

Dashboard:

Distributed Server's Resource Management

Case study Institut Teknologi Del

Tugas Akhir

Disampaikan Sebagai Bagian Dari Persyaratan Kelulusan Diploma 3
Program Studi Teknik Komputer

Oleh :

| | |
|----------|-------------------------|
| 11111047 | Sunday Yusuf Pakpahan |
| 11111066 | Ruhut Adventri Tambunan |



Institut Teknologi Del
2013/2014

**Lembar Pengesahan Tugas Akhir
Institut Teknologi Del**

**Dashboard:
Distributed Server's Resource Management
Case study Institut Teknologi De**

Oleh:

| | |
|----------|-------------------------|
| 11111047 | Sunday Yusuf Pakpahan |
| 11111066 | Ruhut Adventri Tambunan |

Dinyatakan memenuhi syarat dan karenanya disetujui dan disahkan sebagai
Laporan Tugas Akhir Diploma 3
Program Studi Teknik Komputer
Institut Teknologi Del

Sitoluama, 7 Agustus 14

Pembimbing



Marojahana Mula Timbul Sigitro, S.T., M.Sc.

NIDN. 0108098301

Daftar Isi

| | |
|---|----|
| Prakata | 3 |
| Abstrak | 4 |
| Daftar Isi | 5 |
| Daftar Gambar | 7 |
| Daftar Gambar | 8 |
| Daftar Tabel | 9 |
| BAB I Pendahuluan..... | 10 |
| 1.7 Latar Belakang | 10 |
| 1.8 Tujuan | 10 |
| 1.3 Lingkup | 11 |
| 1.9 Pendekatan | 11 |
| 1.10 Sistematika Penyajian | 12 |
| BAB II Tinjauan Pustaka | 13 |
| 2.1 Konsep Dasar <i>Dashboard</i> | 13 |
| 2.1.1 Sejarah <i>Dashboard</i> | 13 |
| 2.2 Pengertian <i>Server</i> | 14 |
| 2.2.1 Jenis <i>Server</i> | 14 |
| 2.2.2 <i>Platform Server</i> | 16 |
| 2.3 Jenis-jenis <i>resource</i> pada <i>server</i> | 16 |
| 2.3.1 Akun pengguna..... | 17 |
| 2.3.2 <i>File</i> | 18 |
| 2.3.3 <i>Memory</i> | 18 |
| 2.3.4 <i>Bandwidth</i> | 18 |
| 2.3.5 <i>Storage</i> | 19 |
| 2.3.6 <i>Device</i> | 19 |
| 2.4 Bahasa Pemrograman | 20 |
| 2.5 Komunikasi <i>Client/Server</i> | 22 |
| 2.5.1 Pengertian <i>Server</i> dan <i>Client</i> | 22 |
| 2.5.2 Mekanisme <i>Client/ Server</i> | 23 |
| 2.5.3 Keuntungan Menggunakan <i>Client/Server</i> | 24 |
| 2.6 Teknik Komunikasi Data | 24 |
| 2.6.1 <i>Socket Programming</i> | 24 |
| 2.6.2 <i>Web service</i> | 25 |
| 2.7 Format Data..... | 26 |
| BAB III Analisis dan Desain..... | 29 |
| 3.1 Analisis Permasalahan | 29 |
| 3.1.1 Analisis Permasalahan Sistem | 29 |
| 3.2 Analisis Sistem..... | 29 |
| 3.2.1 Deskripsi Umum Sistem | 29 |
| 3.2.2 Batasan Sistem | 30 |
| 3.3 Analisis Kebutuhan | 31 |
| 3.3.1 Analisis Kebutuhan Perangkat Keras (Hardware)..... | 31 |
| 3.3.2 Analisis Kebutuhan Perangkat Lunak (<i>User</i>) | 31 |
| 3.4 Desain..... | 31 |
| 3.4.1 Desain Arsitektur Sistem Secara Umum..... | 32 |
| 3.4.2 Desain Arsitektur Sistem Secara Detil..... | 33 |
| BAB IV Implementasi Dan Pengujian Sistem | 35 |
| 4.1 Lingkungan Operasi | 35 |
| 4.2 Implementasi Kelas Pada Java | 36 |
| 4.3 Implementasi <i>Script</i> Pada <i>Client</i> | 37 |
| 4.4 Implementasi Java Daemon Script | 37 |
| 4.5 Implementasi Alur Program | 40 |
| 4.6 Implementasi <i>Database</i> | 45 |
| 4.7 Implementasi Antarmuka Sistem | 47 |
| 4.7.1 Tampilan Menu Utama | 47 |
| 4.7.2 Tampilan Menu Users..... | 50 |

| | | |
|---------|--|----|
| 4.7.3 | Tampilan Menu Add User | 51 |
| 4.7.4 | Tampilan Menu Groups..... | 52 |
| 4.7.5 | Tampilan Menu Add Group..... | 53 |
| 4.7.6 | Tampilan Menu Clients | 54 |
| 4.7.7 | Tampilan Menu Add Client | 55 |
| 4.7.8 | Tampilan Menu Command | 56 |
| 4.7.9 | Tampilan Menu Command Send | 57 |
| 4.7.10 | Tampilan Menu Script Generator | 58 |
| 4.7.11 | Tampilan Menu Command Log..... | 59 |
| 4.8 | Instalasi dan Konfigurasi..... | 59 |
| 4.8.1 | Instalasi dan Konfigurasi pada <i>Server (Linux)</i> | 60 |
| 4.8.1.1 | Web Service dan PHPMyAdmin..... | 60 |
| 4.8.1.2 | <i>ServerManager Service</i> | 61 |
| 4.8.1.3 | Aplikasi Web | 64 |
| 4.8.2 | Instalasi dan Konfigurasi pada <i>Client Windows</i> | 65 |
| 4.8.2.1 | <i>Active Directory</i> | 65 |
| 4.8.2.2 | <i>WindowsAgent Service</i> | 66 |
| 4.8.3 | Instalasi dan Konfigurasi pada <i>Client Linux</i> | 72 |
| 4.8.3.1 | <i>LinuxAgent Service</i> | 72 |
| 4.9 | Batasan dalam Implementasi..... | 75 |
| 4.10 | Pengujian Sistem..... | 76 |
| 4.10.1 | Pengujian pada Kasus Pertama | 77 |
| 4.10.2 | Pengujian pada Kasus Kedua..... | 81 |
| 4.10.3 | Pengujian pada Kasus Ketiga | 82 |
| 4.10.4 | Pengujian pada Kasus Keempat..... | 84 |
| BAB V | Kesimpulan dan Saran..... | 87 |
| 5.1 | Kesimpulan | 87 |
| 5.2 | Saran | 87 |
| Rujukan | | 88 |

Daftar Gambar

| | |
|--|----|
| Gambar 1 Perbedaan PowerShell dan Command prompt | 22 |
| Gambar 2 Ilustrasi Client/Server | 23 |
| Gambar 3 Ilustrasi Kerja Socket Programming..... | 25 |
| Gambar 4 Cara Kerja Web ServerKeterangan gambar: | 25 |
| Gambar 5 Contoh Raw Text..... | 26 |
| Gambar 6 Contoh XML | 27 |
| Gambar 7 Contoh JSON..... | 27 |
| Gambar 8 Desain Arsitektur Dashboard Distributed Server's Resource Management..... | 32 |
| Gambar 9 Desain Komunikasi Antara Aplikasi dengan <i>Server</i> | 33 |
| Gambar 10 Java Daemon Script..... | 40 |
| Gambar 11 Komunikasi database dan aplikasi..... | 41 |
| Gambar 12 Koneksi pada server manager | 41 |
| Gambar 13 Koneksi pada agent di client..... | 42 |
| Gambar 14 Agent dengan Script | 42 |
| Gambar 15 Proses pada agent | 43 |
| Gambar 16 Pengembalian reply | 44 |
| Gambar 17 Pengiriman Berurutan | 44 |
| Gambar 18 Relasi antar tabel pada database | 46 |
| Gambar 19 Tampilan Menu Utama..... | 48 |
| Gambar 20 Tampilan Menu User | 50 |
| Gambar 21 Tampilan Menu Add User | 51 |
| Gambar 22 Tampilan Menu Group | 52 |
| Gambar 23 Tampilan Menu Add Group | 53 |
| Gambar 24 Tampilan Menu Clients | 54 |
| Gambar 25 Tampilan Menu Add Client | 55 |
| Gambar 26 Tampilan Menu Command | 56 |
| Gambar 27 Tampilan Menu Command Send | 57 |
| Gambar 28 Tampilan Menu Script Generator | 58 |
| Gambar 29 Tampilan Menu Command Log | 59 |
| Gambar 30 Menginstal LAMPP | 60 |
| Gambar 31 Menonaktifkan HTTPD dan mengaktifkan LAMPP | 60 |
| Gambar 32 <i>Web Service</i> aktif..... | 60 |
| Gambar 33 PHPMyAdmin aktif..... | 61 |
| Gambar 34 Struktur MySQLJavaConnector | 61 |
| Gambar 35 Pengiriman dan pengambilan pesan MySQLJavaConnector..... | 62 |
| Gambar 36 Struktur ServerManager | 62 |
| Gambar 37 Pengiriman dan pengambilan pesan ServerManager | 63 |
| Gambar 38 ServerProperties | 63 |
| Gambar 39 DRSM.jar pada suatu direktori | 63 |
| Gambar 40 Konfigurasi Java Daemon Script untuk ServerManager | 64 |
| Gambar 41 Mengaktifkan <i>ServerManager Service</i> | 64 |
| Gambar 42 Direktori aplikasi berada dalam direktori "htdocs" pada LAMPP | 64 |
| Gambar 43 Aplikasi Web telah siap digunakan | 65 |
| Gambar 44 DCPromo | 66 |
| Gambar 45 Struktur WindowsAgent | 67 |
| Gambar 46 Variabel pada WindowsAgent | 67 |
| Gambar 47 Pembuatan <i>file</i> oleh WindowsAgent | 68 |
| Gambar 48 Pengambilan dan pengiriman pesan oleh WindowsAgent | 68 |
| Gambar 49 Batch <i>Script</i> berekstensi ".bat" | 69 |
| Gambar 50 Penulisan <i>file</i> oleh ScriptGenerator pada Windows | 69 |
| Gambar 51 Pengubahan karakter oleh Replacer <i>Class</i> | 70 |
| Gambar 52 AgentProperties pada Windows | 70 |
| Gambar 53 AlwaysUp | 70 |
| Gambar 54 General Tab pada AlwaysUp..... | 71 |
| Gambar 55 Menjalankan Service pada AlwaysUp..... | 72 |
| Gambar 56 Pengiriman <i>command</i> dan argumen oleh LinuxAgent..... | 73 |

Daftar Gambar

| | |
|---|----|
| Gambar 57 Shell <i>Script</i> berekstensi ".sh" | 73 |
| Gambar 58 Penulisan <i>file</i> oleh ScriptGenerator pada Linux | 74 |
| Gambar 59 AgentProperties pada Linux | 74 |
| Gambar 60 LinuxAgent.jar pada suatu direktori | 74 |
| Gambar 61 Konfigurasi Java Daemon Script untuk LinuxAgent..... | 75 |
| Gambar 62 Mengaktifkan <i>LinuxAgent Service</i> | 75 |
| Gambar 63 Pengalokasian <i>user "fidelis"</i> ke grup "staff"..... | 78 |
| Gambar 64 Penambahan akun mysql untuk <i>user "fidelis"</i> | 79 |
| Gambar 65 <i>User "fidelis"</i> berstatus aktif..... | 79 |
| Gambar 66 Login dapat dilakukan ke mail-server(1) dengan <i>user "fidelis"</i> | 80 |
| Gambar 67 Penambahan grup "hrd" terhadap <i>user "fidelis"</i> | 80 |
| Gambar 68 <i>User "fidelis"</i> masuk ke grup "staff" dan "hrd" (multigroup)..... | 81 |
| Gambar 69 <i>User "fidelis"</i> dinonaktifkan pada mail-server(1)..... | 81 |
| Gambar 70 <i>User "fidelis"</i> gagal login pada mail-server(1) | 81 |
| Gambar 71 <i>User "fidelis"</i> pada mail-server(1) diaktifkan kembali..... | 82 |
| Gambar 72 <i>User "fidelis"</i> dapat kembali login pada mail-server(1) | 82 |
| Gambar 73 <i>Sharing folder</i> oleh <i>user "fidelis"</i> | 83 |
| Gambar 74 <i>User "fidelis"</i> mengubah grup folder menjadi milik "staff" | 83 |
| Gambar 75 <i>User "fidelis"</i> memberikan <i>full-access permission</i> kepada grup "staff" | 84 |
| Gambar 76 <i>User "refindo"</i> berstatus nonaktif pada ketiga server | 84 |
| Gambar 77 <i>User "refindo"</i> dihapus dari ketiga server beserta <i>home directory</i> miliknya | 85 |

Daftar Tabel

| | |
|---|----|
| Tabel 1 Penggunaan Java | 21 |
| Tabel 2 Penggunaan Shell Script..... | 21 |
| Tabel 3 Spesifikasi Minimum Perangkat Keras | 35 |
| Tabel 4 Lingkungan Perangkat Lunak Client dan Server..... | 35 |
| Tabel 5 Implementasi Kelas pada sistem | 36 |
| Tabel 6 Implementasi <i>Script</i> pada <i>Client</i> | 37 |
| Tabel 7 Deskripsi <i>database</i> | 45 |
| Tabel 8 Kondisi Server yang digunakan | 76 |
| Tabel 9 Resource yang dikelola | 76 |

BAB I

Pendahuluan

Bab pendahuluan menguraikan latar belakang yang merupakan pemikiran dan pertimbangan dalam pemilihan topik, tujuan dari pelaksanaan tugas akhir, batasan lingkup dari sistem yang dikembangkan, pendekatan yang digunakan, dan sistematika penulisan laporan.

1.7 Latar Belakang

Institut Teknologi Del adalah sebuah organisasi yang bergerak di bidang IT. Sebagai sebuah organisasi yang memiliki banyak *resources*. *Resources* pada hal ini merupakan sumber daya yang ada pada komputer seperti *user*, *group* maupun *file*. IT Del memiliki beberapa *server* yang digunakan untuk mengelola setiap *resources* yang ada dan menggunakan *platform* yang berbeda-beda (windows, linux). Secara umum *server* dibagi menjadi empat tugas utama, yaitu *file server*, *application server*, *mail server* dan *account server*. Setiap *server* berisi *resources* yang berbeda. Satu akun *user* bisa berelasi dengan beberapa *resources* yang terdapat pada *server* yang berbeda dan *platform* yang berbeda tetapi bisa dianggap sebagai satu *entity* atau *resources*.

Untuk itu dibutuhkan sebuah aplikasi berupa *dashboard* yang dapat digunakan untuk mengatur *resources* yang digunakan oleh *user* tanpa tergantung terhadap jenis atau *platform server* tempat *resources* tersebut ada. Hal ini sangat berguna untuk mengatur penggunaan *resources* pada dosen, staf maupun mahasiswa di IT Del.

Oleh karena itu, pada tugas akhir ini akan dilakukan pengembangan aplikasi untuk menangani pengaturan *resources server* dalam bentuk *dashboard* untuk memudahkan dalam mengelola *server* yang ada di IT Del.

1.8 Tujuan

Tujuan dari pelaksanaan Tugas Akhir ini adalah membangun aplikasi pengaturan *resources* terdistribusi pada server dengan penyajian dalam satu layar (*single screen dashboard*).

1.3 Lingkup

Ruang lingkup pengerajan tugas akhir ini adalah pembangunan *dashboard* dengan kriteria:

1. *Dashboard* bersifat dinamis dan *user-friendly*. Dinamis pada aplikasi ini berarti aplikasi dapat dikembangkan dengan lebih mudah oleh Administrator.
2. *Dashboard* tidak menangani server (secara fisik), tetapi berhubungan dengan *agent* untuk pengiriman instruksi ke server.
3. *Agent* adalah media perantara antara dashboard dan server yang berupa script.
4. *Dashboard* menangani *resource* terdistribusi pada server. Resource yang ditangani berupa user, group dan file.
5. *Server* yang digunakan adalah 2 (dua) server dengan sistem operasi Linux CentOS 6.0 dan Microsoft Windows Server 2008.
6. Administrator menambahkan fungsi secara manual, seperti menambahkan script untuk menghapus user.

1.9 Pendekatan

Pendekatan yang dilakukan terdiri dari eksplorasi, studi literatur, analisis, dan implementasi. Eksplorasi dilakukan untuk mencari pengetahuan mengenai pembangunan *dashboard*, administrasi *server* di Linux dan Windows, dan *tools* yang dapat digunakan untuk melakukan proses pembangunan tersebut. Studi literatur dilakukan untuk memperbanyak pemahaman terhadap teknologi *dashboard* dan penerapannya pada *server* terdistribusi, implementasi command pada server server dan eksekusinya atas server yang berjalan di platform yang berbeda, dan penerapan *agent* sebagai perantara server dan *dashboard*. Analisis dilakukan untuk menganalisis penerapan *dashboard* dalam menangani beberapa *server* terdistribusi. Implementasi dilakukan untuk membangun *dashboard* yang digunakan untuk mengatur *resource* pada beberapa *server* terdistribusi.

1.10 Sistematika Penyajian

Secara garis besar, dokumen tugas akhir ini terbagi menjadi beberapa bagian besar antara lain sebagai berikut:

Pada Bab Pertama dijelaskan pendahuluan yang mencakup latar belakang, tujuan, lingkup, pendekatan, dan sistematika penyajian Tugas Akhir.

Pada Bab Kedua dijelaskan tinjauan pustaka yang mencakup pengertian dan konsep *dashboard*, pengertian dan jenis *server*, jenis-jenis *resource* pada *server*, bahasa pemrograman yang dapat digunakan, struktur komunikasi *client-server*, teknik komunikasi data, dan format data yang dapat digunakan.

Pada Bab Ketiga dijelaskan analisis dan desain yang mencakup analisis permasalahan *user* dan sistem, analisis system yang berupa deskripsi umum system beserta dengan batasannya, analisis kebutuhan *user* dan system *hardware*, serta desain arsitekur system secara umum dan secara detil.

Pada Bab Keempat dijelaskan implementasi yang mencakup lingkungan operasi yang digunakan, implementasi kelas pada Java, implementasi *script* pada *client*, implementasi Java Daemon Script, implementasi alur program implementasi *database*, implementasi antarmuka sistem, instalasi dan konfigurasi, serta pengujian sistem.

Pada Bab Kelima dijelaskan kesimpulan dan saran yang mencakup hasil yang didapat pada penggeraan Tugas Akhir dan saran yang diberikan untuk sistem yang telah dibangun.

BAB II

Tinjauan Pustaka

Bab Tinjauan Pustaka berisi tinjauan pustaka mengenai konsep dasar dashboard, pengertian *server*, jenis-jenis *resource* pada *server*, bahasa pemrograman, komunikasi *client/server*, dan teknik komunikasi data.

2.1 Konsep Dasar Dashboard

Dashboard adalah tampilan visual dari informasi penting, yang diperlukan untuk mencapai satu atau beberapa tujuan, dengan mengkonsolidasikan dan mengatur informasi dalam satu layar (single screen), sehingga kerja organisasi dapat dimonitor secara sekilas [1]. *Dashboard* harus bisa menampilkan informasi dengan sebaik mungkin, sehingga pengguna dapat mengerti informasi yang terkandung didalamnya maupun menggunakan untuk mengolah informasi.

2.1.1 Sejarah Dashboard

Dashboard, sebelumnya dikenal sebagai Executive Information Sistem (EIS) yaitu sistem berbasis komputer yang mampu melayani kebutuhan informasi bagi eksekutif, mampu mengakses secara cepat informasi mutakhir dan mampu mengakses secara langsung pada laporan manajemen. Selain mudah digunakan, EIS didukung dengan grafik, menyajikan laporan pengecualian dan mampu melakukan penelusuran lebih rinci terhadap informasi yang diperolehnya. EIS dikembangkan pertama kali pada tahun 1980 dan dikenal serta digunakan oleh perusahaan besar dan menengah pada pertengahan tahun 1980. [2]

Seiring dengan munculnya data warehouse dan Business Intelligence dalam antarmuka yang sederhana dianggap tidak praktis disebabkan oleh informasi yang disediakan tidak lengkap, unreliable, dan tersebar dari berbagai *resource*. Hal ini menyebabkan EIS dihapus. [3]

Selama tahun 1990, data *warehouse*, *online analytical processing* (OLAP), dan juga *Business Intelligence* digunakan untuk menyediakan informasi yang lebih baik lagi. Penekannya terdapat pada *collecting* (mengumpulkan), *correcting* (memeriksa), *integrating* (mengintegrasikan), *storing* (menyimpan), dan *accessing* (mengakses)

informasi untuk menjamin *accuracy* (keakuratan), *timeliness* (aktualitas), dan *usefulness* (kegunaan). [3]

Hal yang menyebabkan *dashboard* dikenal hingga saat ini disebabkan oleh Enron Scandal pada tahun 2001. Perusahaan harus mampu untuk memonitor dan mengontrol secara tepat apa yang terjadi ditengah-tengah pemegang saham. Peningkatan tanggungjawab yang dikombinasikan dengan kecenderungan menurunnya ekonomi, mengharuskan *Chief Information Officer* (CIO) untuk menyajikan informasi yang dapat secara mudah dimengerti, efisien, dan selalu *update* sesuai dengan performansi ekonomi, kepada manajer perusahaan. Berdasarkan kebutuhan diatas, *marketplace* (pasar) segera menawarkan *user dashboard* untuk menyajikan informasi yang dibutuhkan oleh perusahaan terkait dengan keadaan finansial dan ekonomi.

2.2 Pengertian Server

Server adalah sebuah sistem komputer yang menyediakan jenis layanan tertentu dalam suatu jaringan komputer. *Server* menjalankan perangkat lunak administratif yang mengontrol akses terhadap jaringan dan sumber daya yang terdapat di dalamnya seperti *file* atau mesin pencetak, dan memberikan akses kepada *client* yang terhubung dalam jaringan [2].

2.2.1 Jenis Server

Berdasarkan fungsinya, *server* dapat dikategorikan dalam beberapa jenis, yaitu:

1. Akun *server*

Akun *server* adalah *server* yang mengatur seluruh akun yang digunakan dalam jaringan. Dengan akun *server*, pengelolaan akun menjadi terpusat, sehingga tidak ada duplikasi akun pada *server-server* yang terhubung dengan akun *server*. Akun *server* dapat digunakan untuk menambah akun, mengedit akun, menghapus akun, serta menentukan property akun dan hak aksesnya terhadap *server-server* yang ada.

2. *File server*

File server memberikan layanan kepada *client* untuk manajemen *file*, menyimpan dan mengambil *file*. *File server* dirancang dengan memori dan *hard disk* berkapasitas besar

dengan tujuan sebagai media penyimpanan tidak hanya *file-file client*, tetapi termasuk sistem operasi jaringan beserta aplikasi dan data yang dibutuhkan dalam jaringan.

3. Database *server*

Database *server* adalah program komputer yang menyediakan layanan data lainnya ke komputer atau program komputer, seperti yang ditetapkan oleh model klien-*server*. Istilah ini juga merujuk kepada sebuah komputer yang didedikasikan untuk menjalankan program *server* database. Database sistem manajemen database yang sering menyediakan fungsi *server*, dan beberapa DBMSs (misalnya, MySQL) secara eksklusif bergantung pada model klien-*server* untuk akses data.

4. *Server* aplikasi

Server aplikasi adalah aplikasi pada sistem komputer yang berfungsi melayani permintaan akses dari komputer pengguna atau klien. Salah satu contoh aplikasinya adalah *Web Server*. *Web Server* berisi tampilan informasi-informasi yang dapat diakses menggunakan *web browser* seperti Mozilla Firefox.

Penerapan *server* aplikasi:

- *Mail server*

Mail Server adalah suatu aplikasi pada komputer yang bertindak sebagai sebuah *server* (penyedia layanan) dalam jaringan atau internet, yang memiliki fungsi untuk melakukan penyimpanan dan distribusi yang berupa pengiriman, penyaluran, dan penerimaan surat elektronik atau *e-mail*.

- *HTTP/Web server*

Server HTTP atau *Web Server*/WWW adalah *web server* yang dapat dijalankan di banyak sistem operasi (Unix, BSD, Linux, Microsoft Windows dan Novell Netware serta *platform* lainnya) yang berguna untuk melayani dan memfungsikan situs *web*. Protokol yang digunakan untuk melayani fasilitas *web/www* ini menggunakan HTTP

2.2.2 Platform Server

Platform server adalah jenis sistem operasi yang berfungsi untuk menangani jaringan. *Platform* yang akan digunakan untuk *server* harus dipilih secara selektif sesuai dengan kebutuhan *server* dan sesuai dengan kriteria perangkat komputer *server*. Adapun beberapa *platform* yang umum digunakan pada *server* adalah sebagai berikut:

- Microsoft Windows *Server*

Windows *server* adalah merek dagang dari sebuah grup sistem operasi *server* yang dikeluarkan oleh Microsoft. Contoh dari windows *server* adalah:

1. Windows *server* 2003
 2. Windows *server* 2008
- Linux

Linux adalah sistem operasi open source yang diciptakan dari kernel buatan Linus Torvalds dibantu dengan pengembangan perangkat lunak gratis dari FSF (Free User Foundation) milik Richard Stallman. Linux unggul dari segi keamanan, pemeliharaan yang mudah, dan biaya penggunaan gratis. Contoh dari linux *server* adalah:

1. Ubuntu
2. Redhat
3. SUSE
4. Debian
5. CentOS

2.3 Jenis-jenis resource pada server

Resource yang ditangani oleh *server* terdiri atas akun pengguna, *file*, *memory*, *bandwidth*, *storage*, dan *device*.

Administrator perlu mengatur akun pengguna untuk mengendalikan sistem. Dengan akun pengguna, *administrator* dapat mengatur hak pengguna untuk tergabung dengan domain lainnya yang terdapat pada jaringan *server*. Selain itu *administrator* juga dapat menentukan stabilitas dan keakuratan *file* yang ada dengan menambah batasan pengguna misalkan mencegah pengguna melakukan perubahan terhadap *file* yang tidak dimiliki oleh pengguna akun yang sedang masuk kedalam sistem.

2.3.2 *File*

File secara umum adalah data yang dibentuk dari *byte* yang disimpan kedalam media penyimpanan. *File* memiliki penamaan yang unik untuk memudahkan *user* dalam mengakses *file* tersebut nantinya. Atribut *file* (sering disebut *file properties*) adalah informasi mengenai *file* yang ada meliputi lokasi *file*, ukuran, waktu pembuatan, nama pemilik *file*, serta grup atau pengguna yang diperbolehkan untuk mengakses *file* dan hak aksesnya (*read, write, execute*).

2.3.3 *Memory*

Memory adalah salah satu komponen yang sangat penting dalam komputer. *Memory* berfungsi sebagai tempat penyimpanan data sementara, dan menyediakan ketersediaan data tersebut sampai data selesai digunakan. Hal ini bertujuan untuk meringankan kerja *hard disk* agar ketika data di-load, data tersimpan di *memory* dan tidak membutuhkan akses kontinyu ke *hard disk*. Semakin besar kapasitas *memory*, semakin cepat performansi komputer. Konsep pembagian *memory* di jaringan memungkinkan beberapa komputer mengakses data secara bersama-sama menggunakan *memory* yang secara logis terdistribusi.

2.3.4 *Bandwidth*

Bandwidth adalah ukuran data yang dapat dikirimkan lewat koneksi jaringan per satuan waktu. *Bandwidth* dapat dibagikan untuk beberapa komputer yang terhubung ke jaringan untuk pengaksesan internet secara bersama. Pembagian *bandwidth* dapat dilakukan menggunakan *switch/hub*. Dibutuhkan *router* yang dapat membagi *bandwidth* ke setiap komputer sesuai kebutuhan untuk meningkatkan performansi koneksi jaringan.

2.3.5 Storage

Storage adalah tempat yang berfungsi sebagai penyimpanan data, baik berupa *file*, *setting*, sistem operasi, dan sebagainya. Penggunaan *storage* secara bersama bertujuan untuk memberikan kapasitas penyimpanan kepada *user* dan berbagi penyimpanan tersebut. Sebagai contoh adalah penyimpanan *user*, musik, dan video. Dengan kapasitas *file* yang cukup besar, akan sangat menghemat apabila hanya ada satu *storage* yang menyimpan setiap satuan *file* tersebut. Ketika *user* ingin menggunakan *file*, *file* dapat digunakan tanpa menambah beban dari komputer yang dipakainya, sehingga kapasitas dalam komputer dapat digunakan untuk lainnya. Contoh dari *storage* adalah *hard disk*.

2.3.6 Device

Device memiliki layanan yang dapat dibagikan kepada multi-*user* dalam jaringan. Tujuan dari pembagian *device* (*sharing devices*) pada *server* adalah untuk menghemat penggunaan peripheral, dan biaya produksi. Beberapa *device* tersebut adalah sebagai berikut:

- *Printer*

Printer dapat dibagikan kepada *user* lainnya didalam satu jaringan. Didukung oleh protocol, *printer* dapat menerima tugas mencetak *file* dari banyak *user* dan mengurnanya di setiap *printer* yang ada. Ketika ada *user* yang ingin mencetak dengan salah satu *printer*, maka *printer* akan aktif dan meletakkan *file* yang akan dicetak dalam antrian, kemudian mencetak *file* tersebut.

- Mesin Fax

Mesin fax adalah peralatan yang dapat digunakan untuk mengirimkan maupun menerima dokumen dengan bentuk fisik berupa kertas menggunakan perangkat yang beroperasi lewat jaringan (modem). Dengan menggunakan mesin fax, wujud dokumen yang dikirimkan akan persis sama dengan dokumen aslinya. Hanya *user* perlu memastikan tinta dan kertas khusus mesin fax selalu tersedia untuk menerima fax.

2.4 Bahasa Pemrograman

Untuk dapat membangun *dashboard* yang berfungsi untuk mengatur *resource* pada *server*, dibutuhkan bahasa pemrograman yang akan diaplikasikan pada *dashboard*. Bahasa pemrograman harus dapat dieksekusi pada *platform server* yang berbeda, dalam tugas akhir ini adalah Windows dan Linux. Adapun bahasa pemrograman yang dapat digunakan untuk pembangunan *dashboard* adalah sebagai berikut:

- Batch File

Batch file adalah sebuah skrip file yang apabila dijalankan akan berhubungan langsung dengan Shell Windows, sehingga bisa menjalankan perintah dasar dari sistem operasi Windows.

Contoh Program :

```
@echo off  
echo Test membuat Batch file  
pause  
dir c:\windows
```

- Java

Bahasa Java banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa *platform* sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun".

Contoh penggunaan java dapat dilihat pada Tabel 1.

Tabel 1 Penggunaan Java

| Code | Run |
|--|---|
| //MyProgram.java public class HelloWorld { public static void main(String[] args) { Sistem.out.println("Hello, World"); } } | 1. Buka cmd.exe 2. Ketik javac MyProgram.java, lalu tekan enter. (compile) 3. Ketik java MyProgram, lalu tekan enter. (execute) |

• *Shell script*

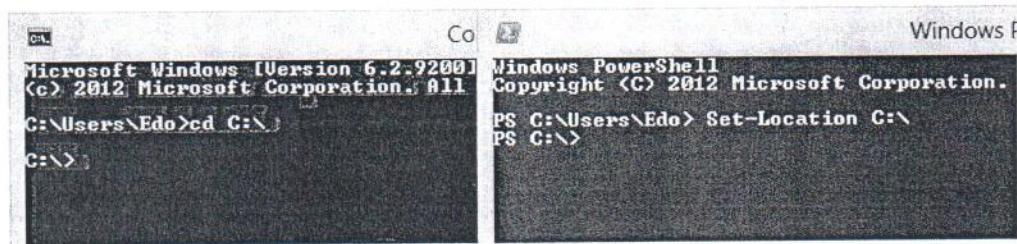
Shell merupakan suatu program yang menghubungkan *user* dengan sistem operasi. *Shell* menterjemahkan perintah dan parameter yang digunakan oleh *user* dalam lingkungan Linux. Pemrograman dengan menggunakan *shell* adalah penyusunan beberapa perintah untuk dieksekusi sesuai dengan tujuan penyusunannya. Hal yang perlu diperhatikan adalah hak akses dari *user* untuk mengeksekusi file berisi *shell script* tersebut. Contoh penggunaan *shell script* dapat dilihat pada Tabel 2.

Tabel 2 Penggunaan Shell Script

| Code | Run |
|---|-------------------------------------|
| //hallo.sh #!/bin/bash echo -n "Masukkan nama anda : " read nama echo -e "Hallo \$nama, Selamat Datang di Linux" | Ketik ./hallo.sh, lalu tekan enter. |

- *Powershell*

Powershell adalah bahasa pemrograman yang berguna bagi pengguna Windows, khususnya *administrator*. *Powershell*, tidak hanya menerima input perintah maupun *file script*, tetapi juga logika pemrograman, yang dikenal dengan cmdlet. Berbeda dengan *command prompt*, tugas administrasi sistem dapat diatur registry dengan WMI (Windows Management Instrumentation), sementara *command prompt* tidak mampu mengatasi hal tersebut. Contoh perbedaan penggunaan *powershell* dan *command prompt* dapat dilihat pada Gambar 1.



Gambar 1 Perbedaan PowerShell dan Command prompt

2.5 Komunikasi *Client/Server*

Pada subbab ini dijelaskan mengenai pengertian *client/server*, dan bagaimana mekanisme *client/server*.

2.5.1 Pengertian *Server* dan *Client*

Server adalah sebuah mesin (komputer), kombinasi dari perangkat keras dan perangkat lunak, yang menyediakan layanan untuk komputer lainnya pada sebuah jaringan. *Server* berfungsi untuk melayani permintaan *client*, sehingga komputer *server* memiliki spesifikasi yang lebih besar dibandingkan dengan komputer *client*.

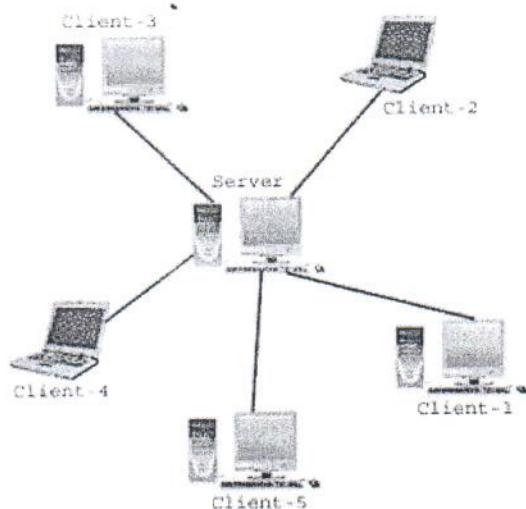
Client adalah pelanggan yang memberikan *request* kepada *server*. *Client* harus terhubung kepada *server* untuk dapat mengirimkan *request*.

2.5.2 Mekanisme Client/ Server

Client Server merupakan model jaringan yang menggunakan satu atau beberapa komputer sebagai *server* yang memberikan *resource*-nya kepada komputer lain (*client*) dalam jaringan, *server* akan mengatur mekanisme akses *resource* yang boleh digunakan, serta mekanisme komunikasi antar-node dalam jaringan.

Selain pada jaringan lokal, sistem ini bisa juga diterapkan dengan teknologi internet. Suatu unit komputer berfungsi sebagai *server* yang hanya memberikan pelayanan bagi komputer lain, dan *client* yang juga hanya meminta layanan dari *server*. Akses dilakukan secara transparan dari *client* dengan melakukan login terlebih dulu ke *server* yang dituju.

Client hanya bisa menggunakan *resource* yang disediakan *server* sesuai dengan otoritas yang diberikan oleh *administrator*. Aplikasi yang dijalankan pada sisi *client*, bisa saja merupakan *resource* yang tersedia di *server*, namun hanya bisa dijalankan setelah terkoneksi ke *server*. Pada implementasi *user* aplikasi yang di-install disisi *client* berbeda dengan yang digunakan di *server*. Ilustrasi arsitektur *client/server* dapat dilihat pada Gambar 2.



Gambar 2 Ilustrasi Client/Server

2.5.3 Keuntungan Menggunakan *Client/Server*

Berikut adalah keuntungan menggunakan *client/server*.

1. Manajemen jaringan terpusat sehingga keamanan terjamin
2. Kecepatan akses lebih tinggi karena fasilitas jaringan dikelola hanya oleh satu *server* yang tidak dibebani oleh tugas lainnya.
3. Sistem *back-up* data lebih baik, karena seluruh data terpusat di *server* sehingga memberikan kemudahan untuk memback-up seluruh data.

2.6 Teknik Komunikasi Data

Pada subbab ini dijelaskan mengenai format data dan cara data dapat berkomunikasi diantara *dashboard*, *agent* dan *server*.

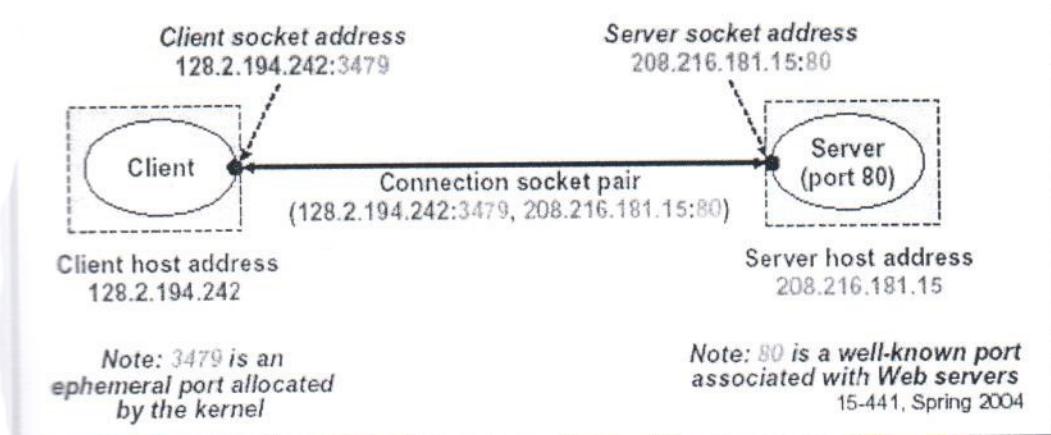
2.6.1 *Socket Programming*

Antarmuka (*interface*) antara program aplikasi dengan protokol komunikasi pada suatu sistem operasi disebut Application Program *Interface* (API). API didefinisikan sebagai suatu kumpulan instruksi yang mendukung proses interaksi antara suatu perangkat lunak dengan suatu protokol yang digunakan. Pada mesin keluarga Linux, *socket* terintegrasi dengan I/O sehingga aplikasi yang berkomunikasi dengan *socket* memiliki cara kerja yang sama dengan suatu aplikasi yang mengakses peralatan I/O.

Pada saat suatu aplikasi berkomunikasi, awalnya aplikasi membuat *socket* baru, maka pada aplikasi tersebut akan diberikan nomer yang digunakan sebagai referensi *socket*. Jika ada suatu sistem yang menggunakan nomer referensi *socket* tersebut, maka akan terjalin suatu jaringan komunikasi antar komputer sebaik transfer data lokal.

Untuk berkomunikasi dengan *server*, *client* harus tahu nomor IP *server* begitu juga nomor *port* yang dituju, nomor *port* menunjukkan *service* yang dijalankan. Contoh *port* 25 untuk *Mail Server* dan *port* 80 untuk *Web Server*. Dalam hal ini aplikasi di *client* sudah mengetahui *port* yang akan dituju. Untuk melihat *service* bisa dilihat pada file */etc/services*. Program yang berjalan di *server*, akan berjalan sepanjang waktu (disebut sebagai *daemon*) sampai mesin/*service* dimatikan, menunggu *request* dari *client* sesuai *service* yang diminta.

Ilustrasi kerja *socket programming* dapat dilihat pada Gambar 3.

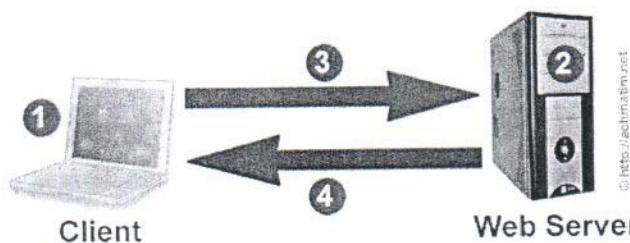


Gambar 3 Ilustrasi Kerja Socket Programming

2.6.2 Web service

Web service adalah aplikasi sekumpulan data (*database*), perangkat lunak (*user*) atau bagian dari perangkat lunak yang dapat diakses secara *remote* oleh berbagai piranti dengan sebuah perantara tertentu. Secara umum, *web service* dapat diidentifikasi dengan menggunakan URL seperti hanya *web* pada umumnya. Namun yang membedakan *web service* dengan *web* pada umumnya adalah interaksi yang diberikan oleh *web service*. Berbeda dengan URL *web* pada umumnya, URL *web service* hanya mengandung kumpulan informasi, perintah, konfigurasi atau sintaks yang berguna membangun sebuah fungsi-fungsi tertentu dari aplikasi.

Cara kerja *web server* dapat dilihat pada Gambar 4.



Keterangan gambar:

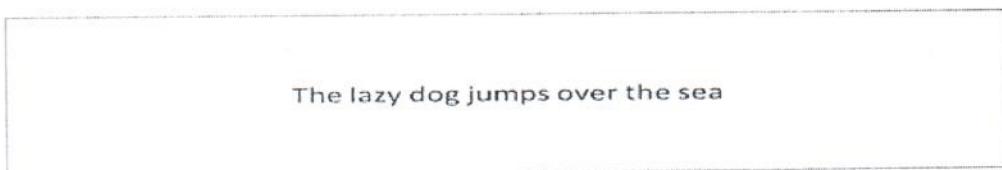
1. *Client* dapat berupa komputer desktop dengan minimal memiliki browser dan terhubung ke *web server* melalui jaringan (intranet atau internet).
2. Komputer yang berfungsi sebagai *server*, dimana didalamnya terdapat perangkat lunak *web server*. Agar komputer ini dapat diakses oleh *client* maka komputer harus terhubung ke jaringan (intranet atau internet). Dalam jaringan internet, komputer ini bisa saja bernama www.google.com, www.bl.ac.id, atau memiliki kode komputer (disebut IP Address) seperti 202.10.20.10 dan 200.100.50.25.
3. *Client (user)* akan meminta suatu halaman ke (*web*) *server* untuk ditampilkan di komputer *client*. Misalnya *client* mengetikkan suatu alamat (biasa disebut URL) di browser http://www.google.com. *Client* menekan tombol Enter atau klik tombol Go pada browser. Melalui media jaringan (bisa internet, bisa intranet) dan melalui protokol http, komputer bernama www.google.com dicari.
4. Dari sisi *server (web server)*, mendapat permintaan halaman utama google dari *client*, *server* akan mencari-cari di komputernya halaman sesuai permintaan. Jika ditemukan, maka halaman yang diminta akan dikirimkan ke *client*, namun jika tidak ditemukan, maka *server* akan memberi pesan “404. Page Not Found”, yang artinya halaman tidak ditemukan.

2.7 Format Data

Adapun format data yang digunakan untuk berkomunikasi dengan *server* adalah sebagai berikut:

- *Raw Text*

Text merupakan bahasa yang digunakan untuk menyimpan maupun memberikan informasi ke suatu aplikasi. Contoh *raw text* dapat dilihat pada Gambar 5.



Gambar 5 Contoh Raw Text

- *XML (Extensible Mark-up Language)*

XML merupakan suatu bahasa *markup* yang digunakan untuk membawa dan menyimpan data. Pada dasarnya XML digunakan untuk menyimpan dan memberikan informasi/data ke suatu aplikasi. Contoh xml dapat dilihat pada Gambar 6.

```
<student>
<nim>11111073</nim>
<name>Christine</name>
<grade>Third grade</grade>
</student>
```

Gambar 6 Contoh XML

- *JSON*

JSON (dilafalkan "Jason"), singkatan dari *JavaScript Object Notation* (bahasa Indonesia: notasi objek *JavaScript*), adalah suatu format ringkas pertukaran data komputer. Formatnya berbasis teks dan terbaca-manusia serta digunakan untuk merepresentasikan struktur data sederhana dan larik asosiatif (disebut objek). Contoh json dapat dilihat pada Gambar 7.

```
var student={
  "nim": 11111073
  "name": Christine
  "grade": third grade
}
```

Gambar 7 Contoh JSON

BAB III

Analisis dan Desain

Pada bab ini akan dibahas mengenai analisis terhadap masalah dari sistem yang akan dibangun secara lebih mendalam, juga menganalisis kebutuhan yang diperlukan dan solusi yang diberikan dalam menjawab permasalahan dari Tugas Akhir.

3.1 Analisis Permasalahan

Analisis merupakan proses penguraian masalah menjadi beberapa bagian yang saling berhubungan dalam memecahkan suatu masalah. Permasalahan muncul pada saat melakukan implementasi pada sistem yang akan dibangun.

3.1.1 Analisis Permasalahan Sistem

Sistem *distributed server resource's management* diatur oleh aplikasi *dashboard*. *Dashboard* dalam pengertian sederhananya adalah tampilan layar tunggal, tampilan ini menyediakan antarmuka *user* dengan sistem tanpa melihat kerumitan pada sistem tersebut. Tujuan dari sistem yang akan dibangun adalah:

1. *Platform* yang sama maupun berbeda dapat diberikan hak akses kepada setiap *user*.
2. Ketentuan kepada setiap *user* dalam mengakses *file* dapat diberikan.
3. Mengatur dan menangani *resource user, group* dan *file*.

3.2 Analisis Sistem

Pada subbab ini dijelaskan tentang analisis sistem yang berjalan, sistem yang dibangun, serta batasan sistem.

3.2.1 Deskripsi Umum Sistem

Sistem *distributed server resource's management* merupakan sistem yang digunakan untuk menjalankan perintah atau *command* yang telah disatukan dan dipermudah oleh adanya *dashboard*. Aplikasi inilah yang berfungsi untuk menjalankan perintah seperti menambah *user* pada *platform* yang berbeda yang dilakukan pada *single screen (dashboard)*.

Analisis Sistem Berjalan

Saat ini IT Del menggunakan cara *native/manual* untuk menambah *user*, menghapus *user*, dan mengakses *file* yaitu dengan mengakses *server* agar *resource* saling berhubungan.

Analisis Sistem yang Dibangun

Pembangunan aplikasi pada Tugas Akhir ini bertujuan untuk membuat sistem baru untuk mempermudah pekerjaan *administrator*, yaitu dapat mengelola *resource* yang berhubungan pada tiap-tiap *server* melalui sebuah *dashboard*.

Pada Tugas Akhir ini ada beberapa hal yang akan dibuat untuk sistem *distributed server's resource* menggunakan *dashboard*:

1. Menambah *user* untuk lebih dari satu sistem operasi yang berbeda.
2. *Permission file* untuk *user* dan *administrator*.
3. Menghapus *user* dan *file*.
4. *Disk usage*.
5. *Reset password*.
6. *Active* dan *non-active user*.
7. Menambah *Group*.
8. Menghapus *Group*.
9. Mengubah *Group* suatu *user*.

3.2.2 Batasan Sistem

Batasan dari sistem yang dikembangkan adalah:

1. Menangani *dashboard* dan konfigurasi yang berhubungan dengan komunikasi antara *server* dengan *dashboard*.
2. Menangani pengelolaan *user* dan *file* yang terhubung dengan *user* yang bersangkutan.
3. Menangani *resource* seperti *user*, *group* dan *file* pada *server* dengan sistem operasi Linux CentOS dan Windows Server 2008.
4. Tidak menangani kesalahan konfigurasi yang dilakukan oleh *user*, seperti memasukkan *script* yang salah pada *server* *server*.

3.3 Analisis Kebutuhan

Dalam subbab ini dijelaskan mengenai analisis terhadap kebutuhan *user* dan kebutuhan sistem.

3.3.1 Analisis Kebutuhan Perangkat Keras (Hardware)

Sistem ini diharapkan dapat mempermudah *administrator* dalam pekerjaanya. Perangkat yang dipakai dalam sistem membangun *dashboard* adalah sebagai berikut:

1. *Personal Computer*, digunakan sebagai *server* yang diinstalasi di berbagai *platform*, seperti windows server dan linux CentOS.
2. Perangkat jaringan, digunakan untuk menghubungkan setiap *platform* satu dengan yang lain. Perangkat yang dimaksud dapat berupa kabel LAN atau *Access Point* untuk menghubungkan *Personal Computer* dengan jaringan lokal (LAN).

3.3.2 Analisis Kebutuhan Perangkat Lunak (User)

Dalam membangun aplikasi, perangkat lunak yang dibutuhkan antara lain:

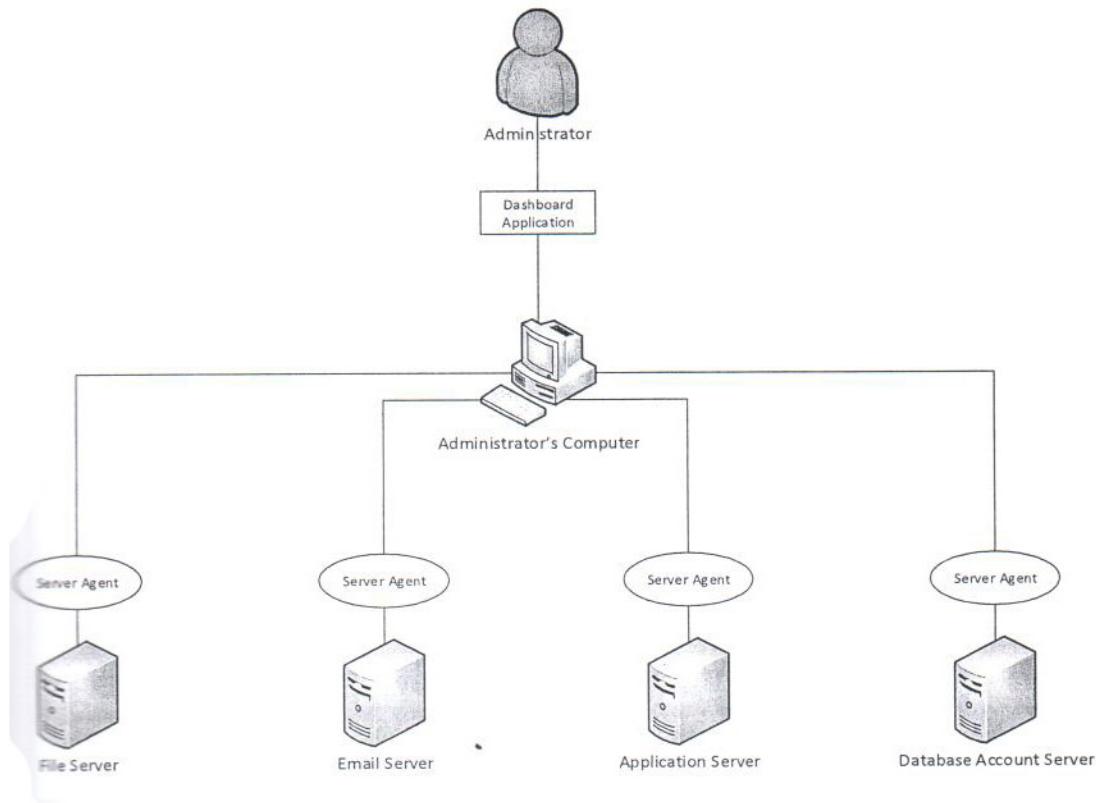
1. VMWare Workstation 9.0, digunakan untuk membuat sebuah *server* percobaan sebelum dihubungkan ke *server* yang sesungguhnya.
2. Netbeans 7.0.1, digunakan untuk membangun aplikasi yang berkomunikasi dengan *server*, yaitu aplikasi *dashboard* dan *server agent*.
3. JDK 1.7, digunakan untuk membangun java berbasis aplikasi.
4. Mysql, digunakan untuk mengatur *database* yang akan digunakan untuk menyimpan data yang telah dibuat pada *dashboard*.

3.4 Desain

Dalam subbab ini digambarkan desain arsitektur sistem secara umum dan desain arsitektur sistem secara detil.

3.4.1 Desain Arsitektur Sistem Secara Umum

Secara umum, desain arsitektur sistem dapat dilihat pada Gambar 8.

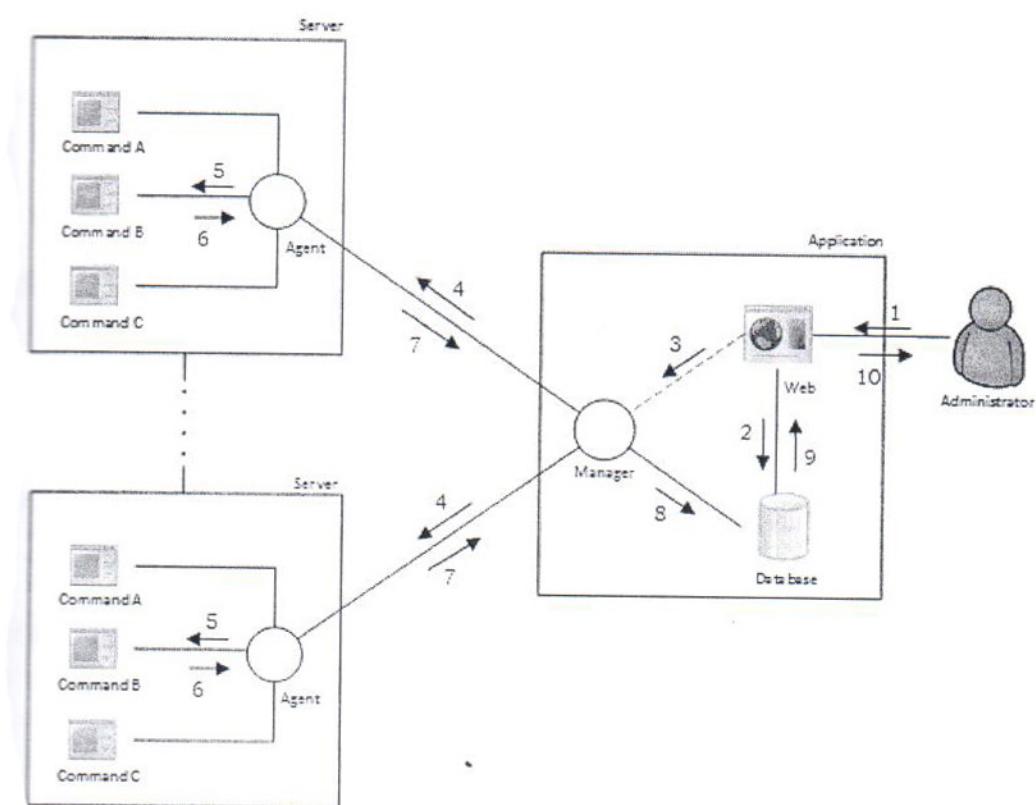


Gambar 8 Desain Arsitektur Dashboard Distributed Server's Resource Management

Administrator menggunakan Aplikasi *Dashboard* untuk menjalankan sistem yang dibangun. Aplikasi yang dibangun terkoneksi langsung ke server-server yang memiliki *resource*. Aplikasi tersebut melakukan komunikasi ke server dengan menggunakan *Server Agent* yang ditanamkan pada tiap server. Administrator mengirimkan pesan kepada *Server Agent* melalui *Manager* yang ada di bagian aplikasi dan kemudian menjalankan perintah untuk dijalankan di server-server yang memiliki *resource*. Hubungan sebuah *resource* dengan *resource* lain di server yang berbeda tersimpan pada *database*.

3.4.2 Desain Arsitektur Sistem Secara Detil

Secara detil, desain arsitektur sistem dapat dilihat pada Gambar 9.



Gambar 9 Desain Komunikasi Antara Aplikasi dengan *Server*

Keterangan gambar berdasarkan urutan berjalannya aplikasi adalah sebagai berikut:

1. Aplikasi *Dashboard* yang dibangun adalah aplikasi yang berbasis web. Oleh karena itu, administrator mengelola resource dengan memasukkan perintah yang akan dikirim ke server melalui web.
2. Aplikasi berinteraksi dengan *database* untuk mendapatkan data berupa perintah yang dikirim ke server sesuai dengan jenis server yang bersangkutan.
3. Aplikasi memiliki *Application Manager* yang dibuat untuk menerima perintah yang dikirimkan kepada *Server Agent* yang berada di server-server tujuan.
4. *Application Manager* mengirim pesan ke *Server Agent* pada tiap server yang memiliki hubungan dengan user yang dikelola *resource*-nya.

5. *Server Agent* menjalankan perintah (*command*) yang berdasarkan pesan yang diterima dari *Application Manager*. Perintah yang dijalankan dapat dipilih. Jika server tersebut adalah server Linux, maka *Server Agent* menjalankan perintah Linux. Jika server tersebut adalah server Windows, maka *Server Agent* menjalankan perintah Windows.
6. Setelah perintah di tiap server selesai dijalankan, *Server Agent* menerima respon berupa pesan apakah perintah berhasil dijalankan, gagal dijalankan, atau informasi lainnya yang dibutuhkan.
7. Respon dari perintah yang telah dijalankan dikirim kembali kepada *Application Manager*.
8. *Application Manager* menyimpan respon dari perintah tersebut ke dalam *database*.
9. *Application Manager* berinteraksi dengan *database* untuk mengambil respon dari perintah yang telah dijalankan pada server tujuan.
10. Aplikasi menampilkan informasi berupa respon server terhadap perintah yang dikirimkan melalui interaksi dengan *database* sehingga administrator dapat memastikan apakah perintah tersebut berhasil dijalankan, gagal dijalankan, atau menerima informasi penting lainnya.

BAB IV

Implementasi Dan Pengujian Sistem

Setelah melalui tahap analisis dan perancangan, tahap selanjutnya untuk mengembangkan suatu perangkat lunak adalah tahap implementasi dan pengujian sistem. Untuk mengetahui apakah implementasi perangkat lunak tersebut berhasil atau tidak, diperlukan pengujian. Berikut ini hasil implementasi dan pengujian dari aplikasi yang telah dibangun.

4.1 Lingkungan Operasi

Lingkungan operasi mencakup lingkungan perangkat keras dan perangkat lunak untuk menghubungkan antar *device*. Spesifikasi minimum perangkat keras untuk dapat menjalankan program dapat dilihat pada Tabel 3.

Tabel 3 Spesifikasi Minimum Perangkat Keras

| Perangkat Keras | Deskripsi |
|-----------------|--------------------------|
| Processor | Pentium(R) Dual-Core CPU |
| RAM | 1 GB |
| Hardisk | 100 GB |

Spesifikasi perangkat lunak yang dipakai *developer* dapat dilihat pada Tabel 4.

Tabel 4 Lingkungan Perangkat Lunak Client dan Server

| Perangkat Lunak | Deskripsi(bagian) |
|--------------------|--|
| Sistem Operasi | Windows Server 2008 R2 (client) Linux CentOS 6.4 (<i>client</i> dan <i>server</i>) |
| Web Server | Apache 2.0 (<i>server</i>) |
| Bahasa Pemrograman | Java (<i>client</i> dan <i>server</i>) Batch Script (<i>client-Windows</i>) Shell Script (<i>client-Linux</i>) |
| Database | MySQL (<i>server</i>) |
| IDE | Notepad++ (<i>client-windows</i>) Netbeans 7.0.1 (<i>server</i>) |
| Framework | Yii Framework (<i>server</i>) |

4.2 Implementasi Kelas Pada Java

Kelas yang diimplementasikan agar Sistem dapat berjalan dan digunakan untuk membangun koneksi antar *client-server* dapat dilihat pada Tabel 5.

Tabel 5 Implementasi Kelas pada sistem

| Package | Kelas | Keterangan |
|---------|-----------------------|---|
| DSRM | ServerManager | Kelas ini berfungsi untuk membangun koneksi antara <i>client-server</i> . |
| | MySQLJavaConnector | Kelas ini berfungsi menghubungkan server agar dapat terhubung ke <i>database</i> . |
| | LinuxAgent | Kelas ini berfungsi sebagai perantara antar server utama dengan server lainnya pada linux. |
| | WindowsAgent | Kelas ini berfungsi sebagai perantara antar server utama dengan server lainnya pada windows. |
| | ScriptGenerator | Kelas ini berfungsi untuk membuat <i>script</i> baru sesuai dengan yang diinginkan pengguna baik di Windows maupun di Linux. |
| | ReplaceListFile | Kelas ini berfungsi untuk mengganti karakter pada fungsi “ListFile” yang diterima oleh WindowsAgent untuk menghindari <i>error</i> pada tiap lingkungan operasi. |
| | ReplacePermissionFile | Kelas ini berfungsi untuk mengganti karakter pada fungsi “SetPermission” yang diterima oleh WindowsAgent untuk menghindari <i>error</i> pada tiap lingkungan operasi. |
| | ServerProperties | Properties ini berfungsi menyimpan konfigurasi yang dibaca oleh kelas ServerManager sesuai dengan kebutuhan <i>server</i> . |
| | AgentProperties | Properties ini berfungsi menyimpan konfigurasi yang dibaca oleh kelas WindowsAgent dan LinuxAgent sesuai dengan kebutuhan <i>client</i> . |

4.3 Implementasi Script Pada Client

Script yang diimplementasikan agar sistem dapat berjalan dan menjalankan fungsi-fungsi utama dapat dilihat pada Tabel 6.

Tabel 6 Implementasi Script pada Client

| Nama Script | Fungsi | Batch Script | Shell Script |
|-------------------|--|--------------|--------------|
| ActivateUser | Mengaktifkan <i>user</i> | ✓ | ✓ |
| AddGroup | Menambah grup | ✓ | ✓ |
| AddUser | Menambah <i>user</i> | ✓ | ✓ |
| AssignUserToGroup | Menempatkan <i>user</i> pada sebuah grup | ✓ | ✓ |
| ChangeGroup | Mengubah grup sebuah file/direktori | ✓ | ✓ |
| ChangeOwner | Mengubah kepemilikan sebuah file/direkori | ✓ | ✓ |
| ChangeUserGroup | Mengubah grup sebuah <i>user</i> | ✓ | ✓ |
| DeactivateUser | Menonaktifkan <i>user</i> | ✓ | ✓ |
| DeleteFile | Menghapus sebuah file | ✓ | ✓ |
| DeleteGroup | Menghapus grup | ✓ | ✓ |
| DeleteUser | Menghapus <i>user</i> | ✓ | ✓ |
| DiskUsage | Mengembalikan jumlah memori yang digunakan | ✓ | ✓ |
| GenerateScript | Menghasilkan script dengan memanggil kelas ScriptGenerator | ✓ | ✓ |
| GetFilePermission | Mengembalikan permission sebuah file | ✓ | ✓ |
| GetGroup | Mengembalikan grup sebuah file | ✓ | ✓ |
| GetOwner | Mengembalikan pemilik sebuah file | ✓ | ✓ |
| ListFiles | Mengembalikan daftar file pada sebuah direktori | ✓ | ✓ |
| ListUser | Mengembalikan daftar <i>user</i> pada sebuah client | ✓ | ✓ |
| RemoveDir | Menghapus sebuah direktori | ✓ | ✓ |
| SetPassword | Menetapkan password sebuah <i>user</i> | ✓ | ✓ |
| SetPermission | Menetapkan permission sebuah file | ✓ | ✓ |

Keterangan gambar:

Tanda ceklis (✓) menandakan bahwa script ada sebagai Batch Script maupun Shell Script.

4.4 Implementasi Java Daemon Script

Java Daemon Script adalah script yang digunakan untuk menjalankan program sebagai sebuah *service*. Java Daemon Script digunakan pada lingkungan operasi Linux. Implementasi Java Daemon Script dapat dilihat pada **Gambar 10**.

```
#!/bin/bash

serviceNameLo="linuxagent"
serviceName="LinuxAgent"
serviceUser="root"
serviceGroup="root"
applDir="/home/ta2/$serviceNameLo"
serviceUserHome="/home/$serviceUser"
serviceLogFile="/var/log/$serviceNameLo.log"
maxShutdownTime=15
pidFile="/var/run/$serviceNameLo.pid"
javaCommand="java"
javaExe="$JAVA_HOME/bin/$javaCommand"
javaArgs="-jar $applDir/LinuxAgent.jar"
javaCommandLine="$javaExe $javaArgs"
javaCommandLineKeyword="LinuxAgent.jar"

function makeFileWritable {
    local filename="$1"
    touch $filename || return 1
    chgrp $serviceGroup $filename || return 1
    chmod g+w $filename || return 1
    return 0; }

function checkProcessIsRunning {
    local pid="$1"
    if [ -z "$pid" -o "$pid" == " " ]; then return 1; fi
    if [ ! -e /proc/$pid ]; then return 1; fi
    return 0; }

function checkProcessIsOurService {
    local pid="$1"
    if [ "$(ps -p $pid --no-headers -o comm)" != "$javaCommand" ];
    then return 1; fi
    grep -q --binary -F "$javaCommandLineKeyword" /proc/$pid/cmdline
    if [ $? -ne 0 ]; then return 1; fi
    return 0; }

to the PID.
function getServicePID {
    if [ ! -f $pidFile ]; then return 1; fi
    pid="$(<$pidFile)"
    checkProcessIsRunning $pid || return 1
    checkProcessIsOurService $pid || return 1
    return 0; }

function startServiceProcess {
    cd $applDir || return 1
    rm -f $pidFile
    makeFileWritable $pidFile || return 1
    makeFileWritable $serviceLogFile || return 1
```

```

cmd="nohup $javaCommandLine >>$serviceLogFile 2>&1 & echo \$!
>$pidFile"
    su -m $serviceUser -s $SHELL -c "$cmd" || return 1
    sleep 0.1
    pid=$(cat $pidFile)
    if checkProcessIsRunning $pid; then :; else
        echo -ne "\n$serviceName start failed, see logfile."
        return 1
    fi
    return 0; }

function stopServiceProcess {
    kill $pid || return 1
    for ((i=0; i<maxShutdownTime*10; i++)); do
        checkProcessIsRunning $pid
        if [ $? -ne 0 ]; then
            rm -f $pidFile
            return 0
        fi
        sleep 0.1
    done
    echo -e "\n$serviceName did not terminate within $maxShutdownTime
seconds, sending SIGKILL..."
    kill -s KILL $pid || return 1
    local killWaitTime=15
    for ((i=0; i<killWaitTime*10; i++)); do
        checkProcessIsRunning $pid
        if [ $? -ne 0 ]; then
            rm -f $pidFile
            return 0
        fi
        sleep 0.1
    done
    echo "Error: $serviceName could not be stopped within
$maxShutdownTime+$killWaitTime seconds!"
    return 1; }

function startService {
    getServicePID
    if [ $? -eq 0 ]; then echo -n "$serviceName is already running";
    RETVAL=0; return 0; fi
    echo -n "Starting $serviceName      "
    startServiceProcess
    if [ $? -ne 0 ]; then RETVAL=1; echo "failed"; return 1; fi
    echo "started PID=$pid"
    RETVAL=0
    return 0; }

function stopService {
    getServicePID
    if [ $? -ne 0 ]; then echo -n "$serviceName is not running";
    RETVAL=0; echo ""; return 0; fi
    echo -n "Stopping $serviceName      "
    stopServiceProcess
    if [ $? -ne 0 ]; then RETVAL=1; echo "failed"; return 1; fi
    echo "stopped PID=$pid"
    RETVAL=0
    return 0; }

```

```

function checkServiceStatus {
    echo -n "Checking for $serviceName: "
    if getServicePID; then
        echo "running PID=$pid"
        RETVAL=0
    else
        echo "stopped"
        RETVAL=3
    fi
    return 0; }

function main {
    RETVAL=0
    case "$1" in
        start)
            startService
            ;;
        stop)
            stopService
            ;;
        restart)
            stopService && startService
            ;;
        status)
            checkServiceStatus
            ;;
        *)
            echo "Usage: $0 {start|stop|restart|status}"
            exit 1
            ;;
    esac
    exit $RETVAL
}

main $1

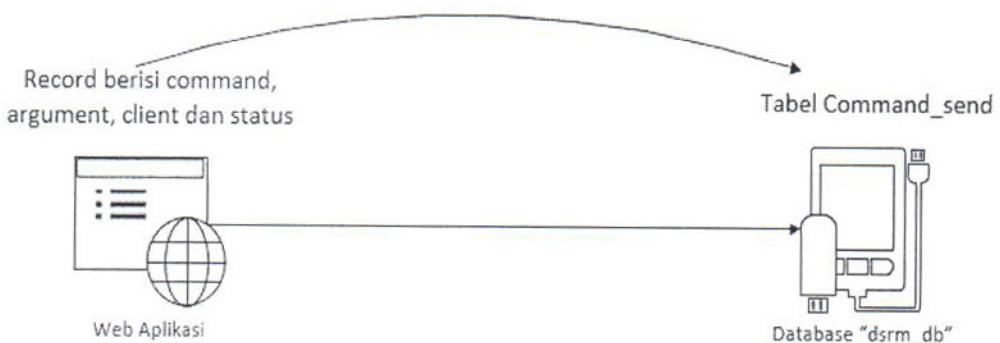
```

Gambar 10 Java Daemon Script

4.5 Implementasi Alur Program

Pada subbab ini akan dibahas alur daripada program yang telah dibangun. Alur dimulai dari aplikasi yang membangun koneksi dengan database, kemudian membuat server manager. Server manager mengirim pesan dari database melalui socket dan berkomunikasi dengan agent pada client dengan port yang sama. Agent kemudian menjalankan script sesuai dengan pesan yang diterima lalu memberikan reply ke server manager yang pada akhirnya dimasukkan ke dalam database. Pengiriman pesan ini dilakukan secara berurutan jika dilakukan pada lebih dari satu client. Untuk penjelasan lebih detail dapat dilihat dibawah ini.

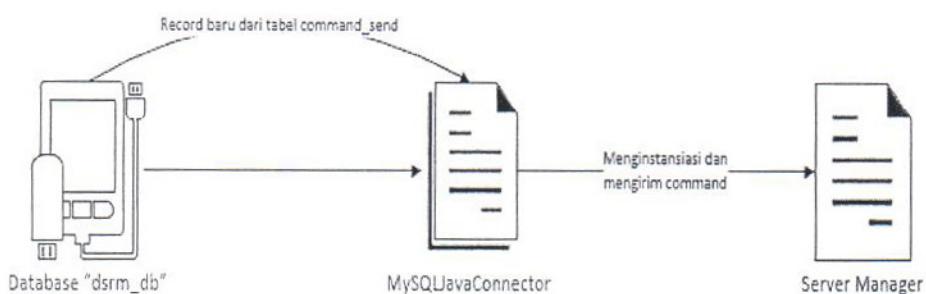
- Pembuatan Koneksi antara Database dengan Aplikasi
- Aplikasi yang dibangun dihubungkan dengan database “dsrm_db”, lalu memasukkan record yang berisi “command”, “argument”, “client”, dan “status” ke tabel “command_send”, dan aplikasi akan memasukkan nilai status “0” ke database untuk dibaca oleh connector.



Gambar 11 Komunikasi database dan aplikasi

- Pembuatan Koneksi pada Server Manager

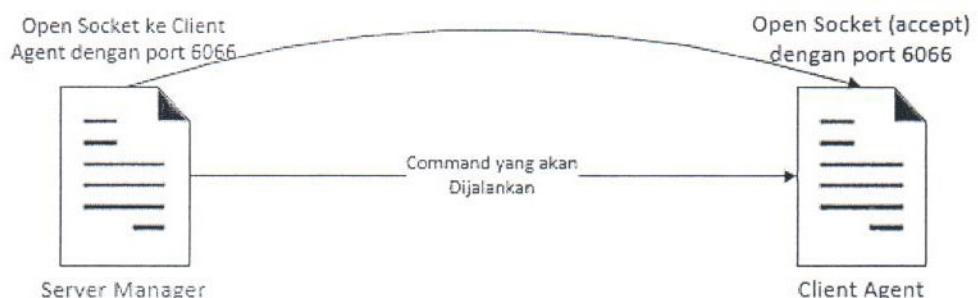
Koneksi antara database dengan server manager membutuhkan connector yang menggunakan driver SQLServerDriver, pada pembahasan sebelumnya telah dibuat hubungan antara aplikasi dengan database yang bernama “dsrm_db” dengan user “root” tanpa password, lalu server manager diinstansiasi dengan nomor port yang ada pada ServerProperties. Connector akan menunggu record baru dari database yang dimasukkan pada tabel “command_send”, dan setelah masuk ke tabel “command-send”, server manager akan mendapat informasi client yang dituju berdasarkan record yang masuk ke database dengan status "0", dan server manager menginstansiasi sebuah socket setiap mendapat record baru dari database.



Gambar 12 Koneksi pada server manager

Pembuatan Koneksi pada Client

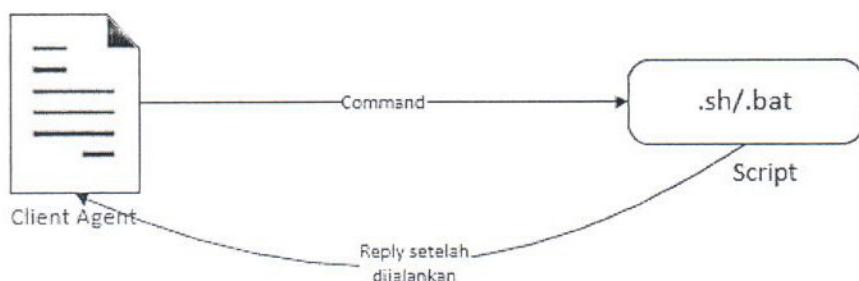
Client memiliki agent yang bertugas membuat sebuah koneksi dengan membuka socket dan menunggu adanya pesan yang datang pada port yang sama seperti server (nomor port dapat diatur melalui AgentProperties) dengan status “accept”. Setelah agent terhubung dengan server manager, agent menerima pesan dan memprosesnya. Agent memisahkan command yang dijalankan dan argument yang mengikuti command berdasarkan command yang diterima, lalu agent menjalankan script pada client dengan nama yang sama diikuti dengan argument yang diterima dari server manager.



Gambar 13 Koneksi pada agent di client

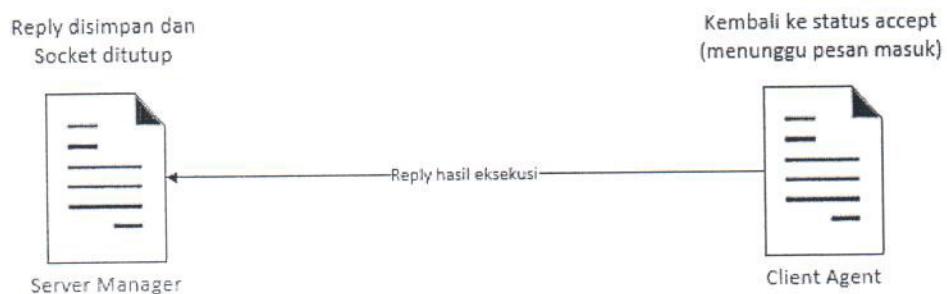
- Pengeksekusian script pada Client

Script menjalankan perintah untuk mengelola resource pada client, setelah menjalankan perintah, script menghasilkan output, dan output tersebut menjadi reply untuk agent.



Gambar 14 Agent dengan Script

- Proses pada Agent
 - Agent memproses output dan dikirimkan sebagai balasan ke server manager, lalu setelah mengirimkan reply, agent kembali ke status “accept” untuk menunggu koneksi selanjutnya dari server manager.
 - Penutupan Socket pada Server Manager
- Setelah server manager menerima reply dari agent, socket pada server manager kemudian ditutup.



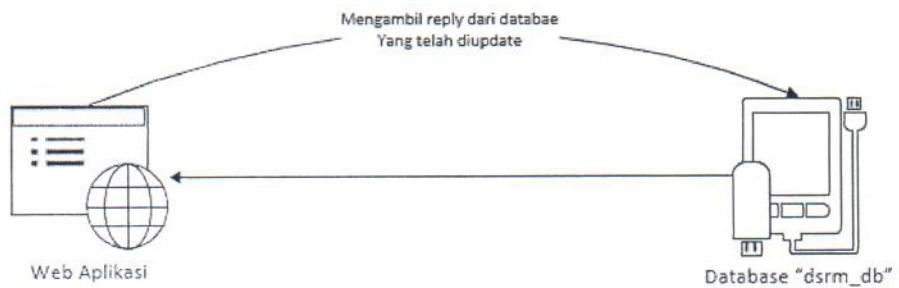
Gambar 15 Proses pada agent

Pengembalian Reply

Reply tersebut kemudian diambil oleh connector, lalu connector mengupdate database dan mengisi field “reply” sesuai dengan balasan yang diterima server manager dari agent, dan setelah mengupdate *database*, connector akan menunggu *record* baru lagi dari *database* untuk kemudian dijalankan oleh server manager.

Waiting Reply

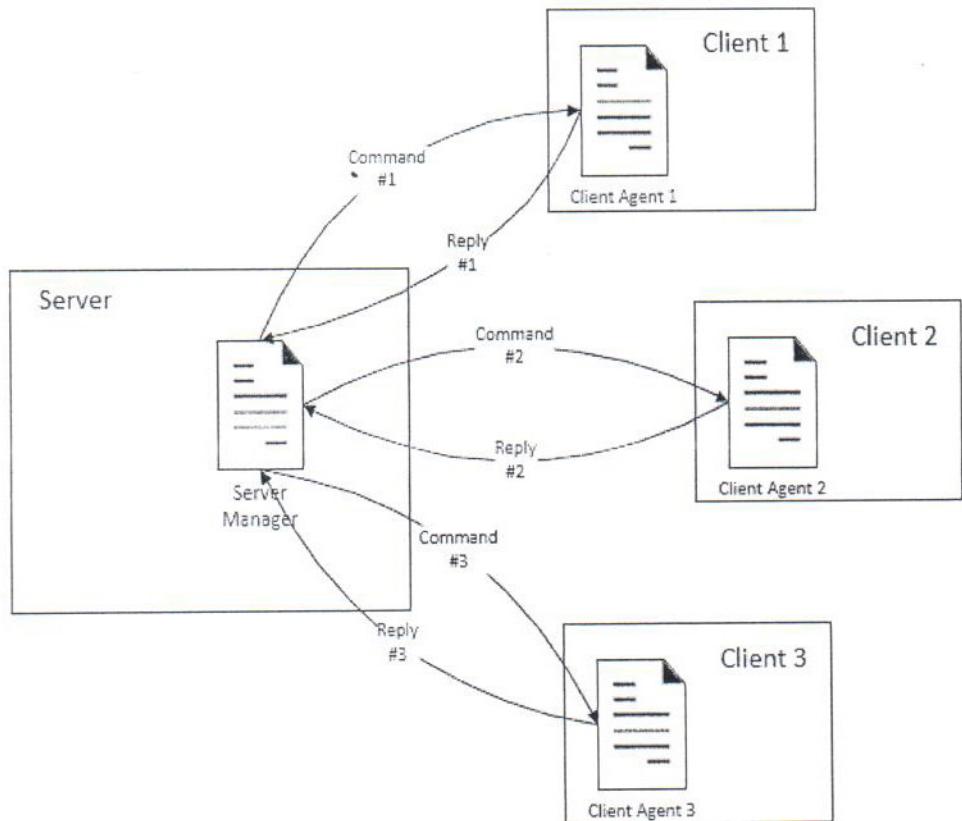
Aplikasi akan menunggu field “reply” pada tabel “command_send” agar diisi oleh connector, dan setelah field “reply” terisi, maka aplikasi akan mengambil informasi dari field tersebut, lalu *reply* tersebut digunakan untuk mengecek keberhasilan perintah mengelola *resource* dan kemudian diproses pada aplikasi sesuai dengan kebutuhan.



Gambar 16 Pengembalian reply

- Pengiriman Berurutan

Pengiriman pesan yang ditujukan kepada lebih dari satu client akan dilakukan secara berurutan. Aplikasi akan memasukkan *record* ke *database* sejumlah client yang ditujukan dan kemudian menunggu sampai perintah pada setiap client diselesaikan.



Gambar 17 Pengiriman Berurutan

4.6 Implementasi Database

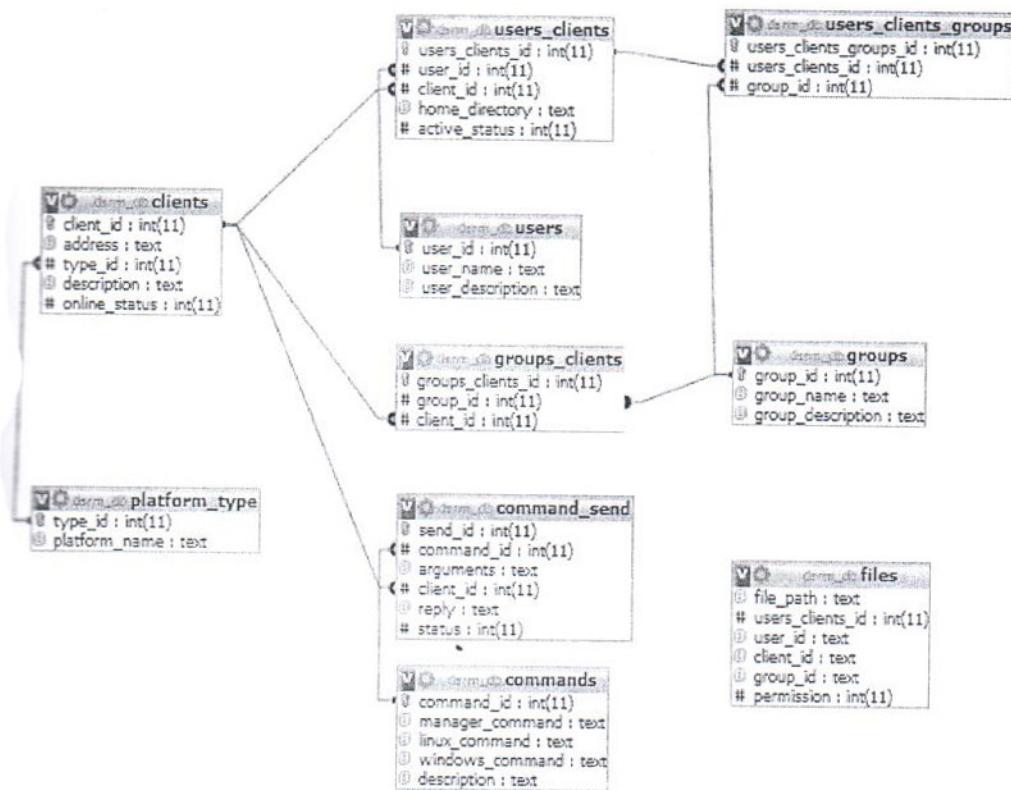
Database diimplementasikan untuk dapat menyimpan data penting dan relasi antar tabel yang telah dibuat.

Tabel 7 Deskripsi *database*

| Nama Database | Table | Struktur tabel(type) |
|---------------|----------------------|---|
| dsrm_db | clients | client_id(int(11)) address(text) type_id(int(11)) description(text) online_status(int(11)) |
| | command | command_id(int(11)) manager_command(Text) agent_command(Text) type_id(Int(11)) description (Text) |
| | command_send | send_id(int(11)) command_id(int(11)) arguments(text) client_id(int(11)) reply(text) status(int(11)) |
| | files | file_path(text) users_clients_id(int(11)) user_id(text) client_id(text) group_id(text) permission(int(11)) |
| | groups | group_id(int(11)) group_name(text) group_description(text) |
| | groups_clients | group_client_id group_id client_id |
| | platform_type | type_id(int(11)) platform_name(text) |
| | users | user_id(int(11)) user_name(text) user_description(text) |
| | users_clients | users_clients_id(int(11)) user_id(int(11)) client_id(int(11)) home_directory(text) active_status(int(11)) |
| | users_clients_groups | users_clients_groups_id(int(11)) users_clients_id(int(11)) |

| Nama Database | Tabel | Struktur tabel(type) |
|---------------|-------|----------------------|
| | | group_id(int(11)) |

Relasi antar tabel pada *database* dapat dilihat pada Gambar 18.



Gambar 18 Relasi antar tabel pada database

Keterangan gambar:

- Relasi antara **platform_type** dan **clients**, pada relasi ini **clients** membutuhkan *primary key* dari **platform_type** untuk menentukan tipe *platform* yang digunakan *client*.
- Relasi antara **clients** dan **users_clients**, pada relasi ini **users_clients** membutuhkan *primary key* dari **clients** untuk menentukan *client* mana saja yang ada pada sebuah *user*.
- Relasi antara **users** dan **users_clients**, pada relasi ini **users_clients** membutuhkan *primary key* dari **users** untuk menentukan *user* mana saja yang tersebar pada sebuah *client*.

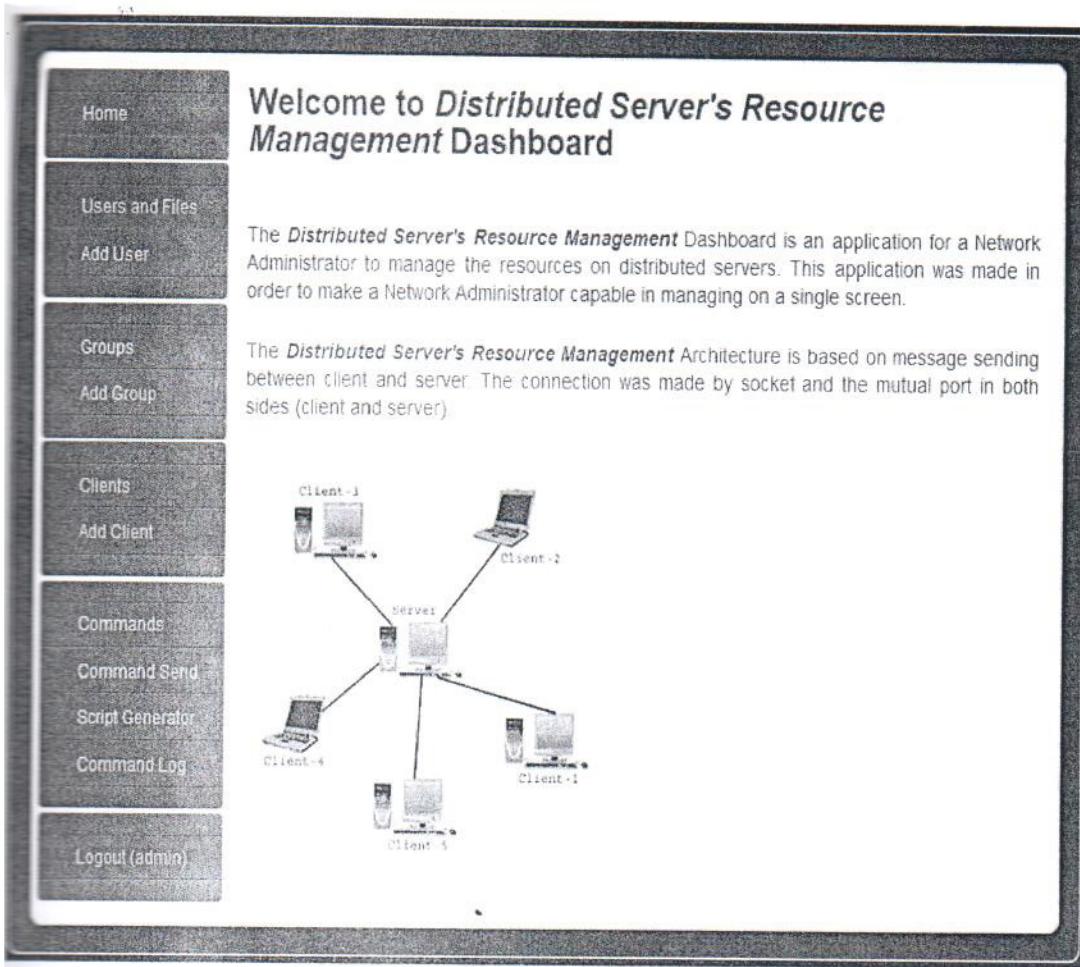
- Relasi Antara clients dan groups_clients, pada relasi ini groups_clients membutuhkan primary key dari clients untuk menentukan grup mana saja yang ada pada sebuah *client*.
- Relasi Antara groups dan groups_clients, pada relasi ini groups_clients membutuhkan primary key dari groups untuk menentukan *client* mana saja yang ada pada sebuah grup.
- Relasi Antara groups dan users_clients_groups, pada relasi ini users_clients_groups membutuhkan primary key dari groups untuk menentukan grup yang ada pada sebuah *client* pada sebuah user.
- Relasi Antara users_clients dan users_clients_groups, pada relasi ini users_clients_groups membutuhkan primary key dari users_clients untuk menyimpan grup yang dimiliki sebuah user pada sebuah *client*.
- Relasi antara clients dan command_send, pada relasi ini command_send membutuhkan *primary key* dari clients untuk menentukan *client* yang mengirim *command*.
- Relasi antara command_send dan commands, pada relasi ini command_send membutuhkan *primary key* dari commands untuk dapat mengirim perintah kepada *agent*.

4.7 Implementasi Antarmuka Sistem

Pada tugas akhir ini, Antarmuka sistem dibangun dengan *framework* yii. Antarmuka sistem berupa *dashboard*. Adapun antarmuka-antarmuka yang diimplementasikan pada tugas akhir ini adalah sebagai berikut.

4.7.1 Tampilan Menu Utama

Pada Gambar 19 ini ditampilkan beranda yang dapat dilihat oleh *guest* dan *administrator*.



Gambar 19 Tampilan Menu Utama

Antarmuka utama dari sistem terdiri dari 11 menu yang fungsinya adalah sebagai berikut:

1. Users

Menu-Menu dapat dilihat ketika login sebagai Administrator, setelah autentikasi berhasil maka Menu user dapat diakses oleh Administrator untuk melihat, mengubah, atau menghapus user yang ada sesuai dengan keinginan Administrator.

2. Add User

Menu Add User berfungsi untuk menambahkan *user* baru kedalam sistem yang disimpan pada database yang telah disediakan.

3. Groups

Menu Group dapat diakses oleh Administrator untuk melihat, mengubah, atau menghapus *group* yang ada sesuai dengan keinginan Administrator.

4. Add Group

Menu Add Group berfungsi untuk menambahkan *group* baru kedalam sistem yang disimpan pada database yang telah disediakan.

5. Clients

Menu Clients berfungsi untuk menampilkan, mengubah ataupun menghapus *client* yang terdapat pada sistem dan yang telah tersimpan pada database.

6. Add Client

Menu Add Client berfungsi untuk menambahkan *client* pada database

7. Commands

Menu Command berfungsi untuk menampilkan *list-list* dari perintah yang telah diciptakan.

8. Command Send

Menu Command Send berfungsi untuk mengirimkan perintah.

9. Script Generator

berfungsi untuk menciptakan *script* baru yang dapat dijalankan pada sistem untuk windows maupun linux.

10. Command log

Menu Command log berfungsi untuk menampilkan perintah yang telah dilakukan di sistem.

4.7.2 Tampilan Menu Users

Menu users memiliki pengaturan dengan menggunakan tiga tombol utama, yaitu “view detail [P]”, “edit [E]”, dan “delete [*]”.

The screenshot shows a software interface titled "Users and Files Management". On the left is a vertical sidebar with navigation links: Home, Users and Files (selected), Add User, Groups, Add Group, Clients, Add Client, Commands, Command Send, Script Generator, Command Log, and Logout (admin). The main area is titled "Users and Files Management" and displays a table of user information. The table has columns: User Name, Client(s), Group, Status, Single Operation, and Broadcast Operation. There are two rows of data:

| User Name | Client(s) | Group | Status | Single Operation | Broadcast Operation |
|-----------|----------------|--------------------|--------------------|------------------|---------------------|
| kamu | [Local-VMWare] | [other] | active [change] | P E * | P E * |
| toqar | [Local-VMWare] | [NM][E] [other] | active [change] | P E * | P E * |

Below the table are two sections of operation options:

- Single Operation Options:**
 - P View User Files on selected client
 - E Update User on selected client
 - * Delete User on selected client
- Broadcast Operation Options:**
 - P View User Detail on all related clients
 - E Update User on all related clients
 - * Delete User on all related clients

Gambar 20 Tampilan Menu User

Keterangan Gambar 20 adalah sebagai berikut:

- Tombol “view detail” akan menampilkan detail dari *user* yang dipilih.
- Tombol “edit” akan menampilkan *form* dari *user* yang dipilih untuk diubah isinya
- Tombol “delete” akan menghapus *user* yang dipilih dari database.

4.7.3 Tampilan Menu Add User

Menu Add User terdiri atas 4 field yang harus diisi, setelah diisi maka tombol *create* berfungsi untuk memasukkan *field* tersebut pada database yang telah ditentukan.

The screenshot shows a web-based application interface for adding a new user. On the left is a vertical sidebar menu with the following items:

- Home
- Users and Files
 - Add User
- Groups
 - Add Group
- Clients
 - Add Client
- Commands
 - Command Send
 - Script Generator
 - Command Log
- Logout (admin)

The main content area is titled "Add User" and contains the following fields:

- User Name * (text input field)
- Client * (checkboxes: Win-Server-08, CentOS_6.4(1), CentOS_6.4(2), Local-VMWare)
- Group * (checkboxes: NM, EI, other)
- User Description * (text area)
- Create (button)

A note at the top states: "Fields with * are required".

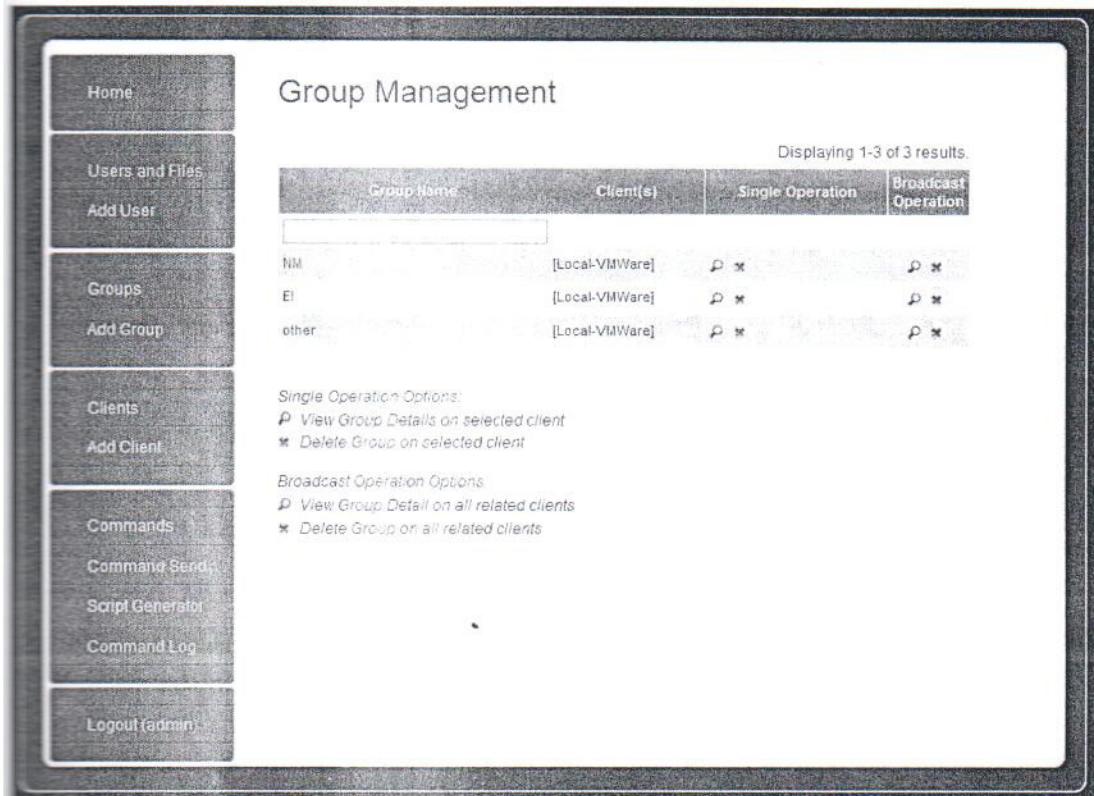
Gambar 21 Tampilan Menu Add User

Keterangan Gambar 21 adalah sebagai berikut:

- *Field* pertama yaitu username yang akan diisi dengan *username* yang telah ditentukan oleh Administrator.
- *Field* kedua yaitu Location
- *Field* ketiga yaitu group yang berisi *combo box* dengan *group* yang telah tersedia pada database.
- *Field* keempat yaitu user description yang berisi deskripsi dari *user*

4.7.4 Tampilan Menu Groups

Menu groups memiliki kesamaan dengan menu users, yaitu mempunyai tiga tombol dengan masing masing fungsi; “view detail”, “edit”, dan “delete” group.



Gambar 22 Tampilan Menu Group

Keterangan Gambar 22 adalah sebagai berikut:

- Tombol “view detail” akan menampilkan *detail* dari *group* yang dipilih.
- Tombol “edit” akan menampilkan *form* dari *group* yang dipilih untuk diubah isinya
- Tombol “delete” akan menghapus *group* yang dipilih dari database.

4.7.5 Tampilan Menu Add Group

Menu add group terdapat dua *field* yang harus diisi. Untuk keterangan lebih lanjut lihat keterangan Gambar 23.

The screenshot shows a web-based administrative interface. On the left is a vertical sidebar menu with the following items:

- Home
- Users and Files
 - Add User
- Groups
 - Add Group
- Clients
 - Add Client
- Commands
 - Command Send
 - Script Generator
 - Command Log
- Logout (admin)

The main content area is titled "Add Group". It contains the following fields:

- A note: "Fields with * are required."
- "Group Name *": An input field with a placeholder value.
- "Client *": A section containing four checkboxes, each corresponding to an IP address: 172.29.4.4, 172.29.5.5, 172.29.6.6, and 192.168.159.141.
- "Description *": A large text input area.
- A "Create" button at the bottom.

Gambar 23 Tampilan Menu Add Group

Keterangan Gambar 23 adalah sebagai berikut:

- *Field* pertama adalah Group Name. Group Name ditentukan oleh Administrator.
- *Field* kedua adalah Group Description. Group Description berisi deskripsi daripada group tersebut.

4.7.6 Tampilan Menu Clients

Menu Clients memiliki kesamaan dengan Menu users, yaitu mempunyai tiga tombol dengan masing masing fungsi; “view detail”, “edit”, dan “delete” client.

| Address | Server Name | Platform Type | Status | Operation |
|-----------------|----------------|---------------|----------|-----------|
| 172.29.4.4 | Win-Server-08 | Windows | [online] | |
| 172.29.5.5 | CentOS_6.4(1) | Linux | [online] | |
| 172.29.6.6 | CentOS_6.4(2) | Linux | [online] | |
| 192.168.159.141 | Local-VNIVWare | Linux | [online] | |

Gambar 24 Tampilan Menu Clients

Keterangan Gambar 24 adalah sebagai berikut:

- Tombol “view detail” akan menampilkan detail dari *Client* yang dipilih.
- Tombol “edit” akan menampilkan *form* dari *Client* yang dipilih untuk diubah isinya.
- Tombol “delete” akan menghapus *Client* yang dipilih dari database.

4.7.7 Tampilan Menu Add Client

Menu Add Client memiliki 3 *field* yang harus diisi, *field* itu antara lain adalah; “Address”, “Type”, dan “Description”.

The screenshot shows a web-based application interface. On the left is a vertical sidebar menu with the following items:

- Home
- Users and Files
 - Add User
- Groups
 - Add Group
- Clients
 - Add Client
- Commands
 - Command Send
 - Script Generator
 - Command Log
- Logout (admin)

The main content area is titled "Add Client". It contains the following fields:

- A label "Address *" followed by an input field.
- A label "Type *" followed by a dropdown menu showing "Linux".
- A label "Server Name" followed by a large text input field.
- A "Create" button at the bottom.

A note at the top of the form states: "Fields with * are required."

Gambar 25 Tampilan Menu Add Client

Keterangan Gambar 25 adalah sebagai berikut:

- *Field* Pertama adalah “Address”, yang akan diisi berupa alamat ip yang akan dituju untuk dikirimkan suatu perintah.
- *Field* Kedua adalah “Type”, yang akan diisi berupa Jenis Operasi sistem yang akan digunakan.
- *Field* Ketiga adalah “Description”, yang akan diisi dengan deskripsi yang telah ditentukan oleh Adminstrator.

4.7.8 Tampilan Menu Command

Menu Command Mempunyai 4 tabel, antara lain; tabel Manager Command, Linux Command, Windows Command, dan Description.

The screenshot shows a web-based application interface titled "Command List". On the left, there is a vertical sidebar with navigation links: Home, Users and Files (Add User), Groups (Add Group), Clients (Add Client), Commands (Command Send, Script Generator, Command Log), and Logout (admin). The main area is titled "Command List" and displays a table with 30 results. The table has columns: Manager Command, Linux Command, Windows Command, Description, and Status. The first few rows of data are as follows:

| Manager Command | Linux Command | Windows Command | Description | Status |
|-----------------|--|-----------------|--|--------|
| AddUser | AddUser | | useradd in linux | P F * |
| DeleteUser | DeleteUser | | menghapus user di linux | P F * |
| ListUser | cat /etc/passwd grep /bin/bash cut -d: -f1 | | Print all users name | P F * |
| AddGroup | AddGroup | | adding group to linux | P F * |
| ListFiles | find {location} -follow | | List files, folders, subfolders of user's home directory | P F * |
| GenerateScript | java ScriptGenerator Script_name \$linux_command | | Generate new script in each clients | P F * |
| ChangeUserGroup | usermod -G \$2 -d /home/\$2/\$1 \$1 | | change the user's group | P F * |
| RenameUser | linux rename | windows rename | change username | P F * |
| SetPassword | echo -e '\$1\n\$1' (passwd --stdin \$1) | | reset user's password with its own username | P F * |
| DiskUsage | du -sh \$1 | | Disk Usage of a resource | P F * |

Gambar 26 Tampilan Menu Command

4.7.9 Tampilan Menu Command Send

Menu Command Send terdiri dari 2 *field*.

The screenshot shows a web-based application interface titled "Send a Command to a Client". On the left, there is a vertical sidebar menu with the following items:

- Home
- Users and Files
 - Add User
- Groups
 - Add Group
- Clients
 - Add Client
- Commands
 - Command Send
 - Script Generator
 - Command Log
- Logout (admin)

The main content area has the following fields:

- Fields with * are required**
- Command ***: A dropdown menu showing "AddUser".
- Arguments**: An empty text input field.
- Client ***: A dropdown menu showing "172.29.4.4".
- Create**: A button at the bottom right of the form.

Gambar 27 Tampilan Menu Command Send

Keterangan Gambar 27 adalah sebagai berikut:

- *Field* pertama yaitu Command, yang berisi perintah yang akan di kirim.
- *Field* kedua adalah argument, yang berisi keterangan dari perintah yang akan dikirim.

4.7.10 Tampilan Menu Script Generator

Menu Script Generator mempunyai 4 *field* yang harus diisi, antara lain; “Script Name”, “Linux Command”, “Windows Command”, dan “Description”.

The screenshot shows a web-based application titled "Script Generator". On the left is a vertical sidebar menu with the following items:

- Home
- Users and Files
 - Add User
- Groups
 - Add Group
- Clients
 - Add Client
- Commands
 - Command Send
 - Script Generator
 - Command Log
- Logout (Admin)

The main content area is titled "Script Generator" and contains the following fields:

- Script Name**: A text input field with the placeholder "(Without file extension)".
- Linux Command ***: A text input field with the placeholder "Write arguments with \$(number). Ex: useradd \$1 -G \$2".
- Windows Command ***: A text input field with the placeholder "Write arguments with %%(alphabets). Ex: dsadd user *cn=%%%a...".
- Description**: A text input field.

At the bottom center is a "Create" button.

Gambar 28 Tampilan Menu Script Generator

Keterangan Gambar 28 adalah sebagai berikut:

- *Field* Pertama adalah Script Name yang berisi nama daripada *script* yang akan di buat.
- *Field* Kedua adalah Linux Command yang berisi perintah yang akan dibuat untuk sistem operasi linux.
- *Field* Ketiga adalah Windows Command yang berisi perintah yang akan dibuat untuk sistem operasi windows server.
- *Field* Keempat adalah Description yang berisi deskripsi dari suatu *script*.

4.7.11 Tampilan Menu Command Log

Menu Command Log terdiri dari 6 tabel, yaitu tabel ID, Manager Command, Arguments, Address, Reply, dan Status.

The screenshot shows a web-based application interface titled "Command Log". On the left, there is a vertical sidebar menu with the following items: Home, Users and Files (with Add User), Groups (with Add Group), Clients (with Add Client), Commands (with Command Script, Script Generator, Command Log), and Logout Admin. The main area is titled "Command Log" and displays a table of log entries. The table has columns: ID#, Manager Command, Arguments, Address, Reply, and Status. The table shows 10 results out of 1018. The log entries are as follows:

| ID# | Manager Command | Arguments | Address | Reply | Status |
|------|-----------------|--------------|------------|---|--------|
| 1709 | AddGroup | NM | 172.29.4.4 | null\$dsadd succeeded: cn=NM,dc=server,dc=del,dc=ac,dc=id processed file: C:\Users\NM Successfully processed 1 files; Failed processing 0 files "group NM has been added" | 1 |
| 1710 | AddGroup | NM | 172.29.5.5 | null\$NMx:501: Group NM has been added | 1 |
| 1711 | AddGroup | NM | 172.29.6.6 | null\$NMx:501: Group NM has been added | 1 |
| 1712 | DeleteGroup | NM | 172.29.4.4 | dsm succeeded: cn=NM,dc=server,dc=del,dc=ac,dc=id "group NM has been deleted" | 1 |
| 1713 | DeleteGroup | NM | 172.29.6.6 | NMx:501: Group NM has been deleted | 1 |
| 1714 | DeleteGroup | NM | 172.29.5.5 | NMx:501: Group NM has been deleted | 1 |
| 1715 | AddGroup | NM | 172.29.4.4 | dsadd succeeded: cn=NM,dc=server,dc=del,dc=ac,dc=id processed file: C:\Users\NM Successfully processed 1 files; Failed processing 0 files "group NM has been added" | 1 |
| 1716 | AddGroup | NM | 172.29.5.5 | NMx:501: Group NM has been added | 1 |
| 1717 | AddGroup | NM | 172.29.6.6 | NMx:501: Group NM has been added | 1 |
| 1718 | AddUser | christine NM | 172.29.4.4 | | 1 |

Go to page: < Previous [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] Next >

Gambar 29 Tampilan Menu Command Log

4.8 Instalasi dan Konfigurasi

Bagian ini menjelaskan bagaimana aplikasi serta *script* yang telah dibangun ditanamkan pada *server* dan *client* yang berperan dalam berjalannya sistem. *Server* berjalan pada lingkungan operasi Linux CentOS 6.4 64bit. *Client* berjalan pada lingkungan operasi Windows Server 2008 64bit dan Linux CentOS 6.4 64bit.

4.8.1 Instalasi dan Konfigurasi pada Server (Linux)

Komponen-komponen yang harus ditanam pada Server yang berjalan pada lingkungan operasi Linux adalah *Web Service*, *PHPMyAdmin*, *ServerManager*, dan Aplikasi Web.

4.8.1.1 Web Service dan PHPMyAdmin

Untuk mengaktifkan aplikasi web pada *server*, diperlukan Apache 2.0 Web Service dan PHPMyAdmin untuk basis data mysql. Kedua komponen tersebut tersedia dalam LAMPP (XAMPP for Linux). LAMPP adalah aplikasi *open source* sehingga dapat diunduh secara bebas.

1. Mengunduh dan menginstal LAMPP.

```
./xampp-linux-x64-1.8.3-4-installer.run
```

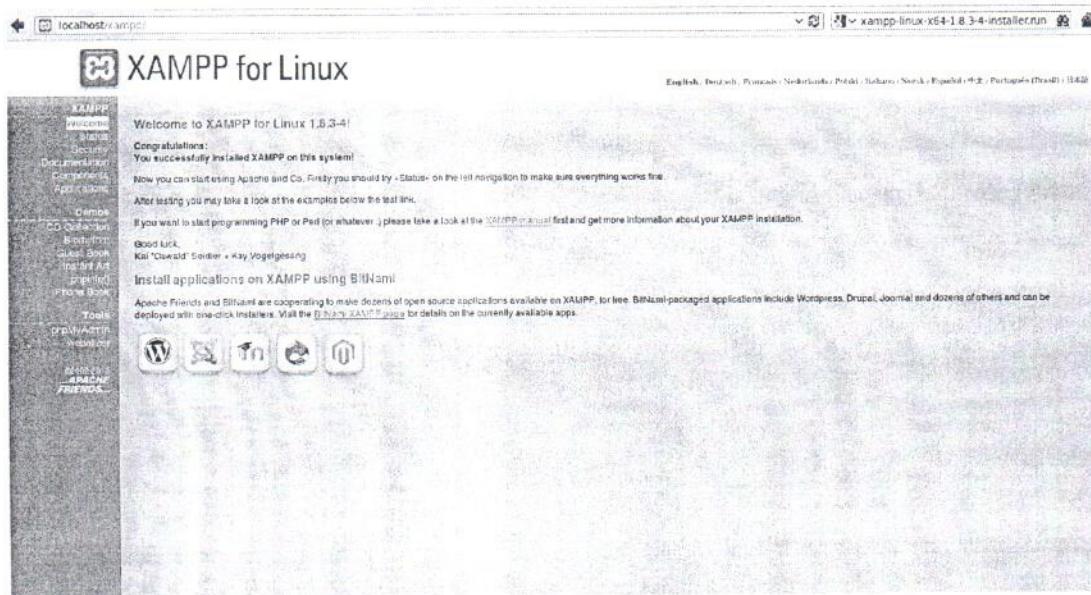
Gambar 30 Menginstal LAMPP

2. Memastikan *web service* lain seperti HTTPD dalam keadaan mati dan mengaktifkan *service* LAMPP. Pengaktifan *service* akan mengaktifkan *Web Service* dan PHPMyAdmin secara bersamaan.

```
service httpd stop  
/opt/lampp/lampp start
```

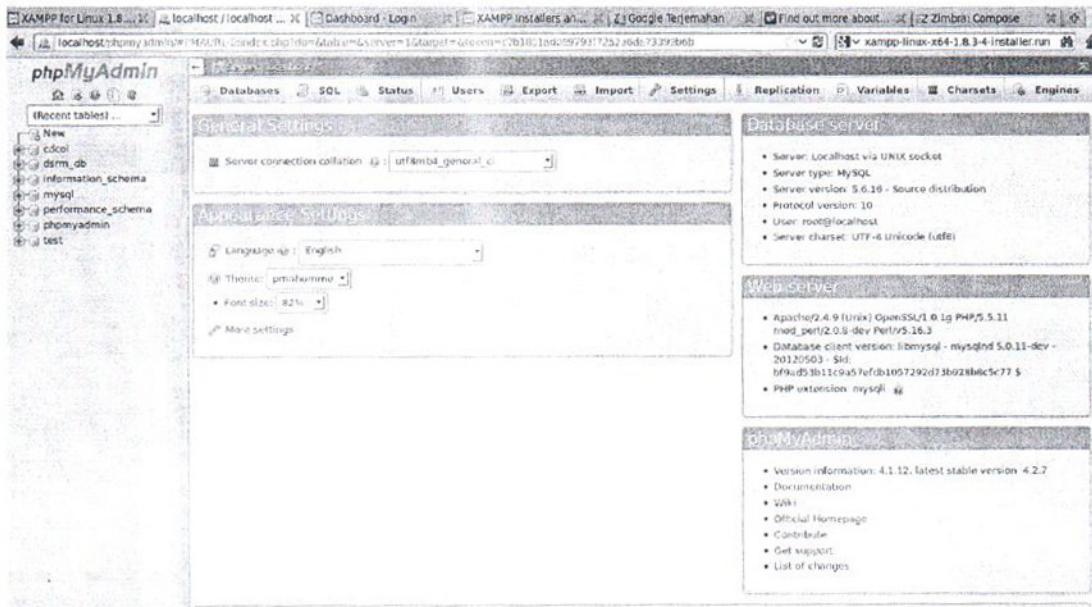
Gambar 31 Menonaktifkan HTTPD dan mengaktifkan LAMPP

3. Mengecek apakah *Web Service* telah aktif dengan mengakses <http://localhost/xampp/>.



Gambar 32 Web Service aktif

4. Mengecek apakah PHPMyAdmin telah aktif dengan mengakses <http://localhost/phpmyadmin/>.



Gambar 33 PHPMyAdmin aktif

4.8.1.2 ServerManager Service

ServerManager Service adalah *service* yang berisi:

1. MySQLJavaConnector Class

Class yang menghubungkan *record* yang masuk di basis data dengan *ServerManager*. MySQLJavaConnector Class menghubungkan ServerManager Class dengan tabel “command_send” pada basis data “dsrm_db” melalui *driver* “com.microsoft.sqlserver.jdbc.SQLServerDriver”.

```
/*
public static void main(String[] args) throws InstantiationException, IllegalAccessException
{
    int stat = 1;
    Connection con = null;
    Statement stmtStatus = null;
    Statement stmtClients = null;
    Statement stmtCommands = null;
    ResultSet rs = null;
    ResultSet rs2 = null;
    ResultSet rs3 = null;
    String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    String url = "jdbc:mysql://localhost/";
    String dbname = "dsrm_db";
    String sqluser = "root";
    String sqlpass = "";
    String query = "SELECT * FROM command_send";
```

Gambar 34 Struktur MySQLJavaConnector

MySQLJavaConnector Class mengirimkan pesan dari basis data ke ServerManager Class dengan menggunakan fungsi “sendCommand”. Setelah mengirim pesan, MySQLJavaConnector Class menunggu dan mengambil hasil berupa respon dari *client* menggunakan fungsi “getReply”.

```
serverManager[rs2.getInt("client_id")].sendCommand();

stmtStatus.executeUpdate("UPDATE command_send SET status = 1, reply = '" +
    serverManager[rs2.getInt("client_id")].getReply() +
    "' where send_id = "+ rs.getInt("send_id") +"");
serverManager[rs2.getInt("client_id")].setReplyNull();
// query = "UPDATE command_send SET status = '1' WHERE id = '" + id + "'";

// stmt.executeQuery(query);
System.out.println("\nScript berhasil dijalankan\n");
```

Gambar 35 Pengiriman dan pengambilan pesan MySQLJavaConnector

2. ServerManager Class

Class yang mengirimkan pesan kepada *client* yang diterima dari MySQLJavaConnector Class. Alamat *client* disimpan pada variabel “clientName”, port pada variabel “port” (diambil dari ServerProperties), dan pesan pada variabel “command”.

```
public class ServerManager {

    String clientName;
    int port;
    String command;
    String reply;

    public ServerManager() throws IOException {
        Properties prop = new Properties();
        String propFileName = "ServerProperties.properties";

        InputStream inputStream = getClass().getClassLoader()
            .getResourceAsStream(propFileName);
        prop.load(inputStream);
        if (inputStream == null) {
            throw new FileNotFoundException("property file '" +
                propFileName + "' not found in the classpath");
        }

        port = Integer.parseInt(prop.getProperty("port", "6066"));
    }
}
```

Gambar 36 Struktur ServerManager

ServerManager Class mengirim pesan kepada client menggunakan socket. Pesan dibawa melalui “*DataOutputStream*” dan diterima melalui “*DataInputStream*”.

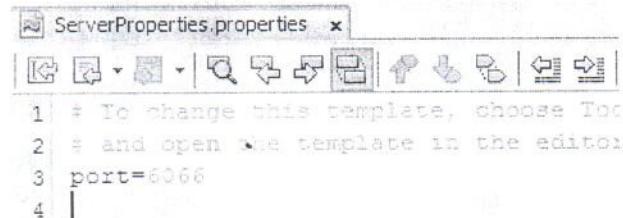
```
Socket client = new Socket(clientName, port);
if (!client.isClosed()) {

    DataOutputStream out = new DataOutputStream(client.getOutputStream());
    out.writeUTF(command);
    DataInputStream in = new DataInputStream(client.getInputStream());
    reply = in.readUTF();
} else {
    reply = "client offline";
}
client.close();
```

Gambar 37 Pengiriman dan pengambilan pesan *ServerManager*

3. ServerProperties

ServerProperties adalah *file* yang menyimpan konfigurasi yang dapat diubah sesuai dengan kebutuhan pengguna pada *Server*. Informasi yang disimpan pada *ServerProperties* adalah variabel yang akan dibaca oleh *class* yang dibangun oleh Java. Dalam hal ini, disimpan informasi tentang port yang digunakan.



Gambar 38 ServerProperties

Langkah-langkah instalasi *ServerManager* adalah sebagai berikut:

1. Mengkompilasi *class* yang dibutuhkan ke dalam *file* DSRM.jar dan meletakkannya ke dalam satu direktori yang ditetapkan.

```
/home/ta2/servermanager/DSRM.jar
```

Gambar 39 DRS.M.jar pada suatu direktori

2. Membuat *Java Daemon Script* untuk mengakses DSRM.jar sebagai *service*. Pastikan variabel mengarah ke bagian-bagian aplikasi yang dibutuhkan.

```
JAVA_HOME=/usr/java/latest  
serviceNameLo="servermanager"  
serviceName="ServerManager"  
serviceUser="root"  
serviceGroup="root"  
applDir="/home/ta2/$serviceNameLo"  
serviceUserHome="/home/$serviceUser"  
serviceLogFile="/var/log/$serviceNameLo.log"  
maxShutdownTime=15  
pidFile="/var/run/$serviceNameLo.pid"  
javaCommand="java"  
javaExe="$JAVA_HOME/bin/$javaCommand"  
javaArgs="-jar $applDir/DSRM.jar"  
javaCommandLine="$javaExe $javaArgs"  
javaCommandLineKeyword="DSRM.jar"  
hers
```

Gambar 40 Konfigurasi Java Daemon Script untuk ServerManager

3. Mengaktifkan *ServerManager Service*.

```
[root@main init.d]# service servermanager start  
Starting ServerManager started PID=7018
```

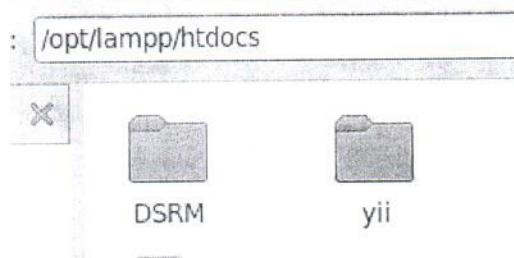
Gambar 41 Mengaktifkan *ServerManager Service*

8.1.3 Aplikasi Web

Antarmuka Aplikasi adalah aplikasi web yang menggunakan *Framework* Yii PHP. Aplikasi web disimpan dalam direktori DSRM.

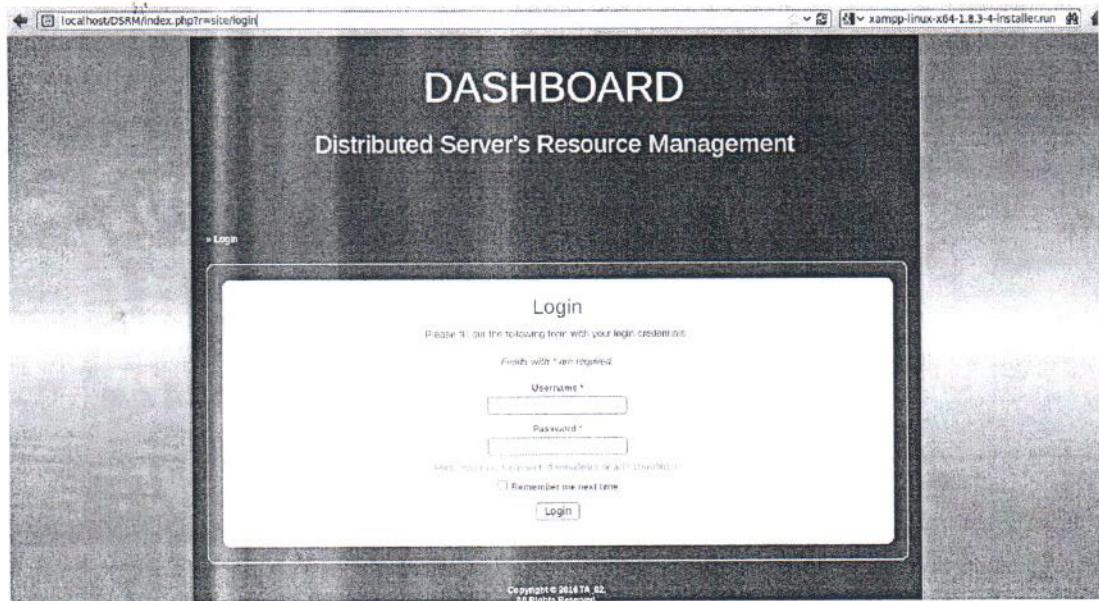
Langkah-langkah instalasi Aplikasi Web adalah sebagai berikut:

1. Memindahkan direktori dan isi aplikasi web bersama dengan *Framework* Yii yang digunakan ke dalam direktori “htdocs” pada LAMPP.



Gambar 42 Direktori aplikasi berada dalam direktori "htdocs" pada LAMPP

2. Aplikasi Web telah siap digunakan. Untuk memastikan cek pada <http://localhost/DSRM/>.



Gambar 43 Aplikasi Web telah siap digunakan

4.8.2 Instalasi dan Konfigurasi pada *Client Windows*

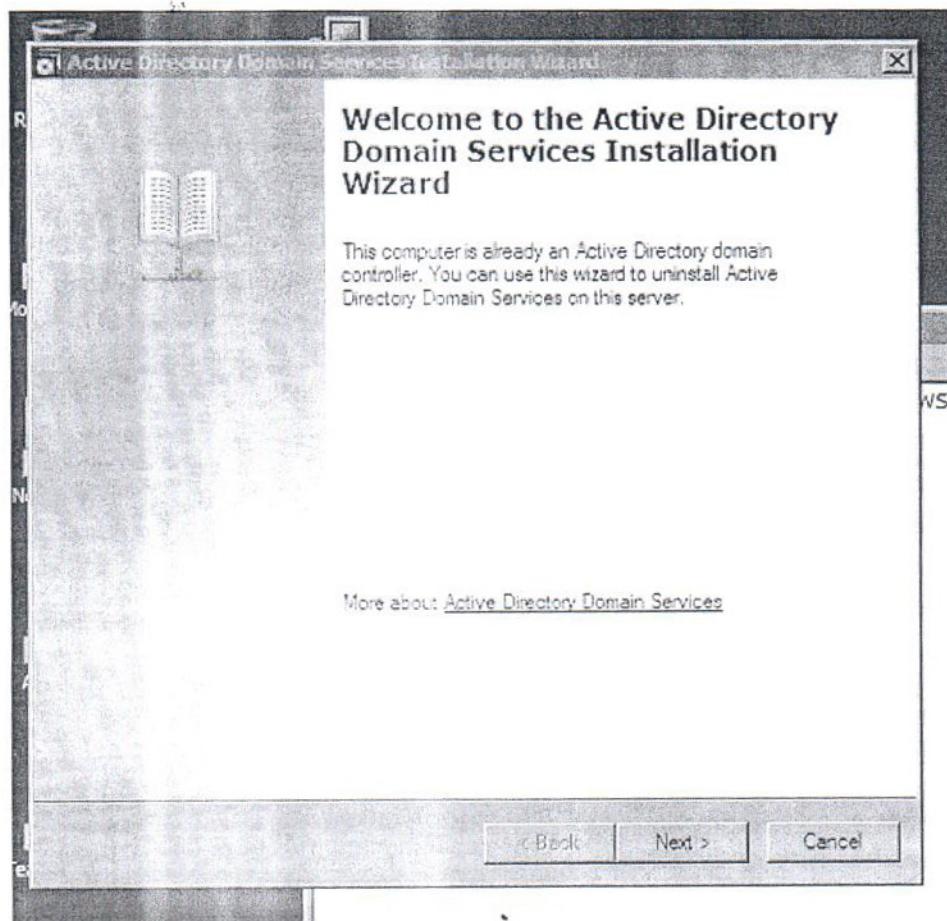
Komponen-komponen yang harus ditanam pada *Client* yang berjalan pada lingkungan operasi Windows adalah *Active Directory* dan *WindowsAgent Service*.

4.8.2.1 *Active Directory*

Sebelum memulai konfigurasi, pastikan *Windows Firewall* dalam keadaan mati, untuk menghindari *block access*.

Langkah-langkah instalasi *Active Directory* adalah sebagai berikut:

1. Ciptakan *Active Directory* dengan “dcromo” pada Windows Server dan ikuti langkah-langkahnya. Sebagai catatan, “dcromo” diketik setelah memilih start dan mengklik run. Ikuti langkah instalasi “dcromo” sesuai dengan kebutuhan *client*.



Gambar 44 DCPromo

4.8.2.2 WindowsAgent Service

WindowsAgent Service adalah *service* yang berisi:

1. WindowsAgent Class

WindowsAgent Class menghubungkan koneksi utama dengan ServerManager melalui socket. Agar dapat terhubung dengan ServerManager, maka port yang digunakan juga diharuskan sama. Nomor port dapat diatur pada AgentProperties yang menyimpan informasi port yang dipakai.

```
public WindowsAgent() throws IOException
{
    Properties prop = new Properties();
    String propFileName = "AgentProperties.properties";

    InputStream inputStream = getClass().getClassLoader()
        .getResourceAsStream(propFileName);
    prop.load(inputStream);
    if (inputStream == null) {
        throw new FileNotFoundException("property file '" +
            propFileName + "' not found in the classpath");
    }

    int port = Integer.parseInt(prop.getProperty("port", "6066"));

    serverSocket = new ServerSocket(port);
    serverSocket.setSoTimeout(60000);
}
```

Gambar 45 Struktur WindowsAgent

Pesan yang masuk dipisahkan dan dimasukkan ke dalam variabel yang berbeda-beda. *Command* yang dijalankan dibawa melalui “theCommand”, argumen melalui “theArguments”, nama *file* output melalui “theFile”, penghitung melalui “tempN”, dan status argumen melalui “withArguments”.

```
String theCommand;
String theArguments;
String thefile;
int tempN=0;
int withArguments=0;
```

Gambar 46 Variabel pada WindowsAgent

Untuk pembuatan suatu file yang berisi perintah agar dijalankan di windows, maka diperlukan proses pembuatan *file* yang sesuai untuk dijalankan oleh Batch *Scripts*.

```

if (withArguments == 1)
{
    theCommand = incomingMessage.substring(0,tempN);
    theArguments = incomingMessage.substring(tempN+1);

String random = generateRandom();

    theFile = theCommand + random + ".txt";

    FileWriter fileWriter = new FileWriter(theFile);

    BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

    bufferedWriter.write(theArguments);

    PrintWriter writer = new PrintWriter(theCommand + ".txt", "UTF-8");
writer.println(theArguments);
writer.close();

    bufferedWriter.newLine();
    bufferedWriter.flush();

    proc = rt.exec("cmd /C " + theCommand + ".bat " + theFile);

    bufferedWriter.close();
    File file = new File(theFile);
    file.delete();
}

```

Gambar 47 Pembuatan file oleh WindowsAgent

Setelah diproses oleh Batch *Scripts*, maka pesan balasan pun disimpan untuk dikirim kembali ke ServerManager.

```

BufferedReader br = new BufferedReader(new InputStreamReader(proc.getInputStream()));

String line = null;
while ((line = br.readLine()) != null)
{
    System.out.println(line);

    outgoingMessage = outgoingMessage + line + "\n";
}

int exitVal = proc.waitFor();

DataOutputStream out = new DataOutputStream(server.getOutputStream());

out.writeUTF(outgoingMessage);

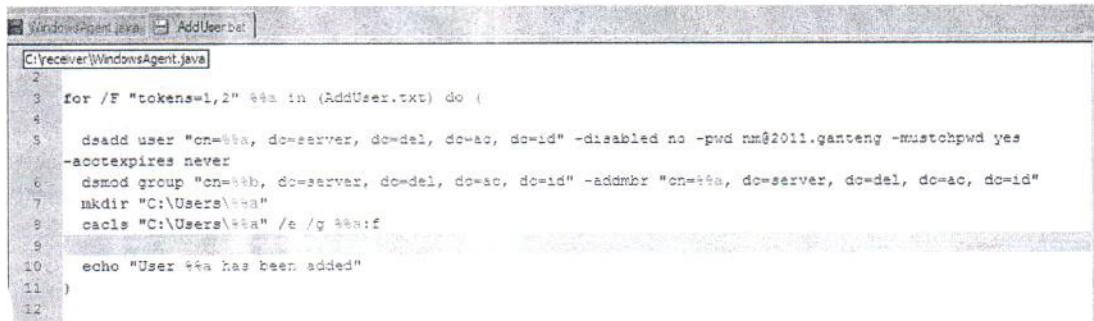
incomingMessage = "";
outgoingMessage = "";
server.close();

```

Gambar 48 Pengambilan dan pengiriman pesan oleh WindowsAgent

2. Batch *Scripts*

Batch Scripts adalah kumpulan *script* yang dimiliki oleh *WindowsAgent Service* yang menjalankan perintah Windows sesuai dengan pesan yang dikirim dan diproses pada *WindowsAgent Class*. Batch Scripts dikenali pada file berekstensi “.bat”.



```

WindowsAgent.java [AddUser.bat]
C:\receiver\WindowsAgent.java
2
3 for /F "tokens=1,2" %%a in (AddUser.txt) do (
4
5     dsadd user "cn=%%a, dc=server, dc=del, dc=id" -disabled no -pwd nm@2011.ganteng -mustchpwd yes
6     -acctexpires never
7     dsmod group "cn=%%b, dc=server, dc=del, dc=id" -addmbr "cn=%%a, dc=server, dc=del, dc=id"
8     mkdir "C:\Users\%%a"
9     cacls "C:\Users\%%a" /e /g %%a:f
10
11 echo "User %%a has been added"
12
13

```

Gambar 49 Batch Script berekstensi ".bat"

3. ScriptGenerator Class

ScriptGenerator Class adalah *class* yang dirancang untuk menghasilkan sebuah *script* baru yang dapat digunakan untuk menjalankan perintah-perintah baru yang belum ada pada *client*. *ScriptGenerator Class* pada *WindowsAgent Service* menghasilkan Batch Script (file berekstensi “.bat”).

```

lic class ScriptGenerator

    public static void main(String[] args)
    {
        try
        {
            FileWriter fileWriter = new FileWriter(args[0]);

            BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

            bufferedWriter.write("Echo off");

            bufferedWriter.newLine();bufferedWriter.newLine();
            bufferedWriter.write("for /F \"tokens=1,2,3,4,5,6,7\" %%a in ("+
                args[0].substring(0, (args[0].length())-4) +".txt) do (");
            bufferedWriter.newLine();
            bufferedWriter.write(args[1].replace("[ (s)]"," ") .replace("[ (v)]","%%"));

            bufferedWriter.newLine();
            bufferedWriter.write(")");
            bufferedWriter.close();
        }
    }

```

Gambar 50 Penulisan file oleh ScriptGenerator pada Windows

4. Replacer Class

Replacer Class adalah *class* yang dirancang untuk mengubah karakter yang menimbulkan *error*. Misalnya, pada sebuah replacer *class* digunakan pengubahan

karakter “/” menjadi “\” akibat terjadinya kerumitan pembacaan karakter “\” pada saat pengiriman pesan dari basis data mysql.

```
BufferedReader bufferedReader = new BufferedReader(fileReader);

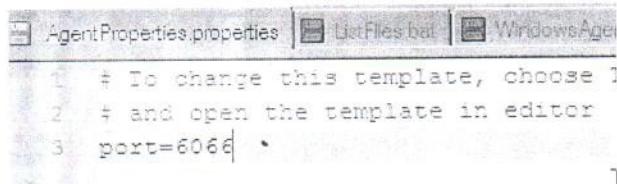
String line = "";

while ((line = bufferedReader.readLine()) != null)
{
    System.out.println(line.replace("\\\\","/"));
}
bufferedReader.close();
```

Gambar 51 Pengubahan karakter oleh Replacer Class

5. AgentProperties

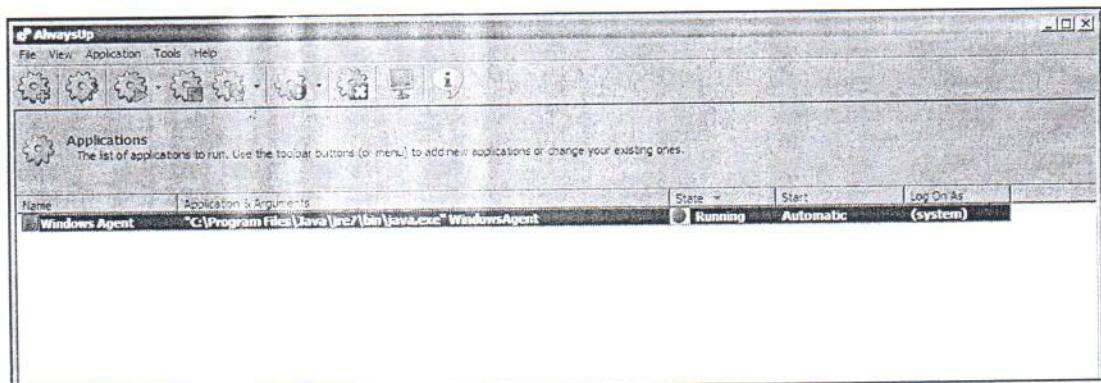
AgentProperties adalah *file* yang menyimpan konfigurasi yang dapat diubah sesuai dengan kebutuhan pengguna pada *Client*. Informasi yang disimpan pada AgentProperties adalah variabel yang akan dibaca oleh *class* yang dibangun oleh Java. Dalam hal ini, disimpan informasi tentang port yang digunakan.



Gambar 52 AgentProperties pada Windows

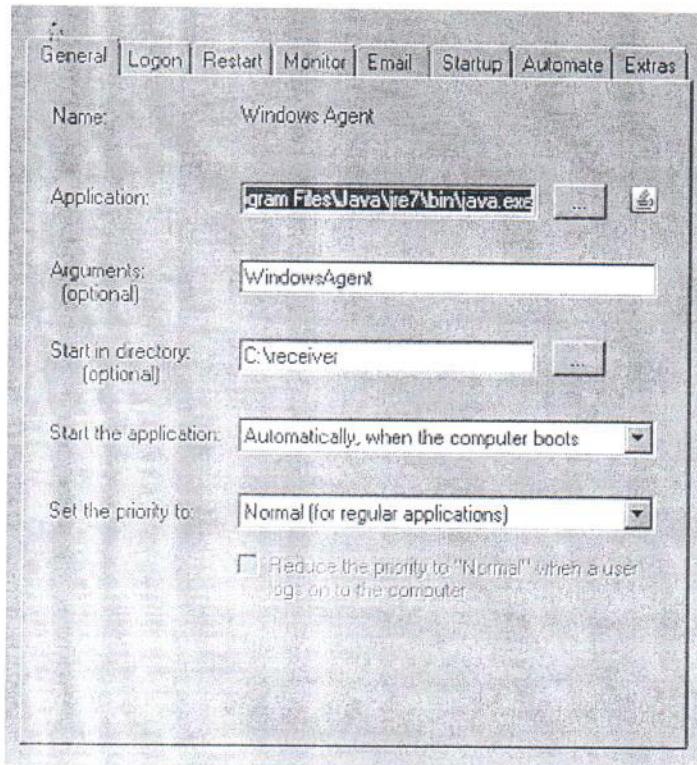
Langkah-langkah instalasi *WindowsAgent Service* adalah sebagai berikut:

1. Untuk membuat kumpulan *class* dan Batch Script tersebut menjadi *service* dapat digunakan perangkat lunak *open source*, AlwaysUp.



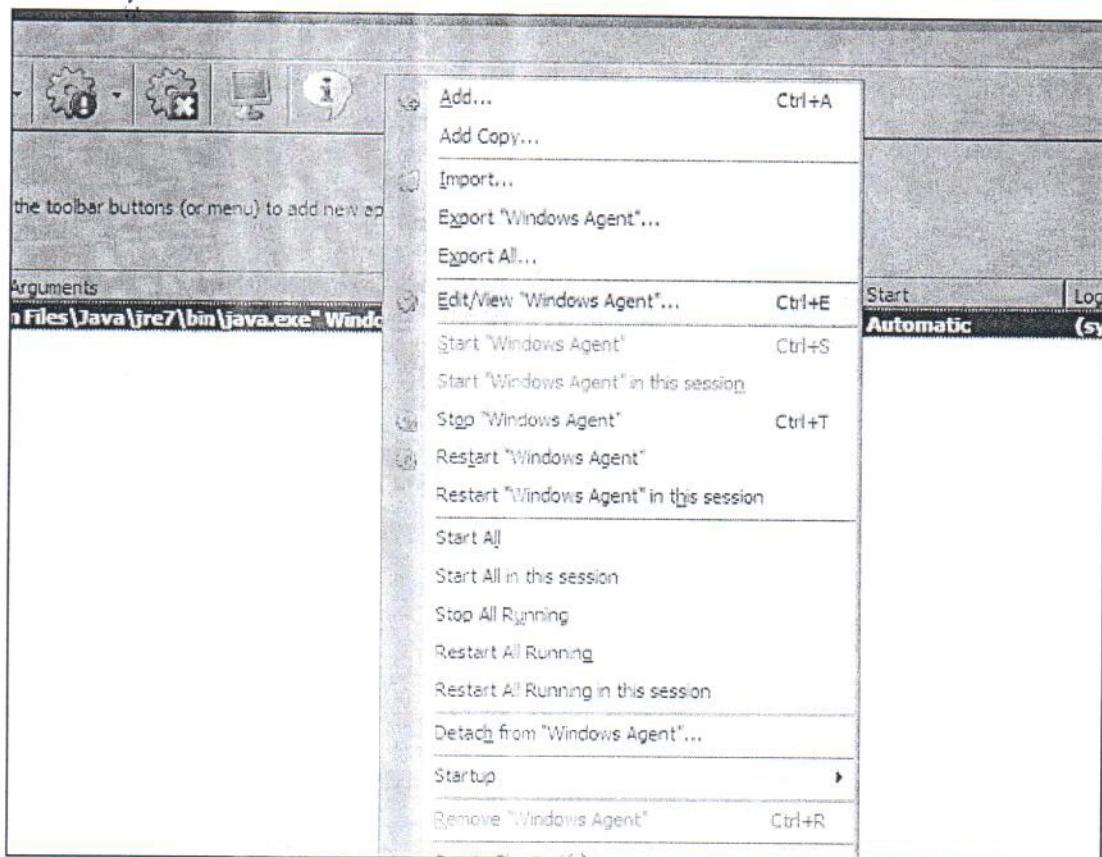
Gambar 53 AlwaysUp

2. Langkah awal pada AlwaysUp adalah menambah *service* baru. Pada Tab "General", isi field sesuai dengan *service* yang akan berjalan.



Gambar 54 General Tab pada AlwaysUp

3. Pada tampilan awal, terbentuk *service* yang telah dibuat. Untuk menjalankan *service*, *service* lalu pilih “Start Service”.



Gambar 55 Menjalankan Service pada AlwaysUp

4.8.3 Instalasi dan Konfigurasi pada Client Linux

Komponen yang harus ditanam pada *Client* yang berjalan pada lingkungan operasi Linux adalah *LinuxAgent Service*.

4.8.3.1 *LinuxAgent Service*

LinuxAgent Service adalah *service* yang berisi:

1. *LinuxAgent Class*

LinuxAgent Class memiliki struktur yang sama dengan *WindowsAgent Class*.

Penggunaan *socket* untuk komunikasi dengan *ServerManager*, penggunaan nomor port, dan penggunaan variabel sama dengan *WindowsAgent Class*. Perbedaan *LinuxAgent Class* hanya terdapat pada proses menjalankan perintah yang menggunakan *Shell Script*. *LinuxAgent Class* tidak menghasilkan *file* untuk

dibaca oleh *Shell Script*, melainkan memberikan argument secara langsung untuk dibaca dan dijalankan oleh *Shell Script*.

```
if (withArguments == 1)
{
    System.out.println("command dengan argumen");
    theCommand = incomingMessage.substring(0,tempN);
    theArguments = incomingMessage.substring(tempN+1);

    proc = rt.exec("./" + theCommand + ".sh " + theArguments);
}
else
{
    System.out.println("command tanpa argumen");
    proc = rt.exec("./" + incomingMessage + ".sh");
}
else
{
    System.out.println("not run as script");
    proc = rt.exec(incomingMessage);
}
```

Gambar 56 Pengiriman *command* dan argumen oleh *LinuxAgent*

2. Shell Scripts

Shell Scripts adalah kumpulan *script* yang dimiliki oleh *LinuxAgent Service* yang menjalankan perintah Linux sesuai dengan pesan yang dikirim dan diproses pada *LinuxAgent Class*. *Shell Scripts* dikenali pada *file* berekstensi “.sh”.

```
#!/bin/sh

if id -u $1 >/dev/null 2>&1; then
    echo "Cannot add user. User $1 already exist"
else
    useradd -l $1 -p $1 -G $2
    chown $1:$1 /home/$1

    if id -u $1 >/dev/null 2>&1; then
        echo "User $1 has been added"
    else
        echo "Add User failed"
    fi
fi
```

Gambar 57 Shell Script berekstensi ".sh"

3. ScriptGenerator Class

ScriptGenerator Class adalah *class* yang dirancang untuk menghasilkan sebuah *script* baru yang dapat digunakan untuk menjalankan perintah-perintah baru yang

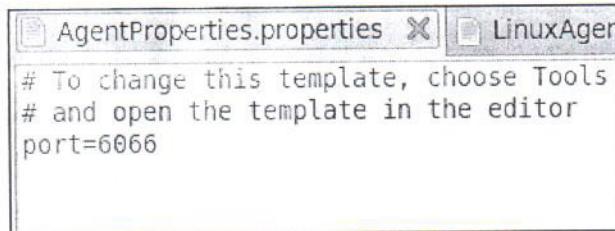
befum ada pada *client*. ScriptGenerator Class pada *LinuxAgent Service* menghasilkan Shell Script (*file* berekstensi “.sh”).

```
ScriptGenerator  
{  
    public static void main(String[] args)  
    {  
        try  
        {  
            FileWriter fileWriter = new FileWriter(args[0]);  
            BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  
            bufferedWriter.write("#!/bin/sh");  
            bufferedWriter.newLine();bufferedWriter.newLine();  
            bufferedWriter.write(args[1].replace("[($)]"," ").replace("[($)]","$"));  
            bufferedWriter.newLine();  
            bufferedWriter.close();  
            Process proc;  
            Runtime rt = Runtime.getRuntime();  
            proc = rt.exec("chmod 755 " + args[0]);  
        }  
    }  
}
```

Gambar 58 Penulisan *file* oleh ScriptGenerator pada Linux

4. AgentProperties

AgentProperties adalah *file* yang menyimpan konfigurasi yang dapat diubah sesuai dengan kebutuhan pengguna pada *Client*. Informasi yang disimpan pada AgentProperties adalah variabel yang akan dibaca oleh *class* yang dibangun oleh Java. Dalam hal ini, disimpan informasi tentang port yang digunakan.



The screenshot shows a text editor window with the title bar 'AgentProperties.properties'. The content of the file is as follows:

```
# To change this template, choose Tools  
# and open the template in the editor  
port=6066
```

Gambar 59 AgentProperties pada Linux

Langkah-langkah instalasi *LinuxAgent Service* adalah sebagai berikut

1. Mengkompilasi *class* dan *script* yang dibutuhkan ke dalam *file* LinuxAgent.jar dan meletakkannya ke dalam satu direktori yang ditetapkan.

```
/home/ta2/linuxagent/LinuxAgent.jar
```

Gambar 60 LinuxAgent.jar pada suatu direktori

2. Membuat Java Daemon Script untuk mengakses LinuxAgent.jar sebagai service.

Pastikan variabel mengarah ke bagian-bagian aplikasi yang dibutuhkan.

```
# Set this to your Java installation
JAVA_HOME=/usr/java/latest

serviceNameLo="linuxagent"
serviceName="LinuxAgent"
serviceUser="root"
serviceGroup="root"
applDir="/home/ta2/$serviceNameLo"
serviceUserHome="/home/$serviceUser"
serviceLogFile= /var/log/$serviceNameLo.log"
maxShutdownTime=15
pidFile="/var/run/$serviceNameLo.pid"
javaCommand="java"
javaExe="$JAVA_HOME/bin/$javaCommand"
javaArgs="-jar $applDir/LinuxAgent.jar"
javaCommandLine="$javaExe $javaArgs"
javaCommandLineKeyword="LinuxAgent.jar"
quish it from others
```

Gambar 61 Konfigurasi Java Daemon Script untuk LinuxAgent

3. Mengaktifkan LinuxAgent Service.

```
[root@tk02 linuxagent]# service linuxagent start
Starting LinuxAgent    started PID=6776
```

Gambar 62 Mengaktifkan LinuxAgent Service

4.9 Batasan dalam Implementasi

Berdasarkan implementasi diatas, sistem memiliki batasan-batasan. Batasan tersebut antara lain:

- Penanganan dilakukan pada *user* dan *file*.

User dan *File* ditangani untuk dapat melihat hasil dari penambahan *user*, hasil dari penghapusan *user*, hasil dari penambahan *group* untuk *user*, hasil dari penghapusan *group* pada *user*, hak akses pada suatu *file*, dan properti *file*.

- *Security* belum dicakup pada sistem yang dibangun.

Penanganan dilakukan pada pembuatan dashboard, pembuatan koneksi *client-server*.

- Penanganan dilakukan pada 2 sistem operasi.

Sistem operasi pada sistem ini menggunakan Windows Server 2008 dan CentOS 6.4.

4.10 Pengujian Sistem

Pengujian sistem dilakukan menggunakan *case study* Institut Teknologi Del. Sistem disediakan menyerupai kondisi server-server (berperan sebagai klien terhadap aplikasi) yang digunakan di Institut Teknologi Del. Server yang disediakan dapat dilihat pada Tabel x.

Tabel 8 Kondisi Server yang digunakan

| No. | Jenis Server | Alias | Sistem Operasi | Servis yang Digunakan | Fungsi |
|-----|--------------|----------------|---------------------|---|---|
| 1. | File Server | file-server(1) | Windows Server 2008 | - Active Directory - File Server Resource Management - Domain Name Server | Menyimpan file-file yang dimiliki oleh user. |
| 2. | Mail Server | mail-server(1) | Linux CentOS 6.4 | - Squirrel Mail Server - Mysql - PhpMyAdmin - Domain Name Server | Menyimpan akun email dan akun database user. |
| 3. | Mail Server | mail-server(2) | Linux CentOS 6.4 | - Squirrel Mail Server - Domain Name Server | Menyimpan akun email user. Mempresentasikan aplikasi yang dapat menangani lebih dari dua server. |

Resource yang dikelola berupa akun user, *file* dan *folder* (direktori), dan akun *mysql*. Adapun user dan grup yang berperan dapat dilihat pada Tabel 9.

Tabel 9 Resource yang dikelola

| No. | Resource | Nama | Letak Resource | Keterangan |
|-----|-----------|---------|----------------------------------|-------------------------|
| 1. | Akun user | fidelis | file-server(1) mail-server(1) | Akun user pegawai baru. |
| 2. | Akun user | refindo | file-server(1) | Akun user |

| No. | Resource | Nama | Letak Resource | Keterangan |
|-----|------------|---------|--|--|
| | | | mail-server(1) mail-server(2) | pegawai lama. |
| 3. | Akun mysql | fidelis | mail-server(1) | Akun mysql pegawai baru. |
| 4. | Grup | staff | file-server(1) mail-server(1) mail-server(2) | Grup staff untuk pegawai dengan posisi Staff. |
| 5. | Grup | hrd | file-server(1) mail-server(1) mail-server(2) | Grup hrd untuk pegawai dengan posisi HRD (Human Resource Development). |
| 6. | Grup | other | file-server(1) mail-server(1) mail-server(2) | Grup other untuk akun yang tidak berada pada posisi Staff atau HRD. |
| 7. | Direktori | project | file-server(1) | Direktori yang diubah kepemilikan grup dan hak aksesnya. |

4.10.1 Pengujian pada Kasus Pertama

Kasus pertama adalah bagaimana memasukkan data pegawai baru pada file-server(1) dan mail-server(1). Dikarenakan pegawai berada di posisi *staff*, maka pegawai dialokasikan pada group *staff* yang dimiliki oleh kedua server tersebut.

The screenshot shows a user interface for managing users and groups. On the left, there's a sidebar with various menu items. The 'Users and Files' item is currently selected. The main area is titled 'Add User' and contains fields for 'User Name' (set to 'fidelis'), 'Client' (with checkboxes for 'file-server(1)' and 'mail-server(1)' checked, and 'mail-server(2)' unchecked), 'Group' (with checkboxes for 'staff' checked, and 'hrd' and 'other' unchecked), and 'User Description' (containing the text 'Fidelis Silalahi'). At the bottom right of the main area is a 'Add' button.

Gambar 63 Pengalokasian user "fidelis" ke grup "staff"

Pada mail-server(1), pegawai tersebut juga diberikan akun untuk *database* pada *mysql* server.

Send a Command to a Client

Fields with * are required.

Command *
AddMysqlUser

Arguments
fidelis

Client *
mail-server(1)

Send

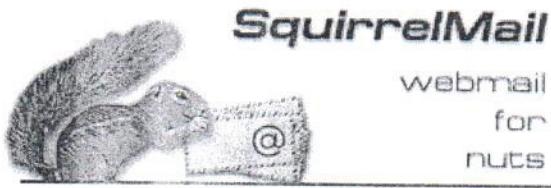
Gambar 64 Penambahan akun mysql untuk user "fidelis"

Pegawai tersebut berstatus aktif pada Institut Teknologi Del dan dapat menggunakan email dari mail-server(1) dengan akun tersebut.

Displaying 1-1 of 1 result.

| User Name | Client(s) | Group | Status | Single Operation | Broadcast Operation |
|-----------|-----------------------------------|---------|---------------------------------|------------------|---------------------|
| | | | | | |
| fidelis | [file-server(1)] [mail-server(1)] | [staff] | active [change] | | |

Gambar 65 User "fidelis" berstatus aktif



SquirrelMail version 1.4.22
By the SquirrelMail Project Team

Web Server TK02 Login

Name: fidelis

Password: *****

Login

Gambar 66 Login dapat dilakukan ke mail-server(1) dengan user "fidelis"

Setelah 3 tahun bekerja pegawai tersebut mendapat jabatan sebagai HRD (*Human Resource Development*) sehingga hak akses pegawai tersebut ditambahkan kepada hak akses grup *hrd* pada file-server(1).

Assign Group to User *fidelis* in Client
file-server(1)

ID# 48

Select the group(s) you want to assign for this user in this client.

Group *

hrd

other

Assign **Back**

Home
Users and Files
Add User
Groups
Add Group
Clients
Add Client
Commands
Command Send
Script Generator
Command Log
Logout (admin)

Gambar 67 Penambahan grup "hrd" terhadap user "fidelis"

| Displaying 1-2 of 2 results. | | | | | |
|------------------------------|-------------------------------|-----------------|--------|------------------|---------------------|
| User Name | Client(s) | Group | Status | Single Operation | Broadcast Operation |
| fidelis | [file-server(1)] [staff][hrd] | active [change] | | | |
| fidelis | [mail-server(1)] [staff] | active [change] | | | |

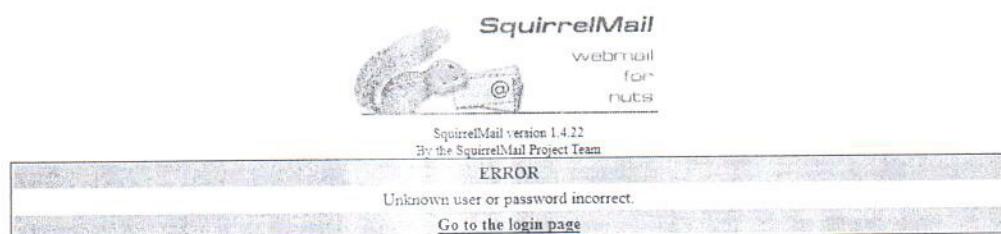
Gambar 68 User "fidelis" masuk ke grup "staff" dan "hrd" (multigroup)

4.10.2 Pengujian pada Kasus Kedua

Pada kasus kedua adalah ketika pegawai tersebut cuti setahun dikarenakan alasan tertentu. Karena itu pegawai dinonaktifkan sementara pada mail-server(1) oleh Administrator.

| Displaying 1-2 of 2 results. | | | | | |
|------------------------------|-------------------------------|-------------------|--------|------------------|---------------------|
| User Name | Client(s) | Group | Status | Single Operation | Broadcast Operation |
| fidelis | [file-server(1)] [staff][hrd] | active [change] | | | |
| fidelis | [mail-server(1)] [staff] | inactive [change] | | | |

Gambar 69 User "fidelis" dinonaktifkan pada mail-server(1)



Gambar 70 User "fidelis" gagal login pada mail-server(1)

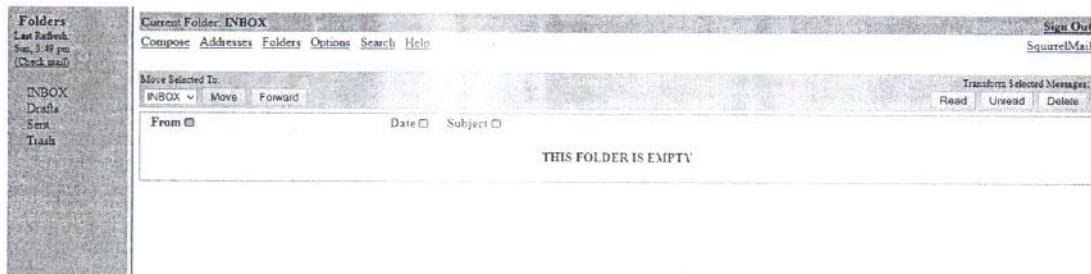
Setelah pegawai selesai dengan cuti nya tersebut, maka pegawai diaktifkan kembali pada mail server oleh Administrator.

Displaying 1-2 of 2 results.

| User Name | Client(s) | Group | Status | Single Operation | Broadcast Operation |
|-----------|---|-------|--------|------------------|---|
| | | | | | |
| fidelis | [file-server(1)] [staff][hrd] active [change] <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> | | | | <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> |

Gambar 71 User "fidelis" pada mail-server(1) diaktifkan kembali

Setelah pegawai diatifkan, pegawai dapat kembali menggunakan akun email tersebut.



Gambar 72 User "fidelis" dapat kembali login pada mail-server(1)

4.10.3 Pengujian pada Kasus Ketiga

Pada kasus ketiga adalah pegawai yang memiliki proyek membagi *folder* proyeknya kepada pegawai lain. *Folder* tersebut berada pada file-server(1) dan berada pada *home directory* milik pegawai tersebut.

Gambar 73 Sharing folder oleh user "fidelis"

Pada kasus ini, pegawai tidak berada di tempat, sehingga hal tersebut harus dikerjakan oleh Administrator. Proyek tersebut adalah proyek yang dimiliki oleh grup *staff*. Oleh karena itu, Administrator mengubah kepemilikan grup dari proyek itu untuk *staff*.

Gambar 74 User "fidelis" mengubah grup folder menjadi milik "staff"

Permission dari folder itu sendiri diubah menjadi *full access* untuk user yang terdaftar pada grup *staff*.



Gambar 75 User "fidelis" memberikan *full-access permission* kepada grup "staff"

4.10.4 Pengujian pada Kasus Keempat

Pada kasus keempat adalah pegawai lama yang sudah tidak berada pada Institut Teknologi Del. Akun pegawai lama berada pada file-server(1), mail-server(1), dan mail-server(2).

| User Name | Client(s) | Group | Status | Single Operation | Broadcast Operation |
|-----------|------------------|--------------|-------------------|------------------|---------------------|
| fidelis | [file-server(1)] | [staff][hrd] | active [change] | D F * | |
| | [mail-server(1)] | [staff] | inactive [change] | D F * | |
| | [file-server(1)] | [staff] | inactive [change] | D F * | |
| refindo | [mail-server(1)] | [staff] | inactive [change] | D F * | D F * |
| | [mail-server(2)] | [staff] | inactive [change] | D F * | |

Gambar 76 User "refindo" berstatus nonaktif pada ketiga server

Karena pegawai sudah tidak aktif lagi maka akun pegawai lama tersebut yang berada file-server(1), mail-server(1), dan mail-server(2) dihapus secara keseluruhan termasuk *file* pada *home directory* milik pegawai tersebut.

| User Name | Client(s) | Group | Status | Single Operation | Broadcast Operation |
|-----------|--------------------------------------|--------------|--------------------------------------|------------------|---------------------|
| fidelis | [file-server(1)] [mail-server(1)] | [staff][hrd] | active [change] inactive [change] | D F * | D F * |

Gambar 77 User "refindo" dihapus dari ketiga server beserta *home directory* miliknya

BAB V

Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan pelaksanaan tugas akhir mulai dari tahap pengumpulan data, analisa, perancangan implementasi sampai pengujian dapat diambil kesimpulan sebagai berikut:

- Sistem yang dibangun telah sesuai dengan ketentuan yang diberikan.
- Pembangunan *Distributed Server Resources Management* menggunakan *Dashboard* telah terimplementasikan.
- *Dashboard Distributed Server Resources Management* dapat digunakan oleh Administrator di Institut Teknologi Del.
- *Dashboard* dapat berinteraksi dengan 2 server yang berbeda yaitu sistem operasi CentOS dan sistem operasi windows server 2008.
- Pembangunan sistem dibangun dengan fitur yang memungkinkan Administrator untuk mengembangkan aplikasi lebih luas lagi, yaitu Script Generator.

5.2 Saran

- Administrator diharapkan untuk mengetahui *script* pada sistem operasi yang dipakai.
- *IP address* diharapkan tidak berubah (bersifat statis) agar pengiriman pesan antara *client* dan server tetap berada pada jalur yang tidak berubah-ubah.
- *Security* diharapkan dapat ditingkatkan dengan menambahkan validasi *client* dan *server* yang sesuai untuk membuat komunikasi *client-server* lebih aman.

Rujukan

- [1] Kusuma, Yogie. Aplikasi EIS untuk monitoring transaksi penjualan dan pembelian. Tasikmalaya: TI Universitas Siliwangi.
- [2] Sinambela, Eka. 2013. *Dashboard* untuk Pemantauan Operasional Jaringan. Bandung: Institut Teknologi Bandung
- [3] <http://id.wikipedia.org/wiki/MySQL>.