BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Tempat Penelitian

Tinjauan tempat penelitian adalah untuk mengetahui keadaan di tempat penelitian diantaranya adalah sejarah berdirinya, logo perusahaan, struktur organisasi, deskripsi tugas serta visi, misi dan tujuan.

1. 1. 1 Sejarah Perusahaan

Bandung Review didirikan pada akhir tahun 2009 oleh pendirinya Wina Paramasti dan kantornya bertempat di Jl. Kesehatan No. 11 Bandung. Tujuan awal didirikannya Bandung Review adalah untuk mendirikan sebuah organisasi yang melayani baik bisnis dan pengguna awam di bidang pariwisata (perhotelan) di kota Bandung dan sekitarnya.

Pada awal terbentuknya tim Bandung Review melakukan pengumpulan data dari berbagai tempat-tempat dan penyedia jasa layanan pariwisata di kota Bandung dan sekitarnya dengan mendatangi secara *door-to-door* sekaligus memperkenalkan layanan Bandung Review. Tim Bandung Review terdiri dari 8-10 karyawan yang bertugas sebagai *marketing*, konten administrator dan pengumpul data. Pengembangan aplikasi *web* BandungReview.com yang sebelumnya dikerjakan oleh jasa konsultan, kemudian diambil alih seiring dengan dibentuknya departemen IT Bandung Review yang berisi desainer dan *programmer*.

Pada tanggal 1 April 2010 merupakan *milestone* dari versi *beta* BandungReview.com dan mendapatkan respon yang baik, terlihat peningkatan yang tajam dari statistik *page rank* Alexa, jumlah pengikut di Twitter dan Facebook. Jika dilihat dari model bisnis yang ditawarkan oleh Bandung Review terbagi menjadi 4 bagian antara lain BandungReview.com, *event organizer*, *tour operator* dan *offline media*. Sebagai *event organizer*, Bandung Review pernah menyelenggarakan beberapa *event* dan bekerja sama dengan produk minuman Pocari Sweat. Bandung Review sebagai *tour operator* bekerja sama dengan beberapa instansi atau komunitas yang membutuhkan jasa pemandu *tour* dan bekerja sama dengan berbagai penyedia layanan akomodasi dan cinderamata khas Bandung. *Offline media* yang dijalankan Bandung Review berbentuk *traveller booklet* yang dibagikan kepada wisatawan dan penyedia jasa layanan.

Versi 1.0 BandungReview.com dirilis pada tanggal 1 Oktober 2010 dengan tampilan yang lebih dinamis dan berwarna. Fitur-fitur unggulan ditambahkan seperti direktori, *review*, artikel, *advertorial*, *event, hot pick* dan promosi.

Pada perkembangan selanjutnya, Bandung Review akan memfokuskan pada pengembangan BandungReview.com dan meninggalkan model bisnis lainnya seperti *event organizer, tour operator* dan *offline media*. Strategi untuk memperkuat konten BandungReview.com dengan berbagai kategori baru, integrasi dengan peta digital, memperbaiki desain *web* yang lebih *compact* dan elegan serta menonjolkan interaktivitas penggunanya akan menjadi fokus dalam pengembangan BandungReview.com.

1. 1. 2 Logo Perusahaan

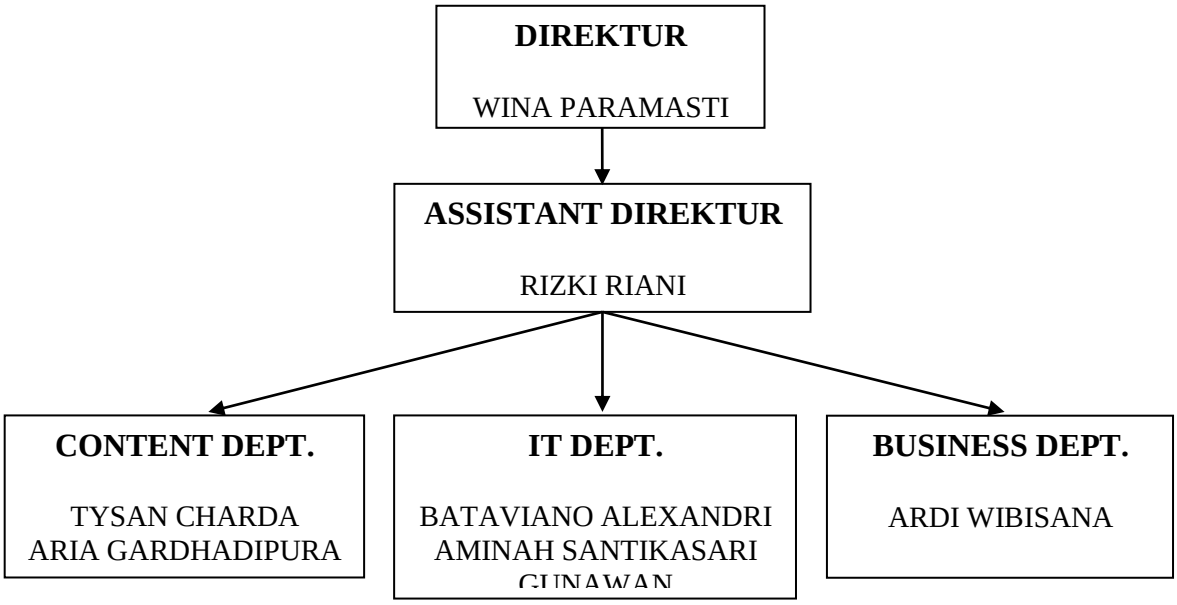
Logo ataupun lambang dari BandungReview.com sekarang adalah seperti Gambar 2.1.



Gambar 2. 1 Logo BandungReview.com

1. 1. 3 Struktur Organisasi

PT. Bandung Review terdiri dari departemen atau divisi yang memiliki peranannya masing-masing. Adapun struktur organisasi PT. Bandung Review dapat dilihat pada Gambar 2.2.



Gambar 2. 2 Struktur Organisasi PT. Bandung Review

1. 1. 4 Deskripsi Tugas

Deskripsi tugas digunakan untuk mengetahui tugas, wewenang dan tanggung jawab dari masing-masing departemen. Adapun deskripsi tugas yang ada di PT. Bandung Review adalah sebagai berikut:

1. Asisten Direktur

- Mengelola finansial dan administrasi perusahaan
- Memberikan laporan finansial dan administrasi ke direktur

2. Departemen Konten

- Mengelola konten web BandungReview.com
- Mengelola informasi Bandung Review pada jejaring sosial
- Bagian *account executive* bertugas mempromosikan layanan Bandung Review kepada tempat-tempat dan penyedia jasa layanan

3. Departemen IT

Terbagi menjadi dua bagian yakni *programmer* dan desainer.

- *Programmer* bertugas melakukan pengembangan dan pengelolaan sistem BandungReview.com
- Desainer bertugas melakukan perancangan visual BandungReview.com

4. Departemen Bisnis

- Mengamati model bisnis yang selama ini telah ada
- Melihat peluang untuk meraih keuntungan yang lebih besar dari bisnis yang ada
- Melihat peluang pengembangan bisnis yang memungkinkan dengan mengacu pada *core business* yang sudah ada

1. 1. 5 Visi, Misi dan Tujuan

Berikut penjabaran dari visi, misi dan tujuan PT. Bandung Review adalah:

2.1.5.1 Visi

Situs *online* direktori pariwisata terbesar untuk kota Bandung dan sekitarnya.

2.1.5.2 Misi

1. Membangun jaringan Business-to-Business (B2B) industri pariwisata dan perhotelan di kota Bandung.
2. Mengumpulkan informasi yang paling lengkap mengenai kota Bandung, terutama tempat-tempat yang menjadi animo masyarakat.
3. Mengimplemantasikan teknologi Web 2.0 untuk memperkaya konten *website.*

2.2.4.3 Tujuan

Organisasi yang melayani baik kepada entitas bisnis dan masyarakat umum di bidang pariwisata (perhotelan) di kota Bandung dan sekitarnya.

1. 2 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan aplikasi *mobile* Bandung Review berbasis Windows Phone.

1. 2. 1 Bandung Review

Bandung Review merupakan pusat direktori terlengkap dan terbesar yang meliputi segala bentuk penyedia layanan pariwisata di kota Bandung dan sekitarnya. Bandung Review adalah sebuah *media online* berbentuk *web portal*

yang dapat diakses oleh pengguna internet sebagai sarana pencari informasi pariwisata di sekitar Bandung.

Bandung Review memberikan kemudahan bagi para potensial konsumen pariwisata untuk dengan tepat mengetahui lokasi, fasilitas dan kualitas para penyedia layanan wisata. Hal ini akan sangat berguna bagi penyedia layanan wisata sebagai media publikasi.

Sampai pada akhir bulan April 2012, Bandung Review telah memiliki 5210 lebih data tempat dengan statistik jumlah pengunjung rata-rata per hari 600 pengunjung sedangkan per bulan rata-rata 20.000 pengunjung dan merupakan situs *online* direktori informasi terbesar seputar kota Bandung dan sekitarnya.

Beberapa fitur unggulan yang terdapat pada Bandung Review antara lain sebagai berikut:

1. Direktori dan *Review*

Merupakan daftar penyedia layanan wisata yang dikelompokkan menjadi beberapa kategori. Khusus untuk kategori kuliner Bandung Review akan melakukan penilaian secara objektif untuk menentukan rating produk tersebut.

2. Artikel

Dengan tim penulis yang handal, Bandung Review akan memperkaya isi *web* dengan artikel-artikel mengenai wisata dengan balutan yang unik dan menarik ala Bandung Review.

3. *Advertorial*

Merupakan alternatif dari direktori, berupa artikel yang ditulis sebagai bentuk promosi untuk suatu penyedia layanan.

4. *Event*

Merupakan daftar kegiatan atau acara bersifat umum yang akan diselenggarakan di kota Bandung dan sekitarnya.

5. Promosi

Merupakan daftar promosi atau penawaran diskon dari berbagai penyedia layanan.

6. Peta

Salah satu kekuatan yang dimiliki Bandung Review adalah penggunaan peta yang cukup lengkap dalam penamaan dan legenda.

1. 2. 2 Sistem Informasi

2.2.2.1 Konsep Dasar Sistem [9]

Suatu sistem sangatlah dibutuhkan dalam suatu perusahaan atau instansi pemerintahan, karena sistem sangatlah menunjang terhadap kinerja perusahaan atau instansi pemerintah, baik yang berskala kecil maupun besar. Sistem merupakan suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu.

Dari pengertian diatas dapat diambil suatu kesimpulan bahwa suatu sistem merupakan sekumpulan elemen yang saling berkaitan dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu.

2.2.2.1.1 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau ciri-ciri tertentu yaitu sebagai berikut:

1. Komponen Sistem (*component*)

Pada suatu sistem terdiri dari beberapa atau sejumlah komponen yang saling berhubungan, atau berinteraksi antara komponen yang satu dengan komponen yang lainnya, atau bekerja sama untuk membentuk satu kesatuan dan dalam komponen sistem ini atau elemen-elemen sistem ini dapat berupa subsistem yang lebih kecil.

2. Batasan Sistem (*boundary*)

Batasan sistem merupakan garis besar abstraksi yang memisahkan antara sistem dan lingkungannya, batas sistem ini juga menunjukan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*environment*)

Lingkungan luar dari suatu sistem adalah apapun yang terdapat diluar batas dari sistem yang memepengaruhi operasi sistem, lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga merugikan sistem tersebut lingkungan luar sistem yang menguntungkan merupakan energi dari sistem dan demikian harus tetap dijaga dan dipelihara.

4. Penghubung Sistem (*interface*)

Penghubung sistem merupakan suatu media antara satu subsistem dengan yang lainnya, melalui penghubung ini memungkinkan sumber-sumber mengalir dari satu subsistem ke subsistem yang lainnya.

5. Masukan Sistem (*input*)

Masukan atau *input* merupakan energi yang dimasukan ke dalam sistem, masukan ini dapat berupa energi manusia, data model, bahan baku, layanan atau

lainnya, *input* atau masukan dapat diklasifikasikan menjadi 3 kategori yaitu, *serial input*, *probable input*, dan *feedback input*.

6. Keluaran Sistem (*output*)

Keluaran sistem merupakan hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna, pembuangan keluaran sistem dapat merupakan masukan untuk susbsistem yang lain atau menuju kepada suatu sistem, definisi lain ialah *output* merupakan hasil dari porses yang merupakan tujuan dari keberadaan sistem.

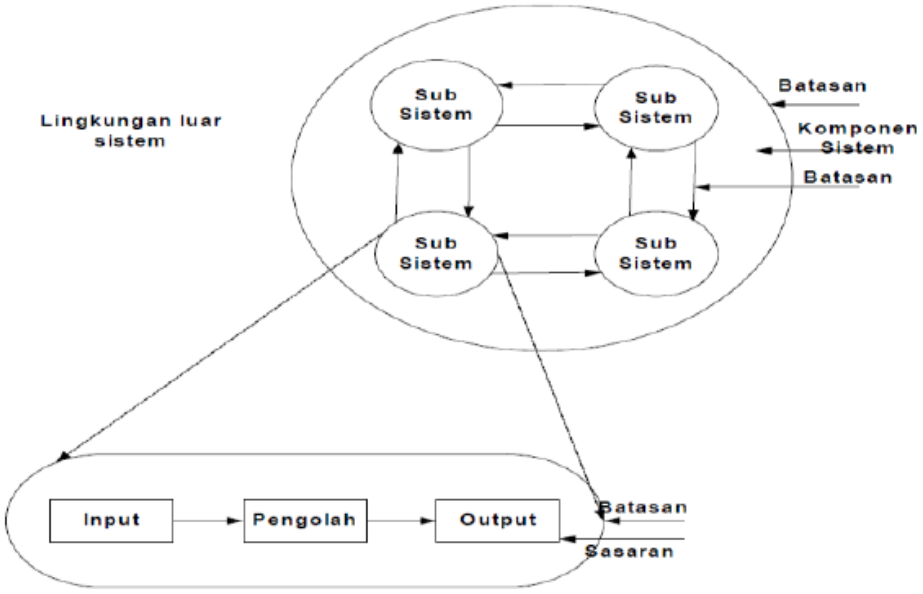
7. Pengolah Sistem (*proses*)

Pengolahan atau proses merupakan perubahan dari masukan atau *input* menjadi keluaran proses ini mungkin dilakukan oleh mesin, orang atau komputer.

8. Sasaran Sistem (*objective*)

Sasaran dari suatu sistem sangat menentukan sekali masukan yang dibutuhkan sistem serta keluaran yang akan dihasilkan suatu sistem. Serta dikatakan berhasil jika mengenai sasaran atau tujuannya, karena tujuan ini merupakan target atau sasaran akhir yang akan dicapai oleh suatu sistem.

Karakteristik suatu sistem dapat dilihat pada Gambar 2.3 dibawah ini:



Gambar 2. 3 Karakteristik Sistem

2.2.2.1.2 Klasifikasi Sistem

Sistem dapat dikelompokan atau diklasifikasikan menjadi beberapa sudut pandang diantaranya adalah sebagai berikut :

1. Sistem abstrak (*abstract system*) dan sistem fisik (*physical system*), sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik.
2. Sistem alamiah (*natural system*) dan sistem buatan manusia (*human made system*). Sistem alamiah adalah sistem yang terjadi melalui proses alam dan tidak dibuat oleh manusia. Sistem buatan manusia adalah sistem yang dirancang oleh manusia dan melibatkan interaksi antara manusia dengan mesin.
3. Sistem tertentu (*deterministic system*) dan sistem tak tentu (*probabilistic system*), sistem tertentu adalah sistem yang beroperasi dengan tingkah laku

yang sudah dapat diprediksi, interaksi diantara bagian-bagian dapat dideteksi dengan pasti sehingga keluaran atau *output* dapat diramalkan. Sistem tak tentu adalah sistem yang kondisi masa depan atau yang akan datang tidak dapat diprediksi karena mengandung unsur probabilitas.

4. Sistem tertutup (*closed system*) dan sistem terbuka (*open system*), sistem tertutup adalah sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luar. Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luar.

2.2.2.2 Konsep Dasar Data dan Informasi [9]

2.2.2.2.1 Pengertian Data

Data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak sehingga perlu diolah lebih lanjut.

2.2.2.2.2 Pengertian Informasi

Informasi adalah data yang telah diolah menjadi suatu bentuk yang berarti bagi penerima dan bermanfaat dalam mengambil keputusan saat ini atau yang akan datang.

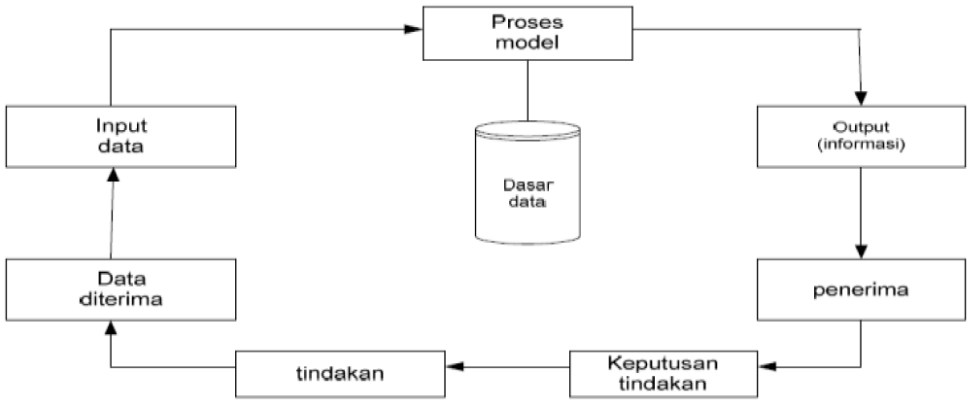
Dalam hal ini, data bisa dianggap sebagai objek dan informasi adalah suatu subyek yang bermanfaat bagi penerimanya. Informasi juga bisa disebut sebagai hasil pengolahan atau pemrosesan data.

Sumber dari informasi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata, tetapi data merupakan bentuk yang masih mentah dan belum dapat memberikan arti banyak bagi penggunanya, sehingga perlu diolah lebih lanjut untuk menghasilkan

informasi yang dibutuhkan. Dengan kata lain informasi adalah hasil dari pengolahan data.

2.2.2.2.3 Siklus Informasi

Informasi merupakan suatu proses perubahan dunia menjadi informasi, data yang diolah menjadi suatu model menjadi informasi penerima, kemudian menerima informasi tersebut membuat suatu keputusan dan melakukan tindakan yang berarti menghasilkan suatu keputusan dan melakukan tindakan yang lain yang akan membuat sejumlah data kembali, untuk lebih jelasnya dapat dilihat pada Gambar 2.4 siklus informasi berikut ini :



Gambar 2. 4 Siklus Informasi

Kegunaan informasi adalah untuk mengurangi ketidakpastian didalam proses pengambilan keputusan tentang suatu keadaan. Informasi yang digunakan di dalam suatu sistem informasi umumnya digunakan untuk beberapa kegunaan.

2.2.2.2.4 Kualitas Informasi

Kualitas dari sistem informasi tergantung dari 6 hal, antara lain :

1. Akurat (*accurate*)

Akurat berarti informasi harus bebas dari kesalahan yang tidak bisa atau menyesatkan.

2. Tepat Pada Waktunya (*timeliness*)

Tepat pada waktunya berarti informasi yang datang pada penerima tidak boleh terlambat.

3. Relevan (*relevance*)

Relevan berarti informasi tersebut mempunyai manfaat untuk pemakainya.

4. Ekonomis (*economy*)

Ekonomis berarti informasi harus mempunyai daya jual yang tinggi, biaya untuk menghasilkan informasi tersebut minimal.

5. Efisien (*efficiency*)

Efisien berarti informasi yang berkualitas memiliki sintak atau kalimat yang sederhana, namun mampu memberikan makna dan hasil yang mendalam.

6. Dapat Dipercaya (*reliability*)

Dapat dipercaya berarti informasi tersebut berasal dari sumber yang dapat dipercaya.

2.2.2.2.5 Nilai Informasi

Nilai dari informasi ditentukan dari dua hal yaitu manfaat dan biaya mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya sebagian besar informasi tidak dapat ditaksir keuntungannya dengan suatu nilai uang, tetapi dapat ditaksir nilai efektifitasnya.

2.2.2.3 Konsep Dasar Sistem Informasi [9]

Dalam suatu instansi atau perusahaan, sistem informasi merupakan jantung dari semua kegiatan manajemen. Juga telah dijelaskan bahwa informasi merupakan hal yang sangat penting bagi manajemen dalam pengambilan suatu keputusan. Sistem informasi diperlukan mulai dari perencanaan, operasi, pemeliharaan hingga pengendalian.

Dalam proses perencanaan sistem informasi diperlukan model perencanaan, data masuk, dan simulasi model berupa sistem konversi untuk mengubah masukan menjadi suatu keluaran yang dapat dijadikan penunjang dalam pengambilan suatu keputusan bagi pihak manajemen.

Berdasarkan pemaparan sebelumnya mengenai sistem, data, dan informasi maka dapat diambil kesimpulan bahwa pengertian dari sistem informasi adalah sekumpulan komponen-komponen dan prosedur-prosedur yang saling bekerja sama untuk mencapai suatu tujuan tertentu. Berikut pengertian-pengertian lainnya dari sistem informasi:

1. Kegiatan dari prosedur-prosedur yang diorganisasikan, bilamana dieksekusi akan menyediakan informasi untuk mendukung pengambilan keputusan dan pengendalian dalam organisasi.
2. Cara yang sudah tertentu untuk menyediakan informasi yang dibutuhkan oleh organisasi untuk beroperasi dengan cara yang sukses dan untuk organisasi bisnis dengan cara yang menguntungkan.
3. Sistem dalam orang yang memperhatikan kebutuhan pengolahan transaksi harian, yang mendukung operasi bersifat manajerial dan kegiatan strategis dari

suatu organisasi dengan pihak-pihak tertentu, dengan menghasilkan laporan-laporan yang diperlukan.

Terdapat tiga aktifitas dalam sistem informasi menghasilkan informasi pada organisasi yang diperlukan untuk membuat suatu keputusan, pengkontrolan, analisa masalah-masalah dan membuat produk baru atau layanan. Aktifitas itu adalah *input*, *process* dan *output*.

Input adalah menangkap atau mengumpulkan data mentah (*raw data*) di dalam organisasi atau dari luar organisasi - lingkungan (*environment*). Proses berarti dengan menterjemahkan data *input* kedalam bentuk yang lebih berarti. Sedangkan *output* adalah mentransfer informasi yang telah diproses kepada manusia dan aktivitasnya.

1. 2. 3 Konsep Basis Data [4]

a. Pengertian Basis Data

Database atau basis data merupakan kumpulan *file-file* yang saling berelasi, relasi tersebut biasa ditunjukkan dengan kunci dari tiap *file* yang ada. Satu basis data menunjukkan satu kumpulan data yang dipakai dalam satu lingkup perusahaan/instansi.

Dalam satu *file* terdapat *record-record* yang sejenis, sama besar, sama bentuk, merupakan satu kumpulan *entity* yang seragam. Satu *record* terdiri dari *field-field* yang saling berhubungan untuk menunjukkan bahwa *field* tersebut dalam satu pengertian lengkap dan direkam dalam satu *record*.

Satu *Database Management System* (DBMS) berisi satu koleksi data yang saling berelasi dan satu *set program* untuk mengakses data tersebut. Jadi, DBMS

terdiri dari *database* dan *set program* pengelola untuk menambah, menghapus, mengambil dan membaca data.

b. Normalisasi

Normalisasi merupakan suatu teknik untuk mengorganisasi data ke dalam tabel-tabel untuk memenuhi kebutuhan pemakai di dalam suatu organisasi.dan tujuan dari normalisasi itu sendiri adalah untuk menghilangkan kerangkapan data, mengurangi kompleksitas, serta mempermudah pemodifikasian data.

c. Tabel Relasi

Pengertian tabel relasi adalah data yang menggambarkan hubungan antara tabel yang satu dengan tabel yang lainnya.

Model basis data relasional sering pula disebut sebagai Model Relasional atau Basis Data Relasional. Model basis data ini ditemukan atau diperkenalkan pertama kalinya oleh E.F Codd. Model basis data menunjukan suatu cara atau mekanisme yang digunakan untuk mengelola atau mengorganisasi data secara fisik dalam memori sekunder yang berdampak pula pada bagaimana kita mengelompokan dan membentuk keseluruhan data yang terkait dalam sistem yang sedang ditinjau.

1. 2. 4 *Object Oriented Programming* (OOP) [10]

Object Oriented Programming (OOP) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah, tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh terdapat sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bagian administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.

Pemrograman orientasi-objek menekankan konsep berikut:

- Kelas (*Class*) — kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh '*class of dog*' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah

class secara tipikal sebaiknya dapat dikenali oleh seorang *non-programmer* sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

- Objek (*Object*) - membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
- Abstraksi (*Abstract*) - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.
- Enkapsulasi (*Encapsulation*) - Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana

objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

- Polimorfisme (*Polimorfism*) melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesan tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tunggal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.
- Inheritas (*Inheritance*)- Mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada - objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).

1. 2. 5 Konsep Dasar Analisis Sistem

2.2.5.1 Unified Modelling Language (UML) [10]

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, memspesifikasikan, dan membangun sistem perangkat lunak. UML adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM. UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

2.2.5.1.1Diagram UML

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use Case Diagram* untuk memodelkan proses bisnis.
2. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. *Sequence Diagram* untuk memodelkan pengiriman pesan (*message*) antar objek.
4. *Collaboration Diagram* untuk memodelkan interaksi antar objek.
5. *State Diagram* untuk memodelkan perilaku objek di dalam sistem.
6. *Activity Diagram* untuk memodelkan perilaku *use case* dan objek di dalam sistem.
7. *Class Diagram* untuk memodelkan struktur kelas.
8. *Object Diagram* untuk memodelkan struktur objek.
9. *Component Diagram* untuk memodelkan komponen objek.
10. *Deployment Diagram* untuk memodelkan distribusi aplikasi.

Berikut akan dijelaskan empat macam diagram yang paling sering digunakan dalam pembangunan aplikasi berorientasi objek, yaitu *use case diagram, sequence diagram, collaboration diagram,* dan *class diagram*.

1. *Use Case Diagram*

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri atas diagram untuk *use case* dan *actor*. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi.

Use case merepresentasikan operasi-operasi yang dilakukan oleh *actor*. *Use case* digambarkan berbentuk elips dengan nama operasi dituliskan di

dalamnnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *use case*.

2. *Sequence Diagram*

Sequence diagram menjelaskan secara detail urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar class, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

3. *Collaboration Diagram*

Collaboration diagram dipakai untuk memodelkan interaksi antar objek di dalam sistem. Berbeda dengan *sequence diagram* yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan, *collaboration diagram* lebih fokus pada pemahaman atas keseluruhan operasi yang dilakukan oleh objek.

4. *Class Diagram*

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class diagram* menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan.

1. 2. 6 Algoritma *Fixed-radius Near Neighbor* [3]

Algoritma *fixed-radius near neighbor* merupakan varian dari algoritma *nearest neighbor search*. Suatu daerah atau *range* diberikan masukan satu set poin di dimensi ruang *Euclidean* dengan besaran jarak yang bernilai tetap. Dari radius tetap tersebut nantinya akan dihasilkan *query* permintaan titik-titik yang berada di

dalam radius tersebut. Algoritma ini juga telah digunakan oleh Bentley pada tahun 1975 untuk memvisualisasikan struktur molekul dan beberapa pengaplikasiannya.

1. 2. 7 Algoritma *Dijkstra* [13]

Edsger Wybe Dijkstra, menemukan suatu algoritma untuk mencari lintasan terpendek pada suatu graf. Algoritma Dijkstra pada awalnya diterapkan pada graf berarah, tetapi ternyata algoritma ini juga dapat diterapkan untuk algoritma graf tak-berarah. Algoritma ini menggunakan prinsip *greedy* yang digunakan untuk menyatakan bahwa pada setiap langkah kita memilih sisi yang berbobot minimum dan memasukannya ke dalam himpunan solusi. Akan tetapi bobot dari graf tersebut harus bernilai bilangan positif (bobot >=0).

Input algoritma ini adalah sebuah graf berarah yang berbobot (*weighted directed graph*) G dan sebuah sumber verteks s dalam G dan V adalah himpunan semua *vertices* dalam graf G.

Algoritma dijkstra dimulai dari sebuah simpul asal dan dalam setiap iterasinya menambahkan sebuah verteks lain ke lintasan terpendek pohon merentang. Verteks ini merupakan titik terdekat ke akar namun masih diluar bagian pohon.

Properti Algoritma Djikstra :

1. Matriks Ketetangga M[m_{ij}]

   m_{ij} = bobot sisi (i,j) ; pada graf tak berarah $m_{ij} = m_{ji}$

   $m_{ii} = 0$

   $m_{ij} = \infty$, jika tidak ada sisi dari simpul i ke simpul j

2. Larik S = [s_i] yang dalam hal ini,

s_i = 1, jika simpul i termasuk ke dalam lintasan terpendek

s_i = 0, jika simpul i tidak termasuk ke dalam lintasan terpendek

3. Larik/tabel D = [d_i] yang dalam hal ini,

   d_i = panjang lintasan dari simpul awal s ke simpul I.

Algoritma Djikstra dalam pseudo-code:

```
procedure Dijkstra (input m: matriks, a:simpul awal)
{mencari lintasan terpendek dari simpul awal a ke semua
simpul lainya
Masukan : matriks ke tetanggaa (m) dari graf berbobot G dan
simpul awal a
Keluaran: lintasan terpendek dari a ke semua simpul lainnya}

Deklarasi
   s1,s2,...,sn : interger {larik interger}
   d1,d2,...,dn : interger {larik interger}
   i : interger

Algoritma
   {Langkah 0 (inisialisasi) : }
   for i ← 1 to n do
      si ← 0
      di ← mai
   endfor

   {Langkah 1: }
   sa ← 1 {karena simpul a adalah simpul asal lintasan terpendek,
           jadi terpilih dalam lintasan terpendek}
   da ← ∞ {tidak ada lintasan terpendek dari simpul a ke a}

   {Langkah 2,3,...,n1:}
   for i ← 2 to n1 do
      Cari j sedemikian sehingga sj = 0
      dan dj = min {d1,d2,...,dn}
      Sj ← 1 {simpul j sudah terpilih ke
      dalam lintasan terpendek}
      perbarui di, untuk i = 1,2,3,...,n
      dengan : di (baru) = min {di(lama),
      dj + mji}
   endfor
```

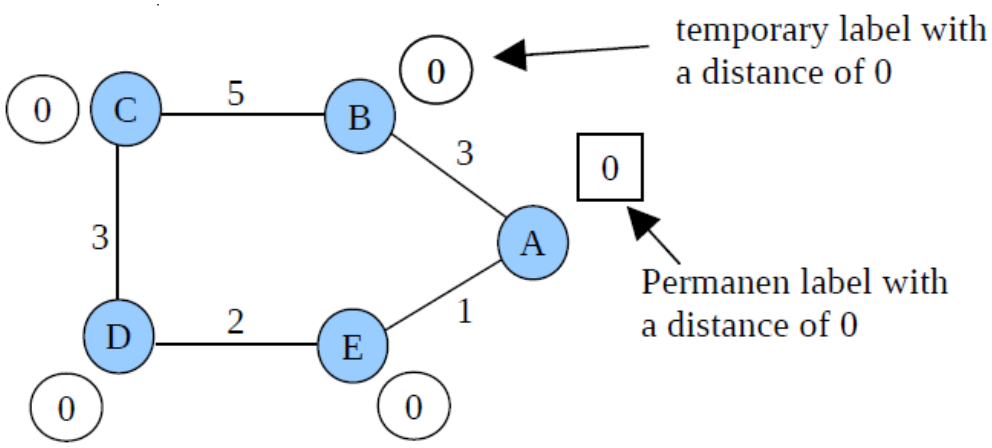
Algoritma Dijkstra membuat label yang menunjukkan simpul-simpul. Label-label ini melambangkan jarak dari simpul asal ke suatu simpul lain. Dalam graf, terdapat dua macam label, sementara dan permanen. Label sementara diberikan untuk simpul-simpul yang belum dicapai. Nilai yang diberikan untuk

label sementara ini dapat beragam. Label permanen diberikan untuk simpul-simpul yang sudah dicapai dan jarak ke simpul asal diketahui. Nilai yang diberikan untuk label ini ialah jarak dari simpul ke simpul asal. Suatu simpul pasti memiliki label permanen atau label sementara. Tetapi tidak keduanya.



Gambar 2. 5 Simpul dan Jaraknya

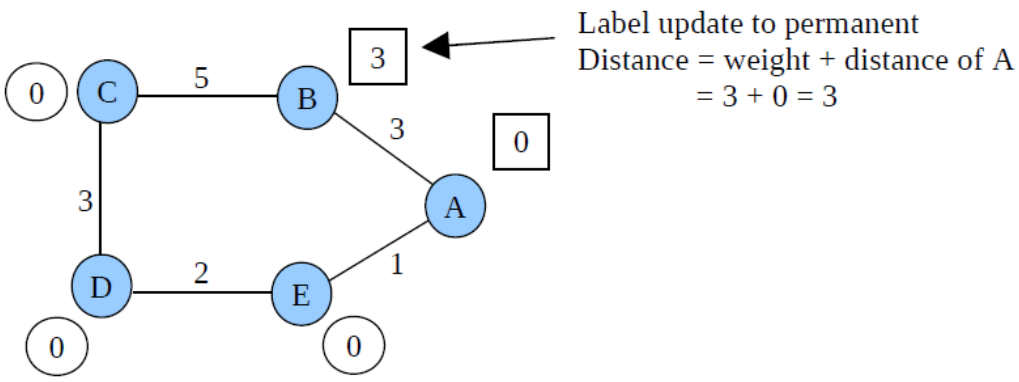
Pada Gambar 2.5 (a). Simpul A berlabel sementara dengan jarak 0, (b). Simpul B berlabel permanen dengan jarak 5. Algoritma dimulai dengan menginisialisasi simpul manapun di dalam graf (misalkan simpul A) dengan label permanen bernilai 0 dan simpul-simpul sisanya dengan label sementara bernilai 0.



Gambar 2. 6 Inisialisasi Awal Simpul

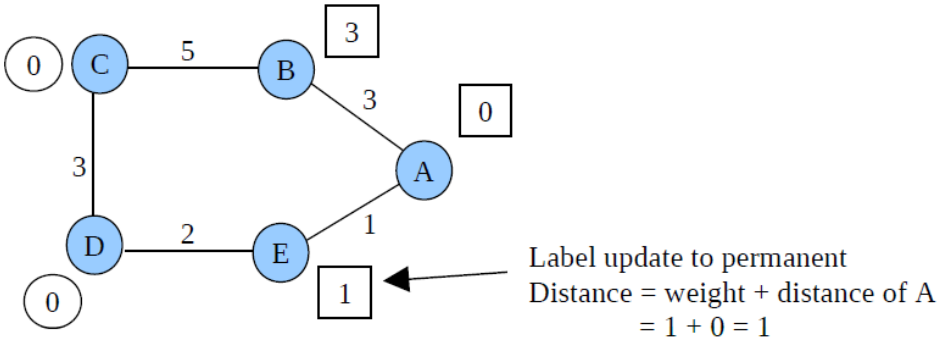
Algoritma ini kemudian memilih nilai sisi terendah yang menghubungkan simpul dengan label permanen (dalam hal ini simpul A) ke sebuah simpul lain yang berlabel sementara (misalkan simpul B). Kemudian label simpul B

diperbarui dari label sementara menjadi label permanen. Nilai simpul B merupakan penjumlahan nilai sisi dan nilai simpul A.



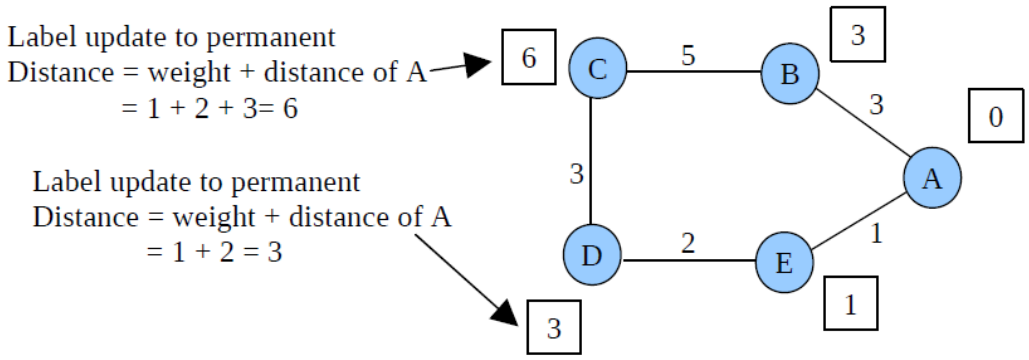
Gambar 2. 7 Nilai Simpul B Menjadi Permanen

Langkah selanjutnya ialah menemukan nilai sisi terendah dari simpul dengan label sementara, baik simpul A maupun simpul B (misalkan simpul E). Ubah label simpul E menjadi permanen, dan ukur jarak ke simpul A.



Gambar 2. 8 Nilai Simpul E Berubah

Proses ini berulang hingga semua label simpul menjadi permanen.



Gambar 2. 9 Semua Nilai Simpul Menjadi Permanen

Lamanya waktu untuk menjalankan algoritma Dijkstra pada suatu graf dengan E (himpunan sisi) dan V (himpunan simpul) dapat dinyatakan sebagai fungsi E dan V menggunakan notasi *Big-O*. Waktu yang dibutuhkan algoritma Dijkstra untuk bekerja ialah sebesar O(V*log V + E). Implementasi paling sederhana dari algoritma Dijkstra ialah penyimpanan simpul dari suatu himpunan ke dalam suatu *array* atau *list* berkait.

1. 2. 8 Metode *Haversine* [19]

Metode *haversine* merupakan salah satu metode yang dapat digunakan untuk memberikan informasi kepada pengguna tentang perhitungan jarak terdekat ke tempat tujuan tertentu dengan posisi pengguna saat mengakses, sehingga pengguna dapat mengetahui jumlah jarak dari suatu tempat ke tempat tujuan atau jarak yang paling dekat.

Proses kalkulasi jarak membutuhkan 4 masukan antara lain titik koordinat *latitude* dan *longitude* tempat asal serta titik koordinat *latitude* dan *longitude* tempat tujuan. Untuk perhitungannya menggunakan metode *Haversine*, dimana antara kedua titik ditarik garis lurus terdekat.

Haversine formula:
$$a = \sin^2(\Delta lat/2) + \cos(lat_1).\cos(lat_2).\sin^2(\Delta long/2)$$
$$c = 2.\text{atan2}(\sqrt{a}, \sqrt{(1-a)})$$
$$d = R.c$$

Dimana *R* adalah jari-jari bumi (6,371 km atau 3959 mil). Metode *haversine* masih sangat baik digunakan untuk perhitungan numerik bahkan pada jarak terpendek, tidak seperti perhitungan berdasarkan hukum bulat dari *cosinus*. Nilai *c* adalah jarak sudut dalam radian dan merupakan kuadrat dari setengah panjang *chord* antara titik-titik.

Pada proses penentuan tempat-tempat terdekat, dilakukan kalkulasi dengan membentuk sebuah lingkaran yang titik pusatnya merupakan titik koordinat pengguna. Berikut formula yang digunakan untuk membuat lingkaran.

Haversine formula = *R* * acos(cos(radians(37)) * cos(radians(lat)) * cos(radians (lng) - radians(-122)) + sin(radians(37)) * sin(radians(lat)))

Dimana *radians* merupakan radius antara 25-37 mil dari lokasi pengguna dan *R* adalah jari-jari bumi (6,371 km atau 3959 mil).

1. 2. 9 NET Framework [5]

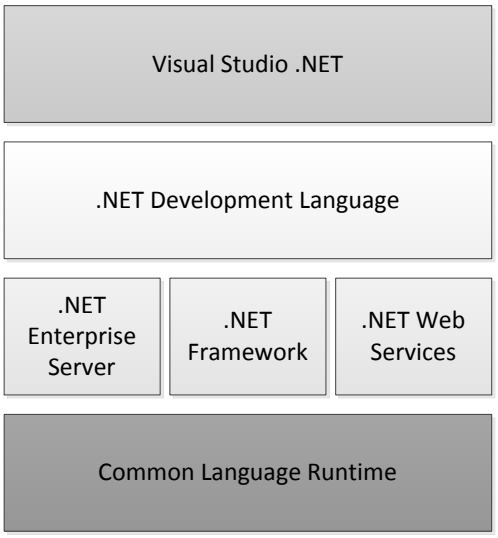
.NET adalah sebuah *platform* yang berisi sekumpulan *tool* dan layanan yang akan memberi makna yang berbeda bagi tiap orang. Bagi seorang pengembang .NET, *platform* ibarat sebagai sebuah sekumpulan API dan *tool* pengembangan aplikasi terkini, yakni Visual Studio .NET. Sementara bagi seorang administrator jaringan *platform* .NET dapat diibaratkan sebagai perangkat lunak yang memudahkan baginya dalam mengelola *server* seperti halnya sekumpulan solusi peranti lunak yang dikenal dengan .NET Server Family.

.NET merupakan sesuatu yang luas, tidak hanya menjelaskan sekumpulan komponen siap pakai yang memudahkan seorang pengembang menulis kode untuk aplikasinya. .NET menghadirkan sebuah cara pandang baru tentang bagaimana aplikasi didesain, dikembangkan, di-*deploy*, dan digunakan oleh pengguna.

.NET memiliki sebuah arsitektur berlapis yang secara sederhana dapat terbagi menjadi beberapa bagian berdasarkan fungsionalnya yakni:

- Sekumpulan peranti pengembangan aplikasi, seperti Visual Studio .NET, .NET Framework, .NET Compact Framework, hingga dukungan multibahasa pemrogaman.
- Sekumpulan aplikasi *server* yang dikenal dengan .NET Enterprise Server. Di dalamnya terdapat sekumpulan peranti lunak yang mendukung infrastruktur bisnis dengan dukungan layanan *web*, seperti SQL Server, BizTalk Server dan sebagainya.
- Sekumpulan layanan bisnis terdistribusi yang dikemas dan didistribusikan melalui layanan *web* baik secara komersial maupun nonkomersial seperti yang sering digunakan yakni .NET Passport.
- Sekumpulan peranti yang mendukung .NET mulai dari *dekstop* PC, *tablet* PC, Windows Phone hingga *server*.

Untuk lebih jelasnya Gambar 2.10 memberikan gambaran mengenai arsitektur komponen .NET Platform.

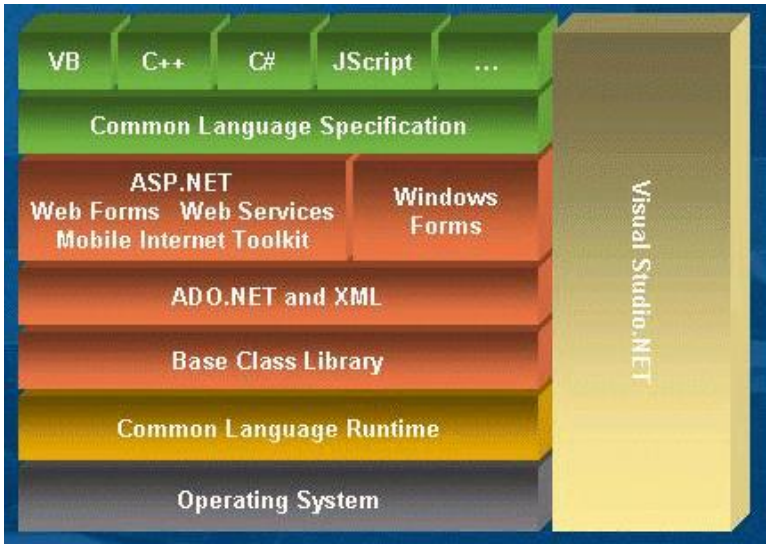


Gambar 2. 10 .NET *Platform Component*

Seperti pada Gambar 2.10 .NET Platform tidak sekedar lingkungan pengembangan sistem. Dari sudut pandang pengembang, maka akan lebih terfokus pada .NET Framework, .NET Development Language dan Visual Studio.

Secara sederhana .NET Framework adalah sebuah bagian dari .NET Platform yang merupakan model pemrogaman untuk mengembangkan, mendistribusikan dan menjalankan aplikasi layanan *web* beserta aplikasi lain seperti aplikasi *dekstop* dan *web*. .NET Framework menyediakan suatu lingkungan eksekusi yang berproduktivitas tinggi dalam pengembangan serta dirancangan untuk memadukan investasi terhadap aplikasi yang sudah ada ataupun yang akan datang.

.NET Framework sendiri bukanlah sesuatu yang sempit secara garis besar .NET Framework terbagi menjadi dua bagian yakni CLR (Common Library Runtime) dan .NET Framework Class Library. Namun demikian, arsitekturnya terbagi-bagi menjadi beberapa *layer* seperti yang digambarkan Gambar 2.11.



Gambar 2. 11 .NET Framework dan Visual Studio .NET

Pada Gambar 2.11 komponen-komponen dari .NET Framework adalah:

- Common Language Runtime sebagai fondasi dasar dari .NET Framework yang akan mengeksekusi aplikasi berbasis .NET.
- ADO.NET dan XML memberikan suatu kemudahan dalam hal pengaksesan data dari berbagai macam sumber data mulai dari DBMS hingga *plain text* berformat XML.
- ASP.NET memberikan suatu infrastruktur mendasar untuk pengembangan aplikasi *web form* untuk pengembangan antarmuka situs *web* secara umum ataupun layanan *web*.
- Windows Form sebagai infrastruktur pengembangan aplikasi berbasis *dekstop*.
- Base Case Library sebagai kumpulan *class-class* dasar yang berperan menyokong fungsionalitas .NET Framework seperti *class-class* yang menangani keamanan aplikasi, konfigurasi aplikasi dan sebagainya.
- CLS (Common Language Specification) berperan sebagai jembatan penghubung antara .NET Framework dengan .NET Supported Language. CLS

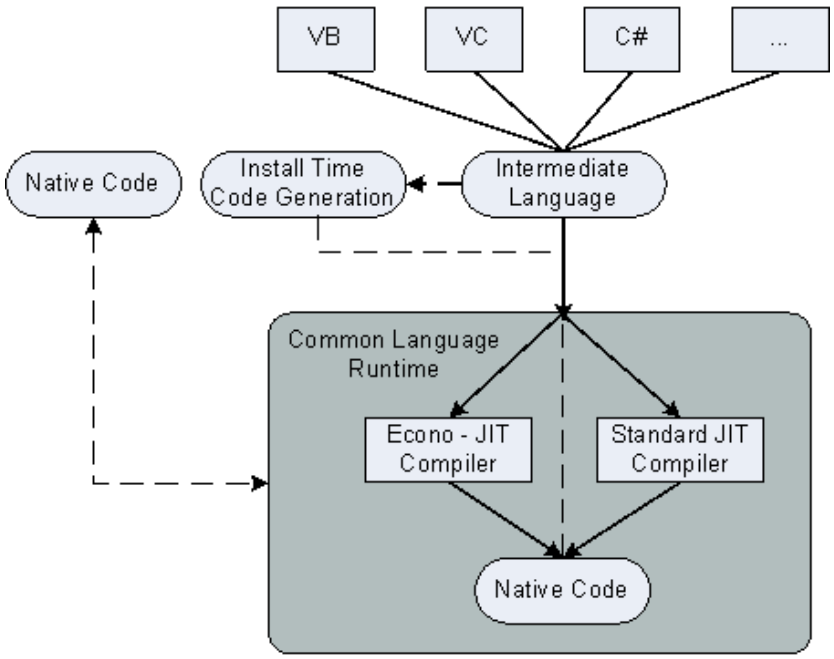
inilah yang memungkinkan suatu aplikasi berbasis .NET dapat dibangun dengan lebih dari satu bahasa pemrogaman. Kuncinya adalah terdapat suatu spesifikasi standar bagaimana suatu bahasa pemrogaman dapat menjadi .NET Supported Language sebagai contoh CLS diimplementasikan standarisasi tipe data yang dikenal sebagai dengan CTS (Common Type System) yang merupakan bagian dari CLR dan berisi.

- .NET Supported Language berupa sekumpulan bahasa yang mendukung pengembangan aplikasi dengan .NET Framework. Pada .NET Framework telah ter-*embed* beberapa bahasa *official* yang dikembangkan oleh Microsoft seperti C#, VB, C+ dan Jscript.

Pada Gambar 2.11 juga digambarkan *block* Visual Studio .NET. Visual Studio .NET bukan merupakan bagian dari .NET Framework, hanya berupa IDE (Integrated Development Environtment) yang membantu pengembangan agar lebih mudah mengembangkan aplikasi. Namun, demikian keberadaanya akan berperan penting dalam situasi nyata pengembangan aplikasi.

Pusat dari sebuah .NET Framework sebenarnya terletak pada CLR. Aplikasi yang dikembangkan oleh .NET Framework akan di-*compile* menjadi sebuah kode khusus yang dikenal dengan MSIL (Microsoft Intermediate Language). Keuntungan dari *managed code* adalah kemampuan integrasi suatu aplikasi tanpa memandang batasan bahasa pemrogaman. Sebagai contoh, sebuah aplikasi yang dibuat oleh C# dapat digunakan secara mudah oleh *class* yang dibuat oleh *class* yang dibuat dengan bahasa C++. *Class* C++ dapat dengan mudah melakukan konsumsi objek atau melakukan derivasi dari *class* C# tersebut.

Tugas utama dari CLR adalah melakukan konversi kode MSIL yang dihasilkan oleh *compiler* masing-masing bahasa pemrograman menjadi kode mesin yang siap dieksekusi. Untuk mengetahui bagaimana sebuah aplikasi .NET dieksekusi, perhatikan Gambar 2.12.



Gambar 2. 12 Model Eksekusi Aplikasi .NET

Sebagai jantung .NET Framework, CLR melakukan beragam fungsi selain melakukan konversi kode seperti:

- Melakukan validasi pengamanan kode dimana sebuah aplikasi yang teresekusi harus dapat berjalan dengan benar dan tentunya mengakses sumber daya yang seharusnya.
- Pengelolaan *Memory* dan *Garbage Collection* yang memberikan kemudahan bagi pengembangan dalam mengalokasikan sumber daya tanpa harus khawatir atau peduli secara penuh terhadap masa hidup sebuah objek. Konsep ini sangat

membantu sistem operasi dalam mengurangi *memory leak* pada sistem operasi.

- Konsisten dalam penanganan kesalahan. Pada saat terjadi kesalahan CLR telah memiliki sekumpulan eksepsi yang memungkinkan sistem aplikasi yang dieksekusi dapat dengan mudah untuk *recover* atau keluar dengan keadaan sebagaimana mestinya tanpa melibatkan tingkat kesalahan di tingkat sistem operasi. Konsep ini sangat aman dari sisi sistem operasi karena kapanpun terjadi kesalahan, sistem operasi sama sekali tidak terpengaruh.
- Memungkinkan pengembang aplikasi dengan multibahasa, sebagaimana yang telah dikemukakan bahwa sangat sedikit alasan bagi sesama bahasa yang mendukung *class* C++ dikompilasi menjadi sebuah *library* yang tak lain adalah sekumpulan kode MSIL akan sangat mudah bagi C# untuk melakukan pewarisan atau penggunaan objeknya secara bebas.
- Dukungan *multiplatform* .NET tidak hanya hadir di berbagai peralatan *mobile* seperti *dekstop* PC hingga *tablet* PC. Dengan adanya CLR dimungkinkan pula aplikasi .NET dapat berjalan di sistem operasi yang berbeda seperti Linux, MacOS, ataupun FreeBSD. Banyak implementasi pihak ketiga yang memungkinkan .NET berjalan di banyak sistem operasi seperti SSCLI, Mono atau .NET *portable*.
- Akhir dari DLL Hell, keadaan yang dikenal dengan DLL Hell, kerap terjadi pada saat sistem baru dipasang dan beberapa *library shared* yang digunakan dan menimpa sistem lama. Akibatnya ada kemungkinan sistem yang lama bergantung pada *library* tersebut berjalan tidak sebagaimana mestinya. CLR

mengadopsi konsep yang dimiliki oleh Windows 2000 dengan konsep *side by side execution*. CLR memungkinkan dua buah aplikasi yang berbeda dan memiliki referensi *library* yang sama, namun berbeda versi, dapat berjalan sesuai dengan versi yang dibutuhkan untuk tipa-tiap aplikasi tanpa adanya saling konflik versi atapun masalah pada saat registrasi *library*. Hal ini dikarenakan konsep distribusi aplikasi .NET yang dikenal dengan *assembly* bersifat *self described*. Dengan kata lain, tidak membutuhkan registrasi pada sisi sistem operasi.

Selain CLR, .NET Framework Class Library (FCL) juga berperan penting terutama bagi pengembang. Di dalamnya terdapat sekumpulan komponen siap pakai mulai dari komponen antarmuka pengguna untuk *web* dan *desktop*, komponen untuk melakukan akses *database*, hingga komponen untuk mengakses sumber daya sistem operasi lokal maupun melalui internet.

Hal yang menarik dari FCL adalah susunannya yang diatur oleh konsep *namespace*. Konsep *namespace* menekankan suatu hirarki penyusunan kumpulan *class* secara logis berdasarkan fungsionalitas *class*. Sebagai contoh, *namespace* System.IO adalah suatu *namespace* yang berisi sekumpulan *class* untuk melakukan akses IO. Satu hal yang patut kita tekankan adalah *namespace* bersifat logis tidak bersifat fisik, artinya sekumpulan *class* yang memiliki *namespace* System.IO tidak harus berada di direktori fisik komputer yang terletak di ...//System/IO. Konsep ini tentunya mengurangi kompleksitas dalam kompilasi maupun susunan hierarki direktori pengembangan sistem.

1. 2. 10 C# [5]

Bahasa ini pertama kali hadir di PDC (Professional Developer Conference) Orlando sebagai salah satu *official language* untuk *framework* Microsoft terbaru .NET. Banyak pihak pengembang yang tidak heran akan kehadiran bahasa terbaru C#. Hal ini sebenarnya sudah terdengar isu dari pihak Microsoft yang mengembangkan bahasa dengan *codename* "*Cool*" dan mungkin C# adalah wujud dari "*Cool*".

Pada bulan September 2000, C# distandarisasi sebagai sebuah bahasa pemrogaman standar oleh badan standarisasi ECMA. Hal ini makin tercantum pada standar ECMA-334, yang di dalamnya juga menyatakan tujuan bahasa C# yakni sebuah bahasa pemrogaman yang "sederhana, modern, *general purpose* dan berorientasi objek".

Dari spesifikasi bahasa C# yang ditulis oleh Anders Hejlsberg dan Scott Wiltamuth, tersirat bahwa C# adalah bahasa pemrogaman yang hadir untuk menjembatani pengembangan aplikasi yang andal, sederhana dan tentunya mempunya performa yang memadai.

Terlepas dari asumsi beberapa pihak. C# dapat dikatakan sebagai pengembangan bercabang dari bahasa C++. Sama seperti C++ yang berasal dari rumpun bahasa C. Baik C maupun C++ adalah bahasa yang powerful dan berperan sebagai bahasa pemrogaman yang bersifat umum mulai dari UNIX, hingga Windows mulai dari *desktop* hingga *mobile*.

C# adalah bahasa modern yang hadir dengan konsep OOP dan mengesampingkan konsep sulit seperti *pointer*, *multiple inheritance* dan juga

alokasi sumber daya. Hal ini dilakukan dengan menghadirkan suatu *framework* khusus diatas sistem operasi yang pada C# lingkungan eksekusi aplikasi ini dikenal dengan .NET CLR. Di dalam .NET CLR ini berlangsung eksekusi, pengeloloaan sumber daya, hingga penanganan *error* secara otomatis.

C# hadir memiliki tujuan yang hampir sama seperti saat C++ muncul. C++ hadir untuk menerapkan pengembangan perangkat lunak berorientasi objek tanpa mengabaikan keterampilan programmer sebelumnya pada C. Walaupun pada kenyataannya C++ tidaklah se-OOP bahasa *Eiffel*, tetapi sebagian besar fitur C++ berbasis konsep OOP. Setelah ditelaah lebih lanjut, ternyata beberapa fitur yang tidak OOP pada C++ dimaksudkan untuk menjaga kompatibilitasnya dengan bahasa C. Bahasa C# juga melakukan hal yang sama, yaitu dengan berdasarkan pada .NET Framework. Bahasa C# hadir untuk menerapkan pengembangan perangkat lunak berorientasi objek tanpa mengabaikan investasi terdahulu pada C++, C, maupun COM.

Walaupun bersifat generik, pada desain dan tujuannya, C# juga dikenal sebagai bahasa yang "*component centric*" artinya objek-objek bahasa C# dikembangkan dari sekumpulan komponen dan setiap aksinya juga berdasarkan objek dari komponen tersebut. Sehingga diharapkan pengembangan pada C# menjadi lebih mudah.

Pada C# konsep komponen seperti properti, metode dan *event* adalah dasar dari pengembangan sistem. Sekumpulan komponen milik C# semuanya terdapat dalam *runtime* khusus pada .NET Framework. Hal ini sedikit berbeda dengan VB.NET atau C++. C# adalah satu-satunya bahasa pemrogaman yang tidak

memiliki pustaka MFC, VB.NET masih dapat mengakses fitur VB6. Hal ini semata-mata dilakukan untuk menjaga konsistensi dan kesederhanaan bahasa C# sebagai bahasa yang "*pure* .NET" dan juga berorientasi objek.

1. 2. 11 *Web Service* [2]

Web Service adalah sekumpulan *application logic* beserta objek-objek dan metode-metode yang dimilikinya yang terletak di suatu *server* yang terhubung ke internet sehingga dapat diakses menggunakan protocol HTTP dan SOAP (Simple Object Access Protocol). Dalam penggunaannya, *web service* dapat digunakan dari hanya untuk memeriksa data pengguna yang *login* ke sebuah *web* ataupun untuk digunakan pada transaksi perbankan *online* yang rumit.

Tujuan dari teknologi ini adalah untuk memudahkan beberapa aplikasi atau komponennya untuk saling berhubungan dengan aplikasi lain dalam sebuah organisasi maupun diluar organisasi menggunakan standar yang tidak terikat *platform* (*platform-neutral*) dan tidak terikat akan bahasa pemrograman yang digunakan (*language-neutral*). Hal tersebut dapat terjadi karena penggunaan XML standar yang didukung oleh banyak perusahaan besar di dunia, yang digunakan untuk bertukar data. Selain daripada itu, penggunaan SOAP menjadikan metode-metode dari objek-objek yang ada dalam sebuah *web service* dapat di akses dari aplikasi lain seperti halnya aplikasi tersebut mengakses metode lokal.

2.2.9.1 Web Services Messaging [11]

W3C mendefinisikan *web service* sebagai "sebuah software aplikasi yang dapat teridentifikasi oleh URI dan memiliki *interface* yang didefiniskan, dideskripsikan, dan dimengerti oleh XML dan juga mendukung interaksi langsung

dengan perangkat lunak aplikasi yang lain dengan menggunakan *message* berbasis XML melalui protokol internet".

Web service adalah sebuah perangkat lunak aplikasi yang tidak terpengaruh oleh *platform*, ia akan menyediakan metode-metode yang dapat diakses oleh jaringan. Ia juga akan menggunakan XML untuk pertukaran data, khususnya pada dua entitas bisnis yang berbeda.

Beberapa karakteristik dari *web service* adalah:

1. *Message-based*
2. *Standards-based*
3. *Programming language independent*
4. *Platform-neutral*

Beberapa *key* standar didalam *web service* adalah:

1. XML
2. SOAP
3. WSDL
4. UDDI

SOAP (Simple Object Access Protocol) adalah sebuah XML-*based mark-up language* untuk pergantian pesan diantara aplikasi-aplikasi. SOAP berguna seperti sebuah amplop yang digunakan untuk pertukaran data objek didalam jaringan. SOAP mendefinisikan empat aspek didalam komunikasi: *Message envelope*, *Encoding*, RPC *call convention*, dan bagaimana menyatukan sebuah *message* didalam protokol *transport*.

Sebuah SOAP *message* terdiri dari SOAP *Envelop* dan bisa terdiri dari berberapa *attachment* atau tidak memiliki *attachment*. SOAP *envelop* tersusun dari SOAP *header* dan SOAP *body*, sedangkan SOAP *attachment* membolehkan non-XML data untuk dimasukkan ke dalam SOAP *message*, di-*encoded*, dan diletakkan kedalam SOAP *message* dengan menggunakan MIME-*multipart*.

2.2.9.2 *Web Services Description* [11]

WSDL (Web Services Description Language) adalah sebuah XML-*based language* untuk mendeskripsikan XML. Ia menyediakan layanan yang mendeskripsikan *service request* dengan menggunakan protokol-protokol yang berbeda dan juga *encoding*. Ia akan memfasilitasi komunikasi antar aplikasi. WSDL akan mendeskripsikan apa yang akan dilakukan oleh *web service*, bagaimana menemukannya dan bagaimana untuk mengoperasikannya.

Spesifikasi WSDL mendefinisikan tujuh tipe elemen:

1. *Types* - elemen untuk mendefinisikan tipe data. *Types* akan mendefinisikan tipe data (seperti *string* atau *integer*) dari elemen didalam sebuah *message*.
2. *Message* - *abstract*, pendefinisian tipe data yang akan dikomunikasikan.
3. *Operation* - sebuah deskripsi abstrak dari sebuah *action* yang didukung oleh *service*.
4. *Port Type* - sebuah koleksi abstrak dari operasi yang didukung oleh lebih dari satu *endpoint*.
5. *Binding* - mendefinisikan penyatuan dari tipe *port* (koleksi dari operasi-operasi) menjadi sebuah protokol *transport* dan data format (contohnya SOAP

1.1 pada HTTP). Ini adalah sebuah protokol konkret dan sebuah spesifikasi data format didalam tipe *port* tertentu.

6. *Port* - mendefinisikan sebuah komunikasi *endpoint* sebagai kombinasi dari *binding* dan alamat jaringan. Bagi protokol HTTP, ini adalah sebuah bentuk dari URL sedangkan bagi protokol SMTP, ini adalah sebuah *form* dari alamat *email*.
7. *Service* – satu set *port* yang terkorelasi atau suatu *endpoint*.

2.2.9.3 *Web Service Discovery* [11]

UDDI (*Universal Description, Discovery and Integration*) adalah sebuah *service registry* bagi pengalokasian *web service*. UDDI mengkombinasikan SOAP dan WSDL untuk pembentukan sebuah *registry* API bagi pendaftaran dan pengenalan *service*. Ia menyediakan sebuah area umum dimana sebuah organisasi dapat mengiklankan keberadaan mereka dan layanan yang mereka berikan (*web service*).

UDDI adalah sebuah *framework* yang mendefinisikan sebuah XML-*based registry* dimana sebuah organisasi dapat meng-*upload* informasi mengenai layanan yang mereka berikan. XML-*based registry* berisi nama-nama dari organisasi tersebut, beserta layanan dan deskripsi dari layanan yang mereka berikan.

1. 2. 12 JSON [12]

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari

bahasa pemprograman JavaScript, *Standart* ECMA-262 edisi ke-3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

JSON terbuat dari dua struktur:

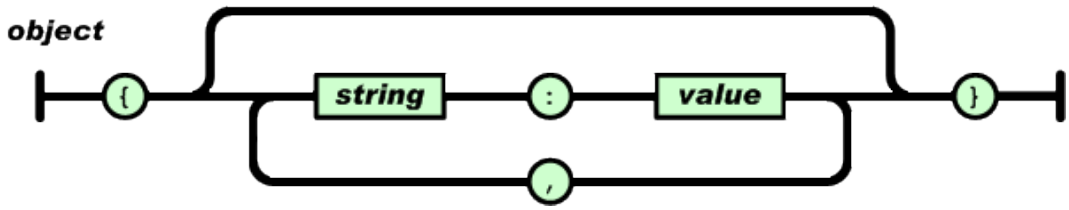
1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau larik assosiatif (*associative array).*
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

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

1. Objek

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

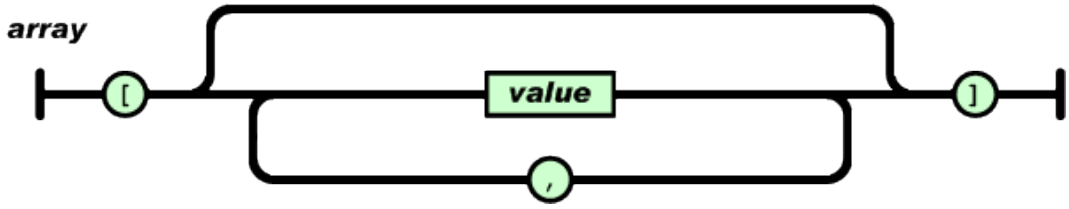
Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).



Gambar 2. 13 Objek JSON

3. Larik

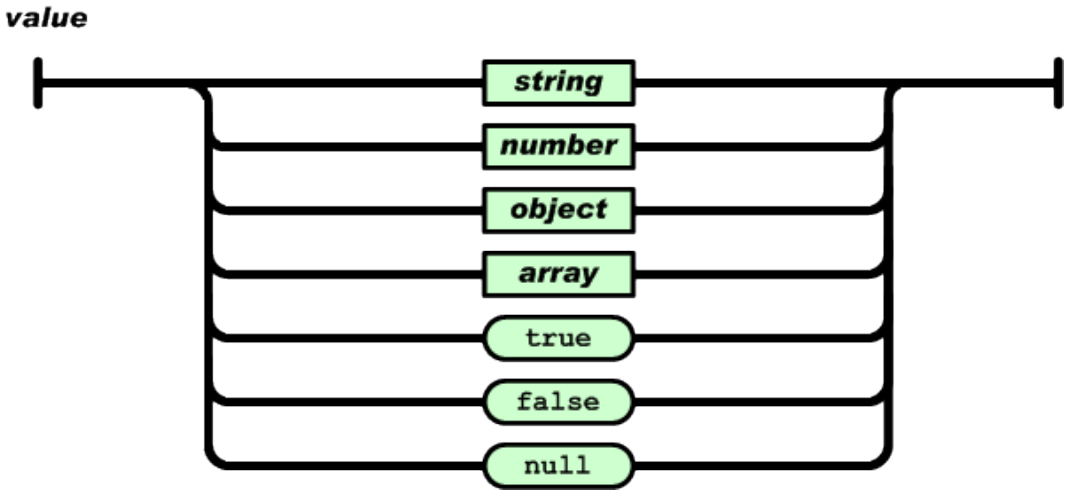
Larik (*array*) adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).



Gambar 2. 14 Larik JSON

4. Nilai

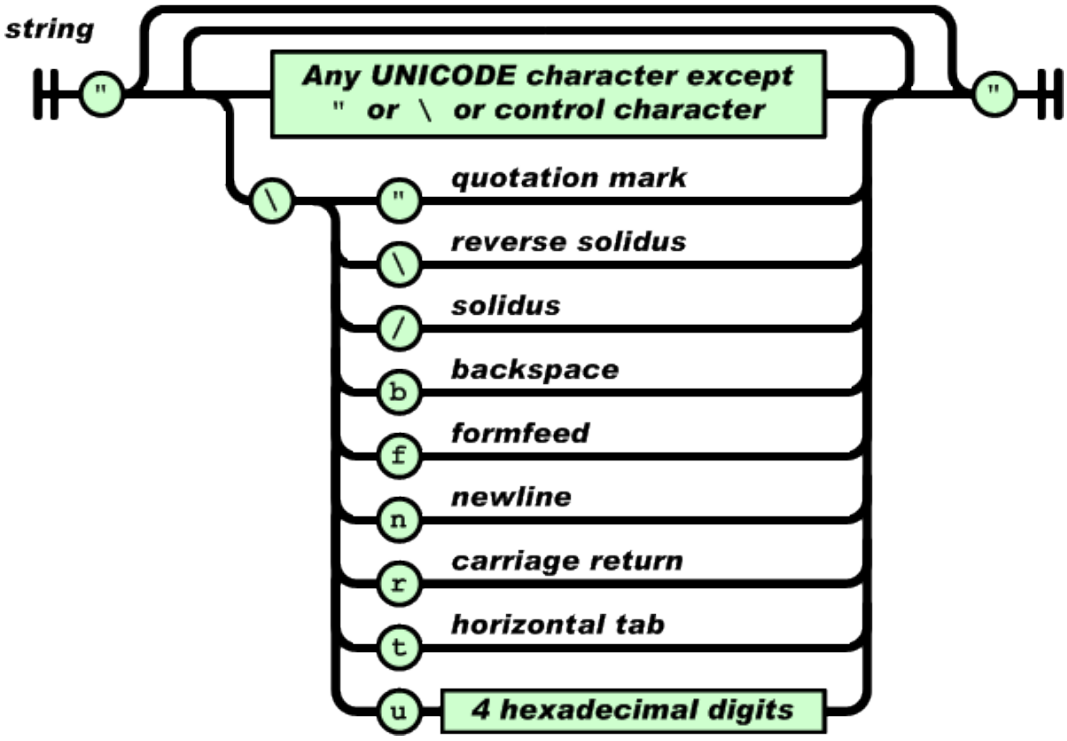
Nilai (*value*) dapat berupa sebuah *string* dalam tanda kutip ganda, atau angka, atau *true* atau *false* atau *null*, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.



Gambar 2. 15 Nilai JSON

5. *String*

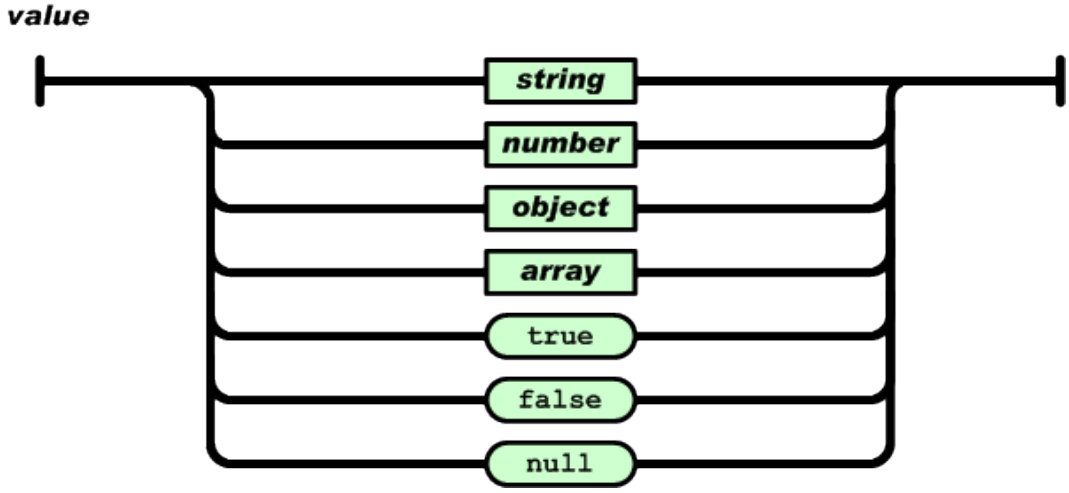
String adalah kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam *string* dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada *string*. *String* sangat mirip dengan *string* C atau Java.



Gambar 2. 16 *String* JSON

6. Angka

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



Gambar 2. 17 Angka JSON

1. 2. 13 PHP [15]

PHP diciptakan oleh Rasmus Lerdorf pada tahun 1994 dan bersifat *open source*. Sampai bulan Januari 2007, PHP sudah digunakan oleh kurang lebih 20 juta domain dan terus berkembang sampai saat ini. PHP merupakan singkatan dari *Hypertext Preprocessor*, adalah sebuah bahasa scripting yang terpasang pada HTML. Sebagian besar sintaks mirip dengan bahasa C, Java dan Perl, ditambah beberapa fungsi PHP yang spesifik. Tujuan utama bahasa ini adalah untuk memungkinkan perancang *web* menulis halaman *web* dinamis dengan cepat. PHP merupakan bahasa pemograman *web* yang bersifat *server-side* HTML atau *embedded scripting*, di mana skripnya menyatu dengan HTML dan berada di *server* artinya sintaks dan perintah-perintah yang kita berikan akan sepenuhnya

dijalankan di *server* tetapi disertakan HTML biasa. PHP dikenal sebagai bahasa *scripting* yang menyatu dengan tag HTML, dieksekusi di *server* dan digunakan untuk membuat halaman *web* yang dinamis seperti ASP (*Active Server Pages*) dan JSP (*Java Server Pages*).

Seluruh aplikasi berbasis *web* dapat dibuat dengan PHP. Namun kekuatan yang paling utama PHP adalah pada konektivitasnya dengan sistem *database* di dalam *web*. Sistem database yang dapat didukung oleh PHP adalah: Oracle, MySQL, Sybase, PostgreSQL.

PHP dapat berjalan di berbagai sistem operasi seperti Windows 98/NT, UNIX/LINUX, Solaris maupun Macintosh. PHP merupakan perangkat lunak yang *open source* yang dapat diunduh secara gratis.

Perangkat lunak ini juga dapat berjalan pada *web server* seperti PWS (*Personal Web Server*), Apache, IIS, AOLServer, fhttpd, phttpd dan sebagainya. PHP juga merupakan bahasa pemograman yang dapat kita kembangkan sendiri seperti menambah fungsi-fungsi baru. Keunggulan lainnya dari PHP adalah PHP juga mendukung komunikasi dengan layanan seperti *protocol* IMAP, SNMP, NNTP, POP3 bahkan HTTP. PHP dapat diinstal sebagai bagian atau modul dari Apache *web server* atau sebagai CGI *script* yang mandiri.

Banyak keuntungan yang dapat diperoleh jika menggunakan PHP sebagai modul dari Apache diantaranya adalah:

7. Tingkat keamanan yang cukup tinggi
8. Waktu eksekusi yang lebih cepat dibandingkan dengan bahasa pemograman *web* lainnya yang berorientasi pada *server-side scripting*.

9. Akses ke sistem basis data yang lebih fleksibel seperti MySQL.

2.2.11.1 CodeIgniter [7]

CodeIgniter adalah sebuah *framework* PHP yang dapat membantu mempercepat developer dalam pengembangan aplikasi *web* berbasis PHP dibanding jika menulis semua kode program dari awal.

CodeIgniter pertama kali dibuat oleh Rick Ellis, CEO Ellislab, Inc. sebuah perusahaan yang memproduksi CMS (*Content Management System*) yang cukup handal, yaitu Expression Engine. Saat ini, CodeIgniter dikembangkan oleh Expression Engine Development Team.

Adapun beberapa keuntungan menggunakan CodeIgniter, diantaranya:

1. Gratis

   CodeIgniter berlisensi dibawah Apache/BSD opensorce.

2. Ditulis Menggunakan PHP 4

   Meskipun CodeIgniter dapat berjalan di PHP 5, namun sampai saat ini kode program CodeIgniter masih dibuat dengan menggunakan PHP 4.

3. Berukuran Kecil

   Ukuran CodeIgniter yang kecil merupakan keunggulan tersendiri. Dibanding dengan framework lain yang berukuran besar.

4. Menggunakan Konsep MVC

   CodeIgniter menggunakan konsep MVC yang memungkinkan pemisahan *layer application-logic* dan *presentation*.

5. URL yang Sederhana

Secara *default*, URL yang dihasilkan CodeIgniter sangat bersih dan *Search Engine Friendly* (SEF).

6. Memiliki Paket *Library* yang Lengkap

CodeIgniter mempunyai *library* yang lengkap untuk mengerjakan operasi-operasi yang umum dibutuhkan oleh sebuah aplikasi berbasis *web*, misalnya mengakses basis data, mengirim *email*, memvalidasi *form*, menangani *session* dan sebagainya.

7. *Extensible*

Sistem dapat dikembangkan dengan mudah menggunakan *plug-in* dan *helper*, atau dengan menggunakan *hooks*.

8. Tidak Memerlukan *Template Engine*

Meskipun CodeIgniter dilengkapi dengan *template parser* sederhana yang dapat digunakan, tetapi hal ini tidak mengharuskan kita untuk menggunakannya.

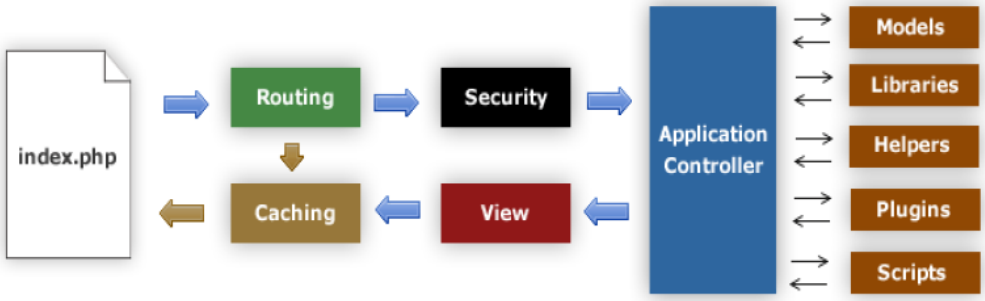
9. Dokumentasi Lengkap dan Jelas

Dari sekian banyak *framework*, CodeIgniter adalah satu-satunya *framework* dengan dokumentasi yang lengkap dan jelas.

10. Komunitas

Komunitas CodeIgniter saat ini berkembang pesat.

Proses aliran data aplikasi pada sistem dapat diilustrasikan seperti terlihat pada Gambar 2.18.



Gambar 2. 18 *Application Flowchart* CodeIgniter

Keterangan:

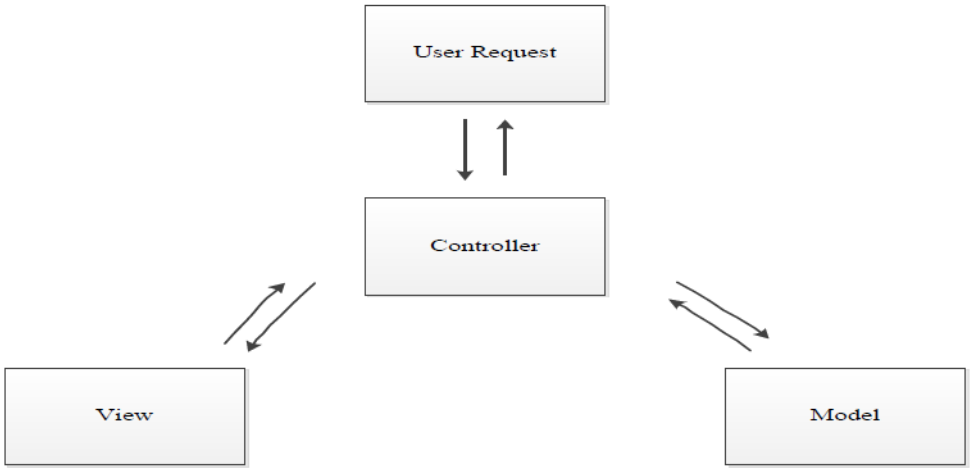
1. Index.php berfungsi sebagai *front controller*, menginisialisasi *base resource* untuk menjalankan CodeIgniter.
2. *Router* memeriksa HTTP *request* untuk menentukan apa yang harus dilakukan dengannya.
3. Jika *Cache* aktif, maka hasilnya akan langsung dikirimkan ke *browser* dengan mengabaikan aliran data normal.
4. *Security*. Sebelum *Controller* dimuat, HTTP *request* dan data yang dikirimkan pengguna akan disaring untuk keamanan.
5. *Controller* memuat model, *core libraries*, *plug-ins*, *helpers* dan semua *resource* yang diperlukan untuk memproses *request*.
6. Akhirnya *View* yang dihasilkan akan dikirimkan ke *browser*. Jika *Cache* aktif, maka *View* akan disimpan sebagai *Cache* dahulu, sehingga pada *request* berikutnya langsung dapat ditampilkan.

MVC (*Model-View-Controller*)

CodeIgniter adalah *framework* PHP yang dibuat berdasarkan kaidah *model-view-controller*. Dengan MVC, maka memungkinkan pemisahan antara

layer application-logic dan *presentation*. Sehingga, dalam sebuah pengembangan *web*, seorang pengembang bisa berkonsentrasi pada *core-system*, sedangkan *web designer* bisa berkonsentrasi pada tampilan *web*. Menariknya, skrip PHP, *query* MySQL, Javascript dan CSS bisa saling terpisah, tidak dibuat dalam satu skrip berukuran besar yang membutuhkan *resource* besar pula untuk mengesekusinya.

Adapun alur program aplikasi berbasis *framework* CodeIgniter dapat dilihat pada Gambar 2.19.



Gambar 2. 19 Model-View-Controller

Gambar 2.19 menerangkan bahwa ketika datang sebuah *user request*, maka akan ditangani oleh *controller*, kemudian *controller* akan memanggil *model* jika memang diperlukan operasi basis data. Hasil dari *query* oleh *model* kemudian akan dikembalikan ke *controller*. Selanjutnya *controller* akan memanggil *view* yang tepat dan mengombinasikannya dengan hasil *query model*. Hasil akhir dari operasi ini akan ditampilkan di *browser*.

Dalam konteks CodeIgniter dan aplikasi berbasis *web*, maka penerapan konsep MVC mengakibatkan kode program dapat dibagi menjadi tiga kategori, yaitu :

1. Model

Kode program (berupa OOP *class*) yang digunakan untuk memanipulasi basis data.

2. View

Berupa template html/xml atau php untuk menampilkan data pada *browser.*

3. Controller

Kode program (berupa OOP *class*) yang digunakan untuk mengontrol aliran aplikasi (sebagai pengontrol *model* dan *view*)

1. 2. 14 *Application Programming Language* (API)

Antarmuka pemrograman aplikasi (*Application Programming Interface*) adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan oleh pengembang saat membangung perangkat lunak untuk sistem operasi tertentu. API memungkinkan pengembang untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi.

2.2.12.1 Bandung Review API

Dalam rangka pengembangan layanan Bandung Review agar dapat berjalan pada *multi-platform*, dengan ini PT. Bandung Review mengenalkan Bandung Review API untuk memungkinkan suatu aplikasi melakukan interaksi dengan BandungReview.com. Bandung Review API merupakan sarana berbagi

data antara suatu aplikasi yang telah didaftarkan. Dengan layanan ini, pengembang aplikasi baik itu *web*, *desktop* ataupun *mobile* dapat membuat sendiri antarmuka Bandung Review berdasarkan *platform* masing-masing dengan data yang sama pada *server* utama BandungReview.com.

2.2.12.2 foursquare API [6]

foursquare merupakan situs jejaring sosial dan aplikasi *mobile* yang mungkinkan pengguna untuk berinteraksi dengan teman pengguna dan secara langsung melakukan *check-in* di suatu lokasi. Layanan ini juga tersedia bagi pengguna *smartphone* yang dilengkapi dengan GPS (*Global Positioning System*). *Check-in* di dalam foursquare adalah ketika pengguna memberi tahu foursquare dimana posisi pengguna berada. *Check-in* bisa dilakukan dimana saja, restoran, museum, pusat perbelanjaan, taman kota, dan lain-lain. Pada saat melakukan *check-in* pengguna akan diberikan poin. Poin juga bisa bisa didapatkan dengan menambahkan lokasi baru yang belum ada di situs foursquare. foursquare juga bisa digunakan oleh pengguna *non-smartphone* dengan aplikasi *mobile web*. Pengguna foursquare juga dapat *check-in* lokasi terbaru mereka sekaligus juga memberikan komentar tentang lokasi tersebut. foursquare dapat terkoneksi dengan Facebook dan Twitter, jadi jika pengguna melakukan *check-in* maka akan secara otomatis terdapat terpublikasi di akun Facebook dan Twitter yang sebelumnya telah diautentikasi oleh pengguna. Pengguna juga bisa mendapatkan sejumlah *badges* sebagai pangkat jika telah memenenuhi persyaratan dengan *check-in* di tempat-tempat yang direkomendasikan dari situs atau mengunjungi sebuah lokasi dengan intensitas yang tinggi.

Mayor adalah sebuah pangkat yang akan diberikan oleh foursquare ini apabila anda telah *check-in* di satu lokasi dengan intensitas yang tinggi. Dan jika pengguna lain telah mengalahkan intensitas anda dalam *check-in* di lokasi tersebut maka dialah yang akan dinobatkan sebagai seorang *mayor* yang baru. Banyak penawaran yang didapat bila menjadi seorang mayor dari sebuah lokasi. Di dalam situs foursquare, pengguna juga bisa membuat sebuah *list* dari hal-hal yang akan dikerjakan (*to do list*) untuk kepentingan pribadi dan juga memberikan *tips* tentang lokasi tersebut sehingga pengguna lain bisa membacanya. *Tips* ini bisa berupa saran tentang apa yang sebaiknya dilakukan, dilihat, atau dimakan pada lokasi tersebut. *Off the Grid check-in* adalah *check-in* yang dilakukan pengguna secara rahasia karena pengguna tersebut tidak mau lokasinya diketahui oleh pengguna lain.

Dalam pembangunan aplikasi *mobile* Bandung Review menggunakan beberapa layanan lokasi yang terintegrasi dengan layanan foursquare API. foursquare API dalam aplikasi *mobile* Bandung Review digunakan untuk beberapa fitur seperti melakukan *check-in* pada tempat tertentu. foursquare API memberikan akses ke semua data yang digunakan oleh aplikasi *mobile* foursquare. foursquare Platform terdiri dari 4 pondasi utama, antara lain:

- *Core* API

foursquare API memungkinkan untuk melakukan semua hal yang bisa lakukan pada aplikasi *mobile* dan *website* foursquare. Pengguna dapat melakukan *check-in*, melihat sejarah mereka, melihat dimana teman-teman, membuat tips dan

daftar, melakukan pencarian, belajar lebih banyak tentang tempat, spesial akses dan rekomendasi.

- *Real Time* API

Terdapat fitur *push notification* API ke *venue manager* atau pemilik tempat, ketika pengguna melakukan *check-in* pada tempat tersebut dan melakukan *push notification* API ke pengembang ketika pengguna melakukan *check-in* dimanapun.

- *Merchant Platform*

Merchant Platform memungkinkan pengembang untuk menulis aplikasi yang membantu pemilik tempat yang telah terdaftar untuk mengelola tempatnya dan memberikan spesial promo.

- Venues Platform

Venue Platform memungkinkan pengembang untuk mencari tempat dan mengakses banyak informasi tentang tempat tersebut, termasuk alamat, popularitas, tips dan foto. Fitur ini tanpa memerlukan otentikasi pengguna, aplikasi selama termasuk atribusi memadai.

1. 2. 15 *Global Positioning System* (GPS) [1]

GPS (Global Positioning System) merupakan sistem satelit navigasi dan penentuan posisi dengan menggunakan satelit. Nama formalnya adalah NAVSTAR GPS, kependekan dari "*NAVigation Satellite Timing adn Ranging Global Positioning System*". Sistem ini digunakan oleh banyak orang sekaligus dalam segala cuaca, didesain untuk memberikan posisi dan kecepatan tiga dimensi yang teliti, dan juga informasi mengenai waktu, secara kontinyu di seluruh dunia.

Pada dasarnya GPS terdiri dari tiga segmen utama, yaitu segmen ruang angkasa (*space segment*) yang terutama terdiri dari satelit-satelit GPS, segmen sistem kontrol (*control system segment*) yang terdiri dari stasiun-stasiun pemonitor dan pengontrol satelit, dan segmen pemakai (*user segment*) yang terdiri dari pemakai GPS termasuk alat-alat penerima dan pengolah sinyal dan data GPS.

GPS dapat memberikan informasi mengenai posisi, kecepatan, dan waktu secara tepat dan teliti, dan murah dimana saja di Bumi pada setiap waktu, siang maupun malam tanpa tergantung pada kondisi cuaca. Sampai saat ini GPS adalah satu-satunya sistem navigasi yang mempunyai karakteristik prima seperti itu.

1. 2. 16 Windows Phone [18]

Tahun 2010 boleh jadi merupakan *milestone* tersendiri bagi Microsoft dan *mobile platform*. Dalam industri komputer, hal yang dilakukan Microsoft adalah *reboot strategy*. Microsoft menyebut Windows Phone sebagai *a revolutionary new platform*. Microsoft membuat seluruhnya dari awal dan dengan *user interface* yang lebih *clean* dan *fresh*. Dengan filosofi desain yang dinamakan Metro, terinspirasi dari tanda-tanda (*sign*) yang terdapat pada *metro subway*, antarmuka Windows Phone menunjukkan ciri yang jelas, informasi yang mudah diperoleh, intuitif dan menggunakan simbol-simbol yang mudah dipahami. Integrasi Windows Phone dengan berbagai layanan di cloud yang telah dimiliki Microsoft, sebut saja Bing, Xbox Live, Push Notification, Office dan layanan pihak ketiga telah memberikan kekuatan yang unik, sesuatu yang seharusnya dimulai Microsoft sejak dulu.

Di sisi *platform* pengembangan, Windows Phone menawarkan pengalaman pengembangan yang menarik bagi para developer. Sebuah Windows Phone diwajibkan memiliki resolusi 800x480 WGA atau 480x320 HVGA, layar sentuh, sensor GPS, *accelerometer*, kompas, cahaya, kamera, multimedia, GPU dengan DirectX9 dan tiga *hardware button*. Sebagai pengembang dapat dijamin keseluruhan spesifikasi ini akan dimiliki pada *device* yang mendukung Windows Phone. Seluruh *device driver* langsung dibuat oleh Microsoft untuk menjamin konsistensi. Untuk mengembangkan aplikasi di atas Windows Phone dapat memilih dua *platform* yang populer dan modern yakni Silverlight dan XNA.

Silverlight terbukti telah memberikan para pengembang *web* kemampuan untuk mengembangkan antarmuka yang menawan dengan kombinasi kontrol, teks, grafik vektor, media, animasi dan data *binding* yang dapat berjalan pada sejumlah *platform* dan *browser*. Sementara XNA merupakan *platform gaming* yang mendukung permainan 2D dan 3D yang ditujukan untuk Xbox 360, konsol dan PC.

2.2.14.1 Silverlight dan Windows Phone

Dalam membangun aplikasi Windows Phone, terdapat dua *platform* yang dapat dengan bebas kita gunakan, yaitu Silverlight dan XNA. Silverlight untuk Windows Phone adalah sama dengan Silverlight 3 yang dirilis sebelumnya untuk pengembangan *web*. Berikut hal-hal yang berhubungan dengan Silverlight di Windows Phone:

- Menggunakan *base class library* yang sama
- Telah dimodifikasi untuk peningkatan performa

- Terdapat integrasi dengan perangkat keras
- Integrasi dengan sistem operasi
- API yang spesifik berhubungan dengan *device* (*accelorometer*, GPS, dan sebagainya)
- Menggunakan model *out-of-browser*

Pada *template* yang telah disediakan untuk pengembangan aplikasi menggunakan *platform* Silverlight disediakan tiga tipe projek yang bisa kita gunakan sesuai dengan kebutuhan :

- **Windows Phone Application**, yang akan menyediakan halaman kosong tanpa kontrol sama sekali
- **Windows Phone List Application**, yang akan menyediakan contoh untuk skenario aplikasi master-detail data
- **Windows Phone Panorama Application**, yang akan menyediakan contoh skenario menggunakan navigasi panorama pada aplikasi
- **Windows Phone Pivot Application**, yang akan menyediakan contoh skenario menggunakan navigasi *pivot* pada aplikasi
- **Windows Phone Class Library**, untuk membangun komponen yang dapat digunakan kembali di projek-projek yang lain

Secara *default* projek akan terdiri dari *file-file* berikut ini :

Tabel 2. 1 *Template* Projek Windows Phone

Item	Deskripsi
App.xaml / App.xaml.cs	*Entry point* dari aplikasi yang akan menginisialisasi *resource* dan tampilan

	dari aplikasi.
MainPage.xaml / MainPage.xaml.cs	Mendefinisikan sebuah halaman dengan antarmuka pada aplikasi.
ApplicationIcon.png	File gambar yang akan menjadi *icon* pada list aplikasi. *Icon* ini dapat kita ganti jika diinginkan.
Background.png	File gambar yang akan menjadi *background* aplikasi
SplashScreenImage.jpg	File gambar yang akan ditampilkan pertama kali ketika aplikasi diaktifikan. *Splash screen* berguna untuk memberikan *response* sesegera mungkin kepada pengguna sebelum halaman pertama dari aplikasi ditampilkan.
Properties\AppManifest.xml	*File manifest* untuk keperluan pembangkitan paket aplikasi
Properties\AssemblyInfo.cs	*File assembly* yang mengandung informasi nama dan versi metadata yang akan dilekatkan pada *assembly* yang dihasilkan
Properties\WMAppManifest.xml	*File manifest* yang mengandung secara spesifik metadata terakit aplikasi

	Windows Phone yang spesifik mengandung fitur-fitur pada Silverlight untuk Windows Phone.

Satu lagi hal yang harus diperhatikan bahwa aplikasi dengan *platfrom* Silverlight di Windows Phone menggunakan sepenuhnya teknik navigasi yang terdapat pada Silverlight 3. Dengan menggunakan kontainer *frame* maka navigasi akan semakin mudah ditangani secara alami dan khusus untuk navigasi *back* sudah terintegrasi dengan tombol yang terdapat pada *hardware*.

2.2.14.2 *Application Life Cycle*

Eksekusi model pada Windows Phone memiliki sebuah siklus lengkap mulai dari aplikasi dipanggil hingga dinon-aktifkan. Model eksekusi ini dirancang agar menyediakan pengalaman yang cepat, responsif pada setiap waktu. Sehingga untuk mencapai hal ini pada Windows Phone hanya mengizinkan satu aplikasi yang berjalan pada satu waktu. Hal ini mencegah kemungkinan *device* menjadi terkesan lambat dan tidak responsif akibat aplikasi yang berjalan pada *background*.

Beberapa terminologi yang harus diketahui dalam mengetahui aspek model eksekusi dari aplikasi pada Windows Phone.

Tabel 2. 2 Model Eksekusi Aplikasi Windows Phone

Istilah	Deskripsi
Tombstoning	Prosedur dimana sistem operasi akan menon-aktifkan proses aplikasi ketika pengguna keluar dari aplikasi. Sistem operasi akan menjaga informasi keadaan dari suatu aplikasi. Jika

	pengguna kembali menggunakan aplikasi maka sistem operasi me-*restart* proses dan mengirimkan kondisi terakhir sebelum aplikasi di non-aktifkan
Page State	Keadaan tampilan dari suatu halaman aplikasi. Hal ini termasuk posisi *scroll*, atau isi dari sebuah *text box*. Pengaturan keadaan halaman dilakukan dengan melakukan *override* terhadap fungsi OnNavigatedTo dan OnNavigatedFrom
Application State	Kondisi dari aplikasi dimana tidak terdapat asosiasi spesifik terhadap suatu halaman. Kondisi ini diatur menggunakan kelas PhoneApplicationService
Persistent Data	Data yang digunakan bersama oleh aplikasi. Data ini disimpan dai bangkitkan dari *isolated storage*. Pengaturan aplikasi adalah salah satu contoh data yang bersifat persisten.
Transient Data	Data yang bersifat transien adalah data yang berhubungan dengan instansiasi dari aplikasi. Data transien disimpan pada *state dictionary* yang disediakan oleh PhoneApplicationService. Sebuah aplikasi yang berada pada kondisi *tombstoned* akan melakukan pembalikan kondisi transien ketika aplikasi kembali diaktifkan. Contoh dari data tipe ini adalah *query* dari *web service*.

Berikut pemaparan alur siklus pada sebuah aplikasi Windows Phone antara lain:

- ***Launching***

Sebuah aplikasi Windows Phone dikatakan telah di-*launch* ketika aplikasi dipanggil baik karena pengguna terus menekan tombol *back* untuk kembali ke aplikasi yang telah dipanggil sebelumnya, dipilih dari daftar aplikasi yang telah diinstal atau dari *tile* pada layar utama. Cara apapun yang digunakan, sebuah instans dari aplikasi akan dibuat. Dan ketika aplikasi mulai berjalan, *event launching* akan dibangkitkan. Aplikasi sebaiknya tidak mengambil data dari *Isolated Storage*. Karena *event* tersebut dibangkitkan sesaat sebelum aplikasi aktif atau tampil, melakukan pekerjaan yang memakan waktu seperti mengakses IsolatedStorage dapat memberikan pengalaman yang buruk karena aplikasi akan terkesan lama pada saat diaktifkan. Sebaiknya pemanggilan data dari *Isolated Storage* atau jaringan dilakukan secara asinkronus setelah aplikasi di-*load*. Ketika aplikasi di-*launch*, aplikasi sebaiknya juga tidak memanggilkan kondisi transien dari instansiasi aplikasi sebelumnya. Ketika sebuah aplikasi di-*launch*, harus tetap terlihat sebagai sebuah instansasi yang baru.

- ***Running***

Setelah *event launching* ditangani, aplikasi akan mulai berjalan. Pada kondisi ini, aplikasi akan mengatur kondisi dari dirinya sendiri ketika misalnya pengguna melakukan navigasi melalui halaman-halaman dalam aplikasi. Satu-satunya aktivitas yang dapat terjadi adalah aplikasi secara inkremental melakukan penyimpanan data atau pengaturan sehingga mengurangi jumlah data yang perlu disimpan ketika kondisi aplikasi berubah. Untuk aplikasi dengan jumlah data sedikit, hal ini menjadi tidak perlu dilakukan.

- ***Closing***

Salah satu contoh skenario yang membangkitkan *event Closing* adalah pengguna menekan tomboh *Back* pada *device* pada saat aplikasi berada pada halaman pertamanya. Aplikasi harus mampu melakukan penyimpanan data yang bersifat persisten ke dalam *Isolated Storage*. Tidak perlu menyimpan data yang bersifat transien, atau data yang hanya berhubungan pada satu instansiasi aplikasi, karena satu-satunya cara bagi pengguna untuk kembali ke dalam aplikasi setelah dinon-aktifkan adalah dengan mengaktifkan aplikasi itu lagi, dan sesuai dengan penjelasan pada bagian sebelumnya, aplikasi harus berupa sebuah instansiasi yang sama sekali baru.

- ***Deactivating***

Ketika sebuah aplikasi berjalan dan digantikan oleh aplikasi lain maka aplikasi pertama akan di-deaktiviasi. Ada beberapa skenario bagaimana kondisi ini tercapai. Pertama ialah menekan tombol *Start* atau *timeout* akibat layar utama terkunci. Sebuah aplikasi juga dapat dideaktivasi karena terdapat invokasi *Launcher* atau *Chooser*, yaitu aplikasi bantu yang dapat memungkinkan pengguna melakukan *task* umum pada suatu perangkat *mobile* seperti mengamil gambar, mengirim *email*. Pada kasus-kasus demikian maka aplikasi yang berjalan akan membangkinkan *event Deactivated* dan masuk ke kondisi *deactivating*. Tidak seperti saat aplikasi di non-aktifkan maka sebuah aplikasi yang dapat kondisi *deactivated* berada pada kondisi *tombstoned*. Artinya, aplikasi sudah tidak berjalan kembali tapi sistem operasi merekam kondisi aplikasi dan menyimpan

sejumlah data yang berhubungan dengannya. Sangat mungkin kalau pengguna kembali ke aplikasi dan ketika itu terjadi maka mencapai kondisi *reactivated*.

Pada kondisi *event Deactivated* sebuah aplikasi harus menyimpan informasi yang berhubungan dengan kondisi saat itu melalui *property State* pada PhoneApplicationService. Data yang disimpan dalam kamus adalah daya yang bersifat transien atau data yang akan membantu aplikasi kembali ke kondisi sesaat aplikasi belum di deaktivasi. Karena tidak ada jaminan sebuah aplikasi yang masuk kondisi *tombstoned* akan pernah diaktivasi kembali maka aplikasi juga harus menyimpan data ke dalam *Isolated Storage*. Seluruh aksi tersebut harus dijamin dapat ditangani dalam jangka waktu 10 detik atau kalau tidak sistem operasi akan menterminasi aplikasi tersebut. Itulah karenanya aplikasi yang menggunakan data berjumlah besar sebaiknya melakukan penyimpanan secara inkremental selama aplikasi berjalan.

Berikut adalah daftar aksi yang akan menyebabkan aplikasi berada dalam kondisi *tombstoned*:

- WebBrowserTask
- MarketplaceDetailTask
- MarketplaceHubTask
- SaveEmailAdressTask
- SavePhoneNumberTask
- SearchTask
- SmsComposeTask

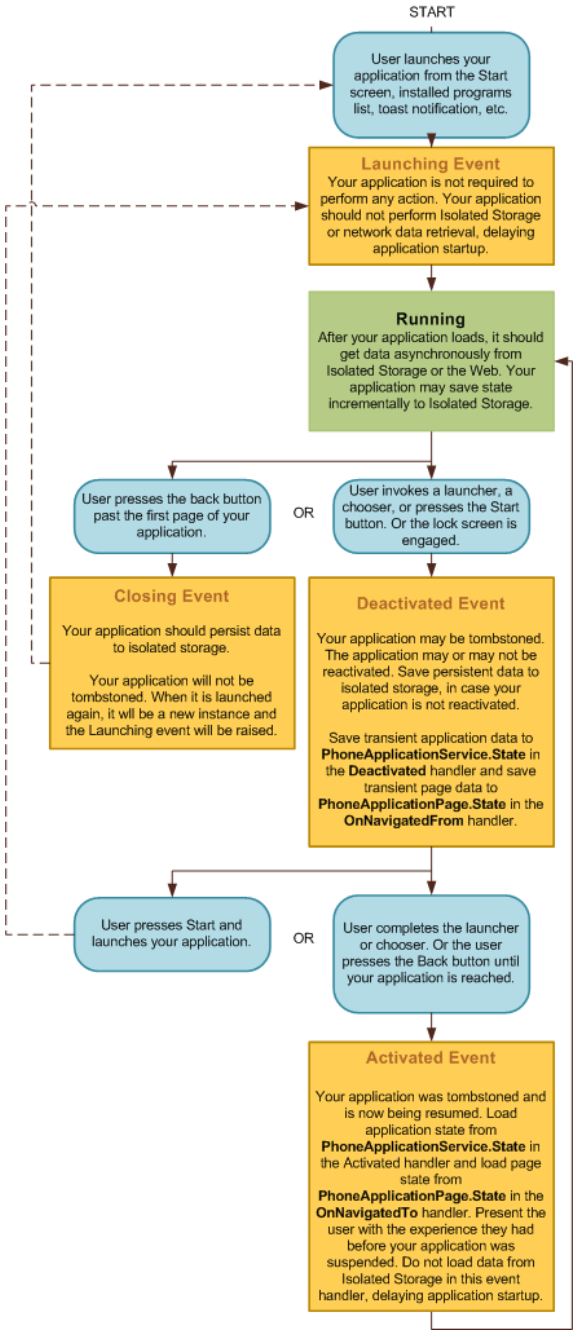
Aksi-aksi berikut tidak secara otomatis akan menyebabkan aplikasi masuk dalam kondisi *tombstoned* sehingga aksi dibawah ini tetap harus ditangani :

- PhotoChooserTask
- CameraCaptureTask
- MediaPlayerLauncer
- EmailAdrdressChooserTask
- PhoneNumberChooserTask
- ***Activating***

Setelah aplikasi dideaktivasi dan masuk dalam kondisi *tombstoned* maka sangat mungkin terjadi proses reaktivasi kembali. Aplikasi dapat diinvokasi menjadi sebuah instansiasi aplikasi baru dari *Start* atau pengguna dapat memanggil aplikasi melalui aplikasi lain sehingga aplikasi yang dalam kondisi *tombstoned* tidak akan pernah dipanggil kembali atau ketika *Launcher* dan *Chooser* adalah penyebab deaktivasi maka pengguna dapat menyelesaikan aktivitas yang berhubungan dengan itu dan kembali ke aplikasi yang dalam kondisi *tombstoned*. Ketika hal itu terjadi aplikasi akan direaktivasi dan *event Activated* akan dibangkitkan. Aplikasi harus membaca nilai dari kamus pada PhoneApplicationService untuk membangkitkan kondisi terakhir dari aplikasi. Seperti pada penanganan *event launching* sebaiknya aplikasi tidak mengakses sumber daya dari jaringan atau dari *Isolated Storage* karena dapat memperlambat jalannya aplikasi.

Pengetahuan mengenai model eksekusi ini mutlak diperlukan untuk menjaga konsistensi dan memberikan pengalaman yang konsisten bagi pengguna.

Tim dari Microsoft sudah memberikan panduan mengenai *best practices* berhadapan dengan model eksekusi pada aplikasi Windows Phone. Gambar 2.20 di bawah ini menunjukkan alur kerja aplikasi untuk memberikan pemahaman yang lebih baik mengenai alur aplikasi yang sebenarnya.



Gambar 2. 20 Alur Kerja Aplikasi Windows Phone

2.2.14.3 *Security*

Keamanan menjadi sebuah isu tersendiri dalam pengembangan aplikasi Windows Phone. Aspek ini secara langsung akan mempengaruhi pengembang dalam membangun aplikasi nantinya. Terdapat sejumlah *built-in* fitur yang memang tersedia pada Windows Phone dan mempengaruhi apa-apa saja yang dapat kita lakukan dan bagaimana kita harus memprogram. Berikutnya adalah jika aplikasi akan mengirimkan atau menerima data yang sensitif melalui *web* maka harus dapat dipastikan data-data tersebut aman. Untuk hal ini Silverlight *for* Windows Phone menyediakan sejumlah kelas yang dapat digunakan.

Silverlight pada Windows Phone dirancang dengan beberapa fitur *built-in* untuk mendukung aspek keamanan ini. Aplikasi pada Windows Phone berjalan pada lingkungan terbatas atau Sandbox, sehingga hanya menyediakan hak yang terbatas seperti tidak dapat mengakses *filesystem*, atau *file* aplikasi lain seperti aplikasi .NET pada umumnya. Hal ini menjamin aplikasi tidak akan mempengaruhi sistem operasi atau fitur tertentu pada device, seperti kamera, email. Hal ini mempengaruhi aspek pengembangan sehingga pengembang hanya perlu belajar bagaimana memanggil *task* yang berhubungan dengan sistem operasi atau fitur tadi melalui *managed-code* karena tidak dapat dipanggil secara langsung. Dan untuk menggunakannya, dikenalkan suatu konsep *Launchers* dan *Choosers*. Selanjutnya, dikarenakan aplikasi tidak dapat mengakses *filesystem* sementara untuk beberapa skenario masih dibutuhkan penyimpanan data maka dalam Silverlight dikenal istilah *Isolated Storage*, tempat kita dapat menyimpan sejumlah data disana. Bersifat *isolated* karena aplikasi hanya dapat mengakses

Isolated Storage miliknya saja. Hal ini sangat menyederhanakan pemrograman untuk menyimpan data pada aplikasi kita.

Fitur di atas tadi merupakan sesuatu yang sudah tersedia dan kita hanya perlu belajar bagaimana menggunakannya. Di lain sisi, Silverlight untuk Windows Phone juga menyediakan sejumlah kakas untuk keamanan aplikasi. Pengiriman data sensitif melalui web merupakan salah satu skenario dimana ingin mengamankan data yang dikirimkan. Terdapat sejumlah *namespace* yang dapat digunakan sepenuhnya untuk keperluan ini yaitu :

- System.Security.Principal, memberikan informasi tentang pengguna dan pengaturan *role.*
- System.Security.Permissions, yang mengekspos fitur-fitur terhadap akses sumber daya tertentu.
- System.Security.Cryptography, yang mendiakan fungsi enrkipsi dan *hash*: AES, SHA1,SHA256 dan HMAC.

2.2.14.4 Desain Metro UI pada Windows Phone [16]

Metro merupakan nama dari bahasa desain baru yang diciptakan untuk antarmuka Windows Phone 7. Pada awalnya, tim desain Windows Phone menarik dari berbagai sumber inspirasi untuk menentukan prinsip-prinsip panduan untuk antarmuka ponsel generasi berikutnya. Berbagai sumber inspirasinya berasal dari desain grafis di negara Swiss dan kemasan dengan penekanan pada kesederhanaan, desain grafis yang hampis sama juga dapat ditemukan di sarana transportasi dan perangkat lunak Microsoft lainnya seperti Microsoft seperti Zune,

Labs Office dan game dengan fokus yang kuat pada gerakan dengan lebih fokus pada tipografi yang bersih dan mengurangi UI *chrome*.

Tidak hanya memiliki bahasa desain baru, Microsoft juga lebih menekankan pengalaman yang unik dan mendalam bagi pengguna Windows Phone 7, dan dapat diimplementasikan juga pada aplikasi pihak ketiga. Standar yang telah dikembangkan untuk Metro UI menyediakan dasar yang besar, untuk menyamakan perspektif antara desainer dengan pengembang. Standar tersebut membantu untuk menciptakan pengalaman dari masukan *gesture* dari pengguna Windows Phone 7 yang dibangun untuk perangkat kecil.

Prinsip Desain Metro UI

- *Typography*. Huruf merupakan salah satu aspek penting dalam keindahan sebuah antarmuka. Tidak hanya menarik bagi mata, tetapi juga dapat menjadi fungsional. Keseimbangan antara ketebalan huruf dan posisi dapat membuat hirarki visual. Selain itu, jenis posisi yang baik dapat membantu membawa pengguna untuk lebih banyak konten.
- *Motion* adalah apa yang membawa antarmuka untuk hidup. Transisi adalah sama pentingnya dengan desain grafis. Dengan mengembangkan sebuah set konsisten gerakan atau animasi, sistem yang dibuat yang menyediakan konteks untuk kegunaan, dimensi ekstra dan kedalaman dan meningkatkan kinerja yang dirasakan dari seluruh antarmuka.
- *Content not Chrome* adalah salah satu prinsip yang lebih unik dari Metro. Dengan menghapus semua pengertian tentang chrome ekstra di UI, konten

menjadi fokus utama. Hal ini sangat relevan karena ukuran layar yang lebih kecil dan gerakan berbasis interaksi.

- *Honesty*. Desain secara eksplisit untuk faktor bentuk tangan memegang perangkat menggunakan sentuhan, layar resolusi tinggi dan bentuk yang disederhanakan dan dipercepat interaksi. Dengan kata lain, menjadi *"authentically digital"*.

2.2.14.5 Bing Maps Control pada Windows Phone

Bing Maps Silverlight Control pada Windows Phone mengombinasikan kekuatan Silverlight dan Bing Maps untuk dukungan pengalaman penggunaan peta di dalam aplikasi. Pengembang kini dapat menggunakan Bing Maps Silverlight Control, termasuk didalamnya layanan lokasi dan pencarian. Bagi yang cukup familiar dalam menggunakan kontrol ini pada aplikasi Silverlight biasa maka sama sekali tidak akan mendapatkan kesulitan untuk menggunakannya pada aplikasi Windows Phone.

Untuk dapat menggunakan layanan ini maka harus mendaftarkan diri untuk memperoleh kunci untuk menggunakan kontrol, SOAP Services, REST dan Bing Spatial Data Services. Tanpa kunci yang valid tidak dapat melakukan pengambilan data melalui *web*.

1. 3 Pembangun Perangkat Lunak

1. 3. 1 Visual Studio [20]

Microsoft Visual Studio merupakan lingkungan pengembangan yang terintegrasi (IDE) dari Microsoft. *Tools* ini digunakan untuk mengembangkan *console* aplikasi maupun aplikasi yang berbasiskan GUI (*Graphical User*

Interface) seperti Windows Forms, *web sites*, *web application*, dan *web service* yang bersifat *managed code* untuk semua *platform* yang didukung oleh Microsoft Windows, Windows Phone, Windows CE, .NET Compact Framework dan Microsoft Silverlight.

Pada editor kode Visual Studio mendukung fitur *IntelliSense* serta *refactoring code*. *Integrated debugger* yang bekerja baik pada *source-level debugger* dan *machine-level debugger*. Selain kakas *built-in* termasuk *forms designer* untuk membangun aplikasi GUI, desainer *web*, desainer *class*, dan perancang skema basis data, pada Visual Studio dapat ditambahkan *plug-in* untuk meningkatkan fungsionalitasnya, termasuk menambahkan dukungan untuk *source-control system* (seperti Subversion dan Visual SourceSafe) dan menambahkan *toolsets* baru seperti *editor* dan desainer *visual*.

Visual Studio mendukung berbagai bahasa pemrograman. *Built-in language* yang didukung antara lain C/C++ (Visual C++), VB.NET (Visual Basic NET.), C# (Visual C#), dan F# (Visual Studio 2010). Dukungan untuk bahasa lain seperti Python dan Ruby tersedia melalui layanan bahasa yang diinstal secara terpisah. Visual Studio juga mendukung format *plain text* lainnya seperti XML/XSLT, HTML/XHTML, JavaScript dan CSS.

1. 3. 2 Windows Phone SDK [17]

Windows Phone SDK adalah seperangkat lingkungan pengembangan yang dibutuhkan untuk memulai pengembangan aplikasi Windows Phone. Saat ini telah tersedia Windows Phone SDK 7.1 RC yang memungkinkan pengembang membuat aplikasi untuk sistem operasi Windows Phone terbaru “Mango”.

Pada Windows Phone SDK sudah terdapat seluruh kakas yang diperlukan antara lain: Windows Phone SDK 7.1, Windows Phone Emulator, Windows Phone SDK 7.1 Assemblies, Silverlight 4 SDK dan DRT, Windows Phone SDK 7.1 *extension for* XNA, Microsoft Expression Blend SDK Preview *for* WIndows Phone 7.1, WCF Data Services Client *for* Windows Phone 7.1 dan Microsoft Advertising SDK *for* Windows Phone.

1. 3. 3 MySQL [14]

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL) ditunjukan pada Gambar II-9, tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

1. Sistem Manajemen Basis Data Relasional

MySQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basis data yang telah ada sebelumnya; SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem basis data (DBMS) dapat diketahui dari cara kerja pengoptimasinya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basis data transaksional maupun operasi basis data nontransaksional. Pada modus operasi nontransaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak basis data kompetitor lainnya. Namun demikian pada modus nontransaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus nontransaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi *blogging* berbasis *web* (*wordpress*), CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basis data transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus nontransaksional.

2. Keistimewaan MySQL

MySQL memiliki beberapa keistimewaan, antara lain :

- Portabilitas

  MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.

- Perangkat lunak sumber terbuka

  MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis.

- Multi-user

  MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

- Performance Tuning

  MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.

- Ragam Tipe Data

  MySQL memiliki ragam tipe data yang sangat kaya, seperti *signed*/*unsigned integer, float, double, char, text, date, timestamp*, dan lain-lain.

- Perintah dan Fungsi

  MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).

- Keamanan

MySQL memiliki beberapa lapisan keamanan seperti level *subnetmask*, nama *host*, dan izin akses user dengan sistem perizinan yang mendetail serta sandi terenkripsi.

- Skalabilitas dan Pembatasan

  MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (*record*) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

- Konektivitas

  MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, Unix Socket (UNIX), atau *Named Pipes* (NT).

- Lokalisasi

  MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meski pun demikian, bahasa Indonesia belum termasuk di dalamnya.

- Antar Muka

  MySQL memiliki antar muka (*interface*) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (Application Programming Interface).

- Klien dan Peralatan

  MySQL dilengkapi dengan berbagai peralatan (*tool*) yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk *online*.

- Struktur Tabel

  MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan basis data lainnya semacam PostgreSQL ataupun Oracle.

3. Bahasa Pemrogaman

Terdapat beberapa API (Application Programming Interface) tersedia yang memungkinkan aplikasi-aplikasi komputer yang ditulis dalam berbagai bahasa pemrograman untuk dapat mengakses basis data MySQL antara lain: bahasa pemrograman C, C++, C#, bahasa pemrograman *Eiffel*, bahasa pemrograman Smalltalk, bahasa pemrograman Java, bahasa pemrograman Lisp, Perl, PHP, bahasa pemrograman Python, Ruby, REALbasic dan Tcl. Sebuah antarmuka ODBC memanggil MyODBC yang memungkinkan setiap bahasa pemrograman yang mendukung ODBC untuk berkomunikasi dengan basis data MySQL. Kebanyakan kode sumber MySQL dalam ANSI C.

4. Penggunaan

MySQL sangat populer dalam aplikasi web seperti MediaWiki (perangkat lunak yang dipakai Wikipedia dan proyek-proyek sejenis) dan PHP-Nuke dan berfungsi sebagai komponen basis data dalam LAMP. Popularitas sebagai aplikasi web dikarenakan kedekatannya dengan popularitas PHP, sehingga seringkali disebut sebagai *Dynamic Duo*.

1. 3. 4 XAMPP *Server* [8]

XAMPP adalah perangkat lunak open source, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya sebagai

server yang berdiri sendiri (*localhost*), yang terdiri atas program Apache HTTP Server, MySQL *database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Program ini tersedia dalam GNU General Public License dan bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis.

XAMPP merupakan singkatan dari huruf X artinya program ini dapat dijalankan dibanyak sistem operasi, seperti Windows, Linux, Mac OS, dan Solaris. Huruf A artinya Apache, merupakan aplikasi *web server*. Tugas utama Apache adalah menghasilkan halaman *web* yang benar kepada pengguna berdasarkan kode PHP yang dituliskan oleh pembuat halaman *web*. Jika diperlukan juga berdasarkan kode PHP yang dituliskan, maka dapat saja suatu *database* diakses terlebih dahulu (misalnya dalam MySQL) untuk mendukung halaman *web* yang dihasilkan. Huruf M artinya MySQL, merupakan aplikasi *database server*. Perkembangannya disebut SQL yang merupakan kepanjangan dari *Structured Query Language*. SQL merupakan bahasa terstruktur yang digunakan untuk mengolah basis data. MySQL dapat digunakan untuk membuat dan mengelola basis data beserta isinya. Kita dapat memanfaatkan MySQL untuk menambahkan, mengubah, dan menghapus data yang berada dalam database. Huruf P artinya PHP, bahasa pemrograman *web*. Bahasa pemrograman PHP merupakan bahasa pemrograman untuk membuat *web* yang bersifat *server-side scripting*. PHP memungkinkan kita untuk membuat halaman *web* yang bersifat dinamis. Sistem manajemen basis data yang sering digunakan bersama PHP adalah MySQl. Namun PHP juga mendukung sistem manajemen basis data

Oracle, Microsoft Access, Interbase, d-base, PostgreSQL, dan sebagainya. Huruf P artinya Perl, bahasa pemrograman.

Beberapa bagian XAMPP yang sering digunakan antara lain:

- htdoc adalah *folder* tempat meletakkan berkas-berkas yang akan dijalankan, seperti berkas PHP, HTML dan skrip lainnya.
- phpMyAdmin merupakan bagian untuk mengelola basis data MySQL yang ada di komputer.
- Kontrol Panel yang berfungsi untuk mengelola layanan (*service*) XAMPP. Seperti menghentikan (*stop*) atau memulai (*start*) layanan.

Paket XAMPP 1.7.7, terdiri dari:

- Apache 2.2.21
- MySQL 5.5.16
- PHP 5.3.8
- phpMyAdmin 3.4.5
- FileZilla FTP Server 0.9.39
- Tomcat 7.0.21 (dengan mod_proxy_ajp sebagai konektor)

1. 3. 5 Zend Studio [21]

Zend Studio merupakan komersial *proprietary* Integrated Development Environtment (IDE) untuk PHP yang dikembangkan oleh Zend Technologies dan dibangun berdasarkan PHP Development Tools (PDT) plugin untuk *platform* Eclipse.

Zend Studio terintegrasi dengan Zend Server, Zend *pre-integrated* dan telah diuji dalam beberapa aplikasi PHP. Integrasi ini memungkinkan

pengembang dengan cepat mengatur lingkungan PHP secara lengkap dan mempercepat analisis akar penyebabnya, *error code* akan segera terdeteksi dalam pengujian atau dalam lingkungan produksi.

Zend Studio juga terintegrasi dengan Zend Framework. Misalnya, memberikan MVC untuk navigasi kode secara mudah dan integrasi dengan Zend Tool untuk menghasilkan kode secara otomatis.

Beberapa fitur unggulan yang terdapat pada Zend Studio seperti *code folding*, kostumisasi *Framework Project Layout*, Zend Framework Zend_Tool Integration, *model–view–controller*, *in-place refactoring* (*smart rename*), analisis *code semantic* dan *quick fix*, mendukung PHP 4 dan PHP 5.X.