BAB II
TINJAUAN PUSTAKA

2.1 Konsep Sistem Informasi

2.1.1 Sistem

Sistem didefinisikan sebagai sekumpulan prosedur yang saling berkaitan dan saling terhubung untuk melakukan suatu tugas bersama-sama. Sistem informasi terdiri atas tiga komponen utama. Ketiga komponen tersebut mencakup *software, hardware, dan brainware* [4].

Sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling tergantung satu sama lain, dan terpadu. Sebuah sistem terdiri atas bagian-bagian komponen yang terpadu untuk satu tujuan. Model dasar dari bentuk sistem ini adalah adanya masukan, pengolahan, dan keluaran [5].

Sistem yang baik memiliki karakteristik sebagai berikut [6]:

1. Komponen

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk satu kesatuan. Komponen sistem terdiri dari komponen yang berupa subsistem atau bagian-bagian dari sistem.

2. Batasan sistem (*boundary*)

   Batasan sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lain atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan. Batasan suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan luar sistem (*environment*)

   Lingkungan luar sistem (*environment*) adalah di luar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan dapat bersifat menguntungkan yang harus tetap dijaga dan yang merugikan yang harus dijaga dan dikendalikan, kalau tidak akan mengganggu kelangsungan hidup dari sistem.

4. Penghubung sistem (*interface*)

   Penghubung sistem merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber data

mengalir dari subsistem ke subsistem lain. Keluaran (*output*) dari subsistem akan menjadi masukan (*input*) untuk subsistem lain melalui penghubung.

5. Masukan sistem (*input*)

   Masukan adalah energi yang dimasukkan ke dalam sistem yang dapat berupa perawatan (*maintenance input*) dan masukan sinyal (*signal input*). Maintenance input adalah energi yang dimasukkan agar sistem dapat beroperasi. *Signal input* adalah energi yang diproses untuk didapatkan keluaran. Contoh dalam sistem komputer, program adalah *maintenance input* sedangkan data adalah *signal input* untuk diolah menjadi informasi.

6. Keluaran sistem (*output*)

   Keluaran sistem adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembangunan. Contoh, komputer menghasilkan panas yang merupakan sisa pembuangan, sedangkan informasi adalah keluaran yang dibutuhkan.

7. Pengolah sistem

   Suatu sistem menjadi bagian pengolah yang akan mengubah masukan menjadi keluaran. Sistem produksi akan mengolah bahan baku menjadi bahan jadi, sistem akuntansi akan mengolah data menjadi laporan-laporan keuangan.

8. Sasaran sistem

   Suatu sistem pasti mempunyai tujuan (*goal*) atau sasaran (*objective*). Sasaran dari sistem sangat menentukan *input* yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem [6].

2.1.2 Informasi

Informasi adalah data yang telah diolah menjadi suatu bentuk yang penting bagi si penerima dan mempunyai nilai nyata atau yang dapat dirasakan dalam keputusan-keputusan yang sekarang atau keputusan-keputusan yang akan datang. Jadi, intinya informasi adalah data yang telah diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi penerimanya. Sumber informasi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian atau kesatuan nyata [6].

Suatu informasi dikatakan bernilai bila manfaat lebih efektif dibandingkan dengan biaya mendapatkannya. Akan tetapi, perlu diperhatikan bahwa informasi yang

digunakan di dalam suatu organisasi umumnya digunakan untuk beberapa kegunaan, sehingga tidak memungkinkan dan sulit untuk menghubungkan suatu bagian informasi pada suatu masalah tertentu dengan biaya untuk memperolehnya, karena sebagian besar informasi dinikmati oleh hanya satu pihak di dalam perusahaan [5].

Fungsi informasi yaitu menambah pengetahuan atau mengurangi ketidakpastian pemakai informasi, karena informasi berguna dalam memberikan gambaran tentang suatu permasalahan sehingga pengambil keputusan dapat menentukan keputusan lebih cepat. Informasi juga memberikan standar, aturan, maupun indikator bagi pengambil keputusan. Kegunaan suatu informasi tergantung pada [6]:

1. Tujuan si penerima

   Bila tujuannya untuk memberi bantuan, maka informasi harus membantu si penerima untuk memperoleh apa yang diusahakan.
2. Ketelitian penyampaian dan pengolahan data

   Dalam menyampaikan dan mengolah data, inti dan pentingnya informasi harus diperhatikan.
3. Waktu

   Apakah informasi itu masih *up to date*?
4. Ruang atau tempat

   Apakah informasi itu tersedia dalam ruangan atau tempat yang tepat?
5. Bentuk

   Dapatkah informasi itu digunakan secara efektif? Apakah informasi itu menunjukkan hubungan-hubungan yang diperlukan, bidang-bidang yang memerlukan perhatian manajemen? Dan apakah informasi itu menekankan situasi-situasi yang ada hubungannya?
6. Semantik

   Apakah hubungan antara kata-kata dan arti yang ingin disampaikan cukup jelas? Apakah ada kemungkinan salah tafsir?

2.1.3 Sistem Informasi

Sistem informasi merupakan suatu kombinasi teratur apapun dari orang-orang, *hardware, software,* jaringan komunikasi, dan sumber daya data yang mengumpulkan, mengubah, dan menyebarkan informasi dalam sebuah organisasi [7]. Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan [5]. Dalam penerapannya, sebuah sistem informasi dapat berupa sebuah *mainframe*, sebuah *server* dari komputer biasa, maupun *hosting* di internet pada sebuah komputer *server*. Namun tetap saja ada kesamaan di antara ketiga penerapan berbeda ini. Kesamaan itu yaitu sama-sama menggunakan sarana jaringan komputer (*intranet* maupun *internet*) untuk melakukan pemrosesan data secara bersama (terdistribusi), baik oleh beberapa pengguna maupun beberapa grup pengguna, menggunakan layanan/fitur/aplikasi yang disertakan [4].

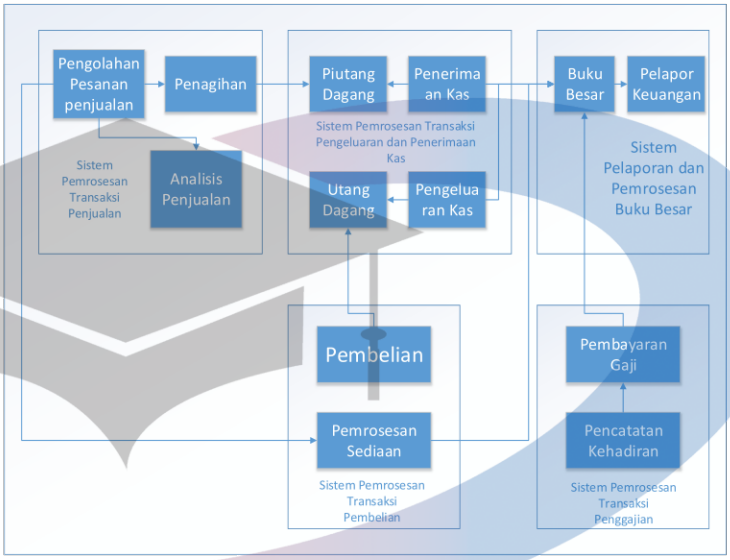
Komponen-komponen dari sistem informasi adalah sebagai berikut [7]:

1. Komponen *input*, adalah data yang masuk ke dalam sistem informasi.
2. Komponen model, adalah kombinasi prosedur, logika, dan model matematika yang memroses data yang tersimpan di basis data dengan cara yang sudah ditentukan untuk menghasilkan keluaran yang diinginkan.
3. Komponen *output*, adalah hasil informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.
4. Komponen teknologi, adalah alat dalam sistem informasi yang digunakan untuk menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan *output*, serta memantau pengendalian sistem.
5. Komponen basis data, adalah kumpulan data yang saling berhubungan yang tersimpan di dalam komputer dengan menggunakan *software database*.
6. Komponen kontrol, adalah komponen yang mengendalikan gangguan terhadap sistem informasi [7].

Jenis-jenis sistem informasi berdasarkan area fungsional adalah sebagai berikut [8]:

1. Sistem informasi akuntansi

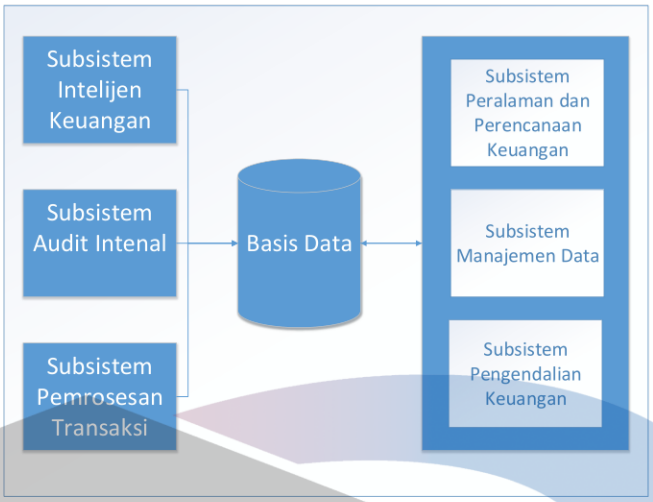
   Sistem informasi akuntansi merupakan sistem informasi yang paling tua dan paling banyak digunakan dalam bisnis. Bodnar dan Hopwood (1993) mendefinisikan sistem informasi akuntansi sebagai kumpulan sumber daya yang dirancang untuk mentransformasikan data keuangan menjadi informasi.



Gambar 2.1 Model Sistem Informasi Akuntansi

2. Sistem informasi keuangan

   Sistem informasi keuangan digunakan untuk mendukung manajer keuangan dalam pengambilan keputusan yang menyangkut persoalan keuangan perusahaan dan pengalokasian serta pengendalian sumber daya keuangan dalam perusahaan. Sistem ini tidak hanya mendasarkan data internal, melainkan juga menggunakan data yang berasal dari sumber eksternal.



Gambar 2.2 Model Sistem Informasi Keuangan

3. Sistem informasi manufaktur

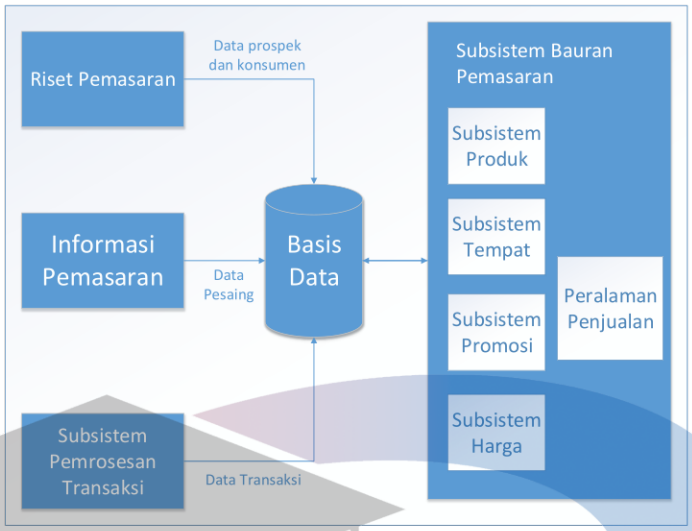
Sistem informasi manufaktur merupakan sistem yang digunakan untuk mendukung fungsi produksi, yang mencakup seluruh kegiatan yang terkait dengan perencanaan dan pengendalian proses untuk memproduksi barang atau jasa.

Gambar 2.3 Model Sistem Informasi Manufaktur

4. Sistem informasi pemasaran

Sistem informasi pemasaran adalah sistem informasi yang menyediakan informasi yang dipakai oleh fungsi pemasaran. Sistem ini mendukung keputusan yang berkaitan dengan bauran pemasaran (*marketing mix*), yang mencakup:

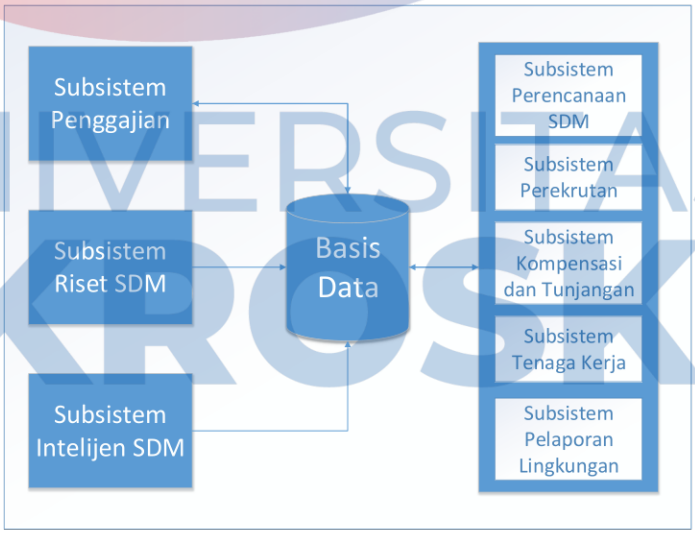
a. produk (barang dan jasa) yang perlu ditawarkan
b. tempat yang menjadi sasaran pemasaran
c. promosi yang perlu dilakukan, dan
d. harga produk



Gambar 2.4 Model Sistem Informasi Pemasaran

5. Sistem informasi sumber daya manusia

Sistem informasi sumber daya manusia biasa disebut HRIS. Selain HRIS, istilah lain yang sering dipakai yaitu *human resource management information system* (HRMIS) dan *human resource management system* (HRMS) [8].



Gambar 2.5 Model Sistem Informasi Sumber Daya Manusia

2.2 Situs Web

Situs web adalah keseluruhan halaman-halaman web yang terdapat dalam sebuah domain yang mengandung informasi. Sebuah situs web biasanya dibangun atas

banyak halaman web yang saling berhubungan. Jadi dapat dikatakan bahwa, situs web adalah kumpulan halaman-halaman. yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing-masing dihubungkan dengan jaringan-jaringan halaman [9].

2.2.1 Pengembangan *Front-End*

Front-end dari sebuah website adalah bagian yang langsung dilihat oleh user. User juga langsung berinteraksi pada bagian ini. Bagian ini dibangun menggunakan HTML dan CSS [10].

HTML5 (*Hyper Text Markup Language*) adalah sebuah *markup* untuk menstrukturkan dan menampilkan isi dari halaman web. Tujuan utama pengembangan HTML5 adalah untuk memperbaiki teknologi HTML agar mendukung multimedia terbaru, mudah dibaca manusia dan mudah dimengerti oleh mesin. HTML5 merupakan salah satu karya *World Wide Web Consortium* (W3C) untuk mendefinisikan sebuah bahasa *markup* tunggal yang dapat ditulis dengan cara HTML maupun XHTML [10].

CSS3 (*Cascading Style Sheet*) merupakan aturan untuk mengendalikan beberapa komponen dalam sebuah web sehingga akan lebih terstruktur dan seragam. CSS bukan bahasa pemrograman. CSS dapat mengendalikan ukuran gambar, warna bagian tubuh pada teks, warna tabel, ukuran *border*, warna *border*, warna *hyperlink*, warna *mouse over*, spasi antarparagraf, spasi antarteks, margin kiri, kanan, atas, bawah, dan parameter lainnya. CSS adalah bahasa *style sheet* yang digunakan untuk mengatur Halaman dokumen. Dengan adanya CSS pengguna dapat menampilkan halaman yang sama dengan format yang berbeda [10].

Alasan mengapa HTML membutuhkan CSS adalah karena CSS memiliki kekuatan yang tidak dimiliki oleh HTML, beberapa diantarnya adalah sebagai berikut [11]:

a. Menentukan gambar, baik latar belakang atau untuk latar depan.
b. Memberi warna latar belakang tag dan mengatur lebar maupun tingginya.
c. Membuat bingkai

d. Membuat halaman dengan dua kolom, dan sebagainya.

2.2.2 Pengembangan *Back-End*

Back-end atau sering disebut *server-side* pada dasarnya adalah tempat dimana proses suatu aplikasi atau sistem berjalan. Pada *back-end* data ditambahkan, diubah atau dihapus. *Back-end* mencakup segala sesuatu yang biasanya tidak dilihat atau berinteraksi langsung kepada *user*, seperti *database* dan *server*. *Back-end* yang seringkali digunakan adalah PHP [12].

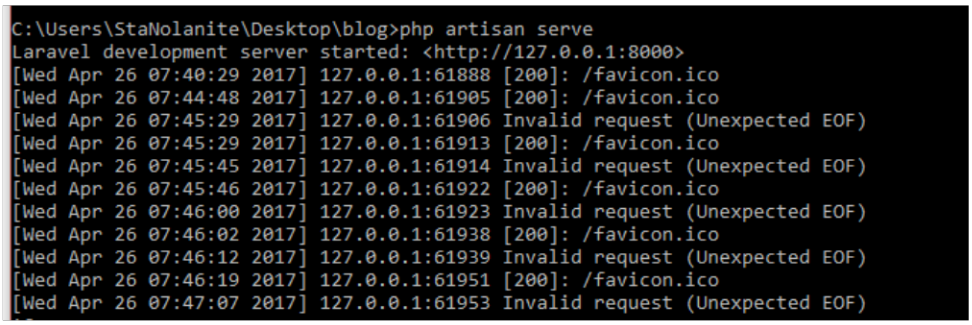
PHP adalah singkatan dari *Hypertext Preprocessor* yang merupakan bahasa *scripting server-side*, artinya dijalankan di server, kemudian outputnya dikirim ke client (*browser*). PHP digunakan untuk membuat aplikasi web dan mengolah data dalam sebuah web. PHP memungkinkan pembuatan web yang sifatnya lebih dinamis, dengan PHP kita juga dapat menampilkan atau menjalankan beberapa file dalam 1 file dengan cara di-*include* atau *require* (yang merupakan salah satu fungsi dalam *Script* PHP) [13].

Framework adalah suatu struktur konseptual dasar yang digunakan untuk memecahkan atau menangani suatu masalah yang kompleks. Singkatnya, *framework* adalah wadah atau kerangka kerja dari sebuah situs web yang akan dibangun. Dengan menggunakan kerangka tersebut waktu yang digunakan dalam membuat situs web lebih singkat dan memudahkan dalam melakukan perbaikan. Laravel adalah *framework* berbasis PHP yang sifatnya *open source*, dan menggunakan konsep *model – view – controller*. Laravel berada di bawah lisensi MIT *License* dengan menggunakan Github sebagai tempat berbagi *code* menjalankannya [14].

Berikut adalah dasar-dasar Laravel [14]:

1. Artisan

   Artisan adalah *command line* atau perintah yang dijalankan melalui terminal dan disediakan beberapa perintah perintah yang dapat digunakan selama melakukan pengembangan dan pembuatan aplikasi. Salah satu fungsi dari php artisan yaitu “php artisan serve”. Php artisan serve berfungsi untuk membuka situs web yang telah dibuat tanpa menggunakan *web server* lokal. Berikut adalah contoh salah satu penggunaan artisan dalam laravel:



Gambar 2.6 PHP Artisan Level

2. *Routing*

   Routing adalah suatu proses yang bertujuan agar suatu item yang dinginkan dapat sampai ke tujuan. Dengan menggunakan *routing* dapat ditentukan halaman halaman yang akan muncul ketika dibuka oleh *user.* Pengaturan *routing* di laravel biasanya terletak di file *web.php.* File *web.php* terletak di dalam folder *routes.*

3. *Controller*

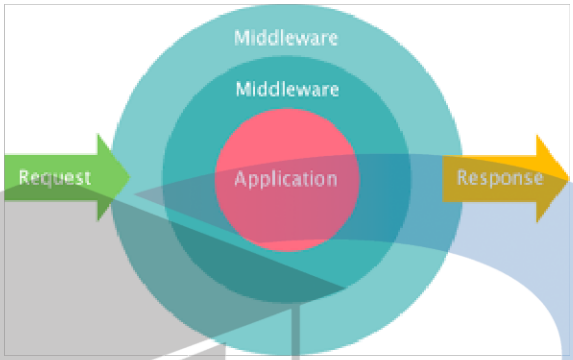
   Controller adalah suatu proses yang bertujuan untuk mengambil permintaan, menginisialisasi, memanggil model unutk dikirimkan ke *view*. Ada dua cara membuat *controller* di laravel. Cara pertama adalah dibuat *file controller* secara manual dan dituliskan *code extends controller* di dalamnya. Cara kedua adalah dibuat *file controller* menggunakan *command line* dengan menuliskan "php artisan make *controller* nama_file_controller". Permintaan yang dibuat dalam laravel harus berada di dalam *controller*, kemudian dilempar melalui *routing* untuk mendapat permintaan yang diinginkan.

4. View (*blade templating*)

   Blade adalah *template engine* bawaan dari laravel. *Blade* memiliki kode kode yang lebih mudah unutk menghasilkan laravel. Cara membuat file.blade dilakukan secara manual dengan membuat nama_file.php.blade di dalam *folder views*. Di dalam *blade* dapat dibuat *template* master dan *template inheritance*. Pembuatan t*emplate* master dan turunannya ini bertujuan agar elemen yang sama tidak ditulis secara berulang-ulang. Pada *template inheritance* diberikan kode *"extend (nama_layout) dan section (nama_content)"*.

5. *Middleware*

Middleware adalah penengah antara *request* yang masuk dengan *controller* yang dituju. Cara membuat *middleware* menggunakan artisan dengan mengetikkan "php artisan make:middleware nama_file". *File middleware* berada di dalam folder *middleware*.



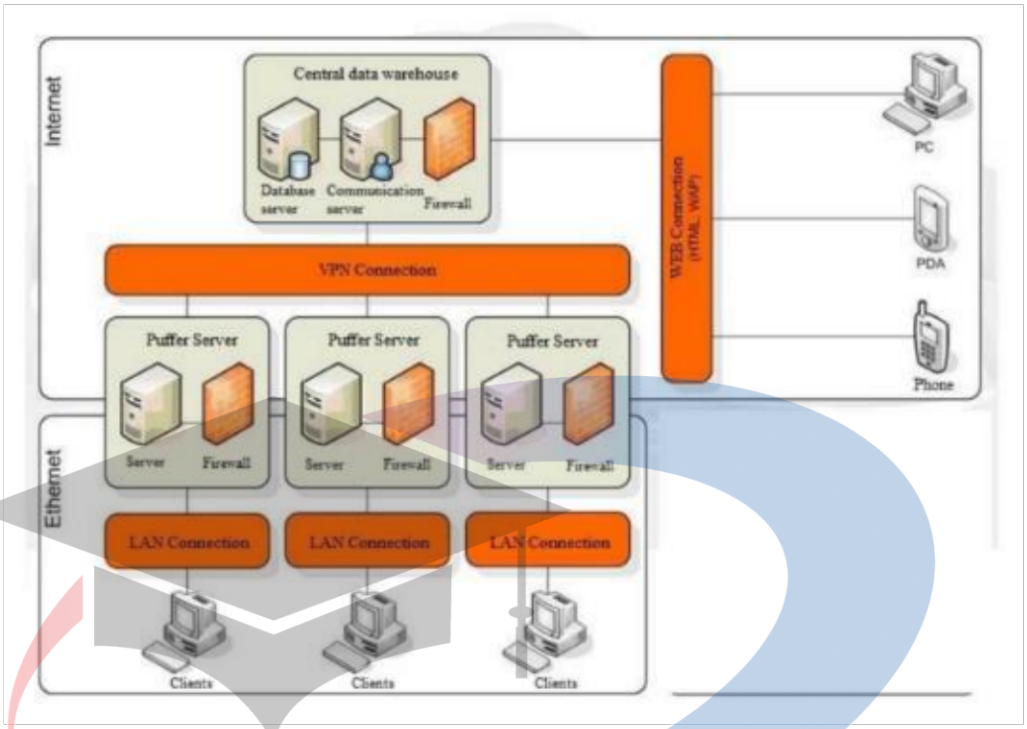
Gambar 2.7 MiddleWare

6. *Session*

Session adalah sebuah cara yang digunakan untuk penyimpanan pada server dan penyimpanan tersebut digunakan pada beberapa halaman termasuk halaman itu sendiri. Dalam menggunakan *session* ada dua cara. Cara yang pertama *session* dapat dibuat menggunakan *Request.* Cara yang kedua dapat digunakan fungsi global *helper session* [14].

2.3 Basis Data

Basis data adalah keseluruhan informasi yang terintegrasi dan terhubung secara logis, antara sistem data dan koneksi diantara mereka yang disimpan sejajar. Untuk dapat bekerja dengan *database*, perancangan yang efektif sangat penting. Konsep dari *database system* terdiri dari *database*, sumber daya komputer, serta *database-administrators* yaitu orang-orang yang melakukan perancangan dan pemrograman *database*. *Database* adalah semacam pengumpul data, yang menyimpan data yang berhubungan dengan tugas yang diberikan, dan teratur. Akses data juga dikelola oleh *database*, serta menjamin perlindungan data, dan juga melindungi integrasi data [15].



Gambar 2.8 *Database Management System*

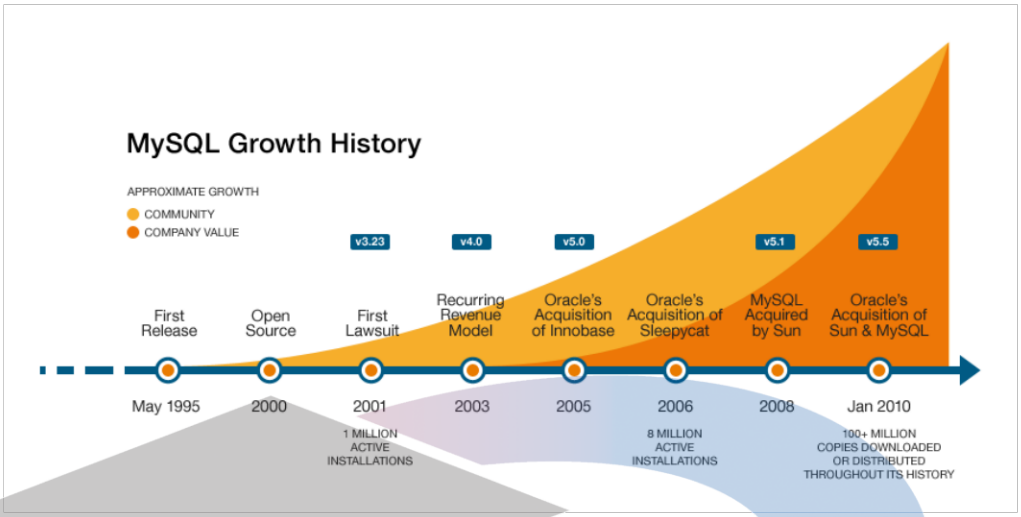
Setiap *database* memiliki struktur bagian dalam yang mencakup uraian semua elemen data dan koneksi di antara mereka, struktur ini disebut dengan *schema database*. *Metadata* yang paling signifikan berisi definisi tipe data dan referensi untuk koneksi dan hubungan antar data. *Metadata* berisi informasi sehubungan dengan administrasi *database*, jadi dengan bantuan *metadata* dapat dilakukan penyimpanan informasi struktural selain data aktual. Pembangunan *database* mungkin saja berbeda,dan bergantung pada model yang diterapkan. Namun, ada beberapa prinsip umum yang hampir digunakan disetiap aplikasi berdasarkan *database*, yaitu [15]:

1. *Table* atau data tabel adalah tabel dua dimensi yang menunjukkan data yang terhubung secara logis, dimana tabel terdiri dari kolom dan baris.
2. *Record* adalah deretan dari *database*, kita menyimpan data-data yang bergantung satu sama lain didalam *record*. Dua baris pada tabel berisikan nilai konkrit dari fitur utama.
3. *Field* adalah kolom dari tabel, setiap kolom pada tabel memiliki fungsi tertentu yang memiliki nama dan tipe.

4. *Elementary data* adalah nilai-nilai dalam sel tabel yang merupakan atribut konkret dari entitas.
5. *Entity* adalah apa yang ingin digambarkan dan data apa yang ingin untuk disimpan dan dikumpulkan dalam *database*. Kita dapat menganggap entitas sebagai seseorang, kita menyebutnya sebagai benda atau benda yang dapat dipisahkan dengan baik dari penyimpanan data, dan apa yang kita tampilkan dengan atribut. Entitas dapat berupa pembayaran pekerja, material, seseorang, dan lain-lain, dalam bentuk ini fungsi entitas sebagai gagasan abstrak. Dapat dikatakan bahwa entitas adalah abstraksi dari hal-hal konkret, dan menjadi kebiasaan untuk menggunakan ekspresi dari tipe entitas ke entitas abstrak.
6. *Attribute* adalah salah satu fitur dari entitas, entitas dapat ditampilkan dengan jumlah atribut. Sebagai contoh, nama seseorang dapat menjadi fitur.
7. *Entity occurence* adalah fitur konkrit yang diberikan entitas, kejadian entitas sesuai dengan catatan, dalam praktiknya tipe entitas juga dapat disebut tipe rekaman (tipe rekaman atau tipe struktur).
8. *Data redundancy* adalah data yang tersimpan lebih dari sekali didalam *database*. Hampir tidak mungkin untuk menghindari redundansi, untuk meminimalkan kejadian berulang metode yang dapat digunakan adalah mengambil data yang diulang selama perancangan *database*, dan menyimpannya secara terpisah dan merujuknya ketempat yang tepat [15].

2.4 MySQL

MySQL adalah *free-software database* yang pada awalnya dikembangkan dan dirilis pertama kali pada tahun 1995. MySQL dinamai *My*, yaitu putri Michael Widenius yang merupakan salah satu pencetus *database* ini. Pada awalnya MySQL diproduksi di bawah GNU General Public License, dimana *source code* yang tersedia dapat diperoleh secara bebas [16].

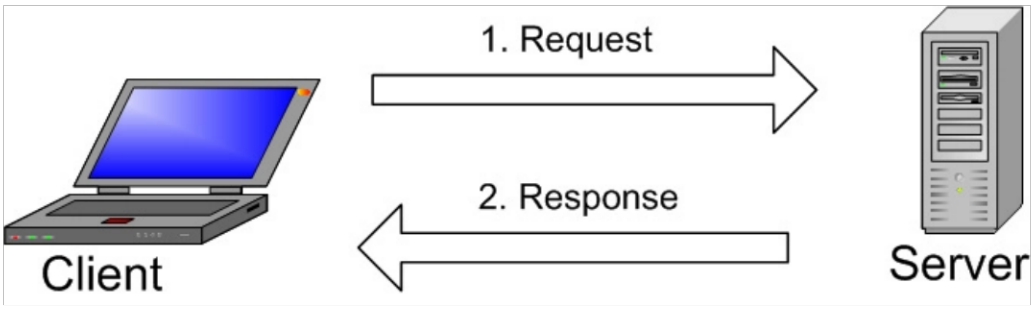


Gambar 2.9 Pertumbuhan Awal MySQL

MySQL sangat populer untuk aplikasi *hosting* Web karena kebanyakan fitur Web dioptimalkan seperti tipe data *Hyper Text Markup Language* (HTML), serta tersedia secara gratis. MySQL merupakan bagian dari arsitektur *Linux*, *Apache*, MySQL, PHP (LAMP), kombinasi *platform* yang sering digunakan untuk mengirim dan mendukung aplikasi Web canggih. MySQL menjalankan *database back-end* dari beberapa situs web terkenal, termasuk *Wikipedia*, *Google* dan *Facebook*, bukti stabilitas dan kekokohannya terlepas dari desentralisasi [16].

MySQL pada awalnya dimiliki oleh *Sun Microsystems*, ketika perusahaan ini dibeli oleh *Oracle Corp.* pada tahun 2010, MySQL merupakan bagian dari *package*. Meskipun secara teknis MySQL merupakan pesaing dari *Oracle DB*, *Oracle DB* biasanya digunakan oleh perusahaan besar, sedangkan MySQL digunakan oleh basis data yang lebih kecil dan lebih berorientasi pada web. MySQL berbeda dari produk *Oracle* karena berada di domain publik [16].

MySQL adalah *Relational Database Management System* (RDBMS) yang bersaing dengan yang lain seperti *Oracle DB* dan *Microsoft SQL Server*. MySQL disponsori oleh perusahaan *Swedia* MySQL *AB*, yang dimiliki oleh *Oracle Corp.* Namun, *source code* dari MySQL tersedia secara bebas, ini dikarenakan pada awalnya dikembangkan sebagai *freeware* [16].



Gambar 2.10 Struktur Client-Server

Gambar diatas menjelaskan struktur dasar dari struktur *client-server*, satu atau beberapa perangkat (*client*) terhubung ke *server* melalui jaringan. Setiap *client* dapat membuat permintaan dari *Graphical User Interface* (GUI) pada layar mereka, dan *server* akan menghasilkan output yang diinginkan, selama kedua ujungnya memahami instruksi. Tanpa terlalu teknis, proses utama yang terjadi di lingkungan MySQL adalah sama, yaitu [16]*:*

1. MySQL membuat *database* untuk menyimpan dan memanipulasi data, mendefinisikan hubungan setiap tabel.
2. *Client* dapat membuat permintaan dengan mengetik pernytaan *SQL* spesifik pada MySQL.
3. Aplikasi *server* akan merespon dengan informasi yang diminta dan itu akan muncul pada sisi *client*.

Semakin ringan dan *user-friendly* GUI-nya, semakin cepat dan mudah juga kegiatan manajemen datanya. Beberapa GUI MySQL yang paling populer adalah MySQL *WorkBench*, *SequelPro*, *DBVisualizer*, dan *Navicat DB Admin Tool*. Beberapa GUI MySQL tersebut gratis, sementara beberapa berisfat komersial, beberapa berjalan khusus untuk MacOS, dan beberapa kompatible dengan sistem operasi utama. *Client* harus memilih GUI tergantung pada kebutuhan mereka, untuk manajemen *database* web, termasuk situs *WordPress*, paling jelas menggunakan *phpMyAdmin* [16].

341 systems in ranking, December 2018

Rank Dec 2018	Rank Nov 2018	Rank Dec 2017	DBMS	Database Model	Score Dec 2018	Score Nov 2018	Score Dec 2017
1.	1.	1.	Oracle	Relational DBMS	1283.22	-17.89	-58.32
2.	2.	2.	MySQL	Relational DBMS	1161.25	+1.36	-156.82
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1040.34	-11.21	-132.14
4.	4.	4.	PostgreSQL	Relational DBMS	460.64	+20.39	+75.21
5.	5.	5.	MongoDB	Document store	378.62	+9.14	+47.85
6.	6.	6.	IBM Db2	Relational DBMS	180.75	+0.87	-8.83
7.	7.	↑ 8.	Redis	Key-value store	146.83	+2.66	+23.59
8.	8.	↑ 10.	Elasticsearch	Search engine	144.70	+1.24	+24.92
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	139.51	+1.08	+13.63
10.	10.	↑ 11.	SQLite	Relational DBMS	123.02	+0.31	+7.82

Gambar 2.11 *Score Model Database*

MySQL bukanlah satu-satunya RDBMS di pasar, tetapi itu adalah salah satu yang paling populer, berada pada posisi kedua setelah *Oracle Database* ketika dinilai menggunakan parameter kritis seperti jumlah sebutan dalam hasil pencarian, tenaga profesional pada *LinkedIn*, dan frekuensi diskusi teknis di forum internet. Berikut ini alasan kenapa banyak perusahaan teknologi raksasa yang bergantung pada *database* ini, yaitu [17]:

1. Fleksibel dan mudah untuk digunakan, kita dapat memodifikasi source code untuk memenuhi kebutuhan, dan tidak perlu membayar apa pun untuk melakukan ini, termasuk untuk meng-*upgrade* ke versi komersialnya. Proses pemasangan yang relatif sederhana, dan tidak perlu lebih dari 30 menit.
2. Memiliki performa yang tinggi, berbagai macam *server cluster* mendukung MySQL. Jika kita menyimpan data *e-commerce* yang besar atau melakukan kegiatan intelijen bisnis yang berat, MySQL dapat membantu dengan memperlancarnya dengan kecepatan optimal.
3. Mengikuti standar industri, industri telah menggunakan MySQL selama bertahun-tahun, yang berarti ada banyak sumber daya pengembang yang terampil. Pengguna MySQL dapat melakukan pengembangan yang cepat terhadap *software*.
4. Lebih aman, kita harus memperhatikan data kita, saat akan memilih software RDBMS yang tepat. Dengan *Privilege Access System* dan *User Account Management*, MySQL memberikan tingkat keamanan yang lebih tinggi. Verifikasi berbasis *host* dan enkripsi kata sandi keduanya tersedia pada MySQL.

2.5 *Rapid Application Development* (RAD)

Pada tahun 1980-an, di Dupont, Barry Boehm dan Tom Gilb mengembangkan metodologi yang disebut dengan *Rapid Iterative Production Prototyping* (RIPP). James Matin kemudian mengembangkan RIPP menjadi proses yang lebih formal yang kemudian dikenal sebagai *Rapid Application Development* (RAD). RAD memampatkan lengkah demi langkah pengembangan metode konvensional menjadi proses berulang. Dengan demikian pendekatan RAD mencakup pengembangan dan penyempurnaan model data, model proses, dan *prototype* secara paralel dengan menggunakan proses berulang. *User requirements* disempurnakan, merancang solusi, membuat *prototype* dari solusi, meninjau kembali *prototype*, menerima masukkan dari pengguna, kemudian proses diulang kembali [18].

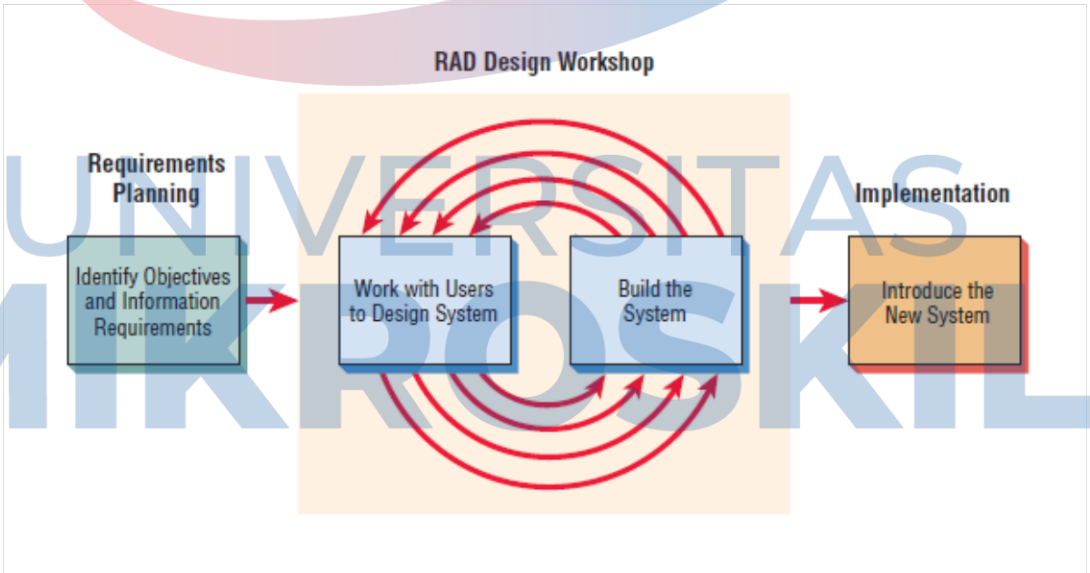
James Martin, dalam bukunya yang pertama kali menciptakan istilah tersebut, ia menulis "*Rapid Application Development* (RAD) adalah siklus hidup pengembangan yang dirancang untuk memberikan pengembangan yang jauh lebih cepat dan menghasilkan kualitas yang lebih tinggi daripada yang dicapai dengan siklus hidup tradisional" [18].

Rapid Application Development (RAD) adalah pendekatan berorientasi objek untuk pengembangan sistem yang mencakup metode pengembangan serta *tools* perangkat lunak [19]. Profesor Clifford Kettem Borough dari *Whitehead College*, *University of Redlands*, mendefinisikan *Rapid Application Development* sebagai "Sebuah pendekatan untuk membangun sistem komputer yang menggabungkan alat dan teknik *Computer Assisted Software Engineering (CASE)*, *prototype user-driven*, serta batas *project delivery time* yang ketat ke dalam formula yang ampuh, teruji, handal dalam kualitas dan produktivitas yang terbaik, RAD secara drastis meningkatkan kualitas sistem sekaligus mengurangi waktu yang dibutuhkan untuk membangunnya [18]. RAD merupakan teknik *team-based* yang mempercepat pengembangan sistem informasi dan menghasilkan sistem informasi yang fungsional. Seperti *Joint Application Development* (JAD), RAD menggunakan pendekatan kelompok tetapi berjalan lebih jauh. Produk jadi JAD adalah model persyaratan, sedangkan produk akhir RAD adalah sebuah sistem informasi baru. RAD adalah metodologi yang lengkap, dengan siklus hidup tiga fase yang paralel dengan fase

SDLC tradisional. Sebuah perusahaan menggunakan RAD untuk mengurangi biaya dan waktu pengembangan, dan meningkatkan probabilitas keberhasilan [20].

Tujuan utama dari semua pendekatan RAD adalah untuk mengurangi waktu dan biaya pengembangan dengan melibatkan pengguna dalam setiap fase pengembangan sistem. Karena ini adalah proses yang berkelanjutan, RAD memungkinkan tim pengembangan untuk membuat modifikasi yang diperlukan dengan cepat, seiring dengan perkembangan desain. RAD merupakan proses yang dinamis dan digerakkan oleh pengguna, untuk itu RAD sangat dibutuhkan ketika perusahaan menginginkan sistem informasi untuk mendukung fungsi bisnis baru. Dengan mendapatkan masukkan dari pengguna pada fase awal, RAD juga membantu tim pengembangan merancang sistem yang membutuhkan *user interface* yang sangat interaktif atau kompleks [20].

Ada tiga fase untuk RAD yang melibatkan pengguna dan analisis dalam penilaian, desain, dan implementasi. RAD melibatkan pengguna di setiap bagian dari upaya pengembangan dengan partisipasi intens dalam bagian desain bisnis [19].



Gambar 2.12 Fase *Rapid Application Development* (RAD)

1. Fase Rencana Kebutuhan Sistem (*Requirement Planning)*

Pada fase perencanaan persyaratan, pengguna dan analis akan bertemu untuk mengidentifikasi tujuan dari pembangunan aplikasi atau sistem dan untuk mengidentifikasi persyaratan informasi yang diperoleh dari tujuan tersebut. Orientasi

dalam fase ini adalah menuju penyelesaian masalah bisnis, meskipun teknologi dan sistem informasi dapat mendorong beberapa solusi yang diusulkan, fokusnya tetap pada pencapaian tujuan bisnis [19]. Fase perencanaan persyaratan berkahir ketika tim menyetujui persyaratan-persyaratan yang diajukan dan mendapatkan izin dari manajemen untuk melanjutkannya [20].

2. Fase Desain/Rancangan Pengguna

Pada fase desain dilakukan perancangan dan perbaikan, pengguna berinteraksi dengan sistem analis untuk mengembangkan model dan *prototype* pada seluruh proses sistem, *input*, dan juga *output* [20]. Selama proses perancangan, pengguna akan merespon *prototype* yang dibuat dan sistem analis akan memperbaiki modul yang dirancang berdasarkan tanggapan yang diberikan oleh pengguna [19]. Fase desain adalah proses interaktif dan berkelanjutan yang memungkinkan pengguan untuk memahami, memodifikasi, dan akhirnya menyetujui model kerja sistem yang memenuhi kebutuhan mereka [20].

3. Fase Implementasi (*Implementation)*

Pada fase ini sistem yang telah dirancang aspek bisnis dan non-teknis sistem, serta telah disepakati dan sistem atau aplikasi dibangun dan disempurnakan, sistem baru atau bagian dari sistem akan diuji dan diperkenalkan ke organisasi. Karena RAD dapat digunakan untuk membuat aplikasi baru yang tidak memiliki sistem lama, sehingga tidak perlu menjalankan sistem lama dan baru secara paralel sebelum dilakukan implementasi [19].

2.6 Teknik Pengembangan Sistem Informasi

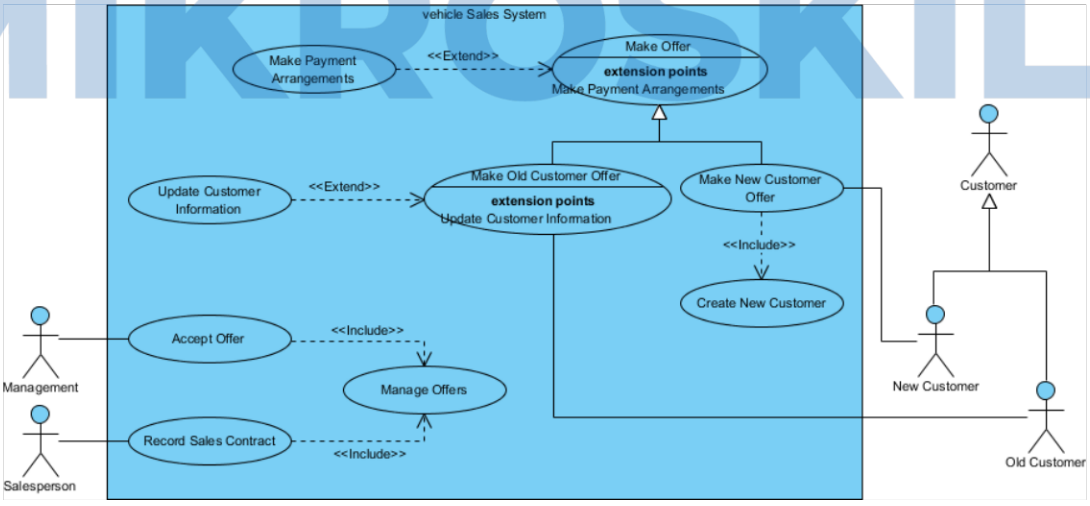
2.6.1 Model *Use Case*

Use case diagram adalah bentuk utama dari persyaratan sistem untuk program perangkat lunak baru yang kurang berkembang. *Use case diagram* menentukan perilaku yang diharapkan (*what*), dan bukan metode yang tepat untuk mewujudkannya (*how*). *Use case diagram* yang telah ditentukan dapat dilambangkan dengan representasi tekstual dan visual. Konsep utama pemodelan *use case diagram* adalah membantu untuk merancang sistem dari perspektif pengguna akhir. Cara ini

merupakan teknik yang efektif untuk mengkomunikasikan perilaku sistem dalam perspektif pengguna dengan menentukan semua perilaku sistem yang terlihat dari sisi eksternal. *Diagram use case* biasanya sederhana, tidak menunjukkan detail kasus penggunaan, urutan, langkah-langkah yang dilakukan untuk mencapai tujuan dari setiap kasus penggunaan, melainkan hanya merangkum beberapa hubungan antara kasus penggunaan, aktor, dan sistem [21].

Use case modeling sering dikaitkan dengan UML, padahal *use case modeling* sudah diperkenalkan terlebih dahulu sebelum UML. Sejarah singkatnya, yaitu pada tahun 1986, Ivar Jacobson pertama kali merumuskan teknik pemodelan tekstual dan visual untuk menentukan *use case* dan kemudian pada tahun 1992, rekan penulisnya pada buku berjudul *Object-Oriented Software Engineering - A Use Case Driven Approach* membantu mempopulerkan teknik untuk menangkap persyaratan fungsional, terutama dalam pengembangan perangkat lunak [21].

Tujuan *use case diagram* adalah untuk menangkap aspek dinamis dari suatu sistem, mengumpulkan persyaratan sistem termasuk pengaruh internal dan eksternal. Persyaratan ini sebagian besar merupakan persyaratan desain. Oleh karena itu, ketika suatu sistem dianalisis untuk mengumpulkan fungsionalitasnya, *use case* disiapkan dan aktor diidentifikasi. Secara singkat, tujuan diagram *use case* dapat dikatakan untuk mengumpulkan persyaratan suatu sistem, mendapatkan tampilan luar dari suatu sistem, mengidentifikasi faktor-faktor eksternal dan internal yang mempengaruhi sistem dan menunjukkan interaksi antara persyaratan dengan aktor [22].

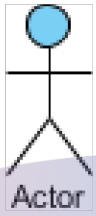


Gambar 2.13 *Use Case Diagram*

2.6.1.1 Deskripsi Notasi

Pada *use case diagram*, terdapat beberapa notasi yang dilambangkan dalam bentuk sebagai berikut [21]:

1. *Actor*



Gambar 2.14 Aktor

Actor merupakan orang yang berinteraksi dengan *use case* (fungsi sistem) dan dinamai dengan kata benda. *Actor* berperan dalam bisnis, memiliki konsep yang mirip dengan pengguna, tetapi pengguna dapat memiliki peran yang berbeda. Notasi ini dapat memicu *use case* karena memiliki tanggung jawab terhadap sistem (*input)* dan memilki harapan terhadap sistem (*output).*

2. *Use Case*

Gambar 2.15 *Use Case*

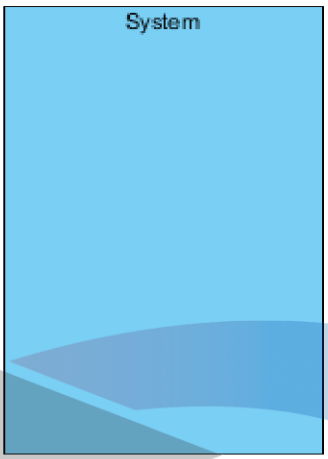
Use case melambangkan fungsi sistem atau proses dan dinamai dengan kata kerja yang diikuti dengan kata benda. Setiap *Actor* harus dikaitkan dengan *use case*, sementara beberapa *use case* mungkin tidak terkait dengan *actor*.

3. *Communication Link*

Gambar 2.16 *Communication Link*

Garis seperti diatas digunakan untuk menghubungkan *actor* dengan *use case.* Hal ini menunjukkan bahwa *actor* dan *use case* memiliki hubungan dengan keterangan tertentu.

4. *Boundary of System*



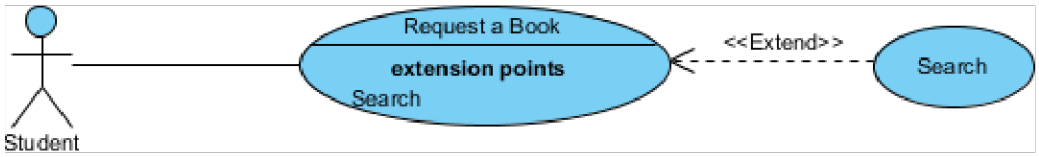
Gambar 2.17 *Boundary of System*

Batasan sistem biasanya merupakan seluruh sistem seperti yang didefinisikan dalam dokumen persyaratan. Untuk sistem yang besar dan kompleks, setiap modul dapat menjadi batasan sistem. Misalnya, pada sistem ERP untuk suatu organisasi, masing-masing modul seperti personil, penggajian, akuntansi dan lainnya dapat menjadi batasan sistem untuk kasus penggunaan khusus di masing-masing fungsi bisnis. Seluruh sistem dapat menjangkau semua modul yang menggambarkan batas sistem secara keseluruhan [21].

2.6.1.2 Deskripsi Relasi

Jenis-jenis relasi yang terdapat di *use case diagram*, yaitu [21]:

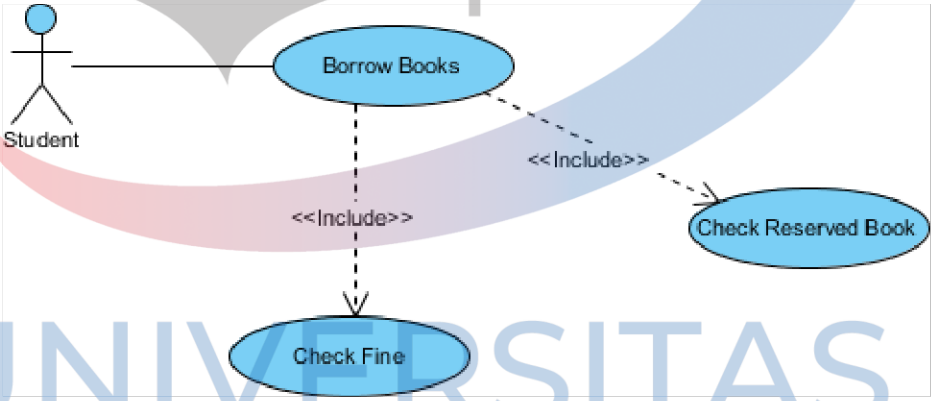
1. *Extends*



Gambar 2.18 Relasi *Extends*

Relasi *extend* menunjukkan bahwa *use case* "*Invalid Password*" dapat mencakup perilaku yang ditentukan oleh *use case* "*Login Account*". Relasi ini menunjukkan suatu hubungan yang diperluas.

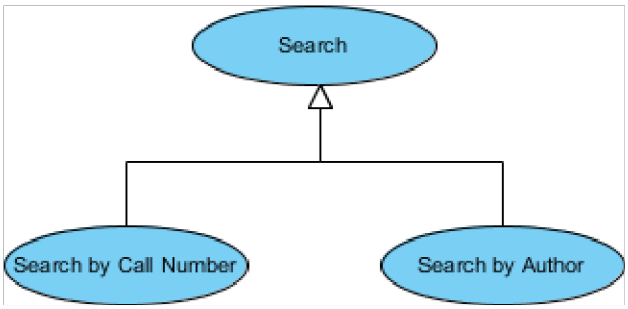
2. *Include*



Gambar 2.19 Relasi *Include*

Relasi *include* terjadi ketika *use case* digambarkan menggunakan fungsionalitas *use case* lain. Sebuah *use case* mencakup fungsi yang dijelaskan dalam *use case* lain sebagai bagian dari alur proses bisnisnya.

3. *Generalization*



Gambar 2.20 Relasi *Generalization*

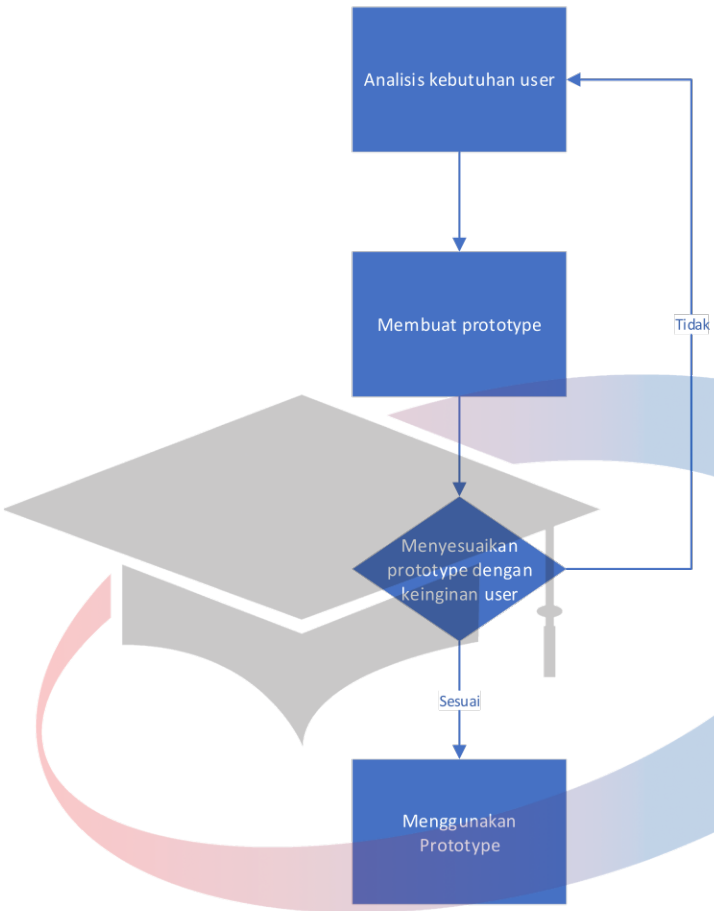
Generalization menggambarkan hubungan induk-anak antara *use case*. *Use case* anak merupakan tambahan dari *use case* induk. [21]

2.6.2 *Prototyping*

Prototyping merupakan teknik pengembangan sistem yang menggunakan *prototype* untuk menggambarkan sistem, sehingga pengguna atau pemilik sistem mempunyai gambaran pengembangan sistem yang akan dilakukannya. Teknik ini sering digunakan apabila pemilik sistem tidak terlalu menguasai sistem yang akan dikembangkannya, sehingga dia memerlukan gambaran dari sistem yang akan dikembangkannya tersebut. Dengan teknik *prototyping*, pengembang dapat membuat *prototype* terlebih dahulu sebelum mengembangkan sistem yang sebenarnya. Dalam pengembangan sistem, *prototype* sering diwujudkan dalam bentuk *user interface* program aplikasi dan contoh-contoh *reporting* yang akan dihasilkan, sehingga dengan demikian pengguna sistem akan mempunyai gambaran tentang sistem yang akan digunakannya nanti. McLeod dan Schell mendefinisikan dua tipe dari *prototype*, yaitu [23]:

1. *Evolutionary Prototype*

   Evolutionary Prototype yaitu, *prototype* yang secara terus menerus dikembangkan hingga *prototype* tersebut memenuhi fungsi dan prosedur yang dibutuhkan oleh sistem. Berikut ini adalah gambaran dari tahapan *evolutionary prototype*:



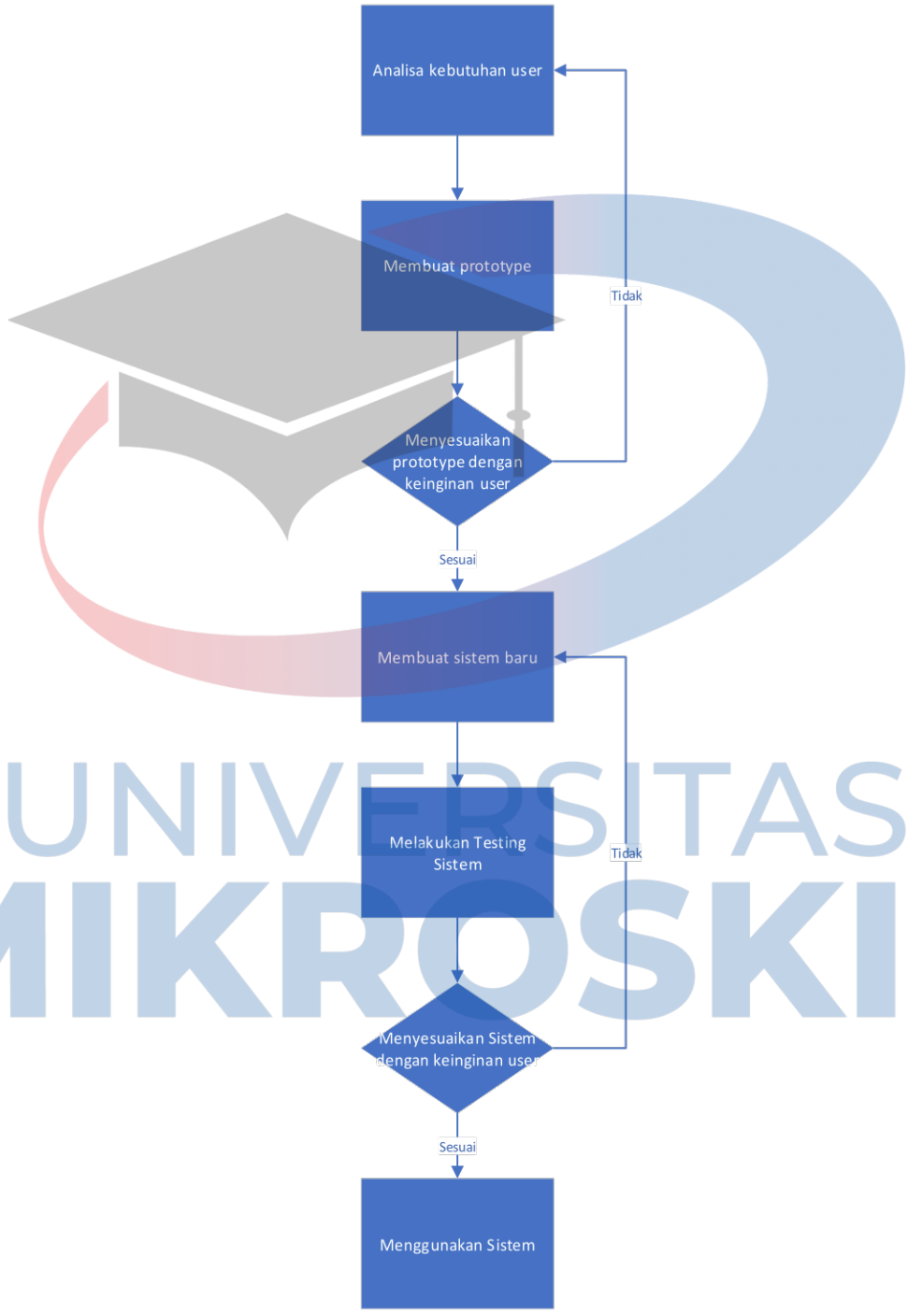
Gambar 2.21 Tahapan *Evolutionary Prototype*

a. Analisis kebutuhan *user*, pengembang dan pengguna atau pemilik sistem melakukan diskusi dimana pengguna atau pemilik sistem menjelaskan kepada pengembang tentang kebutuhan sistem yang mereka inginkan.
b. Membuat *prototype*, pengembang membuat *prototype* dari sistem yang telah dijelaskan oleh pengguna atau pemilik sistem.
c. Menyesuaikan *prototype* dengan keinginan *user*, pengemang menanyakan kepada pengguna atau pemilik sistem tentang *prototype* yang sudah dibuat, apakah sesuai dengan kebutuhan sistem atau tidak.
d. Menggunakan *prototype*, sistem mulai dikembangkan dengan *prototype* yang sudah dibuat.

2. *Requirement Prototype*

Requirement Prototype merupakan *prototype* yang dibuat oleh pengembang dengan mendefinisikan fungsi dan prosedur sistem dimana pengguna atau pemilik

sistem tidak bisa mendefinisikan sistem tersebut. Berikut ini langkah-langkah dari *requirement prototype:*



Gambar 2.22 Tahapan *Requirement Prototyping*

a. Analisis kebutuhan *user*, pengembang dan pengguna atau pemilik sistem melakukan diskusi dimana pengguna atau pemilik sistem menjelaskan kepada pengembang tentang kebutuhan sistem yang mereka inginkan.
b. Membuat *prototype*, pengembang membuat *prototype* dari sistem yang telah dijelaskan oleh pengguna atau pemilik sistem.
c. Menyesuaikan *prototype* dengan keinginan *user*, pengembang menanyakan kepada pengguna atau pemilik sistem tentang *prototype* yang sudah dibuat, apakah sesuai dengan kebutuhan sistem atau tidak.
d. Membuat sistem baru, pengembang menggunakan *prototype* yang sudah dibuat untuk membuat sistem baru.
e. Melakukan *testing* sistem, pengguna atau pemilik sistem melakukan uji coba terhadap sistem yang dikembangkan.
f. Menyesuaikan dengan keinginan *user*, sistem disesuaikan dengan keinginan *user* dan kebutuhan sistem, jika sudah sesuai sistem siap digunakan.
g. Menggunakan sistem.

Kelebihan dari teknik pengembangan *prototyping* adalah menghemat waktu dan biaya pengembangan. Pengguna atau pmilik sistem ikut terlibat dalam pengembangan, sehingga kemungkinan-kemungkinan terjadinya kesalahpahaman dalam sistem bisa diminimalisir. Implementasi akan menjadi lebih mudah, karena pengguna atau pemilik sistem sudah mempunyai gambaran tentang sistem. Kualitas sistem yang dihasilkan lebih baik. Dan memungkinkan tim pengembangan sistem untuk memprediksi dan memperkirakan pengembangan-pengembangan sistem selanjutnya.

Kelemahan dari teknik pengembangan *prototyping* adalah, pengguna ataupun pemilik sistem bisa terus menerus menambahkan kompleksitas sistem sehingga sistem menjadi sangat kompleks, hal ini bisa menyebabkan pengembangan meninggalkan pekerjaannya sehingga sistem yang dikerjakan tidak akan pernah selesai [23].

2.7 Layanan

Layanan merupakan suatu kegiatan atau manfaat yang ditawarkan suatu pihak kepada pihak lain, yang pada dasarnya tidak berwujud dan tidak mengakibatkan kepemilikan apa pun. Produksi layanan bisa berhubungan dengan produk fisik maupun

tidak. Penawaran suatu perusahaan kepada pasar biasanya mencakup beberapa jenis layanan. Komponen layanan ini dapat merupakan bagian kecil atau bagian utama dari keseluruhan penawaran tersebut. Penawaran bisa saja murni berupa barang pada satu sisi dan layanan murni pada sisi lainnya. Sebenarnya sulit untuk membedakan secara tegas antara barang dan layanan yang sering kali disertai dengan layanan-layanan tertentu (misalnya instalasi, pemberian garansi, dan reparasi). Sebaliknya pembelian layanan sering kali juga melibatkan barang-barang yang melengkapinya (misalnya makanan di restoran, telepon dalam layanan telekomunikasi) [24].

2.7.1 Hotel Hewan

Hotel hewan peliharaan yang menargetkan pemilik hewan peliharaan yang sering bepergian atau tidak bisa merawat hewan peliharaan mereka dalam jangka waktu tertentu dengan alasan apapun. Pemilik hewan peliharaan dapat mengirim hewan peliharaan mereka ke penyedia layanan hotel hewan peliharaan, sehingga pemilik dapat memastikan hewan peliharaan kesayangan mereka dalam perawatan yang baik dan dibawah pengawasan, sementara mereka sibuk dan jauh dari rumah. Sebagian besar layanan hewan peliharaan kesayangan perusahaan tidak hanya menyediakan layanan penginapan semalam, tetapi juga penitipan siang hari untuk kebutuhan pemilik hewan peliharaan yang berbeda. Fasilitas dasar seperti air bersih, makanan, waktu bermain harian dan tempat tidur adalah unsur utama yang disediakan untuk hewan peliharaan pada layanan hotel hewan [25].

Hewan peliharaan berada dibawah pengawasan staf, mereka mendapatkan waktu untuk menikmati aktivitas *indoor* dan *outdoor*. Penyedia layanan hewan peliharaan tahu bahwa pemilik hewan peliharaan ingin memperhatikan situasi hewan peliharaan mereka sewaktu hewan peliharaan tinggal di hotel. Sehingga banyak perusahaan memasang *webcam* pemantauan *online* untuk pelanggan mereka, untuk memantau hewan peliharaan mereka kapan saja dan dimana saja. Jika pemilik hewan peliharaan menginginkan detail lebih lanjut dari kondisi hewan peliharaan, mereka dapat menerima laporan, foto melalui email, pesan atau panggilan telepon, pemilik hewan peliharaan dapat mengakses informasi lengkap tentang hewan peliharaan mereka. Dalam kasus darurat, seorang dokter hewan dapat dipanggil jika diperlukan selama *boarding* atau penitipan siang hari. Setiap hewan peliharaan di hotel hewan

memiliki peraturan dan aturan tersendiri, ras tertentu mungkin tidak diterima untuk layanan ini [25].

2.7.2 Salon Hewan

Sebagian besar hewan peliharaan berbulu dan membutuhkan pembersihan dan perawatan yang rutin, agar tetap dalam kondisi yang higienis. *Grooming* adalah hal yang penting untuk hewan peliharaan, selain membuat mereka nyaman dan bersih, tetapi juga untuk menjaga kehidupan sehari-hari hewan peliharaan tetap sehat. Merawat hewan peliharaan dapat memakan waktu dan rumit, karena pemilik hewan peliharaan harus mengurus bagian-bagian yang berbeda dari tubuh hewan pemeliharaan dan memastikan hewan peliharaan tetap terkendali. Beberapa pemilik hewan peliharaan membawa hewan peliharaan mereka ke penjaga *professional* dan membiarkan hewan peliharaannya untuk mendapatkan perawatan. Layanan yang diberikan berupa mandi, memotong bulu, menyikat gigi, memotong kuku, membersihkan telinga, membersihkan noda air mata [25].

Pemilik hewan peliharaan akan memutuskan layanan mana yang dibutuhkan oleh hewan peliharaan mereka, dengan saran dari *groomer*. Pemilik hewan peliharaan dapat berdiskusi dengan *groomer* untuk mempersonalisasikan gaya unik untuk hewan peliharaan. Secara umum, layanan salon hewan peliharaan disediakan di toko, pemilik hewan peliharaan harus memiliki janji dangan penyedia jasa, dengan memesan paket perawatan yang disediakan toko [25].

2.7.3 Klinik Hewan

Struktur khas layanan kedokteran hewan terbentuk dari rumah sakit hewan, klinik hewan, dan perangkat bergerak layanan kedokteran hewan. Terdapat layanan medis berbeda yang ditawarkan oleh dokter hewan yang berbeda [25].

Klinik hewan menjadi pilihan paling populer bagi pemilik hewan peliharaan untuk mendapatkan perawatan medis bagi hewan peliharaan mereka yang sakit atau membutuhkan pemeriksaan kesehatan preventif. Sebagian besar klinik akan menyediakan perawatan pencegahan komprehensif seperti, vaksinasi, pengendalian parasit, perawatan, perawatan gigi, layanan konseling nutrisi dan perilaku, serta

pemeriksaan tubuh. Selain itu, *microchipping* dan *euthanasia* disediakan oleh semua klinik hewan *reguler* [25].

Dengan berbagai fasilitas medis dan staf medis, biasanya klinik dokter hewan akan banyak membantu dalam kasus-kasus seperti, hewan peliharaan yang membutuhkan perawatan medis khusus, yakni kardiologi, operasi onkologis, prosedur invasive minimal, dan lain-lain. Spesialis dokter hewan dan rumah sakit hewan adalah pendukung dari klinik hewan, ini karena rumah sakit hewan memiliki peralatan yang lengkap dan dokter professional yang berguna untuk menawarkan beragam perawatan medis untuk setiap jenis penyakit hewan peliharaan [25].