BAB 2
TINJAUAN PUSTAKA

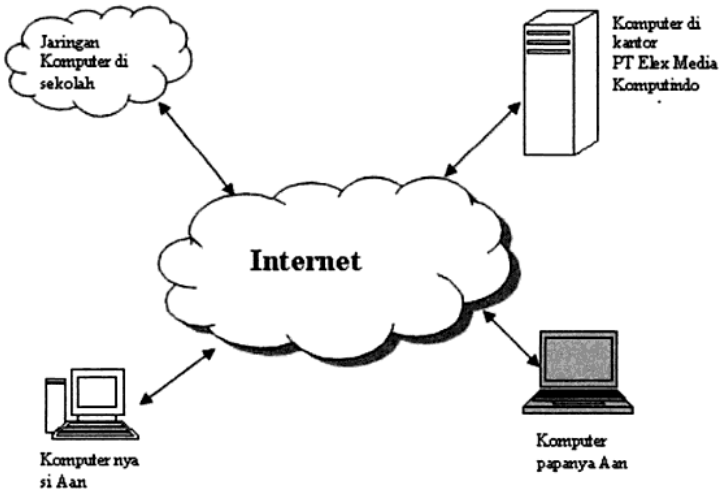
2.1 Profil Perusahaan

UD. Caesar adalah salah satu kelompok tani yang bergerak di bidang pembudidayaan ikan nila dan lele yang berada di kota cimahi dan sudah berjalan selama kurang lebih 2 tahun. UD. Caesar merupakan pembudidaya ikan milik perorangan dan dikelola secara mandiri oleh pemilik dan memiliki 1 orang karyawan yang bertugas mengawasi dan memberi pakan ikan setiap harinya. Kini UD. Caesar memiliki 4 kolam budidaya ikan dengan luas kolam 3x3 m untuk 1 kolamnya dengan produksi ikan yang mencapai 0,5 ton dalam 1x panen setelah 4 bulan masa pembiakan. UD. Caesar memasarkan hasil produksi ikan nila dan lele kepada pelanggannya untuk konsumsi seperti memasarkan langsung ke pasar ataupun pedagang penjual makanan seperti pecel lele dan ikan bakar.

2.2 Internet

Internet adalah singkatan dari *Interconnected Network*. Internet merupakan sebuah sistem komunikasi yang mampu menghubungkan jaringan-jaringan komputer di seluruh dunia. Berbagai jenis komputer dengan spesifikasi berbeda-beda dapat saling berkomunikasi melalui internet [1]. Beberapa bentuk jaringan yang berbeda-beda dapat saling bertukar informasi dan data melalui internet menggunakan seperangkat aturan yang disebut protkol TCP/IP. Untuk membedakan setiap komputer atau jaringan yang terhubung ke internet maka digunakan sebuah identitas tertentu yang disebut alamat IP (*IP Address*). Alamat IP merupakan kombinasi angka-angka yang menunjukan identitas sebuah komputer atau jaringan di internet, contoh : 202.155.2.111. selain menggunakan alamat IP, beberapa komputer atau jaringan dapat juga memiliki identitas berupa nama yang mudah diingat. Nama tersebut disebut sebagai nama domain, contohnya : www.yahoo.com atau www.google.com.internet mampu membuat pekerjaan kita menjadi lebih mudah dan efisien. Segala informasi bisa dengan

mudah didapat melalui internet. Perbedaan jarak tidak lagi menjadi hambatan dalam melakukan komunikasi.



Gambar 2. 1 Komputer dan jaringan dapat terhubung melalui internet

2.2.1 Internet Of Things

Internet Of Things merupakan konsep yang bertujuan untuk memperluas manfaat dari konekivitas internet yang tersambung secara terus menerus. *Internet Of Things* adalah sebuah teknologi yang memungkinkan adanya pengendalian, komunikasi, kerja sama dengan berbagai perangkat keras, berbagi data, memvirtualisasikan segala hal nyata ke dalam bentuk internet dan lain-lain melalui jaringan internet[3]. Pada hakekatnya benda internet atau *Internet Of Things* mengacu pada benda yang dapat diidentifikasikan secara unik sebagai representasi virtual dalam struktur berbasis internet. Istilah *Internet Of Things* awalnya disarankan oleh Kevin Asthon pada 1999 dan mulai populer melalui Auto-ID Center di MIT.

2.2.2 Sejarah Internet Of Things

Pada zaman ini sudah banyak aplikasi yang berbasiskan pada *Internet Of Things* karena tuntutan kemudahan dan keefisiensian waktu contoh umumnya seperti berbelanja online, print dokumen melalui internet, *Home Automation* dan

lain-lain. Sejarah perkembangan *Internet Of Things* dimulai pada bulan September 1991 [2].

1. Pada bulan September 1991 Mark Weiser dengan Xerox PARC menerbitkan arikel yang menjelaskan tentang masa depan dimana elemen-elemen *hardware* dan *software* dihubungkan dengan kabel, gelombang radio dan *infrared* akan menjadi hal yang terdapat dimana-mana dan tidak ada satu orang pun yang akan menyadari keberadaannya.
2. Pada bulan Januari 1992 Trojan Room Coffe Pot dijelaskan dengan detail pada artikel Comm Week. Quentin Stafford-Fraser dan Paul Jardetzky pada universitas Cambridge menggunkaan kamera di laboratorium Trojan Room dan menulis program server untuk menangkap gambar dari pot setiap detiknya untuk memonitor tinggi kopi, sehingga peneliti dapat mengetahui kapan kopi sudah dapat dihidangkan.
3. Pada tahun 1993, peneliti pada Universitas Columbia menggembangka*n Knowledge-based Augmented Reality for Maintenance Assistance* (KARMA), yang melapisi semantil wireframe dan instruksi perbaikan diatas peralatan yang akan diperbaiki.
4. Pada bulan September 1994 B.N. Schilit dan M.M. Theimer menggunakan istilah *context-aware* untuk pertama kalinya dalam artikel bertema network.
5. Pada tahun 1995 siemens menyiapkan departemen untuk mengembangkan dan meluncurkan modul data GSM yang dinamakan M1 untuk aplikasi industri Meachine to meachine.
6. Pada bulan januari 1999 Bill Joy, pendiri Sun Microsystem, menjelaskan komunikasi *device-device* sebagai salah satu web yang diimpikan pada presentasinya dalam forum ekonomi dunia. Pada tahun ini juga disebutkan istilah *Internet of Things* untuk pertama kalinya.
7. Pada bulan oktober tahun 2000 Sanjay Sarma dan David Broke dari MIT menerbitkan *White Paper* yang merupakan impian untuk membangun Auto-ID Center MIT dimana semua benda baik benda fisik maupun elektronik.

8. Pada tahun 2000 LG mengumumkan rencana untuk meluncurkan kulkas yang dapat terhubung dengan internet.
9. Pada tahun 2002 Chana Schoenberger mengeluarkan artikel dengan judul "*The Internet of Things"* pada majalah Forbes dengan kutipan dari Ashton, "Kita membutuhkan sebuah *Internet of Things,* sebuah cara standarisasi agar komputer mengerti dunia nyata."
10. Pada tahun 2003 BigBelly Solar dibiayai untuk membuat tong sampah yang mendapatkan energi dari matahari dan memberitahu status isi tong sampah tersebut melalui internet.
11. Pada bulan September 2004 pada sebuah artikel tentang komputer, G Lawton menyebutkan bahwa M2M dibuat berdasarkan bahwa mesin memiliki nilailebih saat dihubungkan dengan internet dan internet memiliki nilai lebih saat lebih banyak mesin yang dapat saling dihubungkan.
12. Pada bulan januari 2005, Nabaztag dibuat oleh Rafi Haladjian dan Olivier Mevel melalui perusahaan yang bernama violet. Nabaztag adalah kelinci yang mengumpulkan informasi dari internet dan memberitahu user tentang pesan dan berita pesan.
13. Pada november 2005 Departemen Telekomunikasi Internasional PBB mengeluarkan laporan berjudul "*The Internet of Things."*
14. Pada 2008 IPSO didirikan untuk mempropaganda penggunaan IP pada peralatan yang saling berhubungan.
15. Pada bulan Mei 2010 ZigBee dengan forum Ipv6 menjalin kerjasama degan IPSO untuk mengadopsi jaringan IP pada benda-benda *smart* seperti *smartphone* dan *smart TV*.
16. Pada bulan februari 2011 Nest Labs memperkenalkan Nest *Learning Thermostat*, yang menggunakan sensor algoritma, pembelajaran mesin, dan perhitungan awan untuk memahami kondisi rumah pengguna serta menaikan atau menurunkan temperatur sesuai kebutuhan.
17. Pada 2012 Google memperkenalka Google *Glass* yang dapat memberi tahu semua informasi pada benda-benda fisik yang dilihat pengguna. Diadakan

peluncuran Ipv6 yang menyediakan alamat IP tak terbatas untuk peralatan yang dihubungkan dengan internet.

18. Pada tahun 2013 artikel Venture Beat memprediksi bahwa 2014 akan hadir sebaga "tahun *Internet of Things*". Qualcomm perusahaan pembuat chip, membentuk Allseen Alliance yang ditujukan untuk mengembangkan *open framework* untuk merealisasikan *internet of things.*
19. Pada 2014 Google Glass dijual pada masyarakat dengan harga $1500. Intel, perusahaan pembuat chip bersama perusahaan teknologi maju lainnya membentuk sebuah kelompok untuk mempromosikan *open framework Internet of Things* yang bernama *Open Interconnect Consortium* sebagai lawan Allseen Alliance.

2.3 Jaringan Komputer

Jaringan komputer adalah himpunan "interkoneksi" antara 2 komputer *autonomus* atau lebih yang terhubung dengan media transmisi kabel atau tanpa kabel (*wireless*). Dua komputer dikatakan terkoneksi apabila keduanya bisa saling bertukar data / informasi, berbagai *resource* yang dimiliki seperti file, printer, media penyimpanan (*hardisk, floppydisk, cd rom, flashdisk, dll*). Data yang berupa teks, audio, maupun video bergerak melalui media kabel atau tanpa kabel sehingga memungkinkan pengguna komputer dalam jaringan komputer yang dapat saling bertukar *file/data,* mencetak pada printer yang sama dan menggunakan *hardware* atau *software* yang terhubung dalam jaringan secara bersama-sama.

Tiap komputer, printer ataupun peripheral yang terhubung dalam jaringan disebut dengan *node*. Sebuah jaringan komputer sekurang-kurangnya terdiri dari dua unit komputer atau lebih, dapat berjumlah puluhan komputer, ribuan atau bahkan jutaan node yang saling terhubung satu sama lain [3].

2.3.1 Jenis jaringan

1. Jaringan LAN

   Local Area Network (LAN) merupakan jaringan yang menghubungkan sejumlah komputer yang ada dalam suatu lokasi dengan area terbatas seperti ruang tamu atau gedung. LAN dapat menggunakan media komunikasi seperti kabel dan *wireless*.

2. Jaringan WAN

   Wide Area Network (WAN) merupakan jaringan antara LAN satu dengan LAN lain yang dipisahkan oleh lokasi yang cukup jauh. Contoh penggunaan WAN adalah hubungan antara kantor pusat dengan kantor cabang yang berada di daerah-daerah.

3. Jaringan MAN

   Metropolitan Area Network (MAN) merupakan jaringan yang lebih besar dari pada LAN tetapi lebih kecil dari jaringan WAN. Jaringan MAN dan jaringan WAN sama-sama menghubungkan beberapa LAN yang membedakan hanya lingkup areanya yang berbeda.

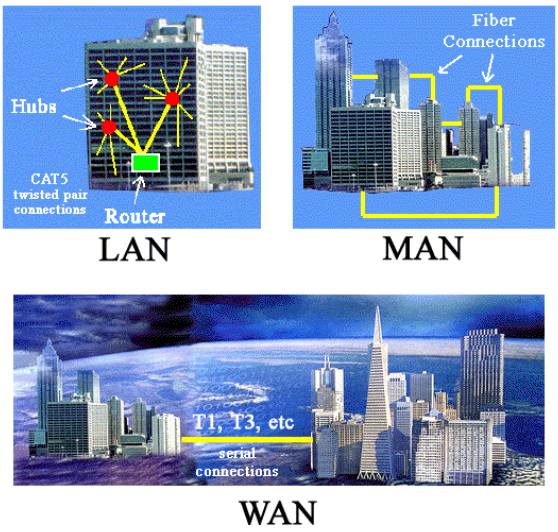
4. Jaringan WLAN

   Wireless Local Area Network (WLAN) adalah jaringan komputer yang menggunakan gelombang radio sebagai media transmisi data. Informas (data) ditransfer dari satu komputer ke komputer lain tanpa menggunakan kabel sebagai media perantara. WLAN sering disebut sebagai jaringan nirkabel atau jaringan tanpa kabel.

   Keuntungan jaringan WLAN :

   a. Jangkauan jaringan lebih luas dibandingkan dengan jaringan yang menggunakan media kabel.

b. Menyediakan pengaksesan informasi *realtime* dimana saja bagi pengguna LAN selama berada di dalam organisasinya.
c. Dapat dipasang dengan mudah serta dapat mengurangi penggunaan kabel yang melalui tembok dan atap, dan dapat digunakan pada jaringan dimana kabel tidak memungkinkan untuk dipasang.
d. Meskipun investasi awal lebih tinggi dari biaya perangkat LAN kabel, tapi biaya instalasi keseluruhan serta biaya penggunaan dapat diperkecil. Keuntungan biaya jangka panjang akan sangat besar pada jaringan kabel apabila sering terjadi perubahan dan perpindahan[5].



Gambar 2. 2 Jaringan LAN, MAN, WAN

2.4 Web

Web adalah fasilitas yang paling sering digunakan dan diakses setiap orang di internet. Web sudah berkembang sedemikian pesat dewasa ini. Banyak web yang bermunculan di internet. Perkembangan web tersebut menarik minat setiap orang untuk mempelajari bagaimana membuatnya. Sebuah web sederhana dan informatif dapat dibangun dengan menggunakan HTML, CSS, dan JavaScript.S

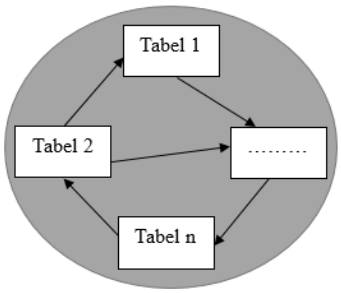
2.5 PHP

PHP adalah singkatan dari *Hypertext Preprocessor*. PHP merupakan bahasa pemrograman untuk membuat web yang bersifat *server-side scripting.* PHP memungkinkan untuk membuat halaman web yang bersifat dinamis. PHP dapat dijalankan pada berbagai sistem operasi, misalnya Windows, Linux, Mac OS. PHP juga mendukung beberapa Web Server lain, misalnya Microsoft IIS, Caudium, PWS, dan lain-lain. PHP dapat memanfaatkan database untuk menghasilkan halaman web yang dinamis. Sistem manajemen database yang sering digunakan bersama PHP adalah MySQL namun, PHP juga mendukung sistem manajemen database lainnya seperti Oracle, Microsoft Access, Interbase, dBase, PostgreSQL.

Saat ini PHP telah berkembang hingga versi 5. PHP 5 mendukung penuh *Object-Oriented Programming* (OOP), integrasi XML, mendukung semua ekstensi terbaru MySQL, pengembangan Web service dengan SOAP dan REST juga peningkatan lainnya dibandingkan versi sebelumnya. PHP juga bersifat *open source* sehingga semua orang dapat menggunakannya secara gratis [4].

2.6 Basis Data

Basis data adalah kumpulan dari data yang saling berelasi. Data sendiri mrupakan fakta mengenai objek, orang, dan lain-lain. Data dinyatakan dengan nilai (angka, deretan karakter atau simbol). 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. basis data relasional yang diimplementasikan dengan table-tabel yang saling memiliki relasi seperti pada gambar berikut.



Gambar 2. 3 Ilustrasi Basis Data

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:

1. Memasukan, menyimpan dan mengambil data
2. 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.

2.6.1 Kelebihan Basis Data

Banyak manfaat/kelebihan dari basis data diantaranya adalah :

1. Kecepatan dan kemudahan (*speed*)
2. Kebersamaan pemakai (*sharability*)
3. Pemusatan kontrol data
4. Efisiensi ruang penyimpanan (*space*)
5. Keakuratan (*accuracy*)
6. Ketersediaan (*availability*)

7. Kelengkapan (*completeness*)
8. Keamanan (*security*)
9. Kemudahan dalam pembuatan program aplikasi baru
10. Pemakaian secara langsung.
11. *User view*

2.6.2 Alur Hidup Basis Data

Tidak hanya perangkat lunak yang memiliki alur hidup, dalam membuat perancangan basis data juga memiliki alur hidup atau *Database Life Cycle* (DBLC). Alur hidup basis data dapat di lihat gambar berikut:



Gambar 2. 4 DBLC

Fase-fase DBLC antara lain:

1. Analisis kebutuhan/*requiretment analysis*

Hal-hal yang harus dilakukan pada tahap ini adalah:

a. Didefinisikan dengan mewawancarai produsen dan pemakai data, data apa sajakah yang butuh untuk disimpan dan terkait dengan aplikasi computer yang akan dikembangkan
b. Membuat kontrak spesifikasi basis data
c. *Entity Relationship Diagram* (ERD) sebagai bagian dari desain konseptual

2. Desain Lojik basis data / *logical database design*

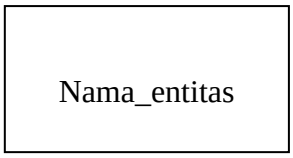
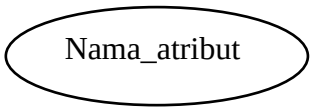
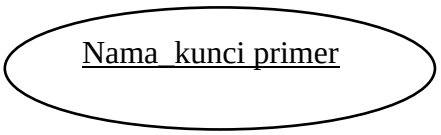
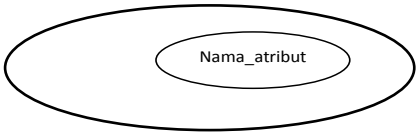
   Pada tahapan ini harus dibuat rancangan lojik basis data.biasanya pada tahap ini dibuat *Conceptual Data Model* (CDM).
3. Desain fisik basis data / *physical database design*

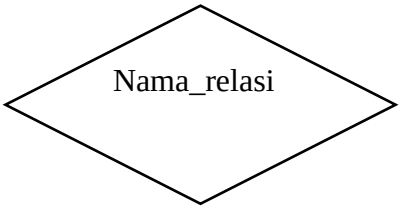
   Pada tahap ini harus dibuat rancangan fisik basis data. Biasanya pada tahap ini dibuat *Physical Data Model* (PDM).
4. Implementasi
   a. Membuat Query SQL
   b. Aplikasi ke DBMS atau file.

2.6.3 ERD

Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan *Entity Relationship Diagra* (ERD). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional sehingga jika penyimpanan basis data menggunakan OODBMS maka perancangan basis data tidak perlu menggunakan ERD. ERD memiliki beberapa aliran notasi seperti notasi Chen (dikembangkan oleh Peter Chen), Barker (dikembangkan oleh Richard Barker, ian Palmer, Harry Ellis), notasi Crows Foot, dan beberapa notasi lain, namun yang banyak digunakan adalah notasi dari Chen. Berikut adalah simbol-simbol yang dignakan pada ERD dengan notasi Chen.

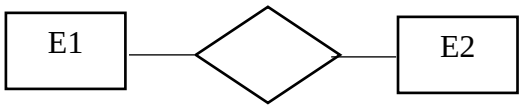
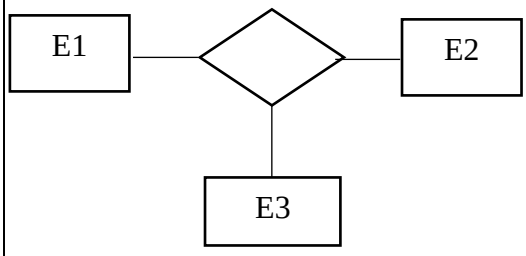
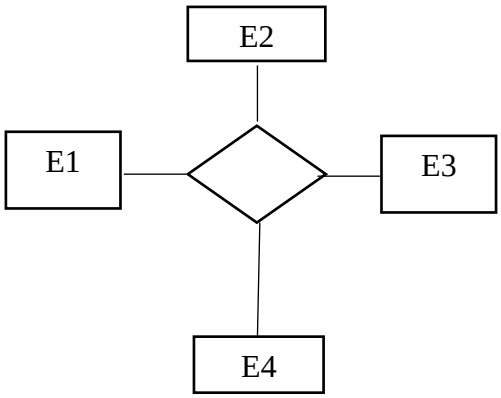
Tabel 2. 1 Simbol-simbol ERD

Simbol	Deskripsi
Entitas /*entity* 	Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data; benda yang memilki data dan harus disimpan datanya agar dapat diakses oleh aplikasi computer; penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel
Atribut 	Field atau kolom data yang butuh disimpan dalam suatu entitas
Atribut kunci primer 	Field atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses record yang diingkan; biasnya berupa ID; kuci primer dapat lebih dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama)
Atribut multinilai/ *multivalue* 	Field atau kolom data yang butuh disimpan dalam suatu entitas; biasanya diawali dengan kata kerja

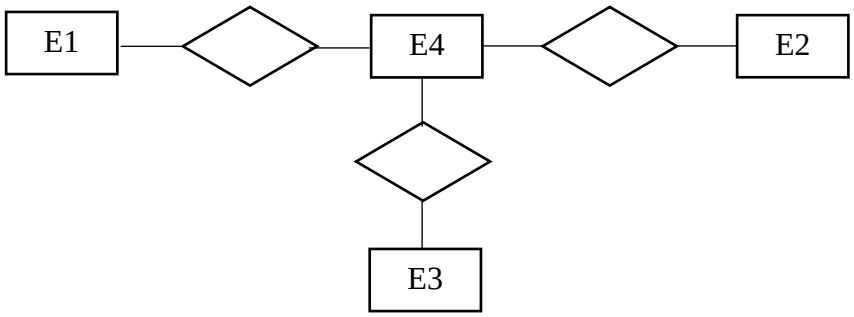
Relasi  	Relasi yang menghubungkan antara entitas; biasanya diawali dengan kata kerja
Asosiasi/ *association*  	Penghubung antara relasi dan entitas dan entitas dimana dikedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian Kemungkinan jumlah maksimun keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinatilas Misalkan ada kardinaltilas 1 ke N atau sering disebut dengan One to many menghubungkan entitas A dan Entitas B

ERD biasanya memiliki hubungan binary (satu relasi menghubungkan dua buah entitas). Beberapa metode perancangan ERD menoleransi hubungan relasi *ternary* (satu relasi menghubungkan tiga buah relasi) atau N-ary (satu relasi menghubungkan banyak enttitas), tiap banyak metode perancangan ERD yang tidak mengizinkan hubungan ternary atau N-ary. Berikut adalah contoh bentuk hubungan relasi dari ERD:

Tabel 2. 2 Relasi ERD

Nama	Gambar
Binary	
Ternary	
N-ary	

Beberapa metode perancangan ERD memberikan solusi untuk perancangan ERD yang tidak binary diubah sebagai berikut:

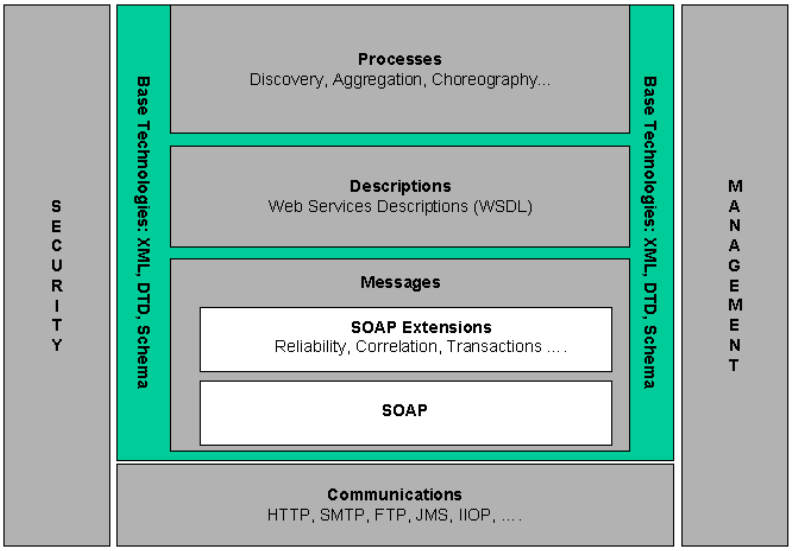


Gambar 2. 5 Menghindari relasi ternary

2.7 Web Service

Web *service* adalah sebuah entitas komputasi yang dapat diakses melalui jaringan internet maupun intranet dengan standar *protokol* tertentu dalam *platform* dan antarmuka bahasa pemograman yang independen. Tujuan pengembangannya adalah untuk menjembatani komunikasi antar program, sehingga aplikasi yang satu dan aplikasi yang lain yang terdapat pada suatu jaringan yang sama atau pada jaringan berbeda dapat saling berkomunikasi asalkan menggunakan standart protokol yang ditetapkan oleh web *service*. Hal ini bisa terjadi, karena standart protokol itu tidaklah terikat pada suatu *platform* atau bahasa pemograman. Protokol itu sendiri dibangun oleh *Extensible Markup Language* (XML) yang memang kenyataannya telah didukung oleh banyak *paltform*, bahasa pemograman, dan oleh *developer* di seluruh dunia.

Arsitektur web *service* dibangun oleh beberapa *laye*r dan teknologi yang saling berhubungan. Banyak cara untuk memvisualisaskan *service*, sama banyaknya dengan cara untuk membangun dan menggunakan web *service* gambar berikut hanyalah salah satu cara untuk menggambarkan arsitekturnya.



Gambar 2. 6 Arsitektur *Web Service*

Web *service* disusun oleh tiga komponen standart, yaitu :

1. *Simple Object Access Protocol* (SOAP), yaitu protokol yang bertanggungjawab dalam pertukuran informasi dalam lingkungan jaringan terdistribusi.
2. *Web service definition Language*(WDSL), dokumen standart yang dituliskan pada format XML, dan mendifinikasan kehadiran web *service* dalam suatu jaringan.
3. *Universal description, Discovery and Inttegration*(UDDI), yaitu suatu lokasi direktori yang beriskan *service* (layanan) dan bersifat bebas *platform (platform independent),* dituliskan berbasis XML dan dapat diakses oleh *entittas* yang berada di dalam dan luar jaringan.
4. Adanya standart tersebut membuat web *service* mudah di akses melalui berbagai antarmuka dan juga memberi peluang dimungkinkannya berbagai sistem yang dibangun pada *platform* yang berbeda dan bahasa yang berbeda untuk berkolaborasi pada dalam suatu pekerjaan [5].

2.8 Temperatur Air

Temperatur air merupakan faktor penting dalam membudidayakan ikan karena suhu rendah dibawah normal dapat menyebabkan ikan kehilangan nafsu makan dan lebih mudah terkena penyakit. Sebaliknya, jika suhu terlalu tinggi ikan dapat mengalami gangguan pernapasan dan bisa menyebabkan kerusakan insang permanen hingga kematian [6]. Temperatur yang cocok dalam membdidayakan ikan adalah 24^0C-28^0C, sedangkan perbedaan temperatur pada siang dan malam adalah 5^0C.

2.9 Sistem Aerasi

Penambahan udara dalam air diperlukan untuk meningkatkan kadar oksigen dalam air. Penambahan udara ini dapat dilakukan dengan menggunakan pompa air. Salah satu cara meningkatkan kontak dengan air yaitu dengan peralatan mekanis yang berfungsi untuk meningkatkan nilai oksigen yang masuk dalam air [6].

Fungsi pompa air antara lain :

1. Menambah oksigen secara langsung ke dalam air .
2. Mensirkulasi atau mencampur lapisa atas air atau permukaan air dengan dasar air untuk memastikan kandungan oksigen di dalam air benar-benar merata.
3. Memindahkan air yang telah teraerasi dengan cepat kearea sekelilignya sehingga belum teraerasi dapat teraerasi.
4. Dengan lapisan sedimen organik di dalam kolam, akan menciptakan permukaan yang teroksidasi gas-gas dan cairan beracun seperti hidrogen sulfida dan amonia tidak dapat masuk air.

2.10 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya[9]. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak

dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini beragam dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).

2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengodean.

3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Karena pendekatan pengembangan lebih dekat dengan dunai nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.10.1 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 tekniks 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 (*attribute*)

   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 kerja-nya.

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 Antarobjek

    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.

2.10.2 Pengenalan 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.

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

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 kebutuhan pemodelan 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.

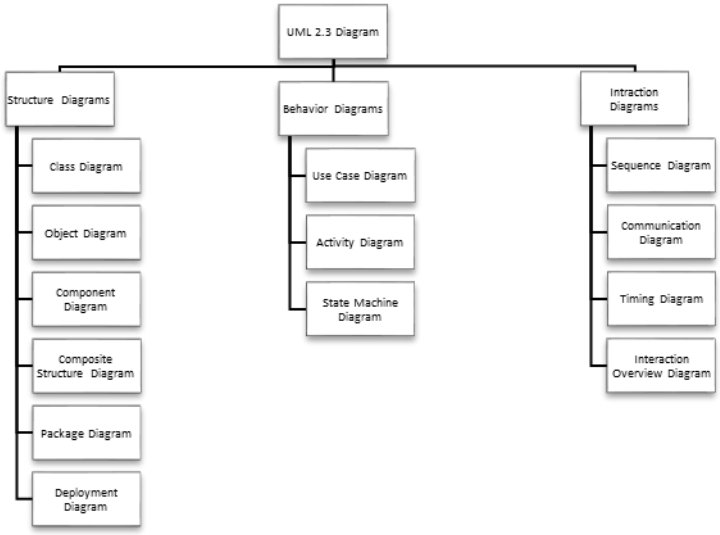
UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

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 merupakan hal yang sangat wajar.

2.10.3 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 2.10.3.1



Gambar 2. 7 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.

2.10.3.1 Use Case Diagram

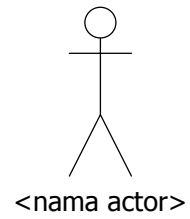
Use Case atau diagram *Use Case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *Use Case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *Use Case* yaitu pendefinisian apa yang disebut aktor dan *Use Case.*

2.10.3.1.1 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.

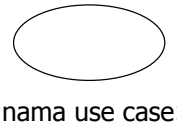
Dalam use case diagram terdapat satu aktor pemulai atau initiator actor yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor lain yang berpartisipasi atau participating actor. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut.



Gambar 2. 8 Bentuk Actor dalam UML

2.10.3.1.2 Use Case

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.

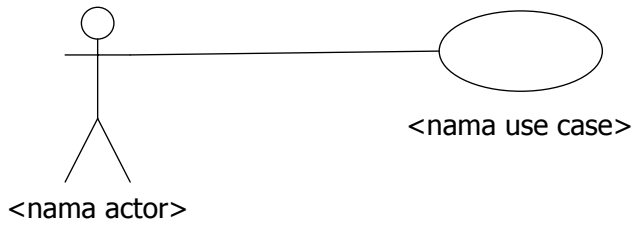


Gambar 2. 9 Bentuk Use Case dalam UML

2.10.3.1.3 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.



Gambar 2. 10 Bentuk Relationship dalam UML

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 ekstension 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>>.

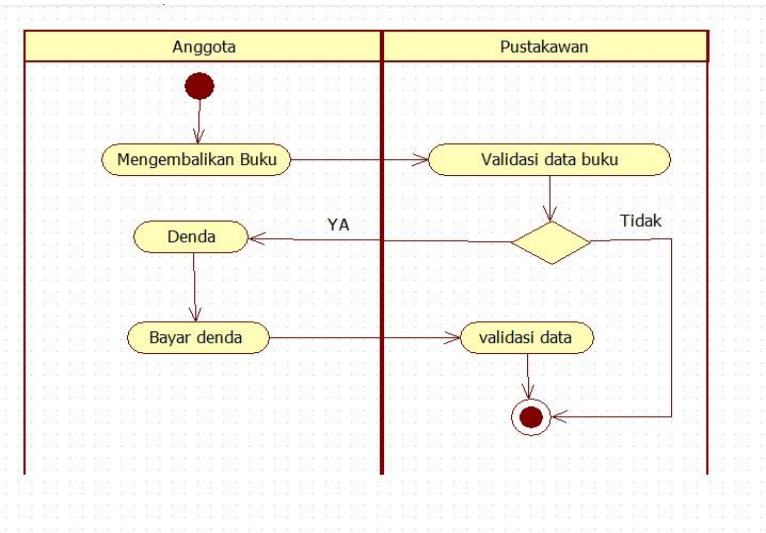
3. *Generalization/Inheritance*, dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan base-parent use case. Digambarkan dengan garis berpanah tertutup dari base use case ke parent use case.

2.10.3.2 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



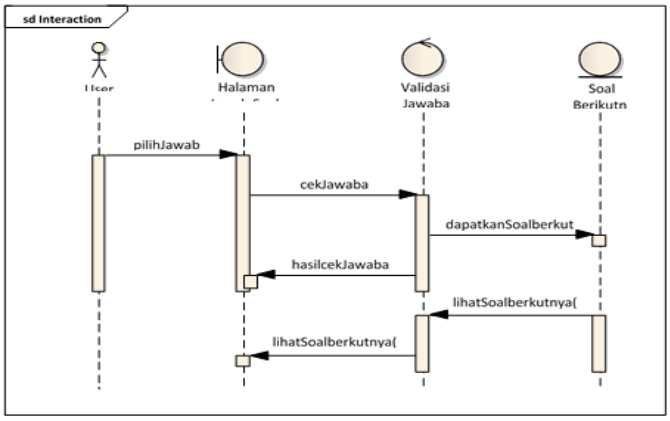
Gambar 2. 11 Contoh activity diagram

2.10.3.3 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.

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

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



Gambar 2. 12 Contoh Sequence diagram

2.10.3.4 *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.

Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

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. Pewarisan memungkinkan suatu kelas mewarisi semua atribut, operasi, relasi, dari kelas yang berada dalam hirarki pewarisannya.

2. *Associations*

   Hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui ekstensi *class* lain. Dalam notasi UML kita mengenal asosiasi 2 arah *(bidirectional)* dan 1 arah *(undirectional).*

3. *Aggregation*

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

4. *Composition*

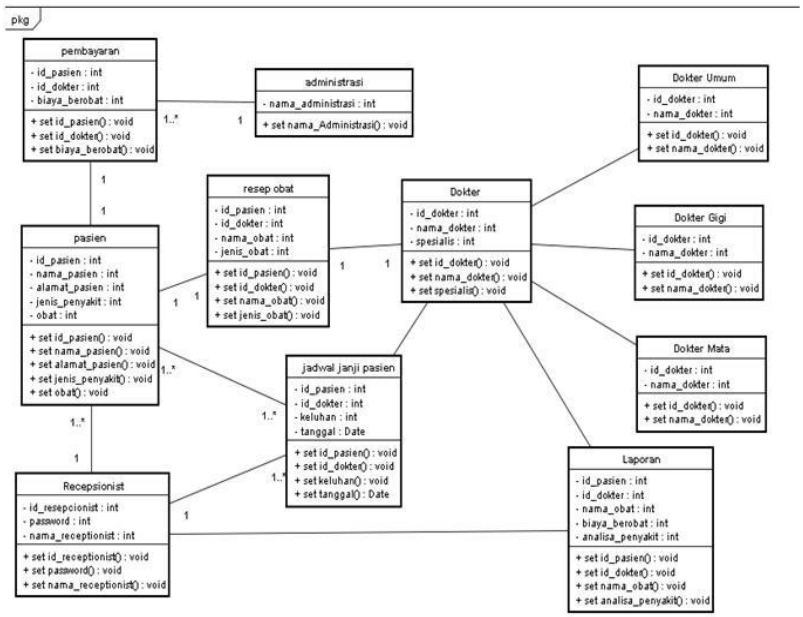
   Aggregation dengan ikatan yang lebih kuat. Di dalam *composite aggregation*, siklus hidup *part class* sangat bergantung pada *whole class* sehingga bila objek *instance* dari *whole class* dihapus maka objek *instance* dari *part class* juga akan terhapus.

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 [7].



Gambar 2. 13 Contoh gambar class diagram

2.11 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak (*software engineering*) merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomi yang dipercaya dan bekerja secara efisien menggunakan mesin. Perangkat lunak banyak dibuat dan pada akhirnya sering tidak digunakan karena tidak memenuhi kebutuhan pelanggan atau bahkan karena masalah non-teknis seperti keengan pemakai perangkat lunak (*user*) untuk mengubah cara kerja dari manual ke otomatis atau ketidakmampuan *user* menggunakan komputer. Oleh karena itu reakaya perangkat lunak

dibutuhkan agar perangkat lunak dibuat tidak hanya menjadi perangkat lunak yang tidak dipakai.

Rekayasa perangkat lunak lebih fokus pada praktik pengembangan perangkat lunak dan mengirimkan perangkat lunak yang bermanfaat kepada pelanggan *(customer).* Adapun ilmu komputer lebih fokus pada teori dan konsep dasar perangkat komputer. Rekayasa perangkat lunak lebih fokus pada bagaimana membuat perangkat lunak yang memenuhi kriteria berikut:

1. Dapat terus dipelihara setelah perangkat lunak selesai dibuat seiring berkembanya teknologi dan lingkungan *(maintainability)*
2. Dapat diandalkan dengan proses bisnis yang dijalankan dengan perubahan yang terjadi *(dependability rebust)*
3. Efisien dari segi sumber daya penggunaan
4. Kemampuan untuk dipakai sesuai dengan kebutuhan *(usability)*

Dari kriteria diatas maka perangkat lunak yang baik adalah perangkat lunak yang dapat memnuhi kebutuhan pelanggan (custumer) atau *user* (pemakai perangkat lunak) atau berorientasi pada pelanggan atau pemakai perangkat lunak, bukan berorientasi pada pembuat atau pengembang perangkat lunak.

Pekerjaan yang terkait dengan rekayasa perangkat dapat dikategorikan menjadi tiga buah kategori umum tanpa melihat area dari aplikasi, ukuran proyek perangkat lunak, atau kompleksitas perangkat lunak yang akan dibuat. Setiap fase dialamatkan pada satu atau lebih pertanyaan yang diajurkan sebelumnya.

Fase pendefinisian fokus pada *"what"* yang artinya harus mencari tahu atau mendefinisikan informasi apa yang harus diproses, seperti apa fungsi dan performasi yang diingikan, seperti apa perilaku sistem yang diingikan , apa kriteria validasi yang dibutuhkan untuk mendifinisikan sistem.

Fase pengembangan yang fokus dengan *"how"* yang artinya selama tahap pengembangan perangkat lunak seorang perekayasa perangakt lunak *(software engineer)* berusaha untuk mendifinisikan bagaimana data distrukturkan dan

bagaimana fungsi-fungsi yang dibutuhkan diimplementasikan didalam arsitektur perangkat lunak, bagaimana detail prosedural diimplementasikan, bagaimana karakter antarmuka tampilan, tampilan desain ditranslasikan ke bahasa pemograman dan bagaimana pengujian akan dijalankan.

Fase pendukung *(support phase)* fokus pada perubahan yang terasosiasi pada perbaikan kesalahan *(eror)* , adaptasi yang dibutuhkan pada lingkungan perangkat lunak yang terlibat dan perbaikan yang terjadi akibat perubahan kebutuhan pelanggan *(costumer).* Fase pendukung terdiri dari empat tipe perubahan antara lain

1. Koreksi *(correction)*

   Walaupun dengan jaminan kualitas yang terbaik, akan selalu ada kecacatan atua keinginan pelanggan *(customer)* yang tidak ada tertangani oleh perangkat lunak. Pemeliharaan dengan melakukan perbaikan terhadap kecacatan perangkat lunak.
2. Adaptasi (adaptation)

   Pada saat tertentu lingkungan asli (seperti CPU, sistem operasi, aturan bisinis, karakteristik produk luar) dimana perangkat lunak dikembangkan akan mengalami perubahan pemeliharaan adaptasi merupakan tahap untuk memodifikasi perangkat lunak guna mengakomodasi perubahaan lingkungan luar dimana perangakat luar dijalankan.
3. Perbaikan *(enchancement)*

   Sejalan dengan digunakannya perangkat lunak, maka pelanggan (*customer*) atau pemakai *(user)* akan mengenali fungsi tambahan yang dapat mendatangkan manfaat pemeliharaan perfektif atau penyempurnaan melakukan ekstensi atau penambahan pada keutuhan fungsional sebelumnya.
4. Pencegahaan *(prevention)*

   Keadaan perangkat lunak komputer sangat memungkinkan untuk perubhaan. Oleh karena itu, pemeliharaan pencegahan *(prevention)* atau sering disebut juga dengan rekayasa ulang sistem *(software reengineering)* harus dikonsisikan untuk mampu melayani kebutuhan pemakainya *(user).* Di lain

sisi biasanya setelah perangakat lunak dikirimkan ke *user* maka masih dibutuhkan asistensi dan help desk dari pengembang perangkat lunak

2.12 Perangkat Lunak Pendukung

Adapun beberapa perangkat lunak pendukung dalam penyusunan laporan Pembangunan *Smart Fish Farm* pada budidaya ikan dengan Android ini adalah

2.12.1 Eclipse

Eclipse adalah sebuah IDE (Integrated Development Environment) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform*-independent). Berikut ini adalah sifat dari Eclipse:

1. Multi-*platform*: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.

2. Multi-language: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, P erl, PHP, dan lain sebagainya.

3. Multi-role: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat. lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *Open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

2.12.2 Arsitektur Eclipse

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah kernel, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP:

1. Core *platform*
2. OSGi
3. SWT *(Standard Widget Toolkit)*
4. JFace
5. Eclipse Workbench

Secara standar Eclipse selalu dilengkapi dengan JDT *(Java Development Tools)*, *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in* Development Environment) untuk mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka *(open)*, mudah diperluas *(extensible)* untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak saja untuk mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (C/C++ Development Tools). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, *plug-in* UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya.

2.12.3 Perkembangan Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama Simultaneous Release. Setiap versi peluncuran terdiri dari Eclipse *Platform* dan juga sejumlah proyek yang terlibat dalam proyek Eclipse. Tujuan dari sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah deployment dan maintenance untuk sistem enterprise. Adapun versi Eclipse yang telah diluncurkan yaitu:

Gambar 2. 14 Tabel versi eclipse

No	Kode Peluncuan
1	Eclipse 3.0
2	Eclipse 3.1
3	Callisto
4	Europa
5	Ganymede
6	Galileo
7	Helios
8	Indigo
9	Juno
10	Kepler

2.12.4 Mysql

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (database management system) atau DBMS yang multithread, multi-user,dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

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 juga memiliki beberapa kelebihan, antara lain :

1. Portability
   MySQL dapat berjalan stabil pada berbagai sistem operasi seperti windows, Linux, FreeBSD, Solaris dan lain-lain.
2. Open Source
   MySQL didistribusikan secara open source (gratis), dibawah lisensi GPL sehingga dapat digunakan cuma-Cuma.
3. Multi User
   MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. Performance Tuning
   MySQL memiliki kecepatan yang menakjubkan dalam menangani query sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.

5. Coloumn Types

   MySQL memiliki tipe kolom yang sangat kompleks, seperti integer, double, char, text, datedan lain-lain.

6. Command and Function

   MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah select danwhere dalam query.

7. Security

   MySQL memiliki beberapa lapisan sekuritas seperti level subnetmask, nama host, dan izin akses user dengan sistem perizinan yang mendetail serta password terenkripsi.

8. Scability and Limits

   MySQL mampu menangani database dalam skala besar, dengan jumlah records lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

9. Connectivity

   MySQL dapat melakukan koneksi dengan clients menggunakan protokol TCP/IP, Unix socket (UNIX) atau Named Pipes (NT).

10. Localisation

    MySQL dapat mendeteksi pesan kesalahan pada client dengan menggunakan lebih dari dua puluh bahasa. Meskipun demikian, bahasa Indonesia belum termasuk didalamnya.

11. Interface

    MySQL memiliki interface (antar muka) terhadap berbagai aplikasi dan bahasa pemograman dengan menggunakan fungsi API (Aplication Programming Interface).

12. Clients and Tools

    MySQL dilengkapi dengan berbagai tools yang dapat digunakan untuk administrasi database dan pada setiap tool yang ada disertakan petunjuk online.

13. Struktur Tabel

    MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan database lainnya semacam PosgreeSQL ataupun Oracle.

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. Contoh fitur SUB-SELECT (nesting SELECT dalam SELECT) yang tidak optimal dan sering salah parsing query SQL dan jalan keluarnya dengan memecah menjadi beberapa query.

2.13 Android

Android adalah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar *(Smartphone)* dan komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti begerak [8]. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan piranti keras, piranti lunak dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile dan Nvidia. Beberapa pengertian lain dari Android, yaitu :

1. Merupakan *platform* terbuka (*Open source*) bagi para *programmer* untuk membuat atau mengembangkan aplikasi.

2. Merupakan sistem operasi yang dibeli Google Inc dari Android Inc.

3. Bukan bahasa pemrograman, akan tetapi hanya menyediakan lingkungan hidup atau *runtime environment* yang disebut DVM *(Dalvik Virtual Machine)* yang telah dioptimasi untuk device dengan sistem memori yang kecil.

2.13.1 Kelebihan Android

1. Multitasking, dimana Android mampu membuka beberapa aplikasi sekaligus tanpa harus menutup salah satunya.
2. Kemudahan dalam Notifikasi, setiap ada SMS, Email, atau bahkan artikel terbaru dari RSS Reader, akan selalu ada notifikasi di *Home Screen* Ponsel Android.
3. Akses Mudah terhadap Ribuan Aplikasi Android lewat Google Android App Market, kemudahan bagi pengguna yang gemar instal aplikasi ataupun *games*, lewat Google Android App Market maka bisa mengunduh berbagai aplikasi dengan gratis.
4. Pilihan Ponsel yang beranekaragam, ponsel Android akan terasa berbeda dibandingkan dengan iOS, jika iOS hanya terbatas pada iPhone dari Apple, maka Android tersedia di ponsel dari berbagai produsen, mulai dari Sony Ericsson, Motorola, HTC sampai Samsung. Dan setiap pabrikan ponsel pun menghadirkan ponsel Android dengan gaya masing-masing seperti Motorola dengan Motoblur-nya, Sony Ericsson dengan TimeScape-nya.
5. Bisa menginstal ROM yang dimodifikasi, jika tidak puas dengan tampilan standar Android, ada banyak Custom ROM yang bisa dipakai di ponsel Android.
6. *Widget* dapat dengan mudah mengakses berbagai *setting* dengan cepat dan mudah.

2.13.2 Kelemahan Android

1. Koneksi Internet yang terus menerus, kebanyakan ponsel berbasis sistem ini memerlukan koneksi internet yang simultan alias terus menerus aktif.

Koneksi internet GPRS selalu aktif setiap waktu, itu artinya harus berlangganan paket GPRS yang sesuai dengan kebutuhan.

2. Iklan Aplikasi di Ponsel Android memang bisa didapatkan dengan mudah dan gratis, namun konsekuensinya di setiap Aplikasi tersebut, akan selalu ada iklan yang terpampang, entah itu bagian atas atau bawah aplikasi.

2.14 Mikrokontroler Arduino

Arduino adalah kit elektronik atau papan rangkaian elektronik *Open source* yang di dalamnya terdapat komponen utama, yaitu sebuah chip mikrokontroler dengan jenis *Advanced Versatile RISC* (AVR) dari perusahaan Atmel. Mikrokontroler itu sendiri adalah chip atau *intergrated circuit* (IC) yang bisa di program menggunakan komputer. Tujuan menanamakan program pada mikrokontroler adalah agar rangkaian elektronik dapat membaca *input*, memproses *input* tersebut dan kemudian menghasilkan *output* sesuai dengan yang di inginkan. Jadi mikrokontroler sebagai "otak" *input*, *output* dan proses pada sebuah rangkaian elektronik [9].

Secara umum, Arduino terbagai dari dua bagian,yaitu:

1. *Hadware* berupa papan *input*/*output* (I/O) yang *Open source*.
2. *Software* arduino yang juga *Open source,* meliputi *software* arduino IDE untuk menulis program dan driver untuk koneksi dengan komputer.

2.14.1 Sejarah Arduino

Pembuatan Arduino di mulai pada tahun 2005, dimana sebuah situs perusahaan komputer *Oliveti* di *Ivrea* Italia, membuat perangakat untuk mengedalikan proyek desain interaksi siswa supaya lebih murah di bandingkan dengan sistem pada saat itu. Dilanjutkan pada bulan mei 2011, diamana sudah lebih dari 300.000 unit arduino terjual. Pendiri dari arduino itu sendiri adalah *Massimo Banzi* dan *David Cuarttielles* sebagai *Founder*. Awalnya mereka memberi nama proyek itu dengan sebutan arduin dari *Ivrea* tetapi seiring dengan perkembangan zaman, nama proyek itu di ubah menjadi Arduino yang berarti

“teman yang kuat” atau dalam versi inggrisnya dikenal dengan sebutan “*Hardwin*”.Proyek pengkabelan diciptakan oleh seniman sekaligus *programmer* asal kolombia bernama *Hernando Barragan.*

Pengkabelan ini adalah proyek tesis *Hernando* pada Desain Interaksi *Institute Ivrea*. Hal tersebut dimaksudkan untuk menjadi versi elektronik pengolahan yang digunakan di lingkungan pemrograman dan mengambil pola sintaks *Processing*. Dengan berkembangnya teknologi, Arduino menjadi sangat popular di kalangan mahasiswa dan pelajar saat ini. Mereka mengemabangkan Arduino dengan bootloder dan software yang user *friendly* sehingga menghasilkan sebuah board mikrokontroler yang bersifat *Open source* yang bisa di pelajari dan di kembangan oleh mahasiswa, pelajar, *professional*, pemula dan penggemar elektronika maupun robot di seluruh dunia *Integrated Development Environment* (IDE) diciptakan oleh *Cesey Reas* dan *Ben Fry*, beberapa *programmer* yang lain juga terlibat seperti *Tom Igoe, Gianluca Martino, David Mellis, dan Nicholas Zambett*.

2.14.2 Kelebihan Arduino

Tentu saja ada banyak mikrokontroler maupun *platform* mikrokontroler tersedia, misalnya saja *Basic* Stamp-nya *Parralax, BX-24-nya Netmedia, Phidget, MIT’s Handyboard,* dan lain sebagainya. Semua alat tersebut bertujuan untuk menyederhanakan berbagai macam kerumitan maupun detail rumit pada pemograman mikrokontroler sehingga menjadi paket mudah digunakan (*easy –to-use)* . Arduino juga menyederhanakan proses bekerja dengan mikrokontroler, sekaligus menawarkan berbagai macam kelebihan antara lain:

1. Murah, Papan (perangkat keras) Arduino biasanya di jual relatif murah dibandingkan dengan *platform* mikrokontoler lainnya. Jika ingin lebih murah lagi, tentu bisa di buat sendiri dan itu sangat mungkin sekali karena semua sumber website arduino bahkan di website-website komunitas arduino lainnya. Tidak hanya cocok untuk *Windows*, namun juga cocok bekerja di *linux, Mac.*

2. Sederhana dan mudah pemogramannya, Perlu diketahui bahwa lingkungan pemograman di arduino mudah di gunakan untuk pemula, dan cukup fleksibel bagi mereka yang sudah tinggkat lanjut. Untuk guru/dosen, Arduino berbasis pada lingkungan pemograman *Processing*, sehingga jika mahasiswa atau murid-murid terbiasa menggunakan *Processing* tentu saja kotak Arduino terdapat tulisan bahwa Arduino di pertuntukan bagi seniman, desainer, penghobi dan siapa saja. sungguh membesarkan hati dan membangkitkan semangat bahwa penggunaan tidak harus teknisi berpengalaman atau ilmuwan.
3. Perangkat lunaknya *Open souce.* Perangkat lunak Arduino IDE dipublikasikan sebagai *Open source*, tersedia bagi para pemogram berpengalaman untuk pengembangan lebih lanjut. Bahasanya bisa di kembangkan lebih lanjut melalui pustaka-pustaka C++ yang berbasis pada bahasa C dan AVR
4. Perangkat kerasnya *Open source*. Perangkat Keras Arduino berbasis mikrokontroler ATMEGA8, ATMEGA168, ATMEGA328 dan ATMEGA1280. Dengan demikian, siapa saja bisa membuat perangkat keras Arduino ini, apalagi bootloader tersedia langsung dari perangkat lunak arduino IDE-nya. Bisa juga menggunakan breadboard untuk membuat perangkat Arduino beserta *peripheral - peripheral* lain yang dibutuhkan.
5. Tidak perlu perangkat *chip programmer*. Karena di dalamya sudah ada bootlader yang akan menangani upload program dari komputer
6. Sudah memiliki sarana komunikasi USB. Sehingga pengguna laptop yang tidak memiliki port serial/RS323 bisa menggunakannya.
7. Bahasa pemograman relatif murah, karena *software* Arduino dilengkapi dengan kumpulan *library* yang cukup lengkap.
8. Memiliki *module* siap pakai (*shiled*) yang bisa ditancapkan pada board Arduino Misalnya *shield GPS, Ethernet, SD Card,* dan lain-lain.

2.14.3 Arduino Ethernet Shield

Arduino *Ethernet Shield* merupakan modul sambungan internet. Arduino akan terhubung ke internet dalam beberapa menit dengan beberapa instruksi dan dapat melakukan pengendalian melalui internet. Arduino *ethernet shield* berbasis chip ethernet wiznet W5100.



Gambar 2. 15 Arduino Ethernet

Wiznet W5100 merupakan jaringan *provider* (IP) yang mendukung TCP dan UDP. Dengan menggunakan library ethernet untuk penulisan/upload sketch. Modul *ethernet shield* digunakan untuk terhubung dengan internet[1].

2.14.4 Relay

Prinsip Kerja Relay - Relay adalah Saklar *(Switch)* yang dioperasikan secara listrik dan merupakan komponen *Electromechanical* (Elektromekanikal) yang terdiri dari 2 bagian utama yakni Elektromagnet (Coil) dan Mekanikal (seperangkat Kontak Saklar/*Switch*). Relay menggunakan Prinsip Elektromagnetik untuk menggerakkan Kontak Saklar sehingga dengan arus listrik yang kecil (low power) dapat menghantarkan listrik yang bertegangan lebih tinggi. Sebagai contoh, dengan Relay yang menggunakan Elektromagnet 5V dan 50 mA mampu menggerakan Armature Relay (yang berfungsi sebagai saklarnya) untuk menghantarkan listrik 220V 2A.