



BAB II

TINJAUAN PUSTAKA

2.1 Teori Umum

2.1.1 Pengertian Komputer

Komputer adalah mesin multiguna yang dapat melakukan pemrograman, diprogramkan, menerima data, dan memproses data tersebut menjadi informasi yang dapat digunakan oleh user-nya (Sawyer dalam Krisbiantoro, 2018: 2). Pengertian lain dari komputer adalah suatu alat pemroses data yang melakukan perhitungan aritmatika dan operasi logika secara cepat tanpa campur tangan manusia (Fuori dalam Fariq dan Studio, 2010: 3).

Maka, Penulis menyimpulkan bahwa Komputer adalah alat bantu pemrosesan data yang cepat secara elektronik mulai dari *input*, proses, dan *output* yang bermanfaat untuk melaksanakan berbagai pekerjaan manusia.

2.1.2 Pengertian Perangkat Lunak

Krisbiantoro (2018: 7) mengatakan, “Perangkat Lunak atau piranti lunak adalah program komputer yang berfungsi sebagai sarana interaksi antara pengguna dan perangkat keras.” Perangkat Lunak juga dapat dikatakan sebagai perintah atau instruksi untuk mengolah data yang dijalankan oleh pengguna komputer (*user*) dengan bantuan perangkat keras (*hardware*) (Fariq dan Studio, 2010: 4).

Jadi, didapatkan kesimpulan bahwa Perangkat Lunak adalah program komputer yang berisi perintah dan instruksi untuk mengolah data dengan bantuan perangkat keras yang dapat memudahkan pekerjaan manusia.

2.1.3 Pengertian Perangkat Keras

Perangkat Keras adalah alat pengolahan data untuk mendapat informasi pada komputer yang secara fisik dapat dilihat bentuknya (Fariq dan Studio, 2010:3), Sedangkan Suyanto (2005: 47) mengatakan, “Perangkat Keras adalah alat pengolahan data yang bekerja secara elektronik dan otomatis serta dapat



bekerja apabila ada unsur manusia yang mengerti dan menggunakan alat tersebut.”

Jadi, Perangkat Keras adalah alat pengolahan data yang dapat dilihat bentuknya dan bekerja secara elektronik dengan adanya unsur manusia yang mengerti dan dapat menggunakan alat tersebut agar dapat digunakan dalam memperoleh informasi.

2.1.4 Pengertian Basis Data

Junindar (2008: 19) mengatakan, “Database (basis data) adalah kumpulan data yang saling berhubungan satu dengan lainnya yang tersimpan di perangkat keras komputer dan diperlukan suatu perangkat lunak untuk memanipulasi basis data tersebut. Sedangkan Basis Data lebih sederhana dapat diartikan sebagai koleksi data yang saling berhubungan, disimpan dan didesain untuk menjumpai informasi sesuai dengan kebutuhan organisasi (Pallaw dalam Subandi dan Syahidi, 2018: 2).

2.1.5 Metode Pengembangan Perangkat Lunak

Penelitian ini menggunakan metode pengembangan perangkat lunak dengan RUP (*Rational Unified Process*). Menurut Sukamto dan Shalahuddin (2018:125), “RUP (*Rational Unified Process*) adalah pendekatan pengembangan perangkat lunak yang dilakukan berulang-ulang (*iterative*), fokus pada arsitektur (*architecture-centric*), lebih diarahkan berdasarkan penggunaan kasus (*use case driven*)”. Adapun tahap-tahap (*fase*) dalam metode pengembangan RUP menurut Sukamto dan Shalahuddin (2018:128-131) adalah sebagai berikut:

1. *Inception* (permulaan)

Tahap ini lebih pada memodelkan proses bisnis yang dibutuhkan (*bussiness modeling*) dan mendefinisikan kebutuhan akan sistem yang akan dibuat (*requirements*) dari hasil wawancara dan observasi yang penulis lakukan.



2. *Elaboration* (perluasan/perencanaan)

Tahap ini lebih difokuskan pada perencanaan arsitektur sistem. Tahap ini juga dapat mendeteksi apakah arsitektur sistem yang diinginkan dapat dibuat atau tidak. Mendeteksi resiko yang mungkin terjadi dari arsitektur yang dibuat. Tahap ini lebih pada analisis dan desain sistem serta implementasi sistem yang fokus pada purwarupa sistem (*prototype*).

3. *Construction* (kontruksi)

Tahap ini fokus pada pengembangan komponen dan fitur-fitur sistem. Tahap ini lebih pada implementasi dan pengujian sistem yang fokus pada implementasi perangkat lunak pada kode program. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/tonggak kemampuan operasional awal. Pada tahap ini juga menjelaskan bagaimana mengimplementasi hasil desain dan melakukan uji coba terhadap aplikasi yang telah dibuat. Dalam tahapan implementasi dijelaskan perangkat keras dan perangkat lunak apa saja yang dibutuhkan untuk mengimplementasi aplikasi ini. Sedangkan pada tahapan uji coba dilakukan testing. Testing diperlukan untuk menjamin kualitas aplikasi yang telah dibuat apakah telah sesuai dengan yang diharapkan.

4. *Transition* (transisi)

Tahap ini lebih pada deployment atau instalasi sistem agar dapat dimengerti oleh user. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/tonggak kemampuan operasional awal. Aktifitas pada tahap ini termasuk pada pelatihan *user*, pemeliharaan dan pengujian sistem apakah sudah memenuhi harapan *user*..

2.2 Teori Judul

2.2.1 Pengertian Aplikasi

Menurut Pane, dkk (2020: 53) mengatakan, “aplikasi adalah perangkat lunak (*software*) atau program komputer yang beroperasi pada sistem tertentu



yang diciptakan atau dikembangkan untuk melakukan perintah tertentu. Lebih sederhana aplikasi bisa diartikan sebagai suatu *software* atau program sistem yang diolah untuk menjadi informasi (Widianti dalam Pane dan dkk, 2020: 53).

Dari definisi diatas penulis menyimpulkan bahwa sistem adalah sekumpulan perangkat lunak yang saling berhubungan satu sama lain dan beroperasi untuk mencapai suatu tujuan.

2.2.2 Pengertian Konsultasi

Konsultasi adalah adalah layanan konseling oleh pihak yang memberikan bantuan keahlian (konsultan) kepada pihak yang membutuhkan pemecahan masalah (konsulti) dengan tujuan memberikan wawasan, pemahaman, dan cara-cara yang perlu dilaksanakan konsulti dalam rangka membantu terselesaikannya masalah yang dialami (Narti, 2019: 139). Sedangkan menurut Yap (2017: 35) mengatakan, “konsultasi adalah pertukaran pikiran untuk mendapatkan kesimpulan yang sebaik-baiknya atau meminta pertimbangan dalam memutuskan sesuatu.”

Sehingga dapat disimpulkan bahwa Konsultasi adalah layanan konseling oleh konsultan yang bertujuan untuk memberikan wawasan dan pemahaman kepada konsulti sebagai pertimbangan dalam memutuskan suatu permasalahan.

2.2.3 Pengertian Bangun Rumah

2.2.3.1 Pengertian Bangun

Bangun adalah sebuah kata hononim karena arti-artinya memiliki ejaan dan pelafalan yang sama tetapi maknanya berbeda. Arti kata Bangun dalam kelas verba atau kata kerja dapat menyatakan suatu tindakan, keberadaan, pengalaman, atau pengertian dinamis lainnya. Berdasarkan Kamus Besar Bahasa Indonesia Kementrian Pendidikan dan Kebudayaan (KBBI Kemendikbud, 2020) menyatakan, “Bangun adalah cara menyusun atau susunan yang merupakan suatu wujud atau struktur.”



2.2.3.2 Pengertian Rumah

Menurut Urip (2014: 371) mengatakan, “rumah adalah bangunan gedung yang berfungsi sebagai tempat tinggal yang layak huni, sarana pembinaan keluarga, cerminan harkat dan martabat penghuninya, serta aset bagi pemiliknya.” Sedangkan berdasarkan Undang-Undang No. 4 Tahun 1992 tentang Perumahan dan Pemukiman pada Pasal 1 ayat 1 menyatakan, “rumah adalah bangunan yang berfungsi sebagai tempat tinggal atau hunian dan sarana pembinaan keluarga Berdasarkan dari pengertian tersebut rumah adalah bangunan yang berfungsi sebagai tempat tinggal untuk meneruskan kelangsungan hidup.

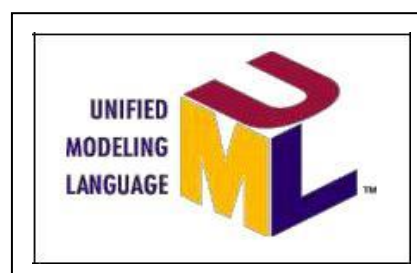
2.2.4 Pengertian Aplikasi Konsultasi Bangun Rumah pada Unit Kerja Digital Marketing PT Semen Baturaja Persero (Tbk)

Aplikasi Konsultasi Bangun Rumah pada Unit Kerja PT Semen Baturaja Persero (Tbk) adalah Aplikasi yang dibuat untuk membantu masyarakat dengan cara memberikan wawasan, pemahaman, dan cara-cara yang perlu dilaksanakan dalam menyelesaikan permasalahan tentang konstruksi bangunan rumah serta menjadi wadah dan sarana aktivitas promosi produk oleh perusahaan.

2.3 Teori Khusus

2.3.1 Pengertian *Unified Modeling Language* (UML)

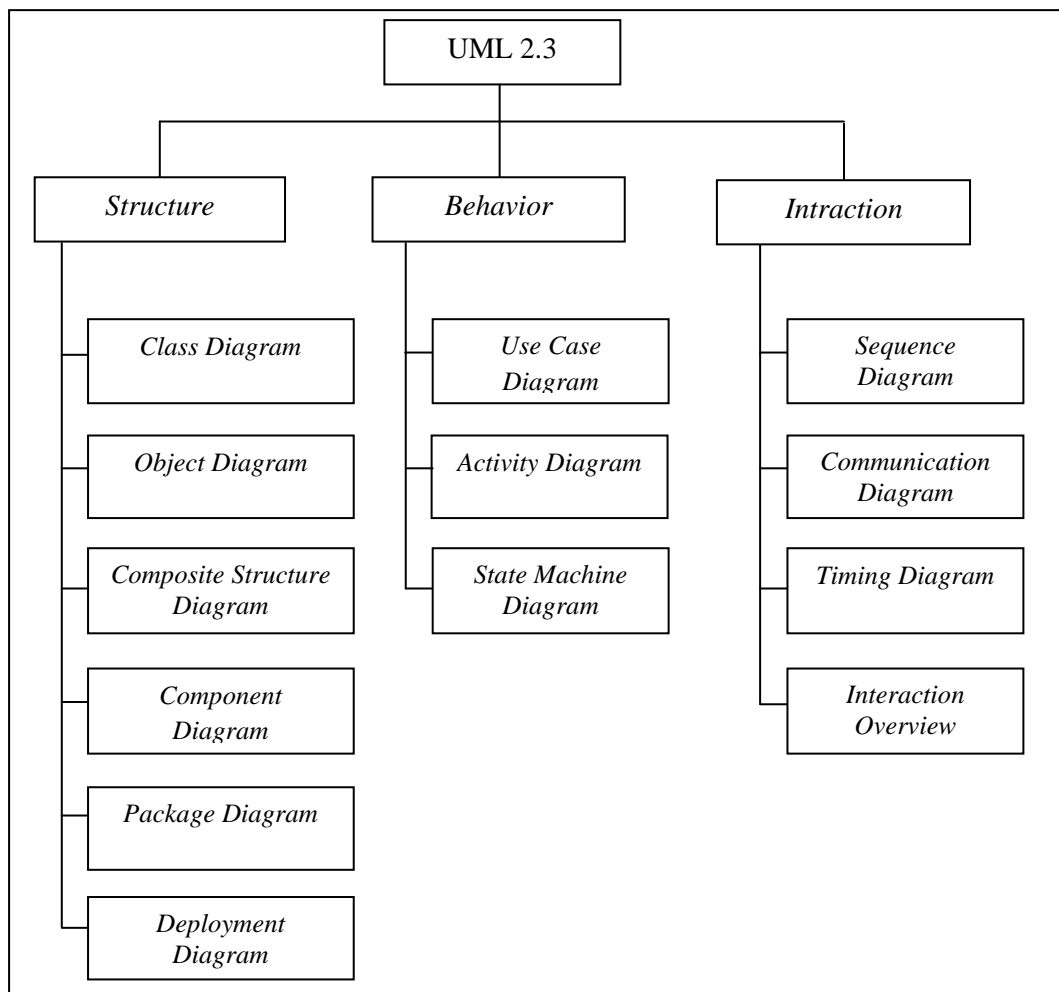
Menurut Sukamto dan Shalahuddin (2013: 133) menyatakan, “*Unified Modeling Language* (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.”



Gambar 2.1 Tampilan Logo UML



Menurut Sukamto dan Shalahuddin (2013:140), “Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori.” Pembagian kategori dan macam-macam diagram Menurut Sukamto dan Shalahuddin tersebut dapat dilihat pada gambar dibawah:



Gambar 2.2 Kategori dan Macam-macam Diagram

Penjelasan singkat dari pembagian kategori pada diagram UML menurut Sukamto dan Shalahuddin (2013: 141) :

- 1) *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.



- 2) *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
- 3) *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

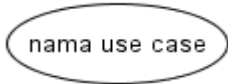
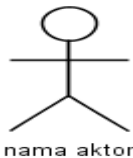
2.3.2 Jenis-Jenis Diagram UML

2.3.2.1 Pengertian *Use case Diagram*


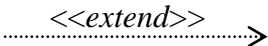
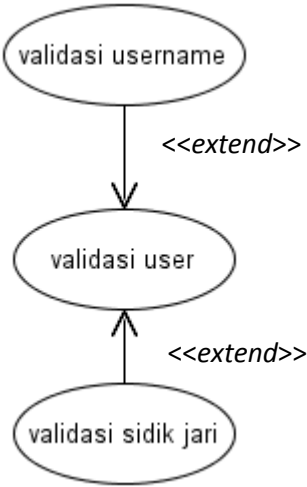
Sukanto dan Shalahuddin (2013: 155) menjelaskan, “*Use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem.”

Adapun simbol-simbol yang digunakan dalam *Use case* adalah sebagai berikut:

Tabel 2.1 Simbol-simbol *Use case Diagram*

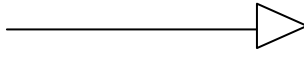
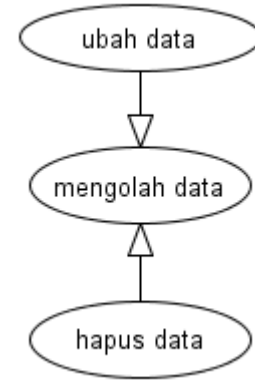
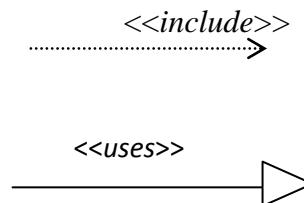
No	Simbol	Deskripsi
1.		<p>fungsi yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal-awal frase nama <i>use case</i></p>
2.		<p>orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang 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</p>

Lanjutan **Tabel 2.1** Simbol-simbol Use case Diagram

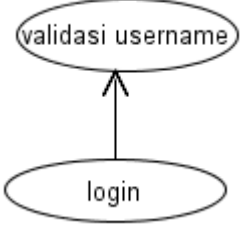
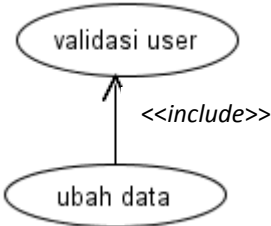
		nama actor
3.	asosiasi / <i>association</i> 	komunikasi antar aktor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan actor
4.	ekstensi / <i>extend</i> 	<p>relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang di tambahkan dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu; mirip dengan prinsip <i>inheritance</i> pada pemrograman berorientasi objek; biasanya <i>use case</i> tambahan memiliki nama depan yang sama dengan <i>use case</i> yang ditambahkan, misalnya</p>  <p>arah panah mengarah pada <i>use case</i> yang ditambahkan; biasanya <i>use case</i> yang menjadi <i>extend</i>-nya merupakan jenis</p>



Lanjutan Tabel 2.1 Simbol-simbol Use case Diagram

		yang sama dengan <i>use case</i> yang menjadi induknya
5.	<p>Generalisasi / <i>generalization</i></p> 	<p>hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya,</p>  <p>misalnya: arah panah mengarah pada <i>use case</i> yang menjadi generalisasinya (umum)</p>
6.	<p>menggunakan / include / uses</p> 	<p>relasi tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat dijalankan <i>use case</i> ini</p> <p>ada dua sudut pandang yang cukup besar mengenai include di <i>use case</i>:</p> <ul style="list-style-type: none"> • <i>Include</i> berarti <i>use case</i> yang ditambahkan akan selalu di panggil saat <i>use case</i> tambahan dijalankan, misalnya pada kasus berikut:

Lanjutan **Tabel 2.1** Simbol-simbol Use case Diagram

		 <pre> graph BT login([login]) --> validasi_username([validasi username]) </pre> <p>• <i>Include</i> berarti <i>use case</i> yang tambahan akan selalu melakukan pengecekan apakah <i>use case</i> yang di tambahkan telah dijalankan sebelum <i>use case</i> tambahan dijalankan, misal pada kasus berikut:</p>  <pre> graph BT ubah_data([ubah data]) -- <<include>> --> validasi_user([validasi user]) </pre> <p>kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.</p>
--	--	---

Sumber: Sukamto dan Shalahuddin (2013:156)

Dua hal utama pada *use case* adalah sebagai berikut:

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, Jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

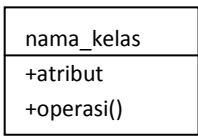
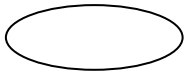

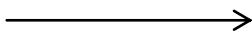
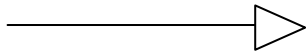
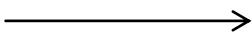


2.3.2.2 Pengertian *Class Diagram*

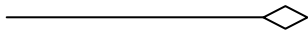
Sukamto dan Shalahuddin (2013: 141) menjelaskan, “*Class Diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.”

Adapun simbol-simbol yang digunakan dalam *class diagram* adalah sebagai berikut:

Tabel 2.2 Simbol-simbol *Class Diagram*

No.	Simbol	Deskripsi
1.	kelas 	Kelas pada struktur sistem
2.	antarmuka / interface  nama_interface	Sama dengan konsep interface dalam pemrograman berorientasi objek
3.	asosiasi / association 	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai <i>multiplicity</i>
4.	asosiasi berarah / <i>directed association</i> 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i>
5.	generalisasi 	Relasi antarkelas dengan makna generalisasi – spesialisasi (umum khusus)
6.	kebergantungan / <i>dependency</i> 	Relasi antarkelas dengan makna kebergantungan antar kelas

Lanjutan **Tabel 2.2** Simbol-simbol *Class Diagram*


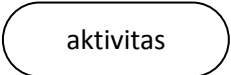



7.	agregasi / <i>aggregation</i> 	Relasi antarkelas dengan makna semua-bagian (<i>whole-part</i>)
----	--	---

Sumber: Sukamto dan Shalahuddin (2013:146)

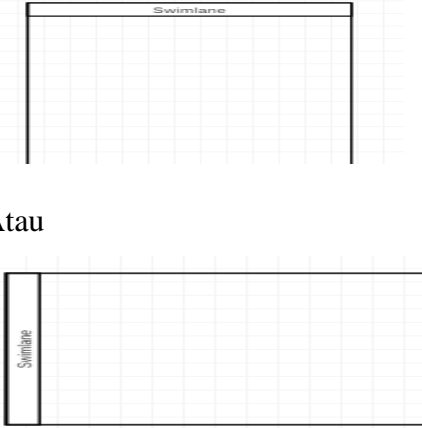
2.3.2.3 Pengertian *Activity Diagram*

Activity Diagram adalah tipe khusus dan diagram status yang menggambarkan suatu aktivitas ke aktivitas lainnya dalam suatu sistem (Muslihudin dan Oktafianto, 2016: 63). Adapun menurut Sukamto dan Shalahuddin (2013: 161) simbol-simbol yang digunakan dalam *activity diagram* adalah sebagai berikut:

Tabel 2.3 Simbol-simbol *Activity Diagram*

No	Simbol	Deskripsi
1.	Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
3.	Percabangan / <i>decision</i> 	Asosiasi percabangan di mana jika ada pilihan aktivitas lebih dari satu
4.	Penggabungan / <i>join</i> 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
5.	Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir

Lanjutan **Tabel 2.3** Simbol-simbol *Activity Diagram*

6.	<p>Swimlane</p>  <p>Atau</p>	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi
----	---	---

Sumber: Sukamto dan Shalahuddin (2013:164)

2.3.2.4 Pengertian *Sequence Diagram*

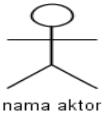
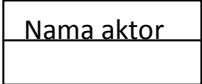
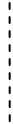
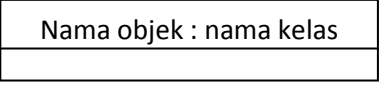

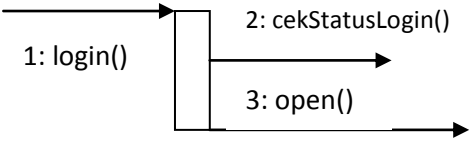
Menurut Sukamto dan Shalahuddin (2013: 165), “*Sequence Diagram* menggambarkan kelakuan objek pada *Use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah beserta metode-metode yang dimiliki kelas yang diinstansikan menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *Use case*.”

Banyaknya *Sequence Diagram* yang harus digambarkan adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

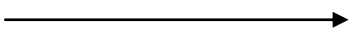
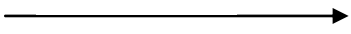
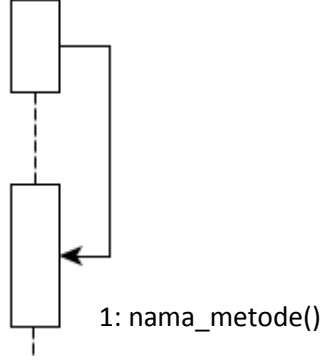
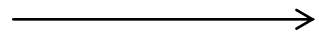
Dapat penulis simpulkan bahwa *Sequence diagram* adalah penggambaran skenario dari sebuah objek yang ada pada *use case* yang meliputi rangkaian langkah-langkah aktivitas dari objek berdasarkan waktu hidup objek dan pesan-pesan yang diterima maupun yang dikirimkan objek kepada objek lainnya.

Berikut simbol-simbol pada *Sequence Diagram* :

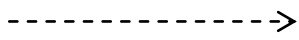
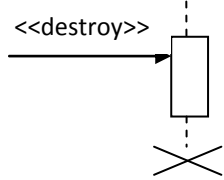
Tabel 2.4 Simbol-simbol pada *Sequence Diagram*

No	Simbol	Deskripsi
1.	<p>Actor</p>  <p>atau</p>  <p>tanpa waktu aktif</p>	<p>orang, proses, atau sistem lain yang berinteraksi dengan 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 <i>actor</i>.</p>
2.	<p>Garis hidup / <i>lifeline</i></p> 	<p>menyatakan kehidupan suatu objek</p>
3.	<p>Objek</p> 	<p>menyatakan objek yang berinteraksi pesan</p>
4.	<p>Waktu aktif</p> 	<p>menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya</p> 

Lanjutan **Tabel 2.4** Simbol-simbol pada *Sequence Diagram*

		maka cekStatusLogin () dan open() dilakukan di dalam metode login() aktor tidak memiliki waktu aktif
5.	Pesan tipe <i>create</i> <<create>> 	menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
6.	Pesan tipe <i>call</i> <<create>> 	menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  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
7.	Pesan tipe <i>send</i> 1: masukan 	menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim

Lanjutan **Tabel 2.4** Simbol-simbol pada *Sequence Diagram*

8	Pesan tipe <i>return</i> 1: keluaran 	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
9.	Pesan tipe <i>destroy</i> 	menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada <i>create</i> maka ada <i>destroy</i>

Sumber: Sukamto dan Shalahuddin (2013:165-167)

2.4 Teori Program

2.4.1 Pengertian XAMPP

XAMPP adalah sebuah *software* yang berfungsi sebagai *server* untuk menjalankan website berbasis *PHP* dan menggunakan pengolahan data *MySQL* (Wicaksono, 2008: 7). Sedangkan Dadan dan Developers (2015: 28) menyatakan, “*XAMPP* adalah satu aplikasi web *server apache* yang terintegrasi dengan *mysql* dan *phpMyAdmin*”.



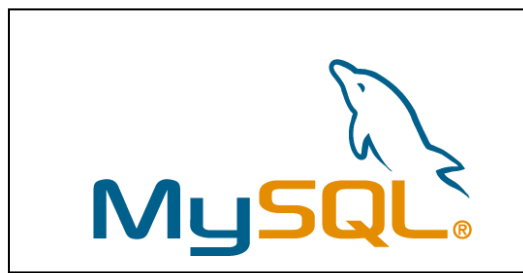
Gambar 2.3 Logo *XAMPP*

Jadi, *XAMPP* adalah sebuah aplikasi perangkat lunak pemrograman dan *database* yang di dalamnya terdapat berbagai macam aplikasi pemrograman yang terdiri dari *Apache*, *MySQL*, *PhpMyAdmin*, *Perl*, *Filezilla* dan lain-lain.

2.4.2 Pengertian *MySQL*



MySQL adalah database yang digunakan untuk mengelola data SQL (*database management system*) atau DBMS yang *multithread* atau *multi-user*. Pengertian tersebut berdasarkan dari pendapat Kadir (2008: 2) yang menyatakan, “MySQL adalah *software* yang tergolong *Database Management System* (DBMS) yang bersifat *open source* dilengkapi dengan kode yang dipakai untuk membuat MySQL (*source code*), dapat dijalankan secara langsung dalam sistem operasi, dan bisa diperoleh dengan cara mengunduh (*download*) di internet secara gratis.” MySQL juga dapat diartikan sebagai database jenis *Relational Database Management System* (RDBMS) yang cepat dan mudah digunakan (Enterprise: 2017: 2).



Gambar 2.4 Tampilan Logo MySQL

2.4.3 Pengertian PHP

Hypertext Preprocessor (PHP) adalah bahasa pemrograman berbasis *server-side* yang bisa kita gunakan untuk membuat aplikasi web yang disisipkan pada HTML. Pengertian tersebut berdasarkan pendapat dari Anhar (2010: 3) yang menyatakan, “*Hypertext Preprocessor* (PHP) adalah bahasa pemrograman web *server-side* yang bersifat *open source* berupa *script* yang diintegrasikan dengan HTML untuk membuat halaman website yang dinamis.”



Gambar 2.5 Tampilan Logo PHP



2.4.3.1 Sintaks Dasar PHP

Menurut Anhar (2010: 23), Ada empat macam cara penulisan kode PHP sebagai berikut:

1. `<? echo {"ini adalah script PHP\n"}; ?>`
2. `<? php echo {"ini adalah script PHP\n"}; ?>`
3. `<script language= " php">`
`echo {"Latihan menulis script PHP"};`
`<\script>`
4. `<% echo {"Belajar PHP"}: %>`

2.4.3.2 Tipe Data PHP

Tipe data merupakan jenis dari suatu data yang akan diproses oleh bahasa pemrograman. Beberapa tipe data dalam PHP (Anhar, 2010: 25) adalah sebagai berikut :

1. **Integer** merupakan tipe data bilangan bulat. Contoh:
`42 // desimal`
`-678900 // negatif`
`0755 // oktal`
`0XC4E // heksadesimal`
2. **Floating Point Number** disebut juga bilangan pecahan. Terdapat tanda titik yang merupakan pemisah antara bilangan bulat dan pecahan. Contoh:
`4.5678 // bentuk biasa`
`8.7e4 // bentuk eksponensial`
3. **Boolean** adalah tipe data yang paling sederhana, hanya berupa **TRUE** dan **FALSE**.
4. **String** adalah tipe data yang terdiri dari huruf atau kata, bias berupa kata tunggal maupun kalimat. Penulisan *string* harus diapit dengan tanda petik, baik berupa petik tunggal (‘...’) maupun petik ganda (“...”). Contoh: “Sekarang Belajar PHP”.
5. **Objek** adalah tipe data dibuat dengan tujuan agar para *programmer*



terbiasa dengan OOP. Tipe data ini bias berupa bilangan.

6. **Array** merupakan **Tipe Compound Primitif**, terdapat pada bahasa pemrograman lain.
7. **Null** adalah tipe data yang tidak memuat apapun. Setiap variabel yang diset menjadi tipe data Null, ini akan menjadikan variabel tersebut kosong.
8. **Resources** tipe data spesial yang satu ini dikhususkan untuk menyimpan *resources*, sumber atau alamat.

2.4.4 Pengertian *phpMyAdmin*

Zaki dan Community (2006: 97) menyatakan, “*phpMyAdmin* adalah *MySQL client* yang berupa aplikasi web dan umumnya tersedia di *server* PHP seperti *xampp* maupun *server* komersial lainnya”. Pengertian lain dari *PhpMyAdmin* adalah administrator *MySQL* yang berfungsi mengolah database (Rahman, 2013: 12). Jadi, dari pengertian tersebut dapat disimpulkan bahwa *PhpMyAdmin* adalah aplikasi PHP sebagai administrator *MySQL* yang digunakan untuk membuat *database*, mengelola tabel, mengelola data, relasi antar tabel, dan mengirim database secara praktis tanpa harus menggunakan perintah (*command*) *SQL*.”



Gambar 2.6 Tampilan logo *phpMyAdmin*

2.4.5 Pengertian HTML

HTML singkatan dari *Hypertext Markup Language* adalah semacam bahasa pemrograman (*script*) yang mengatur tampilan-tampilan (*visual*) layout pada sebuah website (Enterprise, 2016: 16). Sedangkan menurut Enterprise (2014: 1) mengatakan, “HTML adalah *script* pemrograman yang mengatur bagaimana kita



menyajikan informasi di dunia internet dan bagaimana informasi itu membawa kita melompat dari satu tempat ke tempat lainnya. Sehingga dapat disimpulkan bahwa HTML adalah sebuah bahasa pemrograman yang mengatur *visual* layout suatu website sehingga dapat menyajikan informasi bagi pengguna.



Gambar 2.7 Tampilan Logo HTML