PPPL THIN CLIENT MEMANFAATKAN XML-RPC SEBAGAI SARANA TRANSAKSI DATA FORM BERBENTUK XML DENGAN SERVER PYTHON DAN CLIENT MENGGUNAKAN DELPHI

TUGAS AKHIR

Disusun Oleh :

Tyarso Bhari Permana
NRP. 5100 100 086

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2005

PPPL THIN CLIENT MEMANFAATKAN XML-RPC SEBAGAI SARANA TRANSAKSI DATA FORM BERBENTUK XML DENGAN SERVER PYTHON DAN CLIENT MENGGUNAKAN DELPHI

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing 1	Dosen Pembimbing 2
Ir. Muchammad Husni, M.Kom	Wahyu Suadi, M.Kom
NIP. 131 411 100	NIP. 132 303 065

DEPARTEMEN PENDIDIKAN NASIONAL INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI INFORMASI JURUSAN TEKNIK INFORMATIKA

SURABAYA
NOVEMBER, 2005

ABSTRAK

Protokol XML-RPC memungkinkan suatu aplikasi client server dikembangkan dengan mudah. Penanganan request antara client dan server dapat dibangun secara cepat dengan menggunakan fungsi fungsi yang sudah tersedia. Data yang dikomunikasikanpun dapat dalam bentuk yang bermacam macam.

Format data yang dikomunikasikan antara client dan server pada system berbasis protokol xml-rpc pada dasarnya diserahkan kepada programmer. Programmer bebas untuk menentukan bagaimana bentuk suatu data harus dikomunikasikan. Data dapat saja berupa struct dengan isi yang beraneka ragam. Data dapat pula berbentuk array dengan panjang tertentu dan struktur tertentu pula. Untuk memudahkan dalam proses pembuatan aplikasi dan memudahkan dalam proses rebuilding aplikasi, maka tugas akhir ini memfokuskan menggunakan data XML sebagai data yang dikomunikasikan antara client dengan server yang mana disini menggunakan python.

Data yang dikomunikasikan disini kemudian dikembangkan lagi untuk mendukung pengembangan sistem adaptif desktop pada client. Data yang dikomunikasikan bukan hanya berupa data yang berasal dari database untuk ditampilkan namun juga data yang merupakan interpretasi dari tampilan form yang akan ditampilkan kepada user. Data form xml ini berisi bagaimana tampilan suatu form tersebut akan ditampilkan kepada user. Data juga dapat berupa sintaks fungsi yang berjalan pada saat user melakukan aksi terhadap form tersebut.

Data form xml tersebut kemudian digenerate pada saat runtime oleh client sehingga form yang diinginkan dapat tampil berikut juga fungsi fungsi yang menyertainya. Dengan bentuk seperti ini diharapkan nantinya bila terdapat perubahan pada aplikasi client yang mana perubahan tersebut berupa perubahan tampilan form ataupun perubahan fungsi maka perubahan tersebut cukup

dilakukan di sisi server saja dengan merubah bentuk form xml yang diberikan ataupun merubah sintaks fungsi yang akan dikirimkan.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadapan Tuhan Yang Maha Esa atas restuNya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"PPPL THIN CLIENT MEMANFAATKAN XML-RPC SEBAGAI SARANA TRANSAKSI DATA FORM BERBENTUK XML DENGAN SERVER PYTHON DAN CLIENT MENGGUNAKAN DELPHI"

Tugas Akhir ini dibuat guna memenuhi persyaratan akademik dalam rangka ujian akhir bagi mahasiswa Strata 1 (S1) Jurusan Teknik Informatika , Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam Tugas Akhir ini penulis mengetengahkan penggunaan XML sebagai format komunikasi antara client dan server yang menggunakan protokol XML-RPC.

Pada kesempatan ini penulis menyampaikan ucapan terima kasih yang sebesar-besarnya atas bantuan dan dorongan dari semua pihak yang telah bersedia meluangkan waktu serta tenaga sehingga Tugas Akhir ini dapat diselesaikan dengan baik. Oleh karena itu pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih kepada yang terhormat:

1. Allah SWT.
2. Keluargaku. Ibuk, Bapak, Adek.

3. Bapak Ir Muchammad Husni, M.Kom selaku dosen pembimbing I dan Kalab Arsitektur dan Jaringan Komputer (AJK).
4. Bapak Wahyu Suadi, M.Kom selaku dosen pembimbing II.
5. Mantan Walikotamadya Surabaya. (Alm) Sunarto Sumoprawiro.
6. Walikotamadya Surabaya. Bp. Bambang DH.
7. Yayasan Tunas Paratama Bhakti (YTPB). Rumah kedua saya.
8. Seluruh Pembina, Staf Karyawan (terutama Ibu Dapur) Yayasan Tunas Paratama Bhakti (YTPB). Semua teman teman asrama cowok. Semua teman teman asrama cewek.
9. Semua warga AJK. Prevan, Fadelis, Dwi Arie, Surya TW, Januar, Yoyok, Victorio, Ronnie M, Roy S. Dan semuanya yang tidak mungkin saya sebutkan satu persatu.
10. Para senior yang telah banyak membimbing. mas Idi, mas Damen, mas Putu, mas Nunut dan senior senior lainnya
11. Semua Karyawan Teknik Informatika (TC) – FTIf

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyusun Tugas Akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang ada.

DAFTAR ISI

ABSTRAK i
KATA PENGANTAR iii
DAFTAR ISI vi
DAFTAR GAMBAR ix
DAFTAR TABEL xi
BAB 1 PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Tujuan dan Manfaat 2
1.3. Perumusan Masalah 3
1.4. Pembatasan Masalah 3
1.5. Metodologi Pembuatan Tugas Akhir 3
1.6. Sistematika Pembahasan 5
BAB 2 DASAR TEORI 7
2.1 Teknologi RPC (Remote Procedure call) 7
2.1.1 RPC Model 9
2.1.2 RPC Message Protocol 10
2.1.3 Requirement Protokol RPC 11
2.1.4 RPC Message 11
2.1.4.1 RPC Call Message 12
2.1.4.2 RPC Reply Message 12
2.2 Teknologi XML-RPC 13
2.2.1 Pengertian 13
2.2.2 Tujuan 14
2.2.3 Tipe Data 15
2.2.4 XML-RPC dan Protokol Lainnya 16
2.2.4.1 XML-RPC vs. CORBA 16
2.2.4.2 XML-RPC vs. DCOM 16
2.2.4.3 XML-RPC vs. SOAP 17
2.3 XML 17

2.3.1 Pengertian 17
2.3.2 Elemen, Attribute dan entity 20
2.3.3 Well-formed XML 21
2.3.4 Parser 27
2.3.4.1 DOM 27
2.4 Windows Scripting Host 28
2.4.1 TekWshControl 29
BAB 3 PERANCANGAN PERANGKAT LUNAK 31
3.1 Deskripsi Umum 31
3.1.1 Arsitektur Sistem 32
3.2 Perancangan Proses 33
3.2.1 Proses pengepakan data ke dalam XML 34
3.2.1.1 Pengepakan Data Form menjadi XML 34
3.2.1.2 Pengepakan data recordset menjadi XML 38
3.2.1.3 Pengepakan Script Fungsi Form menjadi XML 40
3.2.2 Proses komunikasi antara client dan server 40
3.2.2.1 Proses Pengepakan Request 43
3.2.2.2 Proses Penerimaan Request 45
3.2.2.3 Pemrosesan Request 46
3.2.2.4 Proses Pengepakan Result 48
3.2.2.5 Proses pembacaan hasil 49
3.3 Perancangan Data 51
3.3.1 Struktur Message Client 52
3.3.1.1 Definisi Alamat 52
3.3.1.2 Definisi Parameter 53
3.3.2 Struktur Message Server 53
3.3.3 Struktur Database 54
3.4 Perancangan Antar Muka 59
3.4.1 Perancangan Antarmuka Client 60
BAB 4 IMPLEMENTASI PERANGKAT LUNAK 64
4.1. Implementasi aplikasi pengubah dfm menjadi xml 64

4.2. Konfigurasi webserver (Apache) 67
4.2.1 Kompilasi modul WebKit – Webware 68
4.2.2 Konfigurasi Apache 68
4.3. Implementasi Server Aplikasi (server aplikasi) 70
4.3.1 Pembacaan file xml-form dan file script fungsi 70
4.3.2 Proses pengepakan data menjadi xml 71
4.3.3 Proses pengepakan fungsi script menjadi xml 72
4.3.4 Proses penanganan request 73
4.3.5 Proses koneksi database 74
4.3.6 Memasukkan fungsi kedalam script servlet pada webware 75
4.4. Implementasi Aplikasi Client 76
4.4.1 Proses koneksi 76
4.4.2 Pengolahan string 77
4.4.2.1 Register fungsi script 77
4.4.2.2 Generate form runtime 78
4.4.2.3 Update tampilan aktif form 79
BAB 5 UJI COBA DAN EVALUASI 80
5.1 Deskripsi Hardware dan Software Uji Coba 80
5.2 Skenario Uji Coba 81
5.2.1 Uji Coba Skenario View Data Member 82
5.2.2 Uji Coba Skenario View Data Koleksi 85
5.2.3 Uji Coba skenario Penambahan Fungsi Searching pada form anggota 88
BAB 6 KESIMPULAN DAN SARAN 90
6.1. Kesimpulan 90
6.2. Saran 91
DAFTAR PUSTAKA 92

DAFTAR GAMBAR

Gambar 2.1 Gambar Flow dari RPC 10
Gambar 2.2 Gambaran konsep dasar XML-RPC 14
Gambar 2.3 Diagram Hirarki XML 22
Gambar 2.4 Diagram Alir Windows Scripting Host 29
Gambar 3.1 Arsitektur Sistem 33
Gambar 3.2 Form Login 35
Gambar 3.3 Diagram Alir DFM menjadi XML 36
Gambar 3.4 Diagram Alir Pengepakan data menjadi XML 39
Gambar 3.5 Diagram alir proses umum 41
Gambar 3.6 Diagram Alir Proses Pengepakan Request 43
Gambar 3.7 Diagram Alir Proses Penerimaan Request 45
Gambar 3.8 Diagram Alir Pemrosesan Request 47
Gambar 3.9 Diagram Alir Pengepakan Result 48
Gambar 3.10 Contoh Data Form XML 49
Gambar 3.11 Diagram alir pembacaan hasil 51
Gambar 3.12 Gambar CDM 54
Gambar 3.13 Gambar PDM 55
Gambar 3.14 Tipe data tabel login 56
Gambar 3.15 Tipe data tabel kategori_anggota 57
Gambar 3.15 Tipe data tabel anggota 57
Gambar 3.16 Tipe data tabel kategori_koleksi 58
Gambar 3.17 Tipe data tabel status_koleksi 58
Gambar 3.18 Tipe data tabel master_koleksi 59
Gambar 3.19 Halaman Login 60
Gambar 3.20 Halaman Login 60
Gambar 3.21 Form Ruang 1 61
Gambar 3.22 Halaman Member 62
Gambar 3.23 Halaman Koleksi 63

Gambar 4.1 Hasil Testing Webware Sukses 70
Gambar 5.1 Proses view data member oleh Sistem yang menggunakan XML-RPC 82
Gambar 5.2 Proses view data koleksi oleh Sistem yang menggunakan XML-RPC 85
Gambar 5.3 Form anggota baru 88

DAFTAR TABEL

Tabel 2.1 Daftar entity reference 26
Tabel 5.1 Hasil ujicoba data 10 82
Tabel 5.2 Hasil ujicoba data 25 83
Tabel 5.3 Hasil ujicoba data 50 83
Tabel 5.4 Hasil ujicoba data 75 84
Tabel 5.5 Hasil ujicoba data 100 84
Tabel 5.6 Hasil ujicoba data 10 86
Tabel 5.7 Hasil ujicoba data 25 86
Tabel 5.8 Hasil ujicoba data 50 87
Tabel 5.9 Hasil ujicoba data 75 87
Tabel 5.10 Hasil ujicoba data 100 87

BAB 1
PENDAHULUAN

Dalam bab ini akan akan dijelaskan mengenai latar belakang, permasalahan, serta tujuan dan manfaat dari pembuatan Tugas Akhir ini. Selain itu bab ini juga menjelaskan batasan masalah, metodologi pembuatan Tugas Akhir dan sistematika pembahasan keseluruhan Tugas Akhir.

1.1. Latar Belakang

Protokol XML-RPC memungkinkan suatu aplikasi client server untuk dikembangkan secara mudah dan cepat. Protokol ini menyediakan fungsi fungsi yang dapat digunakan baik client maupun server untuk meminta layanan dan memberikan jawaban dengan mudah. Data yang dikomunikasikan antara client dan server dapat berupa angka, string, date/time, dan juga dapat berupa data record ataupun list.

Kemudahan yang ditawarkan oleh protokol XML-RPC ini menemui sedikit kendala pada saat digunakan untuk membangun aplikasi client server dengan client menggunakan bahasa pemrograman delphi. Hal ini dikarenakan tipe data yang diterima baik oleh delphi hanyalah data yang berbentuk string.

Untuk mengatasi kendala tersebut lalu diusulkan untuk membungkus terlebih dahulu isi dari data yang ingin dikomunikasikan ke dalam bentuk XML. Format penyusunan data tersebut tentu saja bergantung dengan kebutuhan dan diserahkan sepenuhnya kepada pembuat aplikasi. Ide pembungkusan data yang

akan dikomunikasikan tersebut inilah yang menjadi fokus pembahasan dari tugas akhir kami.

Dalam tugas akhir kami ini data yang dikomunikasikan meliputi bentuk dari suatu form, data data yang harus ditampilkan pada form tersebut, dan fungsi fungsi yang menyertai form tersebut. Data bentuk form akan digunakan oleh client untuk melakukan generate secara runtime form yang akan ditampilkan kepada user. Data yang ditampilkan lalu akan dimasukkan ke dalam form yang baru digenerate tersebut. Data fungsi form berupa fungsi fungsi yang menyertai suatu form.

Dengan bentuk data seperti diatas diharapkan aplikasi yang telah dibangun dapat terus dikembangkan sesuai dengan kebutuhan tanpa harus melakukan perubahan pada program client. Perubahan seperlunya dapat dilakukan pada sisi server sehingga akan mempermudah proses pengembangan aplikasi

1.2. Tujuan dan Manfaat

Tujuan pembuatan tugas akhir ini adalah mengimplementasikan protokol XML-RPC untuk digunakan pada suatu aplikasi client server.

Aplikasi yang dibangun merupakan prototipe dari sistem adaptif desktop dengan basis protokol XML-RPC. Aplikasi dikembangkan berdasar pemikiran bahwa jika suatu saat terjadi perubahan pada sisi client, maka perubahan tersebut cukup dilakukan seperlunya pada sisi server. Hal ini dikarenakan semua form yang akan ditampilkan kepada user telah disiapkan di sisi server dan akan dikirimkan pada saat aplikasi dijalankan.

1.3. Perumusan Masalah

Permasalahan yang diangkat dalam tugas akhir ini adalah bagaimana membangun aplikasi client server dengan menggunakan protokol XML-RPC. Permasalahan selanjutnya adalah bagaimana suatu client dapat melakukan generate form secara runtime baik bentuk, data yang ditampilkan dan fungsi fungsi yang menyertainya berdasarkan data xml yang diterima dari server.

1.4. Pembatasan Masalah

Sistem yang dibuat, mencakup komunikasi dan pertukaran data antara client, server dan database. Form yang akan ditampilkan kepada user berasal dari data yang dikirim oleh server. Interface pada user dibatasi hanyalah berupa tampilan sederhana. Perubahan pada form user dapat dilakukan dengan merubah bentuk data yang ada pada sisi server.

Even even yang dapat dibangkitkan terhadap form pada saat runtime hanya even onclick saja terhadap suatu objek.

1.5. Metodologi Pembuatan Tugas Akhir

Pembuatan tugas akhir ini dilakukan dengan mengikuti metodologi sebagai berikut:

1. Studi literature dan software yang ada
   - Pada tahap ini akan dipelajari sejumlah literatur mengenai konsep dan teknologi yang akan digunakan. Informasi yang dibutuhkan dapat diperoleh dari literatur yang berhubungan dengan Teknologi

RPC , teknologi XML-RPC, XML, DOM dan Wsh. Informasi ini dapat meliputi buku referensi, majalah dan dokumentasi internet.

2. Perancangan Sistem dan Aplikasi
   - Pemodelan dan perancangan sistem

     Pada tahap ini akan dibuat arsitektur sistem. Perancangan pertama adalah penyusunan database. Database disusun berdasarkan studi kasus yang dipilih. Lalu yang kedua, merancang interface client. Berdasarkan database yang telah disusun, tampilan client dapat diperkirakan. Yang ketiga, adalah Server aplikasi. Server yang akan dibuat harus bisa menjembatani client dan database. Langkah berikutnya adalah menyatukan server aplikasi yang telah dibuat dengan Web Services. Langkah terakhir adalah membangun aplikasi client yang dapat melakukan generate form secara runtime saat menerima data dari server.

3. Pembuatan Perangkat Lunak
   - Pada tahap ini, model dan rancangan sistem yang telah dibuat akan diimplementasikan. Database yang dibuat menggunakan postgresql, server aplikasi menggunakan Python, client menggunakan Delphi, dan Web Services menggunakan webware yang terintegrasi dengan apache.

4. Uji Coba dan Evaluasi
   - Pada tahap ini, dilakukan uji coba dengan beberapa data untuk kemudian dilakukan evaluasi terhadap hasil uji coba tersebut.

5. Penyusunan Buku Tugas Akhir

   - Tahap ini merupakan tahap terakhir dari proses pengerjaan Tugas Akhir ini. Buku dokumentasi akan disusun sebagai laporan dari seluruh proses pengerjaan Tugas Akhir ini.

1.6. Sistematika Pembahasan

Sistematika penulisan tugas akhir ini dapat dijelaskan sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi latar belakang masalah, tujuan dan manfaat, perumusan masalah, batasan permasalahan, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II DASAR TEORI

Bab ini membahas konsep dasar dari Teknologi RPC, Teknologi XML-RPC, XML, DOM dan Wsh.

BAB III PERANCANGAN PERANGKAT LUNAK

Bab ini membahas desain dari perangkat lunak yang akan dibuat meliputi : desain antarmuka,desain data, dan desain proses.

BAB IV IMPLEMENTASI PERANGKAT LUNAK

Bab ini membahas implementasi dari desain-desain sistem yang dilakukan pada tahap desain termasuk didalamnya disertakan source code yang penting dalam aplikasi.

BAB V UJI COBA DAN EVALUASI

Pada bab ini dibahas mengenai uji coba dari aplikasi yang dibuat dengan melihat output yang dihasilkan oleh aplikasi. Dari output

tersebut dapat dilakukan evaluasi untuk mengetahui kemampuan dari aplikasi yang dibuat.

BAB VI KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi selanjutnya.

BAB 2
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori penunjang dalam perancangan dan pembuatan aplikasi. Diantaranya adalah mengenai RPC, XML-RPC, XML, DOM dan Wsh.

2.1 TEKNOLOGI RPC (REMOTE PROCEDURE CALL)

RPC (Remote Procedure Call) adalah sebuah protokol. RPC menyempurnakan low-level transport protocol yang sudah ada seperti TCP/IP atau UDP dalam hal transportasi message pada program-program yang berkomunikasi [1]. RPC dibangun diatas protokol XDR (eXternal Data Representation), yang melakukan standarisasi dari representasi data pada remote communications. XDR mengonkonversi parameter dan result dari tiap service RPC yang disediakan.

Dengan menggunakan RPC, user dapat bekerja dengan remote procedure seakan-akan prosedur tersebut adalah local procedure. Remote procedure calls, didefinisikan melalui routine yang ada dalam protokol RPC. Setiap call message dicocokkan dengan reply message. Protokol RPC adalah message-passing protocol yangmengimplementasikan protokol non RPC lainnya seperti batching

---

[1] http://publib16.boulder.ibm.com/pseries/en_US/aixprggd/progcomc/ch8_rpc.htm

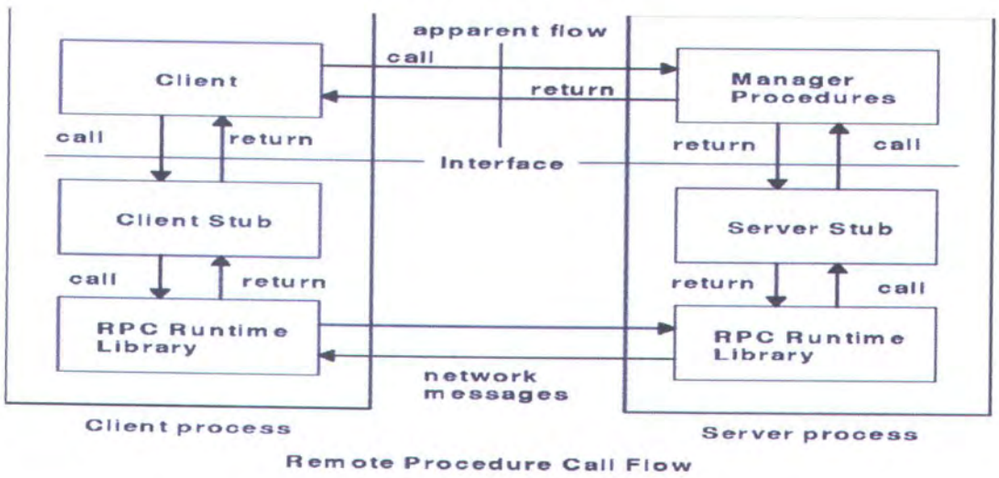
mengirimkan hasilnya kembali pada client. Call prosedur selanjutnya kembali pada client.

RPC interface secara general digunakan utuk komunikasi antar proses pada workstation yang berbeda dalam network. Namun RPC juga berjalan baik pada komunikasi antara proses yang berbeda pada satu workstation yang sama.

2.1.1 RPC Model

Model dari RPC hampir sama dengan model dari local procedure call. Pada model lokal, caller menempatkan argument untuk prosedur pada lokasi spesifik seperti result register. Lalu, caller mentransfer control ke prosedur. Saat sudah mendapatkan control, caller mengekstrak result dari procedure dan melanjutkan proses eksekusinya.

RPC berjalan pada konteks yang sama. Pertama, caller mengirimkan call message yang berisikan parameter dari prosedur pada server. Lalu, caller mengunggu reply message (blocks). Di dalam server, setelah mendapatkan call message dari caller, mengekstrak parameter dari prosedur, mengkomputasikan result dan mengirimkan hasilnya sebagai reply message. Lalu server menunggu datangnya call message yang lain. Akhirnya, caller menerima reply message, mengekstrak result dari prosedur dan melanjutkan proses eksekusi. Diagram dibawah menjelaskan paradigma RPC.



Gambar 2.1 Gambar Flow dari RPC [4]

Pada model RPC diatas, hanya satu dari dua proses yang aktif pada saat yang sama. Namun begitu, protokol RPC tidak membatasi diri pada model tersebut. Implementasi dapat menggunakan asynchronous RPC, yang membuat client dapat terus bekerja tanpa perlu menunggu result dari server. Pada sisi server, server dapat membuat task untuk memroses request yang datang dan sesudahnya bebas menerima request lainnya.

2.1.2 RPC Message Protocol

RPC message protocol terdiri dari dua struktur yag berbeda. Call message dan reply message. Sebuah client membuat remote procedure call pada network server dan menerima reply message yang berisi result eksekusi dari prosedur [5].

---

[4] http://publib16.boulder.ibm.com/pseries/en_US/aixprggd/progcomc/rpc_mod.htm

[5] http://publib16.boulder.ibm.com/pseries/en_US/aixprggd/progcomc/rpc_msg.htm

Dengan menyediakan spesifikasi yang unik untuk remote procedure, RPC dapat mencocokkan reply message utnuk tiap call message.

RPC message protocol didefinisikan menggunakan XDR, yang berisikan structures, enumerations dan unions.

Saat RPC message dilewatkan menggunakan TCP/IP byte-stream protocol untuk transport data, sangat penting untuk mengidentifikasikan akhir dari sebuah message dan awal dari message lainnya.

2.1.3 Requirement Protokol RPC

RPC message memerlukan:

- Spesifikasi unik dari tiap prosedur yang diminta
- Pencocokan response message (reply message) dengan request message.
- Authentication dari caller ke service vise versa.

Untuk mendukung hal tersebut dan sekaligus membantu administrasi network yang berhubungan dengan kebutuhan tersebut, features yang dapat mendeteksi konndisi berikut akan sangat membantu.

- RPC protocol mismatches
- Remote program protocol version mismatches
- Protocol errors (seperti kesalahan menspesifikasikan procedure parameter)
- Alasan mengapa remote authentication gagal
- Alasan mengapa prosedur yang diinginkan tidak dapat diminta

2.1.4 RPC Message

Struktur dari RPC message mengikuti struktur sebagai berikut:

```
Struct rpc_msg{
        unsigned int    xid;
        union switch (enum msg_type mtype) {
                case CALL:
                        call_body cbody;
                case REPLY;
                        reply_body rbody;
        } body;
};
```

Semua call dan reply message dimulai dengan transaction identifier, xid, yang diikuti oleh dua discriminated union. Discriminant tersebut adalah msg_type, yang memilih salah satu tipe: CALL atau REPLY.

2.1.4.1 RPC Call Message

Setiap remote procedure call message selalu berisi fields yang mengidentifikasikan remote procedure yang dimaksud. Filed tesebut:

- Program number
- Program version number
- Procedure number

2.1.4.2 RPC Reply Message

Reply message berisikan informasi yang menjabarkan kondisi berikut:

- Eksekusi prosedur yang diminta call message sukses.
- Remote implementation dari RPC bukan protokol versi 2. RPC version number yang lebih tinggi atau rendah dikembalikan pada client.
- Remote program tidak tersedia di remote sistem.

- Remote program tidak mensupport version number yang diminta. Remote program versi yang lebih rendah atau tinggi dikembalikan pada client.
- Procedure number yang diminta tidak eksis. Hal ini biasanya merupakan kesalahan dari client atau programming error.

2.2 TEKNOLOGI XML-RPC

2.2.1 Pengertian

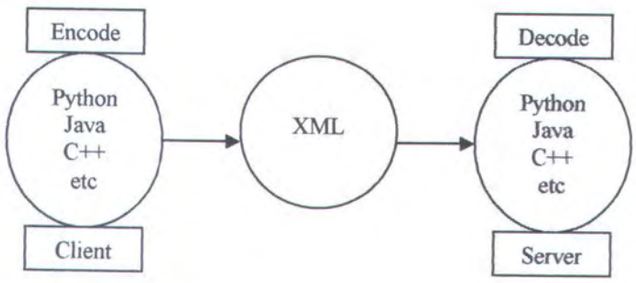
XML-RPC adalah protokol RPC yang bekerja pada intranet dan internet. Sebuah message XML-RPC adalah HTTP-POST request berbasis HTTP 1.0 dan 1.1 [6].

Pada RPC, perlu adanya standarisasi message. Sehingga client dan server yang berkomunikasi dapat saling mengerti. Dalam prakteknya, untuk lebih mempermudah masalah tersebut, digunakan platform yang sama dalam komunikasi. Misalnya: client dan server menggunakan python.

XML-RPC menggunakan XML sebagai standarisasi. Semua message yang dikirim berada dalam format XML. Oleh karena itu, dengan menggunakan XML-RPC, platform client dan server tidak dibatasi lagi. Client dapat saja menggunakan bahasa pemrograman yang lain dengan server.

---

[6] http://www.xmlrpc.com



Gambar 2.2 Gambaran konsep dasar XML-RPC

Prinsip XML-RPC diwakilkan pada gambar diatas. Dengan kata lain XML-RPC adalah sebuah remote procedure yang mengunakan HTTP sebagai transportasi dan XML sebagai encoding. Client mengencode informasi menjadi common language (XML). Server mendecode informasi tersebut. XML-RPC didesain sesimple mungkin. Walaupun begitu, XML-RPC mensupport struktur data yang kompleks untuk ditransmit, diproses dan dikembalikan return valuenya.

2.2.2 Tujuan

Tujuan utama XML-RPC adalah kemudahan. XML-RPC sengaja dibuat sebagai protokol yang mudah diimplementasikan dan beradaptasi dengan lingkungan yang berbeda atau sistem operasi yang berbeda [7]. XML-RPC digunakan untuk mencari cara yang simple dan efektif dalam hal permintaan dan penerimaan informasi.

---

[7] http://www.xmlrpc.com

Kemudahan tersebut selain menguntungkan, dapat pula merugikan. XML-RPC dalam penggunaannya, akan menggenerate sebuah message XML yang dikirimkan pada server. User, hanya perlu memasukkan nama service dan parameter bagi service tersebut. Dengan kata lain, kustomisasi message tidak dapat dilakukan secara menyeluruh. User dituntut untuk mengisi message sesuai template yang sudah ada. Hal tersebut dapat menjadi kerugian bila user ingin secara langsung mengkustomisasi message.

2.2.3 Tipe Data

XML-RPC mensupport tipe data sebagai berikut

int

Signed, 32-bit integer.

String

ASCII string, dapat berupa NULL bytes. (Sebenarnya, beberapa imlementasi dari XML-RPC mensupport Unicode).

boolean

True or False

dateTime.iso8601

Date dan time. Sayangnya, karena XML-RPC tidak menghiraukan penggunaan dari timezone, tipe ini menjadi tidak berguna.

base64

Data biner yang Raw dengan panjang sembarang; di encode mengunakan base64.

array

Array satu dimensi.

struct

Kumpulan dari pasangan key-value. Key adalah string; value dapat berupa tipe-tipe yang lain.

2.2.4 XML-RPC dan Protokol Lainnya

XML-RPC bukanlah satu-satunya cara untuk mengimplementasikan remote procedure calls. Terdapat protokol-protokol lainnya seperti CORBA, DCOM dan SOAP. Tiap-tiap protokol memiliki keunggulan dan kelemahan tersendiri.

2.2.4.1 XML-RPC vs. CORBA

CORBA adalah protokol yang digunakan untuk menulis aplikasi sistem terdistribusi berbasis object orioented. Secara umum digunakan pada aplikasi multi-tier. CORBA disupport oleh banyak vendor dan beberapa free software projects. CORBA berjalan dengan baik pada Java dan C++. CORBA juga memiliki interface definition language (IDL) yang bagus.

Sayangnya, CORBA disusun dengan sangat kompleks. Perlu usaha lebih dalam mengimplementasikannya dan memerlukan client yang cukup kompleks pula. Cocok digunakan untuk aplikasi enterprise dan desktop daripada aplikasi web yang terdistribusi.

2.2.4.2 XML-RPC vs. DCOM

DCOM adalah jawaban MICROSOFT mengenai CORBA. Sangat membantu bila kita sudah pernah menggunakan komponen COM, dan tidak perlu

berhubungan dengan aplikasi non-Microsoft. Sebaliknya, bisa sangat tidak berguna.

2.2.4.3 XML-RPC vs. SOAP

XML-RPC berbeda dengan SOAP, walaupun fungsinya nyaris sama. Perbedaan tersebut ada pada struktur dari XML yang dikirim. Sayangnya, pengguna SOAP akan dituntut lebih pada proses spesifikasi message.

XML-RPC didesain sesimple mungkin walaupun mensupport struktur data yang kompleks untuk ditransmit, diproses dan dikembalikan return valuenya. Walaupun begitu, desain ini adalah juga kelemahan dari XML-RPC. Walaupun kompleks, struktur data yang ditransportasikan haruslah umum. Dalam XML-RPC, kita tidak bisa dengan mudah mendefinisikan sebuah tipe data. Tambahan lagi, selain tipe data masih ada lagi fitur-fitur yang tidak dapat dispesifikasikan di XML-RPC seperti penerima message, pemrosesan message. SOAP dibuat untuk menjawab kesulitan tersebut.

Secara singkat, bila kita menyukai XML-RPC, namun menginginkan protokol tersebut memiliki fitur yang lebih kompleks lagi, kita dapat menggunakan SOAP.

2.3 XML

2.3.1 Pengertian

XML versi 1.0 diperkenalkan oleh XML Working Group pada World Wide Web Consortium (W3C) pada bulan Februari 1998. XML merupakan

singkatan dari eXtensibleMarkup Language yaitu sebuah aturan untuk memecah suatu dokumen menjadi bagian bagiannya dengan menambahkan semantic tag ke dalam bagian bagian dari dokumen tersebut [8]. XML sebagai bahasa markup hampir sama dengan HTML dimana dokumen HTML juga menyertakan tag tag ke dalam elemen eleman dari dokumen HTML tersebut.

Teknologi yang digunakan pada XML sebenarnya bukan teknologi baru, tapi merupakan turunan dari SGML yang telah dikembangkan pada awal 80-an dan telah banyak digunakan pada dokumentasi teknis proyek-proyek berskala besar. Ketika HTML dikembangkan pada tahun 1990, para penggagas XML mengadopsi bagian paling penting pada SGML dan dengan berpedoman pada pengembangan HTML menghasilkan markup language yang tidak kalah hebatnya dengan SGML.

Seperti halnya HTML, XML juga menggunakan elemen yang ditandai dengan tag pembuka (diawali dengan '<' dan diakhiri dengan '>'), tag penutup(diawali dengan '</ 'diakhiri '>') dan atribut elemen(parameter yang dinyatakan dalam tag pembuka misal <form name="isidata">). Hanya bedanya, HTML medefinisikan dari awal tag dan atribut yang dipakai didalamnya, sedangkan pada XML kita bisa menggunakan tag dan atribut sesuai kehendak kita

Tag tag sebagai penanda bagian dalam dokumen XML dapat kita tentukan sendiri. Salah satu contohnya adalah bila kita ingin mengolah data dari mahasiswa

---

[8] Junaedy, Pengantar XML, www.ilmukomputer.com

maka kita dapat menggunakan tag tag semisal: NRP, NAMA, ALAMAT dan lain sebagainya sebagai penanda elemen dari dokumen mahasiswa tersebut.

XML sebagai bahasa markup berisi deskripsi dari struktur dokumen tersebut dan arti yang dikandungnya. Deskripsi dari dokumen dan arti yang dikandungnya dapat berbeda beda bergantung dari orang yang akan menggunakan dokumen XML tersebut. Hal ini berbeda bila kita membandingkan tag tag yang dipergunakan dalam dokumen HTML adalah untuk memberikan deskripsi tampilan dari suatu dokumen. Hal inilah yang membedakan antara penggunaan markup pada XML dan pada HTML.

Berikut ini dalah salah satu contoh bila kita ingin menampilkan suatu data dokumen musik baik menggunakan HTML maupun menggunakan XML.

Data dalam bentuk HTML

```
<dt>Hot Cop
<dd> by Jacques Morali, Henri Belolo, and Victor Willis
<ul>
<li>Producer: Jacques Morali
<li>Publisher: PolyGram Records
<li>Length: 6:20
<li>Written: 1978
<li>Artist: Village People
</ul>
```

Data dalam bentuk XML

```
<SONG>
<TITLE>Hot Cop</TITLE>
<COMPOSER>Jacques Morali</COMPOSER>
<COMPOSER>Henri Belolo</COMPOSER>
<COMPOSER>Victor Willis</COMPOSER>
```

<PRODUCER>Jacques Morali</PRODUCER>
<PUBLISHER>PolyGram Records</PUBLISHER>
<LENGTH>6:20</LENGTH>
<YEAR>1978</YEAR>
<ARTIST>Village People</ARTIST>
</SONG>

Pada bentuk HTML, tag tag yang ada adalah mewakili bentuk tampilan dokumen tersebut saat dipanggil melalui browser. Sedangkan pada bentuk XML, tag tag yang ada mewakili deskripsi dari data lagu yang ada. Dengan menggunakan bentuk XML, maka data akan lebih mudah dibaca dan lebih menggambarkan apa yang diinginkan oleh pembuat dokumen.

XML memungkinkan tiap orang untuk membuat deskripsi dokumen sesuai keinginannya. XML menggunakan ASCII text sehingga mudah untuk mengolahnya. XML sangat ideal untuk menangani data dalam bentuk terstruktur yang berjumlah besar.

2.3.2 Elemen, Attribute dan entity

Dokumen XML selalu memiliki satu elemen sebagai root. Root elemen tersebut kemudian bisa saja memiliki beberapa sub-elemen. Beberapa sub-elemen tersebut juga dapat memiliki sub elemen di dalamnya dan begitu seterusnya. Tiap elemen berupa text yang diapit huruf "<" dan ">". Tiap elemen dapat memiliki isi atau juga tidak. Bila suatu elemen memiliki isi, maka isi dituliskan di antara tag pembuka dan tag penutup. Contoh dari elemen yang memiliki isi adalah <nama>Yanto</nama>, dimana elemen tersebut memiliki nama elemen "nama"

dan memiliki isi "Yanto". Isi suatu elemen bisa saja bukan text tapi berupa elemen lainnya. Bila suatu elemen tidak memiliki isi maka elemen harus ditulis dengan satu tag dengan "/" di akhir tag. Contohnya adalah : <selesai/> dimana elemen tersebut memiliki nama "selesai" dan tidak memiliki isi apapun.

Attribut adalah nilai yang melekat terhadap suatu elemen. Contoh elemen yang memiliki attribute adalah : <mahasiswa nama="yuli"></mahasiswa>. Elemen ini memiliki nama elemen "mahasiswa" dan memiliki attribute "nama" dengan nilai attribut "yuli". Setiap elemen dapat memiliki nilai attribute yang berbeda beda.

Entity adalah cara penulisan untuk string string tertentu. Contohnya adalah apabila elemen dengan nama "buku" dan memiliki attribute "judul" dengan nilai attribute "Bumi & langit" maka untuk menuliskan nilai attribut judul supaya memilki tanda "&" di dalam judulnya adalah dengan menambahkan & di nilai attributnya.

2.3.3 Well-formed XML

Pengertian dari well-formed XML disini adalah aturan penulisan yang benar dari dokumen XML. Aturan ini diperlukan karena setiap orang bebas untuk mendefinisikan sendiri dokumen XML yang dibuatnya. Berikut akan dibahas beberapa aturan penulisan dokumen XML. :

Heading standard untuk Document XML

Biasakanlah setiap membuat dokumen XML diawali dengan heading standard XML. Formatnya adalah sebagai berikut:

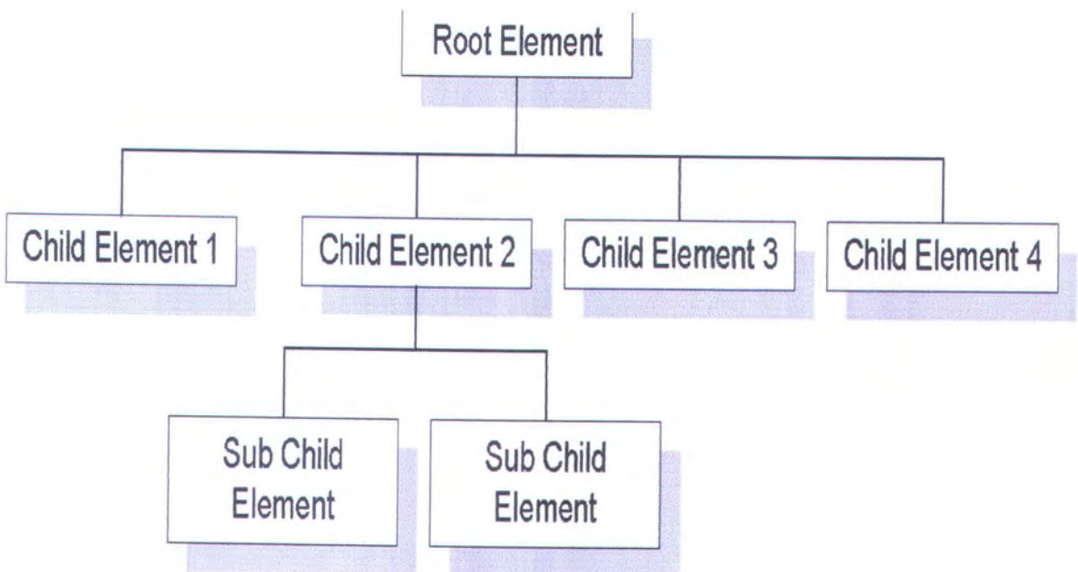
<?xml version="1.0" encoding="iso-8859-1"?>

Dokumen XML harus memiliki Root tag

Sebuah dokumen XML yang baik harus memiliki root tag. Yaitu tag yang melingkupi keseluruhan dari dokumen. Tag-tag yang lain, disebut child tag, berada didalam root membentuk hirarki seperti gambar 2.1

Contoh:

```
<root>
    <child>
        <subchild></subchild>
    </child>
</root>
```



Gambar 2.3 Diagram Hirarki XML [9]

[9] Junaedy, Pengantar XML, www.ilmukomputer.com

Tag pada XML harus lengkap berpasangan

Pada HTML beberapa elemen tidak harus berpasangan. Contoh berikut ini diperbolehkan dalam penulisan HTML.

paragraph pertama

paragarap kedua

yang demikian tidak berlaku pada XML. Kita harus menulis pula tag penutup untuk setiap tag yang kita buat. Penulisannya harus seperti ini

paragraph pertama

paragarap kedua

Tag tunggal hanya diperbolehkan untuk elemen kosong. Contoh penulisannya sebagai berikut:

<anggota nama="budi"/>

XML membedakan huruf besar dengan huruf kecil

Pada XML, <tanggal> berbeda dengan <Tanggal>. Tag pembuka dan tag penutup harus sama susunan huruf besar dan kecilnya.

<contoh>ini penulisan yang salah</Contoh>

<contoh>ini baru betul</contoh>

Penyarangan tag harus benar.

Penulisan tag pada XML harus mengikuti aturan Last In First Out (LIFO). seperti yang kita bahas terdahulu, pada XML kita tidak bisa membuat tag yang saling bersilang seperti dibawah ini

Huruf Tebal tapi harus disusun seperti ini

Huruf tebal

XML mempertahankan spasi seperti apa adanya

Berbeda dengan HTML, XML menampilkan spasi persis bagaimana data ditulis. Lebih jelasnya perhatikan contoh berikut ini:

Pada HTML kalimat

Kami pergi bersama

akan ditampilkan sebagai:

Kami pergi bersama

Sedangkan pada XML akan ditampilkan sama persis dengan kalimat asalnya.

Nilai atribut harus diletakkan diantara tanda petik

Seperti HTML, XML memiliki atribut. Nilai atribut harus diletakkan diantara dua tanda petik. Tidak masalah apakah tanda petik tunggal atau tanda petik ganda. Contoh dibawah ini dua-duanya benar

<pesan dari="lusy"> atau

<pesan dari='lusy'>

Penamaan tag dan atribut

Nama tag bisa terdiri dari huruf, angka dan underscore("_"). Karakter awal nama tag harus berupa huruf atau underscore ("_"), tidak diawali dengan kata xml atau XML, (misal:<xmlstring>), dan tidak mengandung spasi. Aturan penamaan atribut sama dengan aturan penamaan tag.

Menyisipkan komentar

Pada bahasa pemrograman atau scripting kita mengenal adanya komentar (comment). Komentar adalah kalimat/baris yang tidak dieksekusi oleh compiler, browser atau parser. Untuk menyisipkan komentar pada dokumen XML caranya adalah sebagai berikut:

<!—Baris ini tidak di eksekusi oleh parser -->

Menggunakan Karakter Illegal pada XML

Sama seperti pada HTML, anda tidak bisa menggunakan karakter seperti kurung siku (< atau >), petik tunggal (‘), dan petik ganda (“) .

Contoh dibawah ini akan menghasilkan error kalau di eksekusi oleh browser.

<syarat>jika jumlah < 1000 maka</syarat>

Untuk menghindarinya, kita harus menggantikannya dengan entity reference seperti di bawah ini:

<syarat>jika jumlah < 1000 maka</syarat>

Selengkapnya perhatikan tabel dibawah ini:

Entity references	Character	Character Name
<	<	Less Than
>	>	Greater then
&	&	Ampersand
'	‘	Apostrophe
"	“	Quotation mark

Tabel 2.1 Daftar entity reference

Perhatikan bahwa entity reference selalu diawali oleh & dan diakhiri ;

Aturan lainnya adalah dengan mendefinisikan struktur penulisan dari suatu elemen ke dalam DTD (Document Type Definitions). Dalam DTD didefinisikan bagaimana struktur suatu elemen dapat disusun. Elemen apa saja yang dapat menjadi sub dari elemen lainnya. Ditentukan juga bagaimana urutan penulisan suatu elemen terhadap elemen yang lainnya. Attribut apa saja yang dapat dikandung oleh suatu elemen. Berikut contoh dari DTD untuk dokumen XML Buku :

```
<!ELEMENT book (abstract?, preface, chapter*, appendix?)>
<!ELEMENT abstract ...>
<!ELEMENT chapter ...>
<!ATTLIST chapter id   ID   #REQUIRED
          title CDATA #IMPLIED>
```

Dengan DTD penyusunan suatu dokumen XML menjadi lebih terstruktur dan teratur. Dokumen XML yang penyusunannya tidak sesuai dengan DTD nya baik dari urutan maupun elemen yang berada di dalamnya akan dianggap illegal.

2.3.4 Parser

XML Parser atau juga disebut XML prosesor adalah fasilitas untuk melakukan testing terhadap suatu dokumen XML sehingga dapat dihasilkan struktur dokumen dalam bentuk tree.

2.3.4.1 DOM

Document Object Model (DOM) adalah antarmuka yang dikenalkan oleh World Wide Web Consortium (W3C) untuk mengakses dan melakukan modifikasi suatu dokumen XML. Implementasi DOM akan membuat suatu dokumen XML kedalam bentuk struktur pohon sehingga user kemudian dapat melakukan akses terhadap struktur pohon tersebut.

DOM memberikan kemudahan saat kita melakukan akses terhadap suatu xml dokumen secara acak. Dengan sturktur pohon yang merupakan hasil implementasi dari DOM, maka pengaksesan ke masing masing bagian dari pohon tersebut.

Pengolahan dokumen xml dengan dom dapat dimulai dengan melakukan parsing dokumen xml ke dom. Berikut ini operasi operasi terhadap dokumen xml yang dapat dilakukan dengan dom

1. Membuat dokumen xml dari struktur dom yang kita buat
2. Navigasi seluruh cabang pohon

3. Melakukan Penghapusan atau penambahan data pada lokasi struktur tertentu.
4. Pencarian data
5. Penggabungan dua dokumen.

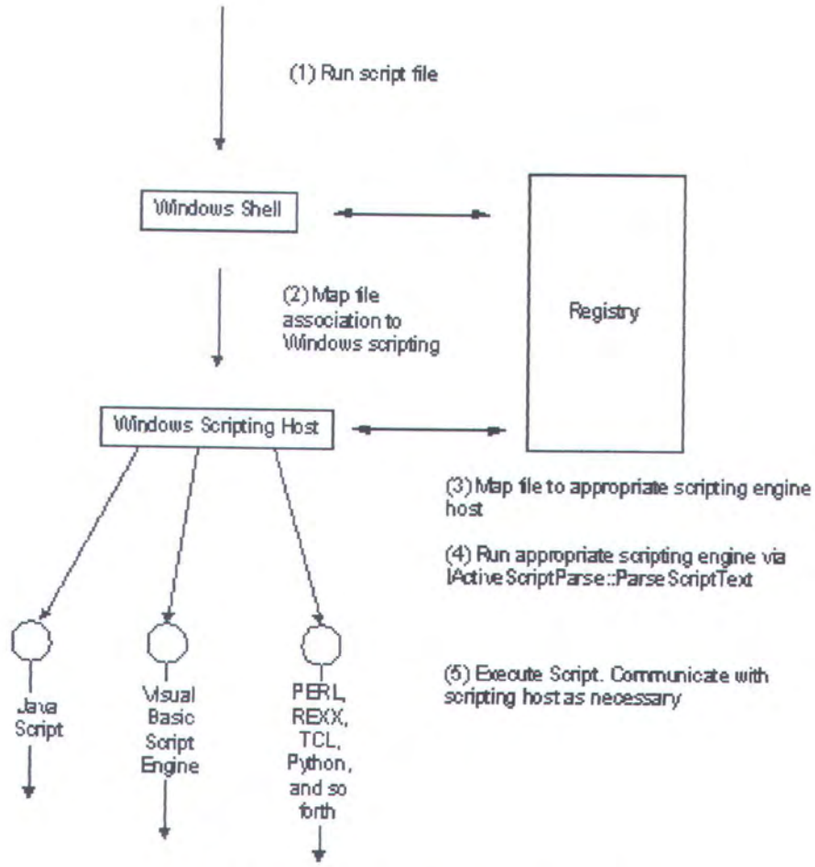
2.4 WINDOWS SCRIPTING HOST

Microsoft Windows Script Host adalah bahasa scripting layaknya shell script pada unix atau bacth file pada dos. Windows Scripting Host menyediakan interface ActiveX untuk melakukan eksekusi suatu skrip fungsi secara langsung. Dengan menggunakan WSH, kita dapat melakukan beberapa pekerjaan berikut yaitu :

- Menampilkan pesan pada layar.
- Menjalan fungsi fungsi dasar semisal membuat jendela baru, form baru, eksekusi program.
- Maping Jaringan
- Koneksi printer.
- Memodifikasi variable sistem
- Memodifikasi nilai registry

Skrip bahasa yang dapat diterima oleh Windows Scripting Host adalah VBScript, JavaScript, Perl, Python, Lua, TCL. Hanya saja penggunaan yang paling umum adalah menggunakan VBScript atau JavaScript.

Diagram Alur eksekusi skrip Windows Scripting Host:



Gambar 2.4 Diagram Alir Windows Scripting Host

2.4.1 TekWshControl

TekWSHControl adalah komponen control yang memungkinkan bagi Delphi untuk menggunakan Windows Script Host. Control ini memungkinkan untuk menjalankan skrip dari suatu aplikasi dan sharing suatu objek dari aplikasi untuk digunakan oleh skrip.

TekWSHControl memungkinkan untuk melakukan management objek dari suatu aplikasi melalui skrip termasuk diantaranya mengakses properti dari objek aplikasi, menjalankan fungsi atau prosedur publik aplikasi juga dapat menambahkan objek baru dan prosedur atau fungsi baru ke dalam aplikasi.

3.2.1 Proses pengepakan data ke dalam XML

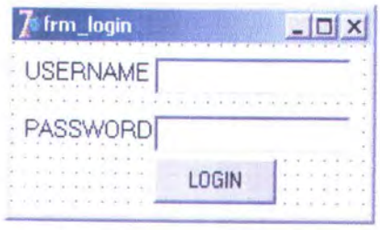
Terdapat tiga proses pengepakan data menjadi XML di sini. Proses yang pertama adalah perubahan data form client dimana pada client delphi maka data form berbentuk file DFM. Data DFM tersebut akan diolah untuk dirubah menjadi data XML-Form. Proses ini dilakukan pada aplikasi pengubah dfm menjadi xml. Output dari aplikasi ini nantinya akan digunakan oleh server sebagai data form yang akan dikirimkan kepada client.

Proses yang kedua adalah pengepakan data dari database yang berbentuk recordset menjadi XML. Proses ini terjadi pada saat aplikasi server membutuhkan data dari database untuk nanti dikirimkan kepada client. Data hasil query yang berbentuk recordset akan dirubah menjadi bentuk xml dan baru akan dikirimkan kepada client.

Sedangkan proses yang ketiga adalah pengepakan script fungsi dari form menjadi XML. Proses ini terjadi saat aplikasi server akan mengirimkan kepada client fungsi yang harus dibangkitkan menyertai form yang akan digenerate secara runtime.

3.2.1.1 Pengepakan Data Form menjadi XML

Data Form client adalah sebuah file dengan ektensi .DFM. File ini pada dasarnya juga merupakan file teks biasa dengan format tertentu. Berikut adalah contoh dari suatu form pada delphi :



Gambar 3.2 Form Login

Form di atas bila kita tampilkan dalam bentuk teks dfm akan menjadi seperti berikut ini :

```
object frm_login: Tfrm_login
  Left = 354
  Top = 233
  AutoScroll = False
  Caption = 'frm_login'
  ClientHeight = 96
  ClientWidth = 220
  Color = clWhite
  Constraints.MinHeight = 50
  Constraints.MinWidth = 130
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  Scaled = False
  WindowState = wsMaximized
  PixelsPerInch = 96
  TextHeight = 13
  object lbl_password: TLabel
    Left = 8
    Top = 40
    Width = 79
    Height = 16
    Caption = 'PASSWORD'
    Color = clWhite
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentColor = False
    ParentFont = False
  end
  object lbl_username: TLabel
    Left = 8
    Top = 8
    Width = 77
    Height = 16
    Caption = 'USERNAME'
    Color = clWhite
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentColor = False
    ParentFont = False
  end
  object txt_username: TEdit
    Left = 88
    Top = 8
    Width = 121
    Height = 21
    TabOrder = 0
  end
  object txt_password: TEdit
    Left = 88
    Top = 40
    Width = 121
    Height = 21
    PasswordChar = '*'
    TabOrder = 1
  end
  object btn_login: TButton
    Left = 88
    Top = 64
    Width = 75
    Height = 25
    Caption = 'LOGIN'
    TabOrder = 2
  end
end
```

BAB 3
PERANCANGAN PERANGKAT LUNAK

Bab ini menjelaskan tentang tahapan proses perancangan perangkat lunak mulai dari deskripsi secara umum sampai perancangan data, perancangan aplikasi dan perancangan antar muka.

3.1 Deskripsi Umum

Dalam tugas akhir ini, akan dibuat suatu sistem yang terdiri dari client, server dan database untuk menunjukkan implementasi dari protokol XML-RPC. Dalam arsitektur sistem ini nantinya akan digambarkan bagaimana pertukaran data yang terjadi pada client, server dan database. Data yang dipertukarkan adalah data yang berupa data XML Form, Data Fungsi dan data dari database. Semua data dipertukarkan dalam bentuk XML.

Data XML Form berisi data data tampilan form yang akan digenerate menjadi form oleh aplikasi client secara runtime. Data fungsi berisi skrip fungsi yang menempel pada form sehingga pada saat form tersebut selesai dibuat maka fungsi fungsi yang menempel pada form tersebut dapat dieksekusi oleh user. Data dari database adalah data yang nantinya akan ditampilkan sebagai informasi dalam form tersebut.

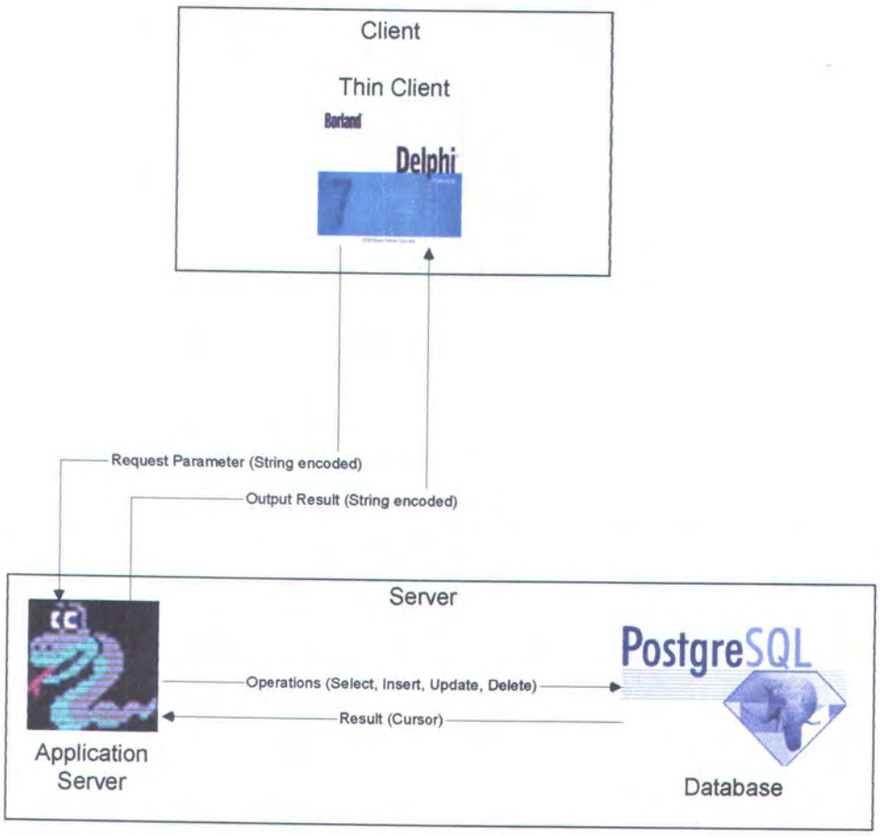
Studi kasus yang diangkat dalam tugas akhir ini adalah Sistem Informasi Ruang Baca Teknik Computer (RBTC).

3.1.1 Arsitektur Sistem

Perangkat lunak yang dibuat dalam tugas akhir ini merupakan aplikasi berbasis desktop (windows application) baik pada client maupun pada servernya. Aplikasi ini membutuhkan minimum satu komputer yang dapat berfungsi sebagai server database, server aplikasi dan client. Namun pada tugas akhir ini, digunakan 2 buah komputer. Satu untuk client, satu lagi untuk database server dan server aplikasi.

Aplikasi yang dibuat terdiri dari aplikasi pengubah dfm menjadi xml, aplikasi server dan aplikasi client. Aplikasi pertama dibuat dengan bahasa pemrograman delphi7. Client berjalan diatas environment windows dengan delphi7 sebagai platformnya, sedangkan server menggunakan python2.4 sebagai platformnya. Untuk database, digunakan postgresql yang juga berjalan pada platform windows..

Arsitektur sistem aplikasi ini dapat dilihat pada gambar 3.1 berikut ini.



Gambar 3.1 Arsitektur Sistem

3.2 Perancangan Proses

Data yang dikirimkan antara client dan server berbentuk XML. Sehingga perlu dilakukan perubahan bentuk data yang ditransmisikan dari bentuk data biasa menjadi bentuk XML. Penjelasan proses pengepakan data menjadi bentuk XML akan diterangkan pada sub bab 3.2.1

Proses selanjutnya adalah bagaimana komunikasi antara client dan server dilakukan. Juga proses komunikasi antara server dan database server untuk melakukan operasi terhadap data tertentu.

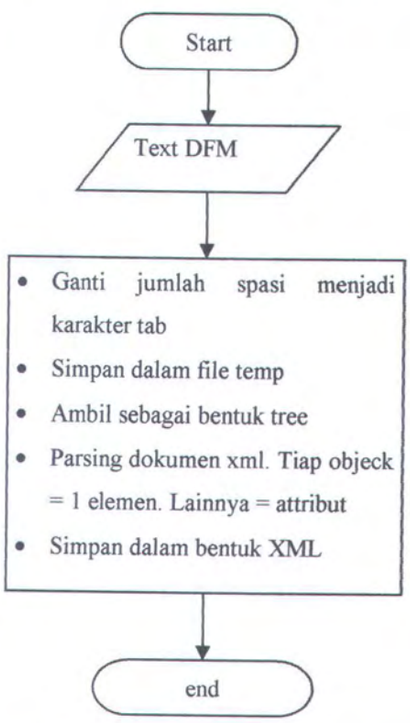
Berikut adalah penjelasan proses proses di atas :

WSH Control memudahkan bila kita ingin mengontrol aplikasi kita melalui skrip sehingga bila kita menginginkan perubahan terhadap aplikasi kita maka kita tidak perlu melakukan compiling ulang.

Alasan mengapa memilih TekWSHControl untuk digunakan pada aplikasi misalnya kita ingin membangun aplikasi hanya saja belum tahu semua fungsi yang harus dipersiapkan. Untuk menangani hal ini cukup kita tambahkan control ekWSHControl dan menambahkan beberapa fungsi untuk melakukan koneksi dengan skrip WSH sehingga kemudian bila ingin menambahkan fungsi baru maka perubahan cukup dilakukan dengan menambahkan skrip yang bisa berasal dari file maupun database. Selanjutnya aplikasi dapat membaca skrip fungsi dan menjalankan skrip tersebut.

Dari bentuk teks diatas terdapat pola bila suatu objek atau control diperlukan pada suatu form maka selalu didahului dengan kata "object". Sedangkan semua nilai properti yang mengikuti objek tersebut diletakkan di bagian bawah dan agak menjorok ke dalam dari objek tersebut.

Pola ini yang kemudian digunakan untuk melakukan parsing terhadap file DFM tersebut sehingga didapatkan hasilnya dalam bentuk XML. Berikut ini adalah gambaran proses yang terjadi terhadap suatu file dfm sehingga dapat diubah menjadi dokumen xml :



Gambar 3.3 Diagram Alir DFM menjadi XML

Pada tugas akhir ini properti yang diproses untuk menjadi atribut dari suatu elemen objek dibatasi hanya nilai properti seperti berikut ini :

- Left Menyatakan posisi kiri suatu objek pada layar.
- Top Menyatakan posisi atas suatu objek pada layar.
- Width Menyatakan lebar/panjang horisontal objek.
- Height Menyatakan tinggi/panjang vertikal objek.
- Name Menyatakan nama objek.
- Caption Menyatakan Sting yang akan ditampilkan pada objek tersebut.
- Color Menyatakan warna dari objek.
- Font.Color Menyatakan warna tulisan objek.
- Font.height Menyatakan besar tulisan objek.
- Font.Name Menyatakan style tulisan objek.
- Clientwidth Menyatakan lebar objek. Utamanya pada editbox.
- Clientheight Menyatakan tinggi objek. Utamanya pada editbox.
- Passwordchar Menyatakan bahwa objek digunakan sebagai tampilan password maka bentuk karakternya harus dirubah menjadi karakter yang kita tentukan. Utamanya pada editbox.
- Wordwrap Menyatakan bahwa text pada objek akan dilakukan wordwrap atau tidak.
- Visible Menyatakan bahwa objek akan ditampilkan pada layar atau tidak.

- ReadOnly Menyatakan bahwa akan dilakukan penguncian pada objek atau tidak.

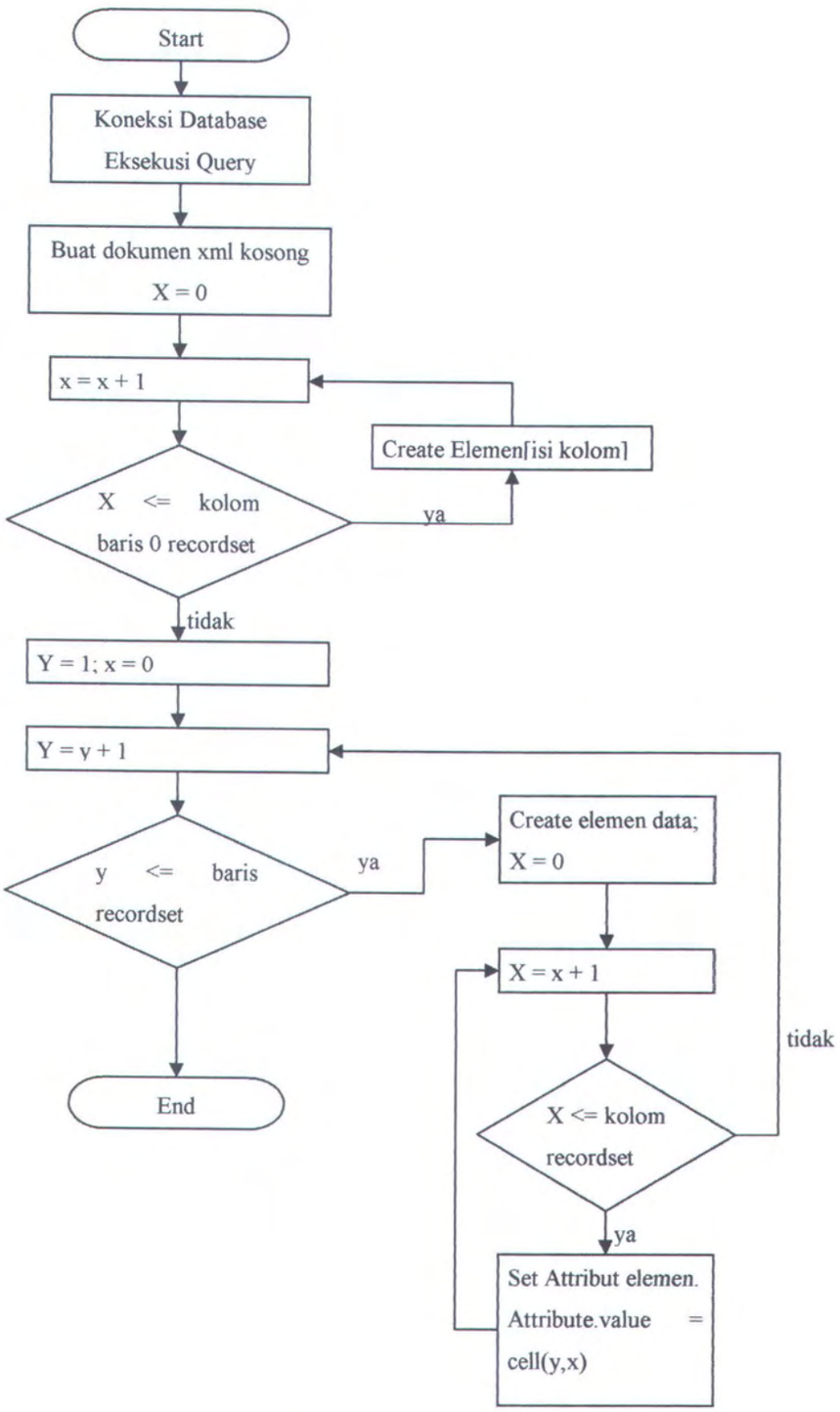
3.2.1.2 Pengepakan data recordset menjadi XML

Recordset yang didapatkan dari hasil query terhadap database pada dasarnya adalah dalam bentuk matriks teks biasa. Pada baris pertama (baris 0) diberikan nilai nilai field kolom hasil query. Kemudian diikuti oleh nilai nilai tiap record untuk tiap baris selanjutnya.

Pemrosesan recordset menjadi dokumen xml pertama dilakukan dengan membuat satu dokumen xml kosong. Selanjutnya adalah membuat elemen yang berisi nilai nilai field kolom dari recordset tersebut.

Setelah nilai nilai header recordset disertakan sebagai elemen dalam doumen xml tersebut, maka yang dilakukan selanjutnya adalah menambahkan satu persatu baris data menjadi satu elemen dengan nilai tiap kolom merupakan atribut yang menyertai elemen tersebut.

Berikut ini adalah diagram alir dari proses pengepakan data recordset menjadi xml:



Gambar 3.4 Diagram Alir Pengepakan data menjadi XML

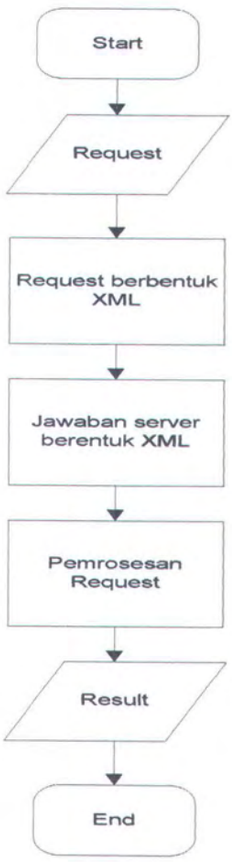
3.2.1.3 Pengepakan Script Fungsi Form menjadi XML

Script fungsi memungkinkan agar form yang dibangun secara runtime dapat memiliki even even yang dapat dieksekusi oleh client. Even even yang mengikuti form tersebut berusaha juga digenerate secara runtime oleh client sehingga kemudian dihasilkan form runtime yang dinamis dan user dapat berinteraksi dengan tampilan antarmuka tersebut.

Script fungsi akan disimpan pada lokasi server dalam bentuk file teks biasa. Kemudian akan diambil oleh server isinya dan kemudian ditambahi sintaks xml sehingga menjadi satu dokumen xml untuk nanti dikirimkan kepada client.

3.2.2 Proses komunikasi antara client dan server

Proses komunikasi antara client dan server dapat digambarkan sebagai berikut.



Gambar 3.5 Diagram alir proses umum

Pada waktu pertama kali berjalan, client hanya memiliki satu fungsi yang dapat dieksekusi oleh user. Fungsi tersebut adalah fungsi untuk meminta request awal kepada server.

Request kepada server berisi fungsi yang diminta kepada server dan parameter yang disertakan. Parameter fungsi tersebut berbentuk XML sehingga pemrosesan parameter menjadi lebih mudah.

Server menerima request dari client dan mengarahkan request kepada fungsi yang diminta. Fungsi tersebut kemudian memproses request

sesuai dengan fungsinya dan parameter parameter yang mengikuti requst tersebut.

Fungsi server tersebut kemudian memberikan jawaban untuk dikirimkan kembali kepada user. Jawaban dari server juga berbentuk xml sehingga client lebih mudah untuk memprosesnya. Jawaban dari server tersebut terdiri dari bentuk form user, fungsi fungsi yang mengikuti form tersebut, dan data data yang akan ditampilkan ke dalam form tersebut.

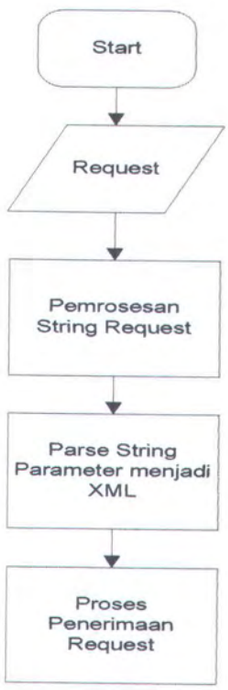
Client selanjutnya melakukan pengolahan jawaban dari server. Pengolahan jawaban terdiri dari tiga fungsi besar yaitu : Fungsi pengolah data form, fungsi pengolah fungsi form, fungsi penampil data database ke dalam form.

Hasil dari pengolahan jawaban server adalah form yang dinamis dengan data data informasi di dalamnya serta fungsi fungsi yang menyertai form runtime tersebut.

Saat user melakukan eksekusi terhadap form runtime tersebut, proses yang terjadi adalah permintaan request kembali kepada server seperti pada saat pertama kali program dijalankan oleh user. Sehingga kemudian akan dihasilkan form runtime baru dengan data baru serta fungsi fungsi baru yang melekat pada form runtime baru tersebut. Alur cerita akan berulang terus menerus sehingga aplikasi dapat berjalan melayani rquest dari user.

3.2.2.1 Proses Pengepakan Request

Gambar berikut menunjukan urutan proses pengepakan request yang dilakukan oleh sistem.



Gambar 3.6 Diagram Alir Proses Pengepakan Request

Request parameter yang masih kasar (raw) dari Client akan disusun menurut aturan fungsi yang ada pada server aplikasi. Fungsi dalam server aplikasi menuntut adanya parameter. Parameter tersebut bisa saja satu atau lebih. Terkadang untuk satu fungsi, bisa terdapat lebih dari sepuluh parameter.

XML-RPC hanya mensupport tipe-tipe data yang umum. Karenanya, tetap dipakai string dalam komunikasi client server. Akibatnya, string yang

ada harus mengalami beberapa modifikasi. Baik yang keluar atau masuk pada client dan server.

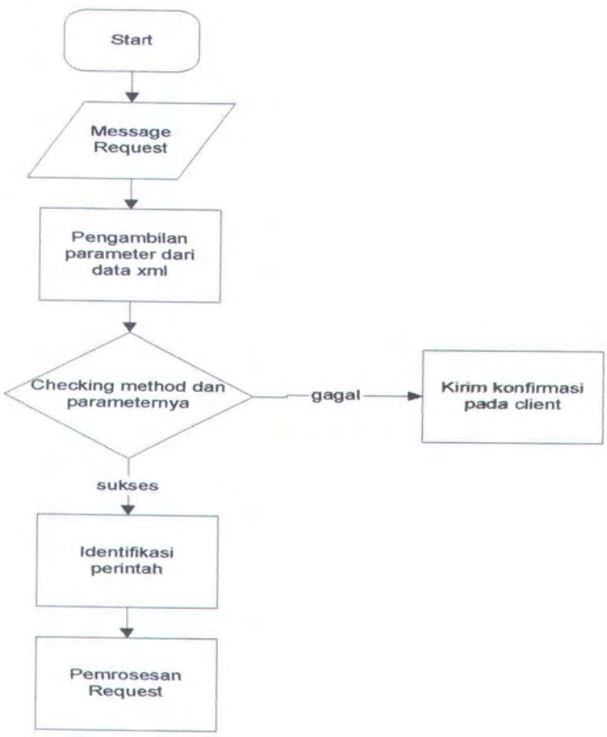
Teknik modifikasi string yang digunakan adalah merubah string parameter menjadi dokumen xml. Parameter berbentuk xml tersebut memiliki format sebagai berikut :

```
<parameter>
    <data isi="isi pertama"/>
    <data isi="isi kedua"/>
    <data isi="isi ketiga"/>
    .....
</parameter>
```

Parameter yang sudah berbentuk string xml tersebut lalu dikirimkan ke server untuk ditangani melalui xml-rpc.

3.2.2.2 Proses Penerimaan Request

Gambar berikut menunjukan urutan proses penerimaan request.



Gambar 3.7 Diagram Alir Proses Penerimaan Request

Paket request dari proses pengepakan dibongkar. Request akan dibaca hostname, port, servlet, method/fungsi dan parameternya. Server menerima hasil akhir berupa fungsi yang dipanggil oleh servlet dan parameter yang mengikuti dalam bentuk dokumen xml.

Parameter dari fungsi yang diinginkan masih dalam bentuk dokumen xml sehingga perlu untuk dilakukan pemrosesan terlebih dahulu. Pemrosesan dilakukan sehingga parameter dari fungsi yang diinginkan

dapat diambil. Bila parameter yang diminta cocok dengan kondisi parameter yang didapatkan dari request tersebut, maka dilanjutkan ke proses pengidentifikasian perintah. Bila gagal, akan dikirim konfirmasi kegagalan pada client.

Fungsi mengeksekusi perintah-perintah yang ada dalam dirinya sesuai dengan parameter-parameter yang ada. Perintah-perintah ini lalu masuk ke proses berikutnya. Proses Pemrosesan Request.

3.2.2.3 Pemrosesan Request

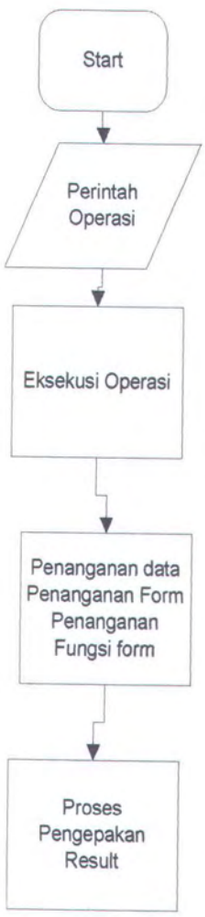
Operasi yang dilakukan oleh server dapat meliputi tiga hal yaitu : penanganan operasi data, penanganan data form runtime, dan penanganan fungsi fungsi dari form yang diinginkan.

Proses penanganan operasi data dilakukan oleh server. Server dapat melakukan operasi penambahan, edit, dan penghapusan data. Operasi lain yang sering dilakukan adalah pengambilan data dari database. Hasil operasi pengambilan data berupa kumpulan record record dengan field yang beragam jumlahnya. Agar data tersebut dapat dikirimkan kembali kepada user, maka data hasil operasi pengambilan dibungkus lagi kedalam bentuk xml. Operasi ini menggunakan fungsi sendiri yang berada di dalam server.

Proses penanganan data form runtime dilakukan dengan mengambil data form yang terimpan di dalam file. Data form sudah berbentuk dokumen xml hasil dari proses pengepakan data form menjadi xml. Data hasil parser tersebut ditambahi beberapa tag fungsi yang nanti akan dibangkitkan eventnya pada saat generate form secara runtime.

Proses penanganan data fungsi fungsi dari form runtime yang diinginkan juga dilakukan dengan membaca isi file fungsi yang tersimpan di lokasi server. Perlu pemrosesan lebih lanjut untuk membuat fungsi fungsi tersebut dirubah ke dalam bentuk xml sehingga dapat dikirimkan kepada user melalui protocol xml-rpc.

Gambar berikut menunjukan urutan proses pemrosesan request

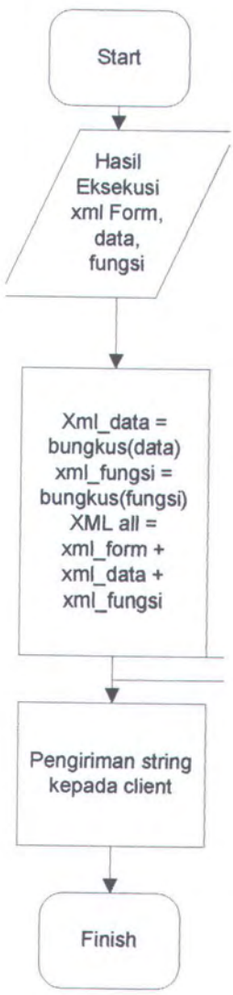


Gambar 3.8 Diagram Alir Pemrosesan Request

3.2.2.4 Proses Pengepakan Result

Hasil dari operasi fungsi server dapat terdiri dari tiga bagian yaitu hasil data dari database, data form dan fungsi form. Hasil fungsi server tersebut lalu digabungkan ke dalam bentuk dokumen xml baru untuk dikirimkan kembali sebagai jawaban dari server.

Berikut diagram alir dari operasi di atas :



Gambar 3.9 Diagram Alir Pengepakan Result

3.2.2.5 Proses pembacaan hasil

Hasil request adalah jawaban dari server yang berbentuk dokumen xml. Dokumen tersebut dapat terdiri dari tiga macam yaitu : Data Form XML, data fungsi form, dan data dari database. Memerlukan penanganan yang berbeda untuk tiap macam jawaban di atas.

Untuk jawaban berupa data form xml, yang dilakukan client selanjutnya adalah melakukan generate form baru secara runtime berdasar dari data form yang ada. Pembacaan jawaban dilakukan sehingga dokumen dapat dibaca dari awal elemen hingga akhir.

Contoh jawaban berupa data form xml adalah sebagai berikut:

```
<Document>
 <Tfrm_login Name="frm_login" Left="192" Top="103" Width="452" Height="180" Caption="frm_login" Color="12040539">
  <TLabel Name="lbl_username" Left="8" Top="24" Width="61" Height="13" Caption="USERNAME" Color="14211494"/>
  <TLabel Name="lbl_password" Left="8" Top="56" Width="63" Height="13" Caption="PASSWORD" Color="14211494"/>
  <TLabel Name="lbl_keterangan" Left="216" Top="24" Width="215" Height="26" Caption="CONTOH FORM" Color="14211494"/>
  <TEdit Name="txt_username" Left="88" Top="24" Width="121" Height="21"/>
  <TEdit Name="txt_password" Left="88" Top="56" Width="121" Height="21"/>
  <TButton Name="btn_login" Left="88" Top="88" Width="75" Height="25" Caption="LOGIN">
   <OnClick/>
  </TButton>
 </Tfrm_login>
</Document>
```

Gambar 3.10 Contoh Data Form XML

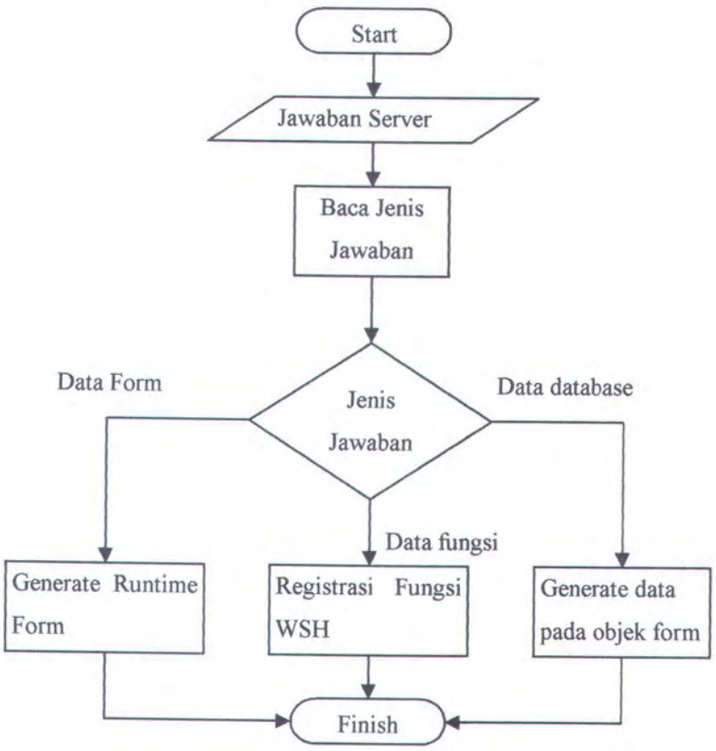
Dari pembacaan diketahui nama nama dari tiap elemen yang di traverse. Dari nama nama tiap elemen itu lalu dilakukan generate objek

secara runtime pada client. Contohnya bila nama elemen sama dengan "TFrm" maka dilakukan generate form baru. Bila nama elemen sama dengan "Tbutton" maka dilakukan generate button secara runtime pada form terakhir yang aktif.

Untuk jawaban berupa data fungsi, maka yang dilakukan oleh client adalah melakukan registrasi pada komponen Windows Script Host (WSH) supaya fungsi tersebut dapat dieksekusi oleh form yang memerlukan.

Untuk jawaban berupa data dari database, maka yang dilakukan oleh client adalah dengan melakukan generate secara runtime tampilan yang perlu dirubah pada objek yang diinginkan. Contoh : Bila data dari database ingin ditampilkan di suatu label maka label tersebut lalu dirubah captionnya dengan data xml tersebut. Contoh lain bila data dari database ingin ditampilkan di suatu stringgrid maka client melakukan pengisian data database ke dalam stringgrid secara runtime.

Berikut ini diagram alir dari pembacaan hasil :



Gambar 3.11 Diagram alir pembacaan hasil

3.3 Perancangan Data

Implementasi dari sistem menggunakan Ruang Baca Teknik Computer sebagai studi kasus. Storage (database) yang digunakan merujuk pada database yang sudah ada yang dimiliki oleh RBTC. Namun begitu, terjadi perpindahan database. Bila pada RBTC digunakan SQL Server 2000, sistem menggunakan PostgreSQL 8.1.

Seperti yang sudah dijelaskan dalam perancangan proses, storage dan client dipisahkan oleh Server aplikasi. Sehingga client tidak dapat melakukan akses secara langsung ke database. Client mengirimkan message untuk selanjutnya dieksekusi oleh fungsi dalam server aplikasi.

3.3.1 Struktur Message Client

Secara eksplisit, kita tidak dapat melihat message dengan jelas. Namun strukturnya dapat dilihat. Secara garis besar, message dibagi dua. Definisi alamat, dan definisi parameter. Dengan analogi sebuah surat, diperlukan alamat dan isi surat untuk menjadikan surat tersebut layak disebut sebagai sebuah surat. Namun disini terdapat perbedaan. Dalam sistem, pendefinisian alamat dapat dilakukan sekali saja selama proses koneksi. Dalam artian, bila koneksi antara client dan server tidak terputus, maka tidak perlu melakukan pendefinisian alamat lagi. Cukup pendefinisian parameter saja.

3.3.1.1 Definisi Alamat

Pertama-tama didefinisikan alamat dari server aplikasi yang dituju. Alamat tersebut memiliki struktur yang bervariasi antara satu platform dengan platform yang lainnya. Namun semua platform memiliki standarisasi yang harus ada. Standarisasi tersebut adalah: HostName, HostPort dan EndPoint.

HostName adalah alamat ip dari server aplikasi. HostPort adalah port yang dibuka oleh server untuk digunakan berkomunikasi. Sedangkan EndPoint adalah detil letak server aplikasi.

Pada implementasi sistem, alamat ditulis sebagai berikut:

- HostName: 10.126.13.88
- HostPort: 80
- EndPoint: /WK/server.py

3.3.1.2 Definisi Parameter

Parameter dapat pula disebut sebagai isi message. Isi yang dimaksud sebenarnya hanyalah nama fungsi yang akan diakses. Apakah fungsi tersebut membutuhkan parameter atau tidak, hal itu bersifat optional. Tergantung bagaimana pendefinisian fungsi dalam server aplikasi.

Seperti yang sudah dijelaskan pada perancangan proses, terdapat teknik pemformatan string request parameter. Pemformatan itu dilakukan sebelum parameter dimasukkan dalam message. Hal ini perlu dilakukan karena message hanya bisa diisi oleh satu parameter saja. Untuk menghemat koneksi yang terjadi, maka parameter dapat dikelompokkan menjadi satu buah dokumen xml.

Sebagai permisalan, terdapat sebuah fungsi X yang memiliki tiga parameter. Maka client menggabungkan ke tiga parameter tersebut menjadi satu dokumen xml. Baru kemudian dokumen tersebut dikirimkan ke server untuk diproses lebih lanjut.

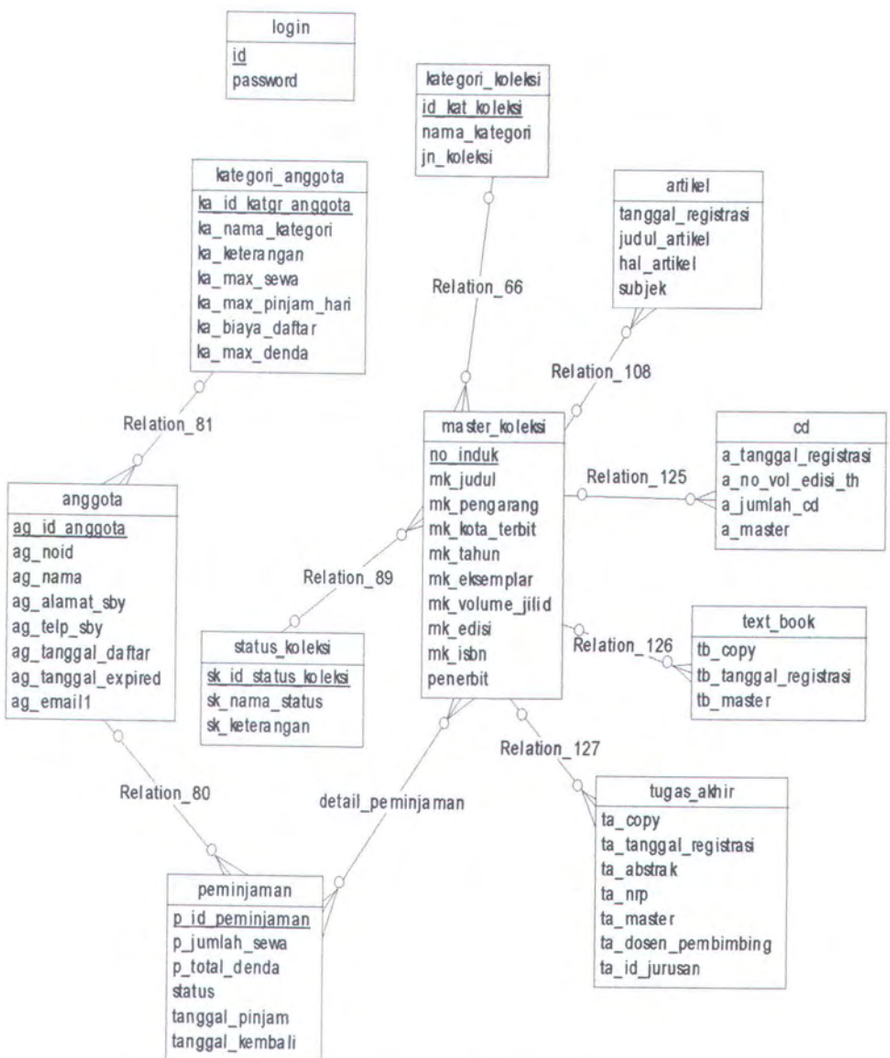
3.3.2 Struktur Message Server

Saat Client membuka sebuah koneksi dengan server, otomatis server terhubung dengan client. Karenanya, server tidak perlu mengidentifikasikan alamat client. Namun demikian, semua message mempunyai struktur message yang sama. Jadi, didalam message yang dikeluarkan server juga terdapat ip address dan port yang akan dituju. Namun tidak perlu didefinisikan secara eksplisit. Protokol secara otomatis akan menggenerate-

nya. Server hanya perlu membuat sebuah dokumen xml untuk dikirimkan kepada client sebagai sebuah output result.

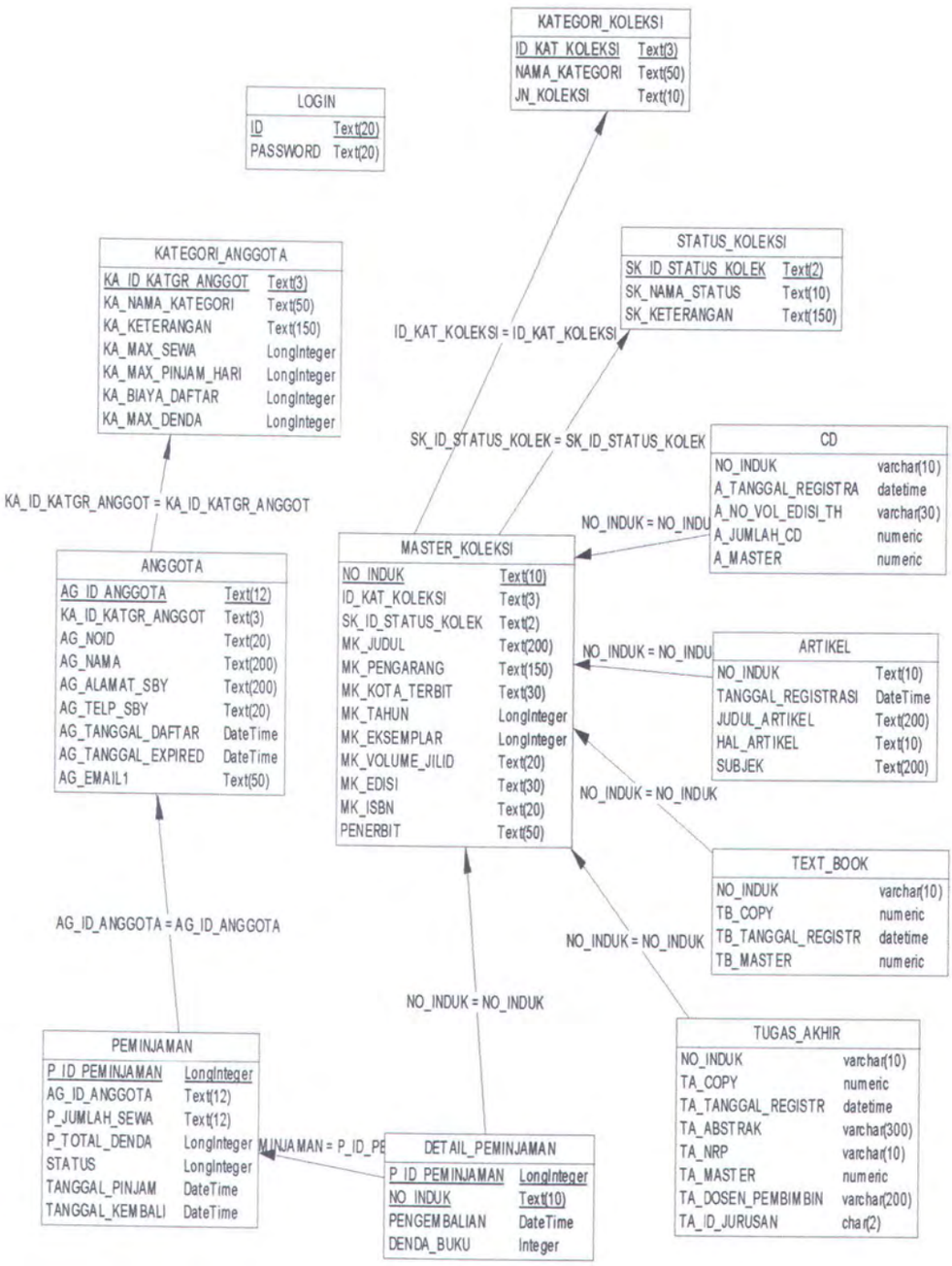
3.3.3 Struktur Database

Berikut adalah gambar CDM dari database sistem.



Gambar 3.12 Gambar CDM

Berikut ini adalah PDM database hasil generate dari CDM yang telah dibuat pada Power Designer 9.

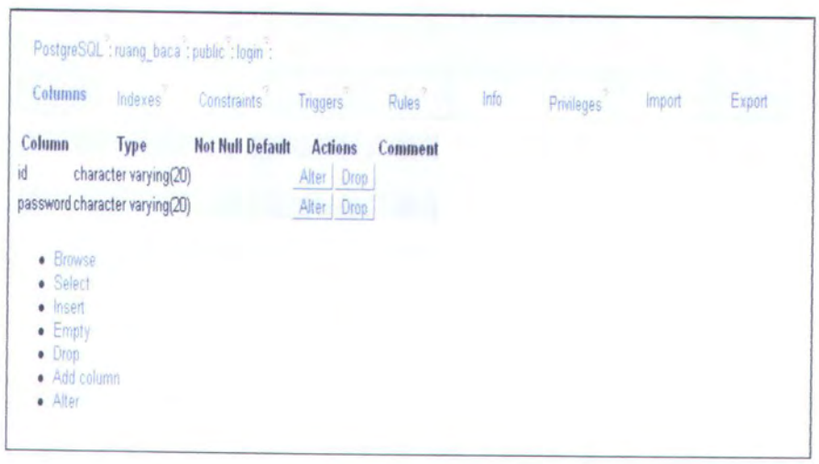


Gambar 3.13 Gambar PDM

Berikut ini adalah struktur database yang ada di dalam PostgreSQL:

1. Tabel login

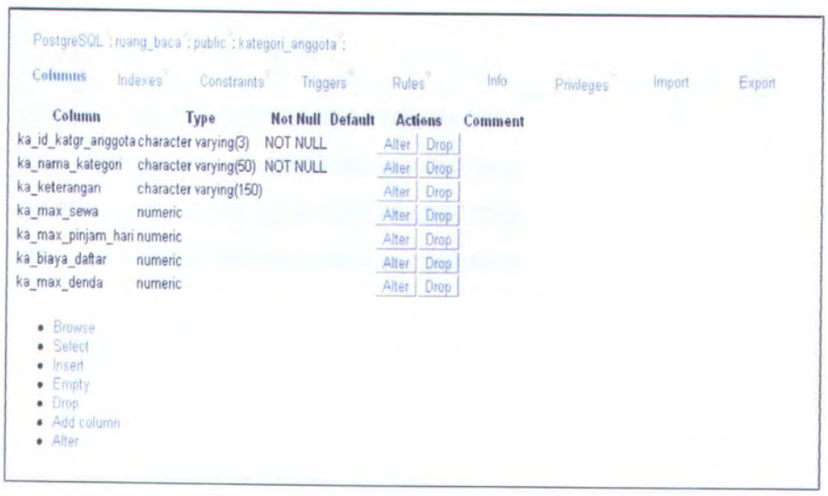
Menyimpan data username dan password dari user yang dapat mengakses Thin Client.



Gambar 3.14 Tipe data tabel login

2. Tabel kategori_ anggota

Berisi pembedaan tipe anggota dalam ruang baca. Tiap kategori mempunyai data meliputi jumlah maksimal peminjaman, biaya pendaftaran dan maksimal denda.


PostgreSQL : ruang_baca : public : kategori_anggota :

Columns | Indexes | Constraints | Triggers | Rules | Info | Privileges | Import | Export

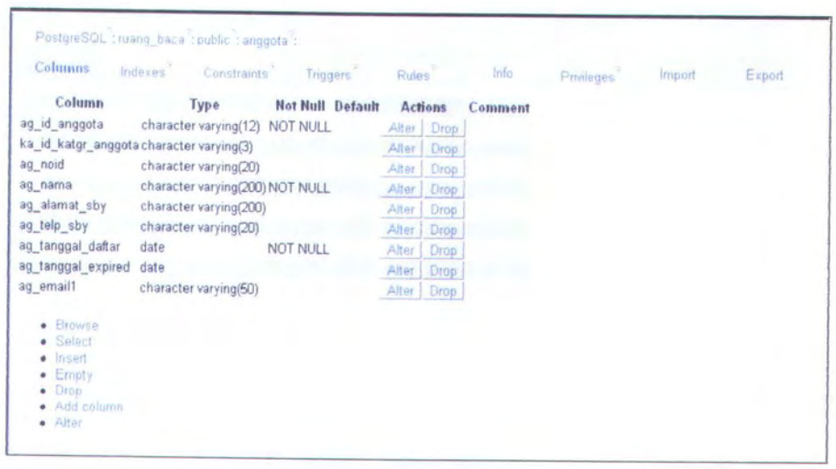
Column	Type	Not Null	Default	Actions	Comment
ka_id_katgr_anggota	character varying(3)	NOT NULL		Alter Drop	
ka_nama_kategori	character varying(50)	NOT NULL		Alter Drop	
ka_keterangan	character varying(150)			Alter Drop	
ka_max_sewa	numeric			Alter Drop	
ka_max_pinjam_hari	numeric			Alter Drop	
ka_biaya_daftar	numeric			Alter Drop	
ka_max_denda	numeric			Alter Drop	

- Browse
- Select
- Insert
- Empty
- Drop
- Add column
- Alter

Gambar 3.15 Tipe data tabel kategori_anggota

3. Tabel anggota

Berisi daftar anggota ruang baca. Data yang ada meliputi nama, NRP/NIM, kategori, alamat, nomor telepon, tanggal pendaftaran, tanggal expired, email.


PostgreSQL : ruang_baca : public : anggota :

Columns | Indexes | Constraints | Triggers | Rules | Info | Privileges | Import | Export

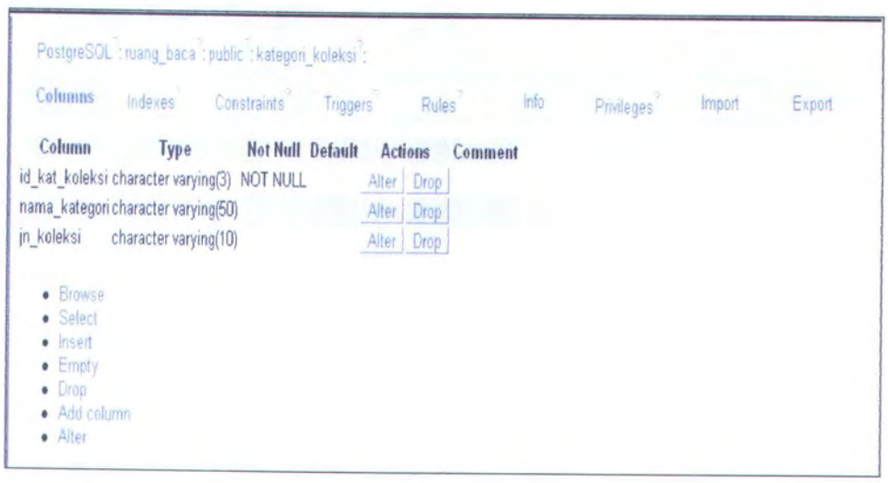
Column	Type	Not Null	Default	Actions	Comment
ag_id_anggota	character varying(12)	NOT NULL		Alter Drop	
ka_id_katgr_anggota	character varying(3)			Alter Drop	
ag_noid	character varying(20)			Alter Drop	
ag_nama	character varying(200)	NOT NULL		Alter Drop	
ag_alamat_sby	character varying(200)			Alter Drop	
ag_telp_sby	character varying(20)			Alter Drop	
ag_tanggal_daftar	date	NOT NULL		Alter Drop	
ag_tanggal_expired	date			Alter Drop	
ag_email1	character varying(50)			Alter Drop	

- Browse
- Select
- Insert
- Empty
- Drop
- Add column
- Alter

Gambar 3.15 Tipe data tabel anggota

4. Tabel kategori_koleksi

Berisi kategori-kategori dari koleksi yang dimiliki ruang baca.


PostgreSQL : ruang_baca : public : kategori_koleksi :

Columns | Indexes | Constraints | Triggers | Rules | Info | Privileges | Import | Export

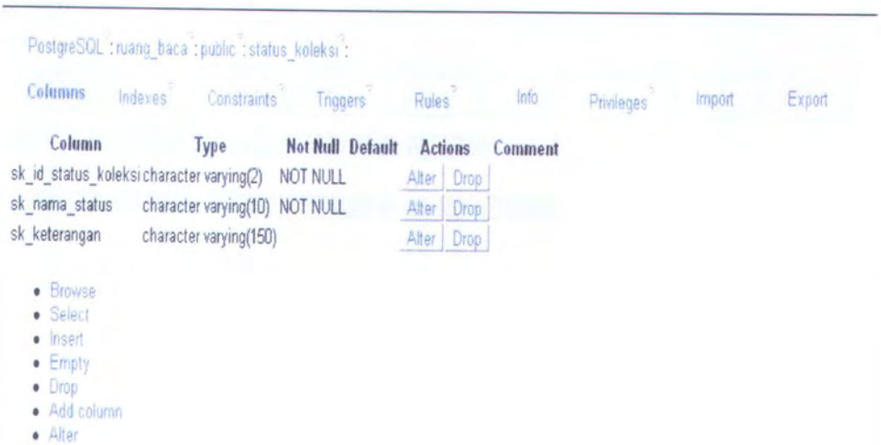
Column	Type	Not Null	Default	Actions	Comment
id_kat_koleksi	character varying(3)	NOT NULL		Alter Drop	
nama_kategori	character varying(50)			Alter Drop	
jn_koleksi	character varying(10)			Alter Drop	

- Browse
- Select
- Insert
- Empty
- Drop
- Add column
- Alter

Gambar 3.16 Tipe data tabel kategori_koleksi

5. Tabel status_koleksi

Tabel ini merupakan tabel yang menyimpan status dari sebuah koleksi.


PostgreSQL : ruang_baca : public : status_koleksi :

Columns | Indexes | Constraints | Triggers | Rules | Info | Privileges | Import | Export

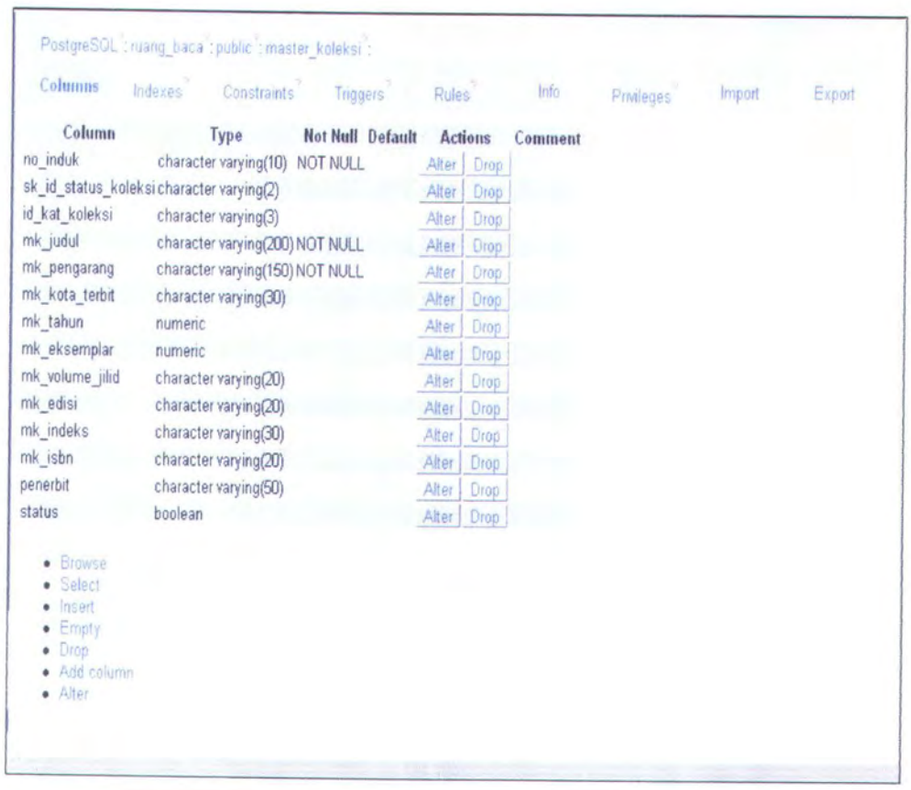
Column	Type	Not Null	Default	Actions	Comment
sk_id_status_koleksi	character varying(2)	NOT NULL		Alter Drop	
sk_nama_status	character varying(10)	NOT NULL		Alter Drop	
sk_keterangan	character varying(150)			Alter Drop	

- Browse
- Select
- Insert
- Empty
- Drop
- Add column
- Alter

Gambar 3.17 Tipe data tabel status_koleksi

6. Tabel master_koleksi

Berisi daftar koleksi yang dimiliki ruang baca.



PostgreSQL : ruang_baca : public : master_koleksi :

Columns | Indexes | Constraints | Triggers | Rules | Info | Privileges | Import | Export

Column	Type	Not Null	Default	Actions	Comment
no_induk	character varying(10)	NOT NULL		Alter Drop	
sk_id_status_koleksi	character varying(2)			Alter Drop	
id_kat_koleksi	character varying(3)			Alter Drop	
mk_judul	character varying(200)	NOT NULL		Alter Drop	
mk_pengarang	character varying(150)	NOT NULL		Alter Drop	
mk_kota_terbit	character varying(30)			Alter Drop	
mk_tahun	numeric			Alter Drop	
mk_eksemplar	numeric			Alter Drop	
mk_volume_jilid	character varying(20)			Alter Drop	
mk_edisi	character varying(20)			Alter Drop	
mk_indeks	character varying(30)			Alter Drop	
mk_isbn	character varying(20)			Alter Drop	
penerbit	character varying(50)			Alter Drop	
status	boolean			Alter Drop	

- Browse
- Select
- Insert
- Empty
- Drop
- Add column
- Alter

Gambar 3.18 Tipe data tabel master_koleksi

3.4 Perancangan Antar Muka

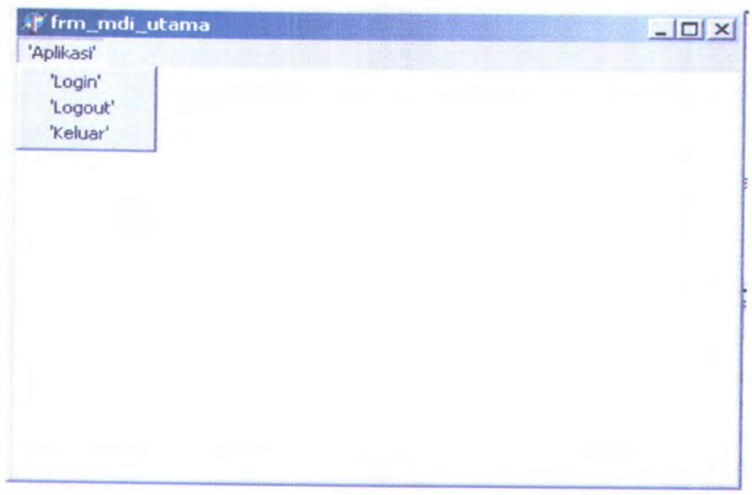
Dalam sistem yang dibuat, yang berhubungan langsung dengan user adalah thin client. Karenanya, interface hanya dibuat pada sisi Thin Client saja. Untuk server aplikasi, tidak diberi interface.

Interface pada client digenerate secara runtime saat program client berjalan dengan mengambil nilai nilai data form dari server.

3.4.1 Perancangan Antarmuka Client

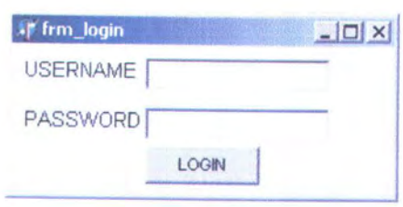
Pada aplikasi ini ada beberapa form yang dibangun secara runtime yaitu:

1. Halaman Utama : Form ini mempunyai menu menu yang digenerate secara runtime. Berikut ini merupakan gambar dari halaman utama :



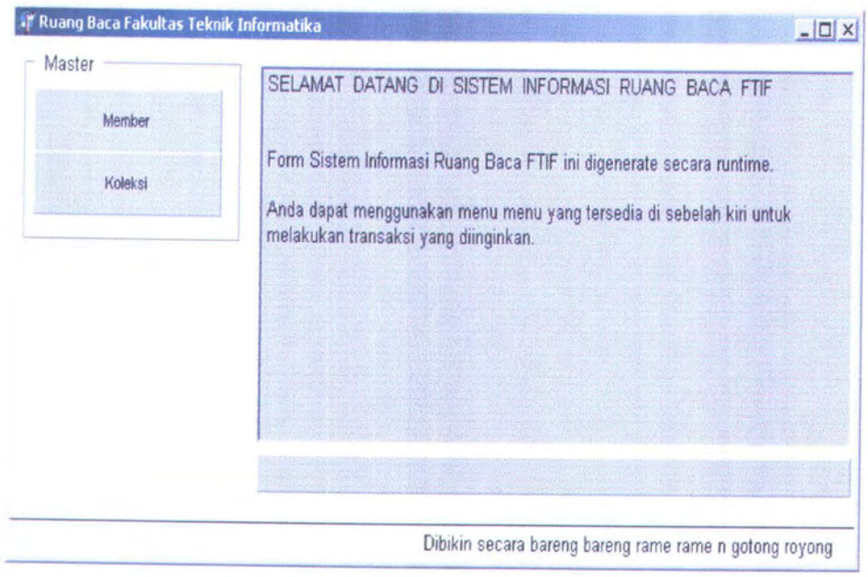
Gambar 3.19 Halaman Login

2. Halaman Login : Halaman yang mempunyai fungsi mengecek validasi dari seorang user. Berikut ini merupakan gambar dari halaman login.



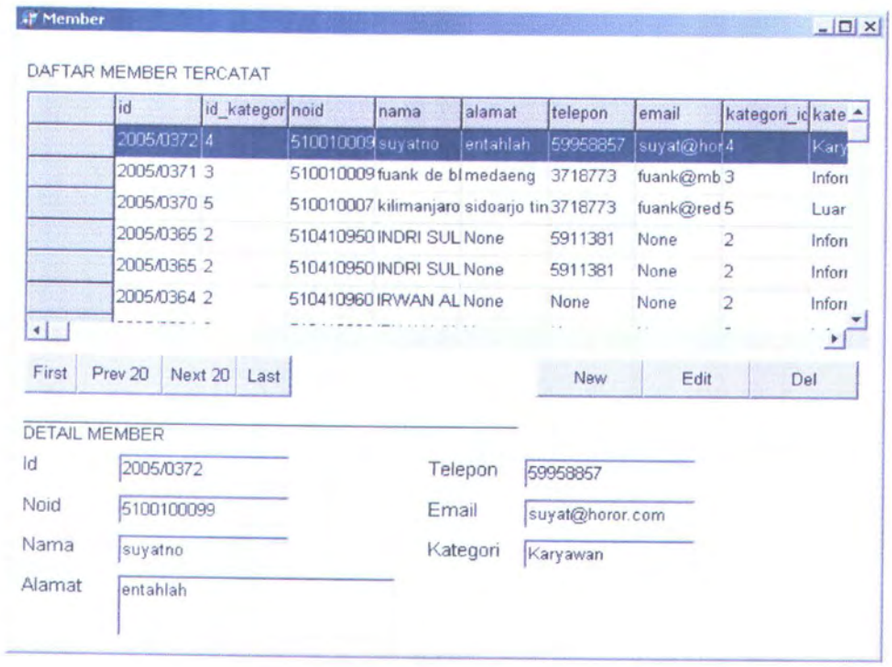
Gambar 3.20 Halaman Login

3. Halaman sapaan : Halaman ini merupakan form yang akan ditampilkan saat seorang user telah berhasil login. Berikut adalah tampilan dari halaman sapaan tersebut :



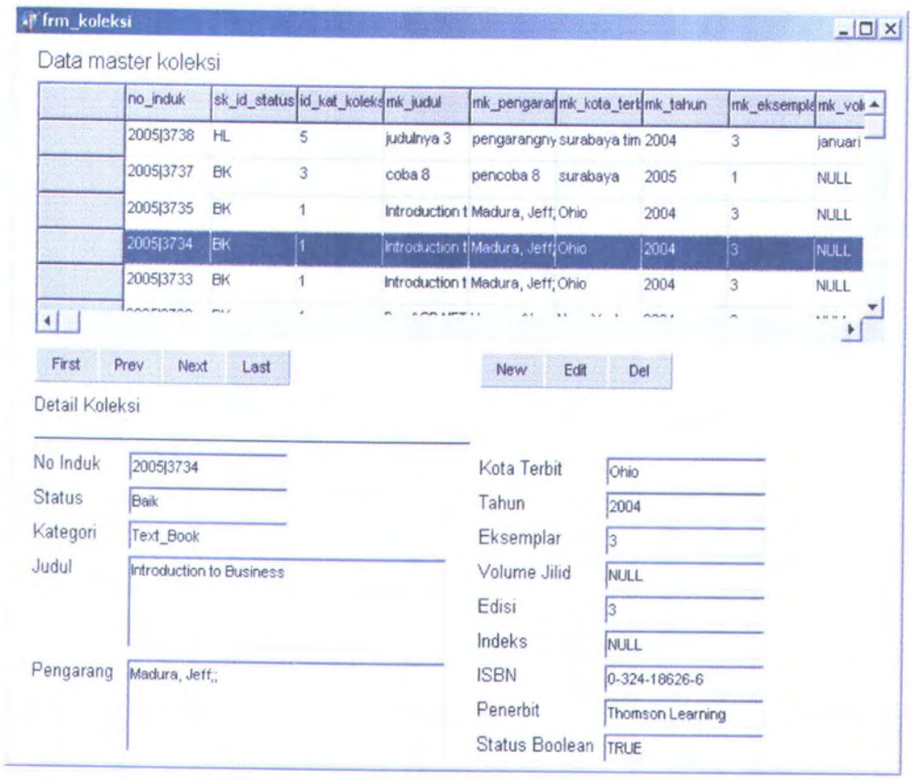
Gambar 3.21 Form Ruang 1

4. Halaman Member : Halaman yang mempunyai fungsi untuk menampilkan data member-member yang terdaftar pada ruang baca. Pada halaman ini jugalah data member dapat ditambahkan, di-edit atau dihapus.



Gambar 3.22 Halaman Member

5. Halaman Koleksi: Halaman yang mempunyai fungsi untuk menampilkan daftar koleksi pada ruang baca. Pada halaman ini dapat dilakukan penambahan koleksi, pengeditan atau menghapus sebuah koleksi. Berikut ini merupakan gambar dari halaman Koleksi.



Gambar 3.23 Halaman Koleksi

BAB 4
IMPLEMENTASI PERANGKAT LUNAK

Berdasarkan perancangan aplikasi yang telah dijelaskan pada bab 3, maka pada bab ini akan dilakukan pembuatan aplikasi dengan mengacu pada perancangan sistem yang sudah ada. Pada sub bab 4.1 akan dijelaskan mengenai implementasi aplikasi pengubah dfm menjadi xml.

Pada bab ini juga akan dijelaskan bagaimana melakukan pengaturan webserver sehingga aplikasi server dapat berjalan dengan menggunakan protokol xmlrpc pada platform web.

Pada sub bab berikutnya akan dijelaskan bagaimana proses komunikasi antara client dan server dapat berjalan. Juga akan dijelaskan bagaimana client dapat melakukan generate form secara runtime dan berikut juga menambahkan fungsi serta data recordset pada form runtime tersebut.

Aplikasi yang dibuat terbagi menjadi tiga aplikasi yaitu aplikasi server, aplikasi client dan aplikasi pengubah DFM menjadi XML.

4.1. IMPLEMENTASI APLIKASI PENGUBAH DFM MENJADI XML

Aplikasi pengubah dfm menjadi xml bekerja dengan inputan berupa file dfm yang akan diubah. File tersebut lalu dibaca dan dilakukan parsing sehingga hasil akhir berupa dokumen xml dihasilkan.

Pertama kali file dfm dibaca sebagai file teks biasa. Kemudian dari tiap baris file dfm tersebut dibaca jumlah spasinya. Setiap ditemui dua spasi maka akan digantikan dengan satu tab. Hasil dari penggantian spasi dengan

tab lalu ditampilkan dalam suatu listbox dan juga disimpan dalam file temporary.

Berikut ini pseudocode proses pembacaan dfm dan penambahan tab :

```
If (fileopen = sukses) then
Begin
        AssignFile(tulisan, nama_file); // menampung isi file sebagai tulisan
        while (not EOF(tulisan)) do
        begin
                readln(tulisan,s) // membaca per baris sebagai s
                i := 1
                while (s[i] = ' ') do
                begin
                        i := i + 1;
                end;                    // baca jumlah spasi sebagai i
        end;
        For j := 1 to round((i-1)/2) do
        Begin
                tabnya := tabnya + #9;  // rubah 2 spasi menjadi 1 tab
        end;
        s := tabnya + RightStr(s,length(s) - i + 1); //tambahkan tab pada string tiap baris.
End;
```

Proses selanjutnya adalah memanggil file temporary dengan langsung melakukan parsing kedalam bentuk treeview. Disini akan terlihat manfaat dari penggantian spasi dengan tab karena bila tidak diganti maka treeviewnya tidak dapat dihasilkan. Sebah dokumen xml kosong dibuat untuk menampung hasil parsing dari treeview.

Dari treeview lalu dilakukan traverse dari bagian root tree hingga akhir dan dilakukan parsing untuk tiap tiap nodenya. Bila node diawali dengan kata "object" berarti kita harus menambahkan satu elemen dalam dokumen xml. Sedangkan bila tidak diawali "object" berarti menandakan bahwa node tersebut merupakan atribut dari elemen yang sebelumnya telah dibuat. Proses parsing akan berakhir bila telah menemui bagian paling akhir dari treeview dan hasil parsing akan disimpan dalam bentuk file xml.

Di bawah ini adalah pseudo code dari proses parsing tree menjadi xml tersebut :

```
Procedure Tree2XML(tree_awal)
Begin
    Xmldoc.create()
    Baca node paling atas
    while anak not nil then
    Begin
        Create node anak
        Xmldoc.addchild(tree,node anak)
        Recursive fungsi dengan tree anak
        Anak = anak.nextsibling
    End
End
```

Berikut adalah hasil parsing dari halaman login dari contoh dfm pada bab 3 :

```
<Document>
        <Tfrm_login Name="frm_login" Left="201" Top="134" Caption="'frm_login'"
ClientHeight="96" ClientWidth="250" Color="clWhite" Font.Color="clWindowText"
Font.Height="-11" Font.Name="'MS Sans Serif'">
                <TLabel Name="lbl_password" Left="8" Top="40" Width="79" Height="16"
        Caption="'PASSWORD'" Color="clWhite" Font.Color="clWindowText"
        Font.Height="-13" Font.Name="'MS Sans Serif'"/>
                <TLabel Name="lbl_username" Left="8" Top="8" Width="77" Height="16"
        Caption="'USERNAME'" Color="clWhite" Font.Color="clWindowText"
        Font.Height="-13" Font.Name="'MS Sans Serif'"/>
                <TEdit Name="txt_username" Left="88" Top="8" Width="121" Height="21"/>
                <TEdit Name="txt_password" Left="88" Top="40" Width="121" Height="21"
        PasswordChar="'*'"/>
                <TButton Name="btn_login" Left="88" Top="64" Width="75" Height="25"
        Caption="'LOGIN'"/>
        </Tfrm_login>
</Document>
```

Supaya form xml tersebut nantinya pada saat generate secara runtime dapat memiliki event event tertentu, maka penambahan secara manual diperlukan pada xml hasil parsing dengan elemen tertentu.

4.2. KONFIGURASI WEBSERVER (APACHE)

Supaya apache dapat berkomunikasi dengan Server aplikasi (webware), perlu ditambahkan beberapa konfigurasi. Komunikasi antara apache dengan webware dilakukan melalui media modul webware yang harus dikompilasi terlebih dahulu supaya dikenali oleh apache.

4.2.1 Kompilasi modul WebKit – Webware

Supaya modul Webkit dapat diinstall pada komputer server, maka webware mensyaratkan komputer tersebut harus sudah terinstal webserver (apache) dan python.

Proses instalasi cukup dilakukan dengan menjalankan perintah :

```
> cd \path\to\Webware
> python install.py
(enter password)
```

Setting webware juga bisa ditambahkan dengan melakukan perubahan direktori kerja kita. Perintah yang digunakan adalah sebagai berikut :

```
$ python Webware/bin/MakeAppWorkDir -l --
cvsignore -c context DIRECTORYNAME
```

Perintah di atas akan menghasilkan DIRECTORYNAME yang berisi struktur dari server aplikasi kita. Pada direktori tersebut terdapat direktori kerja dimana kita dapat menempatkan file aplikasi server kita. Pada direktori tersebut juga terdapat satu file launch.py yang akan mengaktifkan webware bila dijalankan. Supaya aplikasi server dapat berjalan maka file launch.py harus dijalankan.

4.2.2 Konfigurasi Apache

Webware yang sudah terinstall harus diintegrasikan dengan apache sehingga request yang masuk dapat diarahkan oleh apache kepada webware untuk ditangani oleh aplikasi server.

Langkah langkah yang harus dilakukan antara lain adalah :

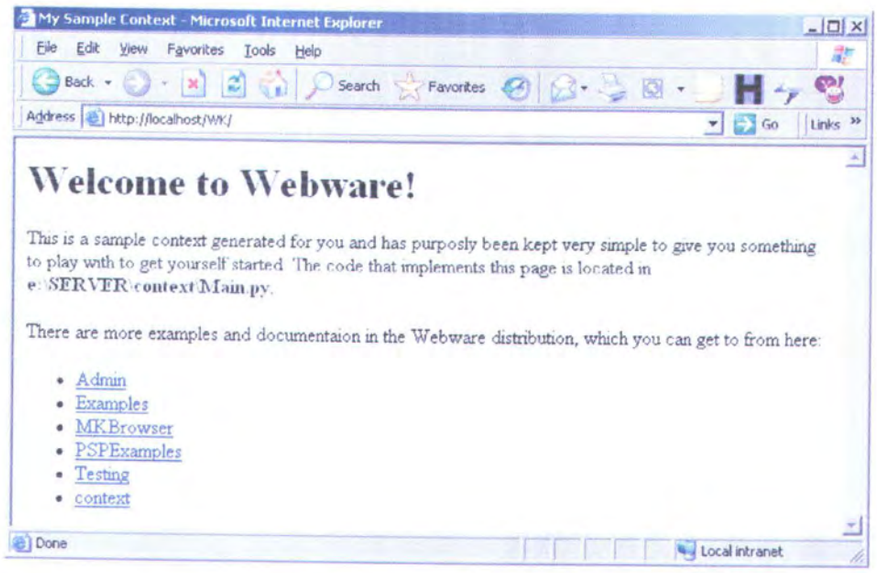
- Kita copy file mod_webkit.dll dari direktori webware (umumnya : WebKit/Adapters/mod_webkit/mod_webkit.dll) ke direktori module dari apache (umunya : Apache Group\Apache\modules).
- Edit hhtp.conf dari apache dengan baris baris berikut :

```
LoadModule                              webkit_module
modules/mod_webkit.dll
<Location /WK>
        WKServer localhost 8086
        SetHandler webkit-handler
</Location>
AddHandler psp-handler .psp
Action py-serverpages /WK/
AddType py-serverpages .py
```

- Restart Apache.

Dengan setting seperti ini, apache yang dijalankan sudah bisa berkomunikasi dengan server aplikasi yang berada pada host yang ditunjukkan pada keyword localhost. Jika server aplikasi berada pada mesin atau komputer yang berbeda, keyword localhost harus diganti dengan IP address atau alamat dari mesin yang bersangkutan. Aturan tersebut juga berlaku pada setingan port, bisa diubah-ubah sesuai dengan kebutuhan.

Testing dapat dilakukan untuk mengecek apakah webware telah berjalan dan terintegrasi dengan apache atau tidak. Testing dilakukan dengan memasukkan URL http://10.126.13.88/WK/. Bila webware berjalan dengan baik maka akan menghasilkan output seperti berikut :



Gambar 4.1 Hasil Testing Webware Sukses

4.3. IMPLEMENTASI SERVER APLIKASI (SERVER APLIKASI)

Terdapat beberapa proses yang terjadi pada server diantaranya adalah proses koneksi database, proses pengepakan data menjadi xml dan proses pengepakan fungsi menjadi xml.

Aplikasi server juga melakukan proses pembacaan file xml-form dan file script fungsi yang lokasinya dalam tugas akhir ini berada pada pada lokasi yang sama dengan aplikasi server tersebut.

4.3.1 Pembacaan file xml-form dan file script fungsi

File xml-form berisi dokumen xml hasil dari aplikasi pengubah dfm menjadi xml. File xml ini mengalami perubahan secara manual sedemikian hingga client dapat membuat tampilan form berdasar dari file xml tersebut secara runtime.

File script fungsi berisi script yang harus dibangkitkan oleh client juga secara runtime sehingga pada saat suatu form digenerate user dapat melakukan event terhadap form tersebut.

Pembacaan dua jenis file ini menggunakan library bacaan dari python dan dibuat pada satu fungsi khusus yaitu fungsi baca_file dengan parameter berupa nama file yang diinginkan.

Berikut sintaks dari fungsi baca_file :

```
def bacafile (self,namafile):
        path = 'http://10.126.13.88/WK/'
        fil_e = path + namafile
        opener = urllib.FancyURLopener({})
        datafile = opener.open(fil_e)
        isifile = datafile.read()
        return isifile
```

Proses pembacaan file xml-form ini menggunakan library urllib.FancyURLopener dengan tujuan agar letak dari file sumber dapat diletakkan pada lokasi dynamis sehingga tidak selalu harus berada pada lokasi lokal server.

4.3.2 Proses pengepakan data menjadi xml

Pada bab 3 telah disebutkan bahwa pada dasarnya pengolahan data hasil query dari suatu database adalah berdasarkan sifat hasil query yang bersifat matrik teks biasa. Dengan analogi matriks m X n, baris 0 hasil query berisi field field dari hasil query dengan kolom mulai 0 sampai dengan n. Sedangkan baris 1 sampai dengan m berisi record record dengan tiap record terdiri dari kolom 0 sampai dengan n.

Setelah recordset hasil query didapatkan, pengolahan dilakukan dengan membaca baris 0 dari recordset tersebut. Dari tiap kolom lalu

ditambahkan elemen baru yang merupakan header dari document xml. Setelah elemen header didapatkan, maka pengolahan dilanjutkan dengan membaca baris 1 sampai dengan n dan mengolah tiap kolomnya menjadi elemen baru pada document xml.

Berikut pseudocode dari pengolahan recordset menjadi xml tersebut :

```
def datakexml
        data_xml = xml.dom.minidom.Document()
        elemennya = data_xml.createElement('datanya')
        data_xml.appendChild(elemennya) // document kosong dengan satu elemen root

        rekor = self.rekorset
        #bikin header data
        headernya = data_xml.createElement('header')
        data_xml.documentElement.appendChild(headernya)
        for j in range(len(rekor.description)):
        headerku = data_xml.createElement(str(rekor.description[j][0])) // baris 0 kolom j
                data_xml.documentElement.firstChild.appendChild(headerku)
        for i in range(len(xmlnya)):
                elemennya = data_xml.createElement('data')
                for j in range(len(rekor.description)):

        elemennya.setAttribute(str(rekor.description[j][0]),str(xmlnya[i][j]))   // baca isi
                data_xml.documentElement.appendChild(elemennya)
        return data_xml
```

Hasil dari parsing recordset tersebut adalah dokumen xml yang elemen elemennya adalah isi dari recordset tersebut.

4.3.3 Proses pengepakan fungsi script menjadi xml

Script fungsi berupa file teks yang berisi fungsi fungsi yang nantinya akan dibangkitkan secara runtime saat form digenerate. Untuk tugas akhir ini script fungsi yang menyertai form dibangung dengan memanfaatkan Windows Script Host dan bahasa yang dipergunakan adalah script visual basic.

database .

4.4. IMPLEMENTASI APLIKASI CLIENT

Aplikasi client melakukan interaksi secara langsung dengan user. Pertama kali aplikasi client dijalankan bisa saja aplikasi tidak memiliki interface sama sekali. Aplikasi lalu meminta kepada server bentuk form yang harus ditampilkan kepada user, data yang menyertainya dan fungsi yang melekat pada form tersebut. Hasil request kemudian digenerate oleh client sebagai form runtime yang dapat berinteraksi dengan user.

Saat user melakukan interaksi dengan form hasil generate, maka client melakukan request lagi terhadap event yang terjadi kepada server. Server yang kemudian menentukan apa yang harus dilakukan oleh client selanjtnya. Apakah membuat satu form baru lagi ataukah melakukan update tampilan dari form yang sudah ada.

Dari penjelasan di atas terdapat beberapa proses yang terjadi pada client yang dapat dijelaskan sebagai berikut :

4.4.1 Proses koneksi

Proses koneksi bisa dikatakan sebagai proses yang paling vital dalam Thin Client. Dalam proses ini, Thin Client mendefinisikan ip address dari, port komunikasi, end point, dan fungsi yang diinginkan dari aplikasi server. Untuk membuat delphi dapat berkomunikasi melalui XML-RPC, digunakan komponen dxmlrpc versi 2.0.

Berikut ini adalah contoh request pada Thin Client

```
FRpcCaller.HostName := '10.126.13.88';
FRpcCaller.HostPort := 80;
FRpcCaller.EndPoint := 'WK/server.py';
FRpcFunction.Clear;
FRpcFunction.ObjectMethod := 'op_login';
FRpcFunction.AddItem(data_login_xml);
RpcResult := FRpcCaller.Execute(FRpcFunction);
```

Pada request ini, Thin Client memngakses fungsi op_login, dan mengirimkan string data_login dalam bentuk xml sebagai parameternya. String result yang didapatkan dari aplikasi server ditempatkan pada variabel RpcResult.

4.4.2 Pengolahan string

String hasil request diterima dari server dalam bentuk dokumen xml. Pengolahan hasil request pertama kali dengan membaca elemen paling atas dari dokumen. Bila elemen paling atas memiliki value 'Fungsi' berarti client harus melakukan register script kepada control script yang ada. Bila elemen paling atas memiliki value 'Document' berarti client harus melakukan generate form baru. Bila elemen paling atas memiliki value 'enviro' bararti client harus melakukan update tampilan form aktif dan bila elemen paling atas memiliki value 'error' berarti ada kesalahan yang dilakukan oleh user.

4.4.2.1 Register fungsi script

Proses register fungsi script dilakukan dengan menambahkan script fungsi pada control windows script host. Sintaks untuk proses ini adalah sebagai berikut :

```
sc.AddCode(node_atas.GetAttribute('isi'));
```

Dimana sc pada sintaks di atas adalah nama kontrol yang melekat pada aplikasi client yang digunakan untuk melakukan manajemen dari script host.

4.4.2.2 Generate form runtime

Langkah langkah yang dilakukan untuk melakukan generate control secara runtime adalah sebagai berikut ini :

1. Mendeklarasikan satu instance dari komponen yang akan digenerate.
2. Gunakan fungsi create dari komponen master untuk kemudian ditampung sebagai instance.
3. Tentukan parent dari komponen yang dibuat.
4. Set atribut tambahan dari komponen sesuai dengan keinginan.
5. Set atribut visible menjadi True.
6. Ingat untuk selalu melakukan destroy komponen saat komponen tidak lagi diperlukan.
7. Tentukan fungsi yang menyertai komponen tersebut.

Proses generate form secara runtime dilakukan dengan melakukan parsing dari dokumen xml hasil request. Dokumen dibaca mulai dari elemen atas hingga elemen paling akhir. Setiap elemen baru akan digenerate menjadi satu control yang akan ditampilkan kepada user.

Setiap elemen juga akan dibaca apa saja atribut yang menyertainya. Setiap atribut menyatakan nilai properti yang melekat pada control yang dibuat.

Berikut ini adalah pseudo code untuk membuat satu form baru :

```
procedure Tform1.bikin_form(nodenya:IXMLNode);
begin
  form_baru := Tms_anak.Create(nil);
  ..... cutted
  setting untuk tiap atrribut dari nodenya
  .....cutted
  form_baru.visible = true;
  form_baru.show
end;
```

Setiap kontrol yang akan dibuat secara runtime harus memiliki nama yang unik. Sehingga pada saat generate tidak terjadi konflik nama dengan kontrol lainnya.

4.4.2.3 Update tampilan aktif form

Form yang aktif dapat saja mengalami update tampilan saat user melakukan aksi terhadap form tersebut. Hasil request dari server juga hanya akan berisi atribut atribut yang harus diset ulang dari tiap tiap control yang harus diupdate. Proses update dilakukan dengan memberikan nilai baru dari properti suatu control. Contoh :

```
.....
Edit1.text := 'nilai baru dari editbox satu'
.....
```

BAB 5
UJI COBA DAN EVALUASI

Pada bab ini akan dilakukan uji coba terhadap sistem yang telah dibuat. Uji coba dilakukan untuk mengetahui barapa lama waktu yang diperlukan bila suatu transaksi dilakukan. Waktu proses dihitung mulai suatu request dikirim kepada server hingga jawaban dari server selesai diolah.

5.1 Deskripsi Hardware dan Software Uji Coba

Uji coba terhadap sistem yang telah dibuat dilakukan dengan menggunakan lingkungan dengan spesifikasi sebagai berikut:

Spesifikasi platform server:

- Windows XP service pack 2
- Python 2.4
- Postgresql 8.1
- Apache 1.3.33
- Webware versi 0.8.1

Spesifikasi hardware server:

- Pentium III 600 Mhz
- RAM 512 MB

Spesifikasi platform client:

- Microsoft Windows XP Service Pack 2
- Windows Script Host ver. 5.6

Spesifikasi hardware client:

- Pentium II 400 Mhz
- RAM 512 MB

5.2 Skenario Uji Coba

Uji coba akan dilakukan dengan mengukur waktu yang diperlukan saat suatu transaksi dilakukan. Waktu proses dihitung mulai dari request dikirim kepada server, respond diterima dari server, pembacaan hasil request hingga pengolahan hasil request selesai dilakukan.

Form yang digunakan untuk uji coba adalah dua form master yaitu master koleksi dan master anggota dari Aplikasi Sistem Informasi Ruang Baca FTIF. Form master koleksi akan digenerate dengan bentuk form berasal dari file koleksi.xml sedangkan form master anggota akan digenerate dengan bentuk form berasal dari file anggota.xml. Jumlah data yang ditampilkan untuk masing masing form juga dirubah rubah.

Uji coba kemudian dilanjutkan untuk menunjukan bahwa perubahan tampilan form beserta fungsinya dapat dilakukan dengan merubah bentuk dokumen xml dari form yang berada pada sisi server. Pada uji coba ini form anggota yang sebelumnya tidak memiliki kontrol penanganan fungsi searching maka akan ditambahkan satu editbox dan satu button baru.

Berikut ini 2 skenario yang akan diujikan, yaitu:

1. Uji Coba Skenario Form Data Memberr
2. Uji Coba Skenario Form Data Koleksi
3. Uji Coba Penambahan Fungsi Searching pada form anggota

	73 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:02:02 AM
Waktu Total	0 jam 0 menit 2 detik 664 milidetik

Hasil dari ujicoba dengan data sebanyak 25:

Tabel 5.2 Hasil ujicoba data 25

Lama waktu proses	
Waktu request dikirim	03:09:11 AM
Waktu request diterima server	12:09:18 PM
Waktu request selesai ditangani server	12:09:21 PM
Waktu total pemrosesan di server	0 jam 0 menit 2 detik 615 milidetik
Waktu respond dari server diterima client	03:09:14 AM
Waktu client menunggu respond	0 jam 0 menit 2 detik 844 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:09:15 AM
Waktu Total	0 jam 0 menit 3 detik 475 milidetik

Hasil dari ujicoba dengan data sebanyak 50:

Tabel 5.3 Hasil ujicoba data 50

Lama waktu proses	
Waktu request dikirim	03:15:57 AM
Waktu request diterima server	12:16:04 PM
Waktu request selesai ditangani server	12:16:07 PM
Waktu total pemrosesan di server	0 jam 0 menit 3 detik 326 milidetik
Waktu respond dari server diterima client	03:16:00 AM
Waktu client menunggu respond	0 jam 0 menit 3 detik 405 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:16:01 AM
Waktu Total	0 jam 0 menit 4 detik 56 milidetik

Hasil dari ujicoba dengan data sebanyak 75:

Tabel 5.4 Hasil ujicoba data 75

Lama waktu proses	
Waktu request dikirim	03:18:59 AM
Waktu request diterima server	12:19:06 PM
Waktu request selesai ditangani server	12:19:10 PM
Waktu total pemrosesan di server	0 jam 0 menit 3 detik 958 milidetik
Waktu respond dari server diterima client	03:19:03 AM
Waktu client menunggu respond	0 jam 0 menit 4 detik 136 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:19:04 AM
Waktu Total	0 jam 0 menit 4 detik 817 milidetik

Hasil dari ujicoba dengan data sebanyak 100:

Tabel 5.5 Hasil ujicoba data 100

Lama waktu proses	
Waktu request dikirim	03:22:44 AM
Waktu request diterima server	12:22:52 PM
Waktu request selesai ditangani server	12:22:56 PM
Waktu total pemrosesan di server	0 jam 0 menit 4 detik 609 milidetik
Waktu respond dari server diterima client	03:22:49 AM
Waktu client menunggu respond	0 jam 0 menit 4 detik 807 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:22:50 AM
Waktu Total	0 jam 0 menit 5 detik 548 milidetik

Waktu antara aplikasi server dan aplikasi client pada waktu ujicoba tidak persis sama. Waktu proses relatif lebih lama adalah pada proses penanganan request oleh server. Hal ini berbanding lurus bila jumlah data yang akan ditampilkan ditingkatkan.

Evaluasi Terhadap Skenario View Master Koleksi:

Hasil dari ujicoba kedua dengan data 10:

Tabel 5.6 Hasil ujicoba data 10

Lama waktu proses	
Waktu request dikirim	03:26:51 AM
Waktu request diterima server	12:26:58 PM
Waktu request selesai ditangani server	12:27:03 PM
Waktu total pemrosesan di server	0 Jam 0 menit 4 detik 540 milidetik
Waktu respond dari server diterima client	03:26:55 AM
Waktu client menunggu respond	0 Jam 0 menit 4 detik 787 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:26:56 AM
Waktu Total	0 Jam 0 menit 5 detik 658 milidetik

Hasil dari ujicoba kedua dengan data 25:

Tabel 5.7 Hasil ujicoba data 25

Lama waktu proses	
Waktu request dikirim	03:29:44 AM
Waktu request diterima server	12:29:52 PM
Waktu request selesai ditangani server	12:29:58 PM
Waktu total pemrosesan di server	0 Jam 0 menit 6 detik 533 milidetik
Waktu respond dari server diterima client	03:29:51 AM
Waktu client menunggu respond	0 Jam 0 menit 6 detik 759 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:29:52 AM
Waktu Total	0 Jam 0 menit 7 detik 581 milidetik

Hasil dari ujicoba kedua dengan data 50:

Tabel 5.8 Hasil ujicoba data 50

Lama waktu proses	
Waktu request dikirim	03:33:33 AM
Waktu request diterima server	12:33:41 PM
Waktu request selesai ditangani server	12:33:49 PM
Waktu total pemrosesan di server	0 Jam 0 menit 7 detik 876 milidetik
Waktu respond dari server diterima client	03:33:41 AM
Waktu client menunggu respond	0 Jam 0 menit 8 detik 72 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:33:42 AM
Waktu Total	0 Jam 0 menit 8 detik 983 milidetik

Hasil dari ujicoba kedua dengan data 75:

Tabel 5.9 Hasil ujicoba data 75

Lama waktu proses	
Waktu request dikirim	03:36:33 AM
Waktu request diterima server	12:36:41 PM
Waktu request selesai ditangani server	12:36:51 PM
Waktu total pemrosesan di server	0 Jam 0 menit 10 detik 512 milidetik
Waktu respond dari server diterima client	03:36:43 AM
Waktu client menunggu respond	0 Jam 0 menit 10 detik 736 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:36:44 AM
Waktu Total	0 Jam 0 menit 11 detik 757 milidetik

Hasil dari ujicoba kedua dengan data 100:

Tabel 5.10 Hasil ujicoba data 100

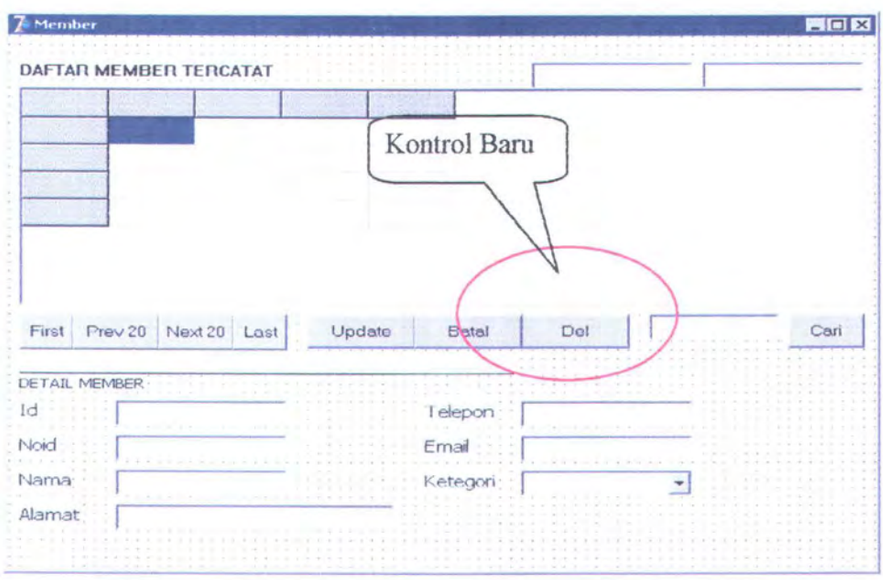
Lama waktu proses	
Waktu request dikirim	03:39:57 AM
Waktu request diterima server	12:40:05 PM

Waktu request selesai ditangani server	12:40:18 PM
Waktu total pemrosesan di server	0 Jam 0 menit 12 detik 626 milidetik
Waktu respond dari server diterima client	03:40:10 AM
Waktu client menunggu respond	0 Jam 0 menit 12 detik 939 milidetik
Waktu pemrosesan jawaban oleh client selesai	03:40:11 AM
Waktu Total	0 Jam 0 menit 14 detik 91 milidetik

Secara umum pada uji coba yang kedua ini memiliki waktu proses lebih lama dibandingkan dengan ujicoba pertama. Hal ini dikarenakan jumlah kontrol yang lebih banyak dan jumlah parsing field tiap record yang lebih banyak pula.

5.2.3 Uji Coba skenario Penambahan Fungsi Searching pada form anggota

Pertama kali tampilan dfm dari form anggota akan diubah menjadi seperti berikut ini :



Gambar 5.3 Form anggota baru

Dokumen xml baru dari dfm di atas didapatkan dari hasil aplikasi dfm menjadi xml dan diletakkan pada lokasi server sebagai dokumen xml baru.

Penambahan skrip fungsi dilakukan dengan sintaks baru:

```
sub mbr_btn_cari_OnClick(Sender)
        text_cari = frm_member.Controls("txt_cari")
        param = "<param><id_cari isi=""" &
text_cari.text & """></id_cari></param>"
        nilai = form1.terima_sc("member_cari",param)
        text_cari = ""
end sub
```

Sedangkan pada aplikasi server terdapat fungsi baru dengan nama "member_cari". Parameter yang diminta oleh fungsi ini adalah id anggota yang dicari. Potongan skrip fungsinya adalah sebagai berikut :

```
def member_cari(self,id_cari):
        param = parseString(id_cari)
        for id_cari_xml in param.getElementsByTagName('id_cari'):
                id_cari = id_cari_xml.getAttribute('isi')
        data_member = self.ambil_data(" SELECT * FROM v_anggota_kategori
where id = '"+id_cari+"'")

        if data_member!=[]: # ID ditemukan
                Update tampilan form client dengan data hasil pencarian
        else:
                Berikan error berupa data yang dicari tidak ada
```

Pada saat aplikasi client dijalankan, maka form anggota telah berubah menjadi tampilan yang baru dengan fungsi searching yang dapat berjalan.

BAB 6
KESIMPULAN DAN SARAN

Bab ini menyimpulkan hasil uji coba yang telah dilakukan. Selanjutnya diberikan beberapa saran yang mungkin dapat dijadikan pertimbangan untuk mengembangkan hasil yang diperoleh pada tugas akhir ini.

6.1. Kesimpulan

Setelah dilakukan serangkaian uji coba dan analisa terhadap sistem yang dibuat, maka dapat diambil kesimpulan sebagai berikut:

1. Aplikasi sistem sebagai prototipe sistem adaptif desktop berbasis xml-rpc telah berhasil diimplementasikan. Struktur dari interface client dan fungsi yang menyertainya terletak di server dan dapat digenerate secara runtime oleh client dengan baik.
2. Semakin besar jumlah data yang ditangani, maka waktu pemrosesan berjalan semakin lama. Besar data bergantung dari jumlah record yang ditangani, jumlah field untuk tiap record dan jumlah control form yang harus dibangkitkan.
3. Aplikasi dapat mengimplementasikan pengembangan sistem secara terpusat. Bila diinginkan perubahan pada sisi client maka yang dilakukan cukup merubah data yang berada di server.

6.2. Saran

Berdasarkan hasil evaluasi yang dilakukan terhadap sistem, ada beberapa saran yang perlu dipertimbangkan dalam pengembangan teknologi ini, yaitu :

1. Pengembangan dapat dilakukan dengan menggunakan control script selain windows script host seperti yang digunakan pada tugas akhir ini.
2. Penambahan fungsi keamanan dapat dilakukan sehingga data yang dikirimkan walau tetap dalam bentuk xml namun telah terenkripsi sehingga transaksi data lebih aman.

DAFTAR PUSTAKA

[1] Prevandhito, Perancangan dan Pembuatan 3-Tier Sistem Menggunakan Web Services dan Thin Client Berbasis Protokol XML-RPC, FTIF/ITS, 2004

[2] xmlrpc homepage
http://www.xmlrpc.com

[3] Python homepage
http://www.python.org

[4] webservices homepage
http://www.webservices.org

[5] Junaedy, Pengantar XML, www.ilmukomputer.com

[6] Extensible Markup Language (XML) http://www.w3.org/XML/

[7] Communication Programming Concepts
http://publib16.boulder.ibm.com/pseries/en_US/aixprggd/progcomc

[8] Billy G. Allie, pyPgSQL, version 2.4 A Python DB-API 2.0 compliant interface for PostgreSQL.

[9] XML-RPC How To
http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/XML-RPC-HOWTO.pdf

[10] Python/XML HOWTO
http://www.sourceforge.net/topics/howto/xml-howto.html

[11] WebKit Install Guide from Webware Manual Document

[12] Creating Object On Fly using Delphi., The Unofficial Newsletter of Delphi Users - Issue #2 - April 1st, 1995

[13] Christoper A. Jones, Fred L. Drake,Jr. Python & XML, Oreilly,2002

[14] Basic XML Parsing with Delphi

http://delphi.about.com/gi/dynamic/offsite.htm?site=http://homepages.borland.com/ccalvert/TechPapers/Delphi/XMLSimple/XMLSimple.html

[15] Exporting a TreeView to XML. Populating a TreeView from XML

http://www.delphi.about.com/library/weekly/aa101904a.htm