BAB 2
LANDASAN TEORI

2.1 Profil Perusahaan

Landasan teori adalah penjelasan definisi, konsep, proposisi yang telah disusun rapi, dan si stematis tentang *variable-variable* dalam sebuah penelitian yang berkaitan dalam pembangunan aplikasi pemesanan travel dan rental mobil serta monitoring mobil berbasis android (studi kasus duta trans). Landasan teori ini akan menjadi dasar yang kuat dalam penelitian yang akan dilakukan. Serta teori – teori terkait yang mendukung penelitian akan dibahas pada bab ini.

2.1.1 DUTA TRANS KUNINGAN

Duta Trans Kuningan adalah perusahaan yang bergerak dibidang jasa transportasi biro perjalanan(*travel*) dan rental mobil. Duta Trans Kuningan didirikan pada tahun 2006 dengan nama awal Dwi Utami oleh Andri Rian Hidayat sebagai Direktur dan Yani Suryani sebagai Komisaris.

Duta Trans Kuningan didaftarkan sebagai badan hukum Perseroan Terbatas(PT) pada tahun 2015. Berdasarkan keputusan MENKUMHAM nomor AHU-2434707.AH.01.01.TAHUN 2015, PT DUTA TRANS KUNINGAN sah berdiri sebagai badan hukum Perseroan Terbatas.

Duta trans melayani perjalanan dari Kuningan-Yogyakarta maupun sebaliknya pada pukul 8 pagi dan 8 malam dengan biaya Rp. 150.000,00 dan Kuningan-Majalengka dengan biaya Rp. 175.000,00. Sedangkan untuk rental mobil

disewakan dengan harga Rp. 350.000,00 untuk 1 harinya, dan tambahan biaya Rp. 100.000,00 jika ingin menyewa dengan supirnya.

2.1.2 Visi dan Misi

Visi dan misi dari Duta Trans Kuningan adalah sebagai berikut:

Visi :

Mandiri, Berdaya, Agamis, Kekeluargaan dan saling menguntungkan.

Misi :

Menyediakan jasa dan kualitas produk terbaik untuk biro perjalanan transportasi angkutan darat yang aman, efisien dan menguntungkan, yang dijalankan oleh profesional yang kompeten dan bermotivasi tinggi.

2.2 Landasan Teori

2.2.1 Pengertian Aplikasi

aplikasi adalah software atau perangkat lunak yang dibuat untuk mengerjakan menyelesaikan masalah-masalah khusus.

Sedangkan menurut Jogiyanto, aplikasi merupakan program yang berisikan perintah-perintah untuk melakukan pengolahan data. Jadi aplikasi secara umum adalah suatu proses dari cara manual yang ditransformasikan ke komputer dengan membuat sistem atau program agar data diolah lebih berdaya guna secara optimal.

Dari uraian diatas dapat disimpulkan bahwa aplikasi adalah sebuah perangkat lunak yang berisi perintah untuk menyelesaikan masalah dan pengolahan data.

2.2.2 Pengertian Pemesanan

Pemesanan adalah “proses, perbuatan, cara memesan (tempat,barang,dsb) kepada orang lain.”

2.2.3 Pengertian Travel

Pengertian travel (biro perjalanan) menurut beberapa ahli adalah sebagai berikut:

a) Biro perjalanan (*Travel*) adalah sebuah perusahaan yang menjual rancangan perjalanan secara langsung pada masyarakat dan lebih khusus lagi menjual transportasi udara, darat, laut; akomodasi penginapan; pelayaran wisata; wisata paket; asuransi perjalanan; dan produk lainnya yang berhubungan.

b) Biro perjalanan (*Travel*) adalah suatu perusahaan yang memperoleh pendapatan dan keuntungan dengan menawarkan dan menjual produk serta jasa-jasa pelayanan yang diberikannya kepada pelanggannya. Selain itu, menurut Yoeti (2003:59) munculnya biro perjalanan memiliki beberapa peran, yaitu:

a) Pengurusan dokumen perjalanan

b) Ticketing (penjualan tiket pesawat domestik dan internasional)

c) Hotel Reservation (dalam dan luar negeri)

d) Agen perjalanan kapal pesiar, charter flight, kapal laut dan kereta api

e) Paket wisata untuk dalam dan luar negeri

f) Escort services (jasa mengiringi)

g) Jemput dan antar tamu dari dan ke bandara

h) Pelayanan Umroh, Ibadah Haji dan perjalanan rohani lainnya.

2.2.4 Pengertian Rental

sewa adalah bagian pembayaran ke atas sesuatu faktor produksi yang melebihi dari pendapatan yang diterimanya dari pilihan pekerjaan lain yang terbaik yang mungkin dilakukannya

sewa sebagai sejumlah uang/ barang yang dibayarkan kepada pemilik tanah oleh pihak yang menggunakan tanah sebagai balas jasa untuk penggunaan tanah tersebut.

Berdasarkan pendapat ahli tersebut diatas, maka dapat ditarik kesimpulan bahwa sewa adalah harga yang dibayar atas penggunaan suatu barang dan faktor-faktor produksi lainnya yang jumlah penawarannya tidak dapat ditambah.

2.2.5 Pengertian *Monitoring*

Monitoring adalah proses menjaga atau pengawasan terhadap keberadaan dan besarnya perubahan keadaan dan arus data dalam sebuah sistem. *Monitoring* bertujuan untuk mengidentifikasi kesalahan dan kemajuan dalam menentukan keputusan selanjutnya.

Teknik yang digunakan dalam monitoring informasi sistem memotong bidang pengolahan *real-time*, statistik, dan analisis data. Satu set Komponen perangkat lunak yang digunakan untuk pengumpulan data, pengolahan, dan presentasi disebut sistem *monitoring*.

2.2.6 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis *linux* yang mencakup sistem operasi. *Middleware* dan aplikasi. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan android, dibentuklah *Open Handset Alliance, konsorsium* dari 34 perusahaan peranti keras, peranti lunak , telekomunikasi , termasuk Google, HTC, Intel, Motorola , Qualcomm, T-Mobile, dan Nvidia. [3]

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat mobile. Di lain pihak, Google merilis kode-kode Android di bawah lisensi *Apache*, sebuah lisensi perangkat lunak dan open *platform* perangkat seluler.[4]

Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Services* (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD).

Pada masa saat ini kebanyakan vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis android, vendor-vendor itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus,

SciPhone, WayteQ, Sony Ericsson, LG, Acer, Philips, T-Mobile, Nexian, IMO, Asus dan masih banyak lagi vendor *smartphone* didunia yang memproduksi android. Hal ini karena android itu adalah sistem operasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun.

2.2.6.1 Android SDK (*Software Development Kit*)

Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemograman *Java*. Android merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi., *middleware* dan aplikasi kunci yang di rilis oleh Google. Saat ini disediakan Android SDK (*Software Deveplmonet Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemograman *Java*. Sebagai platform aplikasi-netral, Android memberi anda kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *Handphone/Smartphone*. Beberapa fitur-fitur Android yang paling penting adalah :

a. *Framework* aplikasi yang mendukung penggantian komponen dan reusable.
b. Mesin Virtual Dalvik diptimalkan untuk perangkat mobile.
c. Integrated browser berdasarkan engine open source *WebKit*.
d. Grafis yang dioptimalkan dan didukung oleh libraries grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1,0 (Opsional akselerasi hardware).
e. SQLite untuk penyimpanan data.
f. Media Support yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM Telephony (tergantung hardware).
g. Bluetooth, EDGE, 3G, dan WiFi (tergantung hardware).
h. Kamera, GPS, kompas, dan *accelerometer* (tergantung hardware).
i. Lingkungan *Development* yang lengkap dan kaya termasuk perangkat emulator, tools untuk debugging, profil dan kinerja memori, dan plugin untuk IDE Eclipse.

2.2.6.2 ADT (*Android Development Tools*)

Android Development Tools (ADT) adalah plugin yang didesain untuk IDE *Eclipse* yang memberikan kita kemudahan dalam mengembangkan aplikasi android dengan menggunakan IDE *Eclipse*. Dengan menggunakan ADT untuk *Eclipse* akan memudahkan kita dalam membuat aplikasi *project* android, membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya, begitu juga kita dapat melakukan running aplikasi menggunakan Android SDK melalui *Eclipse*. Dengan ADT juga kita dapat melakukan pembuatan package android (*.apk*) yang digunakan untuk distribusi aplikasi android yang kita rancang.

2.2.6.3 Arsitektur Android

Secara garis besar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut :

1. *Applications dan Widgets*

Applications dan Widgets ini adalah layer di mana kita berhubungan dengan aplikasi saja, di mana biasanya kita download aplikasi kemudian kkita lakukan instlasi dan jalankan aplikasi tersebut. Di layer terdapat aplikasi inti termsauk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemograman Java.

2. *Applications Frameworks*

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat akses, akses informasi, *resources*, menjalankan *service background*, mengatur alarm, dan menambahkan status notifications, dan sebagainya. Pengembang memiliki akses penuh menuju API framework seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (reuse).

Sehingga bisa kita simpulkan *Applications Frameworks* ini adalah layer di mana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang

akan dijalankan di sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti *content-providers* yang berupa sms dan panggilan telepon.

3. *Libraries*

Libraries ini adalah *layer* di mana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Berjalan di atas kernel, layer ini meliputi berbagai library C/C++ inti seperti *Libc* dan *SSL*, serta :

a) *Libraries* media untuk pemutaran media audio dan video

b) *Libraries* untuk manajemen tampilan.

c) *Libraries* Graphics mencakup *SGL* dan *OpenGL* untuk grafis 2D dan 3D.

d) *Libraries* SQLite untuk dukungan *database*.

e) *Libraries* SSL dan *WebKit* terintegrasi dengan *web browser* dan *sequrity*.

f) *Libraries LiveWebcore* mencakup *modern web browser* dengan *engine embeded web view*.

g) *Libraries* 3D yang mencakup implementasi *OpenGL* ES 1.0 API's.

4. *Android Run Time*

Layer yang membuat aplikasi *Android* dapat dijalankan di mana dalam prosesnya menggunakan Implementasi *Linux*. *Dalvix Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam *Android Run Time* dibagi menjadi dua bagian yaitu :

a) *Core Libraries* : Aplikasi Android dibangun dalam bahasa *Java*, sementara *Dalvik* serbagai virtual mesinnya bukan *Virtual Machine Java*, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa Java/C yang ditangani oleh *Core Libraries*.

b) *Dalvik Virtual Machine* : Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi serta efisien , dimana merupakan pengembangan yang mampu membuat *linux kernel* untuk melakukan *threading* dan manajemen tingkat rendah.

5. *Linux Kernel*

Linux Kernel adalah layer dimana inti dari operating sistem dari *Android* itu berada. Berisi file-file system yang mengatur sistem procressing, memory, *resource, drivers,* dan sistem-sistem operasi android adalah linux kernel release 2.6.

2.2.6.4 Fundamental Aplikasi

Aplikasi Android ditulis dalam bahasa pemograman Java. Kode Java dikompilasi bersama dengan data *file resource* yang dibuthkan oleh aplikasi, di mana prosesnya dipackage oleh *tools* yang dinamakan “apt tools” ke dalam paket Android sehingga menghasilkan file dengan ekstensi apk . Ada enam jenis komponen pada aplikasi Android yaitu :

a. *Activities*

Suatu *activity* akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi android bisa jadi hanya memili satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* tergantung pada tujuan aplikasi dan desain dari aplikasi tersebut. Satu *activity* biasanya akan dipakai untuk menampilkan aplikasi atau yang bertindak sebagai *user interface* (UI) saat aplikasi diperlihatkan kepada *user*.

b. *Service*

Service tidak memiliki *Graphic User Interface* (GUI), tetapi *service* berjalan secara *background*, sebagai contoh dalam memainkan music, *service* mungkin memainkan *music* atau mengambil data dari jaringan, tetapi setiap *service* harus berada dalam kelas induknya. Misalnya, *media player* sedang memutar lagu dari list yang ada, aplikasi ini akan memiliki dua atau lebih *activity* yang memungkinkan

user untuk memilih lagu misalnya, atau menulis sms sambil player sedang berjalan. Untuk menjaga music tetap di jalankan, *activity player* dapat menjalankan *service*.

c. *Broadcast Receiver*

Broadcast Receiver berfungsi menerima dan bereaksi untuk menyampaikan *notifikasi*. Contoh *broadcast* seperti notifikasi zona waktu berubah, baterai *low*, gambar telah selesai diambil oleh kamera, atau pengubahan referensi bahasa yang digunakan. Aplikasi ini juga dapat menginisiasi *broadcast* misalnya memberikan informasi pada aplikasi lain bahwa ada data yang telah di unduh ke perangkat dan siap untuk digunakan.

d. *Content Provider*

Content Provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam file sistem seperti *database* SQLite. *Content Provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu activity, misalnya ketika kita menggunakan aplikasi yang membutuhkan peta (*Map*), atau aplikasi yang membutuhkan untuk mengakses data kontak dan navigasi, maka disinilah fungsi *Content Provider*.

2.2.6.5 Versi Android

Telepon pertama yang memakai sistem Android adalah HTC Dream, yang dirilis pada 22 Oktober 2008. Pada pengunjung tahun 2010 diperkirakan hampir semua vendor seluler di dunia menggunakan Android sebagai operating system. Adapun versi-versi Android yang pernah dirilis adalah sebagai berikut :

1. *Android* versi 1.1

Pada 9 Maret 2009, Google merilis Android versi 1.1 Abdroid versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam, alarm, *voice search* (pencarian suara), pengiriman pesan dengan G*mail*, dan pemberitahuan email.

2. *Android* versi 1.5 (Cupcake)

Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (Software Development Kit) dengan versi 1.5 (Cupcake). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengupload video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke headset bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan sistem.

3. Android versi 1.6 (Donut)

Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan control applet VPN.

4. Android versi 2.0/2.1 (Éclair)

Pada 3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 (Éclair).

5. Android versi 2.2 (Froyo: Frozen Yoghurt)

Pada bulan Mei 2010 Android versi 2.2 Rev 1 diluncurkan. Android inilah yang sekarang sangat banyak beredar dipasaran, salah satunya adalah dipakai Samsung FX Tab yang sudah ada dipasaran.

6. Android versi 2.3 (Gingerbread)

Android versi 2.3 diluncurkan pada Desember 2010.

7. Android versi 3.0 (Honeycomb) dirilis Februari 2011 sebagai Android 3.0 revisi 1 serta Android versi 3.0 revision 2 telah dirilis pada Juli 2011.

8. Android 3.1 dirilis Mei 2011, sedangkan android 3.1 revisi 2 juga dirilis Mei 2011, serta Android 3.1 revision 3 dirilis pada Juli 2011

9. Android versi 3.2 dirilis Juli 2011
10. Android versi 4.0 dirilis November 2011
11. Android versi 4.1
12. Android versi 4.2
13. Android versi 4.3

2.2.7 API (Application Programming Interface)

Berikut adalah penjelasan mengenai apa itu API :

2.2.7.1 Pengertian API

API adalah singkatan dari application programming interface. API dapat memberikan kait untuk rekan kerja, mitra, atau pengembang pihak ketiga untuk mengakes data dan layanan untuk membangun aplikasi. [5]

2.2.7.2 Cara Menggunakan API

Ketika dua sistem (website, desktop, smartphone) menghubungkan melalui API, kami mengatakan mereka "terintegrasi." Dalam sebuah integrasi, Anda memiliki dua sisi, masing-masing dengan nama khusus. Satu sisi kita telah berbicara tentang: server. Ini membantu untuk mengingat bahwa API ini hanya program lain yang berjalan pada server 3. Ini mungkin bagian dari program yang sama yang menangani lalu lintas web, atau dapat menjadi salah satu benar-benar terpisah. Dalam kedua kasus, itu duduk, menunggu orang lain untuk meminta untuk data.

Sisi lain adalah "klien." Ini adalah program terpisah yang tahu apa data tersedia melalui API dan dapat memanipulasi, biasanya atas permintaan dari pengguna. Sebuah contoh yang bagus adalah aplikasi smartphone yang sync dengan website. Ketika Anda menekan tombol refresh aplikasi Anda, itu berbicara ke server melalui API dan mengambil info terbaru.

Prinsip yang sama berlaku untuk situs web yang terintegrasi. Ketika salah satu situs menarik di data dari yang lain, situs yang menyediakan data bertindak sebagai server, dan situs mengambil data adalah klien.

2.2.7.3 Metode API

Metode permintaan memberitahu server apa tindakan klien ingin server untuk mengambil. Bahkan, metode ini sering disebut sebagai permintaan "kata kerja."

Empat metode yang paling sering terlihat di API adalah:

a. GET - Minta server untuk mengambil sumber daya

b. POST - Minta server untuk menciptakan sumber daya baru

c. PUT - Minta server untuk mengedit / memperbarui sumber daya yang ada

d. DELETE - Minta server untuk menghapus sumber daya

Berikut ini adalah contoh untuk membantu menggambarkan metode ini. Katakanlah ada sebuah restoran pizza dengan API yang dapat digunakan untuk menempatkan pesanan. Anda memesan dengan membuat permintaan POST ke server restoran dengan detail pesanan Anda, meminta mereka untuk membuat pizza Anda. Segera setelah Anda mengirimkan permintaan, namun, Anda sadar bahwa Anda memilih salah gaya kerak, sehingga Anda membuat permintaan PUT untuk mengubahnya.

2.2.7.4 Google Maps API

Aplikasi dan situs web yang menggabungkan data atau fungsionalitas dari dua sumber atau lebih disebut sebagai mashup. Mashup menjadi semakin popular dan telah merevolusi cara informasi digunakan dan divisualisasikan. Solusi pemetaan merupakan salah satu unsur penting dalam banyak mashup ini. Google Maps API memungkinkan anda memanfaatkan kekuatan Google Maps untuk digunakan dalam aplikasi anda sendiri untuk menampilkan aplikasi anda sendiri (atau lainnya) dengan cara yang efisien dan bermanfaat. [4]

2.2.7.4.1 Memuat Google Maps JavaScript API

Untuk memuat Google Maps JavaScript API, harus menggunakan tag script seperti contoh berikut ini [6]:

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KE
Y&callback=initMap"> </script>
```

URL yang terdapat dalam tag script adalah lokasi file JavaScript yang memuat semua simbol dan definisi yang dibutuhkan untuk menggunakan Maps JavaScriptAPI. Tag script ini diperlukan.

Atribut async memungkinkan browser merender bagian selebihnya dari situs web yang akan dibuat saat Maps JavaScript API dimuat. Bila API sudah siap, ia akan memanggil fungsi yang ditetapkan menggunakan parameter callback. Parameter key berisi kunci API aplikasi yang dibuat.

2.2.7.4.2 Elemen DOM Peta

Untuk memuat Google Maps JavaScript API, harus menggunakan tag script seperti contoh berikut ini:

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KE
Y&callback=initMap"> </script>
```

URL yang terdapat dalam tag script adalah lokasi file JavaScript yang memuat semua simbol dan definisi yang dibutuhkan untuk menggunaka Maps JavaScriptAPI. Tag script ini diperlukan.

Atribut async memungkinkan browser merender bagian selebihnya dari situs web yang akan dibuat saat Maps JavaScript API dimuat. Bila API sudah siap, ia akan memanggil fungsi yang ditetapkan menggunakan parameter callback. Parameter key berisi kunci API aplikasi yang dibuat.

2.2.7.4.3 Opsi Peta

Ada dua opsi yang diperlukan untuk setiap peta : center dan zoom.

```
    map                          =                          new
google.maps.Map(document.getElementById('map'),   {   center:
{lat: -34.397, lng: 150.644}, zoom: 8 });
```

2.2.7.4.4 Tingkat Zoom

Resolusi awal untuk menapilkan peta disetel oleh properti zoom, dalam hal ini zoom 0 menyatakan peta bumi yang diperkecil maksimum, dan tingkat zoom yang lebih besar akan memperbesar dengan resolusi lebih tinggi. Menetapkan tingkat zoom sebagai integer.

```
   zoom:8
```

Gambar peta dalam Google Maps dan Maps JavaScript API akan dipecah menjadi “petak” peta dan “tingkat zoom”. Pada tingkat zoom rendah, satu rangkaian kecil petak peta mencakup area yang luas; di tingkat zoom yang lebih tinggi, petak memiliki resolusi lebih tinggi dan mencakup area yang lebih kecil. Daftar ini menunjukkan perkiraan tingkat detail yang di harapkan untuk terlihat di setiap tingkat zoom :

1. 1 : Dunia
2. 5 : Daratan luas/benua
3. 10 : Kota
4. 15 : Jalan
5. 20 : Bangunan

2.2.7.4.5 Objek Peta

Kelas JavaScript yang menyatakan peta adalah kelas Map. Objek di kelass ini mendifinisikan sebuah peta pada lama. (Bisa membuat lebih dari satu instance kelas ini – setiap objek akan mendefinisikan peta terpisah pada laman.) kita membuat sebuah instance baru dari kelas ini menggunakan operator new JavaScript.

```
    map                          =                          new
google.maps.Map(document.getElementById("map"), {...});
```

Bila membuat instance peta baru, menetapkan elemen HTML di laman sebagai container untuk peta. Simpul HTML adalah adak dari objek document JavaScript, dan kita memperoleh referensi ke elemen ini melalui metode document.getElementById().

2.2.7.4.6 Meminta Detail Tempat

Permintaan detail tempat adalah sebuah HTTP URL dari form berikut :

https://maps.googleapis.com/maps/api/place/details/output?parameters

Dimana output mungkin salah satu dari nilai berikut :

1. Json (disarankan) menunjukkan output di JavaScript Object Notation (JSON)

2. Xml menunjukkan output sebagai XML. Seperti standar dalam URL, semua parameter dipisahkan menggunakan karakter ampersand (&). Berikut adalah daftar parameter dan nilai yang mungkin mereka dapatkan.

3. Key(wajib) – Kunci API aplikasi anda. Kunci ini mengidentifikasi aplikasi anda untuk tujuan pengelolaan kuota dan sehingga tempat yang ditambahkan dari aplikasi anda dibuat segera tersedia untuk aplikasi anda.

4. Entah placeid atau reference (kita harus menyediakan salah satu dari ini, tapi tidak keduanya).

5. Placeid – Pengenal teks yang secara unik mengidentifikasi tempat, kembali dari penelusuran tempat.

6. Language (optional) – Kode bahasa, yang menujukkan bahasa mana hasilnya harus dikembalikan, jika memungkinkan. Perlu di catat bahwa beberapa bidang mungkin tidak tersedia dalam bahasa yang diminta.

7. Region – Kode wilayah, yang ditentukan sebagai kode ccTLD (kode negara tingkat atas domain) dua karakter. Kebanyakan kode ccTLD identik dengan kode

ISO 3166-1, dengan beberapa pengecualian. Parameter ini hanya akan berpengaruh, tidak sepenuhnya membatasi, hasil.

2.2.7.4.7 Permintaan Arah

Sebuah permintaan Google Maps Directions API mengambil form sebagai berikut :

https://maps.googleapis.com/maps/api/directions/outputFormat?parameters

Seperti standar dalam URL, semua parameter dipisahkan menggunakan karakter ampersand (&). Berikut adalah daftar parameter dan nilai yang mungkin mereka dapatkan.

1. Parameter Wajib

a. Origin – Alamat , nilai lintang/bujur teks, atau ID tempat dari mana anda ingin menghitung arah.

b. Destination – Alamat, nilai linta/bujur teks, atau ID tempat yang anda inginkan untuk menghitung arah.

c. Key – Kunci aplikasi. Kunci ini mengidentifikasi aplikasi yang ada untuk tujuan pengelolaan kuota.

2. Parameter Pilihan

a. Mode (defaults to driving) - Menentukan moda transportasi untuk digunakan saat menghitung arah.

b. Waypoints – menentukan sebuah array titik arah. Titik arah mengubah rute dengan mengarahkannya melalui lokasi yang ditentukan. Titik arah ditentukan sebagai koordinat garis lintang/bujur, polyline yang dikodekan, ID tempat atau alamat yang akan dikodekan.

c. Alternatives – Jika disetel ke true, tetapkan bahwa layanan arah dapat memberikan lebih dari satu rute alternatif dalam respon. Perhatikan bahwa rute alternative dapat meningkatkan waktu respons dari server.

d. Avoid – Menujukkan bahwa rute yang dihitung harus menghindari fitur yang ditunjukkan. Parameter ini mendukung argument berikut :

a) tolls – Menunjukkan bahwa rute yang dihitung harus menghindari jalan tol/jembatan.

b) highways – Menunjukkan bahwa rute yang dihitung harus menghindari jalan raya.

c) ferries – Menunjukkan bahwa rute yang dihitung harus menghindari feri.

d) indoor – Menujukkan bahwa rute yang dihitung harus mengindari langkah-langkah dalam ruangan untuk petunjuk arah berjalan dan transit.

a. Language – Bahasa untuk mengembalikan hasil. Google biasanya menupdate bahasa yang didukung. Jika bahasa tidak disertakan, API mencoba menggunakan bahasa pilihan seperti yang ditemtukan dalam header Accep-Language, atau bahasa asli dari domain tempat permintaan dikirim.

b. Units – Menentukkan sistem unit yang akan digunakan saat menampilkan hasilnya.

c. Regions – Menentukakkan kode wilayah, yang ditentukan sebagai nilai dua karakter ccltd (“domain tingkat atas”).

d. Arrival_time – Menentukkan waktu kedatangan yang diinginkan untu petunjuk arah transit, dalam hitungan detik sejak tengan malam, 1 Januari 1970 UTC. Kita dapat menentukan departure_time atau arrival_time, namun tidak keduanya. Perhatikan bahwa arrival_time harus ditentukan sebagai integer.

e. Departure_time – Menentukan waktu keberangkatan. Kita bisa menentukan waktunya sebagai bilangan integer dalam hitungan detik sejak tengah malam, 1 Januari 1970 UTC.

f. Traffic_model (default to best_guess) – Menentukan asumsi yang digunakan saat menghitung waktu lalu lintas. Setelan ini memengaruhi nilai yang dikembalikan di field duration_in_traffic dalam respon, yang berisi perkiraan waktu

lalu lintas berdasarkan rata-rata historis. Parameter traffic_model hanya dapat ditentukan untuk petunjuk arah mengemudi dimana permintaan mencakup derpature_time.

g. Transit_mode – Menentukan satu atau lebih mode transit pilihan.

h. Transit_routing_preference – Menentukan preferensi untuk rute transit. Dengan menggunakan parameter ini, kita dapat memilah opsi yang dikembalikan, daripada menerima rute default terbaik yang dipilih oleh API.

2.2.7.4.8 Pencarian Tempat

Google Places API Web Service memungkin kita untuk menanyakan informasi tempat pada berbagai kategori, seperti : tempat usaha, tempat menarik, lokasi geografis, dan lainnya. Kita dapat mencari tempat baik dengan kedekatan atau string teks.

2.2.7.4.9 Permintaan Pencarian Sekitar

Penelusuran di sekitar memungkinkan kita mencari tempat dalam area tertentu. Kita dapat memperbaiki permintaan pencarian kita dengan menyediakan kata kunci atau menentukan jenis tempat yang kita cari.

Permintaan penelusuran di sekitar adalah URL HTTP dari form berikut :

https://maps.googleapis.com/maps/api/place/nearbysearch/output?parameters

Parameter tertentu diperlukan untuk memulai permintaan Pencarian Terdekat. Seperti standar dalam URL, semua parameter dipisahkan menggunakan karakter ampersand (&).

1. Parameter Wajib

a. Key – Kunci API aplikasi yang dibangun. Kunci ini mengidentifikasi aplikasi yang dibangun untuk tujuan pengelolaan kuota dan sehingga tempat yang ditambahkan dan aplikasi yang dibangun.

b. Location - Lintang/bujur di mana untuk mengambil informasi tempat. Ini harus ditentukan sebagai garis lintang, bujur.

c. Radius – Tetapkan jarak (dalam meter) di mana untuk mengembalikan hasil tempat radius maksimum yang diizinkan adalah 50.000 meter. Perhatikan bahwa radius tidak disertakan jika rankby=distance.

d. Jika rankby=distance ditentukan, maka satu atau beberapa kata kunci, nama, atau jenis diperlukan.

2. Parameter Optional

a. Keyword – Sebuah istilah yang akan disesuaikan dengan semua konten yang telah diindeks Google untuk tempat ini, termasuk namun tidak terbatas pada nama, jenis, dan alamat, serta ulasan pelanggan dan konten pihak ketiga lainnya.

b. Language – Kode bahasa, yang menunjukkan bahasa mana hasilnya harus dikembalikan, jika memungkinkan.

c. Minprice and maxprice (optional) – Batasi hasil hanya pada tempat di kisaran yang ditentukan. Nilai yang valid berkisar antara 0 (paling terjangkau) sampai 4 (aling mahal), inklusif. Jumlah pasti yang ditunjukkan oleh nilai tertentu akan bervariasi dari satu wilayah ke wilayah lainnya.

d. Name – Sebuah istilah yang harus disesuaikan dengan semua konten yang telah diindeks Google untuk tempat ini. Setara dengan kata kunci. Bidang nama tidak lagi dibatasi untuk menempatkan nama.

e. Opennow – Mengembalikkan hanya tempat-tempat yang terbuka untuk bisnis pada saat query dikirim. Tempat yang tidak menentukan jam buka di basis data Google Places tidak akan dikembalikkan jika kita memasukkan parameter ini ke query kita.

f. Rankby – Menentukan urutan hasil yang tercantum. Perhatikan bahwa rankby tidak boleh disertakan jika radius ditentukan. Kemungkinan values :

a) Prominence (default) – Pilihan ini mengurutkan hasil berdasarkan kepentingan mereka. Peringkat akan menyukai tempat- tempat menonjol di dalam

area yang ditentukan. Keunggulan dapat dipengaruhi oleh peringkat suatu tempat di indeks Google, popularitas global, dan faktor lainnya.

b) Distance – Pilihan ini bias mencari hasil dalam urutan naik menurut jarak dari location yang ditentukan. Bila distance ditentukan, satu atau beberapa kata keyword, name, atau type diperlukan.

c) Type - Batasi hasilnya ke tempat yang sesuai dengan jenis yang ditentukan. Hanya satu jenis yang dapat ditentukan (jika lebih dari satu jenis disediakan, semua jenis mengikuti entri pertama diabaikan).

d) Pagetoken – Mengembalikan 20 hasil berikutnya dari pencarian yang sebelumnya dijalankan. Menetapkan parameter pagetoken akan melakukan pencarian daengan parameter yang sama yang digunakan sebelumnya – semua parameter selain pagetoken akan diabaikan.

e) Region - Kode wilayah, yang ditentukan sebagai kode cctld (country code top-level domain) dua karakter. Kebanyakan kode ccltd identic dengan kode ISO 3166-1, dengan beberapa pengecualian. Parameter ini hanya akan berpengaruh, tidak sepenuhnya membatasi, hasil pencarian. Jika hasil yang lebih relevan ada di luar wilayah uang ditentukan, mereka mungkin disertakan.

2.2.8 LBS (*Location Based Service*)

Layanan lokasi dapat didefinisikan sebagai layanan yang mengintegrasikan lokasi atau posisi perangkat mobile dengan informasi lainnya sehingga memberikan nilai tambah bagi pengguna. [6] Location Service memiliki tradisi yang panjang. Sejak tahun 1970an, Departement of Defense A.S telah mengoperasikan sistem penentuan posisi global (GPS), infrastruktur satelit yang melayani penentuan posisi orang dan objek.

Awalnya, GPS dipahami untuk tujuan militer, namun pemerintah A.S. memutuskan pada tahun 1980an untuk membuat data posisi sistem tersedia secara bebas untuk industri lain di seluruh dunia. Sejak saat itu, banyak industri telah mengambil kesempatan untuk mengakses data posisi melalui GPS dan sekarang menggunakannya untuk meningkatkan produk dan layanan mereka.

Misalnya, industri otomotif telah mengintegrasikan sistem navigasi ke mobil untuk beberapa waktu.[7] Dalam sistem penentuan posisi tradisional, informasi lokasi biasanya diturunkan oleh perangkat dan dengan bantuan sistem satelit (misalnya, penerima GPS) .

Namun, minat yang meluas pada layanan berbasis lokasi (LBS) dan teknologi yang mendasarinya seperti yang dibahas di buku ini benar-benar mulai berkembang hanya di akhir tahun 1990an, ketika teknologi pelokalan tipe baru dan minat pasar baru terhadap layanan data dipicu oleh operator jaringan seluler.

Pada sekitar tahun 1997, jaringan bergerak banyak digunakan di Eropa, Asia, dan Amerika Serikat, dan pendapatan dari layanan telepon telah terbukti signifikan bagi operator seluler. Namun, meski layanan suara seluler terus menjadi penghasil pendapatan utama telco, pertumbuhan telepon seluler terbatas dan harga per menit menurun.[1]

2.2.9 GPS (*Global Positioning System*)

Global Positioning System (GPS) adalah sistem navigasi berbasis satelit yang dikembangkan oleh Departemen Pertahanan A.S pada awal 1970-an. Awalnya, GPS dikembangkan sebagai sistem militer untuk memenuhi kebetuhan militer A.S. namun, kemudian tersedia untuk warga sipil., dan sekarang sistemnya dual-use yaitu bisa diakses oleh militer dan warga sipil. [8]

GPS memberikan informasi postioning dan timing yang terus menerus, dimana saa di dunia dalam kondisi cuaca apapun. Karena melayani jumlah pengguna yang tidak terbatas serta digunakan untuk alasan keamanan, GPS adalah sistem satu arah (pasif). Artinya pengguna hanya bisa menerima sinyal satelit.

2.2.10 Java

Java adalah bahasa object-oriented baru yag mendapat perhatian luas dari industry dan akademisi. Java dikembangkan oleh James Gosling dan timnya di Sun Microsystems di California. Bahasa ini didasarkan pada C dan C++ dan pada awalnya ditujukan untuk menulis program yang mengendalikan peralatan konsumen seperti toaster, microwave, oven, dan lainnya. Bahasa itu pertama kali

disebut Oak, dinamai setelah pohon oak di luar dari kantor Gosling, tapi nama itu sudah diambil, jadi tim menamainya Java.

Java sering digambarkan sebagai bahasa pemograman web karena penggunaannya di meluis program yang disebut applet yang berjalan dalam browser web. Browser web untuk mengeksekusi applet Java. Applet memungkinkan lebih dinamis dan fleksibel di Internet, dan fitur ini saja membuat Java menjadi bahasa yang menarik untuk dipelajari. Namun, kita tidak terbatas pada menulis applet di Java. Kita bisa juga menulis aplikasi Java. Aplikasi Java adalah sebuah program lengkap yang berdiri sendri tidak memerlukan Web Browser . aplikasi Java analog terhadap program yang kita tulis dalam bahasa pemograman lainnya.[10]

2.2.11 Teori Pemodelan dan UML

Menurut Rosa A & Shalahuddin di dalam bukunya bahwa pemodelan adalah gambaran dari realita yang simpel dan dituangkan dalam bentuk pemetaan dengan aturan tertentu. Salah satu pemodelan yang saat ini paling banyak digunakan adalah UML. Unified Modeling Language (UML) adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek [11].

Secara fisik, UML adalah sekumpulan spesifikasi yang dikeluarkan oleh OMG (*Object Management Group*).UML terbaru adalah UML 2.3 yang terdiri dari 4 macam spesifikasi, yaitu *Diagram Interchange Specification,* UML Infrastructure, UML Superstructure, dan Object Constraint Language (OCL). Seluruh spesifikasi tersebut dapat diakses di website http://www.omg.org.

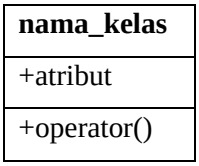
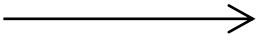
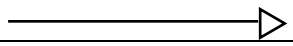
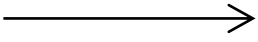
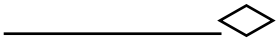
2.2.11.1 Diagram Kelas

Menurut Rosa A & Shalahuddin bahwa diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau opeasi.

- Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
- Operasi atau metode adalah fungsi-sungsi yang dimiliki oleh suatu kelas

Berikut adalah simbol-simbol yang ada padad diagram kelas :

Tabel 2.1 Simbol Diagram Kelas

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Antarmuka / *interface*  nama_interface	Sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi / *association*	Relasi antar kelas dengan makna umum, asosisasi biasanya juga disertai dengan multiplicity
Asosiasi berarah / *directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasnaya juga disertai dengan multiplicity
generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi(umum khusus)
Kebergantungan / *dependency* 	Relasi antar kelas dengan makna ketergantungan antar kelas
Agregasi / *aggregation* 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

2.2.11.2 Use Case Diagram

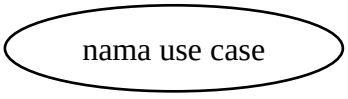
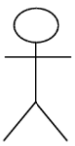
Use Cace atau diagram use case merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, use case digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu(Rosa A & Shalahuddin ,2018 : 155).

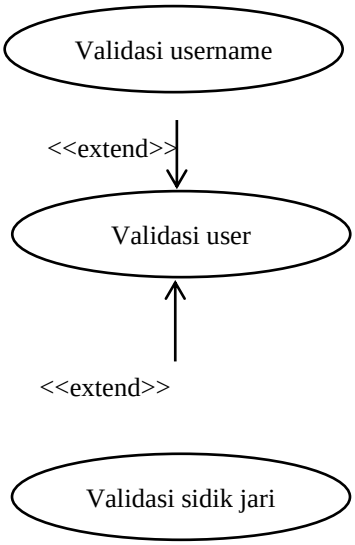
Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case.

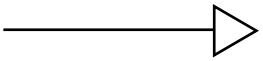
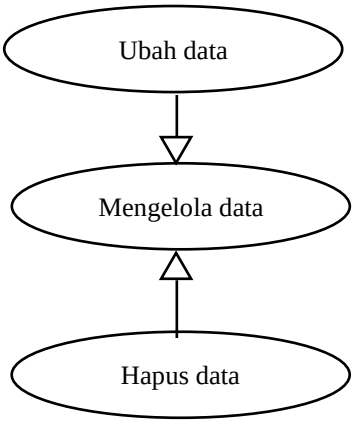
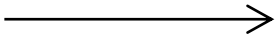
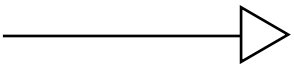
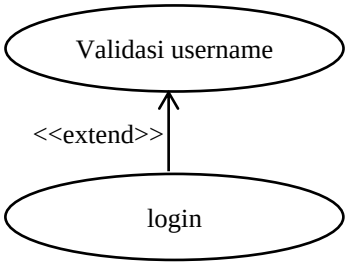
- Aktor merupakan orang, prosesm atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang
- Use case merupakan fungsionalitas yang disediakan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

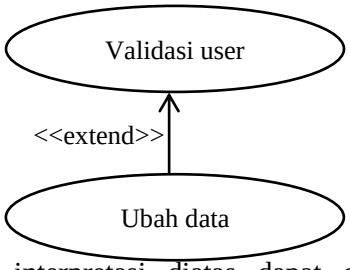
Berikut adalah simbol-simbol yang ada pada diagram use case :

Tabel 2.2 Simbol Diagram Use Case

Simbol	Deskripsi
use case 	Fungisonalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase nam use case
Aktor / actor  nama aktor	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;

	biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / association	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor
Ekstensi / extend <<extend>>	Relasi use case tambahan kesebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip *intheritance* pada pemrograman berorientasi objek;biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, miasl  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*	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case

	dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalanya :  Arah panah mengarah pada use case yang menjadi generalisasinya (umum)
Menggunakan / include / uses <<include>>  <<uses>> 	Relasi use case tambahan kesebuah use case dimana use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini Adadua sudut pandang yang cukup besar mengenai include di use case: • Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, misal pada kasus berikut : 

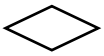
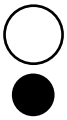
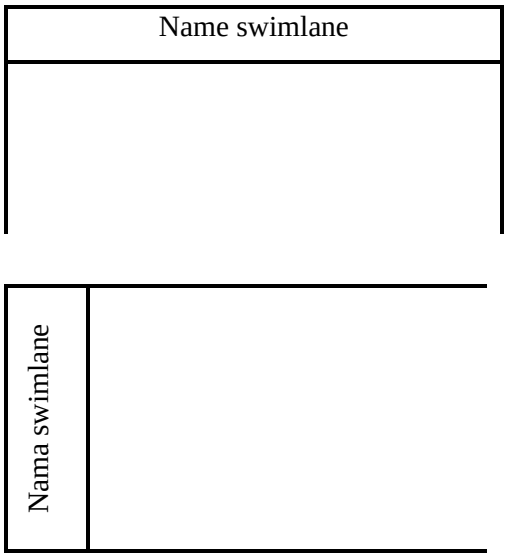
	• Include berarti use case yang tambahan akan selalu melakukan pengecekan apakah use case yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan, misal pada kasus berikut :  Kedua interpretasi diatas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

2.2.11.3 Diagram Aktivitas

Diagram aktivitas atau activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem(Rosa A & Shalahuddin ,2018 : 161).

Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Tabel 2.3 Simbol Diagram Aktivitas

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

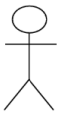
2.2.11.4 Diagram Sekuen

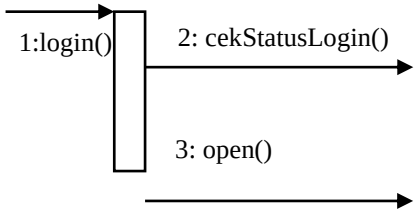
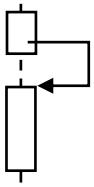
Diagram sekuen menggambarkan kelakukan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada use case(Rosa A & Shalahuddin ,2018 : 165).

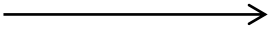
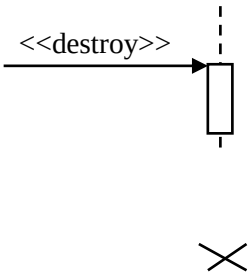
Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak use case yang didefinisikan maka diagram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

Berikut adala simbol-simbol yang ada pada diagram sekuen :

Tabel 2.4 Simbol Diagram SekuenTabel

Simbol	Deskripsi
Aktor  nama aktor Atau nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informais 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; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Garis hidup / lifeline 	Menyatakan kehidupan satu objek

Objek nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  Maka **cekStatusLogin()** dan **open()** dilakukan didalam metode **login()** Aktor tidak memiliki waktu aktif
Pesan tipe create <<create>> 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe call 1:nama_metode() 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek laina atau dirinya sendiri, 

	1: nama_motede() 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 1: masukan 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim
Pesan tipe return 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
Pesan tipe destroy 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaikya jika ada create maka ada destroy