BAB V

IMPLEMENTASI

5.1 Batasan Implementasi

Dalam penelitian ini, pengembangan aplikasi tidak mungkin dilakukan pada perangkat yang sesungguhnya, dalam artian bahwa pembuatan aplikasi jelas tidak dapat dilakukan pada perangkat *wireless* yang nantinya bakal menjadi lingkungan *runtime* yang sesungguhnya. Pengembangan aplikasi hanya dapat dilakukan di perangkat keras komputer, dan proses ujicoba tahap awal dilakukan menggunakan emulator yang tersedia di perangkat komputer.

Secara garis besar, tahapan implementasi dilakukan pada dua lingkungan kerja (*platform*), yaitu lingkungan pengembangan dan lingkungan target (*runtime*).

5.1.1 Lingkungan Pengembangan

Untuk dapat melakukan pengembangan perangkat lunak dengan nyaman dan memadai, spesifikasi perangkat keras komputer juga harus diperhatikan. Berikut spesifikasi yang direkomendasikan:

a. Processor berbasis Intel atau AMD dengan kecepatan 1 GHz atau lebih.
b. Kapasitas memori (RAM) 256 MB atau lebih.
c. Alokasi ruang harddisk 1 GB atau lebih.
d. Modem dengan kecepatan minimum 56 Kbps atau perangkat Ethernet card (LAN card).

Perangkat modem digunakan untuk melakukan akses ke Internet. Apabila memungkinkan akses ke Internet melalui LAN (*Local Area Network*) maka dapat juga menggunakan Ethernet card. Akses ke Internet dibutuhkan ketika proses ujicoba aplikasi untuk menggunakan jaringan IRC di Internet dilakukan pada perangkat lunak emulator.

Selama pengembangan juga diperlukan sebuah aplikasi server IRC untuk melakukan simulasi koneksi dan untuk mempelajari protokol IRC secara lebih mendalam, tanpa harus selalu terhubung ke Internet. Hal ini ditujukan untuk menghemat biaya penelitian dan pengembangan perangkat lunak.

Lingkungan pengembangan dilakukan dengan spesifikasi sebagai berikut:

a. Perangkat notebook dengan spesifikasi: Intel Pentium M Dothan 1,6 GHz, Memori 256 MB L2 Cache 2 MB.
b. Sistem operasi Microsoft Windows XP.
c. Perangkat lunak IRCPlus (untuk simulasi server IRC).

5.1.2 Lingkungan Target

Perangkat *wireless* yang merupakan *target device*, harus memenuhi spesifikasi minimum perangkat keras sebagai berikut:

a. Modem yang dapat digunakan untuk akses Internet via GPRS.
b. Kartu seluler yang telah aktif layanan GPRS-nya.
c. Layar *monochorome* (satu warna), tetapi lebih baik lagi jika berwarna.

Secara umum, penggunaan istilah perangkat *wireless* ditujukan pada perangkat-perangkat sepert telepon seluler, PDA (*Personal Digital Assistant*),

PDA phone, dan *smart-phone*. Namun, dalam penelitian ini fokus pembuatan aplikasi adalah untuk target telepon seluler dan *smart-phone* yang telah mendukung teknologi Java. Hal ini dilakukan untuk menonjolkan sisi penggunaan teknologi Java pada perangkat yang memang biasanya minim fitur, bukan perangkat yang memang biasanya telah dilengkapi dengan banyak aplikasi (misalnya PDA dan communicator).

Perangkat *wireless* yang merupakan *target device*, harus memenuhi spesifikasi minimum perangkat lunak sebagai berikut:

a. Memiliki J2ME Runtime Environment.
b. J2ME Configuration yang digunakan adalah CLDC (*Connected Limited Device Configuration*) versi 1.0.
c. J2ME Profile yang digunakan adalah MIDP (*Mobile Information Device Profile*) versi 1.0.
a. Lingkungan *runtime* pada penelitian ini mengacu pada perangkat telepon seluler Nokia N-Gage (sebagai *target device*) yang memenuhi spesifikasi minimum tersebut di atas.

5.1.3 Bahasa Pemrograman

Bahasa yang digunakan adalah bahasa Java, kemudian untuk melakukan kompilasi program menggunakan J2SE (*Java 2 Standard Edition*) Development Kit versi 1.5. Untuk mempermudah proses *editing*, *compiling*, dan *running* secara terintegrasi dalam tahap pengembangan aplikasi, digunakan NetBeans IDE 4.0 dengan tambahan plugin *Mobility Pack* yang berisi J2ME API dan Emulator J2ME Wireless Toolkit 2.2.

5.1.4 Batasan Fitur pada J2ME IRC Client

Berikut beberapa fitur mendasar yang tidak diimplementasikan secara penuh pada aplikasi J2ME IRC Client:

a. Tidak menyediakan server ident (*ident daemon*) yang berguna untuk mengakses server-server IRC di Internet dengan kebijakan keamanan yang ketat seperti DALnet (irc.dal.net).
b. Tidak menyediakan antarmuka dan menu khusus untuk pengaksesan layanan NickServ dan ChanServ (misalnya untuk melakukan identifikasi nickname).
c. Tidak menyediakan antarmuka dan menu khusus untuk melakukan pengaturan channel (misalnya untuk menyetel topik atau mengubah modus channel).

Meskipun demikian, fitur-fitur di atas (kecuali IDENT server) tetap dapat diakses secara manual menggunakan menu Expert Mode.

5.1.5 Fokus Pemrosesan secara *Concurrent*

Oleh karena *target device* yang digunakan tidak mendukung komunikasi *duplex*, maka fokus implementasi lebih tertuju pada model penggunaan *socket* secara *concurrent*.

Perlu diketahui bahwa cukup banyak produk Nokia yang tidak mendukung komunikasi *duplex* pada teknologi J2ME-nya, yaitu antara lain 7650 dan N-Gage. Listing program yang ditampilkan pada bagian implementasi akan menggunakan asumsi dan batasan untuk pemakaian *socket* secara *concurrent* saja.

5.2 Implementasi

5.2.1 Implementasi Umum Aplikasi

Proses implementasi secara umum melalui tahapan-tahapan sebagai berikut:

1. Melakukan pemeriksaan pada seluruh perangkat keras yang digunakan untuk pembuatan aplikasi, agar tidak muncul asumsi-asumsi salah yang diakibatkan oleh kerusakan sebagian atau seluruh perangkat keras yang digunakan.
2. Melakukan konfigurasi perangkat lunak baik pada perangkat komputer maupun perangkat telepon seluler, yaitu antara lain: instalasi dan konfigurasi server IRC lokal di komputer, konfigurasi *access point* pada GPRS untuk perangkat telepon seluler.
3. Melakukan pengembangan aplikasi pada perangkat komputer dan menguji kinerja aplikasi tersebut menggunakan emulator dan server IRC lokal.
4. Melakukan *deployment* dengan cara meng-*upload* aplikasi ke perangkat telepon seluler, meng-*install*-nya, kemudian menguji kinerja aplikasi tersebut menggunakan koneksi GPRS untuk mengakses server IRC di Internet.

5.2.2 Implementasi Kode Program

5.2.2.1 Pemrograman Socket

5.2.2.1.1 Pemakaian Socket secara *Parallel*

Bahasan ini merupakan bahasan terakhir mengenai pemrosesan *parallel* karena sebagaimana telah disebutkan pada batasan implementasi, metode ini tidak akan dibahas lagi ketika bahasan sampai ke penjelasan struktur aplikasi yang sebagian besar berisi listing program beserta penjelasannya.

Meskipun demikian tidak ada salahnya apabila metode tersebut sedikit dijelaskan dalam sebuah prototipe kode program. Telah dijelaskan sebelumnya bahwa apabila piranti mendukung komunikasi *duplex*, maka *thread* untuk menangani *InputStream* dan *OutputStream* dapat dieksekusi secara terpisah.

Berikut contoh implementasi *thread* untuk *InputStream*:

```
private void startInputStreamThread() {
  Thread t = new Thread() {
    public void run() {
      while (true) {
        try {
          // baca satu karakter
          char ch = in.read();
          // sleep for moments...
          Thread.sleep(1000);
        }
        catch (InterruptedException ie) {}
      }
    }
  };
  t.start();
}
```

Kemudian untuk penggunaan *OutputStream* dapat diletakkan di *thread* manapun di luar *thread* tersebut.

5.2.2.1.2 Pemakaian Socket secara *Concurrent*

Berikut beberapa permasalahan yang banyak terjadi pada piranti Nokia [BEA03a]:

1. Pemanggilan ke *method* Connector.open("socket://...") pada *thread* utama akan memacetkan aplikasi tanpa pesan kesalahan sedikitpun.
2. Apabila *InputStream* sedang digunakan, misalnya pada pemanggilan *method* `read()`, maka *InputStream* berada dalam keadaan terblok untuk proses lain (*blocked*), kemudian hal tersebut mengakibatkan *OutputStream* terblok juga hingga *method* `read()` benar-benar selesai dieksekusi.
3. Pemanggilan *method* `InputStream.available()` selalu mengembalikan nilai 0, tidak peduli berapa bytes yang seharusnya dapat dibaca.

Karakteristik yang sangat aneh tersebut memunculkan sebuah dilema, yaitu adanya keharusan untuk mengetahui secara pasti jumlah byte yang masuk melalui *InputStream*. *InputStream* akan tetap 'setia' menunggu apabila ketika *method* `read()` dipanggil saat itu tidak ada data yang dapat dibaca di *stream*.

Hal ini sebenarnya cukup konyol, karena program bukanlah seorang paranormal yang mengetahui masa depan (maksudnya kapan ada data masuk dan berapa byte jumlah pastinya). Selain itu satu-satunya harapan penyelesaian, yaitu menggunakan *method* `InputStream.available()` telah terbukti tidak dapat digunakan.

Untuk mengatasi masalah yang cukup aneh tersebut, paling tidak terdapat 3 (tiga) solusi darurat (yang cukup aneh dan menyulitkan), yaitu:

1. Menggunakan dua *StreamConnection* agar masing-masing dapat dihubungkan dengan *InputStream* dan *OutputStream* untuk dua keperluan yang berbeda. Tetapi solusi ini mengharuskan adanya sebuah server proxy (perantara) agar dua StreamConnection tersebut dapat 'menyatu' sebagai satu kesatuan.
2. Mencari cara untuk 'memancing' *InputStream* agar melepaskan *stream* agar dapat segera digunakan oleh *OutputStream*. Hal ini dilakukan dengan cara memasang sebuah Bot IRC yang selalu terhubung ke jaringan IRC yang sama dengan jaringan IRC yang dihubungi oleh J2ME IRC Client. Bot tersebut harus mengirim paling tidak satu byte data secara periodik pada tiap selang waktu tertentu.
3. Mencari cara untuk menandai bagian paling ujung pada *InputStream*, yaitu dengan mengenali suatu karakter khusus yang menandai akan segera habisnya data pada *InputStream* sehingga pembacaan karakter menggunakan *method* `read()` dapat segera dihentikan ketika menemukan karakter khusus tersebut tanpa harus mengalami *blocking*.

Dari beberapa solusi yang ada tersebut masih perlu dianalisis secara cermat untuk memilih solusi yang paling banyak memberikan keuntungan dan kerugian seminimal mungkin.

Strategi yang pertama merupakan cara yang sangat kompleks dan sulit, memiliki kelemahan pada sisi portabilitas aplikasi dan sisi portabilitas jaringan. Meskipun demikian, cara itulah yang digunakan oleh seorang programmer Java bernama Russel Beattie [BEA03a] untuk menyelesaikan permasalahan pada *stream*.

Strategi yang kedua memiliki kerugian pada sisi portabilitasnya, yaitu aplikasi J2ME IRC Client hanya dapat digunakan pada jaringan IRC tertentu saja, yang telah memiliki Bot IRC yang selalu terhubung ke jaringan tersebut. Hal ini tentu sangat menyulitkan dari sisi *uptime* yang sangat tinggi. Satu keuntungan dari strategi ini yaitu pada sisi portabilitas aplikasi dapat menggunakan kode program yang sama dengan kode program untuk piranti yang mendukung komunikasi *duplex*.

Strategi yang ketiga memiliki keunggulan yang paling optimal baik dalam hal portabilitas aplikasi maupun jaringan karena sama sekali tidak memerlukan adanya Bot IRC. Salah satu kerugian adalah borosnya transfer data karena 'pekerjaan konyol' yang terpaksa dilakukan, yaitu mengirim data ke diri sendiri sekaligus menerimanya kembali. Hal inilah yang cukup merugikan pada koneksi GPRS karena secara umum menggunakan metode perhitungan tarif per KiloBytes data. Secara logis, semakin banyak transfer data maka semakin tinggi pula biaya GPRS yang dikeluarkan. Dengan beberapa pertimbangan tersebut di atas maka cara yang ketiga inilah yang digunakan pada penelitian ini.

Berikut contoh implementasi *thread* untuk *InputStream*:

```
public void startInputStreamThread() {
   Thread listen = new Thread() {
      public void run() {
         while (true) {
            int dt = 0;
            // aktifkan magic char
            if (sConnection.magicCharActive == false) {
               // tulis magic char ke socket
               sConnection.writeln("PRIVMSG "
               + nickname + " :" + sConnection.magicChar);
               sConnection.magicCharActive = true;
               // tunggu agak lama, biar data di buffer lumayan banyak
               dt = 15000;
            }
            else {
               dt = 1000;
            }
            // run garbage collector
            System.gc();
            // baca data dari socket, pahami protokolnya
            decodeIrcProtocol(sConnection.readln());
            // sleep
            try {
               Thread.sleep(dt);
            }
            catch (InterruptedException ie) {}
         }
      }
   };
   listen.start();
}
```

Ketika karakter khusus yang menandai akan segera habisnya data pada *stream* ditemukan, maka segera dilakukan penandaan ulang.

Kemudian untuk penggunaan *OutputStream* dapat diletakkan di *thread* manapun di luar *thread* tersebut dan penulisan *stream* dapat dilakukan dengan waktu tunda sesuai dengan selang waktu yang telah ditentukan.

5.2.2.2 Protokol IRC

Untuk dapat melakukan komunikasi dengan baik di jaringan IRC, ada beberapa *message* dalam protokol IRC yang harus digunakan. Beberapa message yang diimplementasikan dalam aplikasi J2ME IRC Client dapat dilihat pada tabel 5.1.

Tabel 5.1 Beberapa message utama pada J2ME IRC Client

Syntax	**Deskripsi**
USER <username> <localname> <servername> :<realname>	Menyiapkan data-data untuk registrasi user
NICK <nickname>	Mengubah/menentukan nickname
JOIN <channel>	Bergabung (masuk) ke channel
PART <channel>	Keluar dari channel
PRIVMSG <channel> :<data>	Mengirim pesan ke channel
PRIVMSG <nickname> :<data>	Mengirim pesan ke user
QUIT :<quitmessage>	Keluar dari jaringan (memutuskan hubungan dengan server)

5.2.2.3 Struktur Aplikasi

Pada tahap implementasi, kelas-kelas yang telah dirancang sebelumnya merupakan struktur utama pembentuk aplikasi J2ME IRC Client yang dibagi menjadi beberapa kelas dalam 7 (tujuh) file terpisah, yaitu:

a. IrcApplication.java
b. IrcChannelUser.java
c. IrcConnection.java
d. IrcFunction.java
e. IrcMain.java
f. IrcMessage.java
g. IrcSocket.java

Kelas-kelas tersebut di atas masing-masing merupakan implementasi dari komponen-komponen pembentuk aplikasi yang sebelumnya telah dirancang untuk melakukan berbagai hal yaitu antara lain untuk menangani koneksi TCP/IP, untuk menangani protokol IRC, untuk menangani tampilan terutama pada komunikasi channel dan user, untuk mengatur koordinasi antar kelas-kelas, serta untuk menyediakan fungsi-fungsi tambahan yang diperlukan dalam aplikasi.

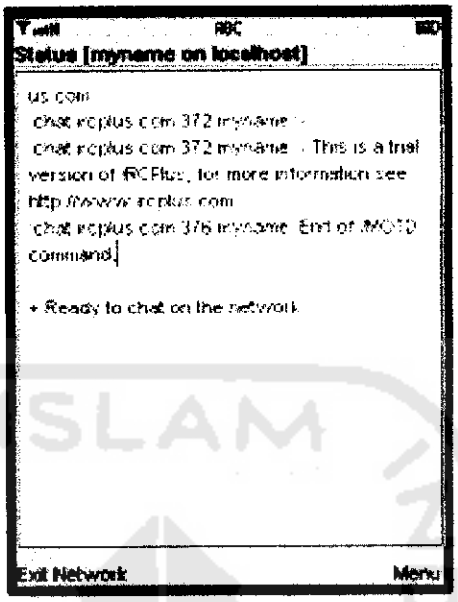
5.2.3 Implementasi Antarmuka

5.2.3.1 Tampilan Aplikasi pada Emulator J2ME Wireless Toolkit 2.2

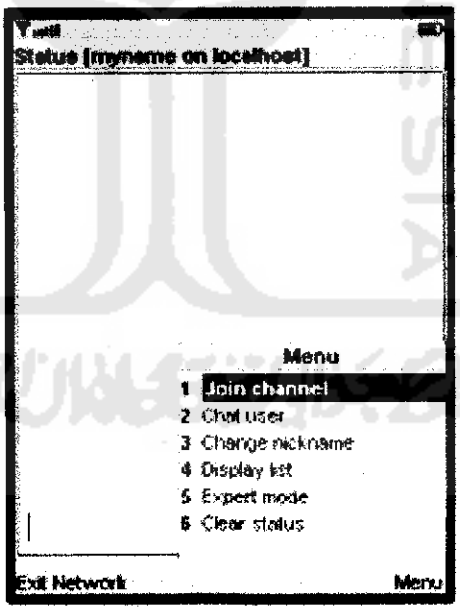
Tampilan pada emulator dapat berbeda-beda tergantung jenis dan versi emulator yang digunakan. Dalam tahap implementasi, emulator yang digunakan merupakan *plug-in* yang terintegrasi pada NetBeans IDE.

Emulator sebenarnya mendukung aplikasi yang menggunakan pewarnaan. Namun berhubung J2ME IRC Client memang tidak mengimplementasikan penggunaan warna, maka kemampuan pewarnaan pada emulator tidak terlihat. Implementasi dari rancangan antamuka aplikasi dapat dilihat pada hasil tangkapan layar (*capture*) pada gambar-gambar berikut.

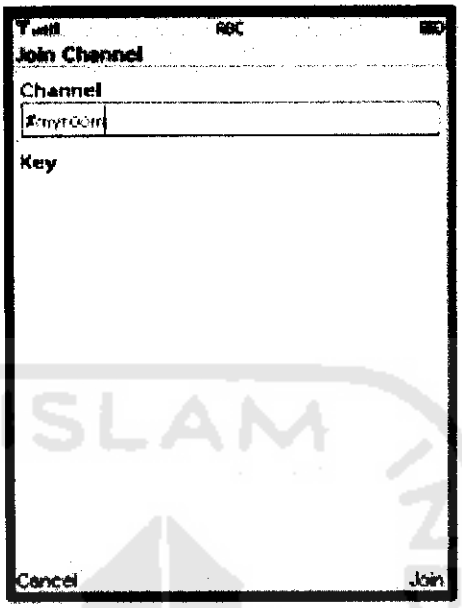
Gambar 5.1 Antarmuka untuk melakukan koneksi ke server



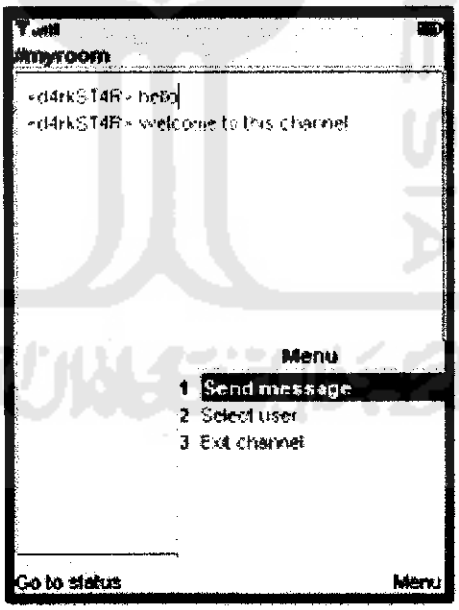
Gambar 5.2 Antarmuka untuk mengetahui status user



Gambar 5.3 Antarmuka untuk menu utama



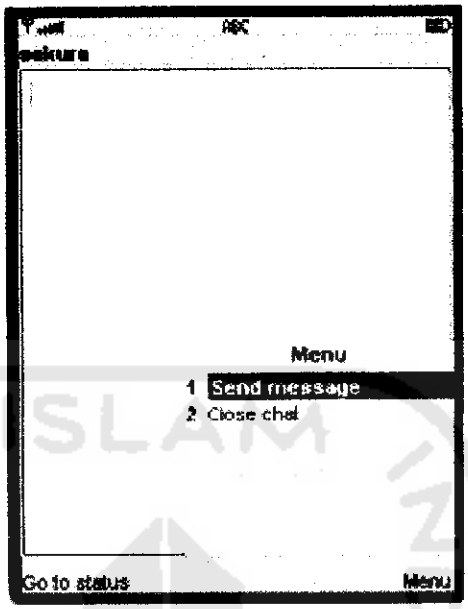
Gambar 5.4 Antarmuka untuk melakukan join ke channel



Gambar 5.5 Antarmuka untuk menu channel

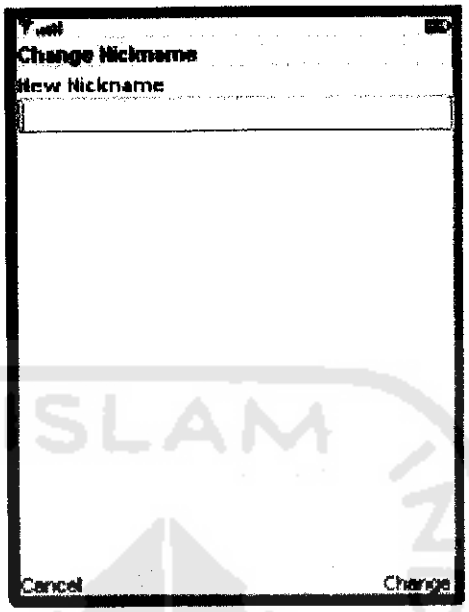
Gambar 5.6 Antarmuka untuk mengirim pesan ke channel

Gambar 5.7 Antarmuka untuk menampilkan user-user di channel



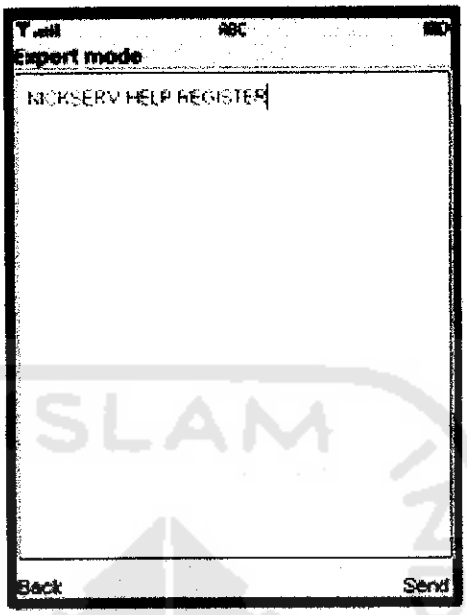
Gambar 5.8 Antarmuka untuk menu user

Gambar 5.9 Antarmuka untuk mengirim pesan ke user



Gambar 5.10 Antarmuka untuk mengganti nickname

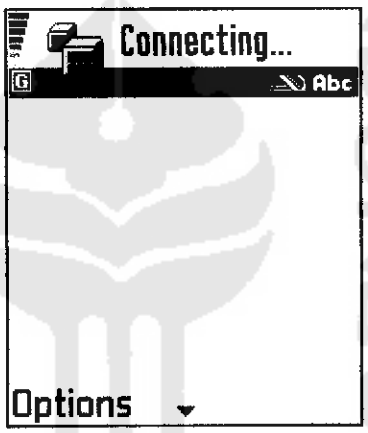
Gambar 5.11 Antarmuka untuk menampilkan daftar display



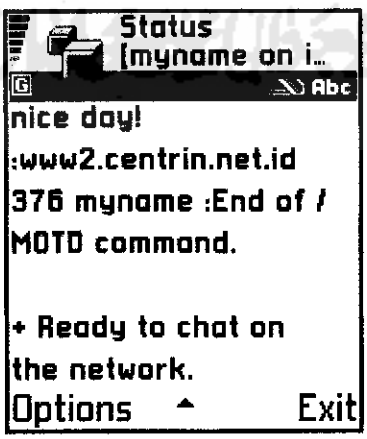
Gambar 5.12 Antarmuka untuk display Expert Mode

5.2.3.2 Tampilan Aplikasi pada Nokia N-Gage

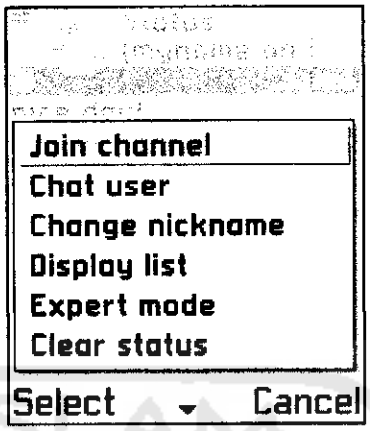
Tampilan pada *target device*, yaitu Nokia N-Gage, sesungguhnya tidak jauh berbeda dengan tampilan pada emulator. Bagian menu letaknya berkebalikan antara kiri dan kanannya. Beberapa perbedaan yang muncul disebabkan oleh *Java Runtime* pada Nokia N-Gage yang berbasis pada rancangan GUI milik *native* OS-nya, yaitu Symbian Series 60. Lebih jelasnya dapat dilihat pada hasil tangkapan layar (*capture*) pada gambar-gambar berikut.



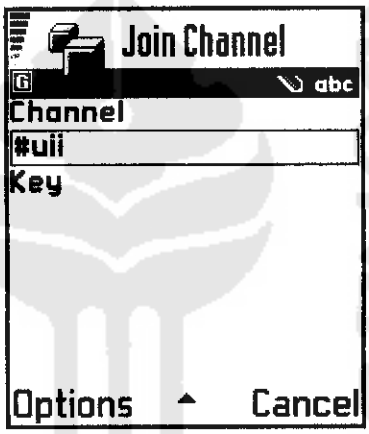
Gambar 5.13 Antarmuka untuk melakukan koneksi ke server



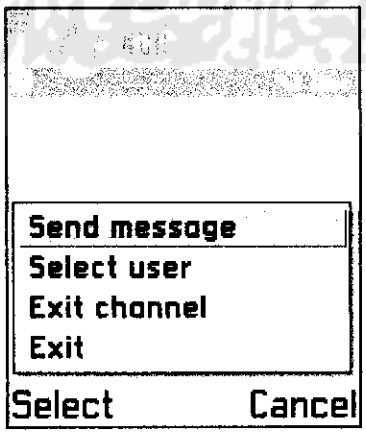
Gambar 5.14 Antarmuka untuk mengetahui status user



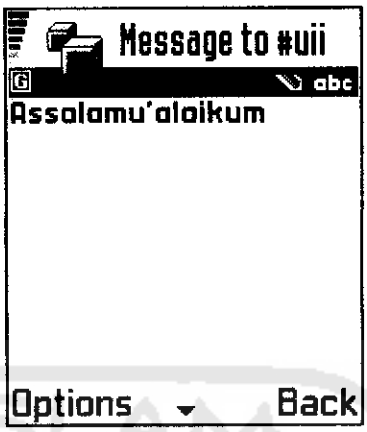
Gambar 5.15 Antarmuka untuk menu utama



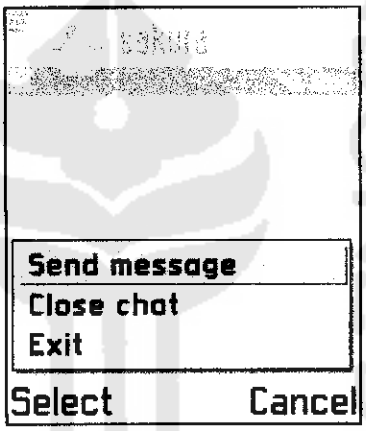
Gambar 5.16 Antarmuka untuk melakukan join ke channel



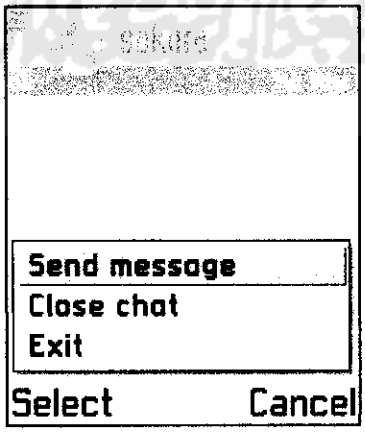
Gambar 5.17 Antarmuka untuk menu channel



Gambar 5.18 Antarmuka untuk mengirim pesan ke channel

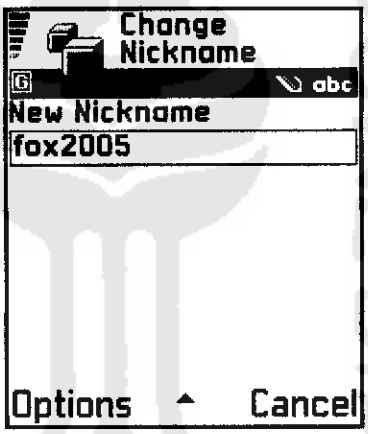


Gambar 5.19 Antarmuka untuk menampilkan user-user di channel

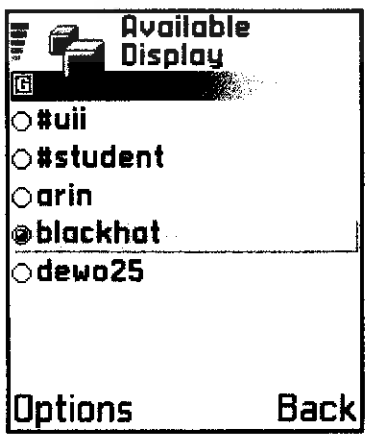


Gambar 5.20 Antarmuka untuk menu user

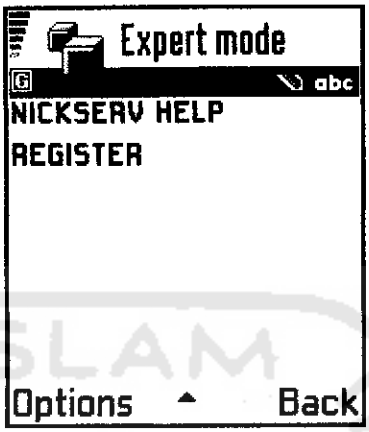
Gambar 5.21 Antarmuka untuk mengirim pesan ke user



Gambar 5.22 Antarmuka untuk untuk mengganti nickname



Gambar 5.23 Antarmuka untuk menampilkan daftar display



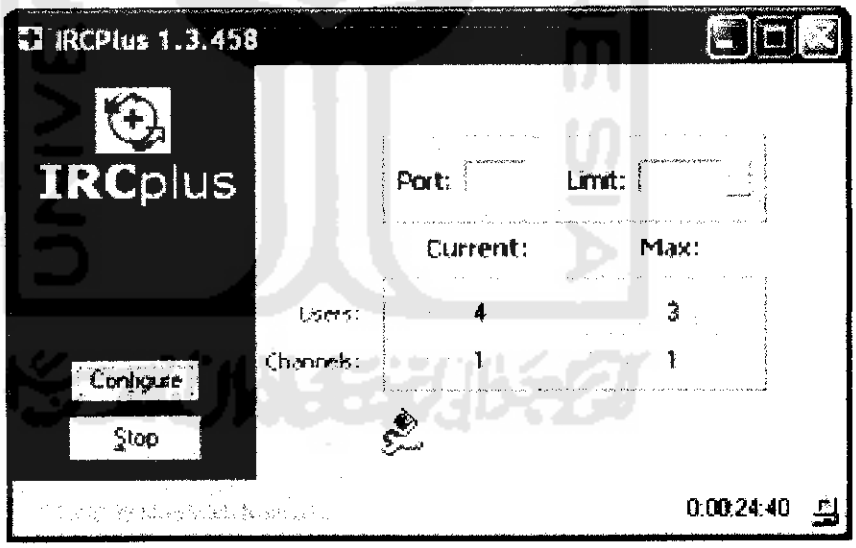
Gambar 5.24 Antarmuka untuk display Expert Mode

5.2.4 Implementasi Tambahan

5.2.4.1 Penggunaan Server IRC pada Komputer Lokal

Untuk dapat melakukan pengujian koneksi ke server IRC tanpa harus terhubung ke Internet, diperlukan sebuah server IRC di dalam komputer lokal. Pada sistem operasi Windows dapat digunakan perangkat lunak IRCPlus untuk melakukan pengujian koneksi dan mengakses beberapa fasilitas standar sebuah server IRC.

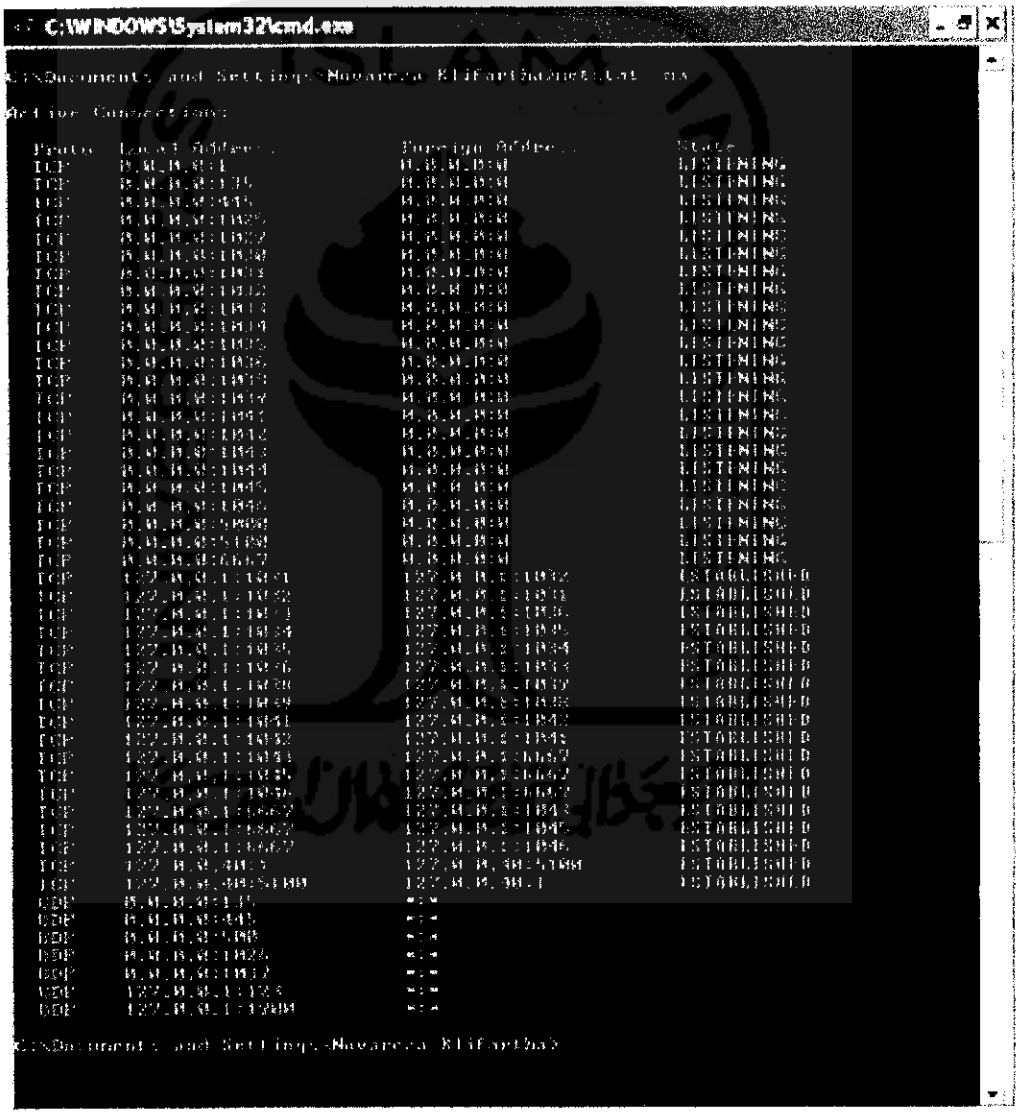
IRCPlus akan menampilkan jumlah user dan jumlah channel yang terhubung ke server, fasilitas panel utama, serta beberapa fasilitas penyetelan konfigurasi. Lebih jelasnya dapat dilihat pada gambar 5.25.



Gambar 5.25 Panel utama IRCPlus

Untuk memastikan bahwa server IRC benar-benar aktif atau memonitor koneksi yang terjadi, dapat menggunakan aplikasi Netstat dengan menyertakan opsi parameter "-na" (tanpa tanda kutip) pada perintah command prompt. Opsi

tersebut digunakan untuk menampikan alamat dan nomor port dalam bentuk numerik (angka) untuk seluruh koneksi dan seluruh port yang terbuka. Dalam hal ini, IRCPlus telah dikonfigurasi untuk menggunakan port 6667 (standar IRC) sehingga seharusnya port tersebut sudah terbuka dan siap melayani permintaan menggunakan protokol IRC. Lebih jelasnya dapat dilihat pada gambar 5.26.

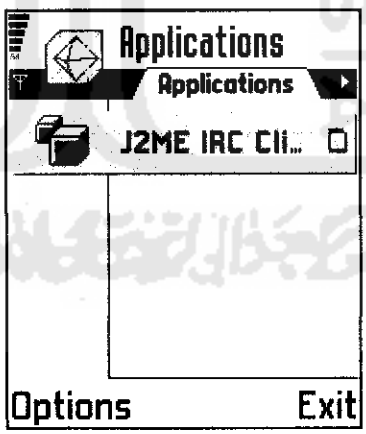


Gambar 5.26 Pengecekan layanan IRC

5.2.4.2 Proses *Deployment* Aplikasi

Pada lingkungan pengembangan, aplikasi dapat secara otomatis terinstall ke dalam emulator J2ME dan juga eksekusi dilakukan secara otomatis dalam NetBeans IDE.

Untuk melakukan *deployment* pada perangkat Nokia N-Gage, dapat dilakukan dengan mengkopi file-file berekstensi **.jad* dan **.jar* ke dalam MMC (*Multi Media Card*) menggunakan kabel data USB (*Universal Serial Bus*). Setelah file-file tersebut masuk ke dalam MMC, instalasi dilakukan dengan menjalankan file yang berekstensi **.jar*. Kemudian setelah aplikasi terinstall dapat dijalankan melalui menu Extras - Apps. Lebih jelasnya dapat dilihat pada gambar 5.27.



Gambar 5.27 Hasil instalasi J2ME IRC Client di Nokia N-Gage