BAB 2
LANDASAN TEORI

2.1. Pengantar

Pada bab ini akan dijelaskan mengenai beberapa teori yang ada pada *literature* yang dijadikan acuan untuk pembangunan sistem, diantaranya meliputi teknologi *GPRS* (*General Packet Radio Service*), teknologi *Java* dan *J2ME (Java 2 Micro Edition)*, *Mobile Marketing, Mobile Sales Assistant.*

2.2. Mobile Marketing

Mobile Marketing merupakan suatu bentuk pemasaran produk dengan menggunakan media telepon seluler. Telekomunikasi seluler memang bersifat lebih personal. Telekomunikasi seluler menawarkan jalur promosi yang bersifat *one-to-one*. Oleh karena itu perusahaan mulai melirik komunikasi dengan media ini untuk meningkatkan efektivitas pemasaran, salah satunya untuk peningkatan *brand.*

Mobile Marketing empat fungsi penting, yaitu:

1. Mobile advertising

   Saat ini, penggunaan advertising dapat dilakukan dengan menggunakan media seperti *mobile internet, mobile search, mobile radio, mobile TV, mobile games, dan ring back tones*.

2. Mobile Sales Promotion

   Sales Promotion adalah bentuk komunikasi pemasaran yang bertujuan untuk mendorong terjadinya peningkatan nilai pembelian produk dan jasa.

3. Mobile Direct Marketing

   Direct Marketing adalah bentuk komunikasi langsung dengan konsumen untuk meningkatkan loyalitas. Perusahaan dapat menerapkan program peningkatan loyalitas pelanggan dengan menggunakan media *Mobile Marketing* karena terbukti mampu meningkatkan hubungan langsung dari perusahaan kepada konsumen.

4. Mobile Customer Relationship Management(CRM)

   Customer Relationship Management(CRM) menjadi aktivitas mutlak yang harus dilakukan perusahaan jika ingin meningkatkan loyalitas pelanggan.

(Sumber:http://www.mobeeindonesia.com/Pdf/White%20Paper%20ODP%20dalam%20dunia%20Mobile%20Marketing.pdf)

2.3. Mobile Sales Assistant

Mobile Sales Assistant (MSA) adalah sebuah sistem informasi *mobile* produk untuk pedagang yang berdasar dari kombinasi *Near Field Communication (NFC)* and *the Electronic Product Code (EPC)*. *MSA* bertujuan pada pengoptimasian proses penjualan untuk toko-toko *retail*. *MSA* memungkinkan asisten toko untuk meneliti ketersediaan dan informasi stok produk secara langsung pada *Point of Sale (PoS)* dengan sebuah *NFC* yang *enabled* dengan *mobile phone*. Dengan demikian asisten toko bisa menginformasikan kepada pelanggan tanpa membiarkan mereka menunggu, yang mana akan mungkin meningkatkan kepuasan mereka serta meningkatkan penjualan itu sendiri.

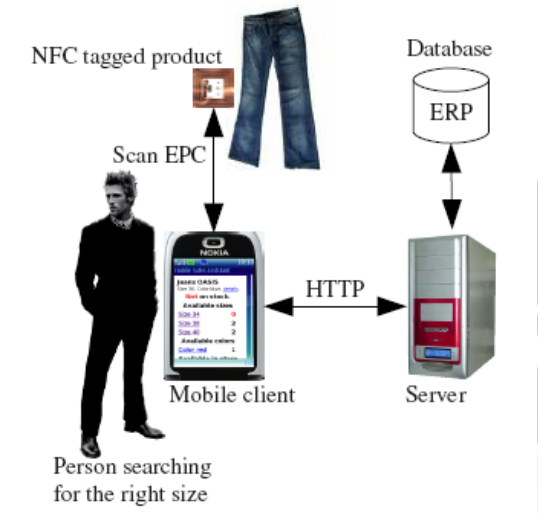
MSA adalah sebuah implementasi *prototipe* yang bertarget pada divisi pakaian pada departemen store.

Pertama kali *MSA* dicoba di Berlin pada Oktober 2007, yang mencoba berfokus pada 10 asisten toko.

MSA adalah sebuah *NFC* yang berbasis *mobile information system*, yang bertujuan untuk meningkatkan kualitas proses penjualan. *MSA* memungkinkan asisten toko untuk mengecek stok barang secara langsung dengan *Point of Sale (PoS)*. Jika asisten toko menyentuhkan barang yg ada label *NFC* nya dengan *NFC-enabled mobile phone*, maka akan ditampilkan informasi mengenai ketersediaan dari item yg diinginkan dan yang sesuai dengan produk tersebut, misalnya ukuran lainnya, atau warna lainnya yang diambil dari *ERP-system* pada *mobile phone*. Dengan informasi yg diberikan, karyawan dapat segera menginformasikan kepada pelanggan mengenai ketersediaan barang dan sebuah benda dapat ditemukan dengan lebih cepat. Sehingga akan menghemat waktu dan pelanggan tidak harus menunggu lama, yang akan meningkatkan kepuasan pelanggan. Jika pelanggan menggunakan aplikasi dengan *mobile phone* nya, mereka tidak perlu lagi membutuhkan bantuan sama sekali dan bisa secara mudah mengecek ketersediaan dari produk sendirian.
(Sumber:http://www.im.ethz.ch/publications/msa_print_20081214_final_sk.pdf)

Jika dilihat dari sisi pelanggan akan banyak sekali keuntungan yang akan didapatkan oleh pelanggan dalam menggunakan sistem informasi ini. Pelanggan tidak akan dipusingkan dalam mencari letak sebuah produk. Sehingga pelanggan dapat menghemat waktu pencarian. Selain itu dengan system informasi tersebut pelanggan juga akan

dapat memesan barang secara *online* melalui handphone mereka.



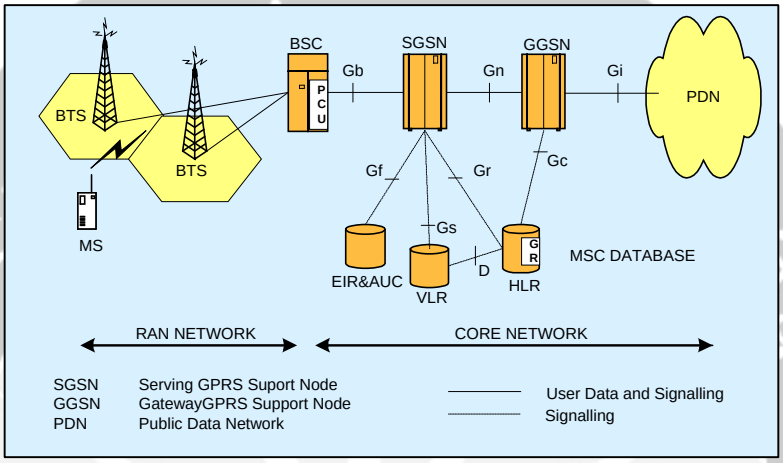
Gambar 2.1 Mobile Sales Assistant System Overview
(http://www.im.ethz.ch)

2.4. General Packet Radio Service (GPRS)

Teknologi transmisi data GSM berupa *General Packet Radio Service* atau *GPRS* adalah sebuah teknologi yang dipergunakan untuk pelayanan data *wireless* seperti pada *wireless* internet atau intranet serta pelayanan *multimedia*. Juga biasanya disebut sebagai *GSM-IP(Internet Protocol),* karena akan menghubungkan pengguna dengan *ISP(Internet Service Provider)*. Secara umum *General Packet Radio Service* atau *GPRS* adalah suatu teknologi yang memungkinkan pengiriman dan penerimaan data lebih cepat jika dibandingkan dengan penggunaan teknologi *Circuit Switch Data* atau *CSD.*

Secara teori kecepatan pengiriman data *GPRS* dapat mencapai 115kb/s. Namun dalam implementasinya sangat tergantung dari berbagai hal seperti konfigurasi dan alokasi *time slot* di level *Radio* atau *BTS*, teknologi *software* yang digunakan dan dukungan ponsel. Hal tersebut dapat menjelaskan mengapa pada saat-saat

tertentu dan di lokasi tertentu akses *GPRS* terasa lambat bahkan bisa lebih lambat dari akses *CSD* yang memiliki kecepatan 9,6kb/s.
(Sumber:http://www.stttelkom.ac.id/staf/UKU/Presentasi%20Publikasi%20UKE/Standard-GPRS-UKU.ppt)



Gambar 2.2 Arsitektur Jaringan GPRS dalam GSM (http://www.stttelkom.ac.id)

2.5. Java

Java menurut definisi dari Sun adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Java berdiri diatas sebuah mesin *interpreter* yang diberi nama *Java Virtual Machine(JVM)*. *JVM* inilah yang akan membaca *bytecode* dalam file *.class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu, bahasa *Java* disebut sebagai bahasa pemrograman yang *portable* karena dapat dijalankan pada berbagai sistem operasi, asalkan pada sistem operasi tersebut terdapat *JVM*. (Shalahudin, M. & A.S., Rosa. 2008)

2.5.1. Kelebihan Java

Terdapat beberapa kelebihan bahasa pemrograman *Java* yaitu:

1. Berorientasi Objek

   Java merupakan bahasa pemrograman berorientasi objek. Ini berarti bahwa dalam sistem berorientasi objek sebuah *class* merupakan *instance* dari sebuah *object* yang berisi data dan *methods* untuk menggambarkan status dan perilaku dari *object* tersebut. Dengan paradigma ini para programer *Java* hanya fokus pada data dan *methods* yang memanipulasi data tersebut dalam aplikasi.

2. Terinterpretasi dan Portabel

   Java adalah sebuah bahasa yang terinterpretasi (Flanagan,1997). *Java compiler* tidak men-*generate* kode mesin tetapi *byte-codes* dan untuk menjalankan sebuah program *Java* (*run-time*) maka dibutuhkan *Java interpreter,* yaitu *Java Virtual Machine* (JVM). Karena *Java byte-codes* adalah *platform* yang independen, maka program-program *Java* dapat dijalankan di bermacam-macam *platform* yang memiliki JVM dan dapat didistribusikan secara bebas. Hal inilah yang memicu jargon *Java* yaitu *"write once, run anywhere ™"*.

3. Dinamis dan Terdistribusi

   Java adalah bahasa pemrograman yang dinamis. Semua *class Java* dapat di-*load* menjadi *Java interpreter* setiap saat. *Java* juga disebut sebagai bahasa pemrograman yang terdistribusi. Ini berarti *Java* menyediakan dukungan yang sangat baik untuk pemrograman jaringan dan internet.

4. Sederhana dan Andal

Java adalah bahasa pemrograman yang mudah dipelajari karena memiliki jumlah konstruksi bahasa yang relatif kecil. *Java* juga didesain untuk pembangunan perangkat lunak yang andal. *Java* tidak mengeliminasi kebutuhan terhadap *Software Quality Assurance* (SQA).

5. Secure

Keamanan adalah salah satu aspek penting yang diusung bahasa *Java* karena sifat bahasa *Java* yang terdistribusi dan independen. Tanpa jaminan keamanan, setiap orang tidak akan men-*download* kode-kode *Java* dari sembarang situs di internet dan menjalankannya di komputer pribadi. Untuk itu *Java* menyediakan beberapa kontrol lapisan keamanan yang memberikan perlindungan terhadap kode-kode berbahaya.

6. High Performance

Sebagai bahasa yang memberikan dukungan terhadap pemrograman aplikasi berbasis GUI (*Graphical User Interface*) dan jaringan, *Java* memiliki performansi yang sangat baik.

7. Multithreaded

Java memberikan dukungan untuk eksekusi *multiple thread* yang dapat menangani *tasks* yang berbeda-beda. Dukungan terhadap *multithreading* ini membuktikan performansi yang interaktif dari *graphical application* untuk *user*. (Flanagan, 1997)

8. *Garbage Collection*

Garbage collection merupakan teknik manajemen *memory* yang digunakan program *Java* untuk menangani alokasi *memory* dinamis. Dengan teknik ini para *programmer* tidak akan mengalokasikan memory secara manual karena alokasi memory secara otomatis akan dilakukan oleh *Java Runtime Environment.*

2.5.2. Arsitektur Java

Arsitektur bahasa *Java* terdiri dari :

1. *Compiler*

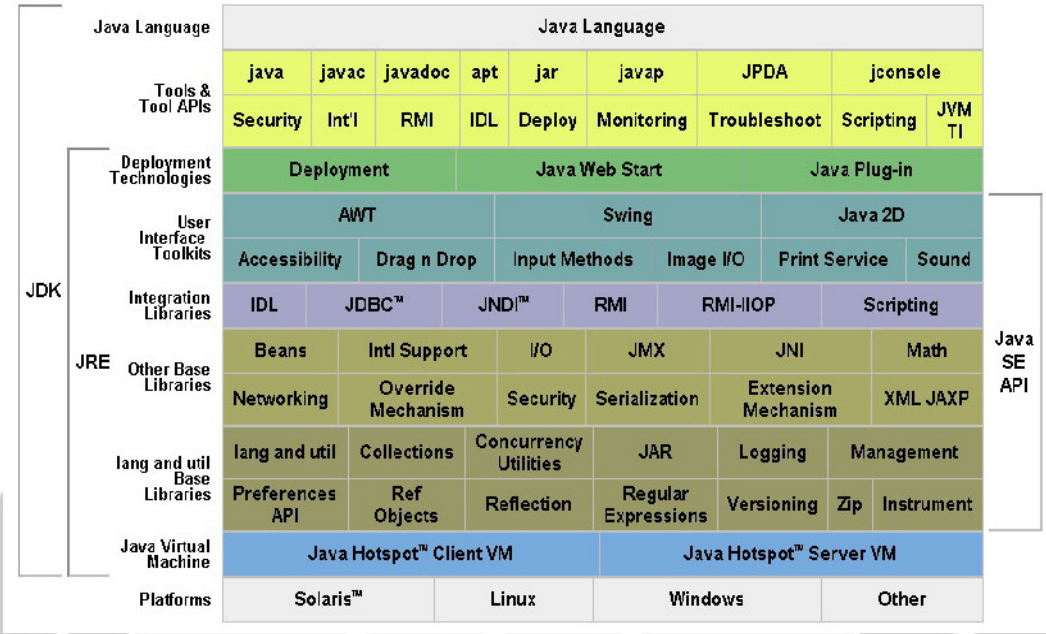
   Compiler Java (*javac*) mentranslasi kode-kode bahasa *Java* menjadi *byte-codes*.

2. *Java Virtual Machine* (JVM)

   Java Virtual Machine (JVM) adalah sebuah mesin komputer abstrak yang sering dinamakan sebagai *interpreter* karena tugasnya menginterpretasikan *Java byte-codes* yang terkompilasi.

3. *Application Programming Interface* (API)

   Java API menyediakan satu set *packages* yang didistribusikan bersama dengan *Java 2 Software Development Kit* (J2SDK) sebagai *class libraries*.



Gambar 2.3 Arsitektur *Java* (http://java.sun.com)

2.5.3. Edisi Java

Sun Microsystems telah mendefinisikan tiga buah edisi dari *Java 2* yang ditujukan untuk mesin dan sistem yang berbeda-beda, yaitu :

1. *Java 2 Standard Edition* (J2SE), yang digunakan untuk mengembangkan aplikasi-aplikasi *desktop* dan *applet* (aplikasi *Java* yang dapat dijalankan di dalam *browser web*).
2. *Java 2 Enterprise Edition* (J2EE), merupakan *superset* dari J2SE yang digunakan untuk mengembangkan aplikasi-aplikasi berskala *enterprise,* yaitu dengan melakukan pembuatan aplikasi-aplikasi di sisi *server* dengan menggunakan EJBs (*Enterprise JavaBeans*), aplikasi *web* dengan menggunakan *Java Servlet* dan JSP (*Java Server Pages*), dan teknologi lainnya seperti CORBA (*Common Object Request Broker Architecture*) dan XML (*Extensible Markup Language*).

3. *Java 2 Micro Edition* (J2ME), merupakan *subset* dari J2SE yang digunakan untuk menangani pemrograman di dalam perangkat-perangkat kecil, yang tidak memungkinkan untuk mendukung implementasi J2SE secara penuh.

   Secara umum pembagian edisi *Java* dapat digambarkan sebagai berikut :

Gambar 2.4 Pembagian Edisi *Java*
(http://developers.sun.com/mobility/getstart/articles/whyjava/)

2.6. Java 2 Micro Edition (J2ME)

J2ME adalah satu set spesifikasi dan teknologi yang fokus kepada perangkat konsumen. Perangkat ini memiliki jumlah memori yang terbatas, menghabiskan sedikit daya baterai, layar yang kecil dan *bandwith* jaringan yang rendah. Kelahiran *platform J2ME* timbul karena dibutuhkan adanya sebuah *platform* komputasi yang mengakomodasi piranti *consumer electronics* dan *embeded*. Piranti ini dikelompokkan menjadi dua kategori, yaitu:

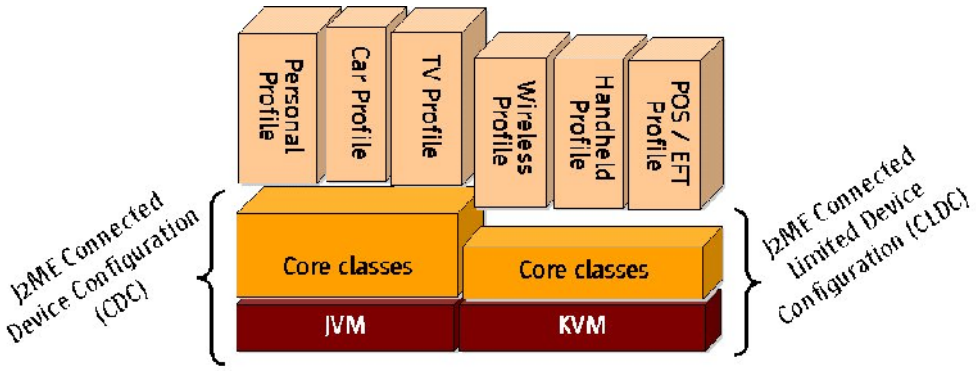
1. Personal, piranti *mobile* yang dapat digunakan untuk komunikasi melalui jaringan tertentu misalnya ponsel, *personal digital assistants(PDA)*, *Palm*, *Pocket PC* dan *organizer*.
2. Piranti informasi yang digunakan bersama dengan jaringan tetap(fixed), koneksi jaringan yang tidak putus-putus misalnya TV Internet, *high-end communicators*, dan sistem navigasi mobil.

2.6.1. Arsitektur J2ME

Banyaknya jenis dan tipe peranti mobile membuat sulit pencapaian standar kinerja dan *portabilitas*. Meskipun *J2ME* menerapkan konsep *run everywhere*, pengembang *J2ME* menspesifikasikan beberapa arsitektur yang terbagi atas: konfigurasi, profil, dan paket opsi (*optional package*). Tujuan dan spesifikasi *J2ME* itu demi mencapai kinerja dengan memanfaatkan kelebihan piranti sekaligus mencapai *portabilitas*.

Konfigurasi adalah *virtual machine* yang menyediakan beberapa pustaka kelas. Konfigurasi menyediakan fungsi dasar dengan karakteristik yang sama. Contohnya: fungsi koneksi jaringan dan manajemen *memory*.

Sementara itu, profil menyediakan lingkungan pustaka-pustaka *API* untuk membangun aplikasi *mobile*. Paket opsi dibuat untuk menyediakan fungsi-fungsi pada piranti *mobile* yang lebih spesifik. Contohnya: piranti yang memiliki akses *Bluetooth* memerlukan *API Bluetooth*.



Gambar 2.5 Arsitektur J2ME

(http://java.sun.com/products/personaljava/faq.html)

Dari Gambar kita bisa melihat bahwa *J2ME* memiliki dua konfigurasi *vitual machine*. Dua konfigurasi itu adalah:

1. *Connected Limited Device Configuration(CLDC): CLDC* bertujuan didesain untuk piranti *mobile* terkecil dengan 128 – 512 Kb memori, *processor* 16 – 32 bit. Profil dasar yang berjalan di atas *CLDC* adalah *MIDP*.
2. *Connected Device Configuration(CDC): CDC* adalah konfigurasi *high-end* yang membutuhkan memori *miNPMum* 2Mb dan processor 32bit. Profil dasar yang berjalan di atas *CDC* adalah *foundation Profile(FP)*.

2.6.2. J2ME Configuration

Configuration mendefinisikan lingkungan kerja J2ME *runtime*. Karena setiap *handheld devices* memiliki fitur yang berbeda-beda, maka dirancanglah J2ME *configuration*, yakni menyediakan *library* standar yang mengimplementasikan fitur standar dari sebuah *handheld devices*. Ada dua kategori J2ME *Configuration* saat ini, yaitu :

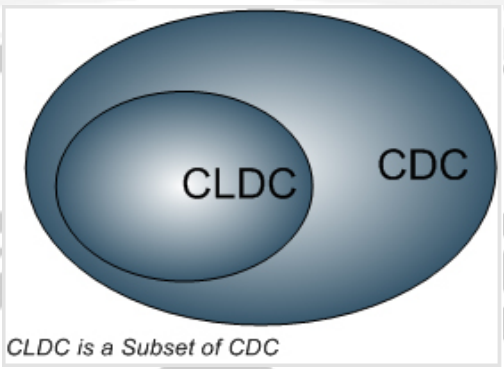
1. CLDC (*Connected Limited Devices Configuration*)

Kategori ini umumnya digunakan untuk aplikasi *Java* pada *handphone* seperti Nokia, Samsung Java Phone, Motorola i85s. PDA semacam PALM, *PocketPC,* dan *two way pagers.* Umumnya perangkat tersebut hanya memiliki ukuran memory 160-512 *KiloBytes.* Mesin *virtual* yang digunakan untuk CLDC adalah *Kilobyte Virtual Machine* (KVM) yaitu mesin virtual yang ditujukan untuk perangkat yang memiliki ukuran *memory* kecil.

2. CDC (*Connected Device Configuration*)

Kategori ini umumnya digunakan untuk aplikasi Java pada perangkat *handheld devices* dengan ukuran memory 2 *MegaBytes.* Contohnya adalah *Internet TV*, Nokia *Communicator* atau TV pada mobil.

Relasi antara CDC dan CLDC dapat digambarkan sebagai berikut:



Gambar 2.6 Relasi antara CDC dan CLDC

(http://developers.sun.com)

2.6.3. J2ME Profile

Profile adalah penyedia *libraries* dan API untuk para *developer* untuk mengembangkan aplikasi untuk masing-masing tipe *device.* Saat ini terdapat beberapa *profile* yang tersedia untuk kebutuhan-kebutuhan

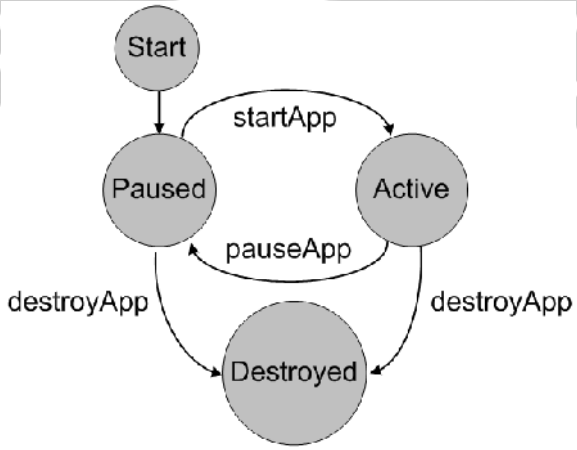
spesifik diantaranya *Mobile Information Device Profile* (MIDP), *Personal Digital Assistant Profile* (PDAP), *Foundation profile, Personal Profile, RMI Profile.*

2.6.4. *Mobile Information Device Profile* (MIDP)

MIDP khusus digunakan pada *handset* dengan kemampuan CPU, *memory*, *keyboard*, dan layar terbatas, misalnya pada *handphone*, *pager*, dan PDA. MIDP menyediakan cukup banyak *library* dan API *(Application Programming Interface*) untuk membantu pembangunan aplikasi.

2.6.4.1. *MIDlet*

MIDlet adalah aplikasi yang dibuat dengan menggunakan *Java 2 Micro Edition* dengan profile *Mobile Information Device Profile* (MIDP). *MIDlet* terdapat dalam *class javax.microedition.midlet.*. MIDlet* memiliki beberapa *state*, yaitu *Pause, Active,* dan *Destroy.* Ketika masing-masing *state* dipanggil, beberapa metoda yang bersesuaian dipanggil. State dalam *MIDlet* dapat digambarkan sebagai berikut :



Gambar 2.7 Siklus Hidup Sebuah *MIDlet*

(http://developers.sun.com/mobility/midp/articles/fsm/)

2.7. Penutup

Aplikasi yang akan dibangun adalah *Mobile Sales Assistant* yaitu *mobile* produk untuk pedagang yang berdasar dari kombinasi *Near Field Communication (NFC)* and *the Electronic Product Code (EPC)*. Tujuan pembuatan aplikasi ini adalah untuk mengoptimasikan proses penjualan untuk toko-toko *retail*. Pembangunan aplikasi tersebut menggunakan teknologi *J2ME* dan *GPRS*.