BAB II

LANDASAN TEORI

2.1 *Website*

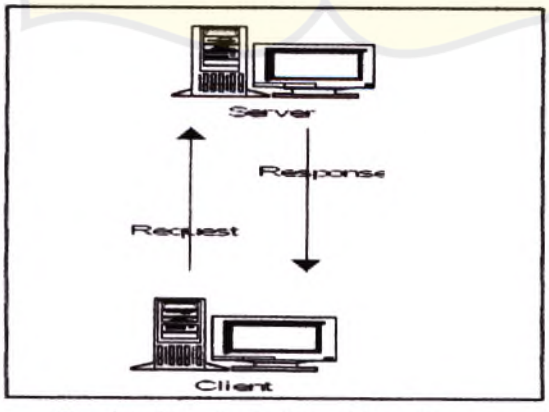
Website adalah kumpulan dari halaman-halaman situs, yang biasanya terangkum dalam sebuah *domain* (nama unik yang diberikan untuk mengidentifikasi nama *server* komputer seperti *web server* atau *email server* di jaringan komputer ataupun internet) atau *subdomain* (bagian dari sebuah nama *domain* induk), yang tempatnya berada di dalam *World Wide Web* (WWW) di *Internet*. Sebuah halaman *web* adalah dokumen yang ditulis dalam format HTML (*Hyper Text Markup Language*) yang hampir selalu bisa diakses melalui HTTP (*Hypertext Transfer Protokol*), yaitu protokol (sebuah aturan atau standar yang mengatur atau mengijinkan terjadinya hubungan, komunikasi, dan perpindahan data antara dua atau lebih titik komputer) yang menyampaikan informasi dari *server website* untuk ditampilkan kepada para pemakai melalui *web browser*. Semua publikasi dari *website-website* tersebut dapat membentuk sebuah jaringan informasi yang sangat besar. (http://id.wikipedia.org/wiki/*Website*, 13 Maret 2010).

2.2 Arsitektur Sistem

Arsitektur dasar dari aplikasi web ini adalah arsitektur *client-server*. Artinya pemrosesan aplikasi ini dijalankan melibatkan kedua sisi yakni sisi mesin *server* pusat dan sisi *client*.

Client-server merupakan sebuah paradigma dalam teknologi informasi yang merujuk kepada cara untuk mendistribusikan aplikasi ke dalam dua pihak: pihak *client* dan pihak *server*.

Dalam model *client-server*, sebuah aplikasi dibagi menjadi dua bagian yang terpisah, tapi masih merupakan sebuah kesatuan yakni komponen *client* dan komponen *server*. Komponen *client* juga sering disebut sebagai *front-end*, sementara komponen *server* disebut sebagai *back-end*. Komponen *client* dari aplikasi tersebut dijalankan dalam sebuah *workstation* dan menerima masukan data dari pengguna. Komponen *client* tersebut akan menyiapkan data yang dimasukkan oleh pengguna dengan menggunakan teknologi pemrosesan tertentu dan mengirimkannya kepada komponen *server* yang dijalankan di atas mesin *server*, umumnya dalam bentuk *request* terhadap beberapa layanan yang dimiliki oleh *server*. Komponen *server* akan menerima *request* dari *client*, dan langsung memprosesnya dan mengembalikan hasil pemrosesan tersebut kepada *client*. *Client* pun menerima informasi hasil pemrosesan data yang dilakukan *server* dan menampilkannya kepada pengguna, dengan menggunakan aplikasi yang berinteraksi dengan pengguna.



Gambar 2.1. Arsitektur *Client/Server*

Sebuah contoh dari aplikasi *client-server* sederhana adalah aplikasi *web* yang didesain dengan menggunakan *Active Server Pages* (ASP) atau PHP. *Script* PHP atau ASP akan dijalankan di dalam *web server* (Apache atau Internet Information Services), sementara *script* yang berjalan di pihak *client* akan dijalankan oleh *web browser* pada komputer *client*. *Client-server* merupakan penyelesaian masalah pada *software* yang menggunakan *database* sehingga setiap komputer tidak perlu diinstall *database*, dengan metode *client-server database* dapat diinstal pada suatu komputer sebagai *server* dan aplikasinya diinstal pada *client*.

a. Server Web

Server web adalah sebuah perangkat lunak *server* yang berfungsi menerima permintaan HTTP atau HTTPS dari *client* yang dikenal dengan *browser* web dan mengirimkan kembali hasilnya dalam bentuk halaman web yang umumnya berbentuk dokumen HTML (http://id.wikipedia.org/wiki/Web_server, 10 Januari 2010). Server web yang terkenal diantaranya adalah Apache dan Microsoft Internet Information Service (IIS). Apache merupakan *server web* antar platform dimana dapat beroperasi di system operasi Windows dan Linux/Unix. IIS hanya dapat beroperasi di sistem operasi Windows.

b. *Server* Basis Data

Server basis data adalah sebuah program komputer yang menyediakan layanan pengelolaan basis data dan melayani komputer atau program aplikasi basis data yang menggunakan model *client/server*.

(http://id.wikipedia.org/wiki/Server_basis_data, 10 Januari 2010). Istilah ini juga merujuk kepada sebuah komputer (umumnya merupakan *server*) yang didedikasikan untuk menjalankan program yang bersangkutan. Sistem manajemen basis data (SMBD) pada umumnya menyediakan fungsi-fungsi *server* basis data, dan beberapa SMBD (seperti halnya MySQL atau Microsoft SQL Server) sangat bergantung kepada model *client-server* untuk mengakses basis datanya.

c. Web Browser

Penjelajah *web* (bahasa Inggris: *web browser*), disebut juga sebagai perambah, adalah perangkat lunak yang berada di *client* berfungsi menampilkan dan melakukan interaksi dengan dokumen-dokumen yang disediakan oleh *server web*. Penjelajah *web* yang populer adalah Microsoft Internet Explorer dan Mozilla Firefox. (http://id.wikipedia.org/wiki/Web_browser, 10-01-2010).

2.3 UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem (Dharwiyanti,Sri & W Satria, Romi. Ilmukomputer.com, 2003).

UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi obyek yang berkembang pesat pada akhir tahun 1980-an dan awal tahun 1990-an. Sejak kehadirannya pada tahun 1997, UML menghancurkan menara Babel tersebut menjadi sejarah. (Martin Fowler, UML DISTILLED,2004).

Martin Fowler dan Steve Mellor menggolongkan tiga cara/metode menggunakan UML:

1. Sketsa

   Developer menggunakan UML untuk membantu menyampaikan beberpa aspek dari sebuah sistem . Sketsa dapat dibuat dalam bentuk *forward engineering* atau *reverse engineering*.

   Forward engineering menggambarkan sebuah diagram UML terlebih dahulu sebelum membuat kode, sedangkan *reverse engineering* adalah kebalikannya, yaitu membuat diagram dari kode yang sudah ada untuk membantu memahaminya.

   Inti dari sketsa adalah pemilihan. Sketsa jenis *forward engineering*, membuat daftar beberapa masalah dalam kode yang akan tulis, biasanya dengan membicarakannya dahulu dengan sekelompok orang dalam tim. Tujuan adalah menggunakan sketsa untuk membantu menyampaikan ide atau alternatif tentang apa yang akan lakukan.

   Dengan *reverse engineering*, menggunakan sketsa untuk menjelaskan bagaimana beberapa bagian sebuah sistem bekerja.

2. *Blueprint*

   Penggunaan UML sebagai *blueprint* menyampaikan suatu keutuhan. Seperti halnya sketsa, blueprint juga bisa disajikan dalam sebuah *forward engineering* maupun dalam *reverse engineering*.

   Dalam *forward engineering*, *blueprint* dibuat oleh seorang desainer yang pekerjaannya membuat desain yang detail untuk dikodekan oleh

seorang programmer. Desain tersebut harus cukup lengkap, dalam artian seluruh keputusan desain telah dijabarkan dan programmer harus dapat mengikutinya secara langsung tanpa perlu berpikir keras. Desainer bisa jadi orang yang sama dengan programmer, tetapi biasanya desainer adalah *developer* yang lebih senior yang mendesain untuk sebuah tim programmer.

Dalam *reverse engineering*, *blueprint* bertujuan untuk menyampaikan informasi detail tentang kode baik dalam bentuk dokumen cetak atau sebagai *browser* grafis interaktif. *Blueprint* dapat menunjukan setiap detail sebuah *class* dalam sebuah bentuk grafis yang memudahkan *developer* dalam memahaminya.

Blueprint membutuhkan piranti yang lebih canggih dari pada yang dibutuhkan sketsa untuk menangani detailnya. Piranti *forward engineering* mendukung penggambaran diagram dan menyimpannya dalam suatu tempat untuk menyimpan informasi. Piranti *reverse engineering* membaca *source code* dan menginterpretasinya untuk dimasukan ke dalam ruang penyimpanan, lalu membuat diagram. Piranti yang dapat melakukan kedua *engineering* tersebut, baik *forward* maupun *reverse*, disebut piranti *roundtrip*.

Batas yang memisahkan *blueprint* dan sketsa sebenarnya kabur, tetapi perbedaannya terletak pada fakta bahwa sketsa dibuat tidak lengkap, hanya menunjukan informasi penting, sedangkan *blueprint* cenderung komprehensif dan bertujuan meringkas pemrograman menjadi

ktivitas yang simple dan sedikit mekanis. Dengan kata lain, sketsa bersifat *eksploratif* sedangkan *blueprint* bersifat *definitive*.

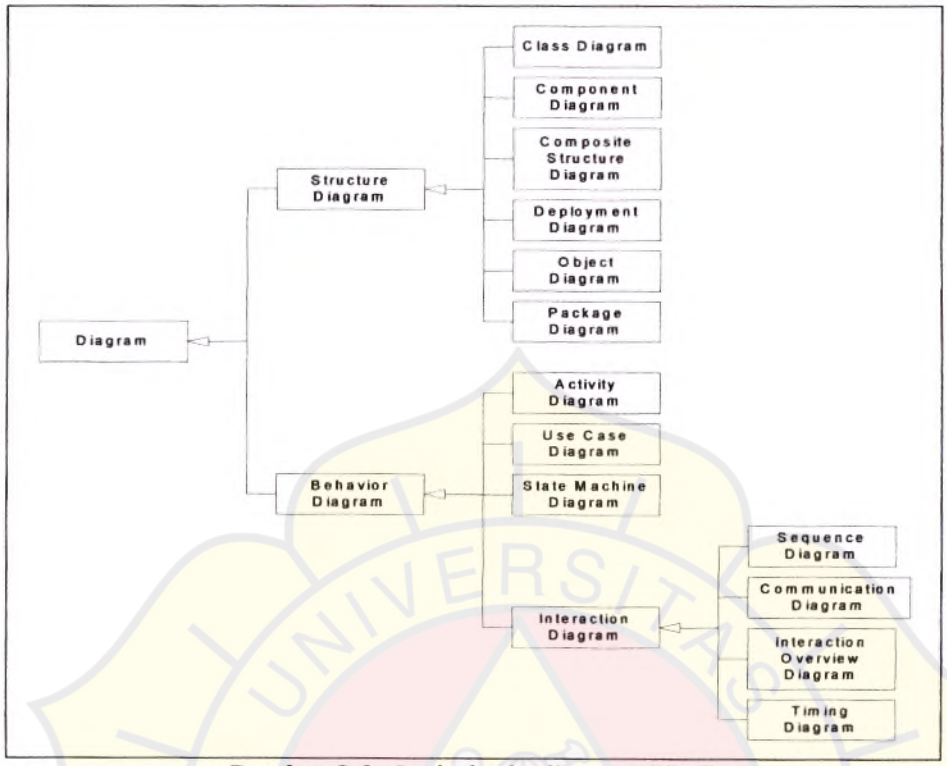
3. Bahasa pemrograman

UML sebagai bahasa pemrograman adalah tentang bagaimana memodelkan logika *behavioral*. UML 2 menawarkan tiga cara untuk pemodelan *behavior*: *interaction diagram*, *state diagram*, dan *activity diagram*.

UML 2 terdiri dari 13 jenis diagram resmi seperti tertulis dalam Table 2.1 dan mengklasifikasi UML dalam diagram-diagram seperti pada Gambar 2.2.

Table 2.1 Jenis – jenis UML

Diagram	Kegunaan	Turunan
Activity	Behavioral prosedural dan paralel	UML 1
Class	Class, fitur, dan hubungan-hubungan	UML 1
Communication	Interaksi antar objek; Penekanan pada jalur	Diagram kolaborasi UML 1
Component	Struktur dan koneksi komponen	UML 1
Composite Structure	Dekomposisi runtime sebuah class	Baru UML 2
Deployment	Pemindahan artifak ke Node	UML 1
Interaction overview	Campuran sequence dan activity diagram	Baru UML 2
Object	Contoh konfigurasi dari contoh-contoh	Tidak resmi UML 1
Package	Struktur hirarki Compile-time	Tidak resmi UML 1
Sequence	Interaksi antar objek; Penekanan pada sequen	UML 1
State machine	Bagaimana event me-ngubah objek selama aktif	UML 1
Timing	Interaksi antar objek; Penekanan pda timing	Baru UML 2
Use Case	Bagaimana pengguna berinteraksi dengan sebuah sistem	UML 1



Gambar 2.2. Jenis-jenis diagram UML

Pada peracangan aplikasi ini penulis mengunakan beberapa *diagram* UML yang relevan, di antaranya : *use case diagram, activity diagram, component diagram* dan *deployment diagram*.

2.3.1 Use Case

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem, yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. (Dharwiyanti,Sri & W Satria, Romi. Ilmukomputer.com, 2003).

Use case mendeskripsikan interaksi antara para pengguna sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan. Dua hal (atau istilah) yang tidak dapat dipisahkan dari *use case*, yakni *scenario* dan *actor*. *Scenario* adalah rangkaian langkah-langkah yang menjabarkan sebuah interaksi antara seorang pengguna dengan sebuah sistem .

Dimulai dengan mengambil salah satu *scenario* sebagai sekenario keberhasilan utama (MSS – *main success scenario*). Isi *use case* dibuat dengan menuliskan sekenario keberhasilan utama sebagai serangkaian langkah-langkah bernomor, kemudian diambil *scenario* lain dan ditulis sebagai ekstensi, hasilnya adalah variasi sekenario keberhasilan utama. Ekstensi-ekstensi tersebut dapat berupa keberhasilan pengguna mencapai keberhasilan.

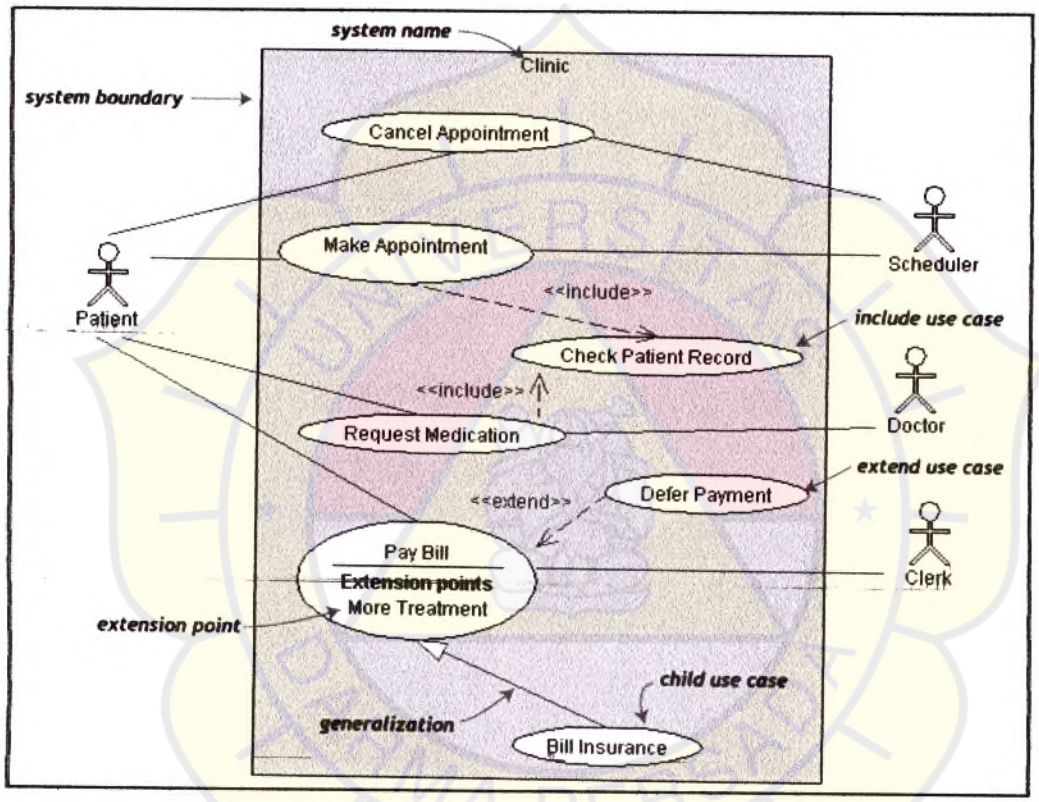
Sebuah akstensi dalam sebuah *use case* menyebutkan sebuah kondisi yang menghasilkan interaksi berbeda dari yang disebutkan dalam MSS dan menyebutkan apa perbedaannya. Sebuah ekstensi dimulai dengan menunjuk langkah di mana kondisi tersebut terdeteksi dan tersedia sebuah gambaran singkat tentang kondisi tersebut. Kondisi tersebut kemudian diurutkan dengan langkah-langkah bernomor menggunakan model yang sama dengan MSS. Langkah-langkah ini diakhiri dengan kembali ke MSS selanjutnya.

Use case diagram dapat sangat membantu bila sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang

di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.



Gambar 2.3 Contoh *use case diagram* UML

2.3.2 Activity Diagram

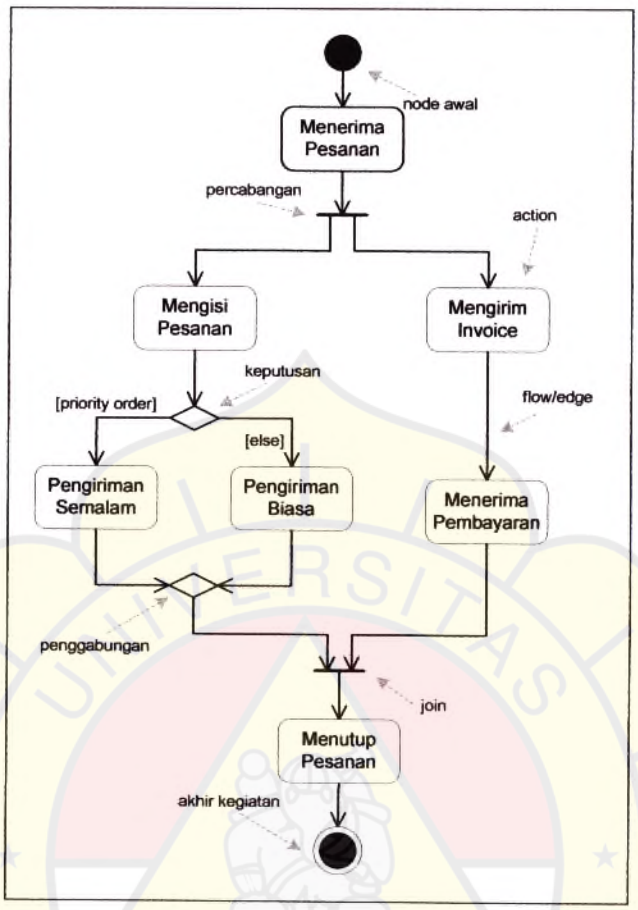
Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). *Activity diagram* lebih menggambarkan proses-proses dan jalur-jalur aktivitas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana *actor* menggunakan sistem untuk melakukan aktivitas.

Standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

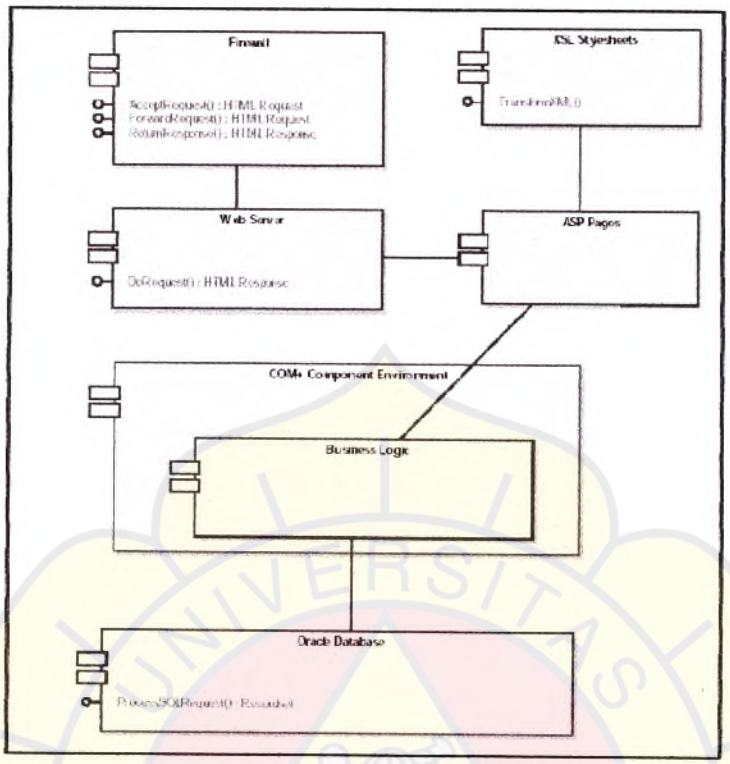
UML 1, activity diagram dianggap sebagai kasus khusus state diagram. UML 2, activity diagram telah dikembangkan sehingga menjadi diagram yang benar-benar berdiri sendiri, tidak ditemukan lagi keterikatan dengan state diagram.



Gambar 2.4 Contoh *Activity diagram* UML

2.3.3 Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

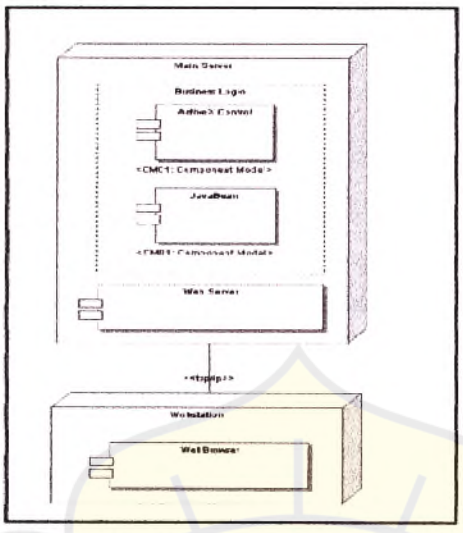


Gambar 2.5 Contoh *Component diagram* UML

2.3.4 Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.

Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.



Gambar 2.6 Contoh *Deployment diagram* UML

2.4 Perangkat Lunak Untuk Merancang Sebuah *Website*

2.4.1 Apache Web Server

Server HTTP Apache atau Server Web/WWW Apache adalah server web yang dapat dijalankan di banyak sistem operasi (Unix, BSD, Linux, Microsoft Windows dan Novell Netware serta platform lainnya) yang berguna untuk melayani dan memfungsikan situs web. Protokol yang digunakan untuk melayani fasilitas web/www ini mengunakan HTTP.

(http://id.wikipedia.org/wiki/Apache_HTTP_Server,10 Januari 2010).

Apache didukung oleh sejumlah antarmuka pengguna berbasis grafik (GUI) yang memungkinkan penanganan server menjadi mudah.

Apache merupakan perangkat lunak *open source* dikembangkan oleh komunitas terbuka yang terdiri dari pengembang-pengembang dibawah naungan Apache Software Foundation.

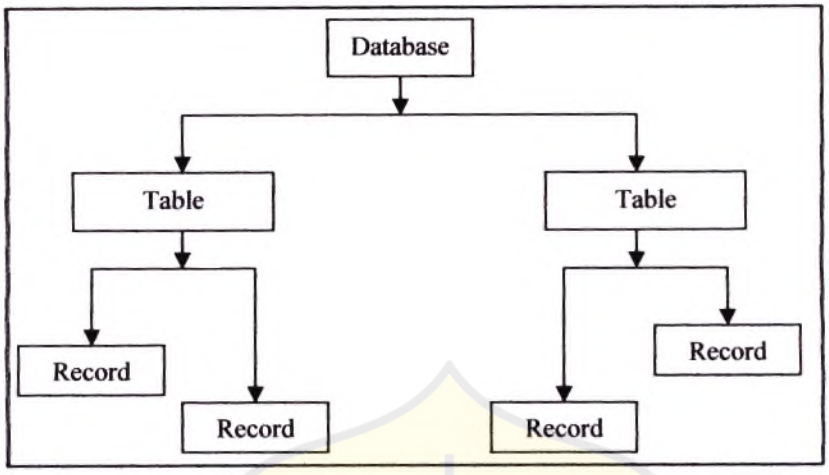
2.4.2 MySql

MySQL adalah merupakan perangkat lunak untuk sistem manajemen *database* (*Database Management System*). Sifatnya yang *open source* dan memiliki kemampuan menampung kapasitas yang sangat besar, menjadikan MySQL menjadi *database* yang sangat popular di kalangan programmer *web*. MySQL dapat dijalankan dalam 2 *operating system* yang sangat popular saat ini, yaitu : Windows dan Linux. Menurut perusahaan pengembangnya, MySQL telah terpasang di sekitar 3 juta komputer, dan puluhan hingga ratusan ribu situs sangat mengadalkan MySQL sebagai *database*nya. (Mohamad Sukarno, Membangun *Website* Dinamis Interaktif dengan PHP-MySQL, 2006).

MySQL dikembangkan oleh sebuah perusahaan Swedia yang pada saat itu bernama TcX DataKonsult AB, dan pada akhirnya berubah nama menjadi MySQL AB. Sekitar tahun 1994-1995, TcX membuat MySQL untuk mengembangkan aplikasi *web* bagi klien-nya. TcX merupakan perusahaan pengembangan *software* dan konsultan *database*.

Dalam pembentukan sistem *website* yang dinamis sudah tentu akan membutuhkan pengolahan data secara otomatis pada *server*, hal itu membutuhkan suatu program *database* engine yang digunakan untuk membantu manajemen data pada *server*. *Database engine* seperti MySQL yang berdasar pada konsep *database relasional*.

Manajemen dasar *database relasional* adalah pengarsipan data-data dalam *record-record* tabel. Sebuah *database* dapat berisi tabel-tabel, dan *record* adalah bagian terkecil dari sebuah tabel. Seperti terlihat pada gambar dibawah ini :



Gambar 2.7 Struktur basis data relasional

2.4.3 PhpMyAdmin

Pada awalnya para programmer *web* dalam membuat atau mengelola suatu *database* dilakukan pada *console* (Linux) atau *dos-prompt* (Windows). Semenjak adanya paket yang menyatukan antara Apache-MySQL-PHP, yaitu salah satunya PHPTriad, sehingga untuk melakukan pembuatan dan pengolahan *database* dapat dilakukan pada *browser*, yaitu dikenal dengan phpmyadmin. Xampp pun menyediakan folder phpmyadmin yang berfungsi dalam pembentukan dan pengolahan *database* pada *browser*. Adapun alamat URL (*Uniform Resourse Locator*) phpmyadmin pada xampp di *operating system* Linux ataupun Windows, adalah http://localhost/phpmyadmin (Mohamad Sukarno, Membangun *Website* Dinamis Interaktif dengan PHP-MySQL, 2006).

phpMyAdmin adalah suatu *tool* perangkat lunak yang tidak berbayar untuk PHP yang diharapkan untuk menangani administrasi MySQL didalam *Website*. phpMyAdmin merupakan *software* yang sering digunakan sebagai antarmuka dalam mengatur *database*, tabel-tabel, *field-field*, relasi, dan lain-lain), bisa juga melaksankan secara langsung melaksanakan setiap perintah SQL.

phpMyAdmin diterjemahkan ke dalam 57 bahasa dan dukungan bahasa LTR (*Left To Right*) dan RTL (*Right To Left*). phpMyAdmin sudah memperoleh beberapa penghargaan, di antaranya terpilih sebagai aplikasi PHP terbaik dan setiap tahun memenangkan SourceForgenet Community Choice Awards sebagai "Alat Terbaik untuk SysAdmins". (http://www.phpmyadmin.net).

2.4.4 phpDesigner 2008

phpDesigner 2008 adalah IDE (*Integrated Development Environment*) PHP yang membantu meningkatkan proses pengeditan, menganalisis, debugging dan penerbitan aplikasi dengan bahasa PHP ataupun bahasa *web* lainnya. phpDesigner 2008 juga mendukung pekerjaan dengan OOP (*Object Oriented Programming*) yang ada didalam PHP5. phpDesigner 2008 memiliki fitur kode *nested* penyelesaian dan kode tip yang ditampilkan saat mengetik sehingga tidak perlu melihat dokumentasi untuk melihat *class* dan fungsi atribut. phpDesigner 2008 dapat bekerja dengan *projects* dan mengintegrasikan setiap PHP *framework* ke dalam *projects*. phpDesigner 2008 dapat mengakses semua file, kelas, fungsi, interface, variabel dan konstanta dideklarasikan didalam *projects* dengan cepat. Debugging yang lebih baru dan mudah dengan integrasi baru Xdebug. phpDesigner 2008 dapat menganalisa kode secara langkah demi langkah menggunakan breakpoints, evaluasi dan lain-lain.

Selain itu tersedia Pallete PHP *code explorer* yang berisi kan data-data dari *script* yang kita tulis seperti variabel-variabel dan di baris mana saja variabel itu di gunakan, *class* yang buat, fungsi dan sebagainya. Tersedia juga fitur PHP

beautifier yang berfungsi untuk merapikan *code* yang kita tulis dengan secara otomatis mengatur *indent* dan baris untuk fungsi-fungsi tertentu, dan lain-lain.

phpDesigner 2008 juga memunculkan fungsi-fungsi, *method*, *class built-in* (bawaan) PHP sehingga memudahkan pencarian fungsi-fungsi yang tidak ketahui sebelumnya

2.4.5 Rational Rose Enterprise Edition 2002

Rational Rose adalah *tools* pemodelan visual untuk pengembangan sistem berbasis obyek yang baik untuk digunakan sebagai bantuan bagi para pengembang dalam melakukan analisis dan perancangan sistem. *Rational rose* mendukung pemodelan bisnis yang membantu para pengembang memahami sistem secara komprehensif. *Rational rose* juga membantu analisis sistem dengan cara pengembang membuat *use case diagram* untuk melihat fungsionalitas sistem secara keseluruhan sesuai dengan harapan dan keinginan pengguna. *Rational rose* juga menuntut pengembang untuk mengembangkan *Interaction Diagram* untuk melihat bagaimana obyek-obyek saling bekerjasama dalam menyediakan fungsionalitas yang diperlukan.

Dalam *Rational rose*, pemodelan adalah cara melihat sistem dari berbagai sudut pandang. *Rational Rose* mencakup semua diagram yang dikenal dalam UML, *actor-actor* yang terlibat dalam sistem , *use-case*, obyek-obyek, *class-class*, komponen-komponen. Model juga mendeskripsikan rincian yang diperlukan sistem dan bagaimana model akan bekerja, sehingga para pengembang dapat

menggunakan model itu sebagai *blue print* untuk sistem yang akan dikembangkan.

Pemodelan yang dapat dibuat dengan rational rose adalah : *Use Case Diagram, Class Diagram, Sequence Diagram, Collaboration Diagram, Membuat State Diagram.*

2.5 *SCRIPT*

2.5.1 HTML

HyperText Markup Language (HTML) adalah sebuah bahasa *markup* yang digunakan untuk membuat sebuah halaman *web* dan menampilkan berbagai informasi di dalam sebuah *browser internet*. Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML *(Standard Generalized Markup Language)*, HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman *web*. HTML saat ini merupakan standar *internet* yang didefinisikan dan dikendalikan penggunaannya oleh *World Wide Web Consortium* (W3C).

HTML berupa kode-kode *tag* yang menginstruksikan *browser* untuk menghasilkan tampilan sesuai dengan yang diinginkan. Sebuah file yang merupakan file HTML dapat dibuka dengan menggunakan *browser web* seperti Mozilla Firefox atau Microsoft Internet Explorer. HTML juga dapat dikenali oleh aplikasi pembuka *email* ataupun dari PDA dan program lain yang memiliki kemampuan *browser*. (http://id.wikipedia.org/wiki/Hypertext_markup_language, 28 Januari 2010)

2.5.2 PHP

PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Jika dilihat dari versi pertamanya bahwa PHP terdiri dari sekumpulan *script* PERL yang digunakan untuk mengelola data *form* dari *web*. Kemungkinan bahwa PHP singkatan dari *Perl Hypertext Preprcessor*. Pada awalnya PHP bernama FI (*Form Interpreted*). Setelah Rasmus melepaskan kode sumbernya, maka terbentuknya nama PHP/FI (*Personal Home Page/Form Interprenter*). Sejak itulah PHP bersifat *open source*. (Mohamad Sukarno, Membangun *Website* Dinamis Interaktif dengan PHP-MySQL, 2006).

Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini *interpreter* PHP sudah diimplementasikan dalam program C. Juni 1998, perusahaan bernama Zend merilis *interpreter* baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0 dan singkatan PHP dirubah menjadi akronim berulang *PHP: Hypertext Preprocessing*. Pada pertengahan tahun 1999, Zend merilis *interpreter* PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi. Juni 2004, Zend merilis PHP 5.0, dalam versi ini, inti dari *interpreter* PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi obyek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi obyek. (http://id.wikipedia.org/wiki/php, 01 Maret 2010).