BAB II

LANDASAN TEORI

2.1 *Virtual Store*

Virtual Store atau toko *virtual* merupakan toko ritel tanpa wujud fisik yang tidak melibatkan tatap muka langsung antara produsen atau penjual dengan konsumen. Belanja melalui media seperti katalog (*catalog shopping),* Televisi (*TV Shopping*), *Vending machine* dan Internet (*Online Shopping*) merupakan contoh dari belanja melalui toko *virtual*.

Tabel 2.1 Perbandingan Fitur Toko Ritel Berwujud Fisik dengan Toko *Virtual* (Lohse, G.L. & Spiller, P., 1999)

Fitur	**Toko Ritel Berwujud Fisik**	**Toko *Virtual***
Pelayanan dan pemberian informasi saat pembelian.	Diberikan oleh seorang *service provider* atau *salesclerk service.*	Diberikan melalui detil informasi dan deskripsi produk yang komprehensif. Seorang *online service provider* melalui media email atau telepon *hotline*
Sales Promotion sebagai alat promosi.	Berupa penawaran special terkait dengan pembelian, yang diberikan baik didalam toko saat pembelian maupun diluar melalui media seperti kupon dan sebagainya.	Berupa penawaran special terkait dengan pembelian yang diberikan melalui media tertentu seperti *online game*, *online lotteries*, hubungan langsung ke *website* lain yang menarik dan sebagainya.

Lanjutan Tabel 2.1 Perbandingan Fitur Toko Ritel Berwujud Fisik dengan Toko *Virtual* (Lohse, G.L. & Spiller, P., 1999)

Fitur	**Toko Ritel Berwujud Fisik**	**Toko *Virtual***
Penempatan produk atau *display*	Diatur dengan tampilan fisik yang menarik dideret rak berdasarkan pembagian katagori tertentu. Layout toko mempertimbangkan sisi estetika dan fungsional	Diatur melalui desain dan kemasan homepage yang menarik dan ditempatkan dengan tingkat hirarki berdasar kategori tertentu. *Layout website* mempertimbangkan sisi estetika, ketajaman gambar, animasi dan fungsional.
Besar dan luas toko	Jumlah dan luas lantai	Jumlah tingkat hirarki
Pengecekan total pembelian sebelum melakukan pembayaran	Dilakukan oleh petugas kasir	Dilakukan dengan melihat formulir order dan kerangjang belanja
Bentuk fisik produk	Diperoleh dengan melihat dan memegang produk secara langsung	Diperoleh melalui gambar dan deskripsi produk. Terkadang ditunjang dengan audio dan video dari produk.

Perbedaan paling mendasar dari toko *virtual* dengan toko ritel terletak pada wujud fisik toko, namun pada praktiknya toko *virtual* memiliki fitur yang sama dengan toko ritel berwujud fisik namun dalam bentuk yang berbeda, tabel 2.1 menunjukkan perbandingan fitur – fitur di toko ritel berwujud fisik dan toko *virtual*.

Belanja melalui internet menawarkan keuntungan yang unik. Keuntungan tersebut diantaranya adalah kustomisasi dalam melayani konsumen, kenyamanan berbelanja dimana saja dan kapan saja serta biaya yang lebih rendah terutama dalam mengakses informasi (Elliot dan Fowell, 2000). Secara garis besar keuntungan berbelanja melalui internet dapat dikategorikan menjadi dua kategori, *intrinsic benefit* dan *extrinsic benefit*. *Extrinsic benefit* meliputi pilihan produk yang banyak, harga yang kompetitif dan akses memperoleh informasi yang mudah dan berbiaya rendah. Sedangkan *intrinsic benefit* meliputi desain dan tampilan toko yang menarik (Shang et al., 2005).

Seperti telah dijelaskan pada Tabel 1 bahwa toko *online* memiliki semua fitur di toko ritel berwujud fisik namun tetap saja terdapat perbedaan antara lain penyampain fitur di toko *online*. Hal ini menyebabkan perbedaan tingkat kepuasan dari konsumen. Misalnya pada fitur bentuk fisik produk, meskipun toko *online* sudah memberi informasi mengenai bentuk fisik produknya, namun karena keterbatasan teknis seperti kualitas gambar terkadang hal ini tidak dapat memberi kepuasan yang sama dengan toko ritel berwujud fisik (Lohse and Spiller, 1999, Li et al., 1999)

2.2 Sistem Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *linux* yang mencakup sistem operasi, *middleware* dan aplikasi (Safaat, 2011). Android menyediakan *platform* yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Aliance* yang merupakan konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, Google merilis kode – kode Android di bawah lisensi *Apache*, sebuah lisensi perangkat lunak dan open *platform* seluler.

Di dunia terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Service* (GMS) dan kedua adalah yang benar – benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (Safaat, 2011).

Sekitar September 2007 Google mengenalkan Nexus One, salah satu jenis *smartphone* yang menggunakan Android sebagai sistem operasinya. Telepon selular ini diproduksi oleh HTC *Corporation* dan tersedia di pasaran pada 5 Januari 2008. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam progam kerja Android ARM Holdings, Atheros Communication, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericson, Toshiba Corp, dan

Vodafone Group Plc. Seiring pembentukan *Open Handset Alliance*, OHA mengumumkan produk perdana mereka Android, perangkat *mobile* yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis telah dilakukan berbagai pembaharuan berupa perbaikan *bug* dan penambahan fitur baru.

Pada massa saat ini kebanyakan vendor – vendor *smartphone* sudah memproduksi *smartphone* berbasis android, vendor – vendor itu antara lain HTC, Motorola, Samsung, LG, Dell, Sony Ericson, Acer, Asus dan masih banyak lagi vendor *smartphone* di dunia yang memproduksi Android. Hal ini karena Android itu adalah sistem operasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun (Safaat, 2011).

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini Android menjadi pesaing utama dari Apple pada sistem operasi *Tablet* PC. Pesatnya pertumbuhan Android selain faktor yang disebutkan diatas adalah karena Android itu sendiri adalah platform yang lengkap, baik itu sistem operasinya, aplikasi dan tools pengembangannya, market aplikasi Android serta dukungan yang sangat tinggi dari komunitas *open source* dunia (Safaat, 2011). Sehingga Android terus berkembang pesat baik dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia.

2.2.1 *The dalvik Virtual Machine* (DVM)

Salah satu elemen kunci dari Android (Safaat, 2011) adalah *Dalvik Virtual Machine* (DVM). Android berjalan di dalam *Dalvik Virtual Machine* (DVM) bukan di *Java Virtual Machine* (JVM), sebenarnya banyak persamaannya dengan *Java Virtual Machine* (JVM) seperti Java ME (*Java Mobile Edition*), tetapi menurut Safaat H, Nazruddin (2011:11) Android menggunakan *Virtual Machine* sendiri

yang dikustomisasi dan dirancang untuk memastikan bahwa beberapa *feature-feature* berjalan lebih efisien pada perangkat *mobile*.

Dalvik Virtual Machine (DVM) adalah "*register bases*" sementara *Java Virtual Machine* (JVM) adalah "*stack bases*", DVM didesain dan ditulis oleh Dan Bornsten dan beberapa *engineers* Google lainnya. Jadi bisa dikatakan "*Dalvik equals(Java) == False*". *Dalvik Virtual Machine* (DVM) menggunakan kernel Linux untuk menangani fungsionalitas tingkat rendah termasuk keamanan, *threading* dan proses serta manjemen memori (Safaat, 2011). Ini memungkinkan kita untuk menulis aplikasi C/ C+ sama halnya seperti pada OS Linux kebanyakan.

Semua hardware yang berbasis Android dijalankan dengan *Virtual Machine* untuk eksekusi aplikasi, pengembang tidak perlu khawatir tentang implementasi perangkat keras tertentu. DVM mengeksekusi *executable file*, sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *The executable file* diciptakan dengan mengubah kelas bahasa java dan dikompilasi menggunakan *tools* yang disediakan dalam SDK Android (Safaat, 2011).

2.2.2 Android SDK (*Software Development Kit*)

Android SDK (Safaat,2011) adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *midlleware* dan aplikasi kunci yang di-release oleh Google. Saat ini disediakan Android SDK sebagai alat bantu dan API untuk mulai mengembangkan aplikasi Android menggunakan bahasa pemrograman Java. Beberapa fitur dari Android yang paling penting adalah (Safaat, 2011).

1. *Framework* Aplikasi yang mendukung penggantian komponen dan *reusable*.
2. Mesin *Virtual* Dalvik dioptimalkan untuk perangkat *mobile*.
3. *Integrated browser* berdasarkan *engine open source WebKit*.
4. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1,0 (*Opsional akselerasi hardware*).
5. SQLite untuk penyimpanan data (*Database*).
6. Bluetooth, EDGE, 3G dan WiFi (tergantung *hardware*).
7. Kamera, GPS, kompas dan *accelerometer* (tergantung *hardware*).

   Lingkungan development yang lengkap dan kaya termasuk perangkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE Eclipse.

2.2.3 Arsitektur Android

Secara garis besar arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut (Safaat, 2011) :

1. *Applications* dan *Widgets*

   Applications dan *Widgets* ini adalah *layer* dimana pengguna berhubungan dengan aplikasi saja, dimana biasanya pengguna *download* aplikasi kemudian melakukan instalasi dan menjalankan aplikasi tersebut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

2. *Applications Frameworks*

   Applications frameworks adalah *layer* dimana para pembuat aplikasi melakukan pengembangan/ pembuatan aplikasi yang akan dijalankan di sistem

operasi Android, karena pada *layer* inilah aplikasi dapat dirancang dan dibuat seperti *content-providers* yang berupa sms dan panggilan telepon.

3. *Libaries*

   Libraries ini adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya.

4. Android *Run Time*

   Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux. *Dalvik Virtual machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Pada Android *Run Time* ada dua bagian yaitu *Core Libraries* dan *Dalvik Virtual machine.*

5. Linux Kernel

   Linux kernel adalah *layer* dimana inti dari *operating system* dari Android itu berada. Berisi file-file sistem yang mengatur sistem *processing*, *memory*, *resources*, *drivers* dan sistem-sistem operasi Android lainnya.

2.2.4 Versi Android

Adapun versi-versi Sistem Operasi Android yang pernah dirilis dan masih digunakan sampai sekarang adalah sebagai berikut (Safaat, 2011):

1. Android Versi 1.1

   Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

2. Android Versi 1.5 (Cupcake)

   Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi 1.5 yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke headset Bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan sistem.

3. Android Versi 1.6 (Donut)

   Versi 1.6 menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol *applet* VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus; kamera, camcorder dan galeri yang dintegrasikan; CDMA / EVDO, 802.1x, VPN, Gestures; dan Text-to-speech engine, kemampuan dial kontak, teknologi text to change speech; serta pengadaan resolusi VWGA.

4. Android Versi 2.0/ 2.1 (Eclair)

   Perubahan yang dilakukan adalah pengoptimalan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML5, daftar kontak yang baru, dukungan flash untuk kamera 3,2 MP, digital Zoom, dan Bluetooth 2.1.

5. Android Versi 2.2 (Froyo: Frozen Yoghurt)

   Perubahan pada versi ini antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, *intergrasi* V8 *JavaScript engine* yang dipakai Google Chrome yang mempercepat kemampuan *rendering* pada *browser*, pemasangan aplikasi dalam SD Card, kemampuan

WiFi Hotspot portable, dan kemampuan *auto update* dalam aplikasi Android *Market*.

6. Android Versi 2.3 (Gingerbread)

   Perubahan yang didapat dari Android versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antar muka (*User Interface*) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (*reverb*, *equalization*, *headphone virtualization*, *dan bass boost*), dukungan kemampuan *Near Field Communication* (NFC), dan dukungan jumlah kamera yang lebih dari satu.

7. Android Versi 3.0/ 3.1 (Honeycomb)

   Android Honeycomb dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada Honeycomb juga berbeda karena sudah didesain untuk tablet. Honeycomb juga mendukung multi prosesor dan juga akselerasi perangkat keras (*hardware*) untuk grafis.

8. Android Versi 4.0 (ICS: Ice Cream Sandwich)

   Membawa fitur Honeycomb untuk smartphone dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari email secara offline, dan berbagi informasi dengan menggunakan *Near Field Communication* (NFC).

Android SDK (Safaat,2011) adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *midlleware* dan aplikasi

kunci yang di-*release* oleh Google. Saat ini disediakan Android SDK sebagai alat bantu dan API untuk mulai mengembangkan aplikasi Android menggunakan bahasa pemrograman Java. Beberapa fitur dari Android yang paling penting adalah (Safaat, 2011) :

1. *Framework* Aplikasi yang mendukung penggantian komponen dan *reusable*.
2. Mesin *Virtual Dalvik* dioptimalkan untuk perangkat *mobile*.
3. *Integrated browser* berdasarkan *engine open source WebKit*.
4. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1,0 (*Opsional akselerasi hardware*).
5. SQLite untuk penyimpanan data (*Database*).
6. Media Support yang mendukung audio, video dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM *Telephony* (tergantung *hardware*).
7. Bluetooth, EDGE, 3G dan WiFi (tergantung *hardware*).
8. Kamera, GPS, kompas dan accelerometer (tergantung *hardware*).

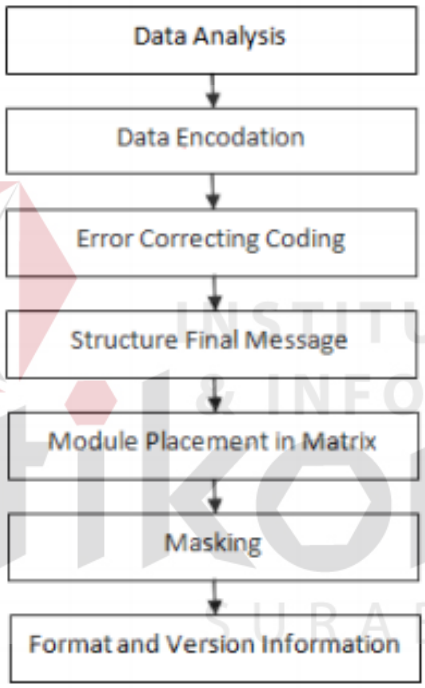
Lingkungan development yang lengkap dan kaya termasuk perangkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan plugin untuk IDE Eclipse.

2.3 *Barcode*

Barcode Atau dalam Bahasa Indonesia seringkali disebut kode batang adalah an *optical machine-readable representation of data*. Kode berbentuk garis dan berwarna hitam putih tersebut mengandung satu kumpulan kombinasi yang berlainan ukuran, dan disusun sedemikian rupa menurut aturan tertentu sehingga dapat diterjemahkan oleh mesin pembacanya (Wahyono, 2010)

2.3.1 *QR Code*

QR Code adalah *image* berupa matriks dua dimensi. *QR Code* merupakan evolusi dari kode batang (*barcode*). *QR Code* merupakan singkatan dari *Quick Response Code*, atau dapat diterjemahkan menjadi kode respon cepat. *QR Code* dikembangkan oleh Denso Corporation, sebuah perusahaan Jepang yang banyak bergerak di bidang otomotif. (Soon, 2008). Prosedur *QR Code* dari sebuah teks dapat dijelaskan dengan diagram alir pada gambar 2.1.



Gambar 2.1 Diagram alir proses pembangkitan *QR Code*

2.3.2 *QR Code Reader*

Untuk Proses Pembacaan *QR Code* diperlukan sebuah pembaca atau pemindai berupa *software* yaitu *QR Code Reader* atau *QR Code Scanner* yang harus diinstal pada perangkat telepon *mobile* (*mobile phone*). Didalam software tersebut terdapat fungsi untuk mengkonversi *QR Code* menjadi *text* yang dapat dibaca manusia.

2.3.3 *QR Code Generator*

Untuk menghasilkan suatu gambar atau symbol *QR Code*, suatu informasi di-*encode* melalui suatu aplikasi *QR Code Generator*. Beberapa aplikasi *QR Code Generator* yang dapat digunakan, contohnya google chart API yang memang sudah mendukung *QR Code*. Google Chart API agar dapat bekerja dan menghasilkan *QR Code* harus terkoneksi dengan *internet* salah satunya dengan melalui *internet browser*.

Tabel 2.2 Tabel Parameter Permintaan Google Chart API (Google Inc, 2011)

Parameter	**Dibutuhkan atau Opsional**	**Keterangan**
Cht=qr	Dibutuhkan	Menentukan sebuah *QR Code*
Chs=<lebar>x<tinggi>	Dibutuhkan	Ukuran Image
Chl=<data>	Dibutuhkan	Data yang akan di *encode*. Data dapat berupa digit (0-9), karakter alfanumerik, data *binary*, atau kanji.
Choe=<output_encoding>	Opsional	Bagaimana untuk men*encode* data pada *QR Code*. Nilai yang tersedia yaitu UTF-8(default), Shift_JIS, ISO-8859-1

Berikut ini adalah algoritma untuk membuat *QR Code* :

a. Menentukan Kapasitas

Kapasitas dari *QR Code* ditentukan oleh versi, tingkat koreksi kesalahan dan tipe data yang akan dikodekan (misalnya *numeric*, *alfanumerik*, dan lain-lain). Sebagai contoh pada *QR Code* Versi 1 dengan tingkat koreksi kesalahan Q, 27

karakter *numeric* dapat disimpan atau 16 karakter *alfanumerik* dapat disimpan dan 11 data *byte* dapat juga disimpan yang tampak pada tabel 2.2. Sebaliknya, versi meningkat bila tingkat kesalahan lebih besar pada data yang sama. Jadi, pertama perlu mempertimbangkan tingkat mengoreksi kesalahan, dan selanjutnya kita mempertimbangkan versi jika perlu (swetake,2011).

b. *Encode Data*

Pada bagian ini, data di-*encode* dengan melakukan perhitungan-perhitungan dan menempatkannya kedalam *QR Code*.

1. Menentukan tipe data

   Data akan dibaca tipe datanya. Masing – masing tipe data akan disimpan kedalam representasi bilangan biner 4 bit dan mempunyai panjang karakter penyimpanan tertentu. Tipe data tersebut terdapat pada table 2.3.

Tabel 2.3 Tabel Tipe Data *QR Code* (Swetake, 2011)

Tipe Data	Representasi data 4 bit	Panjang penyimpanan
Numerik	0001	10bit
Alphanumerik	0010	9bit
Biner(8bit)	0100	8bit
Kanji	1000	8bit

2. Konversi data ke dalam bentuk biner

   Data yang telah diketahui tipe datanya akan dikonversikan kedalam biner berdasarkan table 2.1. Misalnya, data yang mempunyai tipe data numerik akan dikonversikan kedalam 10bit biner.

3. Konversi biner kedalam bentuk *decimal*

   Data yang sudah terkonversi kedalam bentuk biner, akan dirubah kedalam bentuk *decimal* berdasarkan kapasitas dari masing – masing versi *QR Code* yang terlah ditentukan.

4. Menghitung tingkat koreksi kesalahan

   Tingkat koreksi kesalahan ditentukan dengan menggunakan metode *Reed Solomon* berdasarkan versi *QR Code* yang digunakan dengan toleransi kesalahan maksimal yang digunakan adalah 30%.

5. Alokasi data

   Data hasil *encode* akan dialokasikan kedalam bentuk gambar *QR Code*. Data yang akan dialokasikan adalah data hasil *representasi* biner dan data hasil perhitungan koreksi kesalahan. Aturan peletakan data dalam *QR Code* adalah sebagai berikut:

   1. Data akan dialokasikan kedalam *matriks* dengan ukuran sesuai kapasitas data pada versi *QR Code*.
   2. Data pertama kali akan diletakkan pada koordinat pojok kanan bawah.
   3. Data selanjutnya akan diletakkan diatasnya.
   4. Jika pada peletakan awal *QR Code* telah terdapat data sebelumnya, maka peletakkan data akan dimulai pada modul yang kosong dan mempunyai arah dari kiri ke kanan. Jika sebelah kanan penuh maka arah selanjutnya adalah keatas. Jika bagian atas penuh maka arah selanjutnya adalah ke bawah dengan tetap memperhatikan arah peletakkan data yaitu dari kanan ke kiri.

6. Penentuan pola data

Penentuan pola data *QR Code* (*Finder Pattern*) dilakukan berdasarkan kondisi pada table 2.4 Jika kondisi pada table 2.4 tidak terpenuhi, maka pola data tidak akan disimpan pada QR Code sebagai *finding pattern*.

Tabel 2.4 Tabel Pola Data *QR Code* (Swetake, 2011)

Pola Data	**Kondisi**
000	(i+j) mod 2 = 0
001	I mod 2 = 0
010	J mod 3 = 0
011	(i+j) mod 3 = 0
100	((i div 2) + (j div 3)) mod 2 = 0
101	(ij) mod 2 + (ij) mod 3 = 0
110	((ij) mod 2 + (ij) mod 3) mod 2 = 0
111	((ij) mod 3 + (i+j) mod 2) mod 2 = 0

7. Penentuan format informasi data

Indikator pembentuk pola sebanyak 15bit, yang terdiri dari 2 bit untuk koreksi kesalahan, 3 bit untuk pembentuk pola, dan 10bit.

Tabel 2.5 Tabel Format Informasi *QR Code* (Swetake, 2011)

	Level Koreksi Kesalahan	**Indikator**
1	L	01
2	M	00
3	Q	11
4	H	10

c. *Decode Data*

Proses *decode* adalah proses pembacaan data *QR Code*. Langkah – langkah pembacaan data *QR Code* diantaranya adalah:

1. Mengkonversi gambar *QR Code* kedalam bentuk *Grayscale*.
2. Proses binerisasi gambar *grayscale*.
3. Mencari lokasi dan orientasi keberadaan data.
4. Mencari pola baris data
5. Proses *decode*

Adapun proses *decode* tersebut merupakan kebalikan dari proses *encode*.

2.3.4 *Photo Scanner* Pada *Smartphone*

Sementara kamera ponsel tanpa auto-fokus yang tidak ideal untuk membaca beberapa format *barcode* umum, ada 2D *barcode* yang dioptimalkan untuk ponsel, serta *QR Codes* dan kode *Data Matrix* yang dapat membaca dengan cepat dan akurat dengan atau tanpa *auto* fokus. Ini membuka beberapa aplikasi untuk konsumen:

1. Film: DVD / VHS film katalog
2. Musik: katalog CD, memutar MP3 ketika dipindai
3. Buku katalog dan perangkat.
4. Bahan makanan, informasi nutrisi, membuat daftar belanja saat terakhir digunakan, dll
5. Properti inventaris pribadi (untuk asuransi dan tujuan lainnya) dipindai ke dalam perangkat lunak pribadi. Kemudian, penerimaan gambar scan dapat secara otomatis terkait dengan entri yang sesuai. Dimana, kode *barcode* dapat

digunakan dengan cepat untuk menyingkirkan kertas fotokopi jadi tidak perlu disimpan untuk tujuan pajak atau inventaris aset.

6. Sejumlah aplikasi perusahaan menggunakan ponsel: Kontrol akses (misalnya, validasi tiket di tempat), pelaporan persediaan (misalnya, pelacakan pengiriman), pelacakan aset (misalnya, anti-pemalsuan)

2.3.5 Pengolahan Citra Digital

Pengertian sederhana dari pengolahan citra adalah manipulasi dan anilisis informasi gambar oleh komputer. Informasi gambar adalah gambar visual dalam dua dimensi. Segala operasi untuk memperbaiki, analisis, atau penguahan suatu gambar disebut *image processing*.

a. Deteksi Tepi (*Edge Detection*)

Deteksi Tepi (*Edge Detection*) adalah suatu proses yang menghasilkan tepi-tepi dari objek-objek gambar. Suatu titik (x,y) dikatakan suatu tepi (edge) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangga.

b. Grayscalling

Grayscaling adalah teknik yang digunakan untuk mengubah citra berwana menjadi bentuk *grayscale* atau tingkat keabuan (dari hitam - putih).

c. Thresholding

Thresholding digunakan untuk mengubah intensitas piksel menjadi salah satu dari dua nilai, a 1atau a 2. Hasil dari proses 4 thresholding ditentukan oleh suatu parameter yang disebut nilai threshold.

2.4 Pustaka ZXing (*Zebra Crossing*)

ZXing atau *ZebraCrossing* merupakan pustaka *open source* untuk pemrosesan *barcode* 1D/2D (yang diimplementasikan menggunakan bahasa Java. Pustaka ini fokus pada penggunaan kamera untuk memindai dan melakukan *decode QR Code* pada *smartphone* tanpa memerlukan komunikasi dengan *server*. Pustaka ZXing mendukung beberapa format *barcode*, antara lain: *UPC-A, UPC-E, EAN-8, EAN-13, Code 39, Code 93, Data Matrix, QR Code*, dan *Code Bar*.

2.5 *Web Service*

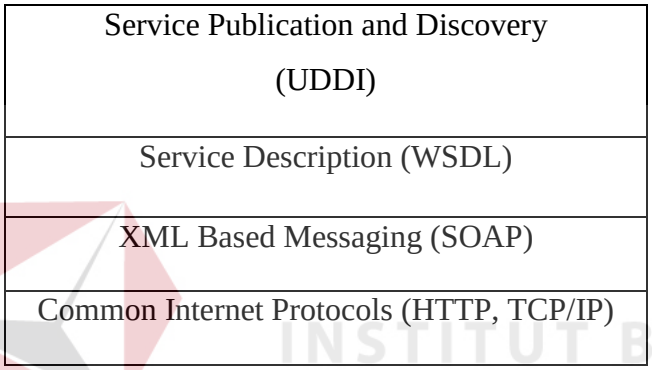
Web service merupakan suatu komponen *software* yang merupakan *self-containing*, aplikasi modular *self-describing* yang dapat dipublikasikan, dialokasikan, dan dilaksanakan pada web (wahli et al, 2006)

Web service adalah teknologi yang mengubah kemampuan internet dengan menambahkan kemampuan *transactional web*, yaitu kemampuan *web* untuk saling berkomunikasi dengan pola *program-to-program (P2P).* Fokus *web* selama ini didominasi oleh komunikasi *program-to-user* dengan interaksi *business-to-consumer* (B2C), sedangkan *transactional web* akan didominasi oleh *program-to-program* dengan interaksi *business-to-business* (Gottschalk, 2002).

Gambar 1 merupakan block bangunan *web service* yang mana menyediakan fasilitas komunikasi jarak jauh Antara dua aplikasi yang merupakan *layer Arsitektur web service* (Tidwell, 2001)

1. Layer 1 : Protokol internet standar yang digunakan sebagai sarana transportasi adalah HTTP dan TCP/IP

2. Layer 2 : *Simple Object Acces Protocol* (SOAP) berbasiskan XML dan digunakan untuk pertukaran informasi antar sekelompok layanan.
3. Layer 3 : *Web Service Definition Language* (WSDL) digunakan untuk mendiskripsikan attribute layanan.
4. Layer 4 : *Universal Description, Discovery* and *Integration*, yang mana merupakan direktori pusat untuk deskripsi layanan.



Gambar 2.2 Blok Bangunan *Web Service*

2.6 MySQL

Pengertian mysql adalah *relational database management sistem* yang cepat dan kuat. Sebuah basis data dapat membuat pengguna untuk menyimpan, mencari, mengurutkan dan mendapatkan data dengan sangat efisien. Server mysql mengendalikan akses kedalam data untuk memastikan bahwa para pengguna dapat bekerja dalam waktu yang bersamaan, untuk mendukung akses secara cepat dan memastikan hanya pengguna yang telah terisolasilah yang mendapatkan hak akses.

Mysql menggunakan bahasa SQL (*structured query language*), yaitu bahasa *query* basis data yang baku bagi seluruh dunia. Mysql kembali diduplikasikan sejak tahun 1996, tetapi sejarah pengembangannya telah dilakukan dari tahun 1979.

Mysql tersedia dengan lisensi *open source*, tetapi lisensi komersialpun tersedia apabila diperlukan (Welling, 2001, p3)

Kelebihan mysql menurut Welling (2001, p5), terdapat beberapa kelebihan dari mysql, yaitu:

1. *Performance*

   Mysql begitu cepat dalam pemrosesan data

2. *Low cost*

   Mysql tersedia dan dapat digunakan tanpa dikenakan biaya, dibawah lisensi *open source*. Namun, tersedia dengan biaya yang sangat murah, dibawah lisensi komersial, jika aplikasi yang pengguna gunakan dibutuhkan.

3. *Ease of use*

   Kebanyakan dari berbagai sistem basis data modern menggunakan sql. Jika pengguna memilih dbms yang lain, pengguna tidak akan menghadapi masalah yang berarti ketika beradaptasi dengan sql. Bahkan mysql lebih mudah dalam persiapan dibandingkan dengan produk lain yang sekelas.

4. *Probability*

   Mysql dapat digunakan dan diimplementasikan pada berbagai sistem UNIX dan juga pada *Microsoft Windows*.

5. *Source Code*

   Sama seperti pemrograman PHP, pengguna dapat mengubah dan menambahkan *source code* bagi MySQL

2.7 UML (*Unfied Modeling Language*)

2.7.1 Pengertian UML

UML (*Unified Modelling Language*) adalah sebuah bahasa untuk menentukan, visualisasi, kontruksi, dan mendokumentasikan *artifacts* dari sistem *software*, untuk memodelkan bisnis, dan sistem non*software* lainnya. UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks (Suhendar & Gunadi, 2002).

2.7.2 *Artifact* UML

Artifact adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifact* dapat berupa model, deskripsi, atau *software*. Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut:

1. *Use-case diagram*
2. *Class diagram*
3. *Behavior diagram*
    1. *Statechart diagram*
    2. *Activity diagram*
    3. *Interaction diagram:*
        1. *Sequence diagram*
        2. *Collaboration diagram*
4. *Implementation diagram*
    1. *Component diagram*
    2. *Deployment diagram*

Diagram-diagram tersebut diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analisis atau rekayasa. Dibuatnya berbagai jenis diagram diatas karena:

1. Setiap sistem yang kompleks selalu paling baik jika didekati melalui himpunan berbagai sudut pandang yang kecil yang satu sama lain hampir saling bebas (*independent*). Sudut pandang tunggal senantiasa tidak mencakupi untuk melihat sistem yang besar dan kompleks.
2. Diagram yang berbeda-beda tersebut dapat menyatakan tingkatan yang berbeda-beda dalam proses rekayasa.
3. Diagram-diagram tersebut dibuat agar model yang dibuat semakin mendekati realistas.

2.7.3 Faktor yang Mendorong Dibuatnya UML

1. Pentingnya Model
    1. Membangun model untuk suatu sistem *software* (terlebih *software* untuk suatu organisasi bisnis) sangat bergantung pada konstruksinya atau kemudahan dalam memperbaikinya. Oleh karena itu, membuat model sangatlah penting sebagaimana pentingnya kita memiliki cetak biru untuk suatu bangunan yang besar.
    2. Model yang bagus sangat penting untuk menghasilkan komunikasi yang baik antar anggota tim dan untuk meyakinkan sempurnanya arsitektur sistem yang dibangun.
    3. Jika kita membangun suatu model dari suatu sistem yang kompleks, tidak mungkin kita dapat memahaminya secara keseluruhan. Dengan

meningkatnya komplektisitas sistem, visualisasi dan pemodelan menjadi sangat penting. UML dibuat untuk merespon kebutuhan tersebut.

2. Kecenderungan Dunia Industri terhadap *Software*

    1. Sebagai suatu nilai yang strategis bagi pasar *software* adalah dengan meningkatnya kebutuhan dunia industri untuk memiliki teknik otomatisasi dengan *software*. Oleh karena itu, penting sekali adanya teknik rekayasa *software* yang dapat meningkatkan kualitas dan mengurangi biaya dan waktu. Termasuk dalam teknik rekayasa *software* adalah teknik *visual programming*, juga pembuatan *framework* dan pola.
    2. Komplektisitas dunia industri semakin meningkat karena bertambah luasnya runag lingkup dan tahapan proses. Satu motivasi kunci bagi para pembangun UML adalah untuk membuat suatu himpunan semantik dan notasi yang mampu menangani kerumitan arsitektural dalam semua ruang lingkup.

3. Terjadinya Konvergensi Metode dan *Tool* Pemodelan

    Sebelum adanya UML, terdapat ketidakjelasan bahasa pemodelan apa yang paling unggul. Para user harus memilih diantara bahasa pemodelan dan *tool* (*software*) pemodelan yang banyak dan mirip UML dibuat untuk membuat integrasi baru dalam bahasa pemodelan antar *tool* dan “proses”.

2.7.4 Tujuan UML

Tujuan utama UML diantaranya adalah:

1. Memberikan model yang siap pakai, bahasa pemodelan *visual* yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.

2. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.