BAB II

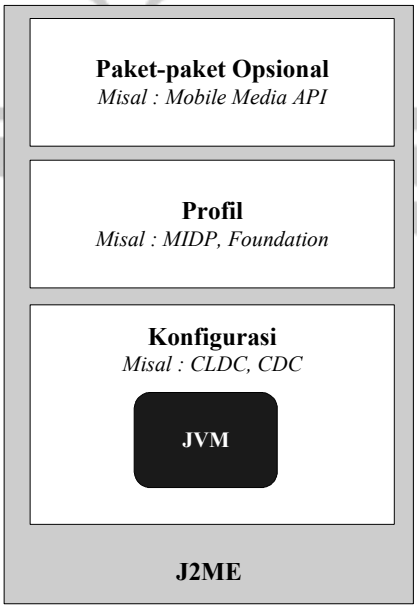
LANDASAN TEORI

II.1. Teknologi JAVA

II.1.1. J2ME

Java 2 Micro Edition (J2ME) merupakan sebuah kombinasi yang terbentuk antara sekumpulan interface Java yang sering disebut dengan JAVA API (*Application Programming Interface*) dengan JVM (Java *Virtual Machine*) yang didesain khusus untuk alat, yaitu JVM dengan ruang yang terbatas. Kombinasi tersebut kemudian digunakan untuk membangun aplikasi-aplikasi yang dapat berjalan di atas alat (dalam hal ini *mobile device*) (Raharjo, Hermantyo, Haryono, 2007).

J2ME pada dasarnya terdiri dari tiga buah bagian, yaitu: konfigurasi, profil, dan paket-paket opsional.



Gambar 2.1 Bagian-bagian di dalam *platform* J2ME

a. Konfigurasi

Konfigurasi merupakan bagian yang bersisi JVM dan beberapa *library* kelas lainnya. Terdapat dua buah konfigurasi yang disediakan oleh Sun Microsystem, yaitu CLDC (*Connected Limited Device Configuration*) dan CDC (*Connected Device Configuration*). Target alat dari konfigurasi CLDC adalah alat-alat kecil, seperti telepon selular, PDA, dan *pager*.

b. Profil

Profil merupakan bagian perluasan dari konfigurasi. Artinya, selain sekumpulan kelas yang terdapat pada konfigurasi, terdapat juga kelas-kelas spesifik yang didefinisikan lagi di dalam profil. Dengan kata lain, profil akan membantu secara fungsional yaitu dengan menyediakan kelas-kelas yang tidak terdapat di level konfigurasi. Profil yang populer penggunaanya disediakan oleh Sun Microsystem, yaitu yang dinamakan dengan MIDP (*Mobile Information Device Profile*).

c. Paket-paket Opsional

Paket-paket opsional merupakan paket-paket tambahan yang dibutuhkan oleh aplikasi sehingga pada saat proses *deployment* paket-paket tersebut perlu didistribusikan juga sebagai bagian dari aplikasi bersangkutan.

II.1.2. Aplikasi J2ME

Aplikasi yang dibuat di dalam *handphone* dengan menggunakan profil MIDP disebut dengan MIDlet. Sun MicroSystem telah menyediakan J2ME *Wireless Toolkit* (J2ME WTK) untuk mengembangkan aplikasi dalam *handphone*. J2ME WTK adalah sekumpulan *tool* yang digunakan untuk mengembagkan aplikasi-aplikasi dalam *handphone* dan *wireless device* lainnya.

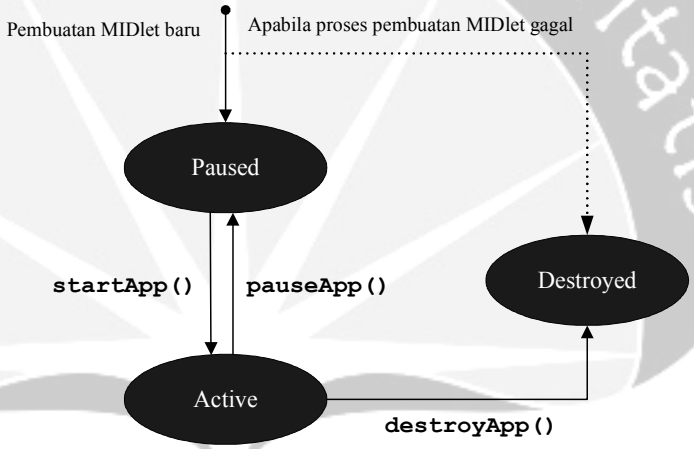
Developer dimungkinkan untuk membangun aplikasi *mobile* yang dapat terkoneksikan dengan *internet* (layanan GPRS) maupun dengan *mobile device file system*.

Dalam membangun suatu MIDlet, J2ME memiliki *packages* javax.microedtion.io, yaitu *class* untuk mendefinisikan koneksi-koneksi yang bersifat umum. *Class* ini terbagi atas tiga bagian, yaitu *Interfaces*, *Classes*, dan *Exception*. Jenis koneksi yang digunakan dalam tugas akhir ini adalah *HttpConnection* yang merupakan bagian dari *Interfaces* serta *FileConnection* yang merupakan bagian dari *Classes-Connector*. *FileConnection* berfungsi untuk mengkases *file system* ponsel pengguna dan *HttpConnection* untuk memanfaatkan teknologi jaringan ponsel.

II.1.2.1. Siklus Hidup Aplikasi J2ME

AMS (*Application Management Software*) atau JAM (*Java Application Manager*) merupakan lingkungan tempat sebuah MIDlet dapat di-*install*, dijalankan, dihentikan, maupun di-*uninstall*. AMS akan membuat setiap *instance* baru dari MIDlet yang dapat mengontrol keadaannya, yaitu dengan cara menjalankannya (*start*), mengistirahatkan (*pause*),

atau menghentikannya (*destroy*) secara langsung oleh dirinya sendiri. Terdapat tiga buah method yang harus diimplementasi oleh setiap MIDlet. Dengan kata lain, setiap MIDlet harus memiliki ketiga buah method tersebut, yaitu: *startApp()*, *pauseApp()*, dan *destroyApp()*. Setiap MIDlet dapat berada dalam salah satu keadaan (*state*) berikut: *Paused*, *Active*, atau *Destroyed* (Raharjo, Hermantyo, Haryono, 2007).



Gambar 2.2. Siklus hidup MIDlet

Tampak pada gambar di atas bahwa pada saat pembuatan MIDlet baru, mula-mula MIDlet akan berada dalam keadaan *Paused*. Apabila proses pembuatan MIDlet gagal atau mengakibatkan kesalahan (menimbulkan eksepsi), maka MIDlet akan langsung berada dalam keadaan *Destroyed*. Namun apabila proses pembuatan MIDlet berjalan dengan baik, maka setelah MIDlet dijalankan AMS secara otomatis akan mengeksekusi *method startApp()* dan hal ini akan mengubah MIDlet untuk berada dalam keadaan *Active*. MIDlet yang berada dalam keadaan *Active* dapat diubah kembali menjadi keadaan *Paused*

melalui pemanggilan method *pauseApp()* atau diubah menjadi keadaan *Destroyed* melalui pemanggilan method *destroyApp()*.

II.1.3. FileConnection

FileConnection API merupakan spesifikasi dari JSR 75 yang digunakan untuk mengkases *file system* ponsel, termasuk *memory sticks*. Pustaka ini memiliki fungsionalitas untuk membuat, menghapus direktori dan file, menampilkan direktori, menset *permissions*, mengambil informasi file, dan memberikan operasi I/O pada file (Sony Erricson, 2008).

II.1.3.1. *Permission*

Terdapat dua macam *permission* untuk mendefinisikan *FileConnection* API, yaitu:

javax.microedition.io.Connector.file.read

javax.microedition.io.Connector.file.write

Read permission digunakan pada saat file dibuka atau diakses dalam mode baca (*read mode*) atau ketika *InputStream* dibuka melalui objek *FileConnection*. Sedangkan *write permission* digunakan untuk membuka file dalam mode tulis (*write mode*) atau ketika *OutputStream* dibuka melalui objek *FileConnection*. *Permission* ini juga digunakan dalam operasi tulis lainnya, seperti *delete* atau *rename*.

Jika tidak menggunakan *permission* untuk mengakses API, maka *SecurityException* akan muncul dan hal ini penting untuk mengatur situasi semacam

ini dalam suatu MIDlet. Seperti pada operasi I/O (input/output) lainnya, sangat direkomendasikan untuk mengeksekusi operasi I/O *file system* di dalam suatu *thread* dari operasi untuk menghindari terjadinya *deadlock*.

II.1.3.2. Spesifikasi Ponsel

JSR 75, dalam hal ini *FileConnection API*, memiliki beberapa batasan dalam pengimplementasiannya pada ponsel. Salah satunya pada ponsel *Sony Ericsson*, menurut situs developer *Sony Ericsson*, ponsel ini memiliki batasan dalam mengimplementasikan JSR 75, yaitu :

a. Folder *Games* dan *Themes* tidak dapat diakses.
b. File dan Direktori bersifat *case sensitive*.
c. Panjang *file path* terbatas oleh karena pembawaan dari *file system*, yaitu 120 karakter.

II.1.3.3. *Class* dan *Interface* dalam *FileConnection API*

Berikut ini merupakan class dan interface yang penting digunakan dalam pengimplementasian *FileConnection API* :

a. *ConnectionClosedException* akan muncul ketika suatu method menggunakan *FileConnection* pada saat koneksi dalam kondisi tertutup.
b. *Class FileSystemRegitry* adalah sentral registrasi dan memiliki kemampuan untuk

menjabarkan *mounted roots* dengan menggunakan *method listRoots()*. *Class* ini juga menyediakan *method* untuk meregristasikan *listener* yang akan memberitahukan jika *file system* ditambahi atau dihapus selama sistem dijalankan.

c. *Interface FileSystemListener* digunakan untuk menerima status pemberitahuan ketika terjadi penambahan atau penghapusan *file system root*.

d. *Interface FileConnection* digunakan untuk mengakses direktori dan file dalam ponsel.

II.1.3.4. Penggunaan *FileConnection API*

Dalam menciptakan suatu *FileConnection* digunakan *method open* dari *Connector* di mana akan mengembalikan suatu koneksi. Penulisannya :

```
Connector.open(stringURL);
```

URL yang valid adalah :

a. *Internal memory* : "file://localhost/c:/" atau "file:///c:/".

b. *Memory Card* : "file://localhost/e:/" atau "file:///e:/".

Jadi, ketika mengakses direktori pada *internal memory*, dibentuklah suatu objek *FileConnection* dengan menggunakan URL sebagai berikut :

```
FileConnection fc
=(FileConnection)Connector.open("file:///c:/");
```

Kemudian *code* untuk menutup objek *FileConnection* adalah :

```
fc.close();
```

FileConnection yang mengacu pada file yang tidak eksis, tidak akan dapat menampilkan operasi spesifik pada file atau direktori manapun. Untuk mengecek file atau direktori tersebut ada atau tidak adalah :

```
fc.exsist();
```

Jika tidak ada, ciptakan file baru :

```
fc.create();
```

Atau menciptakan direktori baru :

```
fc.mkdir();
```

Untuk menghapus file atau direktori :

```
fc.delete();
```

Untuk menjabarkan *content* direktori yang diacu oleh *FileConnection*:

```
Enumeration e = fc.lists();
while(e.hasMoreElements()) {
    String rootName = (String)e.nextElement();
    System.out.println("mounted root:"+rootName);
}
```

II.1.3.5. File I/O

Untuk menulis ke file digunakan *OutputStream* dari objek *FileConnection* yang mengacu pada file yang bersangkutan :

```
DataOutputStream fc = fc. openDataOutputStream ();
os.write(new String("hello")).getBytes();
```

```
os.close();
```

Atau dapat menggunakan *DataOutputStream* untuk menuliskan tipe data primitif Java ke suatu file :

```
Integer i = 1234;
OutputStream ds = fc. openOutputStream ();
ds.writeInt(i);
ds.close();
```

Untuk membaca dari file digunakan *InputStream* atau *DataInputStream* dari objek *FileConnection* yang mengacu pada file yang bersangkutan :

```
byte[] b = new byte[1024];
InputStream is = fc.openInputStream();
is.read();
is.close();
```

II.1.3.6. Informasi File dan Direktori

Berikut ini adalah beberapa method dalam *class FileConnection* yang berfungsi untuk mendapatkan kembali informasi tentang spesifikasi direktori ataupun file :

a. `boolean canRead()` – Apakah file/direktori dapat dibaca?

b. `boolean canWrite()` – Apakah file/direktori dapat ditulisi?

c. `long directorySize(boolean includeSubDirs)` – Mengembalikan ukuran dari semua file dalam bentuk *bytes* dari suatu direktori. Jika

nilai dari paramater *includeSubDirs* adalah *true*, maka ukuran dari semua sub-direktori juga tercakup.

d. `long fileSize()` – Mengembalikan ukuran file dalam bentuk *bytes*.

e. `long lastModified()` – Mengembalikan tanggal terakhir file/direktori dimodifikasi.

II.1.4. HttpConnection

HttpConnection merupakan salah satu spesifikasi J2ME yang digunakan pada aplikasi untuk mengakses *internet* dengan menggunakan teknologi jaringan yang tersedia pada ponsel.

HTTP merupakan salah satu cara untuk melakukan koneksi antara ponsel dan server yang disediakan oleh J2ME, disamping koneksi lainnya seperti socket, SMS, dsb. Koneksi jenis ini menjadi salah satu pilihan yang sering digunakan karena area penggunaanya luas dan tidak sedikit ponsel yang mendukung koneksi HTTP ini.

Menurut J2ME *Foundation Spesification*, HTTP adalah sebuah protokol *request/response*, dimana parameter *request* harus diset terlebih dahulu sebelum *request* tersebut terkirim. Dalam metode ini, *client* mengirimkan *request* kepada *server* dengan alamat yang dispesifikasikan pada URL (*Uniform Resource Locator*), kemudian *server* akan memberikan *response* kepada *client*. Metode request itu sendiri ada tiga jenis, yaitu GET, POST, dan HEAD. Dengan metode GET data dikirimkan sebagai bagian dari URL. Sedangkan dengan metode POST

data dikirim pada *stream* terpisah. Kemudian pada metode HEAD data yang dikirim adalah *meta information*.

II.1.4.1. *Permission*

Berikut ini merupakan *permission* untuk mendefinisikan *HttpConnection*, yaitu:

```
javax.microedition.io.HttpConnection
```

II.1.4.2. Status Koneksi

Koneksi HTTP ini berada dalam satu dari dua status yang ada, yaitu :

a. *Setup*, dimana koneksi ke server gagal diciptakan.
b. *Connected*, dimana koneksi telah tercipta, *request* sudah terkirim, dan *response* sedang ditunggu.

Berikut ini merupakan *method* yang mungkin terlibat dalam status *Setup* :

a. `setRequestMethod`
b. `setRequestProperty`

Terjadinya transisi dari status *Setup* ke *Connected* disebabkan oleh beberapa method yang membutuhkan data untuk dikirim atau diterima dari server. *Method-method* yang dapat menyebabkan terjadinya transisi ke status *Connected* tersebut adalah :

a. `openInputStream`
b. `openOutputStream`
c. `getLength`
d. `getType`
e. `getEncoding`
f. `getHeaderField`
g. `getResponseCode`
h. `getResponseMessage`
i. `getHeaderFieldInt`
j. `getHeaderFieldDate`

Selain semua method di atas, masih banyak method HttpConnection lainnya yang dapat digunakan kapanpun dalam membangun aplikasi.

II.1.4.3. Penggunaan *HttpConnection*

Hal-hal utama yang perlu diperhatikan dalam mengimplementasikan *HttpConnection* adalah :

a. Memasukkan *permission HttpConnection* :

```
import javax.microedition.io.HttpConnection;
```

b. Membuat objek koneksi Http :

```
HttpConnection c = null;
```

c. Membuka URL dengan menggunakan method :

```
c = (HttpConnection)Connector.open(URL);
```

d. Menutup koneksi Http :

```
c.close();
```

II.1.5. J2ME Polish

Menuruts sumber J2MEPolish.org, J2ME *Polish* adalah seperangkat *tool* dan teknologi yang ditujukan kepada *mobile developer* ataupun instansi yang bergerak dalam teknologi *mobile*. Fitur-fitur utama dari J2ME *Polish* adalah :

a. *Lush* : UI *toolkit* yang sangat fleksibel dan yang dapat didesain di luar *source code* aplikasi.

b. *Janus* : perangkat untuk menghubungkan aplikasi *mobile* ke *handset* dan *paltform* yang berbeda-beda.

c. *Touch* : teknologi untuk mengakses *server* dan berkomunikasi secara *remote*.

d. *Trunk* : solusi untuk me-*load* dan menyimpan data yang kompleks hanya dengan satu baris *code*.

e. *Marjory* : komunitas untuk mengatur perangkat *database*.

II.1.5.1. Lush

Dari sekian fitur yang disediakan J2ME *Polish*, fitur yang akan digunakan pada Tugas Akhir ini adalah *Lush*. Fitur ini memungkinkan *developer* untuk mengkustomisasi aplikasi tanpa mengubah *source code* dari aplikasi itu sendiri. Desain dengan berbagai animasi dan efek untuk aplikasi yang dibangun ini dispesifikasikan ke dalam file CSS. Proses kustomisasi ini tidak hanya memisahkan desain dengan pengembangan *business logic* aplikasi, tetapi juga memungkinkan untuk mengkustomisasinya dengan sangat mudah. Melalui fitur *Lush* J2ME *Polish* ini, *developer* dapat mendesain *background* untuk *Form* atau *List* tanpa harus menuliskan code yang rumit. Serta dapat menggunakan format *font* yang tidak biasa/standar, menggunakan berbagai *font effect*, berbagai animasi

seperti *screen transition*, dan tampilan menu dengan menggunakan berbagai macam *view-type*.

Beberapa keuntungan penggunaan fitur *Lush* J2ME *Polish* ini diantaranya adalah :

a. Proses desain yang serupa dengan mendesain halaman *web*.
b. Desainer dan *developer* dapat mengubah desain kapanpun secara bebas.
c. Kustomisasi desain untuk *customer* atau bahkan *user* yang belainan.
d. Adaptasi desain ke resolusi layar yang berbeda-beda.
e. Pengembangannya yang sangat mudah, semenjak J2ME *Polish compatible* dengan standar MIDP.

Konsep dasar penggunaan GUI pada J2ME *Polish* adalah :

a. Mengembangkan GUI dengan menggunakan *class* seperi biasa, yaitu `javax.microedition.lcdui` seperti *Form*, *List* atau *TextField*.
b. Menggunakan penambahan *custom component* dari J2ME *Polish* seperti *TabbedForm*, *TreeItem*, *ChoiceTextField*, dan sebagainya.
c. Jika membutuhkan *control* lebih lanjut dari komponen `javax.microedition.lcdui` seperti menyeting mode input dari *TextField*, gunakan `de.enough.polish.UIAccess`.

d. Menambahkan direktif `#style` pada *source code* aplikasi untuk mengkoneksikan *code* tersebut dengan desain yang dibuat.

e. Merancang dan mengkustomisasi aplikasi dengan menggunakan *textfile* CSS yang sederhana.

f. Mengkonfigurasikan penambahan GUI pada *script* build.xml.

II.2. Mobile *Internet* Access

Mobile internet Access merupakan fitur teknologi jaringan pada ponsel untuk mengakses *internet*. Berikut ini adalah daftar generasi teknologi jaringan yang pada umumnya disediakan oleh ponsel (Wikipedia, 2008):

Tabel 2.1 Tabel Generasi Teknologi Jaringan

Teknologi	**Kecepatan (max)**
GSM (2G)	
▪ GPRS (2,5 G)	115 Kbps
▪ EDGE (EGPRS)	236 Kbps
W-CDMA (3G)	
▪ UMTS	384 Kbps
▪ HSDPA (3,5 G)	3,6 Mbps
4G	14,4 Mbps

Dengan tersedianya fitur tersebut, pengguna dapat mengakses *internet* untuk berbagai kebutuhan. Namun hal ini juga tergantung dari kapasitas kecepatan yang ditawarkan oleh teknologi itu sendiri. Semakin tinggi kecepatannya, maka semakin besar pula kemampuan ponsel dalam mengakses berbagai layanan *internet*.

II.3. SERVER SIDE PROGRAMMING

Server Side Programming (*Scripting*) merupakan sebuah teknologi *scripting* atau pemograman web dimana *script* (program) dikompilasi atau diterjemahkan di *server* dan dikirimkan ke *browser* dalam bentuk HTML. Pemrograman ini biasanya digunakan untuk menghasilkan halaman web yang dinamis dan interaktif, yang dapat berhubungan dengan *database* atau jenis penyimpanan data lainnya. Keuntungan utama dari penggunaan *server side programming* ini adalah kemampuannya dalam mengkustomisasi respon dari *server* yang didasari oleh kebutuhan dan hak akses user, atau pun *query-query* dalam *database* (Wikipedia, 2008).

Beberapa contoh *Server Side Programming* :

1. ASP (*Active Server Page*) dan ASP.NET
2. ColdFusion
3. JSP (*Java Server Pages*)
4. Perl
5. Phyton
6. PHP

II.3.1. Prinsip Kerja Web

Aplikasi web berjalan pada protokol HTTP, dan semua protokol di internet selalu melibatkan sisi server dan client. Ketika seseorang mengetikkan suatu alamat di *browser*, maka *browser* akan mengirimkan perintah tersebut ke *web server*. Jika yang diminta oleh *client* adalah file yang mengandung perintah *server side* maka *web server* akan menjalankan dahulu program tersebut lalu mengirimkannya kembali ke *browser* dalam bentuk HTML sehingga dapat diterjemahkan oleh *browser*. Sedangkan jika yang diminta oleh *client* adalah file yang mengandung *client side* maka oleh *server* file tersebut akan langsung dikirimkan oleh *browser* (Caturstudio weblog, 2008).

II.4. Mobile Software Development

Dalam lingkungan modern saat ini, teknologi mobile menjadi hal yang sangat penting untuk setiap orang. Menurut sumber mengenai *Mobile Application for Business*, Teknik Informatika UAJY, teknologi ini dapat memberikan kemudahan komunikasi maupun akses internet bagi pengguna di seluruh dunia. Selain itu konsumen dan pebisnis dapat melakukan transaksi dan pertukaran informasi melalui perangkat mobile atau wireless handset seperti ponsel, PDA, atau SmartPhone. Kekurangan dari teknologi ini adalah ukuran layar dan resolusinya yang terbatas, minimalnya pemasukan informasi, dan terdapat situs web yang tidak kompatibel.

Definisi dari wireless handset itu sendiri adalah perangkat yang dapat menerima dan mengirim suara

serta data menggunakan sistem nirkabel. Komponen utama yang mencirikan wireless handset adalah :

a. Antena : penguat / pengantara transmisi.
b. Number pad : dialing (memasukkan angka).
c. Speaker dan microphone : komunikasi suara.
d. Operating system : operasi / antarmuka dengan pengguna.
e. Computer chips : fungsi lainnya.

Pada tugas akhir ini, wireless handset yang digunakan dalam pengembangan aplikasi mobile adalah telepon selular atau ponsel. Ciri-ciri dari ponsel adalah :

a. Transmisi sinyal digital dan analog.
b. Adanya *phone books*.
c. *Caller* ID dan *voicemail*.
d. *Organic-electro-luminescent color display screen*.
e. *Voice-activated technology* : menjalankan operasi melalui perintah suara.
f. *Internet access*.

Di dalam wireless handset, *developer* dapat membangun berbagai macam aplikasi mobile dengan menggunakan teknologi mobile yang semakin berkembang. Dalam pengembangan aplikasi mobile tersebut, terdapat beberapa tool yang dapat digunakan yaitu Java Platform, .Net, Symbian, dsb. Pada tugas akhir ini tool yang digunakan adalah Java Platform (J2ME), yang ditujukan pada perangkat dengan RAM (*Random Access Memory*)

sekitar 128 KB dan prosesor dengan kemampuan lebih rendah dari komputer desktop. Berdasarkan arsitektur J2ME (secara khusus MIDP), J2ME ini digunakan pada perangkat mobile dengan karakteristik sebagai berikut :

a. Minimum layar 96 x 54 pixels.
b. 1-bit display *depth*.
c. Memiliki input *device*, seperti keyboard/keypad, atau *touch screen*.
d. Minimum 128 KB nonvolatile memory untuk komponen MIDP.
e. Memiliki dua arah konektivitas wireless.