BAB II
LANDASAN TEORI

Pada bab kedua ini, memberikan pembahasan tentang teori-teori yang berhubungan dengan Aplikasi Peramalan Nilai Tukar Rupiah Berbasis *web* Menggunakan Metode *Fuzzy Time Series*. Hal-hal yang dijelaskan yaitu :

2.1. Tinjauan Pustaka

Tinjauan pustaka yang berhubungan dengan aplikasi peramalan nilai tukar mata uang asing berbasis *web* menggunakan metode *fuzzy time series*, dan membahas landasan teori yang berhubungan dengan aplikasi peramalan nilai tukar mata uang asing.

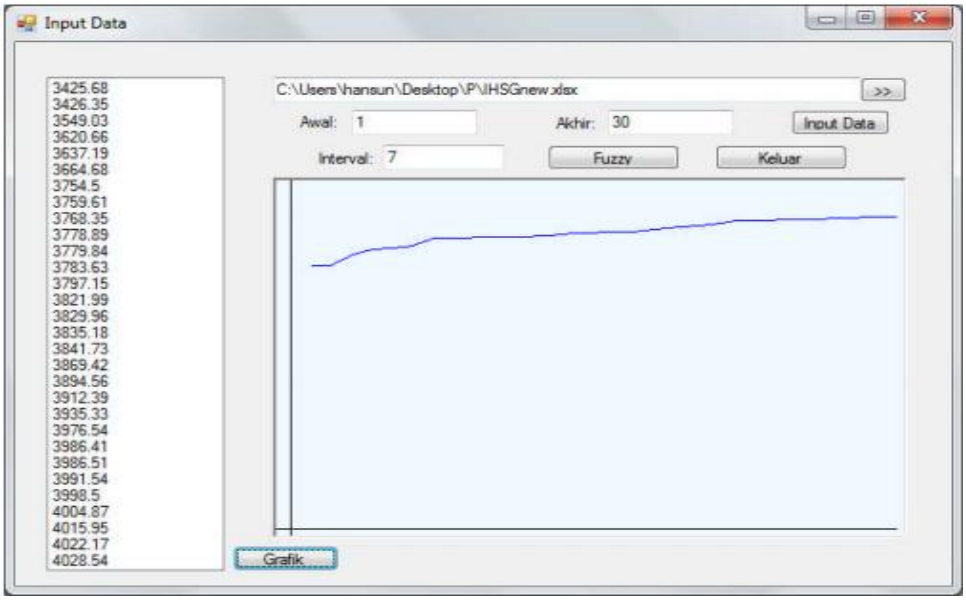
2.1.1. Kajian Pustaka

Hansun (2012), menggunakan *tools* Visual Basic dalam implementasi peramalan data IHSG menggunakan metode *Fuzzy Time Series*. Metode peramalan *fuzzy time series* memberikan hasil peramalan yang cukup baik untuk peramalan data IHSG. Hal ini dapat dilihat dari nilai *Mean Square Error* (MSE) dan *Mean Absolute Percentage Error* (MAPE) yang cukup kecil, yakni 5.404564 untuk MSE dan 0.04777038 untuk MAPE (Gambar 2.1).

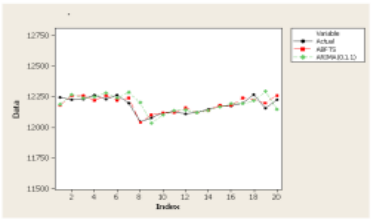
Rachmawansah (2014) menyimpulkan bahwa untuk meramalkan nilai tukar USD-IDR dan EUR-USD metode fuzzy *time series* layak digunakan untuk melakukan perkiraan (Gambar 2.2).

Helmy (2011), menggunakan metode Box-Jenkins (ARIMA) untuk melakukan peramalan kurs Rupiah terhadap Dollar Amerika. Model Box-Jenkins dapat digunakan sebagai alat untuk memprediksi pergerakan nilai tukar rupiah, namun berdasarkan nilai koefisien determinasi yang relative tidak terlalu besar (hanya sekitar 0,7) menunjukan bahwa dalam periode pengamatan kemampuan prediksi model ini tidak terlalu akurat (Gambar 2.3).

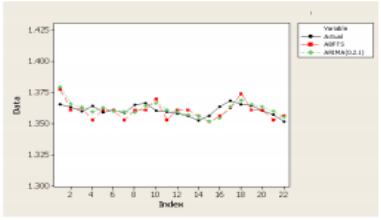
Dari ketiga judul penelitian tersebut belum ada penelitian yang menggunakan *web* sebagai *platform* aplikasinya.



Gambar 2.1. Antarmuka Aplikasi Peramalan IHSG



Gambar 1 Plot Data Aktual dan Hasil Ramalan ABFTS Pada Data USD-IDR

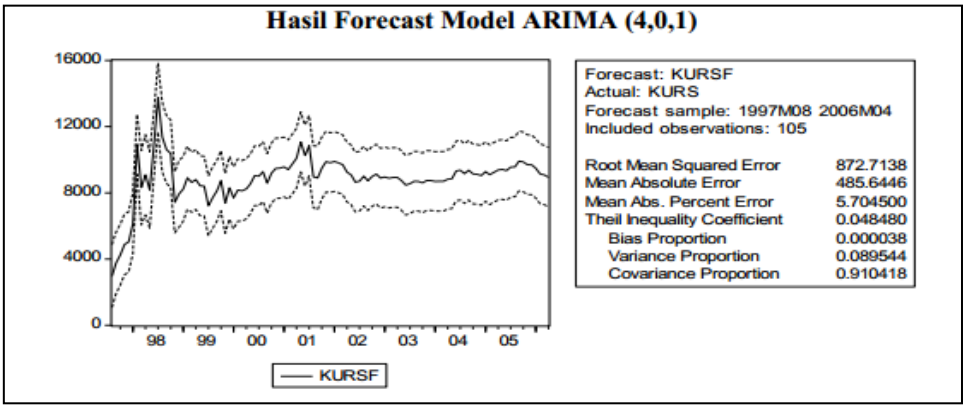


Gambar 2 Plot Data Aktual dan Hasil Ramalan ABFTS Pada Data EUR-USD

Tabel 5 Hasil perhitungan MSE dan MAPE

Data	Metode	MSE	MAPE (%)
USD-IDR	ABFTS	**1142.35**	**0.231**
	ARIMA (0.1.1)	3576.679	0.347
EUR-USD	ABFTS	0.0000355	0.376
	ARIMA (0.2.1)	**0.0000321**	**0.279**

Gambar 2.2. Hasil Perhitungan MSE fuzzy *time series*



Gambar 2.3. Hasil Peramalan Metode Box-Jenkins

2.1.2. Kerangka Pemikiran

Kerangka pemikiran yang digunakan dalam penelitian ini adalah :

1. Latar belakang masalah

   Masalah yang terjadi adalah saat ini tidak ada alat / situs yang menyediakan prediksi nilai tukar mata uang, hanya menyediakan informasi nilai tukar mata uang secara *real time*.

2. Rumusan masalah

   Bagaimana membuat aplikasi peramalan nilai tukar mata uang asing berbasis *web*

3. Penguasaan dasar PHP, *Framework* Laravel dan RDBMS MySQL

   Tahap untuk mempelajari dasar-dasar PHP, *Framework* Laravel dan MySQL agar lebih menguasai program – program yang akan digunakan untuk membangun sistem.

4. Pengumpulan data tertulis dan tidak tertulis

   Pengumpulan data dilakukan baik dengan tanya – jawab (*interview*), observasi, maupun studi literatur di perpustakaan.

5. Observasi aplikasi PHP, *Framework* Laravel dan MySQL

   Merupakan tahap pengamatan sampel – sampel aplikasi yang telah ada, jurnal, buku, maupun karya ilmiah untuk kajian yang dapat dijadikan referensi untuk pembangunan aplikasi.

6. Analisis dan perancangan aplikasi terstruktur

   Merupakan tahap pengolahan hasil observasi ke dalam diagram perancangan menggunakan UML Rational Rose.

7. Implementasi Aplikasi Peramalan Nilai Tukar Mata Uang Asing

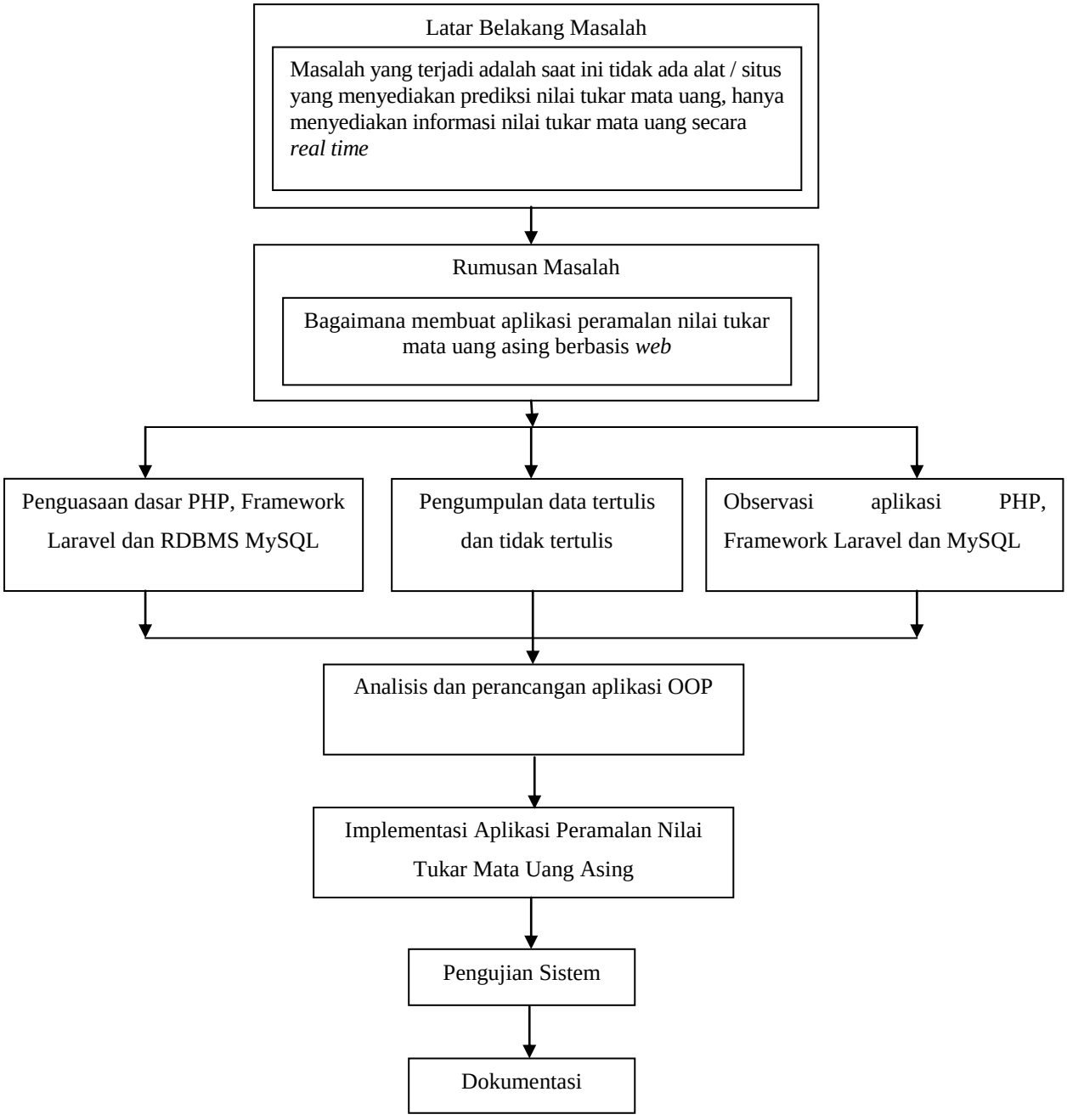
   Implementasi menggunakan *framework* laravel yang menganut pemrograman berorientasi obyek dengan menggunakan *editor* notepad++.

8. Pengujian Sistem

   Pengujian sistem akan dilakukan pada beberapa komputer untuk mengetahui jika ada kesalahan dan kekurangan pada sistem. Pengujian menggunakan metode Webqual.

9. Dokumentasi

Tahapan terakhir, yakni tahap pendokumentasian seluruh proses penyusunan tugas akhir ke dalam laporan. Diagram seperti pada Gambar 2.4.



Gambar 2.4. Diagram Kerangka Pemikiran

2.2. Konsep Dasar Peramalan

Menurut Heizer dan Render (2011), peramalan (*forecasting*) adalah seni dan ilmu untuk memperkirakan kejadian dimasa depan. Hal ini dapat dilakukan dengan melibatkan pengambilan data masa lalu dan menempatkannya kemasa akan datang dengan suatu bentuk model matematis. Pada umumnya peramalan memiliki jangka waktu tertentu yang dapat dikelompokkan menjadi beberapa kategori yaitu :

a. Peramalan jangka pendek, peramalan untuk jangka waktu kurang dari tiga bulan.
b. Peramalan jangka menengah, peramalan untuk jangka waktu antara tiga bulan sampai tiga tahun.
c. Peramalan jangka panjang, peramalan untuk jangka waktu lebih dari tiga tahun.

Sedangkan jika dilihat dari sifat peramalan maka dapat dikelompokkan menjadi 2 jenis peramalan yaitu :

a. Peramalan Subjektif, yaitu peramalan yang didasarkan atas perasaan atau intuisi dari orang yang menyusunnya. Dalam hal ini pandangan atau ketajaman pikiran orang yang menyusunnya sangat menentukan baik tidaknya hasil peramalan.
b. Peramalan objektif, yaitu peramalan yang didasarkan atas data yang relevan pada masa lalu dengan menggunakan teknik-teknik dan metode-metode dalam penganalisisan data tersebut.

2.3. Nilai Tukar Rupiah

Menurut Simorangkir dan Suseno (2004), nilai tukar mata uang atau sering disebut dengan kurs adalah harga satu unit mata uang asing dalam mata uang domestik atau dapat juga dikatakan hargamata uang domestik terhadap mata uang asing.

2.4. Logika Fuzzy

Fuzzy secara bahasa diartikan sebagai kabur atau samar-samar. Suatu nilai dapat bernilai besar atau salah secara bersamaan. Dalam fuzzy dikenal derajat keanggotaan yang memiliki rentang nilai 0 (nol) hingga 1 (satu). Berbeda dengan himpunan tegas yang memiliki nilai 1 atau 0 (ya atau tidak).

Logika fuzzy pertama kali diperkenalkan oleh Lotfi. A. Zadeh pada tahun 1965. Dasar logika fuzzya dalah teori himpunan fuzzy. Dalam teori himpunan dikenal fungsi karakteristik yaitu fungsi dari himpunan semesta X ke himpunan {0,1}

Definisi: Himpunan A dalam semesta X dapat dinyatakan dengan fungsi karakteristik yang didefinisikan dengan persamaan 2.1:

$$x_A(X) = \begin{cases} 1 \; jika \; x \in A \\ 0 \; jika \; x \notin A \end{cases} \forall \; x \; \in X \tag{2.1}$$

Logika fuzzy merupakan suatu logika yang memiliki nilai kekaburan atau kesamaran (fuzzyness) antara benar atau salah. Dalam teori logika fuzzy suatu nilai bisa bernilai benar atau salah secara bersama. Namun berapa besar keberadaan dan kesalahan sesuatu tergantung pada bobot keanggotaan yang dimilikinya. Logika fuzzy memiliki derajat keanggotaan dalam rentang 0 hingga 1. Berbeda dengan logika digital yang hanya memiliki dua nilai 0 atau 1.

Logika fuzzy digunakan untuk menterjemahkan suatu besaran yang diekspresikan menggunakan bahasa *(linguistic)*, misalkan kecepatan laju kendaraan yang diekspresikan dengan pelan, lebih cepat, cepat, dan sangat cepat. Logika fuzzy menunjukkan sejauh mana suatu nilai itu benar dan sejauh mana suatu nilai itu salah. Tidak seperti logika klasik (*crips*) atau tegas, suatu nilai hanya mempunyai 2 kemungkinan yaitu merupakan suatu anggota himpunan atau tidak. Derajat keanggotaan 0 (nol) artinya nilai bukan merupakan anggota himpunan dan 1 (satu) berarti nilai tersebut adalah anggota himpunan.

2.5. *Time Series*

Time series merupakan serangkaian data pengamatan yang berasal dari satu sumber tetap dan terjadi berdasarkan indeks waktu t secara beruntun dengan interval waktu yang tetap. Setiap pengamatan dapat dinyatakan sebagai variabel random Z t dengan notasi Z t1, Zt2, ..., Ztn (Wei, 2006).

Data berkala atau *time series* adalah data yang biasanya digunakan untuk mengGambarkan suatu perkembangan atau kecenderungan keadaan / peristiwa / kegiatan. Biasanya jarak atau interval dari waktu ke waktu sama.

Contoh data berkala adalah informasi harga saham pada rentang waktu tertentu, informasi nilai tukar valas dalam rentang waktu tertentu, informasi penjualan sebuah produk dalam rentang waktu tertentu.

2.6. *Framework* Laravel

Laravel adalah *framework* pengembangan *web* menggunakan modul *Models, Views, Controllers* atau yang disingkat MVC ditulis menggunakan bahasa pemrograman PHP. *Models* merupakan modul yang digunakan untuk mengelola data dalam sebuah *website* yang dikembangkan menggunakan *framework* laravel, yang berarti memudahkan pengembang *web* mengelola hubungan antara *web* dan database. *Views* merupakan modul bawaan laravel yang berfungsi mengelola tampilan yang akan disajikan pada *web* yang dikembangkan menggunakan *framework* laravel. *Controllers* adalah modul yang digunakan untuk mengelola *bussines logic* dari *web* yang dikembangkan menggunakan laravel. MVC dalam *framework* laravel mengadopsi konsep pemgrograman 3 *tier* yaitu *data tier, layout tier* dan *bussines logic tier* yang diwakili masing-masing modul dalam laravel.

Laravel telah dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya, baik biaya pengembangan awal dan biaya pemeliharaan, serta memberikan sintaks ekspresif yang jelas dan set fungsi inti yang akan menghemat waktu pengerjaan.

Laravel adalah salah satu dari beberapa kerangka bahasa pemrograman PHP yang menawarkan *code modular*. Hal ini dicapai melalui kombinasi *driver*

dan sistem *bundle*-nya. *Driver* memungkinkan kita untuk dengan mudah mengubah dan memperluas *caching, session, database,* dan fungsi otentikasi. Penggunaan *bundle* mampu mengemas hingga segala jenis kode untuk digunakan kembali atau untuk memberikan kepada seluruh pengguna Laravel. Laravel sangat menarik, karena apapun yang ditulis dalam Laravel dapat dikemas dalam sebuah kemasan (McCool, 2012).

2.7. Analisis Berorientasi Obyek

Analisis Berorientasi Obyek adalah metode analisis yang memeriksa *requirements* (syarat atau keperluan yang harus dipenuhi suatu sistem) (Suhendar dan Hariman, 2002:11)

Dalam tahap ini kegiatan-kegiatan yang dilakukan dalam menganalisa sistem sebagai berikut :

a. Menganalisa sistem yang ada dan mempelajari apa yang dikerjakan oleh sistem yang ada.
b. Menspesifikasikan sistem yaitu spesifikasi masukan yang digunakan database yang ada, proses yang dilakukan dan keluaran yang dihasilkan.

Tujuan dari analisa berorientasi obyek yaitu untuk menentukan kebutuhan pemakai secara akurat.

Pendekatan-pendekatan yang dipakai dalam analisa berorientasi obyek antara lain :

a. Pendekatan *top down*, yaitu memecahkan masalah ke dalam bagian-bagian terkecil atau per level sehingga mudah untuk diselesaikan.
b. Pendekatan modul, yaitu membagi sistem ke dalam modul-modul yang dapat beroperasi tanpa ketergantungan.
c. Penggunaan alat-alat bantu dalam bentuk grafik dan teks sehingga mudah untuk dimengerti serta dikoreksi apabila terjadi perubahan.

Pendekatan dalam analisa berorientasi obyek dilengkapi dengan alat-alat dan teknik-teknik yang dibutuhkan dalam pengembangan sistem, sehingga hasil

akhir dari sistem yang dikembangkan akan didapatkan sistem yang terdefinisi dengan baik dan jelas.

2.7.1 *Use case* Diagram

Use case diagram menggambarkan kebutuhan sistem dari sudut pandang user. Digunakan untuk menggambarkan hubungan antara internal sistem dan eksternal sistem atau hubungan antara *use case* dan aktor.

2.7.1.1 *Actor*

Actor adalah sesuatu (entitas) yang berhubungan dengan sistem dan berpartisipasi dalam *use case*. *Actor* menggambarkan orang, sistem atau entitas eksternal yang secara khusus membangkitkan sistem dengan *input* atau masukan kejadian-kejadian, atau menerima sesuatu dari sistem. *Actor* dilukiskan dengan peran yang mereka mainkan dalam *use case*, seperti *Visitor*, Admin dan lain-lain.

Dalam *use case* diagram terdapat satu aktor pemulai atau *initiator actor* yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor lain yang berpartisipasi atau *participating actor*. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut. *Actor* dalam UML dinotasikan seperti pada Tabel 2.1.

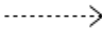
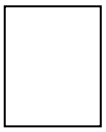
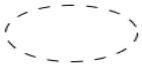
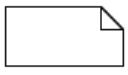
2.7.1.2 *Use case*

Use case yang dibuat berdasar keperluan aktor merupakan Gambaran dari "apa" yang dikerjakan oleh sistem, bukan "bagaimana" sistem mengerjakannya. *Use case* diberi nama yang menyatakan apa hal yang dicapai dari interaksinya dengan aktor. Dalam UML *use case* dinotasikan seperti pada Tabel 2.1.

2.7.1.3 *Relationship*

Relasi (*relationship*) diGambarkan sebagai bentuk garis antara dua simbol dalam *use case diagram*. Relasi antara *actor* dan *use case* disebut juga dengan *asosiasi (association)*. Asosiasi ini digunakan untuk mengGambarkan bagaimana hubungan antara keduanya.

Tabel 2.1 Notasi *Use Case Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4	<<include>> 	*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5	<<extend>> 	*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

Relasi-relasi yang terjadi pada *use case diagram* bisa antara *actor* dengan *use case* atau *use case* dengan *use case*.

Relasi antara *use case* dengan *use case* :

a. *Include*, pemanggilan *use case* oleh *use case* lain atau untuk mengGambarkan suatu *use case* termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. DiGambarkan dengan garis lurus berpanah dengan tulisan <<include>>.
b. *Extend*, digunakan ketika hendak mengGambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak kontrol form dan mendeklarasikan ekstension pada *use case* utama. Atau dengan kata lain adalah perluasan dari *use case* lain jika syarat atau kondisi terpenuhi. DiGambarkan dengan garis berpanah dengan tulisan <<extend>>.
c. *Generalization/Inheritance*, dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan *base-parent use case*. DiGambarkan dengan garis berpanah tertutup dari *base use case* ke *parent use case*.

2.7.2 *Activity Diagram*

Diagram aktivitas mengGambarkan proses bisnis dan urutan aktivitas-aktivitas yang mendukung pengGambaran tindakan sistem baik yang bersifat kondisional maupun paralel. Tindakan kondisional dilukiskan dengan cabang (*branch*) dan penyatuan (*merge*).

Sebuah *branch* memiliki sebuah *transition* masuk atau yang disebut dengan *incoming transition* dan beberapa *transition* keluar atau yang disebut dengan *outgoing transition* dari *branch* yang berupa keputusan-keputusan. Hanya satu dari *outgoing transition* yang dapat diambil, maka keputusan-keputusan tersebut harus bersifat *mutually exclusive*. [else] digunakan sebagai keterangan singkat yang menunjukkan bahwa transition “else” tersebut harus digunakan jika semua keputusan yang ada pada *branch* salah.

Sebuah merge memiliki banyak *input transition* dan sebuah *output*. *Merge* menandakan akhir dari suatu kondisi yang diawali dengan sebuah *branch*. Selain *branch* dan *merge*, di dalam diagram aktivitas terdapat pula *fork* dan *join*. *Fork*

memiliki satu *incoming transition* dan beberapa *outgoing transition*. Sedangkan pada *join*, *outgoing transition* diambil atau digunakan hanya ketika semua *state* pada *incoming transition* telah menyelesaikan aktivitasnya. Notasi pada *activity diagram* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Notasi *Activity Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

2.7.3 *Sequence Diagram*

Diagram yang mengGambarkan bagaimana obyek berinteraksi dengan obyek lainnya melalui pesan (*message*) yang disampaikan, disusun dalam urutan kejadian atau waktu dan secara khusus berasosiasi dengan *use case.* Notasi dalam *sequence* diagram dapat dilihat pada Tabel 2.3.

2.7.4 *Class Diagram*

Class diagram merupakan bagian yang paling penting dalam analisa dan perancangan berorientasi obyek. Dalam UML diagram kelas digunakan untuk memodelkan *static structure* dari aplikasi. Notasi *class diagram* dapat dilihat pada Tabel 2.4

Kelas merupakan himpunan dari obyek yang sejenis yang mempunyai atribut *(attribute*) dan perilaku (*behaviors/method*) yang sama. Atribut adalah sebuah nilai data karakteristik yang dimiliki oleh obyek sebuah kelas sedangkan *method*

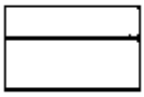
adalah perilaku atau operasi yang dikenakan oleh suatu kelas. Pada Gambar kelas terdapat tiga bagiannya.

Tabel 2.3 Notasi *Sequence Diagram*

Simbol	Nama	Keterangan
 Nama Aktor	*Actor*	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi lain diluar sistem informasi itu sendiri; biasanya dinyatakan menggunakan kata benda di awal frase nama actor
	Lifeline	Menyatakan kehidupan suatu objek
Nama objek	*Objek*	Menyatakan objek yang berinteraksi
	*Waktu aktif*	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan
Message()	*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi tentang aktifitas yang terjadi
	*Boundary*	Digunakan untuk mengGambarkan form
	*Control Class*	Digunakan untuk menghubungkan boundary dengan Tabel pada database
	*Entity Class*	Digunakan untuk mengGambarkan hubungan kegiatan yang akan dilakukan
X	*Pesan tipe destroy*	Menyatakan akhir hidup suatu objek

Diagram kelas mengGambarkan struktur obyek sistem, dimana diperlihatkan hubungan antar mereka. Diagram kelas merupakan fondasi untuk *component diagram* dan *deployment diagram.*

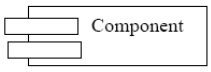
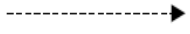
Tabel 2.4 Notasi *Class Diagram*

Simbol	Nama	Keterangan
	*Generalization*	Hubungan di mana objek anak berbagi perilaku dan struktur data dari objek yang di atasnya yaitu objek induk (ancestor)
	*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama
	*Collaboration*	Interaksi aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemennya
	*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek
	*Dependency*	Hubungan di mana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang bergantung padanya

2.7.5 *Component Diagram*

Diagram ini mengGambarkan distribusi fisik dari modul perangkat lunak melalui jaringan. Misalnya, ketika merancang sistem *client-server*, hal ini berguna untuk menunjukkan mana kelas atau paket kelas akan berada pada node klien dan mana yang akan berada di server.

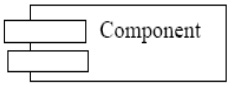
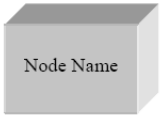
Tabel 2.5 Notasi *Component Diagram*

	SIMBOL	**NAMA**	**KETERANGAN**
1		*Component*	melambangkan sebuah entitas *software* dalam sebuah sistem.
2		*Dependency*	Sebuah *Dependency* digunakan untuk menotasikan relasi antara dua komponen.

2.7.6 *Deployment Diagram*

Deployment diagram merupakan Gambaran proses-proses berbeda pada suatu sistem yang berjalan dan bagaimana relasi di dalamnya. Hal inilah yang mempermudah *user* dalam pemakaian sistem yang telah dibuat dan diagram tersebut merupakan diagram yang statis. Misalnya untuk mendeskripsikan sebuah situs, *deployment diagram* menunjukkan komponen perangkat keras ("*node*") apa yang digunakan (misalnya, *web server*, *server* aplikasi, dan database *server*), komponen perangkat lunak ("artefak") apa yang berjalan pada setiap node (misalnya, aplikasi *web*, basis data), dan bagaimana bagian-bagian yang berbeda terhubung (misalnya JDBC, REST, RMI). Notasi pada *deployment diagram* dapat dilihat pada Tabel 2.6.

Tabel 2.6 Notasi *Deployment Diagram*

SIMBOL	NAMA	KETERANGAN
	*Component*	*Component* yang ada diletakkan didalam node untuk memastikan keberadaan posisi mereka
	*Node*	mengGambarkan bagian-bagian hardware dalam sebuah sistem.
	Association	menghubungkan dua node yang mengindikasikan jalur komunikasi antara element-elemen hardware.

2.8 PHP

Menurut Virgi & Hirin (2011) PHP atau kependekan dari *Hypertext Preprocessor* adalah salah satu bahasa pemrograman *open source* yang sangat cocok atau dikusukan untuk pengembangan *web* dan dapat ditanamkan pada skrip HTML.

Mesin PHP mencari baris-baris yang berada di dalam tag <?php atau <? dan ?> di dalam halaman HTML, dan menerjemahkannya sehingga *web server* dapat memberikan hasil berupa HTML. PHP pertamakali ditulis oleh Pak Rasmus Lexdorf, seorang pemuda yang baru lulus kuliah di Finlandia. Program ini merupakan kumpulan program PERL yang disederhanakan bernama PHP F1.

Karena peminat yang banyak akhirnya dirilislah PHP menjadi PHP F2, dan dibantu dua orang mahasiswa yang membuatkan *engine* untuk PHP yang akhirnya lahir PHP3. dan fersi terbarunya adalah PHP 4.

Model kerja HTML diawali dengan permintaan suatu halaman *web* oleh browser. Berdasarkan URL (*Uniform Resource Locator*) atau dikenal dengan sebutan alamat internet, browser mendapatkan alamat dari *web server*, mengidentifikasi halaman yang dikehendaki, dan menyapaikan segala informasi yang dibutuhkan oleh *web server*. Selanjutnya ketika berkas php yang diminta didapatkan oleh *web server*, isinya segera dikirimkan ke mesin php dan mesin inilah yang memproses dan memberikan hasilnya (berupa kode HTML) ke *web server*, dan *web server* akan menyampaikan ke *client*.

2.9 MySQL

Menurut Virgi & Hirin (2011) MySQL adalah salah satu perangkat lunak sistem manajemen basis data atau *database* SQL atau sering disebut dengan *Database Management Sistem* (DBMS). Berbeda dengan basis data konvensional seperti. *dat, .dbf, .mdb*, MySQL memiliki kelebihan yaitu bersifat *multithread*, dan multi *user* serta mendukung sistem jaringan. MySQL didistribuasikan secara gratis di bawah lisensi GNU *General Public License* (GPL), namun ada juga versi komersial bagi kalangan tertentu yang menginginkannya.

Sebagai *database server* yang memiliki konsep *database* modern, MySQL memiliki banyak sekali keistemewaan antara lain :

1. *Portabilitas*, dapat berjalan stabil pada berbagai sistem operasi, seperti *Windows, Linux, MacOS*, dan lain-lain.
2. *Open Source*, didstribusikan secara gratis dibawah lisensi GPL *(General Public License).*
3. *Multiuser*, dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah.
4. *Performance Tuning*, memiliki kecepatan yang menakjubkan dalam menangani *query* yang sederhana, dapat memproses lebih banyak SQL per satuan waktu.

5. *Security*, memiliki beberapa lapisan sekuritas seperti level *subnet mask*, nama *host*, izin akes *user* dengan sistem perizinan yang mendetail serta *password* yang terenskripsi.
6. *Scalability and Limits*, mampu menangani *database* dalam skala besar, dengan jumlah *record* lebih dari 50 (lima puluh) juta dan 60 (enam puluh) ribu Tabel serta 5 (lima) miliar baris. Selain itu batas *indeks* yang dapat ditampung mencapai 32 (tiga puluh dua) *indeks* pada tiap Tabelnya.
7. *Connectivity*, dapat melakukan koneksi dengan *client* menggunakan *protocol* TCP/IP, Unix socket (*Unix*), atau *Named pipes* (NP).
8. *Localisation*, dapat mendeteksi pesan kesalahan pada *client* dengan mengunakan lebih dari 20 (dua puluh) bahasa.
9. *Interface*, memiliki antar muka (*interface*) terhadap beberapa aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).
10. *Clients and Tools*, dilengkapi dengan berbagai *tool* yang dapat digunakan untuk administrasi *database*, dan pada setiap *tool* yang ada disertakan petunjuk *online*.

2.10 Pengujian Black Box

Pengujian menggunakan sekumpulan aktifitas validasi, dengan pendekatan *black box testing*. Menurut Shalahuddin dan Rosa (2011), *black box testing* adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.