BAB II

LANDASAN TEORI

2.1. *Web* 2.0

2.1.1. Definisi *Web* 2.0

Web 2.0 adalah sebuah istilah yang dicetuskan pertama kali oleh O'Reilly Media pada tahun 2003, dan dipopulerkan pada konferensi *web* 2.0 pertama di tahun 2004, merujuk pada generasi yang dirasakan sebagai generasi kedua layanan berbasis web seperti situs jaringan sosial, wiki, perangkat komunikasi, dan folksonomi yang menekankan pada kolaborasi online dan berbagi antar pengguna. O'Reilly Media dengan kolaborasinya bersama MediaLive International, menggunakan istilah ini sebagai judul untuk sejumlah seri konferensi, dan sejak 2004 beberapa pengembang dan pemasar telah mengadopsi ungkapan ini. Menurut **Tim O'Reilly**, *Web* 2.0 dapat didefinisikan sebagai berikut:

> *"Web 2.0 adalah revolusi bisnis di industri komputer yang disebabkan oleh penggunaan internet sebagai platform, dan merupakan suatu percobaan untuk memahami berbagai aturan untuk mencapai keberhasilan pada platform baru tersebut. Salah satu aturan terutama adalah: membangun aplikasi yang mengeksploitasi efek jaringan untuk mendapatkan lebih banyak lagi pengguna aplikasi tersebut."*

2.1.2. Karakteristik *Web* 2.0

Biasanya terdapat tiga karakteristik yang harus dipenuhi sebuah aplikasi *web* agar dapat dinilai sebagai aplikasi *web* 2.0 (Zibriel dan Supangkat, 2008).

1. Menggunakan SOA *(Service Oriented Architecture*)

Web 2.0 umunya menggunakan SOA , dalam melaksanakan fungsinya sebagai penyedia layanan. SOA adalah sebuah konsep arsitektur sistem komputer yang membuat dan menggunakan langkah-langkah proses bisnis dalam bentuk paket layanan. Bentuk paket layanan yang dimaksud oleh SOA untuk berusaha membungkus kerumitan yang terjadi dari sudut pandang pengguna sistem. Penggunaan SOA memungkinkan perancang sistem untuk menghubungkan berbagai aplikasi yang berlainan jenis tanpa perlu disadari oleh penggunanya. Oleh karena itu implementasi SOA biasanya menggunakan GUI (*Graphic User Interface*) untuk membungkus cara kerja aplikasi yang sebenarnya. Karakter utama dari aplikasi SOA adalah layanan yang menunggu secara terus-menerus untuk digunakan.

2. Menggunakan RIA (*Rich Internet Aplication*)

RIA adalah aplikasi web yang dapat memberikan fitur apapun fungsi aplikasi desktop kepada para penggunannya. Artinya beberapa keunggulan atau kemudahan pada saat menggunakan aplikasi yang berjalan di atas *desktop* dapat dilakukan juga oleh aplikasi web RIA yang berjalan di suatu *server* serta diakses oleh pengguna sistem hanya dengan menggunakan bantuan sebuah *browser*. Contoh dari kemudahan aplikasi *desktop* yang telah beradaptasi oleh aplikasi *web* RIA adalah fitur *drag-and-drop* fitur *shortcut*, fitur *recovery*. Pada umumnya aplikasi *web* RIA hanya mengirimkan sejumlah data yang diperlukan klien tetapi tetap menyimpan seluruh data utama (seperti status pengguna) pada sisi *server* aplikasi. Contoh teknologi yang digunakan untuk mewujudkan RIA adalah Flash dan Ajax.

3. Menggunakan pendekatan *Social Web*

Kriteria yang terakhir yang sekaligus merupakan daya tarik dari aplikasi web 2.0 adalah menggunakan pendekatan *Social Web* dalam memperkaya layanan yang diberikan. Dalam konsep *Social Web* setiap pengguna aplikasi *web* (terdaftar atau tidak) diminta untuk saling berkolaborasi untuk menambah, menghapus, menyunting ataupun mengkategorikan konten dari sebuah layanan sehingga kualitas dan kegunaan layanan benar-benar ditentukan oleh kontribusi dari setiap pengguna layaknya sebuah komunitas dalam dunia nyata.

2.1.3. Prinsip-prinsip *Web* 2.0

Adapun prinsip-prinsip yang mendasari karakter *web* 2.0 adalah sebagai berikut.

1. *The Web as Platform*

   Aplikasi *Web* 2.0 menggunakan *Web* (Internet) sebagai *platform*-nya dimana suatu aplikasi yang dijalankan tidak lagi dibatasi oleh sistem operasi

2. *Harnessing Collective Intelligence*

   Aplikasi Web 2.0 memiliki sifat yang unik, yaitu memanfaatkan kepandaian dari banyak orang secara kolektif. Sebagai hasilnya muncullah basis pengetahuan yang sangat besar hasil gabungan dari pengetahuan banyak orang.

3. *Data is the Next Intel Inside*

   Kekuatan aplikasi Web 2.0 terletak pada data sehingga aplikasi-aplikasi

Internet yang dijalankan dapat berhasil karena selalu didukung oleh basis data yang kuat dan unik.

4. *End of the Software Release Cycle*

   Aplikasi *Web* 2.0 memiliki sifat yang berbeda dengan aplikasi pada *platform* "lama" seperti Windows. Suatu aplikasi Windows biasanya dirilis setiap dua atau tiga tahun sekali. Di lain pihak, aplikasi *Web* 2.0 selalu di-*update* terus-menerus karena sifatnya yang bukan lagi produk melainkan layanan.

5. *Light weight Programming Models*

   Aplikasi *Web* 2.0 menggunakan teknik-teknik pemrograman yang "ringan" seperti AJAX dan RSS. Ini memudahkan orang lain untuk memakai ulang layanan suatu aplikasi *Web* 2.0 guna membentuk layanan baru.

6. *Software Above the Level of a Single Device*

   Aplikasi Web 2.0 bisa berjalan secara terintegrasi melalui berbagai *device*.

7. *Rich User Experiences*

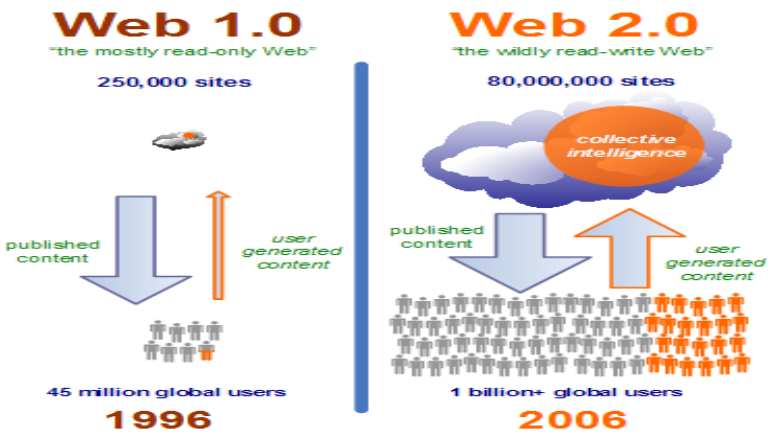
   Aplikasi Web 2.0 memiliki *user interface* yang kaya meskipun berjalan di dalam browser. Teknologi seperti AJAX memungkinkan aplikasi Internet memiliki waktu respons yang cepat dan *user interface* yang intuitif.

2.1.4. Kelebihan *Web 2.0*

Dalam perkembanganya, aplikasi *web* yang dibangun dengan menggunakan orientasi *web* 2.0 ternyata dirasakan mempunyai beberapa nilai positif (Zibriel dan Supangkat, 2008). Nilai positifnya adalah sebagai berikut :

1. *Web* 2.0 berhasil menyajikan sebuah layanan yang komprehensif pada *platform* apapun. Cukup menggunakan sebuah *browser* dan melakukan koneksi dengan *server* maka setiap orang sudah dapat menggunakanya.
2. Dalam penggunaanya, *web* 2.0 lebih mudah digunakan karena aplikasinya berjalan secara terpusat di *server*, pengguna tidak perlu repot lagi untuk memperbaharui *aplikasi* mereka secara mandiri.
3. Dalam segi pemrograman , *web* 2.0 memiliki teknik pemrograman *front-end* yang relatif ringan hal ini dikarenakan *web* 2.0 adalah sebuah aplikasi yang berjalan di sebuah *browser*. Sehingga mudah untuk digunakan kembali (*reuse*).
4. Kelebihan orientasi *web* 2.0 dibandingkan dengan *web* 1.0 (aplikasi yang layanannya hanya berorientasi pada pemenuhan tujuan bisnis) adalah lebih cepat dan lebih mudah mengumpulkan data karena kontributornya berasal dari berbagai sumber, tingkatan dan bidang keahlian.

   Adapun perbandingan *web* 1.0 dan *web* 2.0 dapat terlihat pada Gambar 2.1.



Gambar 2.1. Kelebihan *web* 2.0 dibandingkan dengan *web* 1.0

2.2. Aplikasi *Web*

Pada awalnya aplikasi *web* dibangun hanya dengan menggunakan bahasa yang disebut HTML (*HyperText Markup Language*) dan protocol yang digunakan dinamakan HTTP (*Hypertext Transfer Protokol*). Pada perkembangan berikutnya, sejumlah *script* dan objek dikembangkan untuk memperluas kemampuan HTML. Aplikasi *web* sendiri dapat dibagi menjadi :

1. *Web* Statis

   Web statis dibentuk dengan menggunakan HTML saja. Kekurangan aplikasi seperti ini terletak pada keharusan untuk memelihara program secara terus menerus untuk mengikuti ssetiap perubahan yang terjadi. Kelemahan ini diatasi dengan model aplikasi *web* dinamis.

2. *Web* Dinamis

   Web dinamis diartikan sebagai halaman yang dilengkapi dengan animasi gambar, selain dapat berinteraksi dengan basis data. Dari sisi teknologi yang digunakan

untuk membentuk *Web* Dinamis, terdapat dua macam pengelompokan, yaitu Teknologi pada sisi klien (*client-side technology*), dan Teknologi pada sisi server (*server-side technology*). Teknologi web pada sisi klien diimplementasikan dengan mengirimkan kode perluasan HTML atau program tersendiri dan HTML ke klien. Klienlah yang bertanggung jawab dalam melakukan proses terhadap seluruh kode yang diterima. Kelemahan pendekatan seperti ini adalah terdapat kemungkinan bahwa browser pada klien tidak mendukung fitur kode perluasan HTML. Sebagai contoh kode *VBScript* yang dilekatkan pada kode HTML tidak akan berfungsi sekiranya browser yang digunakan klien tidak mendukungnya. Kelebihan teknologi pada sisi klien adalah memungkinkan penampilan yang bersifat dinamis, misalnya menampilkan jam yang terus menerus berubah ataupun untuk membuat animasi gambar yang mengikuti gerakan penunjuk *mouse*.

Dengan memperluas kemampuan HTML, yakni dengan menggunakan perangkat lunak tambahan, perubahan informasi dalam halaman-halaman *Web* dapat ditangani melalui perubahan data, bukan melalui perubahan program. Sebagai implementasinya, aplikasi *web* dapat dikoneksikan ke basis data. Dengan demikian perubahan informasi dapat dilakukan oleh operator atau yang bertanggung jawab terhadap kemutakhiran data, dan tidak menjadi tanggung jawab pemrogram atau *web* master.

2.3. *Internet*

Internet adalah suatu jaringan global yang terbentuk dari jaringan-jaringan lokal dan regional, memungkinkan komunikasi data antar komputer-komputer yang

terhubung ke jaringan tersebut. *Internet* awalnya merupakan suatu rencana dari Departemen Pertahanan Amerika Serikat (*US Departmentt of Defense*) pada sekitar tahun 60'an. Dimulai dari suatu proyek yang dinamakan ARPANET (*Advanced Research Projects Agency Network*). Proyek ini telah berhasil menghubungkan 4 komputer di lokasi Universitas yang berbeda yaitu UCLA, Standford, UC Santa Barbara dan University of Utah. Perkembangan ARPANET ini sangat pesat jika dilihat perkembangan komputer pada saat itu. Sebagai gambarannya pada tahun 1977, ARPANET telah menghubungkan lebih dari 100 mainframe komputer dan sekitar tahun 1988 terdapat 4 juta host jaringan yang terhubung pada jaringan ini. Jumlah sebenarnya dari komputer yang terhubung tidak dapat diketahui dengan pasti, karena perkembangan jumlah komputer yang terhubung dengan suatu jaringan ini semakin lama semakin besar. Karena perkembangannya sangat pesat, jaringan komputer ini tidak dapat lagi disebut sebagai ARPANET karena semakin banyak komputer dan jaringan-jarinagn regional yang terhubung. Konsep ini yang kemudian berkembang dan dikenal sebagai konsep *Internetworking*. Oleh karena itu istilah Internet menjadi semakin populer, dan orang menyebut jaringan besar komputer tersebut dengan istilah Internet.

2.4. Pengembangan Perangkat Lunak

Untuk menyelesaikan masalah aktual di dalam sebuah seting industri, rekayasa perangkat lunak atau tim perekayasa harus menggabungkan strategi pengembangan yang melingkupi lapisan proses, metode, dan alat-alat bantu serta

fase-fase generik. Strategi ini sering di acukan sebagai model proses atau paradigm rekayasa perangkat lunak.

Sebuah perangkat lunak yang disimpan atau digunkan dalam jangka waktu yang cukup lama akan menimbulkan masalah apabila sering diabaikan atau tidak diperhatikan perkembangannya. Permasalahan yang akan timbul dari penggunaan perangkat lunak yang cukup lama adalah:

1. Adanya kebutuhan baru, sebagai contoh pada saat ini berkembangnnya perangkat lunak berbasis web.
2. Untuk mengurangi kompleksitas, biaya serta waktu pemasaran. Sebagai contoh pemanfaatan dari perkembangan bahasa pemograman tingkat tinggi.
3. Mengurangi cacat. Sebagai contoh dengan cara membuat standarisasi dari pengkodean yang tidak sesuai.

Untuk dapat menyelesaikan permasalahan yang timbul diatas, maka diperlukannya melakukan evolusi dari perangkat lunak tersebut yang terdiri dari:

1. Pembangunan Perangkat Lunak

   Yang termasuk dalam pembangunan perangkat lunak itu sendiri meliputi dari daur hidup perangkat lunak, yaitu : permintaan (*requirement*), spesifikasi, perancangan (*design*), pengujian (*testing*) dan sebagainya.

2. Perawatan Perangkat Lunak

   Merupakan prosesproses untuk memperpanjang waktu penggunaan sistem perangkat lunak yang ada, sehingga tetap dapat dipergunakan sebagaimana mestinya.

3. Migrasi Perangkat Lunak

   Merupakan proses-proses yang memindahkan sistem yang ada ke sistem yang baru dikarenakan perkembangan dari kebutuhan dari perangkat lunak tersebut.

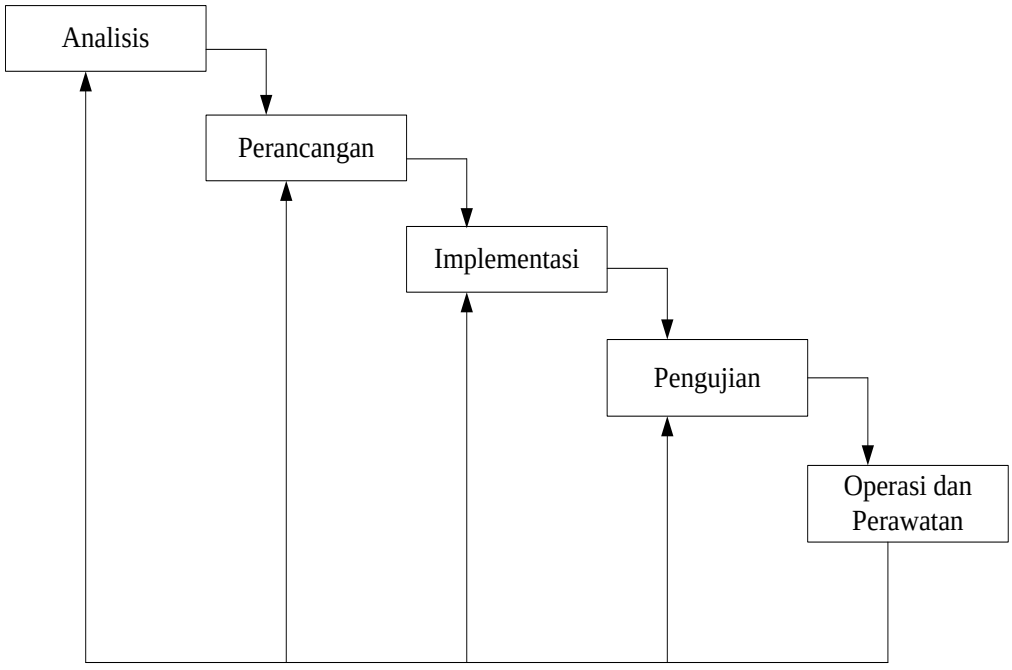
2.5. Model Proses Perangkat Lunak

Model proses perangkat lunak merupakan representasi abstrak dari proses perangkat lunak. Setiap model proses merepresentasikan suatu proses dari sudut pandang tertentu sehingga hanya memberikan informasi parsial mengenai proses tersebut.

Racoon mengusulkan sebuah "model Chaos" yang menggambarkan "perkembangan perangkat lunak sebagai kesatuan dari pemakai ke pengembang dan teknologi." Masing-masing model sudah ditandai dengan cara tertentu sehingga (diharapkan) bisa membantu di dalam control dan koordinasi dari proyek perangkat lunak yang nyata. Dengan demikian, pada intinya, semua model menunjukkan karakteristik dari model Chaos. Adapun model proses perangkat lunak yang dugunakan pada sistem yang dibangun ialah :

2.5.1 Model *Waterfall* (Model Air Terjun/Model Sekuensial Linier)

Model *waterfall* adalah paradigm rekayasa perangkat lunak yang paling luas dipakai dan paling tua. Model *waterfall* mengusulkan sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dan sekuensial yang dimulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Setelah setiap tahap didefinisikan, tahap tersebut 'diakhiri' (*signed off*) dan pengembangan berlanjut ke tahap berikutnya. Adapun Tahapan-tahapan Model waterfall dapat dilihat pada Gambar 2.2. dibawah ini.



Gambar 2.2 Model *waterfall*

2.6. Tahap Perancangan

Tahap perancangan disebut juga tahap pemecahan masalah, yaitu dengan menyusun suatu algoritma, alur sistem, masukan, prosedur proses, keluaran, dan database. Proses perancangan diperlukan untuk menghasilkan suatu rancangan sistem yang baik, karena dengan rancangan yang tepat akan menghasilkan sistem yang stabil dan mudah dikembangkan di masa mendatang. Berikut ini akan dijelaskan rangkaian atau ruang ingkup sistem yang akan dirancang dengan memanfaatkan alat bantu seperti :

2.6.1. Unified Modeling Language

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML tidak berdasarkan pada bahasa pemrograman tertentu. Standar spesifikasi UML dijadikan standar defacto oleh OMG (Object Management Group) pada tahun 1997.UML yang berorientasikan object mempunyai beberapa notasi standar. Spesifikasi ini menjadi populer dan standar karena sebelum adanya UML, telah ada berbagai macam spesifikasi yang berbeda. Hal ini menyulitkan komunikasi antar pengembang perangkat lunak. Untuk itu beberapa pengembang spesifikasi yang sangat berpengaruh berkumpul untuk membuat standar baru. Dalam UML terdapat terdapat beberapa diagram yaitu :

1. *Use case diagram*.

Use Case dibentuk untuk mendeskripsikan apa yang sistem harus lakukan, bukan bagaimana cara sistem melakukannya. Setiap use case berisi minimal: nama yang mencerminkan gunanya dan skenario. Selain itu use case juga dapat berisi:

a. ID unik guna memudahkan traceability
b. Tujuan, yaitu kalimat sederhana yang menyatakan guna dari use case tersebut.
c. Aktor, yaitu pihak-pihak yang bersangkutan untuk mengembangkan use case tersebut.
d. Prioritas, yaitu tingkat kepentingan dari use case
e. Resiko, yaitu tingkat resiko perubahan atau kegagalan saat pengembangan use case
f. Asumsi, yaitu asumsi-asumsi yang dipakai
g. Preconditions, yaitu: kondisi-kondisi yang harus dipenuhi sebelum mengembangkan use case
h. Postconditions, yaitu: kondisi-kondisi yang harus dipenuhi sebelum use case diselesaikan
i. Outstanding issues, yaitu: daftar item-item yang harus ada sebelum use case dapat dianalisis dan didesain
j. Requirements satisfied, yaitu: hal-hal yang harus dipenuhi oleh usecase tersebut.

Use case sebetulnya hanyalah sebuah teknik untuk mendeskripsikan kebutuhan. *Use case* harus bisa dimengerti oleh kedua belah pihak, yaitu *stakeholders* atau *client* dan staff pengembang, oleh karena itu *use case* dibuat se-fleksibel

mungkin. Sebuah *use case* dapat berisi beberapa skenario. Skenario utama, disebut juga *happy path*, adalah skenario dari hal yang umumnya terjadi sedangkan ada skenario lain yang mendeskripsikan jika terjadi hal-hal yang berbeda. Skenario diharapkan deskriptif, yaitu mendeskripsikan dialog antara aktor dan sistem. Hubungan antar *use cases* digambarkan dalam suatu *use case* diagram. Kumpulan lengkap antara *use cases*, aktor, dan diagram form disebut use case model.

2. *Sequence diagram*

Sequence diagram untuk use case adalah diagram interaksi yang mengekspresikan interaksi antara aktor dan sistem dengan penekanan waktu. Setiap skenario pada use case harus dibuat sequence diagramnya. Yang dideskripsikan adalah keinginan aktor, bukan mekanisme implementasi. Sequence diagram menunjukkan hubungan dinamis antara objek. Setiap sequence diagram hanya mendeskripsikan satu skenario.

3. *Use case analysis*

Use case analysis dimulai dengan mengelaborasikan sequence diagram dari use case dasar dengan analysis-object model.Analysis objects adalah instansiasi dari analysis classes yang merepresentasikan hal-hal umum dari sistem yang memiliki sifat dan tugas. Nama-nama class pada tahap ini harus memiliki hubungan yang erat dengan domain. Ada tiga macam tipe *analysis classes*, yaitu:

a. *Boundary objects*, merepresentasikan antar muka antara aktor dan sistem. Dalam sebuah aplikasi web, biasanya berbentuk halaman web.

b. *Entity objects*, adalah sesuatu yang tetap ada setelah kejadian-kejadian pada use case.

c. *Control objects*, hal ini merepresentasikan proses.

Hal untuk dicatat: dalam tahap pencarian kebutuhan, diagram yang harus adahanya diagram usecase. Dua diagram terakhir adalah opsional.

2.6.2. Kamus Data

Kamus data (KD) atau *data dictionary* (DD) atau disebut juga dengan istilah *system data dictionary* adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan menggunakan kamus data, analisis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. Kamus data dibuat pada tahap analisis sistem dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem.

Pada tahap analisis dan perancangan, kamus data dapat digunakan sebagai alat komunikasi antara analsisis sistem dengan pemakai sistem tentang data yang mengalir di sistem, yaitu tentang data yang masuk ke sistem data tentang informasi yang dibutuhkan oleh pemakai sistem.

Pada tahap perancangan, kamus data digunakan untuk merancang *input*, merancang laporan-laporan dan *database*. Kamus data dibuat berdasarkan arus data yang ada di DFD. Arus data di DFD sifatnya global, hanya ditunjukkan nama arus datanya saja. Keterangan lebih lanjut tentang struktur data dari arus data di DFD secara lebih rinci dapat dilihat di kamus data. Kamus data harus dapat mencerminkan

keterangan yang jelas tentang data yang dicatatnya, maka kamus data harus memuat nama arus data, alias, bentuk data, arus data, penjelasan, periode, volume, dan struktur data (Jogiyanto, 2005).

2.7. Konsep Dasar Basis Data

Menurut FAT, Basis data terdiri dari 2 kata, yaitu basis dan data. Basis dapat diartikan sebagai markas, gudang, tempat berkumpul atau tempat penyimpanan. Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia, barang, hewan, peristiwa, konep keadaan, dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya. Basis data merupakan himpunan kelompok data yang saling berkaitan dan tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat dieksplorasi untuk menyusun informasi – informasi dalam berbagai bentuk.

Prinsip utama basis data adalah pengaturan data atau arsip dengan tujuan utama untuk kemudahan dan kecepatan dalam pengambilan kembali data atau arsip tersebut. Dapat disimpulkan bahwa basis data memiliki kesamaan fungsi dan prinsip kerja dengan lemari arsip, perbedaan antara basis data dengan lemari arsip hanya terletak pada media penyimpanan yang digunakan.

2.7.1. Sistem Hierarki Basis Data

Pembangunan sistem informasi sangat bertumpu pada kualitas basis data yang disusun dan dibentuk diharapkan memiliki sifat efektif dan efisien dalam pengorganisasiannya, bebas redudansi, fleksibel, dan sistem database yang dapat

diakses secara bersamaan dalam lingkungan jaringan. Data dalam basis data disusun berdasarkan sistem hierarki SUT , yaitu :

1. Database, merupakan kumpulan file yang saling terkait satu sama lain.
2. File, yaitu kumpulan dari record yang saling berkaitan dan memiliki format field yang sama dan sejenis.
3. Record, yaitu kumpulan field yang menggambarkan suatu unit data atau individu tertentu.
4. Field, yaitu atribut dari record yang menunjukan suatu item dari data seperti nama, alamat, dan sebagainya.
5. Byte, adalah atribut dari field yang berupa huruf yang membentuk nilai dari sebuah field.
6. Bit, yaitu bagian terkecil dari data secara keseluruhan berupa karakter ASCII nol atau satu yang merupakan komponen pembentuk byte.

2.7.2. Perancangan Basis Data

Perancangan basis data merupakan salah satu bagian terpenting dalam proses pengembangan sistem. Perancangan basis data mempunyai beberapa tujuan yaitu :

1. Menghilangkan redundansi data.
2. Meminimumkan jumlah relasi di dalam basis data.
3. Membuat relasi berada dalam bentuk normal, sehingga dapat meminimumkan permasalahan berkenaan dengan penambahan, pembaharuan dan penghapusan.

Basis data adalah satu kelompok organisasi data yang terpusat. Basis data secara umum dianalogikan sebagai lemari dokumen atau sekumpulan lemari dokumen. Pada tahap ini digunakan beberapa peralatan untuk mendukung proses pembentukan database tersebut, peralatan–peralatan yang digunakan untuk mendukung pembentukan basis data adalah :

(1). Normalisasi

Menurut SUT, normalisasi adalah peralatan yang dipergunakan untuk melakukan proses pengelompokan data yang menjadi menjadi tabel – tabel yang menunjukan entitas dan relasinya. Normalisasi merupakan cara pendekatan lain yang dalam membangun desain lojik basis data relasional dengan menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal. Secara umum proses normalisasi terdiri dari dalam tahap, yaitu :

a. Tahap tidak normal.

Pada tahap ini, semua data yang ada direkam tanpa format tertentu. Hal tersebut dapat menyebabkan data mengalami duplikasi.

b. Normalisasi tahap 1

Normalisasi tahap 1 menghilangkan duplikasi data yang terjadi pada tahap tidak normal dengan cara menghapus data-data yang sama.

c. Normalisasi tahap 2.

Tahap normalisasi 3 adalah menentukan kunci dari normalisasi 1 yang akan digunakan sebagai *primary key* pada tabel, membentuk tabel berdasarkan *primary key* dan mengelompokkan data pada tabel-tabel yang sudah dibentuk.

d. Normalisasi tahap 3.

Pada tahap 3 dilakukan penentuan relasi antar tabel sehingga memungkinkan adanya field kunci sekunder.

(2). Skema Relasi

Menurut Budi Sutedjo, Skema Relasi adalah peralatan yang berfungsi untuk menggambarkan relasi atau hubungan dari dua file atau dua tabel yang dapat digolongkan dalam tiga macam bentuk relasi, yaitu satu – satu (1 – 1), satu – banyak (1 – N), banyak – banyak (N – N). Model entity relationship yang berisi komponen – komponen himpunan entitas dan himpunan relasi yang masing–masing dilengkapi dengan atribut–atribut yang merepresentasikan seluruh fakta yang kita tinjau digambarkan lebih sistematis dengan menggunakan Diagram E–R. Pembuatan Diagram E–R menurut SUT meliputi tahap – tahap berikut:

a. Mengidentifikasi dan menetapkan seluruh himpunan entitas yang akan terlihat.
b. Menentukan atribut – atribut *key* dari masing – masing himpunan entitas.
c. Mengidentifikasi dan menetapkan seluruh himpunan relasi di antara himpunan entitas–himpunan entitas yang ada beserta *foreign key* yang terdapat pada relasi tersebut.
d. Menentukan derajat atau kardinalitas relasi untuk setiap himpunan relasi.
e. Melengkapi himpunan entitas dan himpunan relasi dengan atribut – atribut deskriptif (non *key*).

(3). Tabel Relasi

Tabel relasi adalah suatu perangkat yang digunakan untuk menggambarkan representasi struktur dan data dari hubungan antar tabel secara fisik atau nyata yang terjadi pada sistem informasi.

(4). Struktur File

Perangkat yang digunakan untuk memaparkan identitas setiap file mulai dari nama file, jenis, fungsi, dan isi dari file tersebut dalam bentuk tabel kamus data yaitu tabel yang berisi nama field, type field tersebut, ukuran field, dan keterangan dari field.

2.7.3. Konsep Sistem Basis Data

Basis data adalah sebuah objek pasif atau mati yang ada karena ada yang membuat dan hanya berguna jika ada program aplikasi yang mengelola atau menggerakkannya. Program aplikasi dan basis data yang tergabung akan menghasilkan sebuah sistem. Secara umum Sistem Basis Data merupakan sistem yang terdiri atas kumpulan file (tabel) yang saling berhubungan dalam sebuah basis data di sebuah sistem komputer dan sekumpulan program DBMS (*database management system*) yang memungkinkan beberapa pemakai atau program lain untuk mengakses dan memanipulasi file – file tersebut.

Secara lengkap, dalam sebuah sistem basis data akan terdapat komponen – komponen utama sebagai berikut:

(1). Perangkat Keras (*hardware*)

(2). Sistem Operasi (*Operating System*)

(3). Basis Data (*Database*)

(4). Sistem (Aplikasi/Perangkat lunak) Pengelola Basis Data (DBMS)

(5). Pemakai (*user*)

(6). Aplikasi (Perangkat Lunak) lain (bersifat opsional)

2.7.4. Tujuan Basis Data

Basis Data (Database) pada prinsipnya ditujukan untuk pengaturan data agar terdapat kemudahan dalam pengambilan kembali data tersebut. Berikut ini terdapat beberapa tujuan dari Basis Data diantaranya yaitu :

(1). Kecepatan dan kemudahan (Speed)

(2). Efisiensi ruang penyimpanan (Space)

(3). Keakuratan (Accuracy)

(4). Ketersediaan (Avaibility)

(5). Kelengkapan (Completeness)

(6). Keamanan (Security)

(7). Kebersamaan (Sharability)

2.8. PHP

PHP (Hypertext processor) merupakan sebuah bahasa *server side scripting* yang sebagian sintaksnya mirip dengan bahasa pemograman C, Java, dan Perl yang ditambah dengan beberapa fungsi PHP yang spesifik (PHP documentation group, 2004).

2.8.1. Kelebihan dari PHP

Kelebihan dari pemakaian PHP sebagai bahasa pemograman aplikasi *web* yaitu :

2.8.2. Kepraktisan dari PHP

PHP diciptakan dengan kepraktisan sebagai bagian dalam perancangannya. Tujuan awal dari pembuatnya bukan untuk membuat bahasa yang baru, namun untuk menyelesaikan permasalahan yang saat itu tidak memiliki solusi yang siap langsung digunakan, lebih jauh banyak dari evolusi awal php bukanlah hasil dari tujuan terpisah yakni untuk memperbaiki bahasa itu sendiri, tetapi lebih bertujuan untuk meningkatkan fungsionalitas bahasa PHP terhadap penggunanya. Hasilnya adalah sebuah bahasa yang minimalis, baik dalam hal kebutuhan pengguna dan kebutuhan sintaks bahasa. Bagi pemula, sebuah script untuk menghasilkan suatu output dalam bahasa PHP dapat berjumlah sesedikit mungkin, yakni hanya satu baris. Tidak seperti bahasa C, yang mewajibkan menggunakan library tambahan. Berikut ini adalah contoh penggunaan *script* php untuk menampilkan tanggal saat ini.

```
<?php echo date("F j, Y"); ?>
```

Contoh lain yang menggambarkan kecenderungan PHP dalam hal kesederhanaan yaitu kemampuannya dalam *nested function* (fungsi bersarang). Sebagai contoh , pengguna dapat memberikan perubahan pada sebuah nilai dibaris yang sama dengan menumpukkan fungsi pada urutan tertentu, pada kasus berikut yang menghasilkan *pseudorandom string* yang terdiri dari lima karakter *alphanumeric*, a3jh8 sebagai contohnya,

```
<?php $randomstr = substr(md5(microtime()), 0, 5); ?>
```

PHP adalah sebuah bahasa yang tidak memiliki aturan yang ketat, yakni tidak dibutuhkan untuk membuat, mengganti *type* atau menghapus sebuah *variable* secara terpisah, walau pengguna tidak dicegah untuk melakukannya. PHP menangani masalah itu secara internal, membuat *variable* pada saat mereka dipanggil dalam sebuah *script*, dan menggunakan sebuah formula secara otomatis mengubah tipe sebuah *variable*. Sebagai contoh, PHP menganggap statement berikut ini adalah benar.

```
<?php
$number = "5"; /* number adalah sebuah string */
$sum = 15 + $number; /*menjumlahkan integer dengan
string untuk menghasilkan integer */
$sum = "twenty"; /*menimpa $sum dengan sebuah string*/
?>
```

PHP juga secara otomatis menghancurkan *variable* dan mengembalikan sumber daya yang digunakan perangkat keras kepada sistem ketika *script* selesai digunakan, pada kasus ini dan kasus – kasus lainnya, dengan mencoba untuk menangani banyak aspek administratif programming secara internal, PHP memungkinkan *developer* untuk lebih berkonsentrasi penuh pada aplikasi yang dikerjakannya.

2.8.3. Kekuatan

Pada awal pengenalan PHP5 secara tidak langsung mengarahkan pada fakta bahwa versi terbaru ini lebih kualitatif daripada kuantitatif dari versi sebelumnya. Pada versi major sebelumnya (PHP4) dilengkapi dengan banyak tambahan kepada

default library PHP, untuk memingkatkan beberapa ratus fungsi per-*release*. Saat ini terdapat 113 *library* yang tersedia, yang secara bersama mengandung lebih dari 1000 fungsi. Walaupun pengguna mengetahui kemampuan PHP untuk menggunakan *database*, memanipulasi informasi *form*, membuat halaman secara dinamis, pengguna mungkin mengetahui php mampu melakukan hal-hal sebagai berikut:

a. Membuat dan memanipulasi Macromedia Flash, Gambar, dan file PDF.
b. Mengevaluasi *password* untuk pengenalan pola dengan membandingkannya dengan kamus bahasa dan secara mudah menemukan sebuah pola.
c. Berkomunikasi dengan *Lightweight Directory Access Protocol* (LDAP)
d. Mengartikan kalimat bahkan yang paling kompleks dengan menggunakan *Reguler Expression* baik *POSIX* atau *PERL*.
e. Melakukan *authentifikasi User* terhadap informasi *login* yang disimpan dalam file biasa, *database*, maupun *Microsoft Active Directory*.
f. Berkomunikasi dengan banyak jenis protokol, termasuk IMAP, POP3, NNTP, DNS, dan lain sebagainya.
g. Berkomunikasi dengan banyak jenis solusi pemroses kartu kredit

2.9. MVC

MVC (*Model View Controller*) merupakan sebuah metode untuk mengorganisasikan sebuah *website* dinamis (Upton, 2007). Dengan penjelasan tiap bagian sebagai berikut :

a. *Models* adalah *object* yang merupakan representasi dari data pokok.

b. *View* menampilkan keadaan dari *model*. Komponen ini bertanggungjawab untuk menampilkan informasi pada pengguna akhir.

c. *Controller* menawarkan untuk mengubah keadaan *model*. *Controller* juga bertugas untuk mencari informasi dari *model* dan menyediakan data dinamis untuk *views*.

Hasilnya ketika terjadi permintaan (*request*) dari user / pengguna dilakukan proses dengan urutan sebagai berikut :

a. *Client* melakukan proses *request* terhadap *Controller* melalui *dispatcher*.

b. *Dispatcher* menunjuk *controller* yang sesuai dengan *request client*

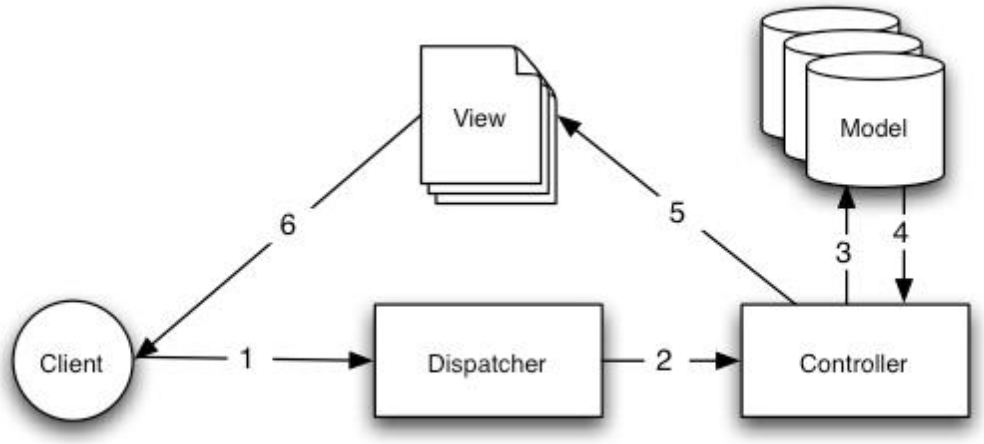
c. *Controller* merespon dengan meminta data dari *Model*.

d. *Model* memberikan data yang diminta oleh *controller*.

e. *Controller* mengolah tampilan dan memberikannya pada *View*.

f. *View* mengembalikan tampilannya kepada pengguna yang meminta layanan.

Alur MVC (Model View Controller) sendiri dapat terlihat pada Gambar 2.3 di bawah ini.



Gambar 2.3 Alur MVC

(sumber http://book.cakephp.org)

Pemisahan aplikasi menjadi tiga buah komponen memberikan beberapa kelebihan diantaranya adalah :

a. Meningkatkan skalabilitas (kemampuan aplikasi untuk berkembang), jika aplikasi menjadi lambat disebabkan karena lambatnya akses database, sebagai contoh, anda dapat meng upgrade hardware yang menjalankan database tanpa ada efek terhadap komponen lainnya.

b. Membuat perawatan lebih mudah, karena tiap komponen memiliki ketergantungan rendah

2.10. *jQuery*

jQuery adalah sebuah *framework javascript* yang bersifat *unobstrusive* (yang tidak membutuhkan teknik binding event pada HTML). *jQuery* memungkinkan

penggunanya untuk memodifikasi bagian dari elemen HTML menggunakan *CSS (Cassade Style Sheet)* selector.

Fungsi utama dari *jQuery* dapat disebut $, yang digunakan untuk menggunakan *CSS (Cassade Style Sheet)* selector dan ditempatkan didalam *function document.ready*() seperti ini :

```
$(document).ready(function(){
//script diletakkan disini
});
```

Penulisan ini mengakibatkan *script* (*jQuery* dan *script* lainnya) dijalankan ketika semua kode HTML telah di-*load* pada *server client*.

2.10.1. Menggunakan *jQuery*

Jika pengguna ingin menggunakan *jQuery* untuk mendapatkan *element* berdasarkan ID, pengguna cukup mengetikkan sebagai berikut

```
Var money = $(#money);
```

menunjukkan sebuah *selector* ID pada CSS atau mendapatkan *element* dapat juga dilakukan dengan cara

```
Document.getElementById("money");
```

Untuk mendapatkan grup dari element berdasarkan nama *tag* pengguna cukup memberikan $ pada *element* tipe *selector* :

```
Var paragraphs = $("p"); 12
```

Dan untuk mendapatkan group dari *element* berdasarkan nama *Class*, pengguna cukup menggunakan *Class Selector*

```
Var tables = $("#content table.datatable");
```

Tables adalah sebuah *array* dari *element table* yang merupakan turunan dari element dengan *ID content*, dan yang memiliki sebuah *Class datatable*.

2.10.2. Kelebihan *jQuery*

Kelebihan dari penggunaan *jQuery* dapat dilihat dari adanya kemudahan sebagai berikut :

a. Mengakses bagian dari halaman. Tanpa adanya *library javascript*, banyak baris *code* harus ditulis untuk menjelajahi *tree document object model* (DOM), dan untuk mendapatkan bagian tertentu dari struktur dokumen HTML. j*Query* menawarkan mekanisme *selector* yang lebih efisien untuk mengambil bagian dari dokumen secara tepat yang akan diperiksa atau dimanipulasi.

b. Memodifikasi tampilan dari sebuah halaman. CSS menawarkan metode yang kuat dari mempengaruhi cara bagaimana sebuah dokumen di tampilkan, tetapi kemudian gagal ketika banyak *web browser* tidak mengikuti standar yang sama. *jQuery* dapat menjembatani hal ini dengan memberikan standar yang sama dengan semua *browser*. Sebagai tambahan, *jQuery* dapat mengubah atau lebih *class* yang diterapkan pada satu bagian dari dokumen bahkan selama halaman tersebut ditampilkan.

c. Mempengaruhi isi dari sebuah halaman, tidak terbatas hanya perubahan tampilan, *jQuery* dapat mengubah isi dari dokumen itu sendiri hanya dengan beberapa perintah sederhana. *Teks* dapat diubah, gambar dapat disisipkan atau

ditukar, *list* dapat diatur kembali, atau seluruh struktur dari html dapat ditulis ulang dan diperluas, semua dengan API yang mudah untuk digunakan.

d. Merespon interaksi pengguna didalam halaman. Interaksi yang paling kompleks tak berguna jika tidak dapat dikendalikan kapan waktunya. Pustaka *jQuery* menawarkan cara yang lebih elegan untuk menangkap banyak jenis dari *event*, seperti jika pengguna menekan sebuah *link*, tanpa harus menambahkan kode HTML pada *event handler*. Pada saat yang bersamaan, API *event handling* menghancurkan ketidakkonsistensian *browser* yang sering mengganggu pengembang *web*.

e. Menambah animasi pada sebuah halaman. Untuk secara efektif mengimplementasi interaktifitas, seorang desainer juga menyediakan respon balik secara visual kepada pengguna. Pustaka *jQuery* memfasilitasi hal tersebut dengan menyediakan banyak efek seperti timbul – hilang, pergeseran, dan juga *toolkit* untuk membuat sesuatu yang baru.

f. Mengambil informasi dari *server* tanpa harus melakukan load kembali halaman, pola kode seperti ini yang dikenal sebagai *Asynchronous javascript and xml (AJAX)*, dan membantu pengembang web dalam membuat sebuah situs yang *responsive* dan kaya akan fitur. Pustaka jQuery menghapus kompleksitas dari *browser* tertentu dalam proses ini, yang memungkinkan pengembang untuk fokus pada fungsionalitas *server*.

g. Menyederhanakan tugas *jQuery* umum. Sebagai tambahan dari fitur spesifik dari *jQuery*, pustaka ini menyediakan perbaikan kepada *javascript* dasar untuk membangun hal hal, seperti *iterasi* dan manipulasi *array*.

2.11. MySQL

MySQL adalah *database* relasional buatan MySQL AB yang dapat didapatkan secara gratis, dan juga memiliki kemampuan yang dapat digunakan untuk perusahaan sekelas *enterprise*, dan tiap versi terbarunya semakin mendekati standar SQL – 92, sehingga mudah digunakan (Gilmore, 2006).

2.11.1. Fleksibilitas

Pada saat ini terdapat 12 buah versi *biner* MySQL untuk berbagai macam platform, termasuk DEC OSF, FreeBSD, IBM AIX, HP-UX, Linux, Mac OS X, Novell Netware, OpenBSD, QNX, SGI IRIX, Solaris, dan Microsoft windows. Dan MySQL juga menyediakan *source*-nya tersedia untuk di-*download* jika file binernya tak tersedia untuk platform yang pengguna inginkan, atau jika pengguna ingin mengkompilasi sendiri.

MySQL juga memiliki API yang banyak tersedia untuk digunakan oleh berbagai macam bahasa pemograman, diantaranya C, C++, PERL, Ruby, PHP, dan Java. MySQL juga memiliki banyak cara untuk menangani data yang disimpan didalamnya (storage engine). Alasannya ada banyak cara tersebut dikarenakan adanya kebutuhan untuk algoritma penyimpanan/penanganan terhadap kasus – kasus tertentu. Beberapa *storage engine* yang dipakai MySQL adalah MyISAM, BDB, HEAP (yang berubah menjadi MEMORY pada versi 5), InnoDB, ISAM, dan MERGE. Dikarenakan pada satu *database* dapat berisi lebih dari satu *table*, dimana *tabel – table* tersebut memiliki fungsi tersendiri pula, maka MySQL memungkinkan untuk penggunaan *storage engine* yang berbeda antar tabel.

2.11.2. Kekuatan

MySQL memliki beberapa kekuatan yang membuatnya layak untuk diimplementasikan pada pembuatan *aplikasi*, diantaranya yaitu :

2.11.2.1. Fitur SQL untuk kelas Enterprise

Pada permulaannya MySQL dianggap tidak cocok untuk diterapkan pada level *enterprise*, dikarenakan tidak mempunyai fitur *stored procedure*, views, dan *subquery*. Namun pada MySQL versi 5 telah menjadi pembuktian bahwa MySQL dapat diadopsi ketingkat *enterprise*, dengan adanya fitur yang tidak ada sebelumnya.

2.11.2.2.Pengindeksan dan pencarian terhadap teks

Pada MySQL ver 3.23.23 telah ditambahkan kemampuan untuk pencarian dan pengindeksan terhadap kolom bertipe *string (CHAR, TINYTEXT, TEXT, MEDIUMTEXT, dan LONGTEXT)*. Fitur ini meningkatkan performa ketika melakukan pencarian, dalam hal pencarian berdasarkan kesamaan dengan isi dari tiap baris.

2.11.2.3.*Query Caching*

Query caching adalah salah satu cara MySQL untuk meningkatkan kecepatannya, dimana jika diaktifkan maka ketika terdapat terdapat perintah *SELECT MySQL* akan menyimpan *query* tersebut beserta hasilnya dimemory, maka apabila terdapat pemanggilan *query* yang kedua maka MySQL akan menggunakan *query* yang sebelumnya, untuk menghilangkan ketidak terbaruannya data *MySQL* menggunakan tehnik khusus untuk membuat ulang *cache* dan menghilangkan setelah *request* selesai.

2.11.2.4 *Replikasi*

Pada versi 3.2.1.15 MySQL telah menambahkan fitur *replikasi*. *Replikasi* memungkinkan *database* yang ditaruh dalam satu server *MySQL* untuk duplikasi ke yang lainnya, dimana dapat memberikan banyak keuntungan, sebagai contoh, dengan adanya *database* replikasi ditempat lain dapat meningkatkan ketersediannya, dimana jika *database* utama mengalami masalah maka dapat langsung menggantikannya. Dapat melakukan *query* dari *client* dapat dibagi ke *database* yang berbeda sehingga terjadi pembagian beban ke-*database* lainnya. Keuntungan lainnya yaitu ketika terjadi backup data aplikasi menjadi *offline* dan menunggu proses *backup* selesai, pengguna dapat menjalankan *backup* pada *database* replikasinya, dimana hal ini memungkinkan waktu *aplikasi* ketika *down* mendekati 0.

2.11.2.5. Keamanan

MySQL memiliki banyak jenis tehnik pengamanan dan pilihan konfigurasinya. Yang memungkinkan penggunanya untuk mengontrol semua aspek yang berhubungan dengan cara kerja *MySQL*. Sebagai contoh dengan pilihan konfigurasi pengguna dapat melakukan hal – hal seperti :

a. Pemilik daemon dapat menggunakan bahasa, *port* standar, lokasi penyimpanan data *MySQL*, dan pilihan global lainnya.
b. Banyak memori yang digunakan pada tiap *thread*, *cache* dari *query*, *table temporer*, *table joins*, dan *buffer* dari *index*.
c. Bermacam aspek dari kemampuan jaringan MySQL, termasuk lama waktu yang diijinkan untuk melakukan sebuah koneksi sebelum memutuskan koneksi tersebut,

apakah untuk mencoba untuk mencari nama dari DNS, besarnya paket data yang diizinkan, dan lain sebagainya.

Pilihan konfigurasi keamanan yang dimiliki oleh *MySQL* juga sangatlah bagus, yang memungkinkan penggunanya untuk melakukan hal – hal seperti :

a. Banyaknya jumlah *query*, *update*, dan koneksi yang diijinkan tiap jam
b. Apakah seorang *user* harus menggunakan sertifikat SSL yang sah untuk terhubung ke suatu *database*.
c. Tindakan yang dapat dilakukan seorang pengguna terhadap suatu *database*, table, bahkan kolom. Dimana seorang user mungkin dapat meng-*update* sebuah tabel, namun tidak dapat menghapus data dalam tabel tersebut.

2.12. *CakePHP Framework*

CakePHP framework yang gratis, *opensource*, *rapid development framework* untuk PHP, sebuah pondasi yang tersturktur untuk *programmer* untuk membuat *aplikasi web*. Tujuan utama *CakePHP framework* adalah agar dapat membuat *programmer* bekerja secara terstruktur dan cepat tanpa kehilangan fleksibilitas.

2.12.1. Fitur fitur *CakePHP*

Beberapa fitur fitur yang ditawarkan oleh *cakePHP* adalah,

a. *Web* komunitas yang aktif dan bersahabat
b. *Lisensi* yang fleksibel
c. Kompatibel dengan PHP versi 4 dan PHP versi 5
d. Interaksi *database* dengan *Create Read Update Delete (CRUD)* yang terintegrasi

e. *Aplikasi Scaffolding*

f. Penciptaan Kode

g. Arsitektur *MVC*

h. Request *dispatcher* dengan clean URL, *costum* URL dan *routes*

i. *Buid-in validation*

j. Template yang cepat dan fleksibel

k. *View helpers* untuk AJAX, Javascript, HTML form dan banyak lagi

l. Komponen untuk *email, cookie, security, session*, dan *request handling*

m. *Access control List (ACL)* yang fleksibel

n. *Data Sanitization*

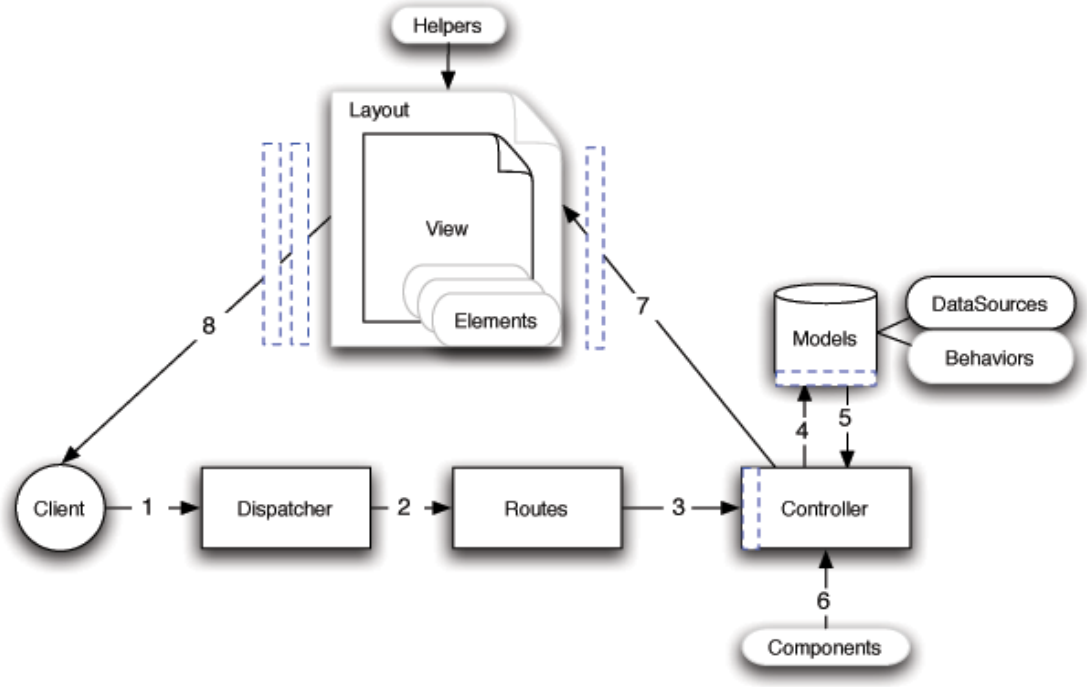
o. *Caching* yang fleksibel

p. *Localization*

q. Dapat bekerja dari semua direktori *website*, dengan sedikit atau tanpa konfigurasi *apache*

2.12.2. Alur MVC pada CakePHP

Konsep *MVC* pada *CakePHP* digambarkan pada Gambar 2.4 dibawah ini,



Gambar 2.4 Alur MVC pada CakePHP
(sumber : http://book.cakephp.org)

Alur MVC pada *CakePHP* pada gambar diatas dijalankan dengan langkah langkah sebagai berikut:

a. *User* mengklik sebuah link, *browser user* membuat sebuah *request* ke *webserver*.

b. *Dispatcher* memparser URL link untuk mengekstrak parameter *request* dimana hasil yang diekstrak ialah *controller*, *action*, dan argument lain yang akan membawa efek kepada *business logic* selama *request* dijalankan.

c. Dengan menggunakan *routes*, sebuah *request* URL ditujukan ke *controller action* (sebuah metode yang berada di *controller* yang telah dispesifikasikan).

Controller before filter() *callback* dipanggil sebelum *controller action logic* dijalankan.

d. *Controller* dapat menggunakan model untuk memperoleh akses ke aplikasi data. Apapun *application* model *callback, behaviors, datasource* dapat digunakan pada operasi ini. Ketika penggunaan model tidak digunakan, semua *controller CakePHP* biasanya membutuhkan paling tidak satu buah model.

e. Setelah model menerima data, data tersebut dikirim ke *controller*.

f. *Controller* dapat menggunakan *components* untuk menyaring data atau melakukan operasi lain, contohnya : *manipulasi session, authentication,* atau mengirim *email*.

g. Setelah *controller* selesai menggunakan *model* dan *component* untuk menyiapkan data, data tersebut dikirim ke *view* dengan menggunakan *controller set method*. *Controller callback* dapat digunakan pada saat data dikirim. *View logic* dilakukan, dimana didalamnya terdapat penggunaan *element* atau *helpers*. Biasanya *view* di *render* didalam *layout*.

h. Hasil dari *view* yang direnderdikirim ke *browser user*.

2.12.3. Contoh Penggunaan *CakePHP*

Cara penggunaan *cakePHP* dimulai dengan membuat *file* (disini akan menggunakan nama post.php) yang diletakkan dalam folder /app/models dengan isi file sebagai berikut :

```
<?php
Class Post extends AppModel{
Var $name = „Post“; }
?>
```

Penggunaan *naming convension* sangat penting pada *CakePHP*. Dengan memberi nama model dengan nama *Post*, *CakePHP* akan secara otomatis merujuk bahwa model ini digunakan pada *PostController*, dan akan ditujukan pada tabel *database* yang bernama *posts*. setelah itu membuat file *controller* dimana semua *business logic* untuk beriteraksi dengan *post* model akan terjadi (disini akan menggunakan nama post_controller.php) yang diletakkan dalam folder /app/controller

```
<?php
class PostsController extends AppController {
var $name = 'Posts';
function index() {
$this->set('posts', $this->Post->find('all'));
}
function view($id = null) {
$this->Post->id = $id;
$this->set('post', $this->Post->read()); }}?>
```

Setelah itu membuat file *view* dimana hasil yang didapat dari *controller* akan ditampilkan ke *browser user* (disini akan menggunakan nama view.ctp) yang diletakkan dalam folder /app/view/post/

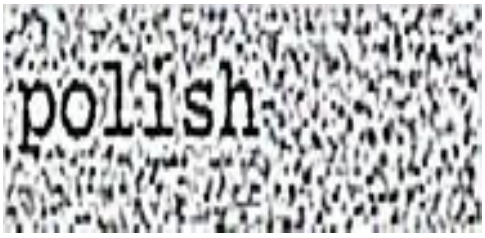
```
<!-- File: /app/views/posts/view.ctp -->
<h1><?php echo $post['Post']['title']?></h1>
<p><small>Created:<?php
echo$post['Post']['created']?></small></p>
<p><?php echo $post['Post']['body']?></p>
```

jika *browser* mengakses url http://localhost/post/ maka *browser* akan menampilkan seluruh data yang ada pada tabel *posts*.

2.13. *CAPTCHA*

CAPTCHA singkatan dari *Completely Automated Public Turing test to tell Computers* and *Humans Apart*. Istilah ini pertama kali digunakan pada tahun 2000 oleh Luis von Ahn, Manuel Blum, Nicholas Hopper dan John Langford dari Carnegie Mellon University. *CAPTCHA* digunakan untuk melakukan *authentifikasi* pada suatu program *web*. *CAPTCHA* memanfaatkan kata-kata yang tidak dapat dibaca dengan OCR (*Optical Computer Recognition*). OCR (*Optical Computer Recognition*) dapat digunakan untuk mengubah hasil *scan* suatu buku (dalam bentuk gambar) ke dalam suatu bentuk dokumen *teks*.

CAPTCHA merupakan metode *challenge* yang banyak dimanfaatkan pada berbagai *aplikasi web* di internet sebagai metode untuk membedakan antara manusia dengan mesin. Contoh penggunaan Captcha dapat terlihat pada Gambar 2.5 di bawah ini.



Gambar 2.5. Contoh Captcha

Seperti kita lihat pada contoh, *CAPTCHA* pada dasarnya adalah sebuah *teks* yang mudah dibaca, bersifat acak dan tak bermakna apapun. Teks tersebut ada pada *background* warna yang sulit dibaca dengan dengan *decipher code*. Sebagai informasi, *decipher code* biasa digunakan untuk memisahkan teks dari gambar.

2.13.1 Tujuan Penggunaan *Captcha*

Adapun tujuan dari penggunaan *Captcha* ialah sebagai berikut,

1. *Security image* adalah suatu bentuk perlindungan terhadap suatu form dari serangan SPAM
2. Penggunaan *captcha* yang *degenerate* secara *random* akan dapat memastikan bahwa pengisi data adalah memang manusia dan bukan *script* (*bot*). Kalau pengisinya adalah *script* maka *submit* datanya akan gagal tidak sampai terekam dalam *database*
3. Ini merupakan sebuah *self test* kepada *user* yang mengakses sebuah situs (yang tersimpan di dalam komputer) bahwa kita adalah manusia dan bukan sebuah "komputer bot". Seperti diketahui, robot atau bot adalah sebuah program mesin yang biasanya digunakan oleh *spammer* untuk mengumpulkan data-data *user* pada sebuah *Web*.

2.14. *Role-Based Access Control* (RBAC)

Mandatory Access Control (MAC) sering digunakan pada dunia militer. MAC adalah suatu teknik pembatasan akses terhadap suatu obyek dengan mendefinsikan berbagai tingkatan sekuriti. Dengan adanya tingkatan sekuriti tersebut maka dapat dibuat suatu aturan dalam mengakses suatu obyek. Tingkatan dari hak akses ditandai dengan label seperti TOP SECRET, SECRET, dan CONFIDENTIAL.

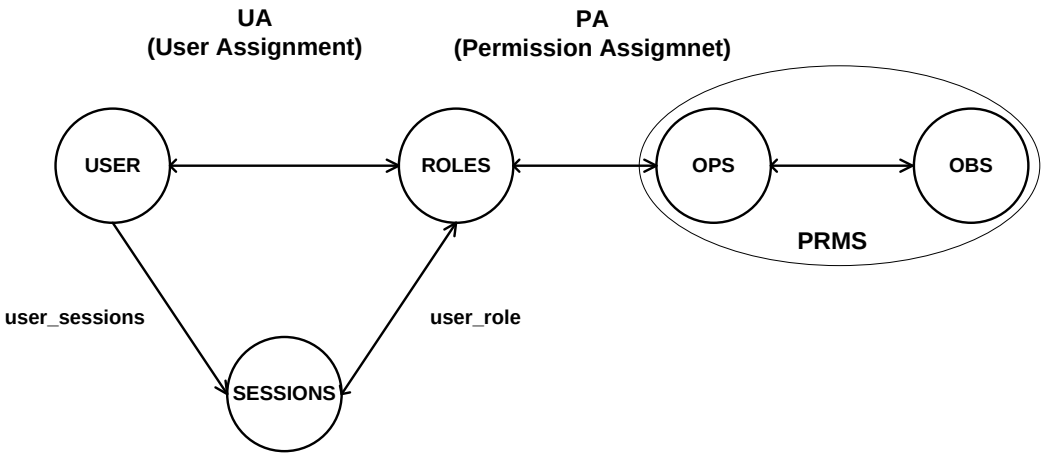
RBAC adalah bentuk dari *Mandatory Access Control*, akan tetapi RBAC tidak berdasarkan pada *multilevel security*, akan tetapi hak akses kontrol berdasakan

dari *role* dari setiap individu yang merupakan bagian dari sebuah organisasi. *Policy* dari setiap organisasi menentukan keanggotaan *role* dan alokasi dari hak setiap *role*. Tidak seperti DAC, pada RBAC *user* tidak dapat mendelegasikan hal akses pada *user* lainnya.

2.14.1. Definisi *Role-Based Access Control* (RBAC)

Konsep RBAC dapat digambarkan seperti pada Gambar 2.6 Ada beberapa definisi yang perlu diketahui sebelum pembahasan tentang konsep RBAC lebih lanjut. Definisi itu adalah :

1. *Role* (ROLES)
2. *User* (USERS)
3. *Permission* (PRMS)
4. *Object* (OBS)
5. *Operation* (OPS)
6. *Sessions* (SESSIONS)



Gambar 2.6 Konsep RBAC

RBAC yang terdiri dari 5 elemen dasar. Secara umum dapat dikatakan bahwa seorang user didaftarkan kedalam sebuah *role*. *Role* dapat diartikan sebagai suatu hubungan *many-to-many* antara *user* dan *permission*. Inti dari RBAC adalah *session* yang merupakaan pemetaan antar seorang *user* dan *subset* dari *role* yang aktif.

1. Role

Pada dasarnya *Role-based Access Control* adalah hak dan ijin diberikan pada *role* bukan pada *user*. *User* memerlukan hak dan ijin secara *virtual* dengan jalan memasukkan *user* tersebut memjadi anggota dari *role* yang bersangkutan. Pada *infrastruktur* UNIX, *root* mempunyai hak akses '*all* or *nothing'* dengan kata lain setiap *root* dapat melakukan pada saja pada sistem. Tanpa pengalaman yang cukup pada seorang administrator (root) dapat menyebabkan kegagalan sistem. Dengan RBAC maka *profile* dari sebuah role ditentukan untuk setiap *user*. Dengan kata lain maka tugas dari administrator sistem tidak pada melakukan perubahan tanpa adanya *role* dari RBAC. *Role* berorientasi pada group, sekumpulan transaksi dibuat. Transakasi disini dapat merupakan obyek yang berupa program yang berhubungan dengan data. Seorang admintrator sekuriti dapat menambah dan menghapus transaksi ke dalam sebuah role atau bahkan menolak user pada suatu role. Dengan mengelompokkan user kedalam role maka ada memudahkan pada proses otorisasi dan kemampuan audit. Hal ini bertolak belakangan dengan *access list* model pada umumnya yang dilakukan dengan jalan mencari seluruh otorisasi yang ada kemudian mengalokasi hak dan ijin untuk *user* tersebut.

2. User

User adalah berarti seorang pengguna komputer. Walaupun secara konsep user dapat diperluas dengan memasukkan user mesin/komputer, jaringan, *intelligent agent*.

3. Permission

Permission adalah persetujuan untuk melakukan suatu operasi pada satu obyek atau lebih pada RBAC. Obyek dapat berupa sebuah program yang dieksekusi. Tipe operasi dan control pada RBAC tergantung pada sistem dimana operasi itu dimplementasikan. Sebagai contoh adalah operasi *read, write,* dan *execute* sedangkan pada manajemen *database* pada operasi dapat dikatakan sebagai *insert, delete*, *append* dan *update*.

2.14.2. Keuntungan *Role-Based Access Control* (RBAC)

Keuntungan utama dari RBAC adalah fleksibilitas dan manajemen yang relatif mudah. Fleksibilitas memungkinkan administrator dapat membuat *privilege* yang seminimal mungkin untuk setiap *user*, menghindari konflik dari tugas antar *user*, pemisahan tugas secara dinamis maupun statis. Administrator dengan mudah dah mendaftarkan atau menolak seorang *user* kedalam sebuah *role* berdasarkan dari tanggung jawab dan tugasnya.

2.15. Pengujian Sistem

Menurut Roger S Pressmane, mengungkapkan bahwa, Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan mempresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean. Pengujian juga dapat diartikan sebagai sebuah proses eksekusi suatu program dengan maksud menemukan kesalahan.

2.15.1. Prinsip-prinsip pengujian

Sebelum mengaplikasikan metode untuk mendesain *test case* yang efektif, perekayasa perangkat lunak harus memahami prinsip dasar yang menuntun pengujian perangkat lunak.

a. Semua pengujian harus dapat ditelusuri sampai ke persyaratan pelanggan.
b. Pengujian itu harus direncanakan lama sebelum pengujian itu mulai.
c. Untuk menjadi paling efektif, pengujian harus dilakukan oleh pihak ketiga yang indefenden.

2.15.2. Pengujian *Black-Box*

Pengujian *black-box* berfokus pada persyaratan *fungsional* perangkat lunak. Dengan demikian, pengujian *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan *fungsional* untuk suatu program. Pengujian *black-box* bukan merupakan *alternative* dari teknik *white-box*, tetapi merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan dari pada metode *white-box*.