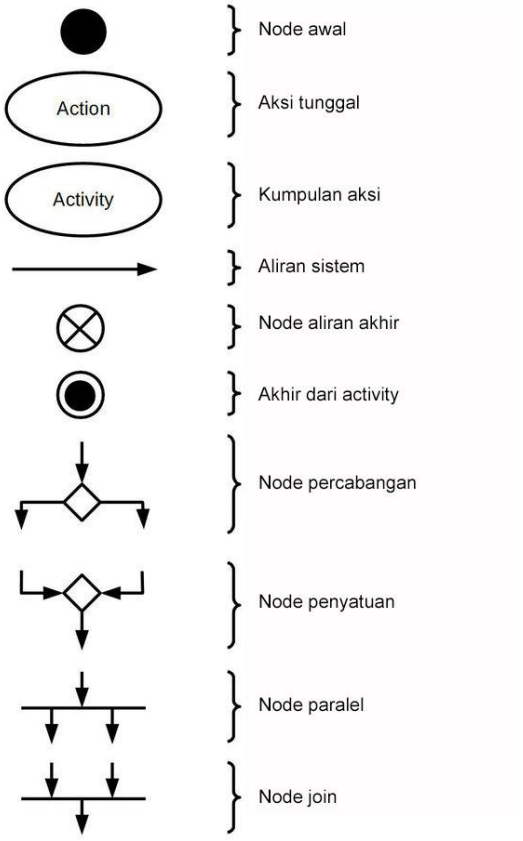
Activity node adalah sesuatu yang dilakukan atau yang sedang terjadi dalam *Activity Diagram*. *Activity node* mempunyai simbol yang hampir mirip dengan *use case* namun mempunyai bentuk yang lebih ramping dan menyerupai bujur sangkar.

b. Decision Node

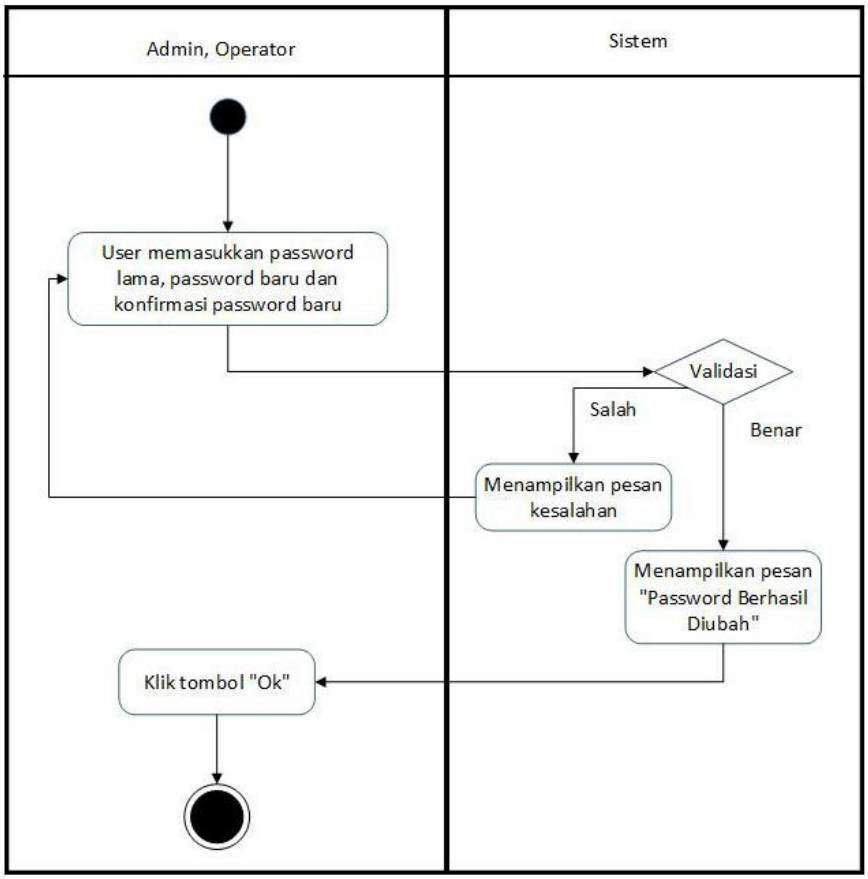
Decision node disebut *decision diamonds* di dalam *flowchart*. Simbol diamond adalah satu elemen yang membuat *Activity Diagram* mengingat akan *flowchart*, yang berguna untuk memberi kondisi percabangan.

c. Transition Fork

Keberadaan sebuah *transition fork* untuk menggambarkan tingkah laku yang pararel atau bercabang. Sebuah transition join untuk mempertemukan tingkah laku yang pararel.



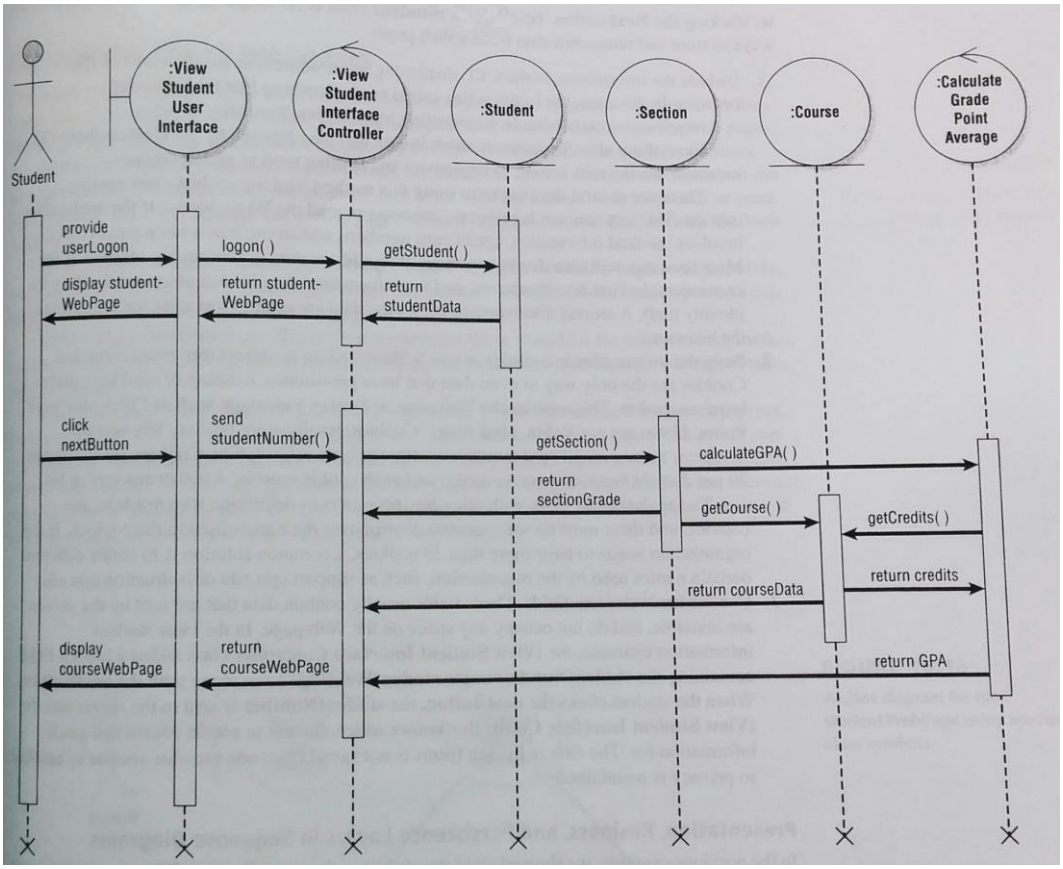
Gambar 2.17 Notasi *Activity Diagram*



Gambar 2.18 Contoh *Activity Diagram* sistem mengganti password

2.14.5. Sequence Diagram

Sequence Diagram digunakan untuk memberikan gambaran detail dari setiap *use case diagram* yang dibuat sebelumnya. *Sequence diagram* dapat mengilustrasikan rangkaian interaksi antara kelas-kelas dan objek pada suatu waktu. *Sequence diagram* diturunkan dari *use case* dan untuk menurunkan interaksi, relasi, dan metode dari objek sebuah sistem [27]. Setiap obyek yang terlibat dalam sebuah *use case* digambarkan dengan garis putus-putus vertikal, kemudian pesan yang dikirim oleh obyek digambarkan dengan garis horizontal secara kronologis dari atas ke bawah.



Sumber gambar: K. Kendall and J. Kendall, Systems Analysis and Design, Pearson, 2011

Gambar 2.19 Contoh *Sequence Diagram*