# BAB 2
# LANDASAN TEORI

Pada bab ini akan menjelaskan tentang teori yang berhubungan dengan perancangan sistem sebagai teori pendukung pada saat melakukan perancangan. Selain itu, pada bab ini menjelaskan tentang spesifikasi dari komponen perangkat keras dan sistem yang digunakan pada saat perancangan sistem.

## 2.1 Dinas Komunikasi dan Informatika

Kementerian Komunikasi dan Informatika, sebelumnya bernama "Departemen Penerangan" (1945-1999), "Kementerian Negara Komunikasi dan Informasi" (2001-2005), dan Departemen Komunikasi dan Informatika (Depkominfo) (2005-2009).

Setelah proklamasi kemerdekaan dibentuk Lembaga Penerangan yang secara fungsional menjalankan kebijakan, pola dan pedoman penerangan dengan tujuan membela dan mempertahankan kemerdekaan, mengajak rakyat agar turut serta mempertahankan dan mengisi kemerdekaan serta memperkenalkan Republik Indonesia di dan ke luar negeri. Selama periode 1959-1965, sesuai Haluan Pembangunan Nasional sebagai ketetapan MPRS, Departemen Penerangan dibentuk untuk menyelenggarakan penerangan melalui media penerangan antara lain radio, film, *toestel* dan foto, percetakan, kendaraan, mesin stensil, dan mesin ketik.

Tugas & Fungsi Kementerian Komunikasi dan Informatika adalah sebagai berikut:

Tugas : Kementerian Komunikasi dan Informatika mempunyai tugas menyelenggarakan urusan pemerintahan di bidang komunikasi dan informatika untuk membantu Presiden dalam menyelenggarakan pemerintahan negara.

Fungsi :

Perumusan dan penetapan kebijakan di bidang pengelolaan sumber daya dan perangkat pos dan informatika, penyelenggaraan pos dan informatika, penatakelolaan aplikasi informatika, pengelolaan informasi dan komunikasi publik;

Pelaksanaan kebijakan di bidang pengelolaan sumber daya dan perangkat pos dan informatika, penyelenggaraan pos dan informatika, penatakelolaan aplikasi informatika, pengelolaan informasi dan komunikasi publik;

Pelaksanaan bimbingan teknis dan supervisi atas pelaksanaan pengelolaan sumber daya dan perangkat pos dan informatika, penyelenggaraan pos dan informatika, penatakelolaan aplikasi informatika, pengelolaan informasi dan komunikasi publik:

1. Pelaksanaan penelitian dan pengembangan sumber daya manusia di bidang komunikasi dan informatika;
2. Pelaksanaan dukungan yang bersifat substantif kepada seluruh unsur organisasi di lingkungan Kementerian Komunikasi dan Informatika;
3. Pembinaan dan pemberian dukungan administrasi di lingkungan Kementerian Komunikasi dan Informatika;
4. Pengelolaan barang milik/kekayaan negara yang menjadi tanggung jawab Kementerian Komunikasi dan Informatika;
5. Pengawasan atas pelaksanaan tugas di lingkungan Kementerian Komunikasi dan Informatika;

## 2.2 *Internet of Things*

*Internet of Things* (IoT) adalah sebuah konsep sekenario dimana suatu objek yang menggunakan pemanfaatan internet yang tersambung dengan jaringan komputer secara terus menerus. adapun kemampuan seperti berbagi data kendali, dan termasuk juga pada benda dunia nyata. Bahan pangan, elektronik, peralatan apa saja, koleksi , termasuk benda hidup, yang semuanya tersambung ke jaringan lokal dan global melalui sensor tertanam dan selalu aktif [1].

Interaksi antara manusia dengan manusia sudah sangat biasa sejak zaman dahulu kala. Interaksi antara manusia dengan mesin sudah biasa pula, sejak adanya penemuan teknologi seperti komputer atau gadget devices lainnya. Namun, Menurut analisa McKinsey Global Institute, internet of things adalah

sebuah teknologi yang memungkinkan kita untuk menghubungkan mesin, peralatan, dan benda fisik lainnya dengan sensor jaringan dan aktuator untuk memperoleh data dan mengelola kinerjanya sendiri, sehingga memungkinkan mesin untuk berkolaborasi dan bahkan bertindak berdasarkan informasi baru yang diperoleh secara independen. Sebuah publikasi mengenai Internet of things in 2020 menjelaskan bahwa internet of things adalah suatu keadaan ketika benda memiliki identitas, bisa beroperasi secara intelijen, dan bisa berkomunikasi dengan sosial, lingkungan, dan pengguna.

Teknologi internet of things sangat luar biasa. Jika sudah direalisasikan, teknologi ini tentu akan sangat memudahkan pekerjaan manusia. Manusia tidak akan perlu lagi mengatur mesin saat menggunakannya, tetapi mesin tersebut akan dapat mengatur dirinya sendiri dan berinteraksi dengan mesin lain yang dapat berkolaborasi dengannya. Hal ini membuat mesin-mesin tersebut dapat bekerja sendiri dan manusia dapat menikmati hasil kerja mesin-mesin tersebut tanpa harus repot-repot mengatur mereka.

Cara kerja dari *internet of things* setiap benda harus memiliki sebuah *IP Address. IP Address* adalah sebuah identitas dalam jaringan yang membuat benda tersebut bisa diperintahkan dari benda lain dalam jaringan yang sama. Selanjutnya, IP address dalam benda-benda tersebut akan dikoneksikan ke jaringan internet. Saat ini, koneksi internet sudah sangat mudah kita dapatkan. Dengan demikian, kita dapat memantau benda tersebut bahkan memberi perintah kepada benda tersebut. Setelah sebuah benda memiliki *IP address* dan terkoneksi dengan internet, pada benda tersebut juga dipasang sebuah sensor. Sensor pada benda memungkinkan benda tersebut memperoleh informasi yang dibutuhkan. Setelah memperoleh informasi, benda tersebut dapat mengolah informasi itu sendiri, bahkan berkomunikasi dengan benda-benda lain yang memiliki *IP address* dan terkoneksi dengan internet juga. Akan terjadi pertukaran informasi dalam komunikasi antara benda-benda tersebut. Setelah pengolahan informasi selesai, benda tersebut dapat bekerja dengan sendirinya, atau bahkan memerintahkan benda lain juga untuk ikut bekerja.

Dapat kita simpulkan bahwa internet of things membuat suatu koneksi antara mesin dengan mesin, sehingga mesin-mesin tersebut dapat berinteraksi dan bekerja secara independen sesuai dengan data yang diperoleh dan diolahnya secara mandiri. Tujuannya adalah untuk membuat manusia berinteraksi dengan benda dengan lebih mudah, bahkan supaya benda juga bisa berkomunikasi dengan benda lainnya. Dengan begitu manusia dalam *internet of things* akan bertindak sebagai pengguna yang akan dilayani oleh benda-benda elektronik disekitarnya. Pengaruh ekonomi dari industri internet of things menempati posisi ketiga terbesar setelah mobile internet dan *automation o f knowledge work*. Hal ini membuktikan bahwa suatu hari nanti teknologi *internet of things* benar-benar akan berkembang dan menjadi tren di dunia.

### 2.3 Jaringan Internet

Internet adalah jaringan atau sistem pada jaringan komputer yang saling berhubungan berbagai komputer untuk dapat berbagi sumber daya, komunikasi dan akses informasi.[20] dengan menggunakan Sistem *Global Transmission Control Protocol* / *Internet Protocol Suite* (TCP/IP) sebagai protokol pertukaran paket atau data (*packet switching communication protocol*) untuk melayani miliaran pengguna di seluruh dunia. Internet juga biasa dikenal sebagai *interconneted-networking* (singkatan dari *Internet*). Internet berasal dari bahasa latin, yaitu “Inter” yang memiliki arti “Antara”. Jadi, apabila digabungkan kata per kata Internet adalah jaringan antara atau penghubung [2].

Internet dapat diartikan sebagai jaringan komputer luas dan besar yang mendunia, yaitu menghubungkan pemakai komputer dari suatu negara ke negara lain di seluruh dunia, dimana di dalamnya terdapat berbagai sumber daya informasi dari mulai yang statis hingga dinamis dan interaktif. Dalam komunikasi ini dapat terjadi perpindahan data ataupun berbagi sumber daya secara terorganisasi di seluruh dunia melalui telepon atau satelit. Dalam skala luas.

### 2.4 Aplikasi Pintar (*Smart Application)*

Ini pada dasarnya aplikasi adalah suatu program berbentuk perangkat lunak yang berjalan pada suatu sistem tertentu berguna sebagai hubungan antar muka user pengguna yamg berfungsi untuk membantu berbagai kegiatan.

Dengan perkembangannya aplikasi saat berkembang dapat disebut sebagai aplikasi pintar (*Smart Apps)* dengan semakin inovatif pengembangan nya dengan konsep pengumpulan sejumlah data dari sensor ataupun lain nya, menggunakan algoritma pembelajaran mesin dan analisis prediktif untuk membuat informasi dapat diolah dan menjadi aplikasi yang canggih selain dari pengolahan informasi memungkinkan juga mengendalikan atau mengorntrol suatu benda yang terhubung dengan aplikasi pintar tersebut. berdasarkan penggolongannya aplikasi dapat disebut sebagai aplikasi pintar apabila memiliki unsur berikut :

1. Intelligent aplikasi pintar ini menggunakan atau mengolah data analytics, sebagai kemampuan agar aplikasi ini dapat belajar dan semakin pintar dengan layanan artificial intelligent untuk memberikan rekomendasi dan prediksi sesuai dengan fungsi dan kebutuhannya.
2. Contextual aplikasi pintar jenis ini menggunakan atau mengolah data khusus atau memanfaatkan data pribadi, sensor dan lokasi sebagai sumber daya datanya menjadikan aplikasi ini dapat mengolah dari informasi yang diberikan sensor dan memberikan keluaran sesuai dengan apa yang dibutuhkan penggunanya.
3. Proactive aplikasi pintar jenis ini memanfaatkan notifikasi push, chat bot dan layanan pesan untuk berinteraksi secara proaktif dengan pengguna, dengan kecerdasan buatan yang disematkan berdasarkan informasi yang ditanamkan pada aplikasi tersebut.

### 2.5 Java Script

*JavaScript* adalah bahasa pemrograman web yang bersifat *Client Side* Programming *Language*. *Client Side Programming Language* adalah tipe bahasa pemrograman yang pemrosesannya dilakukan oleh client [1]. Aplikasi client yang dimaksud merujuk kepada web browser seperti Google Chrome dan Mozilla

Firefox.Bahasa pemrograman Client Side berbeda dengan bahasa pemrograman Server Side seperti PHP, dimana untuk server side seluruh kode program dijalankan di sisi server.Untuk menjalankan JavaScript, kita hanya membutuhkan aplikasi text editor dan web browser. *JavaScript* memiliki fitur: *high-level programming language, client-side, loosely tiped* dan berorientasi objek.

*Java Script* pada awal perkembangannya berfungsi untuk membuat interaksi antara user dengan situs web menjadi lebih cepat tanpa harus menunggu pemrosesan di web server. Sebelum javascript, setiap interaksi dari user harus diproses oleh web server

### 2.6 React JS

*React JS* adalah sebuah library JavaScript yang di buat oleh facebook. React bukanlah sebuah framework MVC. React adalah library yang bersifat composable user interface, yang artinya kita dapat membuat berbagai UI yang bisa kita bagi menjadi beberapa komponen. React memungkinkan pengembang untuk membuat aplikasi web besar yang dapat mengubah data, tanpa memuat ulang lamannya. Tujuan utama dari React adalah menjadi cepat, terukur, dan sederhana [3].

*React* hanya berfungsi pada antarmuka pengguna dalam aplikasi. React bisa juga kamu kombinasikan dengan *library* atau *framework JavaScript* lainnya, seperti Angular JS di MVC.

### 2.7 ESP32 CAM

ESP32 adalah rangkaian sistem hemat biaya dan rendah daya pada mikrokontroler chip dengan Wi-Fi terintegrasi dan Bluetooth mode ganda. Seri ESP32 menggunakan mikroprosesor Tensilica Xtensa LX6 dalam variasi dual-core dan single-core dan mencakup sakelar antena internal, balun RF, penguat daya, penguat penerima kebisingan rendah, filter, dan modul manajemen daya. ESP32 dibuat dan dikembangkan oleh Espressif Systems [4]


**Gambar 0.1 ESP32 CAM**

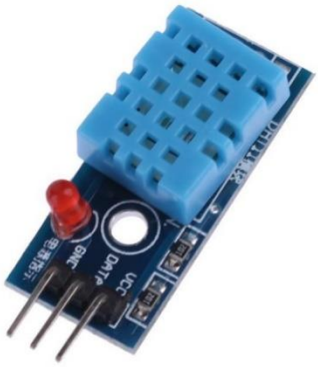
### 2.8 Modul Wifi ESP8266

ESP8266 merupakan modul wifi yang berfungsi sebagai perangkat tambahan mikrokontroler seperti Arduino agar dapat terhubung langsung dengan wifi dan membuat koneksi TCP/IP.

Modul ini membutuhkan daya sekitar 3.3v dengan memiliki tiga mode wifi yaitu Station, Access Point dan Both (Keduanya). Modul ini juga dilengkapi dengan prosesor, memori dan GPIO dimana jumlah pin bergantung dengan jenis ESP8266 yang kita gunakan. Sehingga modul ini bisa berdiri sendiri tanpa menggunakan mikrokontroler apapun karena sudah memiliki perlengkapan layaknya mikrokontroler.

### 2.9 Modul DHT11

DHT11 adalah salah satu sensor yang dapat mengukur dua parameter lingkungan sekaligus, yakni suhu dan kelembaban udara (humidity). Dalam sensor ini terdapat sebuah thermistor tipe NTC (Negative Temperature Coefficient) untuk mengukur suhu, sebuah sensor kelembaban tipe resisitif dan sebuah mikrokontroller 8-bit yang mengolah kedua sensor tersebut dan mengirim hasilnya ke pin output dengan format single-wire bi-directional (kabel tunggal dua arah). Jadi walaupun kelihatannya kecil, DHT11 ini ternyata melakukan fungsi yang cukup kompleks. Kita tinggal ambil outputnya aja, untuk kemudian dimasukkan ke sistem kita.




**Gambar 0.2 DHT 11**

## 2.10 Mikrokontroller

Mikrokontroler adalah sebuah system komputer yang seluruh atau sebagian besar elemennya dikemas dalam satu chip IC, sehingga sering disebut single chip microcomputer. Mikrokontroler merupakan system computer yang mempunyai satu atau beberapa tugas yang sangat spesifik [5]. Elemen mikrokontroler tersebut diantaranya adalah

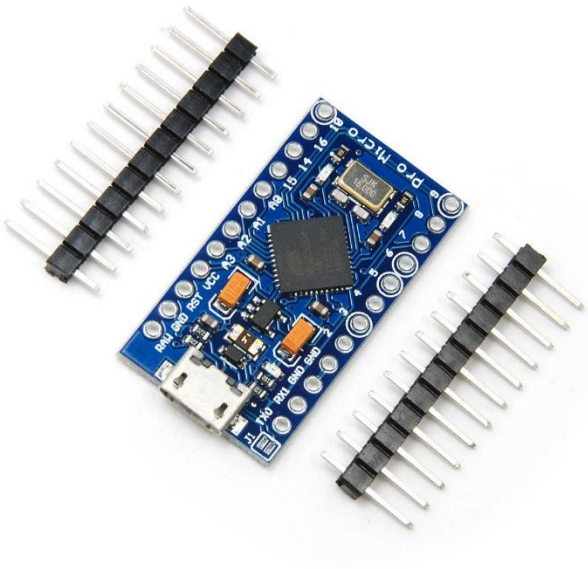
1. Pemproses (processor)
2. Memori
3. Input dan Output

Microcontroller telah banyak digunakan di industri, walaupun penggunaannya masih kurang dibandingkan dengan penggunaan Programable Logic Control (PLC), tetapi microcontroller memiliki beberapa keuntungan dibandingkan dengan PLC. Ukuran microcontroller lebih kecil dibandingkan dengan suatu modul PLC sehingga peletakannya dapat lebih flexible. Microcontroller telah banyak digunakan pada berbagai macam peralatan rumah tangga seperti mesin cuci. Sebagai pengendali sederhana, microcontroller telah banyak digunakan dalam dunia medik, pengaturan lalu% lintas, dan masih banyak

lagi. Contoh alat ini diantaranya adalah komputer yang digunakan pada mobil untuk mengatur kestabilan mesin, alat untuk pengatur lampu lalu lintas

## 2.11 Mikrokontroler Arduino Pro Micro

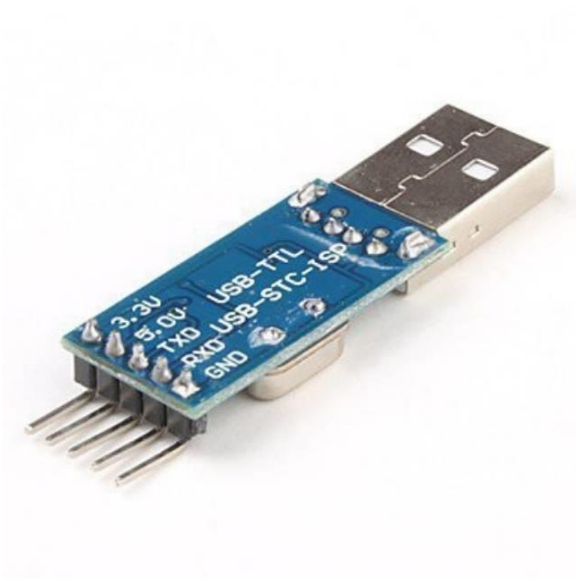
Arduino Pro Micro adalah jenis arduino yang berukuran kecil, ukuranya hampir sama seperti Arduino Pro Mini. Secara fungsi dan kegunaan tidak ada yang berbeda dengan jenis Arduino yang lainnya. Yang membedakan yaitu pada USB downloader menggunakan jenis Micro USB atau yang biasa digunakan pada smartphone android. Gambar dilihat di bawah ini.



**Gambar 0.3 Microkontroler Arduino Pro Micro**

## 2.12 USB to TLL

Adaptor USB adalah jenis konverter protokol yang digunakan untuk mengubah sinyal data USB ke dan dari standar komunikasi lainnya. Biasanya, adaptor USB digunakan untuk mengubah data USB menjadi data port serial standar dan sebaliknya. Umumnya sinyal data USB diubah menjadi RS232 , RS485 , RS422 , atau data serial UART tingkat TTL . Protokol RS423 serial yang lebih lama jarang digunakan lagi, jadi adaptor USB ke RS423 kurang umum




**Gambar 0.4 USB to TLL**

### 2.13 Potensio Rotary

Potensiometer (POT) adalah salah satu jenis Resistor yang Nilai Resistansinya dapat diatur sesuai dengan kebutuhan Rangkaian Elektronika ataupun kebutuhan pemakainya. Potensiometer merupakan Keluarga Resistor yang tergolong dalam Kategori Variable Resistor. Secara struktur, Potensiometer terdiri dari 3 kaki Terminal dengan sebuah shaft atau tuas yang berfungsi sebagai pengaturnya

### 2.14 Basis Data

Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. Pada buku ini menggunakan basis data relasional yang diimplementasikan dengan tabel-tabel yang saling memiliki relasi.

Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, entah berupa file teks ataupun Database Management System (DBMS). Kebutuhan basis data dalam sistem informasi meliputi :

Memasukan, menyimpan dan mengambil data.Membuat laporan berdasarkan data yang telah disimpan.Tujuan dari buatnya table-tabel disini adalah untuk menyimpan data ke dalam tabel-tabel agar mudah diakses. Oleh karena itu, untuk merancang tabel- tabel yang akan dibuat maka dibutuhkan pola piker penyimpanan data nantinya jika dalam bentuk baris-baris data (record) dimana setiap baris terdiri dari beberapa kolom.

## 2.15 Structured Query Langage (SQL)

SQL *(Structured Query Language)* adalah bahasa yang digunakan untuk mengelola data pada RDBMS SQL awalnya dikembangkan berdasarkan teori aljabar relasional dan kalkulus.

SQL mulai berkembang pada tahun 1970an. SQL mulai digunakan sebagai standar yang resmi pada tahun 1986 oleh ANSI *(American National Standards Institute)* dan pada tahun 1987 oleh ISO *(International Organization for Standardization)* dan disebut sebagai SQL-86. Pada perkembangannya, SQL beberapa kali dilakukan revisi.

## 2.16 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*) atau DBMS yang *multithread*, MySQL adalah sebuah aplikasi *Relational Database Management* Server (RDBMS) bersifat *open source* yang memungkinkan data diakses dengan cepat oleh banyak pemakai secara bersamaan dan juga memungkinkan pembatasan akses pemakai berdasarkan privilege (hak akses) yang diberikan.MySQL menggunakan bahasa SQL (*structured query language)* yang merupakan bahasa standar pemograman *database*.MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomastis.

*MySQL dipublikasikan sejak tahun 1996, akan tetapi sebenarnya sudah* dikembangkan sejak tahun 1979. MySQL telah memenangakan penghargaanLinux Journal Reader's Choice Award selama tiga tahun. MySQL sekarang tersedia di bawah lisensi open source, tapi ada juga lisensi utuk menggunakan MySQL yang bersifat komersial. Berikut kelebihan – kelebihan di MySQL:

1. Sederhana Skalabilitas, MSQL dapat menangani database yang besar, yang telah dibuktikan implementasinya dalam organisasi seperti Yahoo, Google, Cisco, HP, NASA dan lain sebagainya.
2. Portabilitas, MySQL dapat berjalan pada berbagai macam sistem operasi termasuk Windows, Unix, Linux, Solaris dan Mac OS. Juga dapat berjalan pada arsitektur yang berbeda, mulai dari low-end PC sampai highend mainframe.
3. Konektivitas, MySQL sepenuhnya mendukung jaringan dan dapat diakses dari mana saja di internet serta pengguna dapat mengakses21database MySQL secara bersamaan. MySQL juga menyediakan berbabagai macam API (*Aplication Program interface*) untuk mendukung konektivitas aplikasi yang ditulis dalam bahasa C, C++, Perl, PHP, Java, Phyton,C#
4. Mudah digunakan, MySQL mudah untuk digunakan dan diimplementasikan.
5. Open Source, MySQLAB membuat kode MySQL tersedia untukdigunakan setiap orang.

Beberapa perintah dasar SQL yang sering dipergunakan pada MySQL:

1. Create Database, perintah yang dipergunakan membuat database baru. Sintaks : CREATE DATABASE DATABASE_NAME.
2. Drop Database, perintah yang digunakan untuk menghapus database. Sintaks : DROP TABEL TABEL_NAME.
3. Create Tabel, perintah yang digunakan untuk membuat tabel baru. Sitntaks : Create Tabel tabel_name (create_definition).

4. Describe, perintah yang digunakan untuk mendeskripsikan tabel. Sintaks : Describe (Desc) tabel [colum].
5. Alter Tabel, perintah yang digunakan untuk menghapus tabel.Sintaks : Alter [Ignor] Tabel table_name.
6. Drop Tabel, perintah yang digunakan untuk menghapus tabel. Sintaks : Drop Tabel tabel_name [tabel_name..].
7. Sensor Delete, perintah yang digunakan untuk menghapus record dari tabel.Sintaks : Delete From tabel_name Where Where_definiition.
8. Select, perintah yang digunakan untuk query ke database.Sintaks : select * from tabel_name.
9. MySQL mendukung beberapa API yang memungkinkan aplikasi yangditulis dalam berbagai jenis bahasa pemrograman untuk berkomunikasi dengandatabase MySQL.

Berikut beberapa API yang didukung oleh MySQL:

1. C API, adalah antarmuka pemrograman utama yang memungkinkanaplikasi C/C++ untuk menghubungkan ke MySQL. Sebagian besaraplikasi klien termasuk dalam distribusi MySQL yang ditulis dalam Cdan bergantung pada API ini.
2. ODBC, MySQL mendukung Open Database Connectivity (ODBC)melalui MySQL Connector / ODBC. ODBC adalah konektivitasdatabase standar yang memungkinkan berbagai jenis aplikasi untukmenghubungkan ke berbagai jenis database.
3. JDBC, MySQL mendukung Java Database Connectivity (JDBC)melalui MySQL Connector / JDBC. JDBC adalah konektivitas database standar yang memungkinkan Java untuk menghubungkan ke berbagaijenis database.
4. PHP API, yang sekarang termasuk dengan preprocessorPHP, memungkinkan *script* PHP pada halaman web untuk berkomunikasi secara langsung dengan database MySQL. Koneksi database dan permintaan data (melalui pernyataan SQL) dikodekanlangsung dalam script PHP.

Dengan aplikasi *mySQL* bersifat *open source* memungkinkan khalayak global untuk berpartisipasi dalam pengembangan.Kelemahan MySQL dari dulu sampai saat ini adalah *feature-creep* artinya MySQL berusaha kompatibel dengan beberapa standar serta berusaha memenuhinya namun jika itu diungkapkan kenyataannya bahwa fitur-fitur tersebut belum lengkap dan belum berperilaku sesuai standar.

## 2.17 Web Service

*Web service* adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem, karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada *platform* yang berbeda. *Web service* digunakan sebagai suatu fasilitas yang disediakan oleh suatu *website* untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (*service*) yang disediakan oleh suatu sistem yang menyediakan *web service. Web service* menyimpan data informasi dalam format XML, sehingga data ini dapat diakses oleh sistem lain walaupun berbeda platform, sistem operasi, maupun bahasa compiler.

*Web service* bertujuan untuk meningkatkan kolaborasi antar pemrogram dan perusahaan, yang memungkinkan sebuah fungsi di dalam *Web Service* dapat dipinjam oleh aplikasi lain tanpa perlu mengetahui detil pemrograman yang terdapat di dalamnya. Beberapa alasan mengapa digunakannya *web service* adalah sebagai berikut :

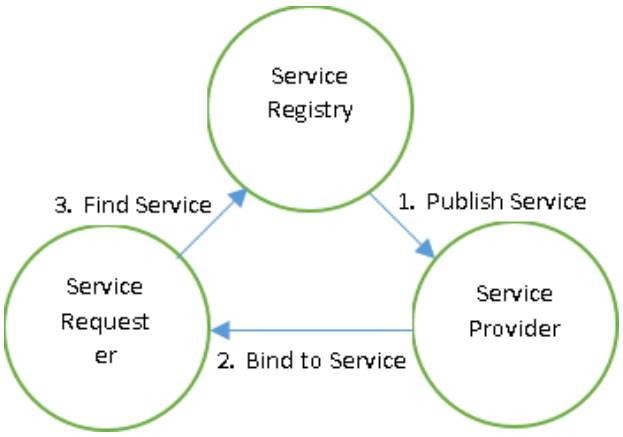
1. Sensor merupakan jenis tranduser yang digunakan sebagai pengubah besaran mekanis, magnetis, panas, sinar, dan kimia menjadi suatu tegangan dan arus listrik. Sensor biasanya digunakan untuk melakukan pengukuran atau untuk melakukan pengendalian. Berikut ini adalah jenis-jenis sensor yang digunakan.
2. Web service memiliki kemudahan dalam proses deployment-nya, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. *Web*

*service* cukup di-*upload* ke *web server* dan siap diakses oleh pihak-pihak yang telah diberikan otorisasi.

3. Web service berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian *web service* tidak memerlukan konfigurasi khusus di sisi *firewall.*

### 2.17.1 Arsitektur Web Service

*Web service* memiliki tiga entitas dalam arsitekturnya, yaitu: *Service Requester* (peminta layanan), *Service Provider* (penyedia layanan), *Service Registry (*daftar layanan) yang setiap entitas arsitektur tersebut ikut berperan penting dalam pembangunan arsitektur *Web Service.* Berikut ini adalah gambar arsitetktur dari *web service* dapat dilihat pada gambar 2.17 :




**Gambar 2.17 Arsitektur Web Service**

a. *Service Provider*: Berfungsi untuk menyediakan layanan/*service* dan mengolah sebuah *registry* agar layanan-layanan tersebut dapat tersedia.
b. *Service Registry*: Berfungsi sebagai lokasi central yang mendeskripsikan semua layanan/*service* yang telah di-register.
c. Sensor *Service Requestor*: Peminta layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut.

### 2.17.2 Operasi-Operasi Web Service

Secara umum, *web service* memiliki tiga operasi yang terjadi di dalamnya, yakni meliputi :

1. *Publish/Unpublish*: Menerbitkan atau menghapus layanan ke dalam atau dari *registry*.
2. *Find*: *Service requestor* mencari dan menemukan layanan yang dibutuhkan.
3. *Bind: Service requestor* setelah menemukan layanan yang dicarinya, kemudian melakukan binding ke *service provider* untuk melakukan interaksi dan mengakses layanan/*service* yang disediakan oleh *service provider*.

## 2.18 Browser Web

*Browser web* adalah *software* yang digunakan untuk menampilkan informasi dari server *web*. *Software* ini kini telah dikembangkan dengan menggunakan user interface grafis, sehingga pemakai dapat dengan melakukan *point* dan *click* untuk pindah antar dokumen. Lynx adalah *web browser* yang masih menggunakan mode teks, yang akibatnya adalah tidak ada gambar yang dapat ditampilkan.

Lynx ini ada pada lingkungan DOS dan Unix. Akan tetapi perkembangan baru web browser dengan GUI. Dapat dikatakan saat ini ada banyak web browser GUI, diantaranya Internet Explorer, Google Chrome, Mozilla Firefox dan Opera Beberapa browser tersebut kini bersaing untuk merebut pemakainya, dengan berusaha danmendekati standar dokumen HTML yang direkomendasikan oleh W3.

## 2.19 Web

*Web* atau lebih dikenal dengan sebutan *website* merupakan halaman situs sistem informasi yang dapat diakses secara cepat. ini didasari dari adanya perkembangan teknologi informasi tentang internet dan komunikasi. Melalui perkembangan teknologi informasi, tercipta suatu jaringan antar komputer yang

salingberkaitan. Jaringan yang dikenal dengan istilah internet secara terus-menerus menjadi pesan–pesan elektronik.

Pada umumnya, halaman situs *web* berupa dokumen yang ditulis dengan format HTML (*Hyper Text Markup Language*) dan dapat diakses melalui HTTP (*Hyper Text Transfer Protocol*). HTTP adalah protokol pengirim informasi dari *server* sebuah *website* yang akan ditampilkan kepada end user melalui *web browser*.

Alamat sebuah *website* dapat menggunakan sebuah *domain* atau *subdomain*. Situs *web* harus ditempatkan pada sebuah hosting yang tergabung ke dalam WWW (*World Wide Web*) agar dapat diakses oleh orang-orang.

## 2.20 Cascading Style Sheet (CSS)

*Cascading Style Sheet* (CSS) merupakan aturan untuk mengatur beberapa komponen dalam sebuah *web* sehingga akan lebih terstruktur dan seragam. CSS bukan merupakan bahasa pemrograman.

Sama halnya *styles* dalam aplikasi pengolahan kata seperti Microsoft Word yang dapat mengatur beberapa *style*, misalnya *heading*, subbab, *bodytext*, *footer*, *images*, dan *style* lainnya untuk dapat digunakan bersama-sama dalam beberapa berkas (*file*). Pada umumnya CSS dipakai untuk memformat tampilan halaman *web* yang dibuat dengan bahasa HTML dan XHTML. CSS dapat mengendalikan ukuran gambar, warna bagian tubuh pada teks, warna tabel, ukuran *border*, warna *border*, warna *hyperlink*, warna *mouse over*, spasi antar paragraf, spasi antar teks, *margin* kiri, kanan, atas, bawah, dan parameter lainnya. CSS adalah bahasa *style sheet* yang digunakan untuk mengatur tampilan dokumen. Dengan adanya CSS memungkinkan untuk menampilkan halaman yang sama dengan format yang berbeda.

Untuk saat ini terdapat tiga versi CSS, yaitu CSS1, CSS2, dan CSS3. CSS1 dikembangkan berpusat pada pemformatan dokumen HTML, CSS 2 dikembangkan untuk memenuhi kebutuhan terhadap format dokumen agar bisa ditampilkan di *printer*, sedangkan CSS3 adalah versi terbaru dari CSS yang

mampu melakukan banyak hal dalam desain *website*. CSS2 mendukung penentuan posisi konten, *downloadable*, huruf *font*, tampilan pada tabel / *table layout* dan media tipe untuk *printer*. Kehadiran versi CSS yang kedua diharapkan lebih baik dari versi pertama dan kedua.

CSS3 juga dapat melakukan animasi pada halaman *website*, diantaranya animasi warna hingga animasi 3D. Dengan CSS 3 desainer lebih dimudahkan dalam hal kompatibilitas websitenya pada *smartphone* dengan dukungan fitur baru yakni *media query*. Selain itu, banyak fitur baru pada CSS3 seperti: *multiple background*, *border-radius*, *drop-shadow*, *border-image*, CSS *Math*, dan CSS *Object Model*.

### 2.21 BootStrap

*Bootstra*p adalah *library* (pustaka / kumpulan fungsi-fungsi) dari Framework CSS yang dibuat khusus untuk bagian pengembangan fontend dari suatu website. Didalam library tersebut terdapat berbagai jenis file yang diantaranya HTML, CSS, dan Javascript. Hampir semua developer website menggunakan framework bootstrap agar memudahkan dan mempercepat pembuatan website. Karena semuanya sudah ada dalam frameworknya sehingga para develop / pengembang hanya tinggal membuat / menyisipkan class nya yang ingin dipakai seperti membuat tombol, grid navigasi dan lain sebagainya.

Bootstrap telah menyediakan kompulan aturan dan komponen class interface dasar sebagai modal dalam pembuatan web yang telah dirancang sangat baik untuk memberikan tampilan yang sangat menarik, bersih, ringan dan memudahkan bagi penggunanya. Dan penggunaan bootstrap ini kita juga diberikan keleluasan salama pengembangan website, anda bisa merubah dan menambah class sesuai dengan keinginan

### 2.22 Unified Modelling Language

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan

perangkat lunak. untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

*Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik.

Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE *(Object-Oriented Software Engineering).*

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (method war) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang

berlainan. Maka dibentuklah sebuah standar dari permodelan perangkat lunak yaitu UML.

### 2.22.1 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi, atau sistem lainnya). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata.

Ada banyak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dan abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dipunyai oleh objek akan dienkapsulasi (dihubungkus) menjadi suatu kesatuan.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancanggan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus (dienkapsulasi) menjadi kelompok daata dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat dan komponen lainnya, dan dapat berinteraksi satu sama lain.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek :

1. Kelas *(class)*

   Kelas adalah sekumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (metode/operasi), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain,

dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara teknik kelas adalah sebuah struktur dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

2. Objek *(object)*

   Objek adalah abstraksi dari sesuatu yang mewakili dunia nyata benda, manusia, satua organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstra. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan. Secara teknis, sebuah kelas saat program dieksekusi makan akan dibuat sebuah objek. Objek dilihat darisegi teknis adalah elemen pada saat *runtime* yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehinga sebuah objek hanya ada saat sebuah program dieksekusi. Jika masih dalam bentuk kode, disebut sebagai kelas jadi pada saat *runtime* (saat sebuah program dieksekusi), yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas.

3. Metode (*method*)

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode atau operasi dapat berasal dari *event*, aktifitas atau aksi keadaan, fungsi, atau kelakuan dunnia nyata. Contoh metode atau operasi misalnya *Read, Write, Move, Copy,* dan sebagainya.

4. Atribut (*atribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat beruoa nilai atau elemen-elemen data yang

dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkusa atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

7. Pewarisan (*inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

8. Antarmuka (*interface*)

   Antarmuka sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antar Objek

    Komunikasi antarobjek dilakukan lewat pesan (*message*) yang dikirim dari satu objek ke objek lainnya.

12. Polimorfisme (*polymorphism*)

    Kemampuan suatu objek digunakan di banyak tujuan yang berbeda dengan nama yang sehingga menghemat baris program.

13. *Package*

*Package* adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

### 2.22.2 Pengenalan Unified ModellingLanguage (UML)

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

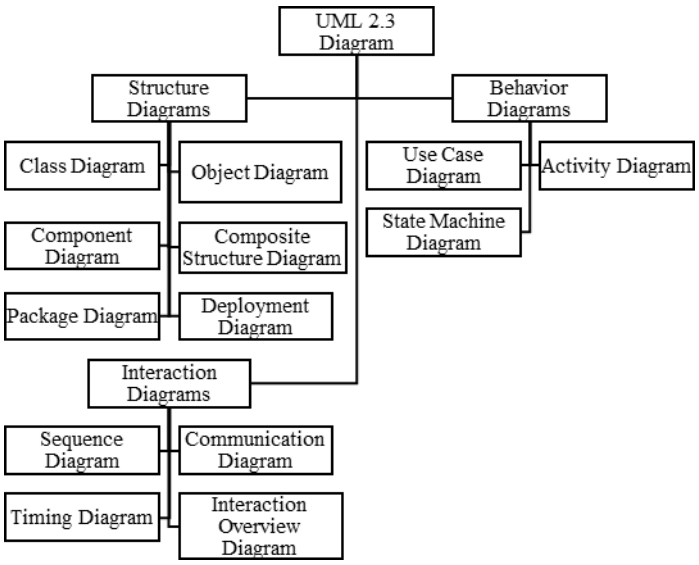
Pada perkembangan teknil pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhanpemodelan voisual untu menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa *visual* untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan digaram dan teks-teks pendukung.UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

Seperti yang kita ketahui di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkunan dan konteksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada level abstraksi penggunaannya. Jadi, belum tentu pandangan yang perbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin digambarkan. Secara analogi jika dengan bahasa yang

digunakan sehari-hari, belum tentu penyampaian bahasa dengan puisi adalahhal yang salah. Sistem informasi bukanlah ilmu pasti, maka jika ada banyak perbedaan dan interpretasi di dalam bidang sistem informasi.

### 2.22.3 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.28.



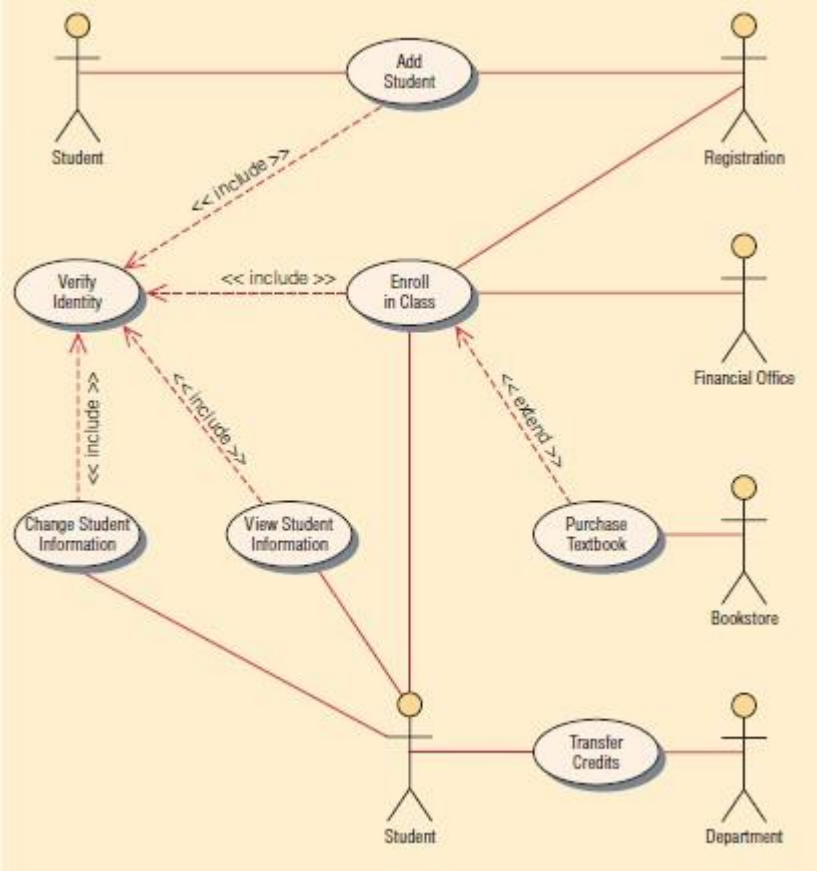

**Gambar 0.5 Diagram UML**

Berikut ini penjelasan singkat dari pembagian kategori tersebut :

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

### 2.22.4 Usecase Digaram

*Use case* atau diagram *Use case* merupakan pemodelan untuk kelakuan (*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 di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.




**Gambar 0.6 UseCase Diagram**

### 2.22.5 Actor

*Actor* adalah sesuatu (entitas) yang berhubungan dengan sistem dan berpartisipasi dalam *use case*. *Actor* menggambarkan orang, sistem atau entitas *Eksternal* yang secara khusus membangkitkan sistem dengan *input* atau masukan

kejadian-kejadian, atau menerima sesuatu dari sistem. *Actor* dilukiskan dengan peran yang mereka mainkan dalam *use case*, seperti *Staff*, Kurir dan lain-lain.

Dalam *use case* diagram terdapat satu aktor pemulai atau initiator *actor* yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor lain yang berpartisipasi atau participating *actor*. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut.

<<nama actor>>

### 2.22.6 Usecase

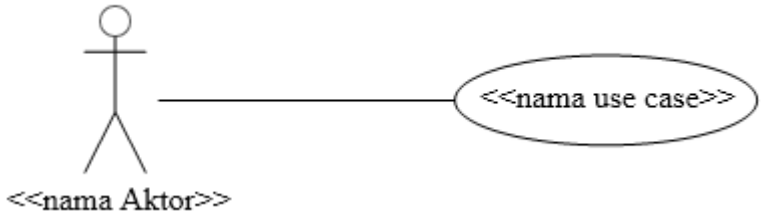
*Use case* yang dibuat berdasarkan keperluan aktor merupakan gambaran dari "apa" yang dikerjakan oleh sistem, bukan "bagaimana" sistem mengerjakannya. *Use case* diberi nama yang menyatakan apa hal yang dicapai dari interaksinya dengan aktor

<<nama use case>>

### 2.22.7 Relationship

Relasi (*relationship*) digambarkan sebagai bentuk garis antara dua simbol dalam *use case diagram*. Relasi antara *actor* dan *use case* disebut juga dengan asosiasi (*association*). Asosiasi ini digunakan untuk menggambarkan bagaimana hubungan antara keduanya.

Relasi-relasi yang terjadi pada *use case diagram* bisa antara *actor* dengan *use case* atau *use case* dengan *use case*.




Relasi antara use case dengan use case :

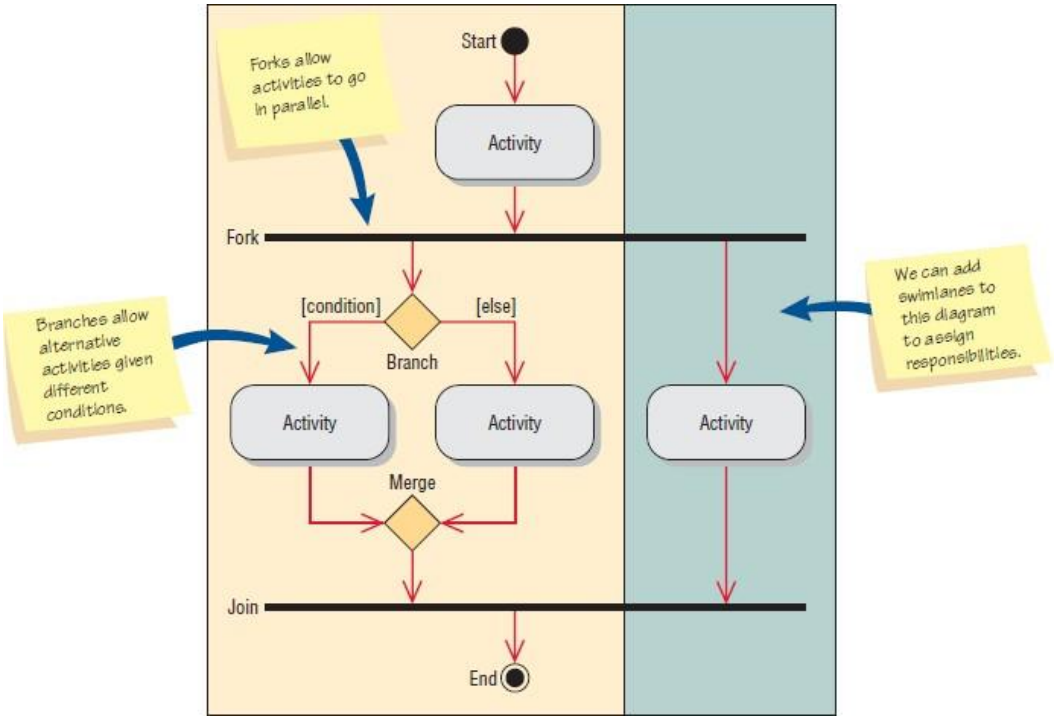
1. *Include*, pemanggilan use case oleh use case lain atau untuk menggambarkan suatu use case termasuk di dalam use case lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan <<include>>.
2. *Extend*, digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak kontrol form dan mendeklarasikan extension pada use case utama. Atau dengan kata lain adalah perluasan dari use case lain jika syarat atau kondisi terpenuhi. Digambarkan dengan garis berpanah dengan tulisan <<extend>>.
3. *Generalization/Inheritance,* dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan base-parent use case. Digambarkan dengan garis berpanah tertutup dari base use case ke parent use case.

### 2.22.8 Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis, yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut.

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

2. Urutan atau pengelompokan tampilan dari sistem/ *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.



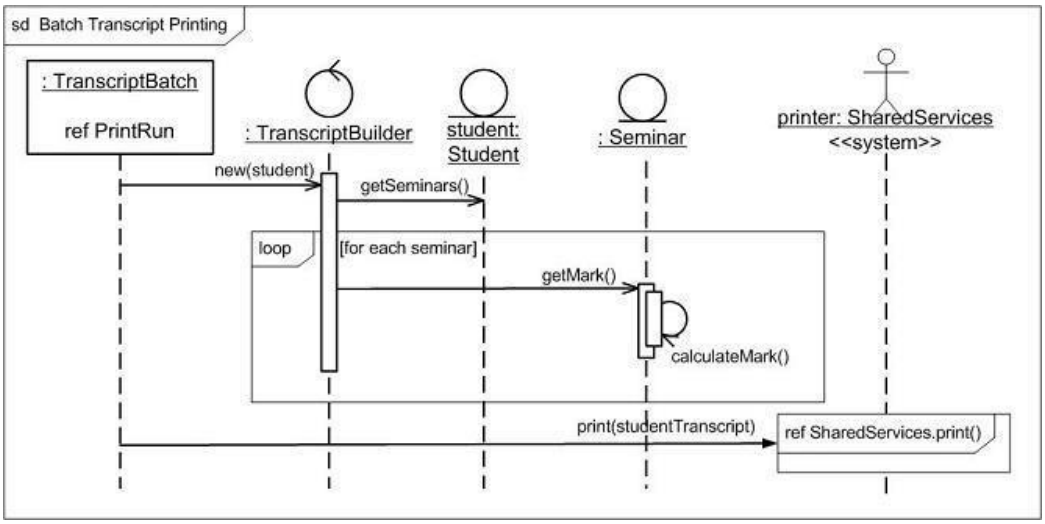

**Gambar 0.7 Activity Diagram**

### 2.22.9 Sequence Diagram

Diagram *sequence* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Diagram *sequence* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.




**Gambar 0.8 Sequence Diagram**

### 2.22.10 Class Diagram

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

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas ebaiknya memiliki jenis-jenis kelas berikut

1. Kelas Main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian usecase

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case.*
4. Kelas yang diambil dari pendefinisian data

Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan coupling. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *choesion* yang kuat dan kadar *coupling* yang lemah. Dalam *class* diagram terdapat beberapa relasi (hubungan antar *class*) yaitu :

1. *Generalization* dan *Inheritence*

   Diperlukan untuk memperlihatkan hubungan pewarisan (*inheritance*) antar unsur dalam diagram kelas. Suatu hubungan turunan dengan mengasumsikan satu kelas merupakan suatu superClass (kelas super) dari kelas yang lain. Generalization memiliki tingkatan yang berpusat pada superClass.

2. *Associations*

   Suatu hubungan antara bagian dari dua kelas. Terjadi association antara dua kelas jika salah satu bagian dari kelas mengetahui yang lainnya dalam melakukan suatu kegiatan. Di dalam diagram, sebuah association adalah penghubung yang menghubungkan dua kelas.

3. *Aggregation*

   Hubungan antar-*class* dimana *class* yang satu *(part class)* adalah bagian dari *class* lainnya *(whole class).*

4. *Composition*

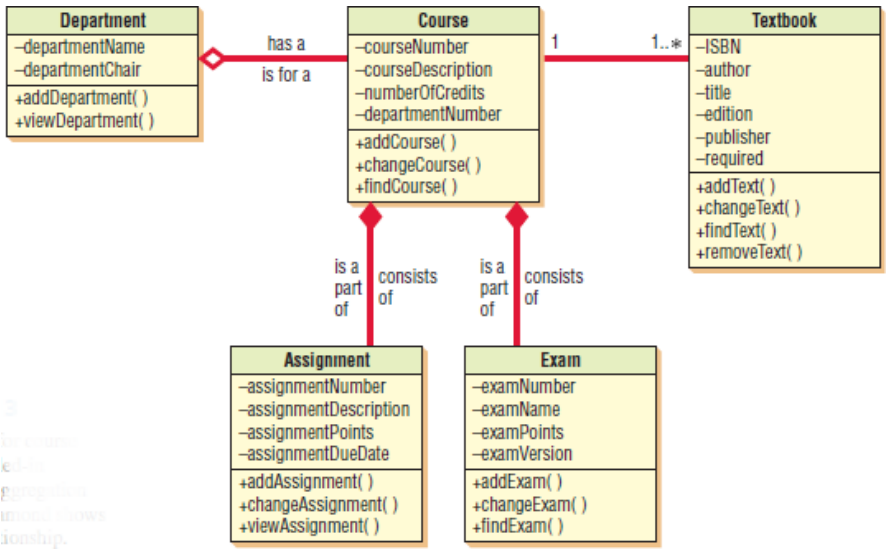
   *Aggregation* dengan ikatan yang lebih kuat. Di dalam *composite aggregation*, siklus hidup *part class* sangat bergantung pada *whole class* sehingga bila objek *instance* dari *whole class* dihapus maka objek *instance* dari *part class* juga akan terhapus.

5. *Depedency*

Hubungan antar *class* dimana sebuah *class* memiliki ketergantungan pada *class* lainnya tetapi tidak sebaliknya.

6. *Realization*

   Hubungan antar-*class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya. Biasanya realization digunakan untk menspesifikasikan hubungan antara sebuah *interface* dengan *class* yang mengimplementasikan *interface* tersebut.




**Gambar 0.9 Class Interface**