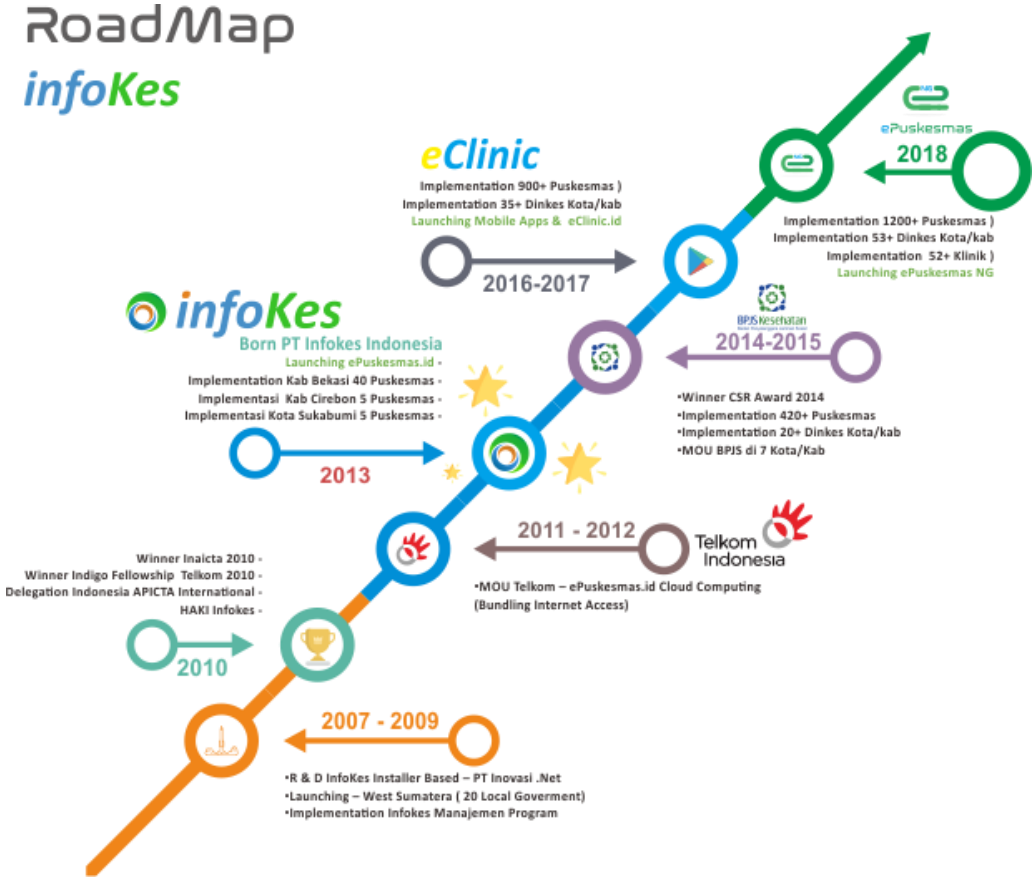
BAB 2
LANDASAN TEORI

2.1 PT Infokes Indonesia

PT. Infokes Indonesia (Infokes) adalah perusahaan Teknologi Informasi yang berfokus pada pengembangan produk dan solusi Teknologi Informasi Kesehatan online dan terpadu di Indonesia. Perusahaan ini beralamat di Komp. Palm Bridge No. 1-E Jl. Cukang Kawung - Cikutra Barat, Bandung. Lebih dari 1 dekade PT Infokes telah dipercaya untuk membantu meningkatkan kualitas layanan kesehatan di Indonesia dengan menerapkan lebih banyak sistem di 1000 titik Puskesmas yang tersebar di seluruh Indonesia secara realtime. Dengan dukungan semua lapisan masyarakat, PT Infokes telah mencapai berbagai prestasi di kancah nasional dan internasional.



Gambar 2.1 *Roadmap* Sejarah PT Infokes

2.1.1 Logo, Visi dan Misi PT Infokes

Visi :

Menjadi penyedia layanan solusi Teknologi Informasi Kesehatan Online dan Terintegrasi di Indonesia (Integrated eHealth Solutions Platform)

Gambar 2.2 Logo PT Infokes

Misi :

PT Infokes memiliki tiga misi yang setiap misinya diwakili oleh warna dari logo PT Infokes.

1. Warna Hijau - Manajemen Pasien

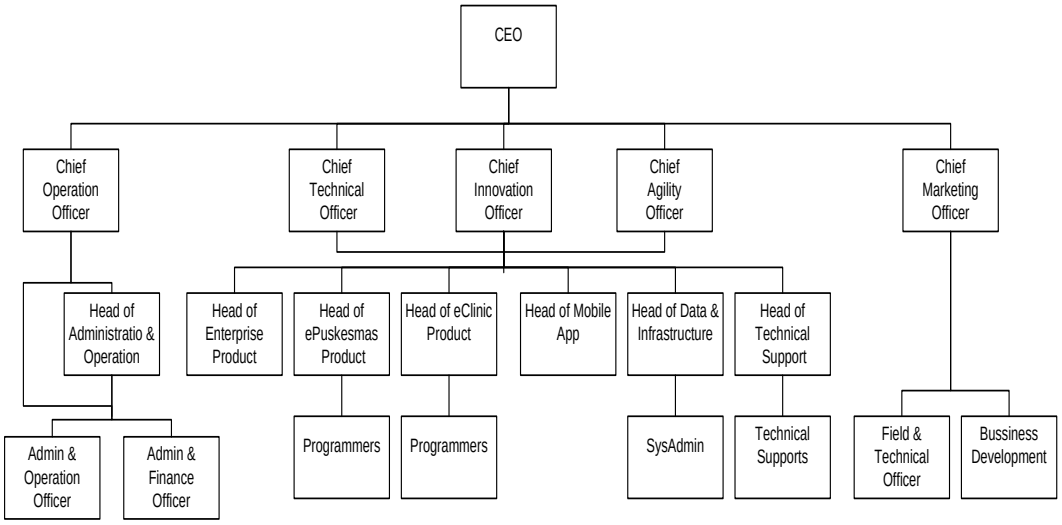
   Meningkatkan mutu pelayanan kesehatan masyarakat menggunakan sarana Teknologi Informasi dan komunikasi (TIK) untuk mewujudkan pelayanan prima
2. Warna Biru - Manajemen Program

   Monitoring data dan pelayanan kesehatan untuk memudahkan melakukan pengambilan keputusan tentang kondisi kesehatan masyarakat terkini.
3. Warna Oranye - Manajemen Organisasi

   Menggunakan sarana teknologi Informasi dan Komunikasi berbasiskan *paperless office* dalam lingkungan internal pemerintahan untuk menjalankan organisasi kepemerintahan yang baik dan profesional.

2.1.2 Struktur Organisasi

Struktur organisasi PT Infokes adalah sebagai berikut:



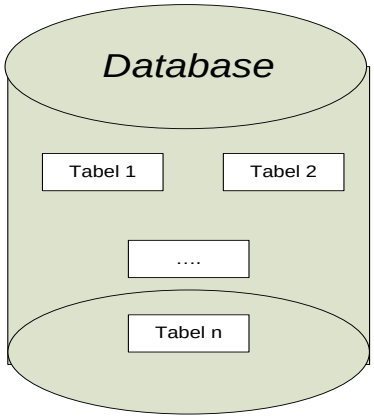
Gambar 2.3 Struktur Organisasi PT Infokes

2.2 *Database*

Secara umum, *database* berarti koleksi data yang saling terkait. Secara praktis, *database* dapat dianggap sebagai suatu penyusun data yang terstruktur yang disimpan dalam media pengingat (*hard disk)* yang tujuannya adalah agar data tersebut dapat diakses dengan mudah dan cepat.

Ada beberapa macam *database*, antara lain yaitu *database* hierarkis, *database* jaringan dan *database* relasional. *Database* relasional merupakakn *database* yang populer saat ini dan telah diterapkan pada berbagai platform, dari PC hingga minikomputer.

Sebuah *database* relasional tersusun atas sejumlah tabel. Sebagai contoh, *database* akademis mencakup tabel-tabel seperti dosen, mahasiswa, KRS, nilai dan lain-lain. *Database* tentang bintang film bisa mencakup info pribadi (nama, jenis kelamin, tanggal lahir, dan sebagainya) dan film-film yang pernah dibintangi [13].



Gambar 2.4 Konsep Umum *Database* [13]

2.2.1 Komponen Penyusun *Database*

Dalam suatu *database*, terdiri dari beberapa komponen yang menyusunnya yaitu [14]:

1. *Entity*

Entity disebut juga dengan Tabel. *Entity* adalah orang, tempat, kejadian aatau konsep yang informasinya direkam.

2. *Atribute*

Setiap *entity* mempunyai *atribute* atau sebutan untuk mewakili suatu *entity*. *Atribute* juga disebut sebagai *data elemen*, *data field*, *data item.*

3. *Data Value* (Nilai atau Isi Data)

Data value adalah data aktual atau informasi yang disimpan pada tiap data elemen atau atribute.

4. *Record/Tuple*

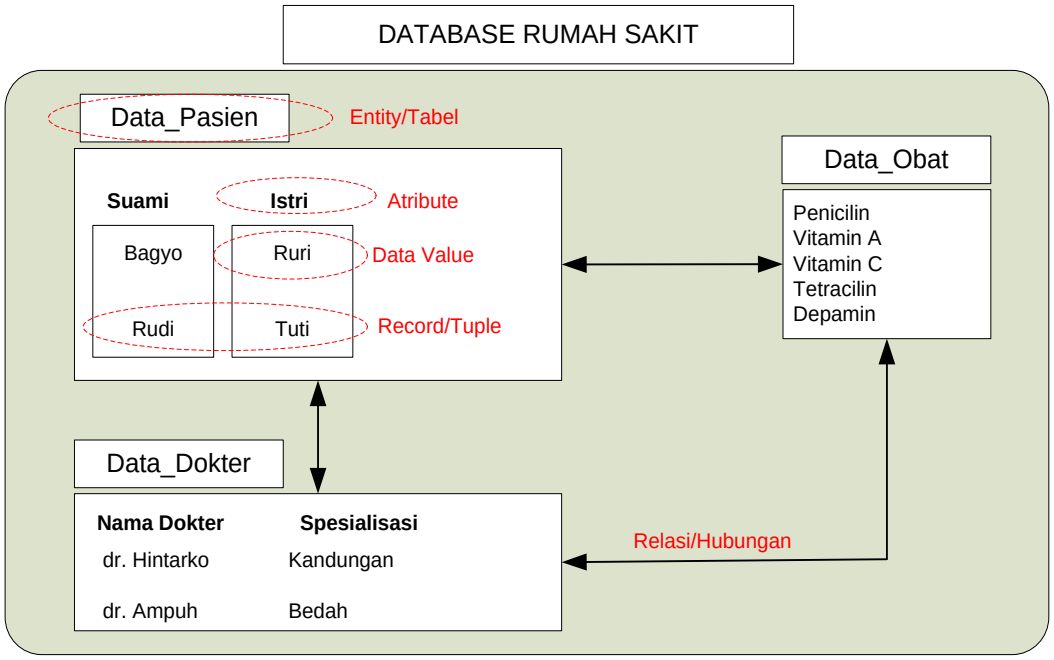
Record/tuple merupakan kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu *entity* secara lengkap. Satu *record* mewakili satu data atau informasi.

5. Relasi/Hubungan

Pada model *database* relasional, kaitan atau asosiasi antara dua buah tabel disebut hubungan *(relationship).* Hubungan dapat berupa:

1) 1-1, yakni satu data pada suatu tabel berpasangan dengan hanya satu data pada tabel lain

2) 1-M, yakni satu data pada suatu tabel berpasangan dengan banyak data pada tabel lain.



Gambar 2.5 Contoh *Database*

Kumpulan file yang saling berkaitan bersama dengan program untuk pengelolaannya disebut sebagai *Database Management System* (DBMS). *Database* adalah kumpulan datanya, sedang program pengelolaannya berdiri sendiri dalam satu paket program yang komersial untuk membaca data, mengisi data, menghapus data, melaporkan data dalam *database*.

2.2.2 Fungsi *Database*

Penyusunan suatu *database* digunakan untuk mengatasi masalah-masalah pada penyusunan data yaitu [14]:

1. Redudansi dan inkonsistensi data dalam beberapa file.

Redudansi merupakan kondisi di mana terdapat data yang sama tersimpan di dalam beberapa tempat (duplikasi data). Hal ini mengakibatkan pemborosan ruang penyimpanan dan juga biaya untuk mengakses jadi lebih tinggi. Penyimpanan data yang sama berulang-ulang di beberapa file dapat mengakibatkan juga inkonsistensi (tidak konsisten).

2. Kesulitan pengaksesan data

Pada suatu saat dibutuhkan suatu proses untuk menampilkan data tertentu namun belum tersedia program untuk mengeksekusinya. Dengan adanya DBMS, data mampu diambil secara langsung dengan bahasa yang familiar dan mudah digunakan (*user friendly)*

3. Isolasi data untuk standarisasi

Jika data tersebar dalam beberapa file dalam bentuk format yang tidak sama, maka ini menyulitkan dalam menyusun program aplikasi terutama proses mengambil dan menyimpan data. Maka haruslah data dalam satu *database* dibuat satu format sehingga mudah dibuat program aplikasinya.

4. *Multiple user* (banyak pemakai)

Dalam rangka mempercepat semua daya guna sistem dan mendapat responsi waktu yang cepat, beberapa sistem mengizinkan banyak pemakai untuk meng"*update*" data secara simultan. Salah satu alasan mengapa *database* dibangun karena nantinya data tersebut digunakan oleh banyak orang dalam waktu yang bervariasi.

5. Masalah keamanan (*security*)

Tidak setiap pemakai sistem *database* diperbolehkan untuk mengakses semua data. Misalkan data mengenai gaji pegawai hanya boleh dibuka oleh bagian keuangan dan personalia, tidak diperkenankan bagian gudang membaca dan mengubahnya.

6. Masalah integrasi (kesatuan)

Dengan adanya *database*, antara data satu sama lain yang berhubungan dapat dikaitkan. Secara teknis pengaitan tersebut dilakukan menggunakan *field* kunci.

7. Masalah *data independence* (kebebasan data)

Paket bahasa yang diciptakan oleh DBMS, perubahan pada struktur *file*/tabel, setiap kali kita hendak melihat data cukup dengan *utility list*, menambah data dengan *Append* (misal untuk DBMS Clipper atau Foxpro), merubah struktur tabel dengan Design Table, melakukan penelurusan data dengan *query* (misal untuk Access, Sql Server, MySql atau Oracle). Ini berarti perintah-perintah dalam paket

DBMS bebas terhadap *database*. Apapun perubahan dalam *database*, semua perintah akan mengalami kestabilan tanpa perlu ada yang diubah.

2.3 Kemanan Jaringan

Kemanan jaringan adalah kumpulan peranti yang dirancang untuk melindungi data ketika transmisi terhadap ancaman pengaksesan, pengubahan dan penghalangan oleh pihak yang tidak berwenang [5].

2.3.1 Ancaman Keamanan

Ancaman keamanan yang terjadi terhadap informasi adalah [15]:

1. *Interruption*: merupakan suatu ancaman terhadap *availability* informasi. Data yang ada dalam sistem komputer dirusak atau dihapus sehingga jika data atau informasi tersebut dibutuhkan maka pemiliknya akan mengalami kesulitan untuk mengaksesnya, bahkan mungkin informasi itu hilang.
2. *Interception*: merupakan ancaman terhadap kerahasiaan (*secrecy*). Informasi disadap sehingga orang yang tidak berhak dapat mengakses kokmputer di mana informasi tersebut disimpan.
3. *Modification*: merupakan ancaman terhadap integritas. Orang yang tidak berhak berhasil menyadap lalu-lintas informasi yang sedang dikirim dan kemudian mengubahnya sesuai keinginan orang tersebut.
4. *Fabrication*: merupakan ancaman terhadap integritas. Orang yang tidak berhak berhasil meniru atau memalsukan informasi sehingga orang yang menerima informasi tersebut menyangka bahwa informasi tersebut berasal dari orang yang dikehendaki oleh si penerima informasi.

Tabel 2.1 Ancaman Terhadap Keamanan [15]

System	***Availability***	***Secrecy***	***Integrity***
Hardware	Dicuri atau dirusak		
Software	Program dihapus	Software dicopy	Program dimodifikasi
Data	File dihapus atau dirusak	Dicuri, disadap	File dimodifikasi
***Line* Komunikasi**	Kabel diputus	Informasi disadap	Informasi dimodifikasi

2.3.2 Layanan Kemanan Jaringan

Layanan-layanan keamanan jaringan didefinisikan berdasarkan kebutuhan yang harus disediakan untuk memenuhi permintaan terhadap keamanan jaringan. Berikut adalah pembahasan jenis-jenis layanan keamanan jaringan [5, 15]:

1) Otentikasi (*Authentication*)

Layanan otentikasi beertujuan agar penerima informasi dapat memastikan keaslian pesan, bahwa pesan itu datang dari orang yang dimintai informasi. Dalam layanan otentifikasi terdapat dua layanan di dalamnya. Layanan pertama disebut dengan otentikasi entitas (*entity authentication*) yaitu layanan yang memberikan kepastian terhadap identitas sebuah entitas yang terlibat dalam komunikasi data. Sedangkan layanan kedua disebut dengan otentikasi terhadap keaslian data (*data origin authentication)* yaitu layanan yang memastikan sumber dari sebuah data.

2) Kendali Akses (*Acces Control*)

Kendali akses adalah layanan keamanan jaringan yang menghalangi penggunaan tidak terotorisasi terhadap sumber daya. Pada aplikasi jaringan biasanya kebijakan kemampuan (baca, modifikasi, tulis dan eksekusi sebuah data/layanan sistem) ditentukan oleh jenis pengguna. Misalnya sebuah data rekam medik elektronik hanya dapat diakses oleh pasien dan paramedis yang terlibat. Kendali askses seringkali dilakukan dengan menggunakan kombinasi *user id* dan *password* ataupun dengan mekanisme lain.

3) Kerahasiaan Data *(Confidentiality)*

Kerahasiaan data merupakan layanan keamanan jaringan yang memproteksi data tertransmisi terhadap pengungkapan oleh pihak yang tidak berwenang. Atau bisa didefinisikan juga bahwa layanan kerahasiaan data merupakan usaha untuk menjaga informasi dari orang yang tidak berhak mengakses. Kerahasiaan ini biasanya berhubungan dengan informasi yang diberikan ke pihak lain.

4) Keutuhan Data *(Integrity)*

Keutuhan data adalah layanan keamanan jaringan yang memastikan bahwa data yang diterima oleh penerima adalah benar-benar sama dengan data yang dikirim oleh pengirim. Ada dua jenis layanan keutuhan data yaitu keutuhan data dengan pemulihan dan tanpa pemulihan.

5) Non-Repudiation

Layanan *Non-Repudiation* adalah layanan keamanan jaringan yang berhubungan dengan pengirim. Dalam hal ini bertujuan untuk menghindari penolakan atas penerimaan/pengiriman data yang telah terkirim. Sehingga pengirim tidak dapat mengelak bahwa dialah yang mengirim informasi.

6) Ketersediaan (*Availability*)

Layanan ketersediaan adalah layanan sistem yang membuat sumber daya sistem tetap dapat diakses dan digunakan ketika ada permintaan dari pihak yang berwenang. Serangan terhadap sistem seperti *denial of services*. Sistem informasi yang diserang atau dijebol dapat menghambat atau meniadakan akses ke informasi.

2.3.3 Serangan Kemanan Jaringan

Sistem keamanan jaringan yang dioperasionalkan pada jaringan publi rentan terhadap serangan oleh siapapun. Orang yang berusaha meruntuhkan keamanan jaringan disebut *attacker* (penyerang). Secara umum serangan pada sistem keamanan jaringan dapat dikategorikan jadi 2 jenis serangan, yaitu [5] :

2.3.3.1 Serangan Pasif

Penyerang hanya mengumpulkan data yang melintas pada jaringan publik (jaringan yang bisa diakses oleh penyerang). Serangan pasif tidak melakukan modifikasi data yang melintasi atau merusak sistem, penyerang hanya membaca saja (*read only*). Karena tidak melakukan perubahan data dan menggangu sistem, serangan pasif lebih pada pencegahan daripada pendeteksian. Berikut ini beberapa jenis serangan yang digolongkan sebagai serangan pasif :

1) ***Snooping***

Kegiatan yang bermaksud mendapatkan data yang tengah terkirim pada jaringan, biasanya melalui akses yang tak berwenang.

2) ***Traffic Analysis***

Kegiatan serangan pasif dengan melakukan *monitoring* terhadap lalu lintas data pada jaringan. Data-data lalu lintas jaringan dikumpulkan dan kemudian dianalisis sehingga penyerang dapat mengetahui maksud data-data itu.

2.3.3.2 Serangan Aktif

Sebuah serangan aktif (*active attack*) dapat mengakibatkan perubahan data yang terkirim dan jalannya sistem terganggu. Pada serangan aktif seakan-akan penyerang memperoleh kemampuan untuk mengubah data pada lalu lintas data selain kemampuan baca. Jenis-jenis serangan aktif adalah sebagai berikut :

1) ***Masquerade*** adalah serangan aktif yang dilakukan oleh penyerang dengan cara penyerang mengambil alih (menirukan) perilaku pengirim atau penerima.
2) ***Modification*** adalah serangan aktif yang dilakukan oleh penyerang dengan cara penyerang mengambil alih jalur komunikasi untuk mengubah atau menghapus atau menunda pesan yang sedang terkirim untuk keuntungan penyerang.
3) ***Replay*** adalah serangan aktif yang terdiri atas pencatatan secara pasif data unit dan transmisi ulang untuk menimbulkan efek yang diinginkan penyerang.
4) ***Denial of Service*** adalah serangan aktif yang bertujuan agar sistem menjadi *collapse* sehingga tidak mampu memberikan respons atau layananyang semestinya kepada pengguna. Serangan ini biasanya dilakukan dengan membuat server menjadi *overload* dengan permintaan palsu (*dummy*).

Untuk mengembangkan sistem keamanan jaringan yang aman, perancang kemanan jaringan harus menganalisis kemungkinan serangan-serangan atas layanan keamanan jaringan. Biasanya sebuah sistem keamanan jaringan dikatakan aman bila sistem itu mampu bertahan terhadap serangan aktif.

2.4 Kriptografi

Kriptografi berasal dari bahasa Yunani, *kripto* dan *graphia. Crypto* berarti *secret* (rahasia) dan *graphia* berarti *writing* (tulisan). Menurut terminologinya kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika dikirim dari suatu tempat ke tempat yang lain [16].

Untuk menjaga pesan, maka pesan tersebut dapat diubah menjadi suatu kode yang tidak dapat dimengerti oleh pihak lain. Algoritma kriptografi terdiri dari tiga fungsi dasar, yaitu [15]:

1) Enkripsi

Merupakan hal yang sangat penting dalam kriptografi, merupakan pengamanan data yang dikirimkan agar terjaga kerahasiaannya. Pesan asli disebut *plaintext*, yang diubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan dengan *cipher* atau kode. Sama halnya dengan jika tidak mengerti akan sebuahkata maka yang dilakukan adalah dengan melihatnya di dalam kamus atau daftar istilah. Beda halnya dengan enkripsi, untuk mengubah teks asli ke bentuk teks-kode digunakan algoritma yang dapat mengkodekan data yang diinginkan.

2) Dekripsi

Merupakan kebalikan dari enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (*plaintext*), disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda atau kebalikan dengan algoritma yang digunakan untuk enkripsi.

3) Kunci

Kunci yang dimaksud di sini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, kunci rahasia (*private key*) dan kunci umum (*public key*).

4) *Plaintext*

Plaintext merupakan pesan yang ditulis atau diketik yang memiliki makna. *Plaintext* inilah yang diproses menggunakan algoritma kriptografi untuk menjadi *ciphertext*.

5) *Ciphertext*

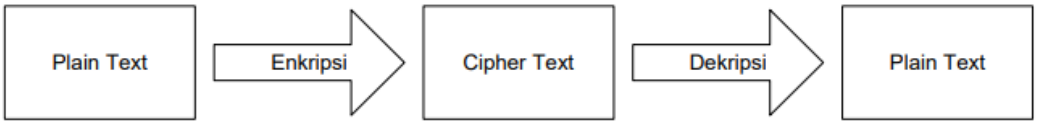
Merupakan suatu pesan yang telah melalui proses enkripsi. Pesan yang ada pada *ciphertext* ini tidak bias dibaca karena berupa karakter-karakter yang tidak mempunyai makna (arti).

6) Kriptanalisis *(Cryptanalysis)*

Kriptanalisis merupakan suatu analisis kode atau suatu ilmu untuk mendapatkan *plaintext* tanpa harus mengetahui kunci sah secara wajar. Jika suatu *ciphertext* berhasil diubah menjadi *plaintext* tanpa menggunakan kunci yang sah, proses tersebut dinamakan *breaking code*. Hal ini dilakukan oleh para kriptanalis. Analisis kode juga dapat menemukan kelemahan dari suatu algoritma kriptografi

dan akhirnya dapat menemukan kunci atau *plaintext* dari *ciphertext* yang dienkripsi dengan algoritma tertentu.

Maka pesan atau data asli sebelum dienkripsi disebut *plaintext*. Sedangkan pesan yang sudah diacak disebut *ciphertext*. Proses pengubahan *plaintext* menjadi *ciphertext* disebut dengan enkripsi, sedangkan proses pengubahan *ciphertext* kembali menjadi *plaintext* disebut dengan dekripsi.



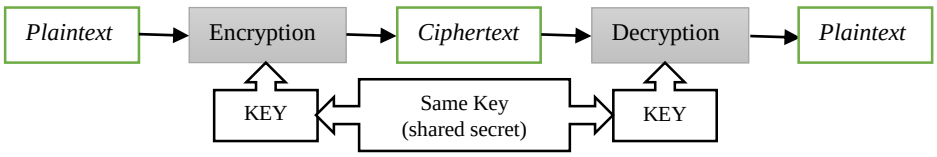
Gambar 2.6 Proses Enkripsi – Dekripsi [15]

2.5 Macam-macam Algoritma Kriptografi

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakainya [15]:

2.5.1 Algoritma Simetris

Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsi. Algoritma ini sudah ada sejak lebih dari 4000 tahun yang lalu. Bila mengirimkan pesan dengan menggunakan algoritma ini, si penerima pesan harus diberitahu kunci dari pesan tersebut agar bisa mendekripsikan pesan yang dikirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci. Jika kunci tersebut diketahui oleh orang lain maka orang tersebut akan dapat melakukan enkripsi dan dekripsi terhadap pesan.



Gambar 2.7 Proses Algoritma Simetris [15]

Pada dasarnya ada dua tipe dasar algoritma simetris:

2.5.1.1 Block Cipher

Algoritma *block cipher* merupakan algoritma yang masukan dan keluarannya berupa satu blok dan setiap blok terdiri dari beberapa bit (misalnya 1 blok terdiri dari 64 bit atau 128 bit). *Blok cipher* mempunyai banyak aplikasi. Aplikasi tersebut digunakan untuk memberikan layanan confidentiality (kerahasiaan) integritas data atau authentication (pengesahan pemakai) dan juga bisa memberikan layanan *keystream generator* untuk *stream cipher*.

Misalkan blok *plaintext* (P) yang berukuran m bit dinyatakan sebagai vektor:

$P = (p_1, p_2, \ldots, p_m)$

Yang dalam hal ini p_i adalah 0 atau 1 untuk i = 1, 2, …, m, dan blok cipher text (C) adalah

$C = (c_1, c_2, \ldots, c_m)$

Yang dalam hal ini c_i adalah 0 atau 1 untuk i = 1, 2, …,m.

Bila *plaintext* dibagi menjadi n buah blok, barisan blok *plaintext* dinyatakan sebagai:

$(P_1, P_2, \ldots, P_n)$

Untuk setiap blok *plaintext* Pi, bit-bit peyusunnya dapat dinyatakan sebagai vektor:

$P_i = (p_{i1}, p_{i2}, \ldots, p_{im})$

Enkripsi dan dekripsi dengan kunci K dinyatakan berturut-turut dengan persamaan enkripsi:

$$E_k(P) = C \quad (1)$$

Dan persamaan dekripsinya:

$$D_k(C) = P \quad (2)$$

Fungsi E haruslah fungsi yang berkorespondensi satu ke satu, sehingga:

$$E^{-1} = D \quad (3)$$

Contoh: *Database* di sebuah bank memiliki kata INCOME, dienkripsi menjadi satu blok. Itu berarti semua rekening memiliki blok enkripsi yang sama. Misalnya, *database* bank memiliki dua record:

MEMBER: HOLLY INCOME $100,000

MEMBER: HEIDI INCOME $100,000

Hasil enkripsinya menjadi:
ABCQZRME GHQMRSIB CTXUVYSS RMGRPFQN
ABCQZRME ORMPABRZ CTXUVYSS RMGRPFQN

Penyerang bisa menyimpulkan bahwa *record* yang ada di dalam *database* tersebut memiliki nilai yang sama. Kesimpulan itu diambil dari kata CTXUVYSS RMGRPFQN. Serangan dilakukan terhadap *block cipher* dengan melihat posisi dan ukuran blok. Untuk meminimalkan serangan terhadap block cipher digunakan beberapa cara agar posisi dan ukurannya menjadi tidak sama. Banyak mode blok kode yang dapat dipakai. Mode operasai block cipher terbagi menjadi empat bagian:

1. Mode Electronic Code Book (ECB)
2. Mode Cipher Block Chaining (CBC)
3. Mode Cipher Feed Back (CFB)
4. Mode Output Feed Back (OFB)

Dari setiap mode tersebut *plaintext* dibagi menjadi blok-blok. Misalnya setiap blok mempunyai nilai 128 bit. Jika terdapat *plaintext* yang dibagi ke bentuk blok yang mempunyai nilai kurang dari 128 bit maka blok tersebut terlebih dahulu ditambah dengan pandding bit (bit tambahan) agar jumlahnya menjadi 128 bit untuk setiap bloknya.

Contoh dari Algoritma Simetri Tipe Block Cipher:

1. Data Encryption Standard (DES)
2. Advance Encryption Standard (AES),
3. RC2, RC5, RC6,
4. Blowfish, dan lain sebagainya

2.5.1.2 *Stream cipher*

Stream cipher (aliran cipher) merupakan suatu *cipher* yang berasal dari hasil XOR. Setiap bit *plaintext* dengan setiap bit kunci. Kunci merupakan kunci utama (kunci induk) yang digunakan untuk membangkitkan kunci acak semu yang dibangkitkan dengan *Pseudo-Random Sequence Generator* yang merupakan suatu nilai yang nampak seperti diacak, tetapi sesungguhnya nilai tersebut merupakan suatu urutan. Secara khusus urutan dari nilai yang dihasilkan oleh *RNG (random number generator), computational mekanisme deterministic* atau FSM (*Finite State Machine*) merupakan kebalikan dari *really random*.

Random Number Generator (RNG) secara umum adalah *Pseudo-Random* yang memberikan *initial state* atau *seed* (nilai yang diinput ke dalam *state*), seluruh urutan tersebut ditentukan secara keseluruhan, tetapi meskipun demikian banyaknya karakteristik yang ditampilkan dari suatu urutan yang acak tersebut Pseudo-Randomness menghasilkan urutan yang sama secara berulang-ulang pada penempatan yang berbeda. Kemudian kunci acak semu tersebut diberikan operasi XOR dengan *plaintext* untuk mendapatkan *ciphertext*.

Stream cipher rawan terhadap serangan pembalikan bit. Jika penyerang mengetahui bahwa *stream cipher* digunakan, penyerang mencoba untuk mengidentifikasi posisi bit yang telah diubah dan mengembalikannya ke bentuk aslinya. Penyerang terlebih dahulu mencari pola dari perubahan bit tersebut untuk mengidentifikasi bentuk asli dari bit tersebut. Contoh Mr. X mentransfer uang antar rekening di sebuah bank dengan jumlah 10 USD dalam sekali transaksi.

Pesan dari pengirim:

Plaintext : QT-TRNSFER USD $000010, 00 FRM ACCNT 1234567 TO

Ciphertext : aMz0rapltxMfpUn7**U**xOrtLm42ZuweeM0qaPtI7wEptAnxfL

Pesan dimodifikasi penyerang:

Ciphertext : aMz0rapltxMfpUn7**T**xOrtLm42ZuweeM0qaPtI7wEptAnxfL

Plaintext : QT-TRNSFER USD $100010, 00 FRM ACCNT 1234567 TO

Jika penyerang bisa memahami *stream cipher* lebih baik maka, dia dapat melakukan serangan pembalikan bit yang akan mengakibatkan pesan dapat diterjemahkan dan dimodifikasi seperti contoh di atas. Pada perjalanan pesan yang

dikirim Mr X yang ditransfer dari sebuah rekening di sebuah bank, di tengan jalan dapat diinterupsi oleh penyerang. Bit “1” dari karakter “U” dimodifikasi menjadi “0” yang mengakibatkan karakter “U” berubah menjadi karakter “T” dan setelah sampai ke tujuan dilakukan dekripsi. Maka, akan didapat bahwa yang sebenarnya uang yang ditransfer 10 USD menjadi 100010 USD. Serangan jenis ini sangat merugikan bila terjadi seperti hal di atas. Pada stream cipher tidak perlu mengetahui kunci yang digunakan untuk dapat melakukan enkripsi sama sekali, dan yang perlu diketahui adalah letak dari posisi bit yang ada [16].

Contoh dari Algoritma Simetris Tipe *Stream cipher* adalah:

1. One Time Pad (OTP)
2. RC4
3. A5, dan lain sebagainya.

2.5.2 Algoritma Asimetris

Algoritma asimetris sering juga disebut dengan algoritma kunci publik, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsi berbeda. Pada algoritma asimetris kunci terbagi menjadi dua bagian yaitu [15]:

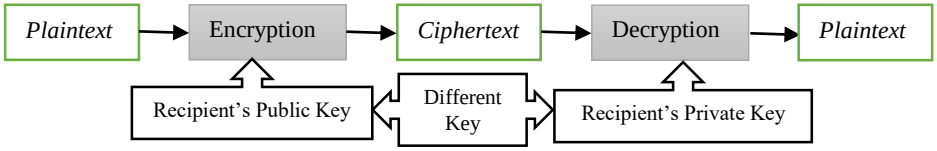
1. Kunci umum (*public key*): Kunci yang boleh semua orang tahu (dipublikasikan).
2. Kunci rahasia (*private key*): Kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang).

Kunci-kunci tersebut berhubungan satu sama lain. Dengan kunci publik orang dapat mendekripsi pesan tetapi tidak bisa mendekripsinya. Hanya orang yang memiliki kunci rahasia yang dapat mendekripsi pesan tersebut. Algoritma asimetris bias mengirimkan pesan dengan lebih aman daripada algoritma simetris.

Contoh algoritma yang menggunakan kunci asimetris adalah [15]:

1. Digital Signature Algorihtm (DSA)
2. RSA
3. Diffle-Hellman (DH)
4. Elliptic Curve Cryptography (EGC)

5. Kriptografi Quantum, dan lain sebagainya.



Gambar 2.8 Proses Algoritma Simetris [15]

2.5.3 Fungsi Hash Satu Arah

Fungsi Hash adalah sebuah fungsi yang masukannya adalah sebuah pesan dan keluaran sebuah sidik pesan (*message fingerprint*). Sidik pesan sering juga disebut *message digest*. Fungsi hash dapat digunakan untuk mewujudkan beberapa layanan keamanan jaringan misalnya untuk keutuhan data dan otentikasi pesan. Fungsi hash yang banyak dipakai yaitu MD5 dan SHA [5].

Fungsi hash satu arah juga dikenal sebagai rangkuman atau fungsi kompresi adalah fungsi matematis yang mengambil input panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap. Lebih jauh, fungsi hash satu arah dirancang dengan cara yang sulit untuk membalik proses, yaitu untuk menemukan rangkaian pada nilai tertentu (karena itu dinamai satu arah). Fungsi hash yang baik juga membuatnya sulit untuk menemukan 2 string yang akan menghasilkan nilai hash yang sama [7].

2.6 Algoritma RC4

Sistem sandi RC4 dikembangkan oleh Ronald Rivest pada tahun 1987 merupakan algoritma *stream cipher* yang paling banyak digunakan. Misalnya pada protokol SSL/TLS. RC4 merupakan *stream cipher* yang berorientasi *byte*. Masukan algoritma enkripsi RC4 merupakan sebuah *byte*, kemudian dilakukan operasi XOR dengan sebuah *byte* kunci, dan menghasilkan sebuah *byte* sandi [5, 7].

Untuk enkripsi model RC4 terdapat beberapa kelemahan maupun kelebihan masing-masing. Adapun kelebihan dan kelemahan algoritma itu antara lain [7]:
Kelebihan:

1. Kesulitan mengetahui sebuah nilai dalam tabel
2. Kesulitan mengetahui lokasi mana di dalam table yang digunakan untuk menyeleksi masing-masing nilai
3. Kunci RC4 tertentu hanya dapat digunakan sekali
4. Model enkripsi ini 10 kali lebih cepat dari DES

Kekurangan:

1. Algoritma RC4 mudah diserang dengan menggunakan analisis dari bagian dalam tabel
2. Salah satu dari 256 kunci dapat menjadi kunci yang lemah. Kunci ini diidentifikasi oleh kriptoanalisis yang dapat menemukan keadaan di mana salah satu dari bit yang dihasilkan mempunyai korelasi yang kuat dengan sedikit bit kun ci.

Pada Algoritma RC4, *ciphertext* diperoleh dengan melakukan penjumlahan modulo 2 satu bit *plaintext* dengan satu bit kunci

$$c_i = (p_i + k_i) \bmod 2 \quad (4)$$

yang dalam hal ini,

p_i : bit *plaintext*

k_i : bit kunci (*key*)

c_i : bit *ciphertext*

Plaintext diperoleh dengan melakukan penjumlahan modulo 2 satu bit *ciphertext* dengan satu bit kunci:

$$p_i = (c_i - k_i) \bmod 2 \quad (5)$$

Oleh karena operasi penjumlahan modulo 2 identik dengan operasi bit dengan operator XOR maka persamaan proses enkripsi dapat ditulis sebagai:

$$c_i = p_i \oplus k_i \quad (6)$$

dan proses dekripsi menggunakan persamaan:

$$p_i = c_i \oplus k_i \quad (7)$$

Proses enkripsi algoritma RC4 yang merupakan algoritma *stream cipher*, bit hanya mempunyai dua buah nilai sehingga proses enkripsi hanya menyebabkan dua

keadaan pada bit tersebut, berubah atau tidak berubah. Dua keadaan itu ditentukan oleh kunci enkripsi yang disebut aliran-bit-kunci (*keystream*).

Keystream dibangkitkan dari sebuah pembangkit yang dinamakan pembangkit aliran-bit-kunci (*keystream generator*). *Keystream* di-XOR-kan dengan bit-bit *plaintext*, p_1, p_2, ….,p_i untuk menghasilkan aliran bit *ciphertext*.

Contoh:

Plaintext : 1100101

Keystream : 1000110

Ciphertxt : 0100011

2.7 Algoritma Base64

Base64 adalah sebuah standar penyandian (*encoding*). *Encoding* merupakan sebuah metode yang bertujuan untuk merubah bentuk atau format data. Beberapa contoh dari *encoding* adalah base64. Base64 berawal dari surat elektronik (*email*). Pada waktu itu, email dikirim dengan protokol SMTP *(Simple Mail Transfer Protocol)* ke *mail server* pengirim, lalu dikirim ke *mailbox* penerima di *mail server* tujuan. "Protokol" adalah tata cara mesin (komputer) saling berkomunikasi via jaringan. Supaya *email* bisa sampai ke penerima, ia harus mengunduhnya terlebih dahulu. Proses mengunduh email menggunakan protokol POP *(Post Office Protocol).* Saat ini POP sudah mencapai versi 3 sehingga disebut POP3. Alternatif yang lebih baik dari POP adalah IMAP *(Internet Mail Access Protocol),* yang saat ini sudah mencapai versi 4.

Baik POP maupun SMTP adalah protokol berbasis teks. *Encoding* yang digunakan adalah ASCII. Tidak masalah bila seseorang hanya ingin mengirim email teks saja. Masalah muncul ketika email berkembang, memerlukan adanya kemampuan untuk mengirim lampiran (*attachment*). Apa yang dilampirkan adalah file, dan file ini bisa file apa saja, termasuk file biner.

POP dan SMTP memiliki persamaan dalam hal terminasi pesan. Keduanya menggunakan deretan karakter [CR LF . CR LF] *(carriage return – line feed – tanda titik – carriage return – line feed)* sebagai akhir dari pesan. Apa yang terjadi bila di

dalam file biner terdapat *byte-byte* berikut: [0D 0A 2E 0D 0A]? Nilai rangkaian *byte* tadi adalah kode ASCII dari [CR LF . CR LF] sehingga server akan menganggap pesan yang dikirim berhenti sampai di sana. File yang dilampirkan oleh pengirim akan putus di tengah. Sehingga untuk mengatasi hal tersebut dibuat Base64 [17].

Transformasi Base64 merupakan salah satu algoritma untuk *encoding* dan *decoding* suatu data ke dalam format ASCII, yang didasarkan pada bilangan dasar 64 atau bisa dikatakan sebagai salah satu metode yang digunakan untuk melakukan *encoding* (penyandian) terhadap data *binary*. Karakter yang dihasilkan pada transformasi Base64 ini terdiri dari A..Z, a..z dan 0..9, serta ditambah dengan dua karakter terakhir yang bersimbol yaitu + dan / serta satu buah karakter sama dengan (=) yang digunakan untuk penyesuaian dan menggenapkan data binary atau istilahnya disebut sebagai pengisi pad.

Tabel 2.2 Tabel Indeks Base64 [4]

Indeks (data 6 bit)	**Karakter *Encoding* Base64**	**Indeks (data 6 bit)**	**Karakter *Encoding* Base64**	**Indeks (data 6 bit)**	**Karakter *Encoding* Base64**	**Indeks (data 6 bit)**	**Karakter *Encoding* Base64**
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	A	42	q	58	6
11	L	27	B	43	r	59	7
12	M	28	C	44	s	60	8
13	N	29	D	45	t	61	9
14	O	30	E	46	u	62	+
15	P	31	F	47	v	63	/
						pad	=

2.8 Pengujian Sistem

Pengujian perangkat lunak adalah sebuah elemen sebuah topik yang memiliki cakupan luas dan sering dikaitkan dengan verifikasi dan validasi. Verifikasi mengacu pada sekumpulan aktivitas yang menjamin bahwa perangkat lunak mengimplementasikan dengan benar sebuah fungsi yang spesifik. Validasi mengacu pada sekumpulan aktivitas yang berbeda yang menjamin bahwa perangkat lunak yang dibangun dapat ditelusuri sesuai dengan kebutuhan pelanggan (customer) [18].

Pengujian untuk validasi memiliki beberapa pendekatan sebagai berikut:

2.8.1 Pengujian *Black Box*

Yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Untuk melakukan pengujian *black box* diperlukan kasus uji yang dibuat untuk mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan atau tidak. Kasus uji yang dibuat dalam pengujian *black box* memerlukan kasus benar dan kasus salah [18]. Sebagai contoh:
Kasus proses *login*, kasus uji:
a. Jika pengguna memasukkan *username* dan *password* benar
b. Jika pengguna memasukkan *username* dan *password* salah, misalnya *username* benar tapi *password* salah, atau sebaliknya, atau keduanya salah.

2.8.2 Pengujian *White Box*

Yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan dan keluaran yang sesuai dengan spesifikasi kebutuhan.

Pengujian *white box* dilakukan dengan memeriksa logika dari kode program. Pembuatan kasus uji bisa mengikuti standar pengujian dari standar pemrograman yang seharusnya [18].

2.8.3 Pengujian Kemanan *Database*

Pengujian keamanan *database* dilakukan untuk mengetahui aman atau tidaknya *database* yang dikirimkan. Pengujian ini menggunakan perangkat lunak yaitu *CrypTool* versi 1.4.41 dan *Wireshark* versi 3.0.1.

CrypTool merupakan perangkat lunak untuk melakukan kriptografi maupun *cryptanalysis* yang dapat diunduh secara gratis dari portal *CrypTool* di www.cryptool.org. Pada awalnya *CrypTool* dirancang sebagai perangkat lunak internal dalam Deutsche Bank untuk menunjang pelatihan keamanan pada sisi teknologi informasi. *CrypTool* kemudian mengalami perkembangan menjadi proyek *open-source* yang mengakibatkan pengguna semakin banyak sehingga membantu pengembang meningkatkan kualitas *CrypTool* melalui masukan-masukan yang diberikan pengguna [19].

Wireshark adalah sebuah aplikasi yang dapat menangkap paket-paket jaringan dan berusaha untuk menampilkan semua informasi di paket tersebut sedetail mungkin. Aplikasi jenis ini merupakan *Network Packet Analyzer* yang diumpamakan sebagai alat untuk memeriksa apa yang sebenarnya sedang terjadi di dalam jaringan [20].

2.9 Internet

Internet adalah kelompok atau kumpulan dari jutaan komputer. Penggunaan internet memungkinkan untuk mendapatkan informasi dari komputer yang ada di dalam kelompok tersebut dengan asumsi bahwa pemilik komputer memberikan izin akses. Untuk mendapatkan sebuah informasi, sekumpulan protokol harus digunakan, yaitu sekumpulan aturan yang menetapkan bagaimana suatu informasi dapat dikirim dan diterima.

2.9.1 Sejarah Internet

Internet pertama kali digunakan sebagai proyek penelitian yang ditemukan oleh Advanced Research Project Agency (ARPA) Department of Defense (DOD) di Amerika Serikat. Pada dasarnya, internet digunakan untuk menghubungkan komputer. Versi yang pertama disebut ARPANET. Pada tahun 1972, ARPA berubah menjadi DARPA dengan tetap mempromosikan proyek ARPANET.

Pengembangan internet dengan jenis peralatan yang berbeda, namun bisa saling berhubungan satu sama lain merupakan tantangan yang besar pada saat itu. Pada tahun 1973-1974, ada penelitian yang merancang sebuah Transmission Control Protocol/Internet Protocol (TCP/IP). Pada awalnya, TCP/IP dimaksudkan untuk menyediakan dukungan untuk kebutuhan berikut [21]:

1. Interoperabilitas antarsistem heterogen,
2. Komunikasi end-to-end berbagai jaringan yang berbeda, dan
3. Operasi otomatis dan sempurna di dalam menghadapi terjadinya kegagalan hubungan data.

Pada awal tahun 1980-an, ARPANET dipecah menjadi dua bagian, yaitu MILNET dan ARPANET karena pertimbangan keamanan. Pihak militer berjalan terus dengan MILNET, sedangkan penelitian, pengembangan, dan sektor lain tetap memakai ARPANET. Pada pertengahan tahun 1980-an, National Science Foundation (NSF) di Washington, D.C. mendistribusikan teknologi internet kepada beberapa universitas. Selanjutnya, internet pun mulai menyebar di dunia.

Pada tahun 1990, DOD memutuskan untuk membubarkan ARPANET dan menggantikannya dengan pendukung (backbone) NSFNET), bekerja sama dnegan agen jaringan lain. Hal inilah yang kemudian menjadi prinsip pendukung jaringan internet.

2.10 *World Wide Web*

World Wide Web (WWW) atau yang sering juga disebut sebagai “Web” saja merupakan sebuah sistem dnegan informasi yang disajikan dalam bentuk teks, gambar, suara dan lain-lain yang tersimpan dalam sebuah *web server* internet yang disajikan dalam bentuk hiperteks. Informasi web dalam bentuk teks umumnya ditulis dalam format HTML (*Hypertext Markup Language*). Informasi lainnya disajikan dalam bentuk grafis (dalam format GIF, JPG, PNG), suara (dalam format AU, WAV) dan objek multimedia lainnya (seperti MIDI, Shockwave, Quicktime, Movie, 3D World).

Web dapat diakses oleh prangkat lunak client web yang disebut browser. Browser membaca halaman-halaman web yang tersimpan dalam *web server* melalui protokol yang disebut HTTP (*Hypertext Transfer Protocol*).

Sebagai dokumen hiperteks, dokumen-dokumen pada web dapat memiliki tautan link, baik yang tersimpan dalam *web server* yang sama maupun pada *web server* lainnya. Tautan memudahkan para pengakses web berpindah dari satu halaman ke halaman lainnya dan dari satu server ke server lainnya. Kegiatan penelusuran halaman web ini biasa disebut browsing, namun ada juga yang menyebutnya surfing (berselancar) [21].

2.10.1 Sejarah *World Wide Web*

Pada Thaun 1991, Tim Bernes-Lee mengembangkan visi untuk *Network Information Project pada le Centre Europeén de Recherche Nucléaire* (CERN) di Swiss. Misinya adalah untuk menciptakan sistem informasi global yang mudah, namun kuat berdasarkan pada hiperteks. Dua bagian utama yang muncul dari proyek ini adalah *HyperText Markup Language* (HTML) dan *HyperText Transfer Protocol* (HTTP).

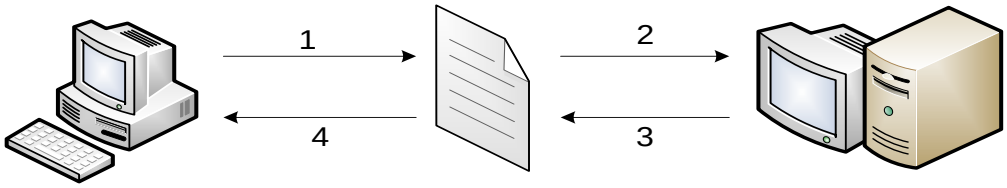
HTML adalah bahasa markup untuk menyebarkan informasi pada web. Ketika merancang HTML, ide ini diambil dari Standard General Markup Language (SGML). SGML adalah cara yang terstandarisasi dari pengorganisasian dan informasi yang terstruktur di dalam dokumen atau sekumpulan dokumen.

HTTP adalah komponen sentral lainnya dari proyek awal di CERN. HTTP adalah protokol komunikasi stateless yang berbasiskan pada TCP yang awalnya digunakan untuk mengambil kembali file-file HTML dari *web server* ketika dirancang pada tahun 1991.

HTML dan HTTP telah dikembangkan lebih lanjut sejak pertama kali keduanya diusulkan. World Wide Web Consortium (W3C) pada awal Oktober 1994 dan temuan Tim Berners-Lee telah menyatu dan memimpin evolusi teknis dari Web [21].

2.10.2 Arsitektur Web Tradisional

Ada dua komponen dasar di dalam arsitektur web, yaitu browser web dan *web server*. *Web browser* menawarkan antarmuka grafis untuk pengguna dan bertanggung jawab untuk komunikasi dengan *web server*. Protokol komunikasi antara *browser* dan *web server* mengikuti protokol HTTP yang distandarisasi.



Gambar 2.9 Interaksi Antara Pengguna dan *Web server* [21]

Penjelasan interaksi antara pengguna dan *web server* adalah sebagai berikut:

1. Pengguna meminta suatu layanan dengan mengklik tautan (*link*) atau dengan mengetikkan sebuah perintah dengan *keyboard. Web browser* menangkap perintah tersebut dan menerjemahkannya ke dalam permintaan HTTP.
2. *Browser* kemudian meneruskan permintaan yang baru saja diciptakan kepada *server web* dari penyedia konten. Ketika *server* menerima sebuah permintaan, permintaan tersebut akan diproses.
3. Ketika pemrosesan dilakukan, *web server* kemudian mengirimkan kembali respon tersebut ke *browser*.
4. Ketika *browser* menerima respon tersebut, *browser* menerjemahkannya ke dalam bentuk yang dapat dibaca oleh manusia.

2.10.3 Aplikasi *Web*

Aplikasi *web* adalah sebuah sistem informasi yang mendukung interaksi pengguna melalui antarmuka berbasis *web*. Fitur-fitur aplikasi *web* biasanya berupa *data persistence,* mendukung transaksi dan komposisi halaman *web* dinamis.
Interaksi web dibagi dalam tiga langkah, yaitu:

1) Permintaan

Pengguna mengirimkan permintaan ke *web server,* biasanya via halaman *web* yang ditampilkan pada *web browser*.

2) Pemrosesan

Web server menerima permintaan yang dikirimkan oleh pengguna, kemudian memproses permintaan tersebut.

3) Jawaban

Browser menampilkan hasil dari permintaan pada jendela *browser*.

Penyebaran aplikasi untuk *desktop* mempunyai keuntungan yang jelas ketika ada perluasan dan ketika dibutuhkan, yaitu saat aplikasi mampu beroperasi tanpa koneksi jaringan. Aplikasi *web* mempunyai akses yang lebih luas sebagai komputer dengan *browser* dan koneksi internet yang dapat mengakses situs tanpa harus menginstal sebuah *client* secara terpisah untuk masing-masing aplikasi.

Tabel 2.3 Perbandingan Aplikasi *Web* dan *Desktop*

Fitur	**Aplikasi *Web***	**Aplikasi *Desktop***
Grafis	Terbatas	Tidak terbatas (namun harus sesuai spesifikasi perangkat)
Interaksi pengguna	Luas	Tidak terbatas
Pengguna jaringan	Tinggi	Beragam
Dapat diakses dari	Perangkat mana saja (PC atau Mobile) asal terdapat web browsernya.	Komputer yang terinstal
Fungsionalitas perbaikan	Pada server	Pada desktop
Popularitas	Meningkat	Dominan

2.11 HTTPS (*HyperText Transfer Protocol)*

HTTPS merupakan HTTP yang menggunakan SSL (*Secure Socket Layer*). SSL adalah protokol enkripsi yang dipanggil melalui web server dengan menggunakan HTTPS. Penggunaan protokol SSL pada jaringan sangat penting untuk mengamankan data di dalam sambungan jaringan.. SSL merupakan jenis *sockets communications* yang berada di antara *transmision control protocol/internet protocol* (TCP/IP) dan *application layer* [22]. Selain SSL ada pula protokol TLS (*Transport Layer Security*) yang menjadi sublayer HTTPS. Sehingga HTTPS disebut juga kombinasi dari HTTP dan SSL/TLS.

2.12 PHP

PHP secara umum dikenal sebagai bahasa pemrograman *script* yang membuat dokumen HTML secara *on the fly* yang dieksekusi di *server web*, dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan editor teks atau editor HTML. PHP dikenal juga sebagai bahasa pemrograman *server side*. Bahasa ini dibuat pertama kali pada musim gugur tahun 1994 oleh Rasmus Lerdoff, awalnya digunakan pada *website*nya untuk mencatat siapa saja yang berkunjung dan melihat biodatanya.

PHP/FI merupakan nama awal dari PHP. PHP pada mulanya kependekan dari *Personal Home Page* dan FI merupakan *Form Interface*. PHP awalnya merupakan program CGI *(Common Gateway Interface)* yaitu suatu standar uang menghubungkan aplikasi eksternal dengan *server web*, yang dikhususkan untuk menerima masukan melalui form yang ditampilkan dalam *browser web*. Perangkat lunak ini disebarkan dan dilisensikan sebagai perangkat lunak *open source.*

Integrasi PHP dengaan *server web* dilakukan dengan teknik CGI, Fast CGI dan modul server web. Teknik CGI dan FastCGI memisahkan antara *server web* dan PHP; sedangkan modul *server web* menjadi PHP sebagai bagian dari *server web*. Saat ini PHP merupakan kependekan dari PHP: *HyperText Preprocessor*, merupakan bahasa utama *script server side* yang disisipkan pada HTML yang dijalankan di *server* dan juga bisa digunakan untuk membuat aplikasi *desktop* [23].

2.12.1 Perkembangan Versi PHP

Berikut adalah ringkasan perkembangan versi PHP [23]:

Tabel 2.4 Perkembangan Versi PHP

Versi Mayor	Versi Minor	Tanggal Keluar	Catatan
1	1.0.0	8 Juni 1995	Versi PHP yang pertama kali dikeluarkan, yang disebut sebagai *Personal Home Page Tools (PHP Tools).*
2	2.0.0	1 November 1997	Dianggap sebagai *tool* yang paling cepat dan mudah untuk membuat halaman *web* yang dinamik.
3	3.0.0	6 Juni 1998	Pengembangan *tools* berpindah dari satu orang menjadi beberapa orang. Zeev Suraski dan Andi Gutmans menulis ulang dasar dan versi ini.

Versi Mayor	Versi Minor	Tanggal Keluar	Catatan
	3.0.18	20 Oktober 2000	Versi terakhir dari 3.0.x yang dikeluarkan
4	4.0.0	22 Mei 2000	Engine Zend yang digunakan untuk melakukan 2 tahap sistem parsing dan eksekusi *tag*
	4.1.0	10 Desember 2001	Mengenalkan variabel 'superglobal' ($_GET, $_POST, $_SESSION, dll.)
	4.2.0	22 April 2002	Konfigurasi register_globals secara *default* di*disable*. Data yang diterima dari jaringan tidak disisipkan secara langsung ke dalam namspace global lagi. Konfigurasi ini dimaksudkan untuk menutup lubang keamanan yang ada.
	4.3.0	27 Desember 2002	CLI *(Command Line Interface)* diperkenalkan, sebagai tambahan dari CGI
	4.4.0	11 Juli 2005	Halaman man di lingkungan *nix/Linux disediakan untuk *script php* dan php-config
	4.4.9	7 Agustus 2008	Peningkatan keamanan dan perbaikan bug. Versi terakhir dari versi 4.4
5	5.0.0	13 Juli 2004	Zend Engine II dengan model objek baru
	5.1.0	24 November 2005	Peningkatan kinerja dengan mengenalkan variabel kompiler dalam re-engineering engine PHP. Penambahan PDO (PHP Data Object) sebagai antarmuka untuk pengaksesan database yang konsisten.
	5.2.0	2 November 2006	Dukungan JSON secara default
	5.2.17	6 Januari 2011	Perbaikan kerentanan yang kritis terhubung pada floating point
	5.3.0	30 Juni 2009	Dukungan namspace, late static, bindings, Jump label (limited goto), Native closures, Native PHP archives (phar), garbage collection untuk referensi circular, peningkatan dukungan untuk Windows, sqlite3, mysqlnd, sebagai ganti libmysql untuk librari untuk bekerja dengan MySQL, fileinfo sebagai pengganti mime_magic untuk dukungan MIME yang lebih baik, ekstensi ereg.
	5.3.1	19 November 2009	Lebih dari 100 *bug* diperbaiki, beberapa di antaranya berhubungan dengan keamanan
	5.3.2	4 Maret 2010	Perbaikan banyak *bug*.
	5.3.3	22 Juli 2010	Perbaikan pada *bug* dan keamanan; peningkatan FPM SAPI

Versi Mayor	Versi Minor	Tanggal Keluar	Catatan
	5.3.4	10 Desember 2010	Perbaikan pada *bug* dan keamanan; peningkatan FPM SAPI
	5.3.5	6 Januari 2011	Perbaikan pada kerentanan yang terhubung dengan floating point.
	5.3.6	10 Maret 2011	Lebih dari 60 *bug* yang diperbaiki dari laporan versi sebelumnya
	5.3.7	18 Agustus 2011	Release ini memfokuskan pada peningkatan stabilitas PHP 5.3.x, lebih dari 90 bug yang diperbaiki, beberapa berhubungan dengan keamanan.
	5.3.8	23 Agustus 2011	Perbaikan pada isu yang ada pada versi PHP 5.3.7
	5.4.0 RC4	22 Desember 2011	Penghapusan item: register_globals, safe_mode, allow_call_time_pass_reference, session_register(), session_unregister() dan session_is_registered(). Beberapa peningkatan untuk fitur yang ada.
6	?	Masih belum diketahui	Masih belum diketahui akan dikeluarkan, ada keterlambatan dan penundaan pada pengembangannya. Akan tetapi *update* untuk versi 6 ini telah ditambahkan mulai versi 5.3.0

2.13 SQL (*Structured Query Language*)

SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengelola data pada RDBMS (*Relational Database Management System*). RDBMS sendiri merupakan suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola dan menampilkan data. DBMS versi *open source* yang cukup berkembang dan paling banyak digunakan saat ini adalah: MySQL, PostgreSQL, Firebird, SQLite. Hampir semua DBMS mengadopsi SQL sebagai bahasa untuk mengelola data pada DBMS.

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. Berikut sejarah perkembangan SQL sampai saat ini [18]:

Tabel 2.5 Perkembangan SQL [18]

No	Tahun	Nama
1	1986	SQL-86
2	1989	SQL-89
3	1992	SQL-92
4	1999	SQL:1999
5	2003	SQL:2003
6	2006	SQL:2006
7	2008	SQL:2008
8	2011	SQL:2011

Berikut ini adalah contoh pengaksesan data pada DBMS dengan SQL yang secara umum terdiri dari 4 hal sebagai berikut [18]:

1. Memasukkan data (*insert*)

```
INSERT INTO Tabel_mahasiswa
(nim, nama, tanggal_lahir)
VALUES
    ('10114900','Alif','1997-01-01');
```

Query di atas digunakan untuk memasukkan data mahasiswa dengan NIM 10114900, nama Alif dan tanggal lahir 1 Januari 1997 ke tabel "Tabel_mahasiswa".

2. Mengubah data (*update*)

```
UPDATE Tabel_mahasiswa
SET
    Tanggal_lahir='1996-01-01';
WHERE
Nim='10114900';
```

Query di atas digunakan untuk mengubah data tanggal lahir mahasiswa dengan NIM = 10114900 menjadi 1 Januari 1996 dalam tabel "Tabel_mahasiswa".

3. Menampilkan data (*select*)

```
SELECT nim, nama
FROM Tabel_mahasiswa
WHERE
nim='10114900';
```

Query di atas digunakan untuk menampilkan data mahasiswa yang tersimpan dalam "Tabel_mahasiswa" dengan NIM = 10114900.

4. Menghapus data (*delete*)

```
DELETE FROM Tabel_mahasiswa
WHERE
Nim='10114900';
```

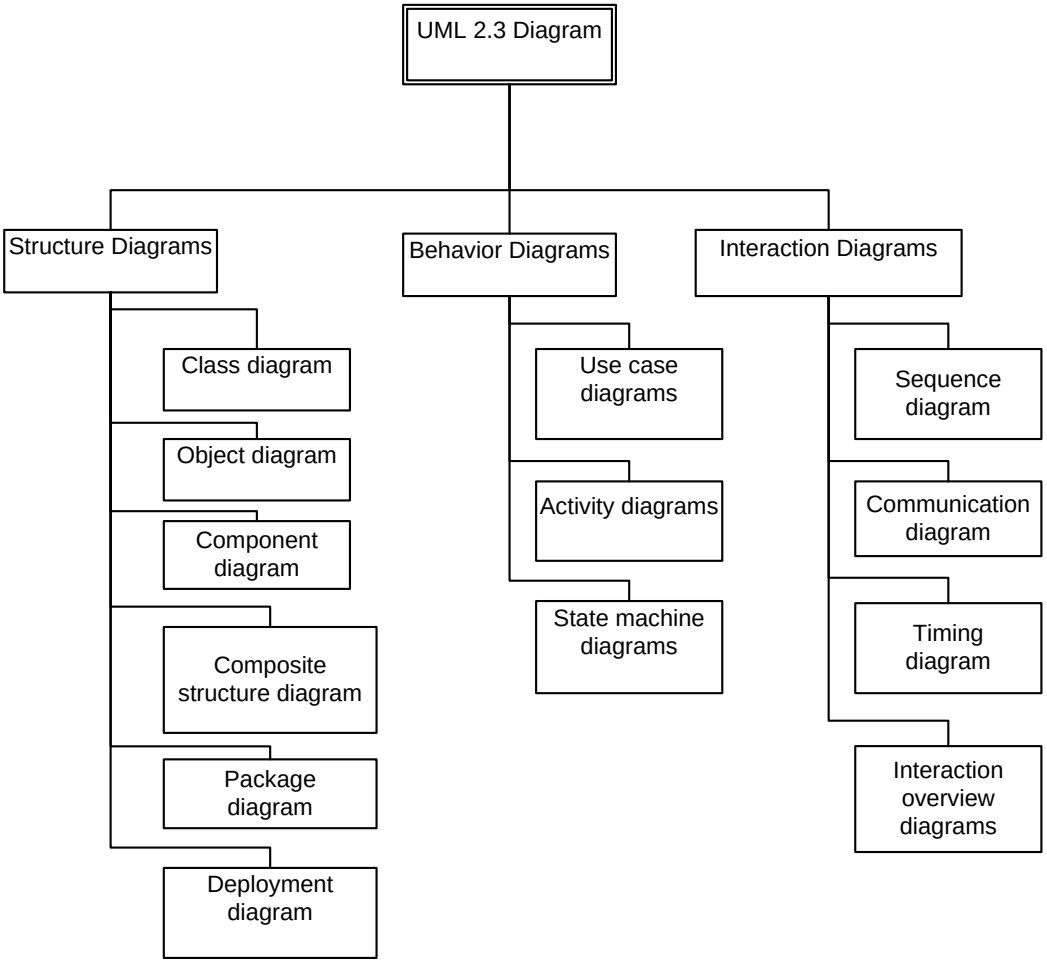
Query di atas digunakan untuk menghapus data mahasiswa dengan NIM = 10114900 dari tabel "Tabel_mahasiswa".

2.14 *Unified Modeling Language (UML)*

Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakana di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung [18].

2.14.1 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:



Gambar 2.10 Diagram UML [18]

Kategori diagram UML:

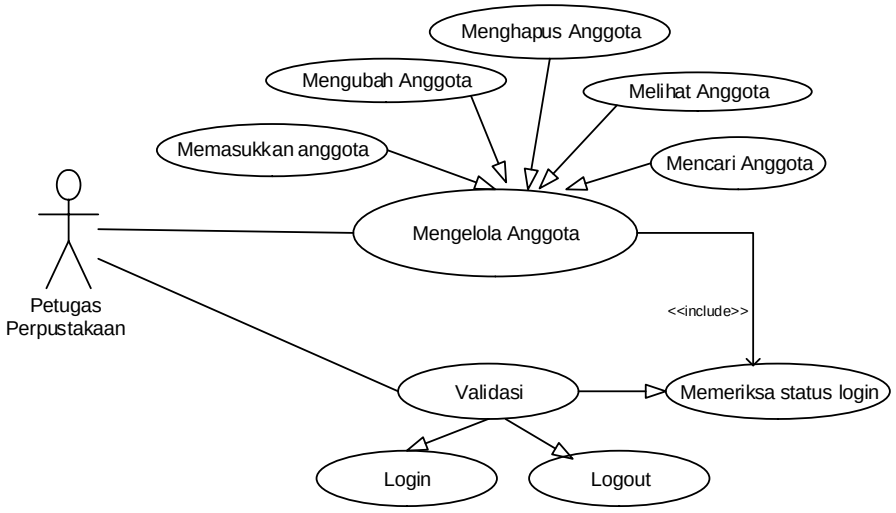
1. *Structure Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari system yang dimodelkan
2. *Behavior Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan system atau rangkaian perubahan yang terjadi pada sebuah system.
3. *Interaction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

Berikut diagram yang akan digunakan dalam penelitian ini:

1) *Use case Diagram*

Use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem 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 aktor dan *use case* [18].

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



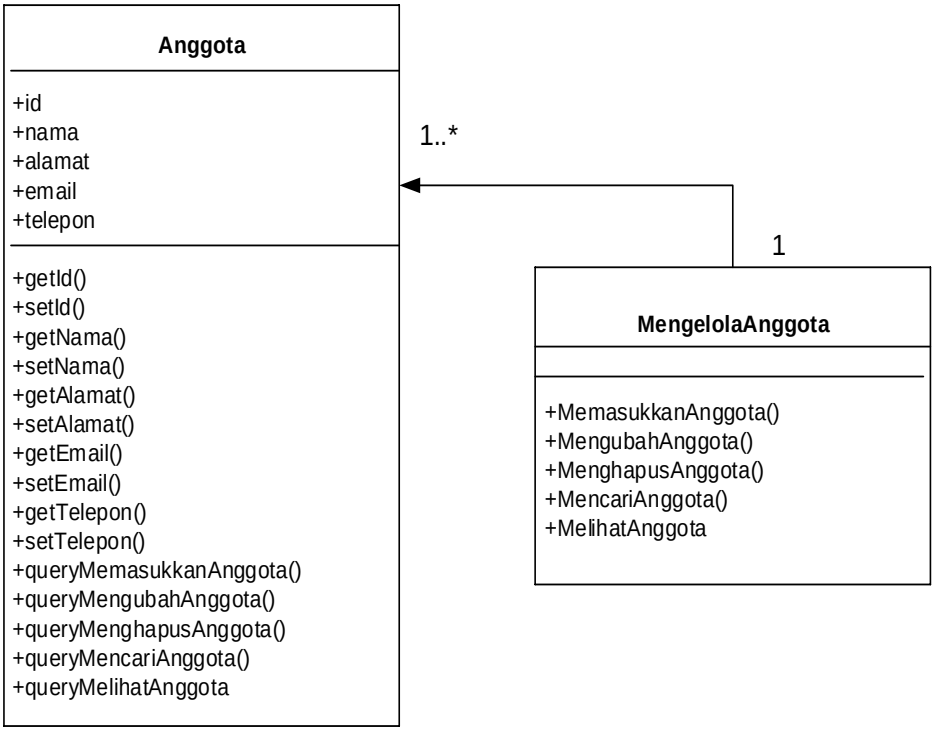
Gambar 2.11 Contoh *Use case* [18]

Pada *use case* di atas, Petugas Perpustakaan merupakan aktor dan *use case* Mengelola Anggota memiliki hubungan generalisasi dan spesialisasi (umum – khusus) dengan beberapa *use case* lain. Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum). Sementara itu *Use case* mengelola anggota juga berelasi dengan *use case* memeriksa status *login* dan memiliki keterangan *include*

yang berarti *use case* memeriksa status *login* akan selalu dipanggil saat *use case* mengelola anggota dijalankan.

2) *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 [18].



Gambar 2.12 Contoh *Class* Diagram [18]

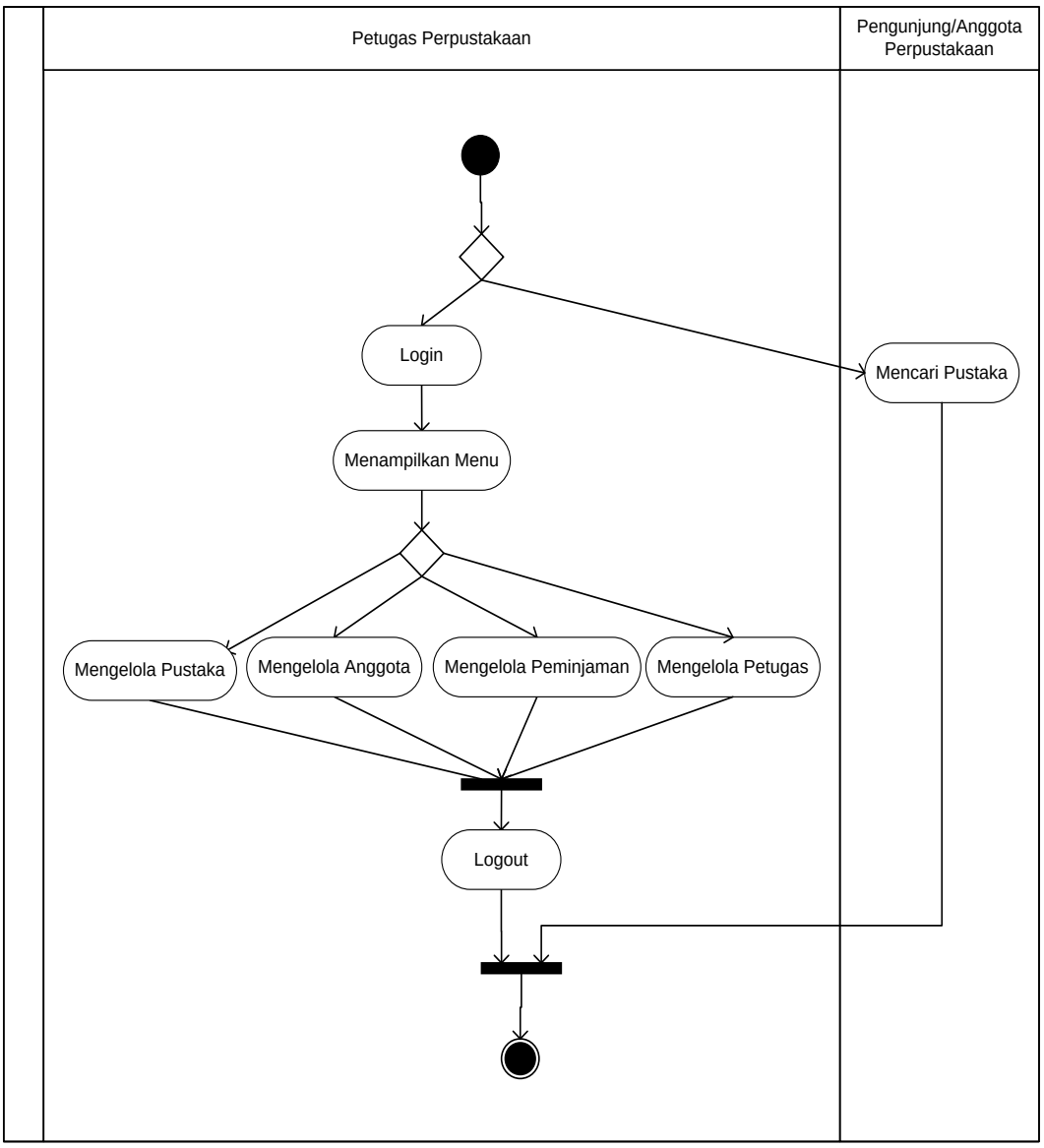
Pada kelas anggota memiliki atribut id, nama, alamat, email dan telepon sedangkan untuk operasinya seperti getID, setID, dan seterusnya. Kelas anggota berasosiasi dengan kelas MengelolaAnggota. Arah panah relasi pada diagram kelas mengarah pada diagram kelas yang lebih besar kontrolnya atau yang dipakai. Dan memiliki kardinalitas yang berarti satu proses MengelolaAnggota dapat dilakukan pada satu atau lebih anggota. Pada kelas MengelolaAnggota merupakan kelas proses yang diambil dari pendefinisian *use case* mengelola anggota yang di dalamnya harus juga menangani proses memasukkan anggota, mengubah anggota, menghapus anggota, mencari anggota dan melihat anggota.

3) Activity diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

Activity diagram juga banyak digunakan untuk mendefinisikan hal-hal berikut [18]:

a) Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis system yang didefinisikan
b) Urutan atau pengelompokan tampilan dari sistem/*user interface* di mana ssetiap aktivitas dianggap memiliki sebuah rancangan antarmuka pilihan
c) Rancangan pengujian di mana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
d) Rancangan menu yang ditampilkan pada perangkat lunak

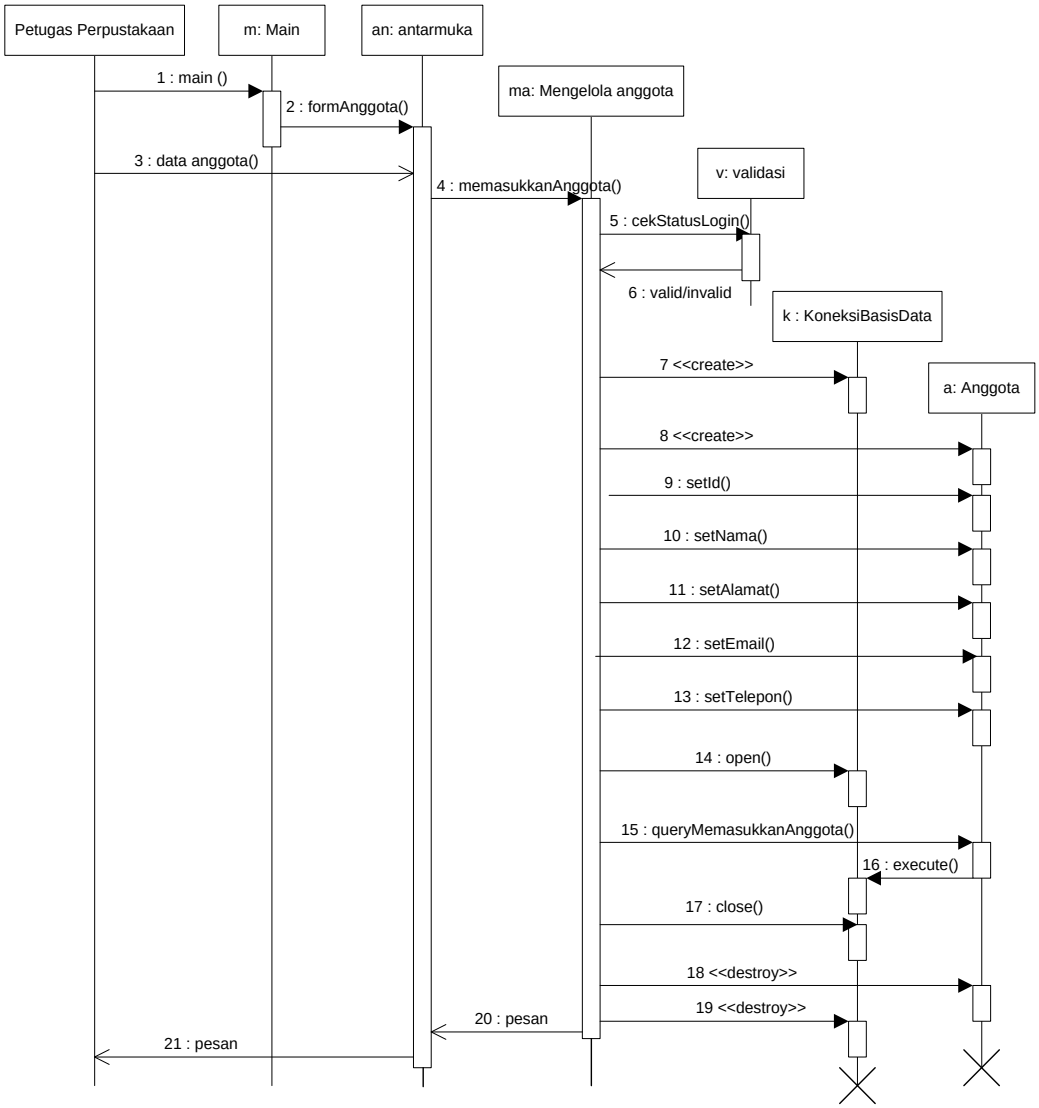


Gambar 2.13 Contoh *Activity diagram* [18]

Pada *activity diagram* di atas, aktivitas diawali dari kondisi percabangan di mana ada dua pilihan yaitu aktivitas Login dan aktivitas Mencari Pustaka. Login hanya bisa dilakukan oleh Petugas Perpustakaan untuk memilih aktivitas-aktivitas yang ada di dalamnya. Pada akhir aktivitas terjadi asosiasi penggabungan di mana lebih dari satu aktivitas digabungkan menjadi satu. Keseluruhan aktivitas sistem ini berada di dalam sebuah *swimlane* vertikal yang memisahkan organisasi bisnis Petugas Perpustakaan dengan Pengunjung/Anggota Perpustakaan.

4) *Sequence diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuck menggambar *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki. Membuat *sequence diagram* juga dibutuhkan untuk melihat skenario yang ada pada *use case* [18].



Gambar 2.14 Contoh *Sequence diagram* [18]

Pada *sequence diagram* di atas memiliki aktor yaitu Petugas Perpustakaan. Aktor bisa terdiri dari orang, proses atau sistem lain yang berinteraksi dengan sistem yang akan dibuat. Sedangkan objek yang berinteraksi dengan pesan ditandai dengan "nama objek : nama kelas" yaitu m: Main; an: Antarmuka; dan lain-lain. Nomor-nomor yang mengawali pesan merupakan urutan interaksi pesan. Objek m: Main aktif setelah Petugas Perpustakaan memanggil main() yang ditandai dengan adanya waktu aktif yang menyatakan objek dalam keadaan aktif dan berinteraksi. Sehingga formAnggota() dilakukan di dalam metode main(). Pada akhir objek ma: MengelolaAnggota terdapat pesan tipe *destroy* yang arah panahnya mengarah ke objek k: KoneksiBasisData dan a: Anggota. Artinya, objek ma: MengelolaAnggota mengakhiri hidup objek lain setelah sebelumnya membuat objek ditandai dengan pesan tipe *create*. Di akhir sesi an: Antarmuka mengirim data/masukan/informasi ke Petugas Perpustakaan.