# BAB II

# LANDASAN TEORI

## 2.1. Konsep Dasar *Web*

Aplikasi berbasis *web* sudah marak digunakan dengan fasilitas pembantu seperti komputer dan terkoneksi dengan *internet*. Komputer-komputer di dalam jaringan (*web*) terkoneksi untuk mendapat data melalui *data transfer*. *Data transfer* berfungsi sebagai cara untuk memindahkan informasi dari komputer ke komputer lain yang saling terkoneksi di dalam jaringan tersebut.

### 2.1.1. *Website*

Menurut Ginanjar (2014:5) mendefinisikan bahwa "*website* adalah rangkaian atau sejumlah halaman di *internet* yang memiliki topik saling terkait untuk mempresentasikan suatu informasi".

Sedangkan menurut Yuhefizar (2013:2) mengemukakan bahwa "*website* adalah keseluruhan halaman-halaman *web* yang terdapat dari sebuah domain yang mengandung informasi".

Dari definisi para ahli yang telah diuraikan di atas, dapat disimpulkan bahwa *website* merupakan rangkuman dari keseluruhan halaman-halaman *web* yang ada pada sebuah domain yang mengandung informasi teks, gambar diam atau gerak, animasi, suara, yang bersifat dinamis atau statis yang membentuk suatu rangkaian bangunan yang saling terkait, yang dapat terhubung dengan jaringan.

**A. *Web Server***

*Web server* merupakan sebuah perangkat lunak *server* yang berfungsi menerima permintaan dari klien yang dikenal dengan *browser web* dan

mengirimkan kembali hasilnya dalam bentuk halaman-halaman *web* melalui protokol HTTP atau HTTPS dan bertugas mengelola halaman-halaman *web* dan dokumen-dokumen lainnya (Solichin, 2016:6).

> Menurut Supono & Putratama (2016:6) mengemukakan bahwa:
> Paket *web server* adalah sebuah perangkat lunak *server* yang berfungsi untuk menerima permintaan dalam bentuk situs *web* melalui HTTP atau HTTPS dari klien itu, yang dikenal sebagai *browser web* dan mengirimkan kembali (reaksi) hasil dalam bentuk situs yang biasanya merupakan dokumen HTML.

Berdasarkan pendapat para ahli di atas, dapat disimpulkan bahwa *web server* merupakan perangkat lunak yang mengandung bermacam *protocol web* untuk dapat menjalankan perintah dari *client*, *server internet* yang digunakan sebagai k*one*ksi dan transfer data (HTML, asp, aspx, php, js, dan lain).

**B. *Web Browser***

Menurut Solichin (2016:9) mendefinisikan bahwa "peramban *web* atau lebih dikenal dengan *web browser* merupakan perangkat lunak yang berfungsi untuk menerima dan menyajikan sumber informasi di *internet*".

*Web browser* merupakan perangkat lunak yang dapat memproses paket HTTP dan menampilkannya kembali kepada *user* dengan format HTML (Supono & Putratama, 2016:5).

Berdasarkan pengertian diatas, dapat disimpulkan bahwa *web browser* adalah aplikasi yang digunakan untuk menampilkan halaman *web* untu proses pengolahan informasi, pengambilan dan penyajian informasi pada *website*.

### 2.1.2. Bahasa Pemrograman

Bahasa pemrograman berbasis *web* yang digunakan terdiri *hypertext preprocessor* (PHP), *hypertext markup language* (HTML), *cascading style sheet*

(CSS), Javascript, JQuery dan Bootstraps. Pembahasan dari bahasa pemrograman yang digunakan diuraikan sebagai berikut:

**A. *Hypertext Preprocessor* (PHP)**

Menurut Supono & Putratama (2016:3) mendefinisikan bahwa "PHP (PHP: *Hypertext Preprocessor*) adalah suatu bahasa pemrograman yang digunakan untuk menerjemahkan baris kode program menjadi kode mesin yang dapat dimengerti oleh komputer yang berbasis *server-side* yang dapat ditambahkan ke dalam HTML".

Sedangkan, menurut Solichin (2016:11) mengemukakan bahwa "PHP merupakan salah satu bahasa pemrograman berbasis *web* yang ditulis oleh dan untuk pengembang *web*".

Berdasarkan pendapat dari para ahli di atas, dapat disimpulkan bahwa PHP merupakan bahasa pemrograman yang dapat mengolah *database*, *content website* sehingga *website* yang dibuat merupakan *web* dinamis.

**B. *Hypertext Markup Language* (HTML)**

Menurut Solichin (2016:10) mengemukakan bahwa "HTML merupakan bahasa pemrograman *web* yang memberitahukan peramban *web* (*web browser*) bagaimana menyusun dan menyajikan konten di halaman *web*".

*Hypertext markup language* (HTML) adalah bahasa *markup* standar untuk membuat dan menggambarkan struktur halaman *web* yang terdiri dari elemen HTML sebagai blok bangunan sebuah halaman dan direpresentasikan dengan *tag* (Atmoko, 2018:1).

Berdasarkan definisi dari para ahli yang telah dikemukakan di atas, dapat disimpulkan bahwa *hypertext markup language* (HTML) merupakan bahasa

standar yang digunakan untuk menyusun dan menyebarkan informasi serta menampilkan halaman *web*.

**C. *Cascading Style Sheet* (CSS)**

*Cascading style sheet* (CSS) merupakan bahasa pemrograman yang berfungsi untuk mempercantik tampilan *web* (Solichin, 2016:10).

Menurut Prasetio (2014:252) menyatakan bahwa "CSS adalah suatu teknologi yang digunakan untuk memperindah tampilan halaman *website* (situs)".

Berdasarkan pengertian diatas, maka dapat disimpulkan bahwa *cascading style sheets* (CSS) merupakan bahasa yang digunakan dan membantu *programmers* dalam merancang sebuah tampilan *website* dan menimbulkan efek animasi yang bagus.

**D. *Javascript***

Javassript dikembangkan oleh Netscape dengan nama awal *LiveScript* yang berfokus pada proses pengolahan data di sisi *client* dan menyajikan komponen *web* yang lebih interaktif serta berfungsi untuk menambah fungsionalitas dan kenyamanan halaman *web* (Solichin, 2016:11).

Menurut Suryana & Koesheryatin (2014:181) mengemukakan bahwa "Javascript adalah bahasa *script* berdasar pada objek yang memperbolehkan pemakai untuk mengendalikan banyak aspek interaksi pemakai pada suatu dokumen HTML dimana. Dimana objek tersebut dapat berupa suatu *window*, *frame*, URL, dokumen, *form*, *button*, atau *item* yang lain".

Dapat disimpulkan bahwa *javascript* merupakan bahasa pemrograman yang berbasis *client* dan *script* untuk tampilan pendukung pada *website* sehingga membuat halaman menjadi lebih menarik dan interaktif.

**E. *JQuery***

Menurut Prasetia (2013:85) mengemukakan bahwa "*jQuery* merupakan sebuah *framework* (*library*) *JavaScript* yang ditujukan untuk membantu Anda dalam menghasilkan aplikasi-aplikasi *cross-platform* yang responsif dan mudah".

*JQuery* merupakan sebuah *plugin* yang berfungsi sebagai *framework JavaScript* yang sering digunakan dalam pengembangan *web* (Winarno, 2014:52).

Dapat ditarik sebuah kesimpulan *JQuery* merupakan pustaka *javascript* yang memudahkan bagi para *programmer* dalam penulisan kode *javascript*.

**F. *Bootstrap***

Menurut Subagia (2018:45) mendefinisikan bahwa "*bootstrap* adalah *template* desain *web* dengan fitur plus (*framework CSS*). *Bootstrap* diciptakan untuk mempermudah proses desain *web* bagi berbagai tingkat pengguna, mulai dari level pemula hingga yang sudah berpengalaman".

Sedangkan menurut Rozi (2016:225) mengemukakan bahwa "*bootstrap* adalah paket aplikasi siap pakai untuk membuat *template web*".

Berdasarkan pendapat dari para ahli di atas, dapat disimpulkan bahwa *bootstrap* merupakan paket aplikasi yang menyediakan *template* dengan format *cascading style sheet* (CSS) untuk mendesain *web* agar tampilan *web* menjadi lebih menarik.

### 2.1.3. Basis Data

Sukamto & Shalahuddin (2015:43) mengemukakan bahwa "sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan".

Sedangkan menurut Ladjamudin (2013:129) menyatakan bahwa "*database* adalah sekumpulan *data store* (bisa dalam jumlah yang sangat besar) yang tersimpan dalam *magnetic disk*, *optica ldisk*, *magnetic drum* atau media penyimpanan sekunder lainnya".

Berdasarkan pengertian diatas, maka dapat disimpulkan bahwa basis data merupakan suatu media penyimpanan data dimana dapat menampung berbagai macam data, dimana sistem pendataannya di data dengan manajemen data yang baik.

**A. *Structured Query Language* (SQL)**

Menurut Sukamto & Shalahuddin (2015:46) mengemukakan bahwa "SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengelola data pada DBMS".

Sedangkan menurut Subagia (2018:67) mendefinisikan bahwa "*structure query language* (SQL) merupakan bahasa yang banyak digunakan dalam berbagai produk *database*".

Berdasarkan pernyataan di atas, maka *structured query language* (SQL) merupakan suatu bahasa standar yang digunakan untuk mengakses data dalam sebuah basis data dan melakukan pengolahan data.

Pengaksesan data pada DBMS dengan SQL yang secara umum terdiri dari empat (4) hal (Sukamto & Shalahuddin, 2015:47), yaitu:

1. Memasukkan data (*insert*)

   Perintah yang digunakan untuk menambah data pada basis data.

2. Mengubah data (*update*)

   Perintah yang digunakan untuk mengubah data pada basis data.

3. Menghapus data (*delete*)

   Perintah yang digunakan untuk menghapus data pada basis data.

4. Menampilkan data (*select*)

   Perintah yang digunakan untuk menampilkan data pada basis data.

**B. MySQL**

MySQL merupakan *software database open source* yang sering digunakan untuk mengolah basis data yang menggunakan bahasa SQL (Subagia, 2018:67).

Menurut Risnandar (2013:92) mengemukakan bahwa "MySQL merupakan basis data yang bersifat *open source* sehingga banyak digunakan di dunia. Walaupun gratis, MySQL tetap berkualitas dan sudah cukup memberikan performa yang memadai".

Berdasarkan pendapat dari para ahli di atas, dapat ditarik kesimpulan bahwa MySQL merupakan aplikasi yang digunakan untuk mengolah basis data yang banyak digunakan untuk membangun aplikasi yang menggunakan *database*.

### 2.1.4. Sistem Informasi Akuntansi

Menurut Mulyani (2016:21) mendefinisikan bahwa "sistem informasi akuntansi digunakan sebagai alat untuk melakukan analisis keputusan ataupun sebagai pembuat keputusan yang terkait dengan transaksi-transaksi perusahaan".

Sedangkan menurut Mahatmyo (2014:9) mengemukakan bahwa "sistem informasi akuntansi merupakan sekelompok struktur dalam sebuah entitas yang mengelola sumber daya fisik dan sumber daya lain untuk mengubah data ekonomi menjadi informasi akuntansi, agar dapat memenuhi kebutuhan informasi berbagai pihak".

Berdasarkan pendapat para ahli di atas, dapat disimpulkan bahwa sistem informasi akuntansi merupakan suatu sistem yang terdiri dari berbagai formulir, catatan dan laporan yang telah disusun dan menghasilkan suatu informasi keuangan yang dibutuhkan oleh perusahaan.

Terdapat enam (6) komponen dalam sistem informasi akuntansi (Mulyani, 2016:22), yaitu:

1. *User*, yaitu orang yang menggunakan atau mengoperasikan sistem.
2. *Procedure* atau *instructions*, yaitu pemrosesan dan penyimpanan data kegiatan organisasi.
3. Data, yaitu representasi dari dunia nyata terkait dengan organisasi.
4. *Software*, yaitu kumpulan program komputer yang digunakan untuk memproses data.
5. *Information technology infrastructure*, yaitu struktur yang akan digunakan oleh sistem seperti misalnya, struktur jaringan komputer.
6. *Internal control and integrity measure*.

### 2.1.5. Laporan Keuangan

Laporan keuangan yang menginformasikan aset perusahaan serta perubahannya dan merupakan cerminan aktivitas dan posisi keuangan perusahaan pada periode tertentu (Lee, 2014:3).

Menurut Hery (2016:3) mengemukakan bahwa “laporan keuangan pada dasarnya adalah hasil dari proses akuntansi yang digunakan sebagai alat untuk mengkomunikasikan data keuangan atau perusahaan kepada pihak-pihak yang berkepentingan”.

Dapat disimpulkan bahwa laporan keuangan merupakan laporan yang menunjukkan kondisi finansial suatu perusahaan dalam periode tertentu.

Urutan laporan keuangan disajikan berdasarkan urutan penyajiannya. Urutan penyajian laporan keuangan (Hery, 2016:3) diuraikan sebagai berikut:

1. Laporan laba rugi (*income statement*)

   Laporan laba rugi merupakan laporan yang sistematis tentang pendapatan dan beban perusahaan untuk satu periode tertentu. Laporan laba rugi ini pada akhirnya memuat informasi mengenai hasil kinerja manajemen atau hasil kegiatan operasional perusahaan.

2. Laporan ekuitas pemilik (*statement of owner's equity*)

   Laporan ekuitas pemilik adalah sebuah laporan yang menyajikan ikhtisar perubahan dalam ekuitas pemilik suatu perusahaan untuk satu periode tertentu.

3. Neraca (*balance sheet*)

   Neraca adalah sebuah laporan yang sistematis tentang posisi aset, kewajiban dan ekuitas perusahaan per tanggal tertentu.

4. Laporan arus kas (*statement of cash flows*)

   Laporan arus kas adalah sebuah laporan yang menggambarkan arus kas masuk dan arus kas keluar secara terperinci dari masing-masing aktivitas, yaitu mulai dari aktivitas operasi, aktivitas investasi, sampai pada aktivitas pendanaan atau pembiayaan untuk satu periode waktu tertentu.

## 2.2. Peralatan Pendukung

Peralatan pendukung digunakan untuk mendukung keberhasilan pembuatan aplikasi yang menguraikan tentang uraian tambahan mengenai aplikasi pendukung

seperti *xampp* dan *sublime text*, *entity relationship diagram* (ERD) *logical record structure* (LRS), *unified modeling language* (UML) yang terdiri dari *use case diagram*, *activity diagram*, *sequence diagram* dan *deployment diagram* serta pengujian aplikasi yang menggunakan metode *black box testing*.

### 2.2.1. Aplikasi Pendukung

Beberapa aplikasi atau *software* digunakan guna mendukung keberhasilan dalam pembuatan aplikasi. Adapun aplikasi atau *software* pendukung yang digunakan terdiri dari Xampp dan *sublime text* yang diuraikan sebagai berikut.

**A. Xampp**

Xampp merupakan aplikasi *web server* yang banyak dipakai oleh pengembang *website*, terdiri dari apache *web server*, MySQL, PHP, Perl, FTP Server dan phpMyAdmin (Supono & Putratama, 2016:7).

Xampp merupakan perangkat lunak yang terdiri dari PHP, Apache, MySQL, dan phpMyAdmin sehingga menjadi satu kesatuan atau dikenal dengan *software package/installer* dimana proses dan konfigurasi dilakukan secara otomatis, mudah dan praktis (Solichin, 2016:15).

Maka dari itu dapat disimpulkan bahwa xampp *server* merupakan sebuah *software* yang mengemas Apache, MySQL dan PHP untuk membantu para *programmer* membangun dan mengembangkan sebuah *website*.

**B. *Sublime Text***

Menurut Supono & Putratama (2016:14) mendefinisikan bahwa "*sublime text* merupakan perangkat lunak *text editor* yang digunakan untuk membuat atau meng-*edit* suatu aplikasi".

*Sublime text* merupakan text editor yang memiliki ragam fitur pendukung untuk para *programmers* (Rachmanto, 2017:21).

Berdasarkan pendapat para ahli di atas, dapat disimpulkan bahwa *sublime text* merupakan sebuah *text editor* yang sering digunakan untuk para pengembang *web* dalam menulis *script*.

### 2.2.2. *Entity Relationship Diagram* (ERD)

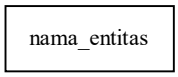
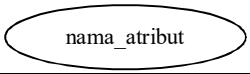
Menurut Sukamto & Shalahuddin (2015:53) menyatakan bahwa "ERD adalah bentuk paling awal dalam melakukan perancangan basis data relasional. Jika menggunakan OODMBS maka perancangan ERD tidak perlu dilakukan".

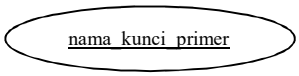
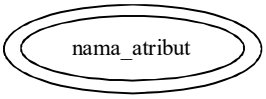
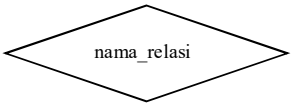
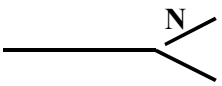
*Entity relationship diagram* (ERD) menjelaskan tentang beberapa entitas dan hubungan yang terjadi antar entitas tersebut yang berada pada suatu basis data (Pratama, Sihombing & Putra, 2014).

Maka dari itu, dapat disimpulkan bahwa *entity relationship diagram* (ERD) merupakan tahapan untuk memodelkan rancangan basis data berupa gambar atau simbol untuk mengambarkan hubungan basis data yang terjadi.

Komponen *entity relationship diagram* (ERD) diuraikan sebagai berikut (Sukamto & Shalahuddin, 2015:50).

**Tabel II.1.**
**Komponen *Entity Relationship Diagram* (ERD)**

| Notasi | Komponen | Keterangan |
|---|---|---|
|  | Entitas/*entity* | Entitas merupakan data inti yang akan disimpan, bakal tabel pada basis data, benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer. Penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel. |
|  | Atribut | *Field* atau kolom data yang butuh disimpan dalam suatu entitas. |

| | | |
|---|---|---|
|  | Atribut kunci primer | *Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan, biasanya berupa id. Kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama). |
|  | Atribut multinilai/*multivalue* | *Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki lebih dari satu. |
|  | Relasi | Relasi yang menghubungkan antar entitas, biasanya diawali dengan kata kerja. |
|  | Asosiasi/*association* | Penghubung antara relasi dan entitas dimana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan *one to many* menghubungkan entitas A dan entitas B. |

Sumber: (Sukamto & Shalahuddin, 2015:50)

### 2.2.3. *Logical Record Structure* (LRS)

LRS merupakan hasil transformasi diagram E-R (ERD) menggunakan aturan aturan tertentu. Aturan-aturan tersebut yaitu: (1) setiap *entity* akan diubah ke dalam bentuk sebuah kotak dengan nama *entity* berada di luar kotak dan atribut berada di dalam kotak, (2) sebuah relasi kadang disatukan dalam sebuah kotak bersama *entity*, kadang dipisah dalam sebuah kotak tersendiri (Ladjamudin, 2013:159).

*Logical record structure* (LRS) merupakan representasi dari struktur *record-record* pada tabel dimana tabel-tabel tersebut terbentuk dari hasil himpunan antar entitas pada *entity relationship diagram* (ERD) yang telah ditransformasikan menjadi bentuk LRS (Pratama, Sihombing & Putra, 2014).

Maka dari itu, dapat disimpulkan bahwa *logical record structure* (LRS) merupakan teknik penggambaran basis data yang mentransformasikan ERD ke LRS melalui proses kardinalitas.

Aturan pokok dalam melakukan transformasi E-R Diagram ke *logical record structure* sangat dipengaruhi oleh elemen yang menjadi titik perhatian utama pada langkah transformasi dengan proses kardinalitas, yang terdiri dari tiga (3) kardinalitas (Ladjamudin, 2013:159) yaitu:

1. *One to One*

   Yaitu proses kardinalitas yang panahnya lebih diarahkan di entity dengan jumlah atribut yang lebih sedikit.

2. *One to Many*

   Relasi harus digabungkan dengan entity pada pihak *many,* dan tidak perlu melihat banyak sedikitnya pada entity tersebut.

3. *Many to Many*

   Yaitu proses kardinalitas pada *relationship* berubah status menjadi *file* konektor, sehingga baik *entity* maupun relasi akan menjadi struktur *record* sendiri.

### 2.2.4. *Unified Modeling Language* (UML)

*Unified modeling language* (UML) merupakan metodologi yang digunakan untuk mengembangkan sistem berorientasi objek dan juga sebagai alat bantu dalam pengembangan sistem (Hendini, 2016).

Menurut Sukamto & Shalahuddin (2015:137) mengemukakan bahwa "UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung".

Berdasarkan pendapat para ahli di atas, dapat disimpulkan bahwa *unified modelling language* (UML) merupakan sebuah bahasa standar atau alat yang digunakan untuk memodelkan pembangunan perangkat lunak menggunakan simbol-simbol tertentu.

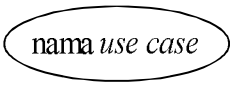
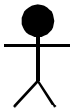
**A. *Use Case Diagram***

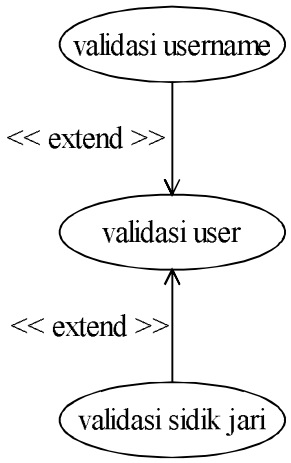
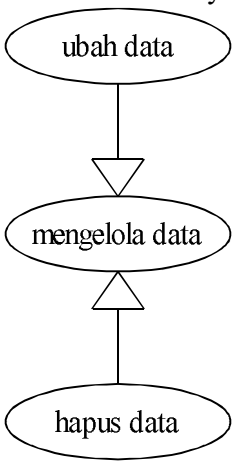
*Use case diagram* digunakan untuk memudahkan pengembang dalam memahami kebutuhan fungsional dari sistem yang akan dibangun (Maulana, 2016).

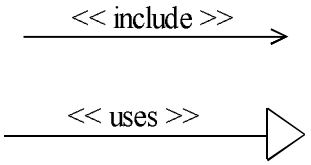
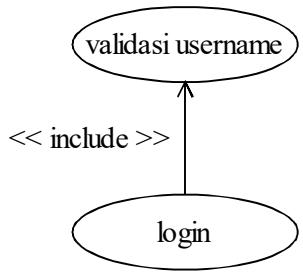
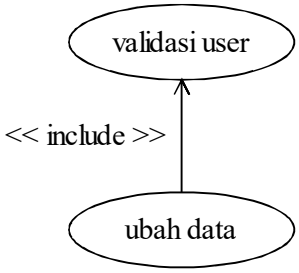
*Use case* diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem informasi dan digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi serta siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Hendini, 2016).

Berdasarkan pendapat para ahli di atas, dapat disimpulkan bahwa *use case diagram* merupakan diagram UML yang menggambarkan kelakuan objek (*behavior*) berdasarkan sudut pandang luar sistem.

**Tabel II.2.**
**Simbol *Use Case Diagram***

| Simbol | Deskripsi |
|---|---|
| *use case*<br>  | Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan kata kerja di awal frase nama *use case*. |
| Aktor/*actor*<br>  <br>Nama Aktor | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor. |
| Asosiasi/*association*<br>_____ | Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor. |

| | |
|---|---|
| Ektensi/*extend*<br><br><< extend >> → | Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walaupun tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, contoh:<br>  <br><br>Arah panah mengarah pada *use case* yang ditambahkan, biasanya *use case* yang menjadi *extend*-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya. |
| Generalisasi/*generalization*<br><br>→ | Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari yang lainnya, contoh:<br>  <br><br>Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum). |
| Menggunakan/*include*/*uses* | Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan |

| | |
|---|---|
|  | fungsinya atau sebagai syarat dijalankan *use case* ini.<br>Ada dua sudut pandang yang cukup besar mengenai *include* di *use case*:<br>• *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan, missal pada kasus berikut:<br>  <br>• *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah berjalan sebelum *use case* tambahan dijalankan, missal pada kasus berikut:<br>  <br>Kedua interpretasi di atas dapat dianut salah satu satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan. |

Sumber: (Sukamto & Shalahuddin, 2015:156)

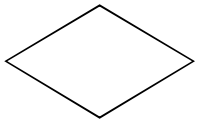
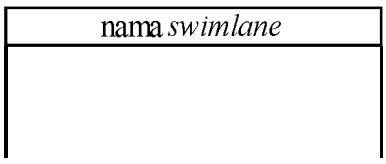
**B. *Activity Diagram***

*Activity diagram* merupakan diagram yang menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Hendini, 2016).

*Activity diagram* merupakan diagram yang menerangkan tentang aktifitas-aktifitas yang dapat dilakukan oleh seorang *actor* atau pengguna terhadap aplikasi yang dirancang (Meilinda, 2016).

Maka dari itu, dapat disimpulkan pada *activity diagram* merupakan sebagai diagram yang menggambarkan rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

**Tabel II.3.**
**Simbol *Activity Diagram***

| Simbol | Deskripsi |
|---|---|
| Status awal  | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal. |
| Aktivitas   | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja. |
| Percabangan/*decision*  | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |
| Penggabungan/*join*  | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu. |
| Status akhir  | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir. |
| *Swimlane*   | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. |

Sumber: (Sukamto & Shalahuddin, 2015:162).

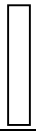
## C. *Sequence Diagram*

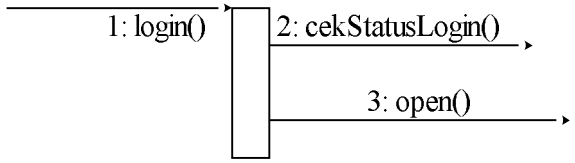
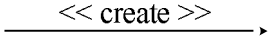
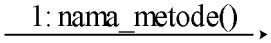
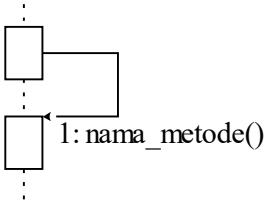
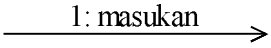
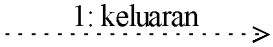
*Sequence diagram* berfungsi sebagai alat untuk menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek (Hendini, 2016).

*Sequence diagram* merupakan UML yang menggambarkan interaksi antar objek di dalam dan disekitar sistem, termasuk pengguna, *display*, dan sebagainya berupa *message* yang digambarkan terhadap waktu (Meilinda, 2016).

Berdasarkan pendapat beberapa ahli di atas, dapat disimpulkan bahwa *sequence diagram* merupakan diagram yang menggambarkan interaksi objek pada *use case* sesuai dengan urutan penggunaan yang mendeskripsikan waktu hidup objek berupa *message*.

**Tabel II.4.**
**Simbol *Sequence Diagram***

| Simbol | Deskripsi |
|---|---|
| Aktor<br>  <br>nama aktor<br>Atau<br>nama_aktor<br>Tanpa waktu aktif | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor. |
| Garis hidup/*lifeline*<br> | Menyatakan kehidupan suatu objek. |
| Objek<br>nama_objek: nama_kelas | Menyatakan objek yang berinteraksi pesan. |
| Waktu aktif<br>  | Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya: |

| | |
|---|---|
| |  <br>Maka cekStatusLogin() dan open() dilakukan di dalam metode login()<br>Aktor tidak memiliki waktu aktif. |
| Pesan tipe *create*<br>  | Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat. |
| Pesan tipe *call*<br>  | Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,<br>  <br>Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi. |
| Pesan tipe *send*<br>  | Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi. |
| Pesan tipe *return*<br>  | Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian. |
| Pesan tipe *destroy*<br>  | Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy* |

Sumber: (Sukamto & Shalahuddin, 2015:165)

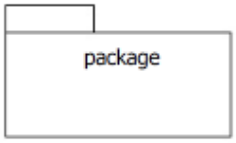
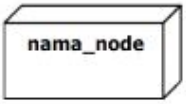
## D. *Deployment Diagram*

*Deployment diagram* digunakan untuk menggambarkan detail bagaimana komponen disusun di infrastruktur sistem (Hendini, 2016).

*Deployment diagram* menunjukan konfigurasi komponen dalam proses eksekusi aplikasi (Sukamto & Shalahuddin, 2015:154).

Dapat ditarik kesimpulan bahwa *deployment* diagram merupakan diagram UML yang berfungsi untuk menggambarkan konfigurasi komponen yang disusun sebagai infrastruktur aplikasi.

**Tabel II.5.**
**Simbol *Deployment Diagram***

| Simbol | Deskripsi |
|---|---|
| *Package*   | *Package* merupakan sebuah bungkusan dari satu atau lebih komponen. |
| *Node*   | Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak (*software*), jika di dalam node disertakan komponen untuk mengkonsistensikan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen. |
| Kebergantungan/*dependency*  | Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai. |
| *Link*  | Relasi antar *node*. |

Sumber: (Sukamto & Shalahuddin, 2015:154)

### 2.2.5. *Black Box Testing*

Menurut Sukamto & Shalahuddin (2015:275) *black box testing* "yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program".

*Black box testing* merupakan teknik pengujian yang melakukan pendekatan kebutuhan dasar program dalam mengecek fungsional program (Swastika & Putra, 2016:73).

Maka, *black box testing* adalah teknik pengujian perangkat lunak yang harus bebas dari kesalahan atau *error* yang menguji fungsional program.