AGENT UNTUK PEMANTAU KEAMANAN SERVER PADA JARINGAN INTERNET MENGGUNAKAN MOBILE DEVICE

Bambang Sugiantoro
Jurusan Teknik Informatika UPN "Veteran" Yogyakarta
Jl. Babarsari no 2 Tambakbayan 55281 Yogyakarta Telp (0274)-485323
e-mail : edo_lapis@yahoo.com

Abstrak
Aplikasi penelitian agent remote akan memudahkan kerja admin untuk melakukan pengawasan terhadap server yang dikelolanya. Admin akan melakukan remote dengan cara login telnet melalui mobile device. Kemudian akan muncul tampilan terminal di layar mobile device. Mobile device yang digunakan adalah mobile device yang mendukung Mobile Information Device Profile (MIDP) dan juga mendukung layanan GPRS. Server yang digunakan adalah server yang mempunyai layanan telnet. Pendekatan yang digunakan dalam penelitian ini adalah dengan menggunakan konsep Object Oriented Programming (OOP).

***Keyword** : Agent remote, mobile device , OOP*

1. PENDAHULUAN

Dunia teknologi khususnya komputer juga berkembang secara pesat dalam beberapa tahun terakhir ini, baik secara perangkat keras (*hardware*) maupun perangkat lunak (*software*). Perkembangan dunia teknologi tidak lepas juga dari permintaan dan tuntutan manusia akan teknologi. Sehingga terciptanya teknologi yang ditujukan untuk membantu kehidupan manusia. Manusia sekarang dituntut untuk selalu berkomunikasi dan selalu bergerak cepat (*mobile*). Sesuai dengan tuntutan maka terciptalah telepon seluler atau yang lebih sering disebut *mobile device* sebagai salah satu bentuk teknologi yang membantu manusia yang selalu bergerak (*mobile*).
Pada perkembangannya, *mobile device* tidak hanya digunakan sebagai media berkomunikasi saja, tetapi *mobile device* juga mendukung untuk fasilitas *game*, mendengarkan musik, foto, merekam *video*, bahkan *mobile device* juga dapat melakukan koneksi ke *internet* yang tentunya dapat digunakan untuk melihat *website* tertentu, melihat dan mengirim *email*.

Untuk dapat melakukan koneksi ke *internet*, *mobile device* harus mendukung fasilitas GPRS dan menggunakan *simcard* yang mendukung fasilitas GPRS. *Mobile device* akan membutuhkan informasi tentang IP (*internet protocol*) dan *port* dari *server*, nama *user* dan *password* untuk *login* ke *server*.

Tugas seorang *administrator server* atau lebih sering disebut *admin* adalah mengelola *server*. Peran sebuah *server* adalah sangat vital dalam sebuah jaringan. Seorang *admin server* dituntut untuk memiliki perhatian penuh terhadap *server* yang dikelolanya. Salah satu cara yang digunakan oleh *admin* dalam mengelola *server* adalah dengan melakukan pemantauan server yang dikelola berbasis software agent yang ditanam pada mobile device .

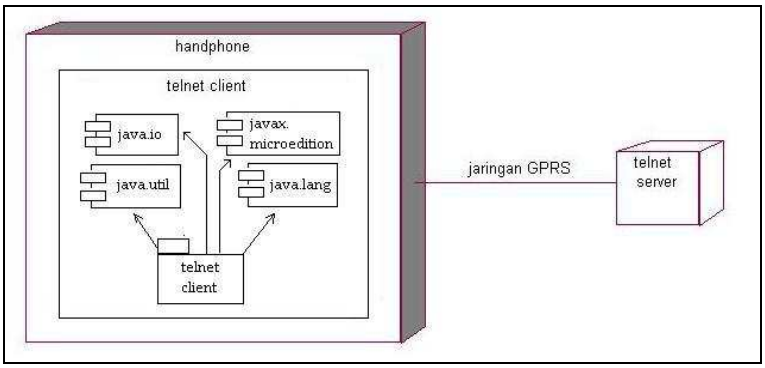
Dalam penelitian ini dibuat suatu agent yang ditanam *mobile device* dengan menggunakan bahasa pemrograman *java* yang mempunyai fasilitas untuk melakukan *remote* terhadap *server* dengan memanfaatkan *service telnet*. Dalam penilitian ini, *server* yang digunakan adalah *server* dengan menggunakan *linux* sebagai sistem operasi. Dengan memanfaatkan teknologi GPRS, *mobile device* akan melakukan koneksi ke *server*. Untuk menggunakan aplikasi ini, *admin* cukup memasukkan *username* dan *password* root. Apabila *username* dan *password* valid, maka akan muncul tampilan terminal (*shell*) di layar *mobile device*. *Mobile device* yang digunakan adalah *mobile device* yang mendukung MIDP.

2. TINJAUAN PUSTAKA

Penelitian yang sudah ada saat ini *admin* dapat mengendalikan *server* tanpa harus berada di depan komputer (*server*) yaitu dengan cara melakukan *remote*. Aplikasi *remote server* yang sering digunakan *admin* adalah *putty* (www.putty.nl). Dengan *putty, admin* dapat mengelola *server*. Akan tetapi, *putty* dijalankan di dalam komputer yang terhubung ke jaringan *internet*. Akan menjadi masalah ketika *admin* tidak memiliki atau tidak berada di depan komputer yang terhubung ke jaringan *internet*. Sebagai contoh, ketika *admin* berada dalam perjalanan. Pada saat itu admin sangat membutuhkan sebuah aplikasi *remote* yang dapat dibawa kemanapun *admin* pergi (*mobile*). *Mobile device* adalah sebuah alat komunikasi yang dapat dibawa kemana-mana. Selain sebagai alat komunikasi, *mobile device* juga dapat digunakan untuk menjalankan aplikasi-aplikasi seperti *game*, *calculator*, *converter*, *calendar*. Berdasarkan masalah itu maka akan dibangun sebuah aplikasi *mobile device* yang dapat digunakan oleh *admin* untuk mengelola *server* tanpa harus berada di depan *server*. Aplikasi ini akan dapat dibawa oleh *admin* ke mana-mana (*mobile*) karena ter*install* pada *mobile device*.

3. METODE PENELITIAN

Perancangan aplikasi digunakan untuk menentukan arsitektur dari aplikasi yang akan dibangun sehingga memiliki konstruksi yang baik, proses pengolahan data yang tepat dan akurat, memiliki nilai, memiliki aspek *user friendly* dan memberikan dasar-dasar untuk pengembangan selanjutnya.

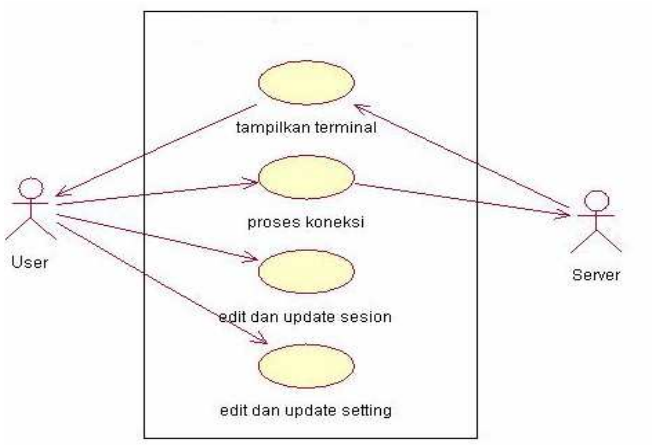


Gambar 1. Skema arsitektur sistem agent pemantau *(Deployment Diagram)*

Dari gambar di atas dapat dilihat bahwa aplikasi yang dibuat akan berjalan di dalam sistem operasi *mobile device*. Data akan dikirim ke GPRS *network* melalui *connection protocol*. Kemudian *network* akan mencari alamat dari *server* dan menyampaikan data ke *server*, lalu *server* akan mengirim data ke *admin*.

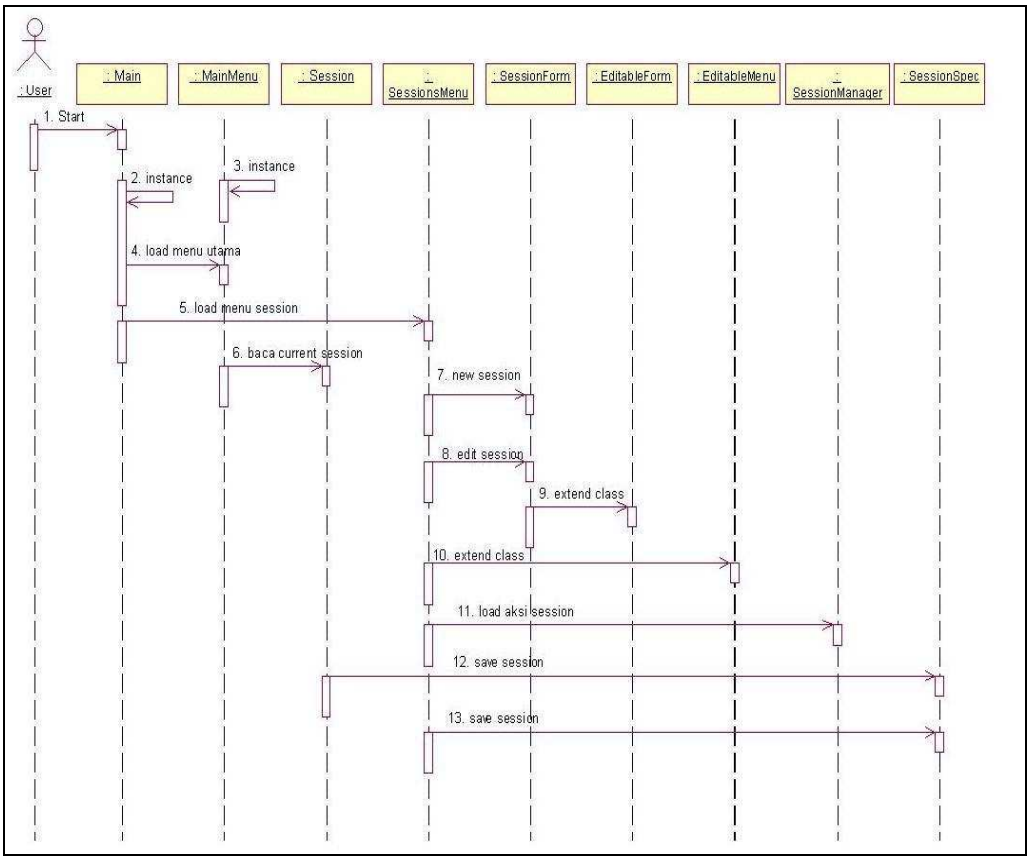
Developments and Interactions merupakan bagian pertama dari tahap ketiga metode OOP yaitu tahap *Design*. Pada tahap ini diagram *use case* dan paket yang telah dibuat pada tahap *Domain Analysis* akan dikembangkan dan ditentukan bentuk interaksi dan realisasinya pada *logical view*.

Dari analisis kebutuhan sistem dapat dibuat diagram *use case* yang sama dengan *high level use case diagram*. Berikut merupakan gambar diagram *use case* :



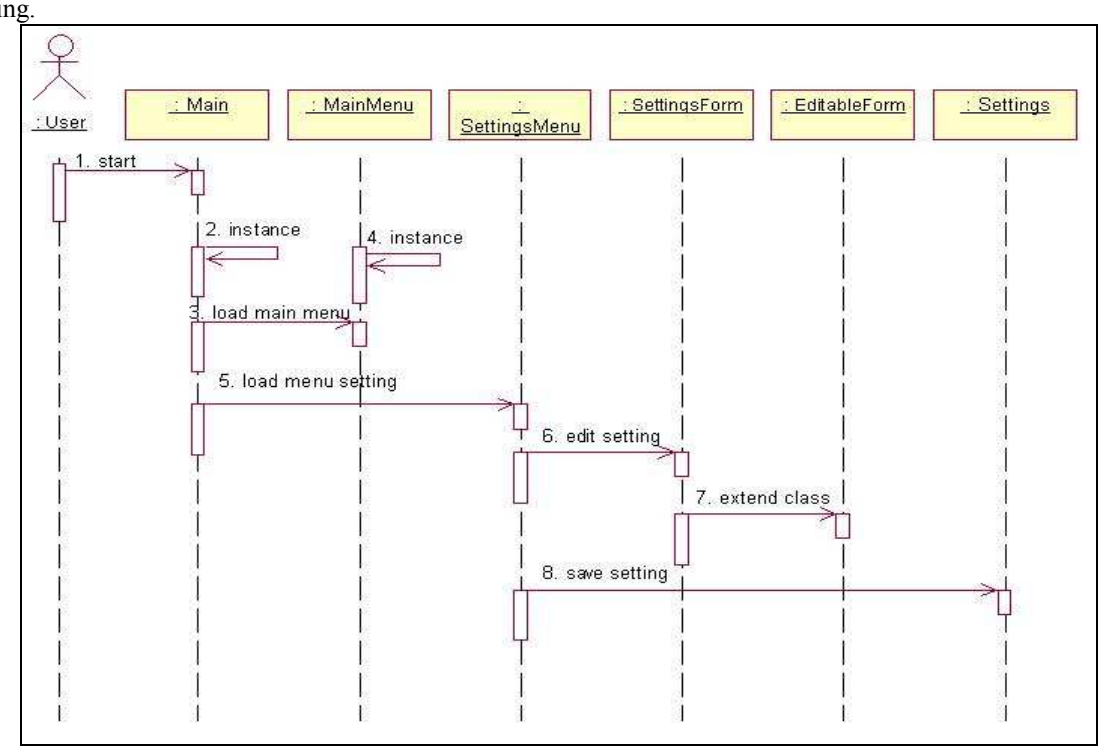
Gambar 2. Diagram *Use Case* Agent pemantau Keamanan Server di internet

Sistem akan melihat terlebih dahulu memanggil tampilan menu utama. Kemudian akan melihat konfigurasi *session* yang akan digunakan, termasuk *session* yang sudah didefinisikan terlebih dahulu (*current session*). Dari *class* SessionMenu, dapat melakukan *edit*, *add*. Ketika akan melaukukan *add* dan *edit* maka sistem akan memanggil tampilan menu *session* dari *class* SessionForm. Kemudian penyimpanan konfigurasi *session* dilakukan pada *class* SessionSpec. Spesifikasi dalam yang dibutuhkan dalam *session* berupa *alias* untk memberi nama *session*, *host* berisi informasi alamat *server*, dan *port* berisi nomor *port* yang digunakan protokol *telnet*.



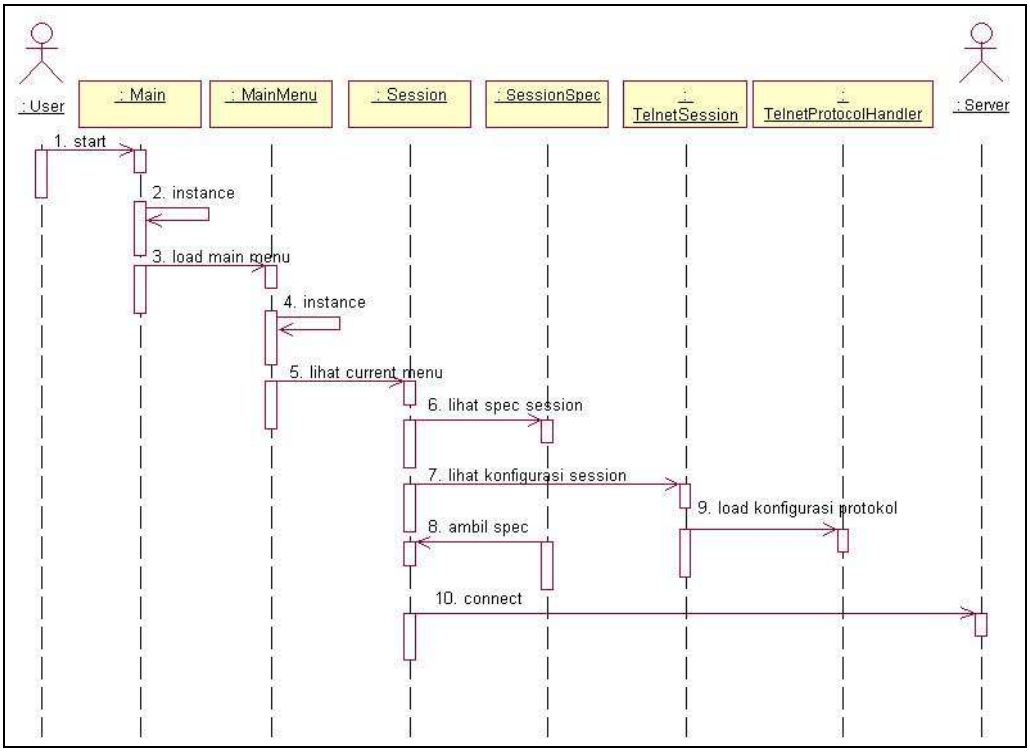
Gambar 3. Realisasi *use case edit* dan *update Session* (*Sequence Diagram*)

Pada tahap selanjutnya , sistem akan memanggil tampilan menu utama terlebih dahulu. Kemudian sistem dapat melakukan *edit* pada *setting*. Tetapi sistem tidak dapat melakukan penghapusan (*delete*) pada *setting*. Konfigurasi *setting* ini nanti akan digunakan untuk tampilan *terminal*, antara lain berupa *font* yang digunakan, ukuran *terminal*, dan warna *background* serta warna *foreground*. Konfigurasi *setting* disimpan pada *class* Setting.



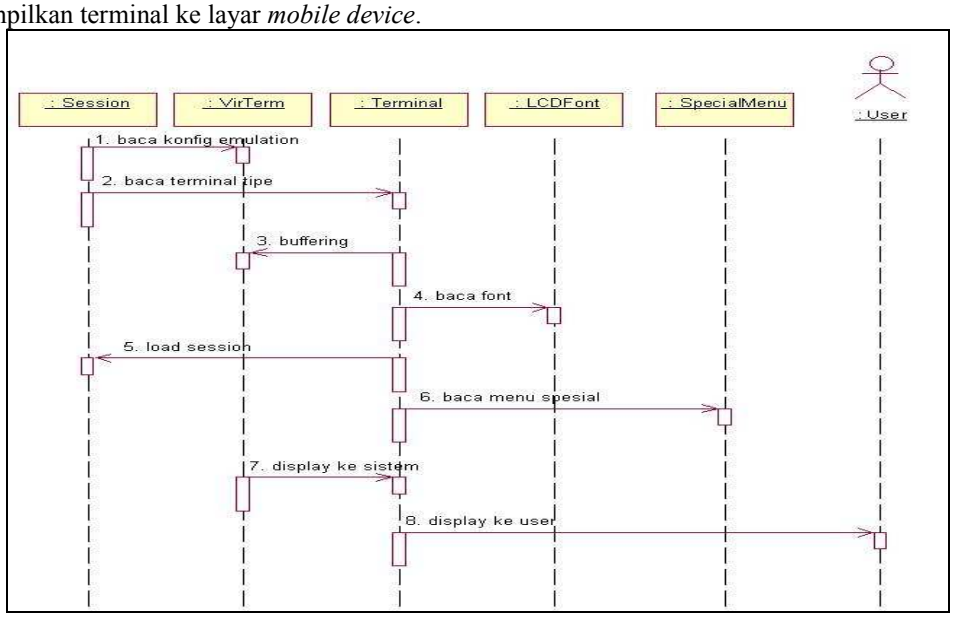
Gambar 4 Realisasi *use case edit* dan *update setting*

Proses selanjutnya merupakan proses yang utama dalam aplikasi ini. Koneksi akan dilakukan berdasarkan informasi yang didapat dari *session*. *Session* yang di *load* akan membaca konfigurasinya pada *class* SessionSpec. Koneksi akan dilakukan oleh *class* Session. Dengan membaca konfigurasi dari *class* TelnetSession dan *class* TelnetProtocolHandler.



Gambar 5. Realisasi *use case* proses koneksi

Proses kemudian akan melakukan konfigurasi *terminal* yang akan ditampilkan ke layar *mobile device*. Dari *class* terminal akan melihat konfigurasi pada *setting*. Kemudian akan melakukan konfigurasi pada *class* VirTerm sebagai *virtual terminal*. Kemudian melihat konfigurasi *font* yang tersedia. Melihat *special menu* yang tersedia. Lalu menampilkan terminal ke layar *mobile device*.



Gambar 6. Realisasi *use case* tampilkan terminal (*Sequence Diagram*)

4. HASIL DAN PEMBAHASAN

Dihasilkan *package* yang terdapat satu *sub package* dan enam buah *class*, yaitu *sub package* session, *class* Main, *class* settings, *class* SessionSpec, *class* SessionManager, *class* TelnetRecordStore, dan *class* LineInputStream. Secara umum *package* ini berfungsi untuk menjalakan seluruh proses agent ini .

Class Session merupakan *class* yang berfungsi membuat koneksi dengan *server*. *Class* ini mengimport *class* Activatable dan *class* MainMenu dari *package* gui, *class* Terminal dan *class* VirTerm dari *package* terminal, serta *class* SessionSpec dan *class* Settings dari *package* app.

```
private boolean connect() throws IOException {
   String host = spec.host;   String port = spec.port;
   emulation.putString( "Connecting to " + host + ":"+ port +"..." );
   socket=(StreamConnection)Connector.open("socket://"+host+":"+port,
Connector.READ_WRITE, false );
   in = socket.openDataInputStream();
   out = socket.openDataOutputStream();
   emulation.putString( "OK\r\n" );
   return true; }
```

Modul Program 1. *Method* untuk melakukan koneksi

Class ini mengurusi tentang *input* data dan *output* data yang terjadi selama dalam proses *remote*.

```
public interface SessionIOHandler
{
   public void handleSendData( byte[] data, int offset, int length )
throws IOException;
   public void handleReceiveData( byte[] data, int offset, int length
) throws IOException;
}
```

Modul Program 2. *Method* untuk *input* dan *output* data

Dalam *package* ini terdapat dua *class*, yaitu *class* Dimension dan *class* TelnetProtocolHandler. Secara umum, dalam *package* ini mengkonfigurasikan *protokol telnet* yang digunakan agar bisa terkoneksi ke *server*. *Class-class* dalam *package* ini akan dijelaskan dalam sub bab-sub bab berikut.
Class Dimension berfungsi untuk melihat ukuran *window*, yaitu berupa *width* dan *height*.

```
public class Dimension {
   public int width, height;
   public Dimension() { }
   public Dimension(int width, int height) {
      this.width = width;
      this.height = height;
   }
}
```

Modul Program 3. *Method class* Dimension

Untuk dapat melakukan koneksi melalui protokol *telnet*, maka konfigurasi protokol *telnet* juga harus dibuat dalam *class* tersendiri.

```
private void handle_sb( byte type, byte[] sbdata, int sbcount )
throws IOException {
      switch ( type )   {
            case TELOPT_TTYPE:
                  if ( sbcount > 0 && sbdata[0] == TELQUAL_SEND ) {
                        write( new byte[] {
                                    IAC, SB, TELOPT_TTYPE, TELQUAL_IS
                        } );
                        String ttype = getTerminalType();
                        write( ttype.getBytes() );
                        write( IACSE );
                  }
      }
}
```

Modul Program 4. Salah satu *method* konfigurasi protokol pemantuan agent

5. KESIMPULAN

Telah berhasil dirancang pemantauan server di jaringan internet berbasis agent yang dapat mempermudah kerja admin dalam mengelola dan mengawasi server.Aplikasi ini dapat melakukan remote ke server melalui dengan memanfaatkan jaringan GPRS / CDMA dan dapat dikembangkan dengan menambahkan protokol ssh sebagai protokol remote yang memiliki fitur keamanan untuk melindungi pertukaran data yang terjadi selama proses remote.

6. DAFTAR PUSTAKA

Doyle, Morgan, 2003, *GPRS Tutorial*, http://www.morgandoyle.co.uk

Dreamtech, 2001, Wireless Programming With J2ME <http://as.wiley.com

Hartanto, Antonius A, 2003, *Tip dan Trik Java 2 Micro Edition Mobile Interface Device Programing*, Elex Media Komputindo, Jakarta.

Hartanto, Antonius A, 2004, *Pemrograman Mobile Java dengan MIDP2.0*, Penerbit Andi, Yogyakarta.

IBM, 2006, Networking with J2ME, http://www.ibm.com

___,Putty, <http:/www.putty.nl>

Wibowo, Sugiharnito., Sugiantoro, Bambang dan Charibaldi ,Novrido., 2006 , Aplikasi Remote Server menggunakan Handphone.