BAB II
LANDASAN TEORI

2.1. Prediksi

Prediksi adalah suatu proses memperkirakan secara sistematis tentang sesuatu yang paling mungkin terjadi di masa depan berdasarkan informasi masa lalu dan sekarang yang dimiliki, agar kesalahannya (selisih antara sesuatu yang terjadi dengan hasil perkiraan) dapat diperkecil. Prediksi tidak harus memberikan jawaban secara pasti kejadian yang akan terjadi, melainkan berusaha untuk mencari jawaban sedekat mungkin yang akan terjadi [2]. Pengertian Prediksi sama dengan ramalan atau perkiraan. Menurut kamus besar bahasa Indonesia, prediksi adalah hasil dari kegiatan memprediksi atau meramal atau memperkirakan nilai pada masa yang akan datang dengan menggunakan data masa lalu [3]. Prediksi menunjukkan apa yang akan terjadi pada suatu keadaan tertentu dan merupakan input bagi proses perencanaan dan pengambilan keputusan.

Prediksi bisa berdasarkan metode ilmiah ataupun subjektif belaka. Sebagai contoh, prediksi cuaca selalu berdasarkan data dan informasi terbaru yang didasarkan pengamatan termasuk oleh satelit. Begitupun prediksi gempa, gunung meletus ataupun bencana secara umum. Namun, prediksi seperti pertandingan sepakbola, olahraga, dll umumnya berdasarkan pandangan subjektif dengan sudut pandang sendiri yang memprediksinya. Permulaan awal, walaupun pengkajian yang mendalam mengenai alternatif masa depan adalah suatu disiplin baru, barangkali orang telah menaruh perhatian besar tentang apa yang akan terjadi kemudian semenjak manusia mulai mengetahui sesuatu. Populasi para peramal pada zaman kuno dan abad pertengahan merupakan satu manifestasi dari keinginan tahu orang tentang masa depannya. Perhatian tentang masa depan ini berlangsung terus bahkan berkembang menjadi kolom astrologi yang disindikatkan pada tahun 1973.

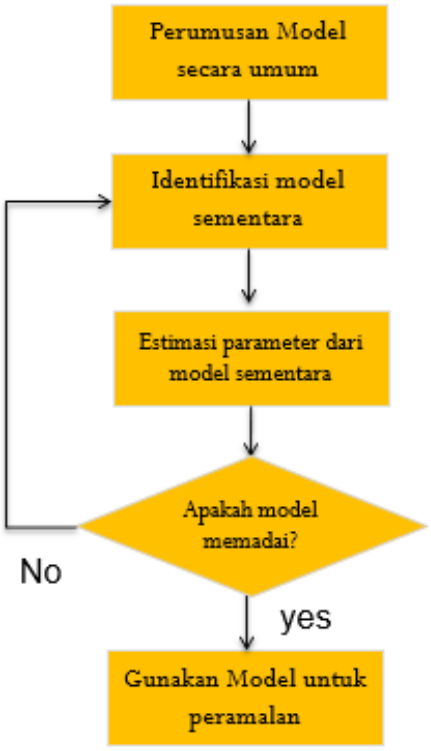
Secara Eksplisit, pembahasan mengenai teori peramalan kebijakan sangatlah sedikit. Namun, secara implisit, peramalan kebijakan terkait menjadi satu dengan 6-7 proses analisa kebijakan. Karena didalam menganalisa kebijakan, untuk menformulasikan sebuah rekomendasi kebijakan baru, maka diperlukan adanya peramalan-peramalan atau prediksi mengenai kebijakan yang akan diberlakukan dimasa yang akan datang. Namun, satu dari sekian banyak prosedur yang ditawarkan oleh para pakar Dunn, masih memberikan pembahasan tersendiri mengenai peramalan kebijakan. Menurut Dunn, Peramalan Kebijakan (policy forecasting) merupakan suatu prosedur untuk membuat informasi factual tentang situasi social masa depan atas dasar informasi yang telah ada tentang masalah kebijakan.

2.2. Metode ARIMA

Autoregressive integrated moving-average (ARIMA) merupakan metode peramalan berdasarkan sintesa terhadap pola data historis. Metode Box-Jenkins (ARIMA) tidak mengasumsikan adanya pola data historis tertentu sebagaimana halnya metode-metode peramalan yang lain. Sebuah model matematis dikatakan mirip (fit well) data time-series apabila kondisi ini terpenuhi:

1. Residu antara model peramalan data historis kecil,
2. residu terdistribusi secara acak (distribusi normal),
3. Residu bersifat independen.

Metode ARIMA merupakan gabungan antara metode *autoregressive* dan *moving average*. Metode ini bersifat rekursif dalam hal menentukan model yang paling sesuai untuk sebuah data time series dengan cara membandingkan distribusi autokorelasi antara data dengan model teoritis [4]. Perumusan metode ARIMA dapat dilihat pada gambar 2.1



Gambar II. 1. Tahapan Peramalan ARIMA

Untuk melakukan suatu peramalan menggunakan metode ARIMA dibutuhkan beberapa tahap yaitu :

1. Identifikasi Model

Langkah pertama pada model identifikasi adalah menentukan apakah sebuah data time-series bersifat stasioner (nilai rata-rata tidak bergeser sepanjang waktu). Apabila data tidak bersifat stasioner, maka konversi data harus dilakukan (agar menjadi stasioner) dengan menggunakan metoda diferensiasi. Metode diferensiasi digunakan untuk mengkonversi data yang tidak stasioner agar menjadi data yang stasioner (sebagai syarat implementasi metode ARIMA). Diferensiasi orde-1 didefinisikan dalam persamaan 2.1. dibawah :

$$X' = X_t - X_{t-1} \quad (2.1)$$

Adapun persamaan orde-2 yang dapat didefinisikan dalam persamaan 2.2. dibawah :

$$X'' = X'_t - X'_{t-1} \quad (2.2)$$

$$X'' = (X_t - X_{t-1}) - (X_{t-1} - X_{t-2})$$

Setelah data time-series merupakan data stationer, langkah selanjutnya adalah menentukan model yang akan digunakan. Penentuan model dilakukan dengan cara membandingkan koefisien autokorelasi (ACF) dan autokorelasi parsial (PACF) dari data dengan model ARIMA untuk menentukan model yang paling sesuai. Tujuannya adalah untuk mengetahui pola yang berkaitan dengan waktu di dalam sebuah data deret waktu. Sedangkan autorelokasi parsial (PACF) digunakan sebagai PACF dengan model lain untuk menentukan model yang paling sesuai. Namun sebelum penentuan nilai ACF dan PACF perlu diketahui jumlah *lag* sebagai acuan untuk penentu ordo pada model ARIMA. Untuk menghitung jumlah lag yang diperlukan digunakan persamaan 2.3 sebagai berikut

k = n/4 (2.3)

Dimana, *k* adalah jumlah lag n adalah jumlah data

Persamaan autorelokasi (ACF) dapat didefinisikan dengan persamaan 2.4. dibawah

$$r_k = \frac{\sum_{t=1}^{n-k}(X_t - \bar{X})(X_{t+k} - \bar{X})}{\sum_{t=1}^{n}(X_t - \bar{X})^2} \quad (2.4)$$

Kemudian Adapun persamaan autorelokasi parsial (PACF) yang didefinisikan dengan persamaan 2.5. dibawah

$$r_{kk} = \frac{r_k - \sum_{j=1}^{k-1} r_{k-1,j}\, r_{k-j}}{1 - \sum_{j=1}^{k-1} r_{k-1}, r_j} \quad (2.5)$$

Dimana :

$r_{kj} = r_{k-1,j} - r_{kk} r_{k-1,k-1}$ untuk j = 1,2, …, k-1

$r_{11} = r_1$

$$r_{22} = \frac{r_2 - r_{11} r_1}{1 - r_{11} r_1} = \frac{r_2 - {r_1}^2}{1 - {r_1}^2}$$

Untuk lag ke-3, memerlukan

$r_{21} = r_{11} - r_{22} r_{11}$

Setelah ditemukan nilai ACF dan nilai PACF, maka selanjutnya diperlukan distribusi sampling nilai ACF. Distribusi nilai ACF diperlukan untuk mengetahui apakah data yang dimiliki stasioner atau tidak. Suatu data dikatakan stasioner apabila koefisien ACF untuk lag yang cukup Panjang memberikan nilai r_k mendekati nol. Untuk memastikannya, digunakan persamaan seperti persamaan 2.6 dibawah :

$$se_{rk} = \frac{1}{\sqrt{n}} \quad (2.6)$$

Dimana n merupakan banyak data.

Sebagai contoh, apabila suatu data memiliki banyak data sebanyak 40 buah data. Maka untuk memastikan apakah semua datanya stasioner atau tidak maka digunakan persamaan seperti persamaan 2.5. berikut perhitungannya :

$$se_{rk} = \frac{1}{\sqrt{40}}$$

Maka,

$$se_{rk} = 0{,}158$$

Sebuah data dikatakan stasioner dengan keyakinan 95% apabila nilai koefisien ACF berada pada rentang seperti pada persamaan 2.7

$$0 - 1{,}96 * se_{rk} \leq r_k \leq 0 + 1.96 * se_{rk} \quad (2.7)$$

Maka,

$$-1{,}96 * 0{,}158 \leq r_k \leq 1.96 * 0{,}158$$

$$-0{,}31 \leq r_k \leq 0{,}31$$

Artinya adalah, apabila r_k-nya bernilai lebih besar dari -0,31 dan lebih kecil dari 0,31, maka data tersebut sudah stasioner dapat melanjutkan ke tahap identifikasi model yang tepat. Didalam peramalan dengan metode ARIMA, terdapat tiga model yaitu metode *Autoregressive*, metode *Moving Average* dan juga gabungan keduanya (*Autoregressive and Moving Average*). Untuk persyaratan model yang digunakan dapat dilihat pada table 2.1

Tabel II. 1. Prasyarat Penggunaan Model ARIMA

Model	ACF	PACF
MA(q) : *Moving Average* dengan ordo q	*Cuts Off* setelah lag ke-q	Menurun secara eksponensial atau membentuk gelombang sinus (*sinewave*)
AR(p) : *Autoregressive* dengan ordo p	Menurun secara eksponensial atau membentuk gelombang sinus (*sinewave*)	*Cuts Off* setelah lag ke-p
ARMA(q,p) : Gabungan model AR dan MA	Menurun secara eksponensial atau membentuk gelombang sinus (*sinewave*)	Menurun secara eksponensial atau membentuk gelombang sinus (*sinewave*)

Untuk model AR digunakan persamaan 2.8 sebagai berikut

$$Y_i = a_0 + a_1 Y_{i-1} + a_2 Y_{i-2} + \cdots + a_p Y_{i-p} + e_i \quad (2.8)$$

Sedangkan untuk mencari nilai estimasi AR digunakan persamaan 2.9 sebagai berikut

$$\hat{Y}_i = a_0 + a_1 Y_{i-1} + a_2 Y_{i-2} + \cdots + a_p Y_{i-p} \quad (2.9)$$

Dimana :

a. $\hat{Y}_i$ = nilai pencocokan (*fitted value*) dari data untuk waktu ke-i

b. Y_{i-1} = nilai observasi data untuk waktu ke-(i-1)

c. Y_{i-2} = nilai observasi data untuk waktu ke-(i-2)

d. Y_{i-p} = nilai observasi data untuk waktu ke-(i-p)

e. $a_0 a_1 a_2 a_p$ = koefisien estimasi dari regresi

Untuk model MA digunakan persamaan 2.10 sebagai berikut

$$Y_i = b_0 + b_1\epsilon_{i-1} + a_2\epsilon_{i-2} + \cdots + b_q\epsilon_{i-q} + \epsilon_i \quad (2.10)$$

Sedangkan untuk mencari nilai estimasi AR digunakan persamaan 2.11 sebagai berikut

$$\hat{Y}_i = b_0 + b_1\epsilon_{i-1} + b_2\epsilon_{i-2} + \cdots + a_p\epsilon_{i-q} \quad (2.11)$$

Dimana :

a. $\hat{Y}_i$ = nilai pencocokan (*fitted value*) dari data untuk waktu ke-i

b. ϵ_{i-1} = nilai kesalahan data untuk waktu ke-(i-1)

c. ϵ_{i-2} = nilai kesalahan data untuk waktu ke-(i-2)

d. ϵ_{i-q} = nilai kesalahan data untuk waktu ke-(i-q)

e. $b_0 b_1 b_2 b_q$ = koefisien estimasi dari regresi

Secara umum, spesifikasi ARIMA dituliskan sebagai ARIMA (*p,d,q*) dimana :
p = orde dari komponen *Autoregressive*
d = orde dari diferensiasi
q = orde dari komponen *Moving Average*

a. ARIMA (*p,d,q*)

Gabungan antara AR(*p*) dan MA(*q*) membentuk persamaan ARIMA (*p,q*) yang dapat dilihat pada persamaan 2.12

$$Y_i = a_0 + a_1 Y_{i-1} + a_2 Y_{i-2} + \cdots + a_p Y_{i-p} - b_1\epsilon_{i-1} - b_2\epsilon_{i-2} - \cdots - b_q\epsilon_{i-q} + \epsilon_i \quad (2.12)$$

Adapun estimasi ARIMA (*p,q*) didefinisikan dengan persamaan 2.13 seperti berikut :

$$\hat{Y}_i = a_0 + a_1 Y_{i-1} + a_2 Y_{i-2} + \cdots + a_p Y_{i-p} - b_1\epsilon_{i-1} - b_2\epsilon_{i-2} - \cdots - b_q\epsilon_{i-q} \quad (2.13)$$

b. ARIMA (0,0,0)

ARIMA (0,0,0) merupakan model *time-series* yang bersifat stasioner (*random*). Nilai observasi Y_i ini ditentukan oleh rata-rata μ dan komponen kesalahan ϵ_i sehingga menjadi persamaan 2.14 dibawah ini

$$Y_i = \mu + \epsilon_i \quad (2.14)$$

Persamaan diatas didefinisikan sebagai ARIMA (0,0,0) karena tidak memiliki komponen AR, nilai diferensiasi, dan juga komponen MA

c. ARIMA (1,0,0)

ARIMA (1,0,0) merupakan model *time-series* yang bersifat stasioner. Nilai observasi Y_i ini bergantung pada Y_{i-1} dan juga koefisien dari auto-regresf a_1 sehingga menjadi persamaan 2.15 dibawah ini

$$Y_i = a_0 + a_1 Y_{i-1} + \epsilon_i \quad (2.15)$$

Dimana :

$$a_0 = \mu(1 - a_1 - a_2 - \cdots - a_p$$

d. ARIMA (0,1,0)

ARIMA (0,1,0) merupakan model *time-series* yang bersifat non-stasioner. Model ARIMA (0,1,0) didefinisikan dengan persamaan 2.16 dibawah ini

$$Y_i = Y_{i-1} + \epsilon_i \quad (2.16)$$

ARIMA (0,1,0) memiliki kemiripan dengan model AR(1) dengan konstanta bernilai 1

e. ARIMA (0,0,1)

ARIMA (0,0,1) merupakan model *time-series* yang bersifat stasioner. Nilai observasi Y_i ini bergantung pada nilai ϵ_{i-1} dan juga koefisien dari *Moving Average* b_1 sehingga menjadi persamaan 2.17 dibawah ini

$$Y_i = b_0 + b_1 \epsilon_{i-1} + \epsilon_i \quad (2.17)$$

f. ARIMA (1,0,1)

ARIMA (1,0,1) merupakan kombinasi model AR dan juga model MA. Nilai observasi Y_i ini bergantung pada nilai Y_{i-1} dan juga nilai residu ϵ_{i-1} sehingga menjadi persamaan 2.18 dibawah ini

$$Y_t = a_0 + a_1 Y_{i-1} + \epsilon_i - b_1 \epsilon_{i-1} \quad (2.18)$$

2. Estimasi Model

Setelah melakukan identifikasi model, selanjutnya yaitu melakukan proses estimasi model yang meliputi :

a. Melakukan estimasi parameter terhadap model ARIMA sementara yang telah ditentukan. Untuk menghitung parameter digunakan perangkat pembantu seperti Minitab, EViews, dan lain-lain.
b. Menghitung nilai *Mean Percentage Error* (MPE) sebagai perangkat pembanding untuk memilih model yang paling layak [5]. Nilai MPE didefinisikan dengan persamaan 2.19 dibawah

$$MPE = \frac{\sum_{t=1}^{n} \frac{Y_t}{\hat{Y}_t} x\ 100\%}{n} \quad (2.19)$$

Dimana *n* adalah jumlah data.

3. Cek Kelayakan Model

Untuk memeriksa kelayakan model, perlu diketahui bahwa model dianggap layak apabila memiliki nilai MPE terkecil. Kemudian model juga harus memiliki parameter yang signifikan (*p-value* = 0). Jika tidak ada parameter yang signifikan, maka harus mengidentifikasi ulang model yang akan digunakan.

4. Peramalan dengan Model

Setelah model dianggap layak dan didapatkan persamaannya, maka dapat dilakukan peramalan.

2.3. Pemrogram berorientasi Objek

Pemrograman berorientasi objek adalah paradigma pemrograman yang berorientasikan kepada objek yang merupakan suatu metode dalam pembuatan program, dengan tujuan untuk menyelesaikan kompleksnya berbagai masalah

program yang terus meningkat. Objek adalah *entitas* yang memiliki atribut, karakter (*behaviour*) dan kadang kala disertai kondisi. Pemrograman berorientasi objek ditemukan pada Tahun 1960, dimana berawal dari suatu pembuatan program yang terstruktur (*structured programming*). Metode ini dikembangkan dari bahsa C dan Pascal. Dengan program yang terstruktur inilah untuk pertama kalinya kita mampu menulis program yang begitu sulit dengan lebih mudah [6]

Ide dasar pada OOP adalah mengkombinasikan data dan fungsi untuk mengakses data menjadi sebuah kesatuan unit yang dikenal dengan nama objek. Objek adalah struktur data yang terdiri dari bidang data dan metode bersama dengan interaksi mereka untuk merancang aplikasi dan program komputer. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

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

Dalam pemrograman berorientasi objek, terdapat beberapa konsep dasar. Yaitu :

1. Kelas (*Class*)

Kelas (class) merupakan penggambaran satu set objek yang memiliki atribut yang sama. Kelas mirip dengan tipe data ada pemrograman non objek, akan tetapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya. Kelas baru dapat dibentuk lebih spesifik dari kelas ada umumnya.kelas merupakan jantung dalam pemrograman berorientasi objek.

2. Objek (*Object*)

Objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan teknik yang efektif dalam menemukan cara yang tepat dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak. Orientasi objek merupakan teknik pemodelan sistem riil yang berbasis objek.

Objek adalah entitas yang memiliki atribut, karakter dan kadang kala disertai kondisi. Objek mempresentasikan sesuai kenyataan seperti siswa, mempresentasikan dalam bentuk konsep seperti merek dagang, juga bisa menyatakan visualilasi seperti bentuk huruf (font).

3. Abstraksi (*Abstraction*)

Kemampuan sebuah program untuk melewati aspek informasi yang diolah adalah kemampuan untuk fokus pada inti permasalahan. Setiap objek dalam sistem melayani berbagai model dari pelaku abstrak yang dapat melakukan kerja, laporan dan perubahan serta berkomunikasi dengan objek lain dalam sistem, tanpa harus menampakkan kelebihan diterapkan.

4. Enkapsulasi (*Encapsulation*)

Pembungkusan merupakan penggabungan potongan-potongan informasi dan perilaku-perilaku spesifik yang bekerja pada informasi tersebut, kemudian mengemasnya menjadi sesuatu yang disebut objek. Enkapsulasi adalah proses memastikan pengguna sebuah objek tidak dapat menggantikan keadaan dari sebuah objek dengan cara yang tidak sesuai prosedur. Artinya, hanya metode yang terdapat dalam objek tersebut yang diberi izin untuk mengakses keadaan yang diinginkan.

Setiap objek mengakses interface yang menyabutkan bagaimana objek lainnya dapat berintegrasi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

5. Polimorfisme

Polimorfise merupakan suatu fungsionalitas yang diimplikasikan dengan berbagai cara yang berbeda. Pada program berorientasi objek, pembuat program dapat memiliki berbagai implementasi untuk sebagian fungsi tertentu.

6. Inheritas (*Inheritance*)

Konsep inheritas mempunyai fungsi mengatur polimorfise dan enkapsulasi dengan mengizinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada. Objek-objek ini dapat membagi dan memperluas perilaku mereka tanpa mengimplementasikan perilaku tersebut.

2.4. UML

UML (*Unified Modeling Language*) merupakan pengganti dari metode analisis berorientasi object dan design berorientasi object (OOAD&D/*object oriented analysis and design*) yang dimunculkan sekitar akhir tahun 80-an dan awal tahun 90-an [7]. UML merupakan gabungan dari metode *Booch,Rumbaugh* (OMT) dan *Jacobson*. Tetapi UML mencakup lebih luas daripada OOAD. Pada pertengahan saat pengembangan UML, dilakukan standarisasi proses dengan OMG (*Object Management Group*) dengan harapan UML bakal menjadi bahasa standar pemodelan pada masa yang akan datang (yang sekarang sudah banyak dipakai oleh berbagai kalangan). UML digunakan untuk memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakan konsep berorientasi object. Dan juga untuk menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin [8].

Adapun beberapa bagian UML, yaitu :

1. *View*

Digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. Beberapa Jenis *view* dalam UML antara lain : *use case view, logical view, component view, concurrency view,* dan *deployment view*.

2. *Use case View*

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. Actor yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. *View* ini digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams.* View ini digunakan terutama untuk pelanggan,perancang (*designer*), pengembang(*developer),* dan penguji sistem(*tester*).

3. *Logical View*

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object,* dan *relationship*) dan kolaborasi dinamis yang terjadi ketika object mengirim pesan ke object lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam *class diagrams* untuk struktur statis dan dalam *state, sequence, collaboration,* dan *activity diagram* untuk model dinamisnya. *View* ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

4. *Component View*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administratif lainnya. *View* ini digambarkan dalam *component view* dan digunakan untuk pengembang (*developer*).

5. *Concurrency View*

Membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam diagram dinamis (*state, sequence, collaboration*, dan *activity diagrams*) dan

diagram implementasi (*component* dan *deployment diagrams*) serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

6. *Deployment View*

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. *View* ini digambarkan dalam *deployment diagrams* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

2.5. Website

Sebuah situs web (sering pula disingkat menjadi situs saja, *website* atau *site*) adalah sebutan bagi sekelompok halaman web (*web page*), yang umumnya merupakan bagian dari suatu nama domain (*domain name*) atau subdomain di *World Wide Web* (WWW) di Internet.

Sebuah *web page* adalah dokumen yang ditulis dalam format HTML (Hyper Text Markup Language), yang hampir selalu bisa diakses melalui HTTP, yaitu protokol yang menyampaikan informasi dari *server website* untuk ditampilkan kepada para pemakai melalui web browser baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait dimana masing-masing dihubungkan dengan jaringan-jaringan halaman (*hyperlink*) [9]. Bersifat statis apabila isi informasi *website* tetap, jarang berubah, dan isi informasinya searah hanya dari pemilik *website*. Bersifat dinamis apabila isi informasi *website* selalu berubah-ubah, dan isi informasinya interaktif dua arah berasal dari pemilik serta pengguna website. Contoh *website* statis adalah berisi profil perusahaan, sedangkan *website* dinamis adalah seperti *Friendster, Multiply*, dan lain-lain. Dalam sisi pengembangannya, *website* statis hanya bisa diupdate oleh pemiliknya saja, sedangkan website dinamis bisa diupdate oleh pengguna maupun pemilik. Halaman-halaman sebuah situs web diakses dari sebuah URL yang menjadi “akar” (*root*), yang disebut *homepage* (halaman induk; sering diterjemahkan menjadi “beranda”, “halaman muka”), dan biasanya disimpan dalam server yang sama.

Tidak semua situs web dapat diakses dengan gratis. Beberapa situs *web* memerlukan pembayaran agar dapat menjadi pelanggan, misalnya situs-situs yang menampilkan pornografi, situs-situs berita, layanan surat elektronik (*e-mail*), dan lain-lain. Website ini dibuka melalui sebuah program penjelajah (*Browser*) yang berada di sebuah komputer. Program penjelajah yang bisa digunakan dalam komputer diantaranya: IE (*Internet Explorer*), *Mozilla Firefox*, *Netscape*, *Opera* dan yang terbaru adalah *Google Chrome*.

2.6. Database

Basis data adalah kumpulan file-file yang saling berelasi, relasi tersebut biasa ditunjukan dengan kunci dari tiap file yang ada. Satu basis data menunjukkan kumpulan data yang dipakai dalam satu lingkup informasi. Dalam satu file terdapat record-record yang sejenis, sama besar, sama bentuk, merupakan satu kumpulan entity yang seragam. Satu record terdiri dari fieldfield yang saling berhubungan untuk menunjukan bahwa field tersebut dalam satu pengertian yang lengkap dan direkam dalam satu record. Suatu sistem manajemen basis data berisi satu koleksi data yang saling berelasi dan satu set program untuk mengakses data tersebut. Jadi sistem manajemen basis data dan set program pengelola untuk menambah data, menghapus data, mengambil data dan membaca data [10].

2.7. *Laravel Framework*

Laravel framework merupakan suatu *Framework* yang berfungsi untuk memaksimalkan penggunaan PHP dalam pengembangan *Website* [11]. Laravel diluncurkan sejak tahun 2011 dan mengalami pertumbuhan yang cukup eksponensial. Di tahun 2015, Laravel adalah framework yang paling banyak mendapatkan bintang di Github. Sekarang framework ini menjadi salah satu yang populer di dunia, tidak terkecuali di Indonesia.

Laravel fokus di bagian end-user, yang berarti fokus pada kejelasan dan kesederhanaan, baik penulisan maupun tampilan, serta menghasilkan fungsionalitas aplikasi web yang bekerja sebagaimana mestinya. Hal ini membuat *developer*

maupun perusahaan menggunakan framework ini untuk membangun apa pun, mulai dari proyek kecil hingga skala perusahaan kelas atas.

Laravel mengubah pengembangan website menjadi lebih elegan, ekspresif, dan menyenangkan, sesuai dengan jargonnya "*The PHP Framework For Web Artisans*". Selain itu, Laravel juga mempermudah proses pengembangan website dengan bantuan beberapa fitur unggulan, seperti *Template Engine*, *Routing*, dan *Modularity*.

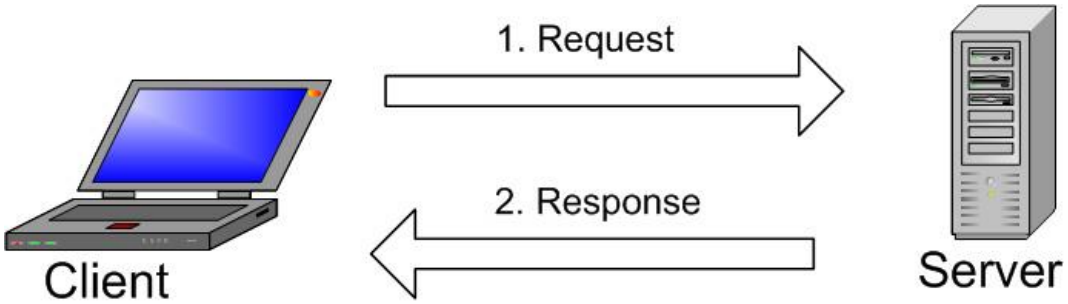
2.8. PHP

PHP Adalah bahasa scripting server-side, Bahasa pemrograman yang digunakan untuk mengembangkan situs web statis atau situs web dinamis atau aplikasi Web. PHP singkatan dari *Hypertext Pre-processor*, yang sebelumnya disebut *Personal Home Pages*. Script sendiri merupakan sekumpulan instruksi pemrograman yang ditafsirkan pada saat runtime. Sedangkan Bahasa scripting adalah bahasa yang menafsirkan skrip saat runtime. Dan biasanya tertanam ke dalam lingkungan perangkat lunak lain. Karena php merupakan *scripting server-side* maka jenis bahasa pemrograman ini nantinya script/program tersebut akan dijalankan/diproses oleh server. Berbeda dengan javascript yang client-side. PHP adalah bahasa pemrograman umum yang berarti php dapat disematkan ke dalam kode HTML, atau dapat digunakan dalam kombinasi dengan berbagai sistem *template* web, sistem manajemen konten web, dan kerangka kerja web [12].

2.9. MySql

MySQL adalah sistem manajemen database relasional open source (RDBMS) dengan client-server model. Sedangkan RDBMS merupakan software untuk membuat dan mengelola database berdasarkan pada model relasional. MySQL AB, sebuah perusahaan asal Swedia, menjadi yang pertama dalam mengembangkan MySQL di tahun 1994. Hak kepemilikan MySQL kemudian diambil secara menyeluruh oleh perusahaan teknologi Amerika Serikat, Sun Microsystems, ketika mereka membeli MySQL AB pada tahun 2008. Di tahun

2010, Oracle yang adalah salah satu perusahaan teknologi terbesar di Amerika Serikat mengakuisisi Sun Microsystems. Semenjak itulah, MySQL sepenuhnya dimiliki oleh Oracle [13]. Untuk Struktur Dasar *Client-Server* dapat dilihat pada gambar 2.2



Gambar II. 2 Struktur Dasar *Client - Server*

Gambar di atas menjelaskan struktur dasar dari client-server. Satu atau banyak perangkat terhubung ke server melalui network atau jaringan khusus. Setiap client dapat membuat permintaan (request) dari antarmuka pengguna grafis atau graphical user interface (GUI) di layar, dan server akan membuat output yang diinginkan, sepanjang server dan juga client memahami instruksi dengan benar. Idealnya, proses utama yang terjadi di ruang lingkup MySQL sama, yaitu:

1. MySQL membuat database untuk menyimpan dan memanipulasi data, serta menentukan keterkaitan antara masing-masing tabel.
2. Client membuat permintaan (request) dengan mengetikkan pernyataan SQL yang spesifik di MySQL.
3. Aplikasi server akan merespons dengan memberikan informasi yang diminta. Informasi ini nantinya muncul di sisi klien.