BAB II
LANDASAN TEORI

2.1 Tiket

Tiket merupakan suatu alat yang diperuntukan untuk menentukan hak akses dari seseorang untuk dapat masuk ke dalam suatu event, menggunakan suatu fasilitas khusus ataupun mendapatkan hal lainnya yang tidak dapat dimiliki jika tidak memiliki tiket. Saat ini tiket di pergunakan oleh pengelola transportasi sebagai tanda bahwa penumpang telah melakukan pembayaran, kemudian digunakan oleh para *event organizer* sebagai tanda bahwa pengunjung *event* telah melakukan pembayaran untuk mendapatkan akses pada *event* tersebut.



Gambar 2.1 Contoh Tiket Konser

2.2 Aplikasi

Aplikasi adalah penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi output [6]. Menurut kamus besar Bahasa Indonesia (2005 : 52), "Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu". Dari kedua definisi tersebut, dapat disimpulkan bahwa aplikasi adalah suatu program pada komputer yang dibuat untuk melakukan suatu kegiatan atau tugas dari penggunanya dengan memproses input dari pengguna dan menghasilkan output.

2.3 Basis Data (*Database)*

Database atau basis data dapat diartikan sebagai kumpulan data *store* yang tersimpan dalam disk magnetik, disk optikal atau media penyimpanan sekunder lainnya, yang terdiri dari satu atau lebih tabel yang terintegrasi satu sama lain, di mana setiap pemakai (*user*) diberi wewenang (otorisasi) untuk dapat mengakses (mengubah, menghapus, menganalisis, menambah, memperbaiki) data dalam tabel-tabel tersebut [2]. Dengan kata lain, basis data merupakan sekumpulan data yang saling terkait (berelasi) antara satu dengan yang lainnya yang menggambarkan suatu organisasi (*enterprise*). Sedangkan sistem basis data adalah suatu sistem yang mengelola data dan dapat menyediakan data tersebut ketika diperlukan.

2.3.1 Keuntungan Basis Data

Penggunaan basis data dapat memberikan beberapa keuntungan, diantaranya:

a. Mereduksi redundansi yang akibatnya mengurangi inkonsistensi.
   Penyimpanan data yang sama berulang-ulang di beberapa *file* dapat mengakibatkan inkonsistensi. Apabila salah satu dari *file* yang mengandung data tersebut terlewat diperbaharui maka terjadilah data yang tidak konsisten lagi.
b. Data dapat terintegrasi antar aplikasi.
   Data dapat dipakai secara bersama-sama oleh beberapa program aplikasi (secara *batch* maupun on-*line*).
c. Standarisasi data dapat dilakukan.
   Karena adanya pengontrolan yang terpusat, maka DBA dapat menerapkan standarisasi data yang disimpan sehingga memudahkan pemakaian, pengiriman maupun pertukaran data. Jika data tersebar dalam beberapa file dalam bentuk format yang tidak sama, maka ini menyulitkan dalam program aplikasi untuk mengambil dan menyimpan data. Maka haruslah data dalam suatu basis data dibuat format yang standar sehingga mudah dibuat program aplikasinya.

d. Batasan keamanan dapat diterapkan.

   DBA dapat memberikan batasan-batasan pengaksesan data, misalnya dengan memberikan *password* dan pemberian hak akses bagi pengguna.

e. Mengelola integritas (Keterjaminan Akurasi) data.

   Jika kerangkapan data dikontrol dan kekonsistenan data dapat dijaga maka data menjadi akurat.

f. Menyeimbangkan kebutuhan yang saling konflik.

g. Independensi data (objektif DBS).

   Kekebalan aplikasi terhadap perubahan struktur penyimpanan dan teknik pengaksessan data. Basis data harus dapat berkembang tanpa mempengaruhi aplikasi yang telah ada. Apapun perubahan dalam basis data semua perintah akan mengalami kestabilan tanpa perlu ada yang diubah.

2.3.2 Komponen Utama Sistem Basis Data

Komponen utama sistem basis data adalah sebagai berikut.

a. Data yang disimpan dalam basis data.

b. Perangkat keras.

   Berupa komputer dan bagian-bagian di dalamnya seperti processor, memory dan hardisk. Komponen inilah yang melakukan pemrosesan dan juga untuk menyimpan basis data

c. Perangkat lunak.

   Perangkat lunak ini terdiri dari program pengelola basis data. Perangkat lunak inilah yang akan menentukan bagaimana data diorganisasi, disimpan, diubah dan diambil kembali. Perangkat lunak ini juga menerapkan mekanisme pengamanan data, pemakaian data secara bersamaan, pemaksaan keakuratan/konsistensi data, dsb.

d. DBMS, Report-writer, design arts, dll.

e. Basis data.

   Sebuah DBMS dapat memiliki beberapa basis data, setiap basis data dapat berisi sejumlah objek basis data(file, table, indeks, dll). Disamping berisi data, setiap basis data juga menyimpan definisi struktur(baik untuk basis data maupun objek-objeknya secara detail).

f. Pengguna.

1) Pengguna Awam (Naïve User)
2) Pengguna Biasa (Casual User)
3) Programmer.

   Orang yang membuat program aplikasi yang menggunakan basis data.

4) Administrator.

   Orang yang bertanggung jawab terhadap pengelolaan basis data.

2.3.3 Bahasa dalam Sistem Basis Data

Secara umum terdapat dua jenis bahasa dalam sistem basis data, yaitu:

a. *Data Definition Language* (DDL)

Perintah-perintah yang digunakan oleh database administrator untuk mendefinisikan sekema ke DBMS. Secara detil hal yang perlu dijabarkan pada DBMS adalah:

1) Nama basis data.
2) Nama seluruh berkas pada basis data.
3) Nama rekaman dan medan.
4) Deskripsi berkas, rekaman dan medan.
5) Nama medan kunci.
6) Nama indeks dan medan yang menjadi indeks.
7) Hal lain seperti ukuran basis data.

DDL juga digunakan untuk menciptakan, mengubah, dan menghapus basis data.

b. *Data Manipulation Language* (DML)

DML adalah perintah-perintah yang digunakan untuk mengubah, memanipulasi dan mengambil data pada basis data. Tindakan seperti menghapus, mengubah, dan mengambil data menjadi bagian dari DML. DML dibagi atas 2 jenis, yaitu:

a. Prosedural

   Menuntut pengguna menetukan data apa saja yang diperlukan dan bagaimana cara mendapatkannya.

b. Nonprosedural

Yang menuntut pengguna menentukan data apa yang diperlukan tetapi tidak perlu menyebutkan cara mendapatkannya.

2.3.4 Operasi Basis Data

Operasi dasar basis data diantaranya adalah pembuatan, penghapusan, pembuatan tabel, pengisian tabel yang dijelaskan sebagai berikut.

a. Pembuatan *database* baru (*create database*) yaitu menciptakan *database* baru.
b. Penghapusan *database* (*drop database*) yaitu penghapusan suatu *database*.
c. Pembuatan file/tabel baru ke dalam suatu databse yaitu penambahan file/tabel ke dalam database.
d. Penambahan/pengisian data baru ke sebuah file/tabel di sebuah *database* (*insert*).
e. Perubahan data yang ada pada sebuah file/tabel di sebuah *database* (*update*).
f. Penghapusan data yang ada pada sebuah file/tabel di sebuah *database* (*delete*).

2.4 Jaringan Komputer

Jaringan komputer adalah kumpulan antara teknologi komputer, dan teknologi komunikasi. Gabungan teknlogi ini melahirkan pengolahan data yang dapat didistribusikan, mencakup pemakaian *database*, *software* dan *hardware*. Setiap komputer, printer atau periferal yang terhubung dengan jaringan disebut *node*. Sebuah jaringan komputer dapat memiliki dua, puluhan, ribuan atau bahkan jutaan node.

Jaringan komputer dapat diklasifikasikan menurut skala sebagai berikut :

a. Local Area Network (LAN):

Suatu jaringan komputer yang menghubungkan suatu komputer dengan komputer lain dengan jarak yang terbatas.

b. Metropolitant Area Network (MAN):

Prinsip sama dengan LAN, hanya saja jaraknya lebih luas, yaitu 10-50 km.

c. Wide Area Network (WAN):

Jaraknya antar kota, negara, dan benua. ini sama dengan internet.

Pada dasarnya setiap jaringan komputer ada yang berfungsi sebagai *client* dan juga *server*. Tetapi ada jaringan yang memiliki komputer yang khusus didedikasikan sebagai *server* sedangkan yang lain sebagai *client*. Ada juga yang tidak memiliki komputer yang khusus berfungsi sebagai *server* saja. Karena itu berdasarkan fungsinya maka ada dua jenis jaringan komputer :

a. Client-server

Yaitu jaringan komputer dengan komputer yang didedikasikan khusus sebagai *server*. Sebuah *service*/layanan bisa diberikan oleh sebuah komputer atau lebih. Contohnya adalah sebuah domain seperti www.unikom.ac.id yang dilayani oleh banyak komputer *web server*. Atau bisa juga banyak *service*/layanan yang diberikan oleh satu komputer. Contohnya adalah server if.unikom.ac.id yang merupakan satu komputer dengan multi *service* yaitu *mail server, web server*, file *server, database server* dan lainnya.

b. Peer-to-peer

Yaitu jaringan komputer dimana setiap *host* dapat menjadi *server* dan juga menjadi *client* secara bersamaan, atau dengan kata lain dapat dikatakan dima satu komputer terhubung ke komputer lainnya secara langsung.

Berdasarkan media transmisi datanya, jaringan komputer dapat dibedakan atas beberapa kategori, diantaranya :

a. Jaringan Berkabel (*Wired Network*)

Pada jaringan ini, untuk menghubungkan satu komputer dengan komputer lain diperlukan penghubung berupa kabel jaringan. Kabel jaringan berfungsi dalam mengirim informasi dalam bentuk sinyal listrik antar komputer jaringan.

b. Jaringan Nirkabel (*Wireless Network*)

Merupakan jaringan dengan medium berupa gelombang elektromagnetik. Pada jaringan ini tidak diperlukan kabel untuk menghubungkan antar komputer karena menggunakan gelombang elektromagnetik yang akan mengirimkan sinyal informasi antar komputer jaringan.

2.5 Internet

Internet dengan berbagai aplikasinya seperti *web*, VoIP, E-Mail pada dasarnya merupakan media yang digunakan untuk mengefisienkan proses komunikasi [7]. Internet merupakan jaringan komputer yang saling terhubung secara mendunia yang memungkinkan jutaan jaringan terhubung antara satu dengan yang lainnya sehingga *user* lebih mudah terhubung dengan *user* lainnya [4].

Berdasarkan kedua pendapat diatas, dapat ditarik kesimpulan bahwa internet merupakan jaringan komputer yang saling terhubung yang berguna untuk mengefisienkan proses komunikasi.

Asal usul internet dari jaringan komputer yang disusun oleh ARPA (*Advance Research Project Agency*) yang bernaung dibawah departemen pertahanan Amerika Serikat atau DoD (*Departement of Defense*) sehingga terbentuk ARPAnet, jaringan komputer bentukan departemen pertahanan Amerika Serikat, tahun 70-an. Pada awalnya, ARPAnet hanya menghubungkan empat buah situs saja, yaitu:

a. SRI (Stanford Research Institute)
b. USCB (University of California at Santa Barbara)
c. UCLA (University of California at Los Angeles)
d. University of Utah at Utah

Jaringan tersebut selanjutnya diperbaharui dan dikembangkan, dan saat ini digunakan sebagai tulang punggung jaringan informasi yang disebut dengan internet. Penelitian di Stanford menghasilkan sebuah protokol yang disebut TCP/IP, inilah yang berkembang terus hingga sekarang dan menjadi protokol standar dalam internet.

2.6 Website

Website merupakan kumpulan halaman *web* yang saling terhubung dan file-filenya saling terkait [1]. Web terdiri dari *page* atau halaman, dan kumpulan halaman yang dinamakan *homepage*. Homepage berada pada posisi teratas, dengan halaman-halaman terkait berada di bawahnya. Biasanya setiap halaman di

bawah *homepage* disebut *child page*, yang berisi *hyperlink* ke halaman lain dalam *web*.

Website awalnya merupakan suatu layanan sajian informasi yang menggunakan konsep *hyperlink*, yang memudahkan *surfer* atau pengguna *internet* melakukan penelusuran informasi di *internet*. Informasi yang disajikan dengan web menggunakan konsep multimedia, informasi dapat disajikan dengan menggunakan banyak media, seperti teks, gambar, animasi, suara, video atau kombinasi dari semuanya.

Selain menyediakan informasi, *website* juga menyediakan mekanisme bagi pengunjungnya untuk dapat berinteraksi dengan dengan situs tersebut. Untuk dapat berinteraksi dengan *website* tersebut, perlu ditambahkan kode-kode pemograman seperti PHP, ASP, JSP, Pearl dan sebagainya. *Website* yang telah ditambahkan kode program tersebut biasa dikenal dengan nama aplikasi web (*web application)*.

2.7 QR Code

Kode Batang atau yang biasa disebut *barcode* merupakan representasi data yang dapat dibaca oleh mesin optik yang berhubungan dengan objek yang terpasang. Pada awalnya, *barcode* mewakili data dengan memvariasikan lebar dan jarak dari garis pararel secara sistematis, dan dapat disebut sebagai linear atau satu dimensi (1D). Kemudian *barcode* berevolusi menjadi persegi panjang, titik, segi enam dan pola geometris lainnya dalam dua dimensi (2D). Meskipun sistem 2D menggunakan berbagai simbol, tapi pada umumnya tetap disebut sebagai *barcode* juga atau dikenal juga dengan sebutan kode matriks. Kelebihan dari kode matriks ini adalah dapat merepresentasikan lebih banyak data untuk setiap area unitnya. Awalnya *barcode* dipindai oleh pemindai (*scanner*) optic khusus yang disebut pembaca *barcode* (*barcode scanner*). Kemudian, *scanner* dan *software* interpretative mulai tersedia pada perangkat termasuk printer *desktop* dan *smartphones*.

Quick Respons Code atau biasa disebut QR *code* merupakan salah satu jenis kode matriks yang pertama kali dirancang oleh industry otomotif di Jepang. Sebuah QR *code* menggunakan empat mode standar *encoding* (yaitu numerik,

alfanumerik, *byte* / biner dan kanji) untuk menyimpan data secara efisien. Sistem QR *code* pun telah manjadi populer di luar industri otomotif karena pembacaan yang cepat dan kapasitas penyimpanan yang lebih besar dibandingkan dengan standar *barcode* UPC.



Gambar 2.2 QR Code untuk URL Halaman Pengguna Twitter

QR *code* terdiri dari modul hitam (titik persegi) diatur dalam kotak persegi pada latar belakang putih yang dapat dibaca oleh perangkat pencitraan seperti kamera dan diolah menggunakan koreksi kesalahan Reed-Solomon hingga gambar dapat dengan tepat di interpretasikan. Data tersebut kemudian diekstraksi dari pola yang ada dari kedua komponen horizontal dan komponen vertical pada gambar.

2.7.1 Standar

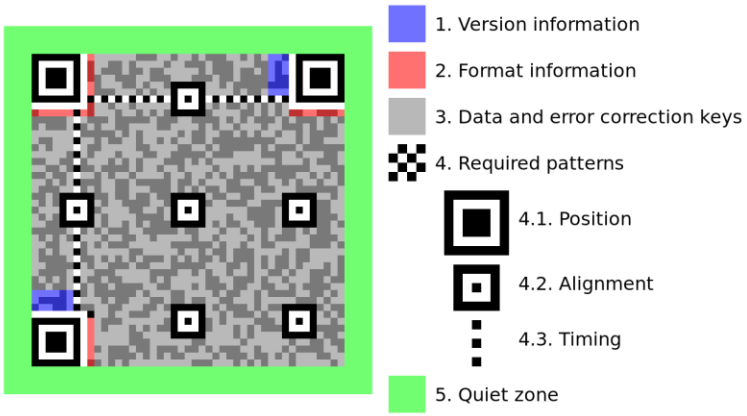
Ada beberapa standar yang mencakup pengkodean data sebagai QR *code*:

a. October 1997 – AIM (*Association for Automatic Identification and Mobility*) *International*
b. January 1999 – JIS X 0510
c. June 2000 – ISO/IEC 18004:2000 *Teknologi informasi – Indentifikasi otomatis dan teknik data capture – Simbologi barcode – QR code*
d. 1 September 2006 – ISO/IEC 18004:2006 *Teknologi informasi – Indetifikasi otomatis dan teknik data capture – QR code 2005 spesifikasi simbologi barcode.* Mendefinisikan simbol QR *code* 2005, perpanjangan QR *code* model 2. Tidak menentukan cara membaca simbol QR *code* model 1.

Pada lapisan aplikasi, terdapat beberapa variasi antara sebagian besar implementasi. NTT Docomo Jepangtelah membentuk standar de facto untuk pengkodean URL, informasi kontak dan beberapa jenis data lainnya. Proyek *open-source* "ZXing" menyimpan daftar tipe data QR *code*.

2.7.2 Desain

Tidak seperti *barcode* satu dimensi yang dirancang secara mekanis untuk dipindai oleh sinar sempit cahaya (*narrow beam of light*), QR *code* dideteksi oleh sensor gambar digital dua dimensi dan kemudian dianalisis secara digital oleh prosesor yang terprogram. Prosesor menempatkan togak kotak khas di sudut-sudut gambar QR *code* menggunakan persegi yang lebih kecil (atau beberapa kotak) dekat sudut keempat untuk menormalkan gambar untuk ukuran, orientasi dan sudut pandang. Titik-titik kecil di seluruh QR *code* tersebut kemudian dikonversi ke bilangan biner dan divalidasi dengan kode koreksi kesalahan.



Gambar 2.3 Struktur QR Code dan Elemen Fungsionalnya

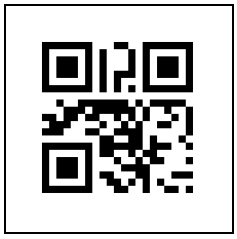
2.7.3 Storage

Jumlah data yang dapat disimpan pada simbol QR *code* tergantung dari tipe data (*mode*, atau kumpulan karakter masukan), versi (1, …, 40, mengindikasikan keseluruhan dimensi simbol), dan tingkat kode koreksi kesalahan. Kapasitas penyimpanan maksimum saat ini adalah untuk simbol 40-L (versi 40 dengan koreksi kesalahan tingkat *low*).

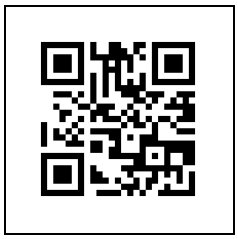
Tabel 2.1 Kapasitas Maksimum Penyimpanan Karakter (40-L)

Input Mode	Maks. Karakter	Bits/Karakter	Default Encoding
Numeric only	7.089	$3^1/_3$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Alphannumeric	4.296	$5^1/_2$	0–9, A–Z (upper-case only), space, $, %, *, +, -, ., /, :
Binary/byte	2.953	8	ISO 8859-1
Kanji/kana	1.817	13	Shift JIS X 0208

Setiap versi akan mempunyai ukuran yang berbeda dengan versi lainnya. Gambar 2.4 sampai Gambar 2.9 adalah contoh simbol QR *code* dengan versi yang berbeda.



Gambar 2.4 QR Code Versi 1 (21x21)



Gambar 2.5 QR Code Versi 2 (25x25)



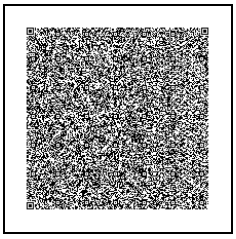
Gambar 2.6 QR Code Versi 3 (29x29)



Gambar 2.7 QR Code Versi 4 (33x33)



Gambar 2.8 QR Code Versi 25 (117x117)



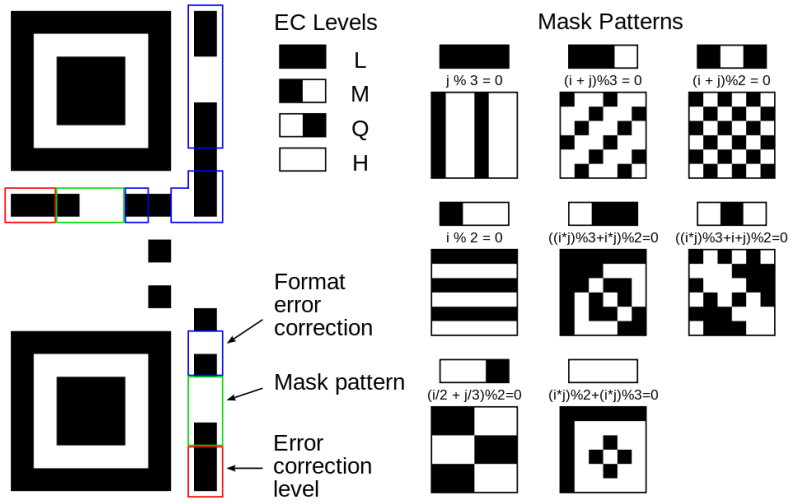
Gambar 2.9 QR Code Versi 40 (177x177)

2.7.4 Enkripsi

QR *code* yang dienkripsi sangatlah tidak umum, hanya sedikit yang mengiplementasikannya. Sebuah aplikasi Android misalnya, mengelola enkripsi dan dekripsi QR *code* menggunakan algoritma DES (56 bit). Sistem imigrasi Jepang menggunakan QR *code* yang dienkripsi ketika mengeluarkan visa paspor.

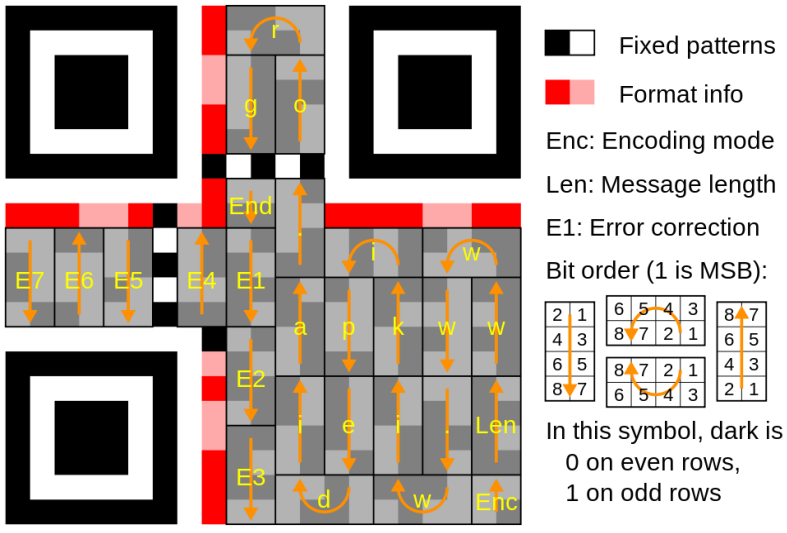
2.7.5 Encoding

Informasi format mencatat dua hal: tingkat koreksi kesalahan dan *mask pattern* yang digunakan untuk simbol. *Masking* digunakan untuk memecah pola di area data yang mungkin membingungkan scanner, seperti daerah kosong besar atau fitur menyesatkan yang terlihat seperti tanda *locator*. *Mask pattern* didefinisikan pada *grid* yang diulang sediperlukannya untuk menutupi seluruh simbol. Modul yang sesuai dengan daerah gelap *mask* terbalik. Informasi format dilindungi dari kesalahan dengan kode BCH, dan dua salinan lengkap termasuk dalam setiap simbol QR.

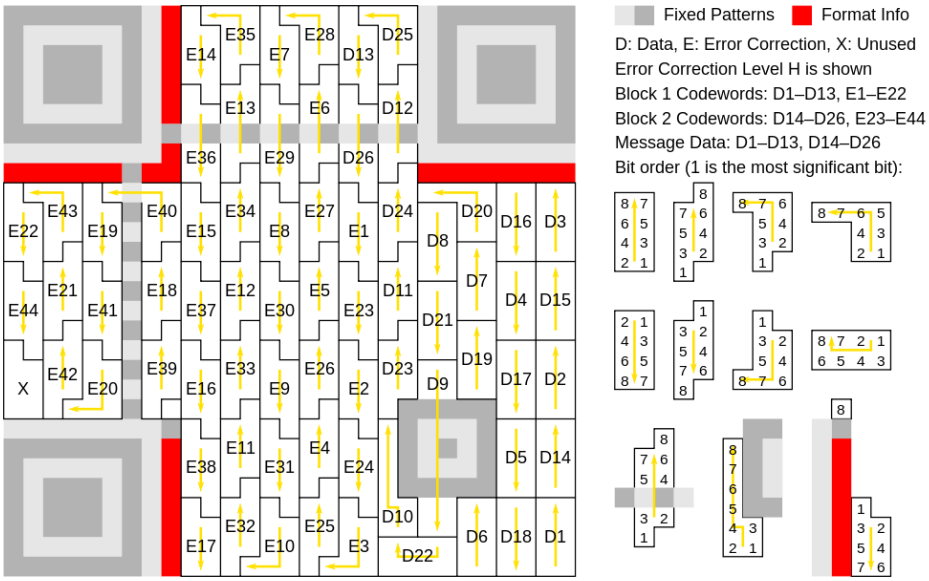


Gambar 2.10 Maksud Dari Informasi Format

Data pesan ditempatkan dari kanan ke kiri dalam pola zigzag. Dalam simbol yang lebih besar, cukup rumit karena adanya pola keselarasan dan penggunaan beberapa blok *interleaved* koreksi kesalahan.



Gambar 2.11 Penempatan Pesan dan Simbol QR



Gambar 2.12 Ilustrasi Blok Interleaved dengan Simbol yang Lebih Besar

Indikator 4 bit digunakan untuk memilih mode *encoding* dan menyampaikan informasi lainnya. Mode pengkodean dapat dicampur sesuai kebutuhan dalam simbol QR.

Tabel 2.2 Menu Encoding

Indikator	Keterangan
0001	Numeric encoding (10 bits per 3 digits)
0010	Alphanumeric encoding (11 bits per 2 characters)
0100	Byte encoding (8 bits per character)
1000	Kanji encoding (13 bits per character)
0011	Structured append (used to split a message across multiple QR symbols)
0111	Extended Channel Interpretation (select alternate character set or encoding)
0101	FNC1 in first position (see Code 128 for more information)
1001	FNC1 in second position
0000	End of message

Setelah setiap indikator yang memilih mode *encoding*, panjang *field* lah yang memberitahu berapa banyak karakter dikodekan dalam mode tersebut. Jumlah bit pada panjang *field* tergantung pada *encoding* dan versi simbol.

Tabel 2.3 Jumlah Bit Per Panjang Field

Encoding	Ver.1–9	10–26	27–40
Numerik	10	12	14
Alfanumerik	9	11	13
Byte	8	16	16
Kanji	8	10	12

Mode *encoding* alfanumerik menyimpan pesan lebih kompak daripada yang mode byte, tetetapi mode alfanumerik tidak bisa menyimpan huruf-huruf non-kapital dan memiliki pilihan tanda baca terbatas yang cukup untuk alamat *web* yang belum sempurna. Dua karakter dikodekan dalam nilai 11-bit dengan rumus:

$$V = 45 \times C_1 + C_2$$

Tabel 2.4 Kode Karakter Alfanumerik

Kode	Karakter	Kode	Karakter
00	0	**23**	N
01	1	**24**	O
02	2	**25**	P
03	3	**26**	Q
04	4	**27**	R
05	5	**28**	S
06	6	**29**	T
07	7	**30**	U
08	8	**31**	V
09	9	**32**	W
10	A	**33**	X
11	B	**34**	Y
12	C	**35**	Z
13	D	**36**	SP
14	E	**37**	$
15	F	**38**	%

16	G	**39**	*
17	H	**40**	+
18	I	**41**	–
19	J	**42**	.
20	K	**43**	/
21	L	**44**	:
22	M		

2.7.6 Koreksi Kesalahan

Codeword dengan panjang 8 bit dan menggunakan algoritma koreksi kesalahan Reed-Solomon dengan empat tingkat koreksi kesalahan. Semakin tinggi tingkat koreksi kesalahan, kapasitas penyimpanan akan semakin kurang. Tabel 2.5 mencantumkan perkiraan kemampuan koreksi kesalahan pada masing-masing empat tingkatan.

Tabel 2.5 Tabel Tingkat Koreksi Kesalahan

Tingkat EC	Kemampuan Koreksi Kesalahan	Indikator
Level L (Low)	7% dari codewords dapat dikembalikan	01
Level M (Medium)	15% dari codewords dapat dikembalikan	00
Level Q (Quartile)	25% dari codewords dapat dikembalikan	11
Level H (High)	30% dari codewords dapat dikembalikan	10

Dalam simbol QR *code* yang lebih besar, pesan akan dipecah menjadi beberapa blok kode Reed-Solomon. Ukuran dipilih sehingga paling banyak 15 kesalahan dapat diperbaiki di setiap blok, ini membatasi kompleksitas dari algoritma *decoding*. Blok kode tersebut kemudian disisipkan bersama-sama, sehingga lebih kecil kemungkinannya bahwa kerusakan lokal untuk simbol QR akan membanjiri kapasitas setiap blok tunggal.

Tingkat Koreksi kesalahan (*EC Level*) dan versi dari QR *code* ini dapat menentukan jumlah data yang dapat ditampung dalam setiap QR *code*. Rincian dari versi 1 sampai versi 5 dengan tingkat koreksi kesalahan L hinggga H dapat dilihat pada Tabel 2.6 untuk seluruh versi QR *code* dapat dilihat pada lampiran.

Tabel 2.6 Tabel Error Correction Code Words dan Informasi Blok Versi 1-5

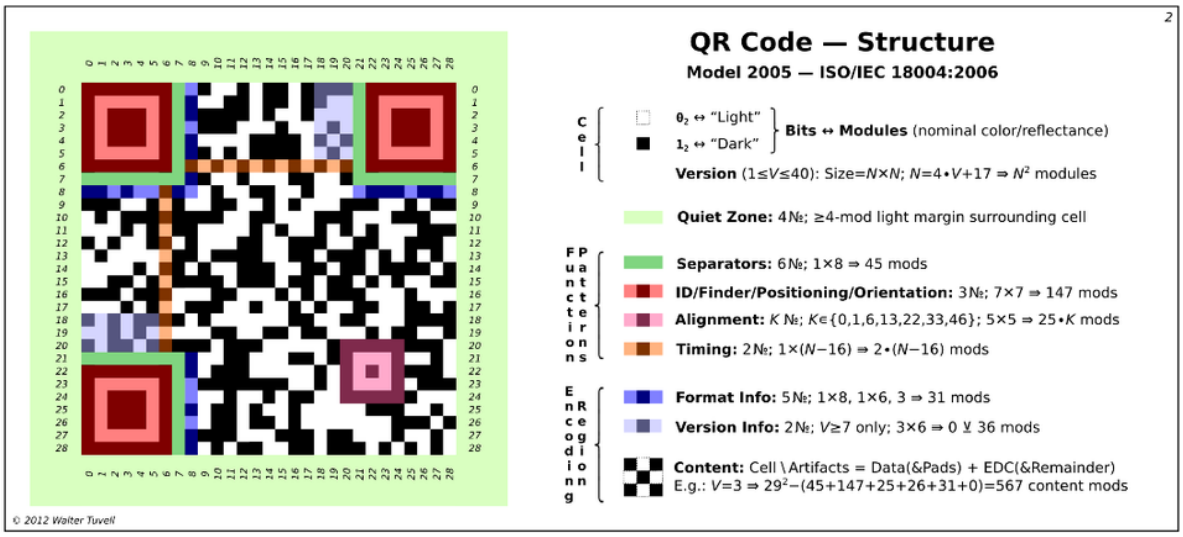
Versi dan EC Level	Total Banyak Data pada Codewords	EC Codewords Per Blok	Banyak Blok pada Group 1	Banyak Data Codewords pada Setiap Blok Group 1	Banyak Blok pada Group 2	Banyak Data Codewords pada Setiap Blok Group 2	Total Data Codewords
1-L	19	7	1	19			(19*1) = 19
1-M	16	10	1	16			(16*1) = 16
1-Q	13	13	1	13			(13*1) = 13
1-H	9	17	1	9			(9*1) = 9
2-L	34	10	1	34			(34*1) = 34
2-M	28	16	1	28			(28*1) = 28
2-Q	22	22	1	22			(22*1) = 22
2-H	16	28	1	16			(16*1) = 16
3-L	55	15	1	55			(55*1) = 55
3-M	44	26	1	44			(44*1) = 44
3-Q	34	18	2	17			(17*2) = 34
3-H	26	22	2	13			(13*2) = 26
4-L	80	20	1	80			(80*1) = 80
4-M	64	18	2	32			(32*2) = 64
4-Q	48	26	2	24			(24*2) = 48
4-H	36	16	4	9			(9*4) = 36
5-L	108	26	1	108			(108*1) = 108

5-M	86	24	2	43			(43*2) = 86
5-Q	62	18	2	15	2	16	(15*2) + (16*2) = 62
5-H	46	22	2	11	2	12	(11*2) + (12*2) = 46

Berkat koreksi kesalahan, besar kemungkinan untuk membuat QR *code* artistik yang masih dapat dipindai dengan benar, tapi mengandung kesalahan yang disengaja untuk membuat QR *code* lebih mudah dibaca atau menarik bagi mata manusia, serta untuk menggabungkan warna, logo, dan fitur lainnya ke dalam blok QR *code*.

2.7.7 Decoding

Decoding adalah proses pembacaaan QR *code* untuk menghasilkan informasi dari data yang ada pada QR *code*. *Decoding* merupakan kebalikan dari proses *encoding* yang merupakan proses untuk mengubah data ke dalam bentuk QR *code*. Ada beberapa bagian pada QR *code* yang dapat dilihat secara manual seperti pada Gambar 2.13.



Gambar 2.13 Lokasi Area QR Code

a. *Finder pattern*

Finder pattern untuk membantu mendeteksi area simbol QR *code* dalam aplikasi *decoder*. Simbol ini memiliki beberapa sudut menurun yang mampu rotasi 360 derajat melalui *finder pattern* di sudut-sudut.

b. *Alignment pattern*

Alignment pattern sebagai pembantu atau penyeimbang *finder pattern*. *Finder pattern* dan *Alignment pattern* adalah bagian paling penting untuk menemukan simbol. Satu-satunya perbedaan antara versi QR *code* adalah bahwa pada versi 1 tidak mengunakan *alignment pattern*, untuk versi 2 sampai versi 6 hanya satu *alignment pattern*, dan untuk versi 7 sampai versi 40 menggunakan lebih dari satu *alignment pattern*.

c. *Timing Pattern*

Pola ini digunakan untuk membantu menentukan koordinat modul simbol dalam aplikasi QR *code decoder*.

d. *Quiet Zone*

Bagian ini adalah ruang kosong ditempatkan di sekitar simbol QR *code* dan digunakan untuk meningkatkan pola *finder* untuk terdeteksi dengan cepat.

e. *Format Area*

Bagian ini menyimpan beberapa informasi tentang simbol QR *code*, seperti versi, tipe data, dan sebagainya.

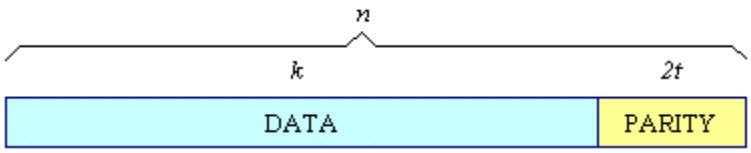
f. *Data Area*

Daerah ini adalah bagian paling penting dari QR *code* karena bagian ini lokasi data yang dikodekan.

2.8 Reed-Solomon *Codes*

Reed-Solomon *code* adalah kode siklik nonbiner yang terbuat dari 2n bit biner dimana m lebih besar daripada 2. Diciptakan oleh Irving S. Reed dan Gustave Solomon. Mereka menjelaskan secara sistematis kode bangunan yang dapat mendeteksi dan memperbaiki beberapa kesalahan simbol acak. Untuk sebuah *codeword*, panjang kode adalah 8 bit, maka Reed-Solomon *code*: $28 - 1 = 255$. Untuk mengkoreksi kesalahan pada *codeword*, maka ditambahkan Reed-

Solomon *code* agar dapat terlindung dari kerusakan tanpa harus kehilangan data. Kemampuan koreksi error-nya bergantung pada jumlah data yang dikodekan. Reed-Solomon *code* terdiri atas 2 bagian, yaitu bagian data dan bagian paritas. Reed-Solomon *code* dinyatakan dengan kode (n,k) atau RS(n,k) dimana n adalah maksimum *codeword*, yaitu 255, sedangkan k adalah jumlah dari *codeword* data. Berikut ini merupakan penjelasan dari gambar 2.15 mengenai struktur Reed-Solomon *code*.



Gambar 2.14 Struktur Reed-Solomon Code

2t atau simbol paritas adalah *codeword* yang digunakan untuk koreksi kesalahan dengan nilai maksimun adalah t. Sebagai contoh, RS (255,223) artinya terdapat total 255 *codeword* yang terdiri atas 223 *codeword* data dan 32 *codeword* paritas.

$$n = 255, k = 223$$

$$2t = (255\text{-}223) \rightarrow t = 16$$

sehingga kesalahan yang dapat diperbaiki adalah sebanyak 16 *codeword*. Sebagai informasi 1 *codeword* adalah 8 bit, sehingga total koreksi yang dapat diperbaiki adalah 16 x 8 bit = 128 bit.

Ketika panjang Reed-Solomon *code* kurang dari 28 – 1, maka padding 0 digunakan untuk membuat Reed-Solomon code tepat 28 – 1. Sewaktu proses pembacaan kode, padding 0 akan dibuang. Hal ini disebut dengan penyingkatan Reed-Solomon *code*. Sebagai contoh, misalkan terdapat 100 *codeword* data untuk mengkoreksi 8 buah kesalahan, sehingga akan dibutuhkan tambahan 16 *codeword* paritas. Jadi total keseluruhan dari *codeword* menjadi 116, yang mana masih kurang dari 28 – 1, sehingga harus ditambahkan 139 *codeword padding* 0.

Kode Reed-Solomon telah menemukan aplikasi penting dari dalam ruang komunikasi untuk elektronik konsumen. Mereka jelas digunakan dalam elektronik konsumen seperti CD, Blu-ray Disc, dalam teknologi transmisi data seperti DSL

dan WiMAX , dalam sistem penyiaran seperti DVB dan ATSC, dan dalam aplikasi komputer seperti RAID 6 sistem.

Hampir semua *barcode* dua dimensi seperti PDF-41, MaxiCode, datamatrix, QR Code, dan Aztec menggunakan kode koreksi kesalahan Reed-Solomon untuk memungkinkan pembacaan yang benar bahkan jika sebagian dari *barcode* rusak. Ketika *barcode* scanner tidak dapat mengenali simbol *barcode*, ia akan memperlakukannya sebagai penghapusan.

2.9 Aplikasi Pendukung (*Tools*)

Aplikasi pendukang merupakan aplikasi yang mendukung sistem dalam proses pembangunan maupun dalam silkus hidup sistem tersebut. Aplikasi pendukung ini sangat membantu terhadap fungsional dari sistem.

2.9.1 Android

Android adalah sebuah sistem operasi berbasis *Linux kernel* dan dirancang untuk perangkat *mobile* dengan layar sentuh seperti *smartphone* dan komputer tablet. Awal mulanya dikembangkan oleh Android, Inc., yang didukung oleh Google secara finansial dan kemudian dibeli pada tahun 2005. Android ini diresmikan pada tahun 2007 seiring dengan berdirinya *Open Handset Alliance* yang merupakan konsorsium dari *hardware*, *software* dan perusahaan telekomunikasi yang ditujukan untuk memanjukan standar terbuka untuk perangkat *mobile*.

2.9.1.1 Version History

Version history dari sistem operasi *mobile* Android dimulai dari Android *beta* yang dirilis pada Nopember 2007. Android 1.0 yang merupakan versi komersil pertama, dirilis pada September 2008. Android berada di bawah pengembangan Google dan OHA (*Open Handset Alliance*) dan memiliki banyak perbaharuan pada sistem operasinya sejak pertama kali dirilis. Tabel 2.7 merupakan sejarah versi dari Android berdasarkan API levelnya.

Tabel 2.7 Versi Android berdasarkan API level

API Level	Versi	Code-Name	Tanggal Rilis
API Level 1	1.0		28 September 2008
API Level 2	1.1		9 Februari 2009
API Level 3	1.5	Cupcake	30 April 2009
API Level 4	1.6	Donut	15 September 2009
API Level 5	2.0	Éclair	26 Oktober 2009
API Level 6	2.0.1		3 Desember 2009
API Level 7	2.1		12 Januari 2010
API Level 8	2.2	Froyo	20 May 2010
	2.2.1		18 Januari 2011
	2.2.2		22 Januari 2011
	2.2.3		21 November 2011
API Level 9	2.3	Gingerbread	6 Desember 2010
	2.3.1		Desember 2010
	2.3.2		Januari 2011
API Level 10	2.3.3		9 Februari 2011
	2.3.4		28 April 2011
	2.3.5		25 Juli 2011
	2.3.6		2September 2011
	2.3.7		21 September 2011
API Level 11	3.0	Honeycomb	22 Februari 2011
API Level 12	3.1		10 Mei 2011
API Level 13	3.2		15 Juli 2011
	3.2.1		20 September 2011
	3.2.2		30 Agustus 2011
	3.2.3		
	3.2.4		Desember 2011
	3.2.5		Januari 2012
	3.2.6		Februari 2012

API Level 14	4.0	Ice Cream Sandwich	19 Oktober 2011
	4.0.1		21 Oktober 2011
	4.0.2		28 November 2011
API Level 15	4.0.3		16 Desember 2011
	4.0.4		29 Maret 2012
API Level 16	4.1	Jelly Bean	9 Juli 2012
	4.1.1		23 July 2012
	4.1.2		9 Oktober 2012
API Level 17	4.2		13 November 2012
	4.2.1		27 November 2012
	4.2.2		11 February 2013
API Level 18	4.3		24 Juli 2013
	4.3.1		3 Oktober 2013
API Level 19	4.4	KitKat	31 Oktober 2013
	4.4.1		5 Desember 2013
	4.4.2		9 Desember 2013

2.9.1.2 Fitur

Android memiliki banyak sekali fitur. Fitur-fitur pada Android dapat diklasifikasikan sebagai berikut.

a. *General*

1) *Messaging*

SMS dan MMS tersedia dalam bentuk pesan, termasuk pesan teks *threaded* dan Google *Cloud Messaging* (GCM).

2) *Web browser*

Web browser yang tersedia pada Android adalah berbasis Blink *open-source layout engine*, ditambah dengan Chrome's V8 JavaScript Engine.

3) *Voice Based features*

Pencarian Google melalui suara telah tersedia sejak awal rilis. Perintah suara untuk perintah lainnya didukung pada Android 2.2 dan seterusnya.

4) *Multi-touch*

Android memiliki dukungan asli untuk *multi-touch* yang awalnya tersedia pada *handset* seperti HTC Hero.

5) *Multitasking*

Tersedianya *multitasking* bagi aplikasi dengan penanganan alokasi memori yang unik.

6) *Screen capture*

Android mendukung pengambilan *screenshot* dengan cara menekan tombol *power* dan *volume-down* secara bersamaan.

7) *Video calling*

Android tidak mendukung panggilan video asli, tetapi beberapa perangkat memilki versi kostumisasi dari sistem operasi yang mendukung, baik melalui jaringan UMTS ataupun melalui IP (VoIP). Panggilan video melalui applikasi lainnya telah tersidia saat ini seperti Skype, Hangouts (Google Talk), Vibe, dan aplikasi lainnya.

8) *Multi language support*

Android mendukung multi bahasa.

9) *Accessibility*

Built in text to speech disediakan melalui TalkBack bagi orang-orang dengan gangguan penglihatan. Penambahan fitur bagi orang-rang dengan gangguan pendengaran tersedia seperti alat bantu lainnya.

b. *Connectivity*

1) *Connectivity*

Android mendukung teknologi konektifitas termasuk GSM/EDGE, Wi-Fi, Bluetooth, LTE, CDMA, EV-DO, UMTS, NFC, IDEN dan WiMax.

2) Bluetooth

Mendukung panggilan suara dan mengirim kontak antar telepon, mengirim file (OOP), mengakses buku telepon (PBAP), A2DP dan AVRCP. Dukungan keyboard, mouse dan joystick (HID) tersedia pada Android versi 3.1 keatas dan pada versi sebelumnya melalui kostumisasi produsen dan aplikasi pihak ketiga.

3) *Tethering*

Android mendukung *tethering*, yang memungkinkan ponsel untuk berbagi konektifitas internet. Dapat dilakukan dengan media kabel ataupun nirkabel sebagai Wi-Fi hotspot. Sebelum Android 2.2, fitur ini didukung oleh aplikasi pihak ketiga atau kostumisasi pabrikan.

c. Media

1) *Streaming* media *support*

RTP / RTSP *streaming* (3GPP PSS, ISMA), progresif *download* HTML (HTML5 <video> tag). Adobe Flash *Streaming* (RTMP) dan *Streaming* HTTP Dinamis didukung oleh plugin Flash. Apple HTTP *Live Streaming* didukung oleh RealPlayer for Android, dan oleh sistem operasi Android 3.0 (Honeycomb) keatas.

2) Media *support*

Android mendukung format media audio/video sebagai berikut: WebM, H.263, H.264, AAC, HE-AAC (in 3GP or MP4 *container*), MPEG-4 SP, AMR, AMR-WB (in 3GP *container*), MP3, MIDI, Ogg Vorbis, FLAC, WAV, JPEG, PNG, GIF, BMP, WebP.

3) *External Storage*

Kebanyakan perangkat Android menyertakan slot microSD dan dapat membaca kartu microSD dengan format Fat32, Ext3 atau sistem file Ext4. Untuk memungkinkan penggunaan media penyimpanan berkapasistas tinggi seperti USB *flash drive* dan HDD USB, banyak tablet Android yang juga menyertakan USB 'A' *receptable* yang biasa disebut USB *On-The-Go* atau USB *host*..

d. *Hardware Support*

Perangkat Android dapat menyertakan kamera video, layar sentuh, GPS, akselerometer, giroskop, barometer, magnetometer, kontrol *game* khusus, sensor kedekatan dan tekanan, thermometer, 2D *bit blits* terakselerasi (dengan orientasi perangkat keras, *scaling*, konversi format piksel) dan grafis 3D terakselerasi.

e. Fitur Lainnya

1) Java *support*

Sementara sebagian besar aplikasi Android ditulis dengan Java, tidak ada Java *Virtual Machine* di *platform* dan kode *byte* Java tidak dieksekusi. Kelas Java dikompilasi kedalam *executable* Dalvik dan berjalan pada Dalvik, yaitu mesin virtual khusus untuk Android dan dioptimalkan untuk perangkat *mobile* bertenaga baterai dengan memori dan CPU yang terbatas. Dukungan J2ME dapat diberikan melalui aplikasi pihak ketiga.

2) *Handset layout*

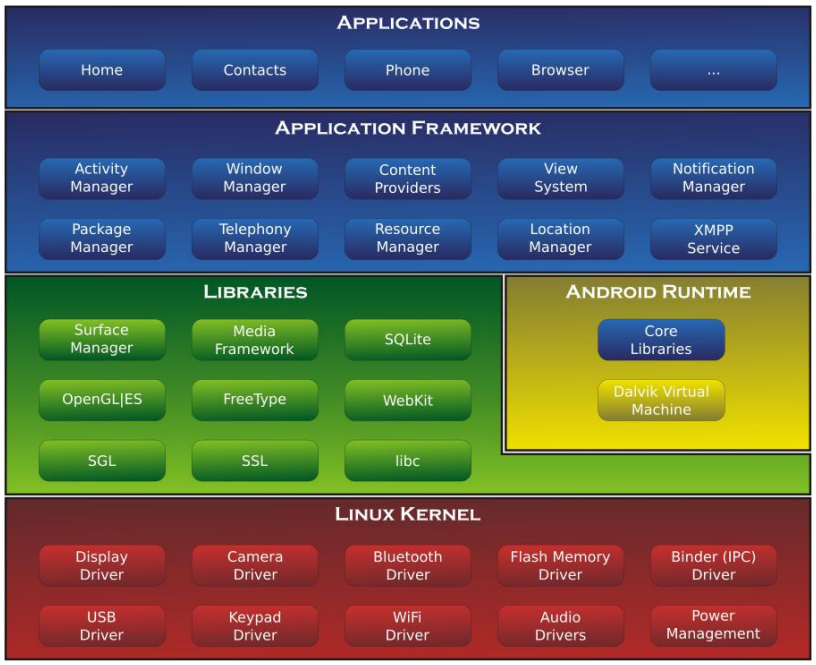
Platform ini bekerja untuk berbagai ukuran layar dari ukuran *smartphone* hingga ukuran tablet dan dapat terhubung ke layar eksternal, misalnya melalui HDMI atau secara nirkabel dengan *Miracast*. Orientasi potret dan lansekap juga didukang dan biasanya beralih antar dengan memutar. *Library* grafis 2D, *library* grafis 3D yang digunakan bebasis OpenGL ES 2.0.

3) *Storage*

SQLite, sebuah *database* relasional yang ringan. Digunakan untuk tujuan penyimpanan data.

2.9.1.3 Arsitektur

Google sebagai pencipta Android yang kemudian diasuh oleh *Open Handset Alliance* mengibaratkan Android sebagai sebuah tumpukan software. Setiap lapisan dari tumpukan ini menghimpun beberapa program yang mendukung fungsi-fungsi spesifik dari sistem operasi. Tumpukan paling bawah adalah kernel. Google menggunakan kernel Linux versi 2.6 untuk membangun Android, yang mencakup memory management, security setting, power management, dan beberapa driver hardware. Bertempat di level yang sama dengan library adalah lapisan runtime yang mencakup serangkaian inti library Java. Dengannya, para programmer dapat mengembangkan aplikasi untuk Android menggunakan bahasa pemrograman Java. Lapisan selanjutnya adalah application framework, yang mencakup program untuk mengatur fungsi- fungsi dasar smartphone.



Gambar 2.15 Arsitektur Sistem Operasi Android

a. Linux Kernel

Android dibangun di atas kernel Linux 2.6 hingga 3.4 Namun secara keseluruhan android bukanlah linux, karena dalam android tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal dalam manajemen memori dan proses. Oleh karenanya pada android hanya terdapat beberapa servis yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan driver. Kernel linux menyediakan driver layar, kamera, *keypad*, Wi-Fi, *Flash Memory*, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan lubang keamanan. Kernel juga berperan sebagai lapisan abstract antara *hardware* dengan tumpukan aplikasi yang berjalan diatasnya.

b. *Libraries*

Android menggunakan beberapa *library* yang terdapat pada C/C++ dengan standar Berkeley Software Distribution (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa *library* diantaranya:

1) *System C Library* merupakan implementasi dari standar sistem *library* C (libc) untuk perangkat berbasis Linux.

2) Media *Libraries* untuk memutar dan merekam berbagai macam format audio dan video.
3) *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi.
4) *Graphic Library* termasuk didalamnya SGL dan OpenGL, untuk tampilan 2D dan 3D.
5) FreeType untuk rendering font bitmap dan vector.
6) SQLite untuk mengatur relasi *database* yang digunakan pada aplikasi.
7) SSl dan WebKit untuk *browser* dan keamanan internet.

c. *Android Runtime*

Pada android tertanam paket *library* inti yang menyediakan sebagian besar fungsi android. Inilah yang membedakan Android dibandingkan dengan sistem operasi lain yang juga mengimplementasikan Linux. Android *Runtime* merupakan mesin virtual yang membuat aplikasi android menjadi lebih tangguh dengan paket *library* yang telah ada.

d. *Application Framework*

Framework aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi android. Selain itu, juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan *user interface* dan sumber daya aplikasi. Bagian terpenting dalam *framework* aplikasi android adalah sebagai berikut [Hello Android 2nd Edition]:

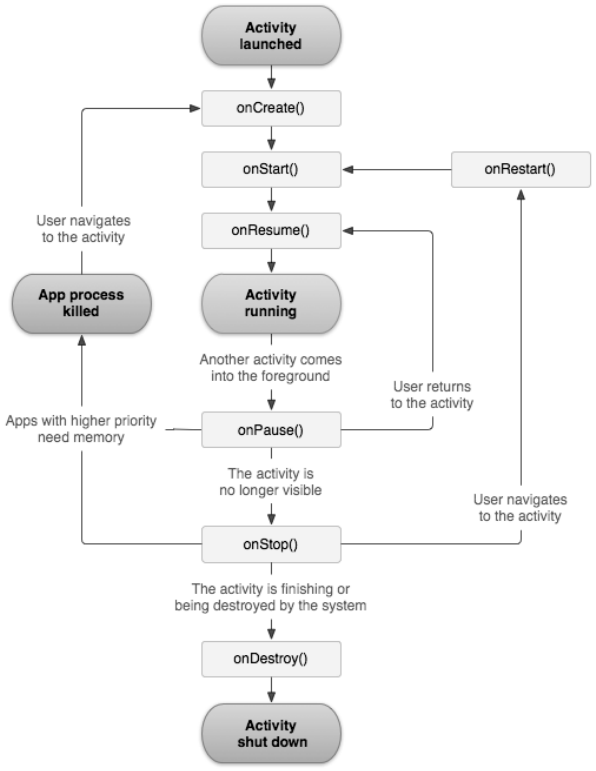
1) *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan ”Backstack“ untuk navigasi penggunaan.
2) *Content Providers*, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.
3) *Resuource Manager*, untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya diluar kode program, seperti karakter, grafik, dan file *layout*.
4) *Location Manager*, berfungsi untuk memberikan informasi detail mengenai lokasi perangkat android berada.

5) *Notification Manager*, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada status bar.

e. *Application Layer*

Puncak dari diagram arsitektur android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android *runtime* dengan menggunakan kelas dan service yang tersedia pada *framework* aplikasi.

Lapisan aplikasi android sangat berbeda dibandingkan dengan sistem operasi lainnya. Pada android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama. Android mempunyai siklus hidup aplikasi seperti pada Gambar 2.17.



Gambar 2.16 Siklus Hidup Aplikasi Android

2.9.1.4 Android SDK

Android SDK (Android *Software Development Kit*) merupakan perangkat lunak Android untuk proses pengembangan dimana proses aplikasi baru diciptakan untuk sistem operasi Android. Aplikasi tersebut biasanya dikembangkan dalam bahasa pemrograman Java dengan menggunakan Software Development Kit (SDK) Android, tetapi *tools* lainnya juga tersedia. Pada Juli 2013, lebih dari satu juta aplikasi telah dikembangkan untuk Android, dengan lebih 25 juta unduhan. Sebuah riset menunjukkan bahwa lebih 67% pengembang aplikasi seluler menggunakan platform Android. Pada Q2 2012, sekitar 105 juta unit telepon cerdas Android telah dipasarkan, dengan total pangsa pasar 68% secara keseluruhan.

Saat ini Android SDK termasuk seperangkat alat pengembangan yang terdiri dari debugger, *library*, sebuah handset emulator berdasarkan QEMU, dokumentasi, kode contoh, dan tutorial. Platform pengembangan saat ini, dapat berjalan pada modern Linux *distribution*, Microsoft Windows XP dan lebih baru serta Mac OS X 10.5.8 keatas.

IDE yang resmi didukung oleh Android adalah Eclipse menggunakan ADT (Android *Development Tools*) *Plugin* meskipun IntelliJ DEA mendukung penuh pengembangan Android dan NetBeans IDE pun mendukung pengembangan Android melalui sebuah *plugin*. Selain itu, pengembang dapat menggunakan editor teks apapun untuk mengedit file Java dan XML, kemudian menggunakan *command line tools* (diperlukan Java *Development* Kit dan Apache Ant) untuk menciptakan, membangun dan melakukan debug terhadap aplikasi Android serta pengendalian perangkat Android yang terpasang (misalnya, memicu reboot, dan menginstal paket perangkat lunak jarak jauh).

2.9.2 PHP QR Code

PHP QR *Code* adalah *open-source library* (LGPL) untuk menghasilkan QR *code*, barcode 2 dimensi. Berdasarkan *library* libqrencode C, menyediakan API untuk membuat gambar *barcode* QR *code* (PNG, JPEG berkat GD2). Diimplementasikan murni dalam PHP, tanpa dependensi eksternal (kecuali GD2 jika diperlukan).

2.9.3 ZXing (Zebra Crossing)

Merupakan sebuah *library* yang mendukung *decoding* dan mengasilkan *barcode* seperti QR *code*, PDF 417, EAN, UPC, Aztec, Data Matriks dan Codabar) dalam gambar. Proyek ini merupakan port dari pembaca dan generator *barcode* berbasis Java.

2.9.4 JSON

JSON (JavaScript Object Notation) adalah format data-*interchange* ringan. Sangat mudah bagi manusia untuk membaca dan menulisnya. Sangat mudah untuk mesin untuk mengurai dan menghasilkan. Hal ini didasarkan pada subset dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang merupakan bahasa independen tetapi menggunakan konvensi yang akrab bagi programmer dari keluarga bahasa C, termasuk C, C + +, C #, Java, JavaScript, Perl, Python, dan banyak lainnya. Properti ini membuat JSON menjadi bahasa data-*interchange* yang ideal.

JSON dibangun di atas dua struktur:

1) Koleksi nama / nilai pasangan. Dalam berbagai bahasa, ini direalisasikan sebagai objek, rekord, *struct*, *dictionary*, tabel *hash*, *keyed list*, atau array asosiatif.
2) Daftar nilai dipesan. Dalam kebanyakan bahasa, ini direalisasikan sebagai array, vektor, *list*, atau *sequence*.

Ini adalah struktur data universal. Hampir semua bahasa pemrograman modern mendukung JSON dalam satu bentuk atau bentuk lainnya. Masuk akal bahwa format data yang dipertukarkan dengan bahasa pemrograman juga didasarkan pada struktur ini.

2.9.5 MySQL

MySQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan

salah satu konsep utama dalam basisdata yang telah ada sebelumnya; SQL (*Structured Query Language*).

SQL adalah sebuah konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem basisdata (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basisdata transaksional maupun operasi basisdata non-transaksional.

Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basisdata kompetitor lainnya. Namun demikian pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi blogging berbasis web (*wordpress*), CMS, dan sejenisnya.

Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional.

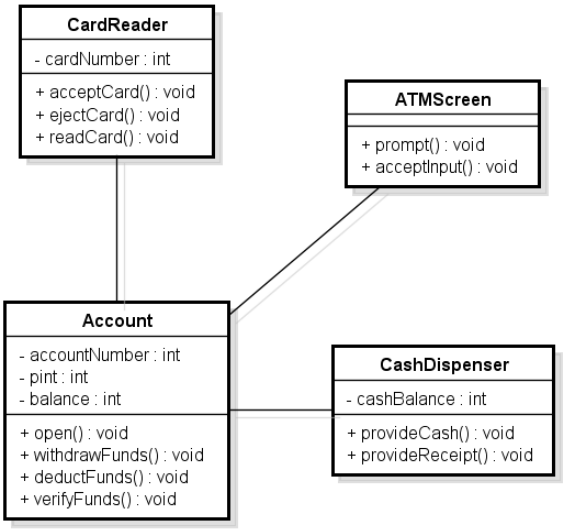
2.10 Object Oriented Programming (OOP)

Pemograman terstruktur adalah arus utama pada hari-hari sebelumnya dalam rekayasa perangkat lunak. Programmer mulai mengembangkan blok standar kode untuk melakukan operasi seperti pencetakan, dan kemudian menyalin dan menyisipkan kode tersebut ke setiap aplikasi yang mereka tulis [3]. Sementara hal ini mengurangi waktu pengembangan untuk aplikasi baru, cukup sulit jika perubahan yang diperlukan dalam blok kode, karena pengembang harus membuat perubahan di mana-mana. Dalam pemrograman terstruktur terdapat beberapa hal yang mana pemrograman berorientasi obyek dirancang untuk memecahkan hal tersebut.

Paradigma berorientasi objek adalah cara yang berbeda dalam memandang aplikasi. Pendekatan berorientasi objek, membagi aplikasi ke dalam banyak potongan kecil, atau benda, yang cukup independen satu sama lain. Kemudian aplikasi dibangun dengan menggabungkan semua benda atau potongan kecil ini bersama-sama. Beberapa prinsip orientasi objek diantaranya enkapsulasi, pewarisan, dan polimorfisme.

a. Enkapsulasi (*Encapsulation*)

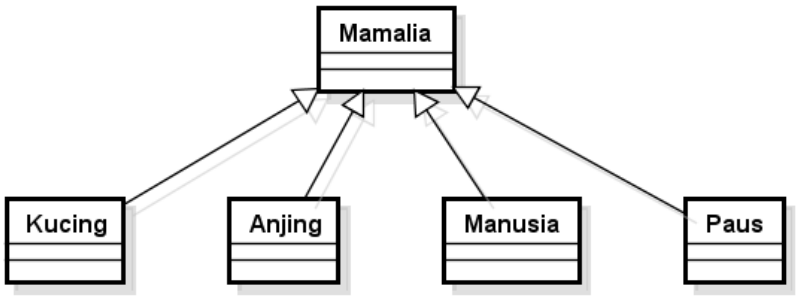
Sistem berorientasi objek menggabungkan sepotong informasi dengan perilaku tertentu yang bertindak atas informasi tersebut. Lalu dirangkum menjadi sebuah objek. Hal ini disebut dengan enkapsulasi. Cara lain untuk melihat enkapsulasi adalah dengan membagi aplikasi menjadi bagian-bagian kecil dari fungsi terkait. Sebagai contoh, informasi yang berkaitan dengan rekening bank, seperti nomor rekening, saldo, nama pelanggan, alamat, jenis account, tingkat bunga, dan tanggal pembukaan. Lalu perilaku untuk rekening bank: membuka, menutup, deposit, withdraw, jenis perubahan, perubahan pelanggan, dan perubahan alamat. Setelah itu, informasi dan perilaku ini dirangkum menjadi sebuah objek akun. Akibatnya, setiap perubahan ke sistem perbankan mengenai rekening hanya dapat diimplementasikan dalam objek akun. Enkapsulasi bekerja seperti *one-stop shop* untuk semua informasi account dan perilaku.



Gambar 2.17 Enkapsulasi Model Penarikan Uang di ATM

b. Pewarisan (*Inheritance*)

Pewarisan adalah prinsip dasar berorientasi objek yang kedua. Dalam sistem berorientasi objek, pewarisan adalah mekanisme yang memungkinkan untuk membuat objek baru berdasarkan objek lama: Objek anak mewarisi sifat-sifat dari objek induk. Salah satu manfaat utama dari pewarisan adalah kemudahan dalam pemeliharaan. Ketika suatu perubahan yang mempengaruhi objek anak, hanya perlu mengubah objek orangtua maka objek anak secara otomatis akan mewarisi perubahannya. Contohnya jika mamalia tiba-tiba menjadi berdarah dingin, hanya objek mamalia akan perlu diubah. Kucing, anjing, manusia, ikan paus, dan benda-benda lainnya secara otomatis objek anak akan mewarisi karakteristik baru, yaitu karakteristik berdarah dingin dari mamalia .



Gambar 2.18 Model pewarisan alami

c. Polimorfisme (*Polymorphism*)

Prinsip ketiga orientasi objek adalah polimorfisme. Dari arti katanya, polimorfisme merupakan terjadinya berbagai bentuk, tahapan, atau jenis. Polimorfisme berarti memiliki banyak bentuk atau implementasi dari fungsi tertentu. Dalam hal sistem berorientasi objek, ini berarti bahwa perancang sistem dapat memliliki banyak implementasi dai fungsi tertentu seperti pada contoh kode untuk menggambar bangun ruang berikut.

```
Function Shape.drawMe()
{
SWITCH Shape.Type
Case "Circle"
  Shape.drawCircle();
Case "Rectangle"
  Shape.drawRectangle();
```

```
Case "Line"
   Shape.drawLine();
END SWITCH
}
```

Dengan polimorfisme, kode untuk menggambar hanya akan memanggil fungsi drawMe() seperti contoh kode berikut.

```
Function draw()
{
    Shape.drawMe();
}
```

Setiap bentuk (lingkaran, persegi panjang, dan garis) akan memiliki fungsi drawMe( ) untuk menggambar bentuk tertentu. Salah satu manfaat dari polimorfisme sama seperti prinsip-prinsip lain dari orientasi objek, yaitu kemudahan pemeliharaan.

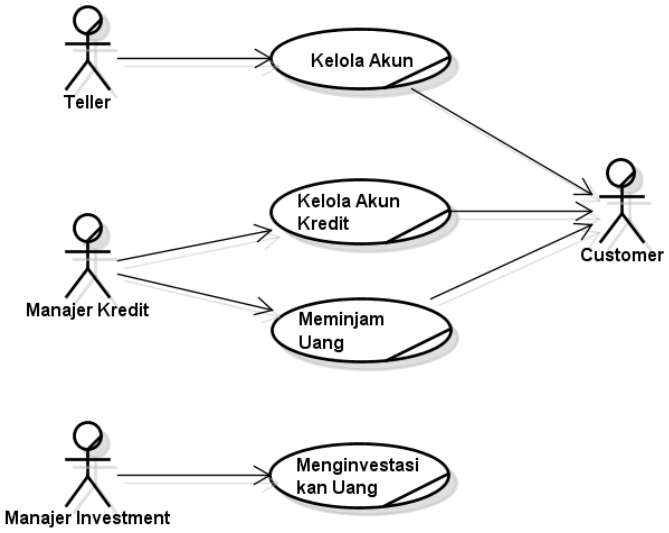
2.11 UML

Unified Modeling Language (UML) adalah bahasa yang telah menjadi standar untuk visualisasi, pemodelan, perancangan dan mendokumentasikan artifak suatu sistem perangkat lunak berbasis objek yang terdiri dari kumpulan diagram, dikembangkan untuk membantu para pengembang sistem [3]. Dikembangkan oleh Grady Booch, Ivar Jacobson dan James Rumbaugh dari Rational Software pada 1990-an. UML telah diadopsi oleh Object Management Group (OMG) pada tahun 2007 dan telah dikelola organisasi ini sejak saat itu. Pada tahun 2000 UML diterima oleh International Organization for Standardization (ISO) sebagai standar untuk pemodelan sistem perangkat lunak intensif.

Hingga saat ini, UML telah berkembang hingga versi 2.5. UML memiliki banyak sekali diagram, terutama versi terbaru UML yaitu UML 2.5. Berikut akan dijelaskan macam-macam diagram yang paling sering digunakan dalam pembangunan aplikasi berbasis objek.

a. *Business Use Case* Diagram

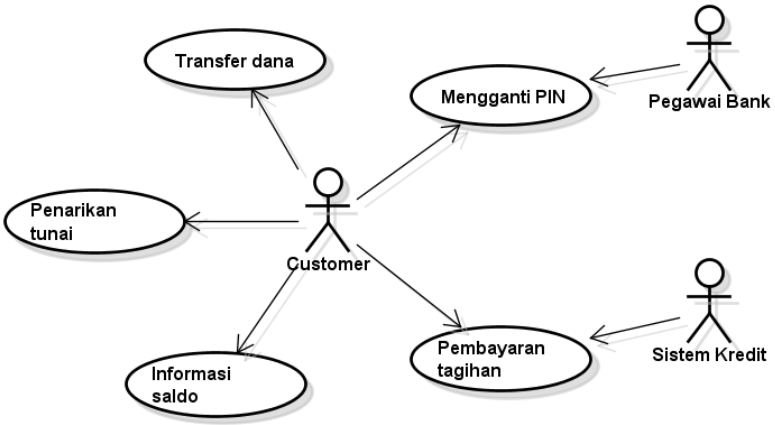
   Business use case digunakan untuk mewakili fungsi yang disediakan oleh organisasi secara keseluruhan. *Business use case* juga digunakan secara luas selama kegiatan pemodelan bisnis untuk mengatur konteks sistem dan untuk membentuk sebuah fondasi untuk menciptakan use case.



Gambar 2.19 Diagram Business Use Case untuk Institusi Finansial

b. *Use Case* Diagram

Use case digunakan untuk memodelkan proses bisnis berdasarkan perspektif pengguna sistem. *Use case* diagram terdiri atas diagram untuk use case dan actor. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi.

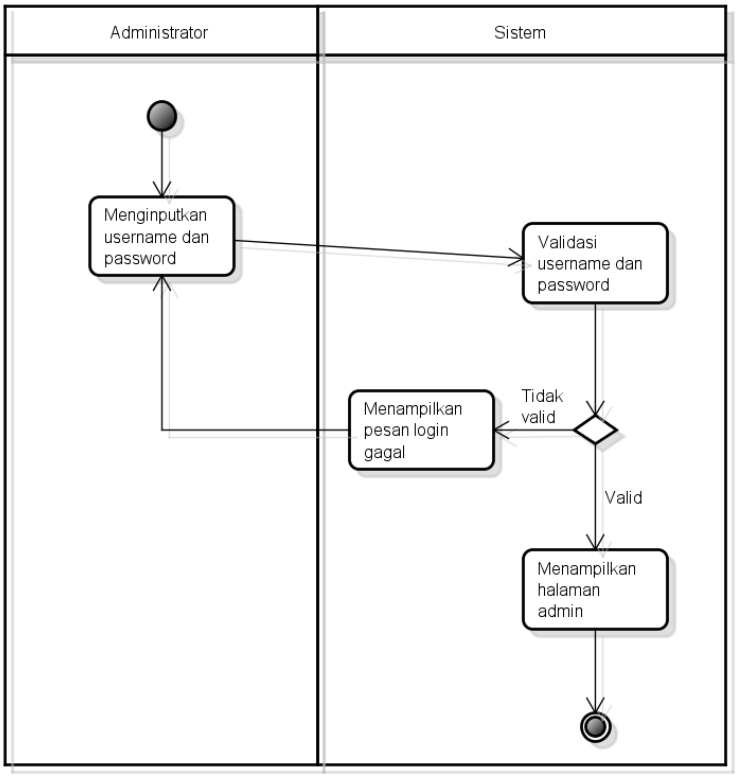


Gambar 2.20 Diagram Use Case untuk Sistem ATM

c. *Activity* Diagram

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat

menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity* diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

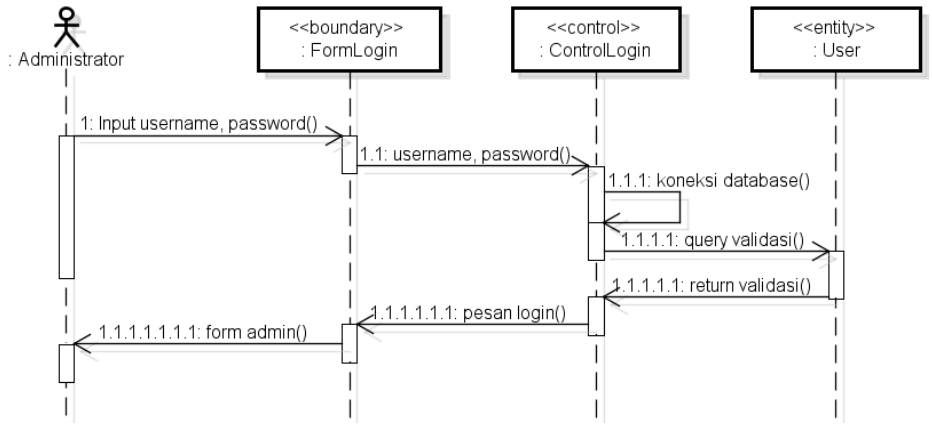


Gambar 2.21 Diagram Aktifitas untuk Login Administrator

d. *Sequence* Diagram

Sequence diagram menjelaskan secara detail urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*, diantaranya:

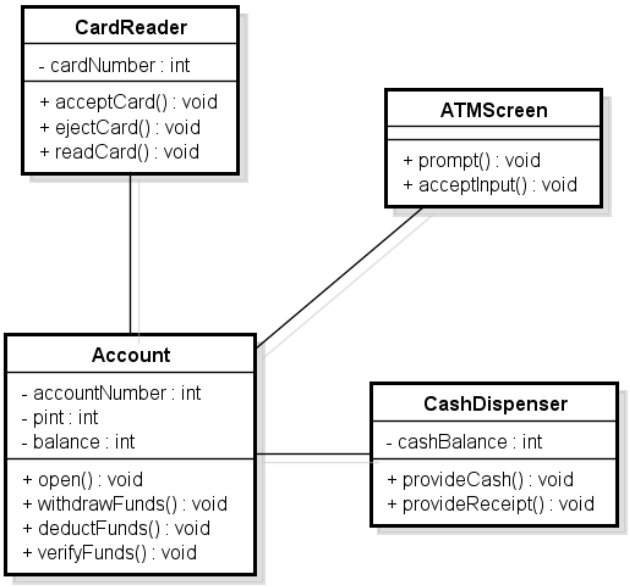
1) Interaksi yang terjadi antar *class*.
2) Operasi apa saja yang terlibat.
3) Urutan antar operasi.
4) Informasi yang diperlukan oleh masing-masing operasi.



Gambar 2.22 Diagram Sequence untuk Proses Login

e. *Class* Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Selain itu, *class* diagram menunjukan juga atribut dan *method* yang ada pada setiap *class*.



Gambar 2.23 Class Diagram untuk Use Case Penarikan Uang Sistem ATM

2.12 PHP

PHP (Hypertext Preprocessor) adalah bahasa pembangun perangkat lunak yang dapat ditanamkan atau disisipkan ke dalam suatu HTML. PHP banyak dipakai untuk membangun suatu situs *web* dinamis atau aplikasi *web*. PHP dapat digunakan untuk membangun sebuah CMS (*Content Management System*).

Pada awalnya PHP merupakan kependekan dari *Personal Home Page*. PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama *Form Interpreted* (FI), yang wujudnya berupa sekumpulan sintaks yang digunakan untuk mengolah data formulir dari web. Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FI. Dengan perilisan kode sumber ini menjadi *open-source*, maka banyak *programmer* yang tertarik untuk ikut mengembangkan PHP.

Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini, interpreter PHP sudah diimplementasikan dalam program C. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP/FI secara signifikan. Pada tahun ini juga, sebuah perusahaan bernama Zend menulis ulang interpreter PHP menjadi lebih bersih, lebih baik, dan lebih cepat. Kemudian pada Juni 1998, perusahaan tersebut merilis interpreter baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0 dan singkatan PHP diubah menjadi akronim berulang PHP: *Hypertext Preprocessing*.

Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi. Pada Juni 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek.

Beberapa kelebihan PHP dari bahasa pemrograman web, antara lain:

a. Bahasa pemrograman PHP adalah sebuah bahasa pembangun yang tidak melakukan sebuah kompilasi dalam penggunaanya.
b. Web Server yang mendukung PHP dapat ditemukan dimana - mana dari mulai Apache, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
c. Pengembangan yang cukup mudah, karena banyaknya milis - milis dan developer yang siap membantu dalam pengembangan.
d. PHP adalah bahasa *scripting* yang paling mudah dipahami karena memiliki referensi yang banyak.
e. PHP adalah bahasa open source yang dapat digunakan di berbagai *platform* (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah sistem.

2.13 Java

Inovasi bahasa komputer dimotivasi oleh dua faktor: perbaikan dalam seni pemrograman dan perubahan dalam lingkungan komputasi, tidak terkecuali Java [10]. Dibangun di atas warisan yang kaya dari C dan C + +, Java menambahkan perbaikan dan fitur yang mencerminkan keadaan seni dalam pemrograman saat ini. Menanggapi munculnya lingkungan online, Java menawarkan fitur yang merampingkan pemrograman untuk arsitektur yang sangat terdistribusi.

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (bytecode) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM).

Versi awal Java ditahun 1996 sudah merupakan versi rilis sehingga dinamakan Java Versi 1.0. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

a. java.lang: Peruntukan kelas elemen-elemen dasar.
b. java.io: Peruntukan kelas input dan output, termasuk penggunaan berkas.
c. java.util: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas kelas penanggalan.
d. java.net: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
e. java.awt: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI)
f. java.applet: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

Seperti telah dibahas sebelumnya, banyak jenis komputer dan sistem operasi yang terhubung ke Internet. Untuk program-program untuk secara dinamis didownload ke semua berbagai jenis platform, beberapa sarana untuk menghasilkan kode dieksekusi portabel diperlukan. Mekanisme yang sama yang membantu menjamin keamanan juga membantu menciptakan portabilitas karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda. Saat ini java merupakan bahasa pemrograman yang populer digunakan dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

Java memliki beberapa kelebihan dibandingkan dengan bahasa pemograman lain, diantaranya:

a. Multiplatform.

   Kelebihan utama dari Java ialah dapat dijalankan di beberapa platform / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan.

b. OOP (*Object Oriented Programming*)

Object Oriented Programming merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Berbeda dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

c. *Library Class* yang Lengkap.

Java terkenal dengan kelengkapan *library* (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan *library* ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat *library* baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

d. Mewarisi kekayaan C/C++.

Memiliki sintaks seperti bahasa pemrograman C/C++ dan mewarisi kekayaan yang dimiliki C/C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java.

e. Pengumpulan sampah otomatis

Memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

Selain dari beberapa kelebihan yang dimiliki oleh java, terdapat pula beberapa kekurangan yang dimiliki java, diantaranya:

a. Mudah didekompilasi.

Dekompilasi adalah proses membalikkan hasil kompilasi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan bytecode

yang menyimpan banyak atribut bahasa tingkat tinggi. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/direverse-engineer.

b. Penggunaan memori yang banyak.

   Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal. Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru, tetapi bagi komputer yang telah berumur lebih dari 4 tahun akan menyebabkan masalah, karena ketersesiaan memori yang terbatas.