OPTIMASI WAKTU KOMPUTASI PENGHITUNGAN REKENING PELANGGAN PT. PLN DISTRIBUSI JAWA TIMUR DENGAN MENGGUNAKAN TEKNOLOGI *WEBSERVICE*

TUGAS AKHIR

Disusun Oleh :

FARID REDRIKANA
NRP.5100 100 071

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

OPTIMASI WAKTU KOMPUTASI PENGHITUNGAN REKENING PELANGGAN PT. PLN DISTRIBUSI JAWA TIMUR DENGAN MENGGUNAKAN TEKNOLOGI *WEBSERVICE*

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 I	Dosen Pembimbing II
(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

NOPEMBER, 2005

ABSTRAK

Salah satu sumber data untuk menganalisa kondisi perusahaan, PT. PLN, adalah proyeksi pendapatan perusahaan yang tercermin dari hasil penjualan energi listrik kepada para pelanggannya. Untuk mendapatkan proyeksi pendapatan yang dimaksud di atas, maka harus dilakukan proses penghitungan rekening pelanggan untuk mendapatkan tagihan pelanggan sebagai sumber pendapatan perusahaan. Namun proses penghitungan ini akan memakan waktu yang sangat lama karena terdapat lebih dari satu juta data pelanggan yang harus diproses. Oleh karena itu dibutuhkan sebuah sistem yang mampu menyelesaikan proses penghitungan rekening pelanggan tersebut dengan waktu yang lebih cepat.

Hal ini mendorong penulis untuk memanfaatkan teknologi webservice untuk dapat mendistribusikan data pelanggan ke mesin – mesin penghitung rekening, sehingga mesin – mesin penghitung rekening tersebut dapat melakukan proses penghitungan rekening secara paralel. Dengan demikian diharapkan penghitungan rekening untuk seluruh data pelanggan dapat lebih cepat diselesaikan sehingga waktu komputasi totalnya dapat dioptimalkan

Dari hasil pengujian dan evaluasi, dengan skenario penambahan data dan penambahan mesin penghitungan rekening, sistem penghitungan rekening pelanggan yang terdistribusi dengan menggunakan teknologi Webservice ini mampu menunjukan adanya percepatan dalam menyelesaikan proses penghitungan rekening.

KATA PENGANTAR

Segala puji dan syukur Alhamdulillah semata ditujukan ke hadirat ALLAH SWT yang telah memberikan rahmat dan hidayah-Nya sehingga memungkinkan penulis untuk menyelesaikan Tugas Akhir yang berjudul "***OPTIMASI WAKTU KOMPUTASI PENGHITUNGAN REKENING PELANGGAN PT. PLN DISTRIBUSI JAWA TIMUR DENGAN MENGGUNAKAN TEKNOLOGI WEBSERVICE***".

Mata Kuliah Tugas Akhir yang memiliki beban sebesar empat satuan kredit disusun dan diajukan sebagai salah satu syarat untuk menyelesaikan program strata satu (S-1) pada jurusan Teknik Informatika di Institut Teknologi Sepuluh Nopember Surabaya.

Dalam penyusunan Tugas Akhir ini, Penulis berusaha untuk menerapkan ilmu yang telah didapat selama menjalani perkuliahan dengan tidak terlepas dari petunjuk, bimbingan, bantuan, dan dukungan berbagai pihak.

Dengan tidak lupa akan kodratnya sebagai manusia, Penulis menyadari bahwa dalam karya Tugas Akhir ini masih mengandung kekurangan di sana-sini sehingga dengan segala kerendahan hati Penulis masih dan insya Allah akan tetap terus masih mengharapkan saran serta kritik yang membangun dari rekan-rekan pembaca.

Surabaya, Nopember 2005

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

Dalam bab ini dijelaskan beberapa hal dasar yang meliputi latar belakang, permasalahan, batasan permasalahan, tujuan dan manfaat, metodologi pelaksanaan serta sistematika penulisan buku Tugas Akhir ini. Dari uraian tersebut diharapkan, gambaran umum permasalahan dan pemecahan yang diambil dapat dipahami dengan baik.

1.1 Latar Belakang

Saat ini PT PLN Jawa Timur telah memiliki Sistem Pusat Data (SPD) yang bertugas mengumpulkan dan dan memelihara data operasional perusahaan. Akan sangat sayang apabila data – data tersebut dibiarkan disimpan begitu saja. Data ini seharusnya dapat dijadikan sumber data untuk analisa kondisi kekinian perusahaan. Salah satu analisa yang bisa dilakukan adalah analisa proyeksi pendapatan PT. PLN Jawa Timur pada bulan tahun tertentu berdasarkan hasil penjualan listrik pada bulan tahun tersebut yang bisa dilihat melalui pemakaian listrik pelanggan.

Untuk mendapatkan proyeksi pendapatan yang dimaksud di atas, maka harus dilakukan proses penghitungan rekening pelanggan untuk mendapatkan tagihan pelanggan sebagai sumber pendapatan perusahaan. Namun proses penghitungan ini akan memakan waktu yang sangat lama karena terdapat lebih dari satu juta data pelanggan yang harus diproses. Oleh karena itu dibutuhkan sebuah sistem

yang mampu menyelesaikan proses penghitungan rekening pelanggan tersebut dengan waktu yang lebih cepat.

Hal ini mendorong penulis untuk memanfaatkan teknologi *webservice* untuk dapat mendistribusikan data pelanggan ke mesin – mesin penghitung rekening, sehingga mesin – mesin penghitung rekening dapat melakukan proses penghitungan rekening secara paralel. Dengan demikian diharapkan penghitungan rekening untuk seluruh data pelanggan dapat lebih cepat diselesaikan sehingga waktu komputasi totalnya dapat dioptimalkan.

1.2 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk merancang dan mengimplementasikan sistem terdistribusi yang mampu mengoptimalkan waktu komputasi penghitungan rekening pelanggan dengan menggunakan teknologi *webservice*. Optimasi dilakukan dengan mengusahakan adanya percepatan sistem dalam menyelesaikan proses penghitungan rekening dengan cara melakukan penghitungan rekening pelanggan secara parallel pada beberapa mesin penghitung rekening, dimana data yang diproses didistribusikan oleh sebuah sistem pendistribusian data dengan menggunakan teknologi *webservice*.

Dengan optimalnya waktu penghitungan tersebut maka proyeksi pendapatan perusahaan dapat diperoleh dalam waktu yang lebih cepat yang kemudian dapat dimanfaatkan sebagai bahan pelengkap dalam menganalisa kondisi perusahaan.

1.3 Permasalahan

Permasalahan yang dihadapi adalah dalam pembuatan Tugas Akhir ini adalah:

a. Bagaimana merancang dan mengimplementasikan sistem yang mampu mendistribusikan data ke mesin – mesin penghitungan rekening dengan memanfaatkan teknologi *webservice.*

b. Penanganan proses penyimpanan data hasil penghitungan rekening dari mesin penghitung rekening ke SPD.

c. Pembuatan sebuah format data hasil *query* ke SPD pada sistem pendistribusian data, dimana dengan format data tersebut mesin penghitung rekening dapat dengan mudah mengubahnya kedalam bentuk skema data yang menyerupai *recordset* untuk kemudahan dalam proses penghitungan rekening.

d. Pengukuran unjuk kerja sistem dalam mengoptimalkan proses penghitungan rekening dengan mendapatkan percepatan sistem dalam menyelesaikan proses penghitungan rekening.

1.4 Batasan Masalah

Sejumlah batasan masalah dalam pembuatan Tugas Akhir ini adalah:

a. Uji coba dilakukan dengan menggunkan sebuah *server* database dengan menggunakan oracle10g DBMS, sebuah *Webserver* sederhana yang mampu menjalankan *webservice*, dan beberapa komputer yang melakukan proses penghitungan rekening, dimana uji coba ini dilakukan dalam lingkungan LAN jurusan Teknik Informatika ITS tanpa melakukan proses enkripsi data.

b. Sistem tidak mengimplementasikan *Universal Description Discovery Integration* (UDDI). Dan untuk deskripsi servis berupa *Webservice Defintion Language* atau WSDL akan disediakan pada *Webserver* dan dapat diakses dengan metode GET melalui URI yang telah ditentukan.

c. Dalam mengimplementasikan sitem terdistribusi berbasis *webservice* ini, aplikasi – aplikasi yang dibangun akan menggunakan *library – library* atau modul – modul Python2.3 dan Visual Basic 6.0 baik yang telah disediakan maupun yang ditambahkan, misalnya untuk *XML-messaging* via HTTP yang biasa disebut *Simple Object Access Protocol* (SOAP) sebagai salah satu standard *webservice* akan digunakan *library* MSSOAP.1 di sisi sistem penghitungan rekening dan modol SOAPpy di sisi sistem pendistribusian data.

d. Optimasi dilakukan dengan mengusahakan adanya percepatan sistem dalam meyelesaikan penghitungan rekening dengan menambahkan penggunaan beberapa mesin penghitung rekening secara bersama – sama.

e. Percepatan sistem adalah:

$$\frac{\textbf{Waktu komputasi sistem terkecil}}{\textbf{Waktu komputasi sistem setelah diekspansi}}$$

f. Sistem yang paling kecil ini adalah sebuah *server* pendistribusi data, sebuah sistem penghitungan rekening, dan SPD. Ekspansi sistem dilakukan dengan menambahkan mesin penghitung rekening.

1.5 Metodologi

Pembuatan Tugas Akhir ini terdiri dari beberapa tahapan sebagai berikut:

a. Studi Literatur

Pada tahap ini dilakukan pengumpulan informasi – informasi yang diperlukan dalam proses perancangan dan implementasi sistem yang akan dibangun. Adapun informasi – informasi yang diperlukan diantaranya adalah

spesifikasi *webservice*, penerapan *webservice* dengan bahasa pemrograman Python 2.3 sebagai *server* dan bahasa pemrograman Visual Basic 6.0 sebagai sistem penghitungan rekening.

b. Perancangan dan Implementasi Sistem

Pada tahap ini dilakukan perancangan dan implementasi sistem yang meliputi:

- Perancangan Arsitektur Sistem

  Perancangan arsitektur sistem adalah tahapan untuk menentukan model arsitektur sistem untuk mengimplementasikan penghitungan rekening terdistribusi dengan tujuan untuk mengoptimalkan proses penghitungan rekening itu sendiri.

- Perancangan Data

  Yaitu perancangan format baku dalam pertukaran data, serta perancangan data data di sisi sistem penghitungan rekening untuk menyimpan sementara data pelanggan yang diterima dari *server* (hingga mendekati format *recordset*) agar mudah untuk diambil kembali guna dilakukan proses penghitungan rekening.

- Perancangan Proses

  Yaitu perancangan proses pendistribusian data yang pada kenyataannya adalah proses pengambilan data oleh *webservice* dari SPD, dan pertukaran data antara komputer penghitung rekening dengan *webservice*. Selain itu terdapat pula proses penghitungan rekening sendiri

yang berjalan pada komputer – komputer penghitung rekening dengan terlebih dahulu meminta data pelanggan dan data tarif dasar listrik ke SPD melalui *webservice*.

- Perancangan Antarmuka

  Yaitu perancangan antarmuka untuk memudahkan pengoperasian perangkat – perangkat lunak di dalam sistem yang akan dibangun.

- Implementasi

  Yaitu pembuatan perangkat lunak sesuai dengan perancangan yang telah dilakukan dengan menggunakan bahasa pemrograman Python 2.3 di sisi *server* dan Visual Basic 6.0 di sisi sistem penghitungan rekening.

c. Uji Coba dan Analisa

Pada tahap ini dilakukan uji coba dengan skenario tertentu untuk mendapatkan percepatan sistem dalam menyelesaikan proses penghitungan rekening sebagai pengukur unjuk kerja sistem dalam mengoptimalkan proses penghitungan rekening.

d. Penyusunan Laporan Tugas Akhir

Penyusunan laporan Tugas Akhir ini berisi informasi – informasi mengenai sistem yang telah dibangun dan sekaligus merupakan tahapan akhir dari pelaksanaan Tugas Akhir.

1.6 Sistematika Penulisan

Buku Tugas Akhir ini terdiri dari beberapa bab yang tersusun secara sistematis, yaitu :

BAB I PENDAHULUAN

Bab ini merupakan gambaran umum yang membahas latar belakang dan tujuan pembuatan perangkat lunak, serta permasalahan yang dihadapi dalam pengerjaan Tugas Akhir ini. Selain itu dijelaskan pula pembatasan terhadap masalah yang akan dihadapi serta metodologi yang dipakai untuk menyelesaikannya.

BAB II TEORI PENUNJANG

Bab ini membahas teori-teori dasar yang menunjang pengerjaan Tugas Akhir, meliputi arsitektur *webservice*, *Webservice Definition Languange* (WSDL), *Simple Object Access Protocol* (SOAP), penerapan *webservice* dengan bahasa pemrograman Python 2.3 sebagai *server* dan Visual Basic 6.0 sebagai sistem penghitungan rekening, dan sistem terdistribusi.

BAB III PERANCANGAN SISTEM

Dalam bab ini diuraikan mengenai sistem yang akan dibangun yakni mengenai perancangan arsitektur sistem, perancangan data beserta proses – proses yang ada, serta perancangan antarmuka untuk menjalankan perangkat – perangkat lunak yang berada dalam sistem.

BAB IV IMPLEMENTASI

Dalam bab ini diuraikan implementasi dari perancangan yang telah dilakukan sebelumnya yakni implementasi arsitektur sistem dengan pengadaan komputer – komputer sistem penghitungan rekening dan *server* beserta perangkat – perangkat lunaknya.

BAB V UJI COBA DAN ANALISA

Berisi tentang uji coba yang telah dilakukan terhadap sistem yang telah dibuat beserta analisa dari hasil uji coba tersebut.

BAB VI PENUTUP

Berisi kesimpulan yang di dapat dari pembuatan Tugas Akhir ini dan dilengkapi dengan saran untuk kemungkinan pengembangan selanjutnya.

BAB II

TEORI PENUNJANG

Dalam bab ini dijelasakan mengenai teori – teori penunjang, khususnya mengenai *webservice* dan implementasinya dengan menggunakan Python dan Visual Basic 6.0, yang digunakan dalam menyelesaikan Tugas Akhir ini.

2.1 Teknologi *Webservice*

Berikut ini adalah penjelasan beberapa konsep yang berkaitan dengan teknologi *webservice* yang meliputi definisi, arsitektur serta konsep – konsep lain yang digunakan sebagai teori penunjang bagi penulis dalam pengerjaan Tugas Akhir ini.

2.1.1 Definisi *Webservice*

Webservice sendiri merupakan teknologi yang masih dalam tahap pengembangan, sehingga ada beberapa bagian baik dalam konsep maupun penerapannya yang berbeda antara satu dengan lainnya diantara para pengembangnya, termasuk di dalamnya definisi dari *webservice* itu sendiri.

Berikut ini beberapa definisi *webservice* dari berbagai sumber yang didapatkan oleh penulis:

a. *Webservice* adalah sebuah sistem perangkat lunak yang didesain untuk mendukung interoperabilitas antar mesin yang berkomunikasi melalui jaringan komputer yang ada. *Webservice* ini memiliki *interface* publik yang

b. dideskripsikan melalui sebuah dukumen dengan format yang dapat diproses oleh mesin (secara spesifik disebut *Webservice Defintion Language* atau WSDL). Deskripsi ini dibaca oleh sistem lain yang ingin berinteraksi dengan *webservice* dengan menggunakan *Simple Access Object Protocol* atau (SOAP) yang ditransportasikan melalui protokol HTTP [W3C04].

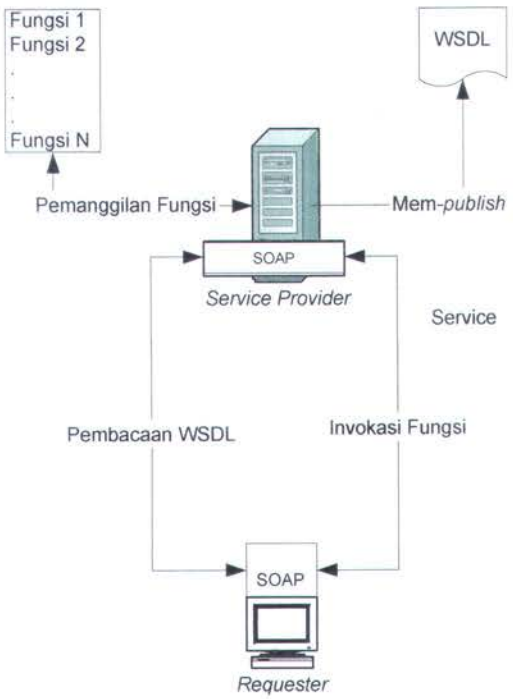
c. *Webservice* adalah berbagai servis yang tersedia di internet, menggunakan pertukaran data standard berbasis XML, dan tidak terikat pada satu sistem operasi atau bahasa pemrograman [ETH02].

d. *Webservice* adalah seperangkat standard yang memungkinkan interoperabilitas diantara sistem yang heterogen. Werbservice merupakan sebuah aplikasi berbasis web yang menyediakan fungsi dan interoperasi dari sebuah operasi bisnis sederhana ke operasi bisnis yang lebih kompleks [IBM05].

e. *Webservice* adalah aplikasi yang menyediakan sebuah API (*Aplication Programming Interface*) berbasis web, dimana API tersebut memungkinkan aplikasi bertindak sebagai servis dan dapat diakses melalui sebuah URI [ROS04].

2.1.2 Arsitektur *Webservice*

Dari pengertian di atas maka dapat disimpulkan bahwa sebuah *webservice* minimal memiliki satu atau lebih servis yang dipublikasi oleh penyedia servis (*service provider*) dimana fungsi – fungsi didalamnya haruslah memiliki *interface* publik yang dideskripsikan dengan sebuah dokumen berbasis XML yang biasa disebut WSDL, dan satu atau lebih pengguna service (*requester*).



Gambar II-1 Arsitektur *Webservice*

Untuk mempermudah pencarian dan penggunaan servis yang disediakan, *service provider* juga dapat meyediakan sebuah direktori pencarian publik yang mengintegrasikan *interface* dan invokasi fungsi secara langsung melalui sebuah (Universal Description, Discovery, and Integration / UDDI) .

Walaupun teknologi *webservice* masih belum memiliki arsitektur yang pasti dan disepakati oleh para pengembangnya akan tetapi spesifikasi minimal seperti yang telah dijelaskan diatas harus dapat dipenuhi. Sementara ini secara garis besar terdapat empat lapisan utama untuk mengimplementasikan sebuah *webservice* yakni lapisan *discovery*, *description*, *messaging*, dan *transport* yang biasa disebut dengan susunan protokol *webservice* (*webservice protocol stack*).

Discovery	**UDDI**
Description	**WSDL**
XML-Messaging	**XML-RPC, SOAP,XML**
Transport	**HTTP, FTP, SMTP**

Gambar II-2 *Webservice Protocol Stack*

Lapisan *Dicovery*

Lapisan ini bertugas sebagai pusat publikasi dan informasi servis – servis yang disediakan oleh sebuah penyedia servis, ditempat ini pula dilakukan integrasi *interface* fungsi – fungsi yang disediakan sehingga *requester* dapat mencari service yang disediakan dan langsung dapat menggunakan servis tersebut tanpa harus melakukan pembacaan WSDL. Implementasi dari lapisan ini adalah UDDI yang biasanya digunakan untuk mencari dan mencoba sebuah service yang disediakan.

Lapisan *Description*

Lapisan ini bertugas mendekripsikan setiap service yang ada. Deskripsi tersebut meliputi nama servis dan bagaimana mengakses servis tersebut. Implementasi dari lapisan ini adalah sebuah dokumen WSDL yang harus di-publish sebagai informasi *interface* publik.

Lapisan *XML-Messaging*

Lapisan ini bertugas untuk menyediakan format pengiriman data berbasis XML yang digunakan sebagai *input* dan *output* komunikasi antar mesin yang terlibat. Implementasi dari lapisan ini adalah format data bebasis XML yang akan ditransportasikan via-HTTP yang biasa disebut SOAP.

Lapisan *Transport*

Lapisan ini bertugas untuk mengirimkan data melalui protokol transportasi tertentu seperti HTTP, SMTP, FTP, dan lain - lain. Pada kenyataannya protokol HTTP adalah prtotokol yang paling sering digunakan.

2.1.3 *Webservice* Definition Language (WSDL)

WSDL merupakan sebuah spesifikasi pendefinisian sebuah *webservice* yang berupa dokumen berbasis XML [WKW05]. Secara garis besar WSDL mendeskripsikan empat hal yaitu:

- Informasi *interface* publik untuk fungsi – fungsi yang disediakan.
- Informasi tipe data yang digunakan dalam message – message yang akan dipertukarkan.
- Informasi binding yang menunjukkan bagaimana message – message ditransportasikan.
- Lokasi dimana service disediakan yang beruapa sebuah URI.

Misalkan sebuah aplikasi ingin menggunakan servis **X** maka aplikasi tersebut harus membaca terlebih dahulu WSDL dari servis **X**, baru kemudian memanggil fungsi – fungsi yang disediakan didalam servis **X** tersebut dengan mengikuti standar pemanggilan yang diperoleh dari WSDL dan menunggu respon dari penyedia servis.

Berikut ini adalah spesifikasi dari sebuah dokumen WSDL:

<definition>

Merupakan root-document atau elemen paling atas dari sebuah dokumen WSDL, berisi definisi – definisi yang digunakan dalam dokumen WSDL tersebut.

<types>

Elemen ini berisi mengenai tipe – tipe data yang akan digunakan dalam pertukaran data.

<message>

Elemen ini mendefinisikan message – message yang akan digunakan, baik untuk *request* ataupun response. Pada elemen juga bisa terdapat elemen **<part>** yang mendefinisikan parameter – parameter atau nilai pengembalian dari fungsi – fungsi yang menggunakan message yang dimaksud.

<portType>

Elemen ini adalah merupakan definisi dari operasi – operasi atau fungsi – fungsi yang ada lengkap dengan message – message yang digunakan.

<binding>

Elemen ini menjelaskan secara spesifik bagaimana fungsi – fungsi yang ada yang diwakili oleh message ditransportasikan.

<service>

Elemen ini berisi nama servis dan lokasi dimana servis tersebut tersedia.

Berikut ini adalah contoh sebuah dokumen WSDL yang mendeskripsikan servis **Temperature**, dimana terdapat fungsi **getTemp(zipcode)** yang

mengembalikan suhu pada lokasi dengan kode pos yang disebutkan dalam paramater fungsi.

```
<?xml version="1.0" ?>
<definitions name="TemperatureService"
 targetNamespace=
 "http://www.xmethods.net/sd/TemperatureService.wsdl"
 xmlns:tns="http://www.xmethods.net/sd/TemperatureService.wsdl"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">

<message name="getTempRequest">
  <part name="zipcode" type="xsd:string" />
</message>
<message name="getTempResponse">
  <part name="return" type="xsd:float" />
</message>

<portType name="TemperaturePortType">
<operation name="getTemp">
  <input message="tns:getTempRequest" />
  <output message="tns:getTempResponse" />
</operation>
</portType>

<binding name="TemperatureBinding"
 type="tns:TemperaturePortType">
<soap:binding style="rpc"
 transport="http://schemas.xmlsoap.org/soap/http" />
<operation name="getTemp">
<soap:operation soapAction="" />
  <input>
     <soap:body use="encoded"
      namespace="urn:xmethods-Temperature"
      encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/" />
  </input>
  <output>
    <soap:body
     use="encoded" namespace="urn:xmethods-Temperature"
     encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/" />
  </output>
</operation>
</binding>

<service name="TemperatureService">
<documentation>Returns current temperature in a given U.S.
 zipcode</documentation>
<port
 name="TemperaturePort" binding="tns:TemperatureBinding">
<soap:address
```

```
 location=
"http://services.xmethods.net:80/soap/servlet/rpcrouter" />
</port>
</service>
 </definitions>
```

2.1.4 *Simple Access Object Protokol* (SOAP)

SOAP merupakan sebuah protokol digunakan untuk pertukaran pesan berbasis XML antar aplikasi dengan memanfaatkan protokol transportasi yang telah ada [WKS05]. Walaupun SOAP dapat direalisasikan dengan beberapa protokol transportasi akan tetapi pada awalnya dan pada umumnya SOAP direalisasikan dengan menggunakan protokol HTTP.

Dengan menggunakan SOAP, maka sebuah aplikasi sistem penghitungan rekening dapat dengan mudah melakukan koneksi ke *webservice* dan melakukan pemaggilan fungsi – fungsi yang disediakan. Karena menggunakan format XML dalam penulisannya dan memanfaatkan HTTP dalam transportasi data, maka SOAP memiliki interoperabilitas yang tinggi walaupun aplikasi yang berinteraksi memiliki perbedaan platform sistem operasi, bahasa pemrograman bahkan jaringan.

Sebagai sebuah protokol pertukaran data berbasis XML, SOAP memiliki beberapa aturan – aturan dalam pengimplentasiannya yang berupa format penulisan pesan berbasis XML yang dapat dijelaskan sebagai berikut:

Ada beberapa elemen yang membentuk SOAP, yaitu:

a. *Envelope*

Envelope merupakan elemen paling atas (*root element*) dari SOAP dimana didalamnya terdapat dua atribute utama yang harus disertakan, yaitu:

- *Namespaces* yang berisi URI http://www.w3.org/2001/12/soap-envelope
- *encodingStyle* yang berisi URI yang digunakan sebagai acuan dalam mendefinisikan tipe data yang digunakan dalam SOAP. Biasanya diisi dengan URI http://www.w3.org/2001/12/soap-encoding.

Contoh:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soape-ncoding">
  ...
  isi pesan
  ...
</soap:Envelope>
```

b. *Header*

Header adalah elemen anak dari *Envelope* yang berisi tentang informasi mengenai aplikasi yang diwakili oleh SOAP itu sendiri. Elemen ini bisa tidak disertakan, dan apabila disertakan maka harus menjadi elemen anak pertama dari *Envelope*.

Contoh:

```
<soap:Header>
<m:Billing
 mlns:m="http://localhost:8080/web-svcs/">PLN</m:Billing>
</soap:Header>
```

c. *Body*

Body adalah elemen yang berisi data aktual yang dibawa oleh SOAP dan menjadi elemen anak dari Envelope.

Contoh:

```
<soap:Body>
<m:Data xmlns:m=" http://localhost:8080/web-svcs/">
<m:IDPEL>5111</m:IDPEL>
   </m: Data >
</soap:Body>
```

Dari elemen – elemen tersebut terbentuklah dokumen SOAP yang siap dijadikan standard dalam pengiriman data antar aplikasi yang berjalan dalam lingkungan *webservice*. Contoh lengkap sebuah dokumen SOAP adalah sebagai berikut:

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
<soap:Header>
<m:Billing xmlns:m="http://localhost:8080/web-svcs/">
  PLN
</m:Billing>
</soap:Header>
<soap:Body>
<m:Data xmlns:m=" http://localhost:8080/web-svcs/">
<m:IDPEL>5111</m:IDPEL>
</m: Data >
</soap:Body>
</soap:Envelope>
```

2.2 Python 2.3 dan *Webservice*

Python 2.3 merupakan sebuah bahasa pemrograman tingkat tinggi yang masuk ke dalam golongan interpreter dan dapat digunakan dan didistribusikan secara cuma – cuma. Python 2.3 memiliki struktur data tingkat tinggi dan telah mendukung pendekatan pemrograman berorientasi obyek dan dapat dijalankan hampir di semua sistem operasi yang ada.

Sebagaimana halnya beberapa bahasa pemrograman tingkat tinggi yang terkemuka, seperti Java dan .NET, Python 2.3 juga mendukung pengimplentasian *webservice*. Terdapat dua modul yang kini dapat digunakan dalam membangun sebuah *webservice* berbasis Python, yakni SOAPpy yang dibuat oleh Cayce Ullman and Brian Matthews, dan sekarang dikembangkan oleh Gregory Warnes dan Ivan R. Judson, serta ZSI (Zolera SOAP Infrastructure) yang diprakarsai oleh Rich Salz, dan sekarang dikembangkan oleh by Rich dan Christopher serta ZSI. Kedua modul ini dapat mengimplentasikan *webservice* baik di sisi sistem penghitungan rekening maupun *server*.

2.3 Visual Basic 6.0 dan *Webservice*

Visual Basic 6.0 adalah bahasa pemrograman tingkat tinggi yang termasuk di dalam golongan interpreter dan merupakan salah satu produk dari Microsoft Corp. Visual Basic 6.0 hanya berjalan di atas lingkungan sistem operasi Windows. Bahasa pemrograman inipun juga telah mendukung pengimplementasian *webservice*, baik di sisi sistem penghitungan rekening dan di sisi *server*.

Adapun library standard yang terdapat dalam Visual Basic 6.0 ini library MSSOAP.1 yang mampu membaca dokumen WSDL dan mencari informasi – informasi yang terkandung di dalamnya serta melakukan pertukaran data antara sistem penghitungan rekening dengan *webservice*.

2.4 Sistem Terdistribusi

Sistem terdistribusi merupakan sekumpulan sistem (dapat berupa komputer) yang berdiri sendiri – sendiri dimana dalam pemanfaatnnya seolah – olah hanya berupa sebuah sistem (komputer) [TAN02].

Tujuan utama dari sistem terdistibusi ini adalah untuk memungkinkan pengguna – pengguna komputer untuk saling berkomunikasi dan berbagi *resource* tanpa harus tahu dengan pasti dimana sebenarnya *resource* tersebut tersedia, cukup dengan menggunkan *interface* serta aturan – aturan pengaksesannya untuk memanfaatkan *resource* yang disediakan. Dengan adanya aktifitas saling berbagi dan saling berkomunikasi, maka akan terbentuk sebuah sistem yang makin lama akan semakin membesar seiring dengan bertambahnya pihak yang berpartisipasi di dalamnya.

Contoh dari sistem terdistribusi yang paling mudah dipahami adalah sistem internet, dimana ada banyak yang terlibat baik dari sisi *internet service provider*, penyedia sistus, hingga para pemakai jasa internet. Semua Bagian – bagian tersebut membentuk sebuah kesatuan sistem dengan saling berbagi *resource* dan membentuk sebuah sistem terdistribusi.

BAB III

PERANCANGAN SISTEM

Tahap ini meliputi perancangan arsitektur sistem, perancangan data, perancangan proses dan perancangan antarmuka aplikasi – aplikasi yang terdapat di dalam sistem.

3.1 Perancangan Arsitektur Sistem

Secara umum sistem yang akan dibangun harus memiliki kemampuan sebagai berikut:

a. Sistem pendistribusian Data

- Mengambil data yang akan didistribusikan dari SPD. SPD menggunakan Oracle10g sebagai sistem manajemen databasenya.
- Mampu mendistribusikan data sistem penghitungan rekening dengan memberikan data tarif dasar listrik atau sejumlah data pelanggan ketika komputer – komputer dalam sistem penghitungan rekening melakukan permintaan data.
- Mampu menerima kembali data yang telah diproses oleh sistem penghitungan rekening dan menyimpannya di dalam SPD.
- Pendistribusian dan penerimaan kembali data dilakukan dengan meyediakan servis yang dipublikasikan melalui *webservice* (WSDL, SOAP, HTTP) yang kemudian dapat digunakan sistem penghitungan

rekening untuk melakukan pemanggilan fungsi lewat servis tersebut untuk mendapatkan data yang dibutuhkan untuk proses penghitungan rekening.

b. Sistem Penghitungan Rekening

- Mampu melakukan pemaggilan fungsi secara remote fungsi – fungsi yang telah disediakan oleh *webservice* melalui servis yang telah dipublikasikan untuk mendapatkan data yang akan digunakan untuk proses penghitungan rekening.
- Mampu melakukan penghitungan rekening pelanggan dengan data yang telah diperoleh dari SPD melalui fungsi permintaan data dari *webservice*.
- Mampu mengembalikan hasil dari proses penghitungan ke *webservice* dengan memanfaatkan fungsi yang juga telah disediakan dalam servis yang telah dipublikasikan oleh *webservice*.
- Komunikasi dengan *server* dilakukan dengan melakukan pembacaan WSDL dan kemudian melalakukan pengambilan fungsi dengan protokol SOAP pada protokol transportasi HTTP.

Dengan demikian akan ada sebuah sistem pendistribusian data yang mengambil data dari SPD, dan sebuah sistem penghitungan rekening yang mengambil data dari sistem pendistribusian data, dimana komunikasi antar sistem ini menggunakan protokol SOAP dengan terlebih dahulu membaca deskripsi servis yang terdapat dalam dokumen WSDL yang telah disediakan. Dalam proses pembacaan WSDL, sistem penghitungan rekening harus melakukan *request* via

HTTP ke *server* melalui URI yang menunjukkan dimana dokumen WSDL tersebut berada.

3.2 Perancangan Data

Perancangan data masukan maupun keluaran dibutuhkan pada setiap sistem yang terlibat, baik di sistem pendistribusian data maupun sistem penghitungan rekening.

3.2.1 Perancangan Data pada Sistem Pendistribusian Data

a. Data masukan

Data masukan yang digunakan oleh sistem pendistribusian data dibagi menjadi dua yaitu data masukan hasil query data ke SPD, dan data masukan dari sistem penghitungan rekening yang berupa data pelanggan yang telah dilakukan proses penghitungan rekening.

Adapun hasil dari query data ke SPD adalah berupa data tarif dasar listrik dan data pelanggan. Spesifikasi data tarif dasar listrik dapat dilihat pada bagian Lampiran.

b. Data keluaran

Data keluaran yang dihasilkan oleh sistem pendistribusian data adalah data hasil penghitungan rekening dari sistem penghitungan rekening yang akan disimpan ke SPD.

Disamping data keluaran dan data masukan diatas di dalam *webservice* juga harus dibuat sebuah dokumen WSDL sebagai deskripsi dan *interface* publik dari servis yang akan disediakan.

3.2.2 Perancangan Data pada Sistem Penghitungan Rekening

a. Data masukan

Data masukan yang digunakan oleh sistem penghitungan rekening merupakan data tarif dasar listrik dan data pelanggan yang telah dimintakan ke sistem pendistribusian data, dimana data – data ini telah dibungkus di dalam pesan SOAP dalam bentuk XML. Data – data ini kemudian disimpan dalam memori aplikasi dalam suatu skema data yang menyerupai *recordset*, dan kemudian dijadikan sumbaer data untuk melakukan penghitungan rekening.

b. Data keluaran

Data keluaran yang dihasilkan oleh sistem penghitungan rekening adalah data hasil penghitungan rekening yang akan dikirimkan kembali ke sistem pendistribusi data melalui pemanggilan fungsi yang telah disediakan oleh *webservice* serta pesan kepada aplikasi pemonitor sistem mengenai state mulai penghitungan dan state selesai.

3.2.3 Perancangan Data pada Aplikasi Pemonitor Sistem

Aplikasi ini digunakan untuk menjalankan *webservice* dan memonitor unjuk kerja sistem. Aplikasi ini akan menerima data masukan berupa pesan – pesan yang menunjukkan aktifitas mesin – mesin dalam sistem penghitungan rekening selama

proses penghitungan rekening berlangsung dari mesin – mesin yang menjalankan proses penghitungan.

Pesan – pesan yang diterima oleh aplikasi pemonitor sistem ini akan direpresentasikan ke dalam *state – state* dari proses penghitungan rekening yang berlangsung pada tiap – tiap mesin di dalam sistem penghitungan rekening yang kemudian peristiwa – peristiwa ini akan digambarkan lewat antarmuka aplikasi.

3.3 Perancangan Proses

Proses – proses yang terjadi selama sistem bekerja adalah sebagai berikut:

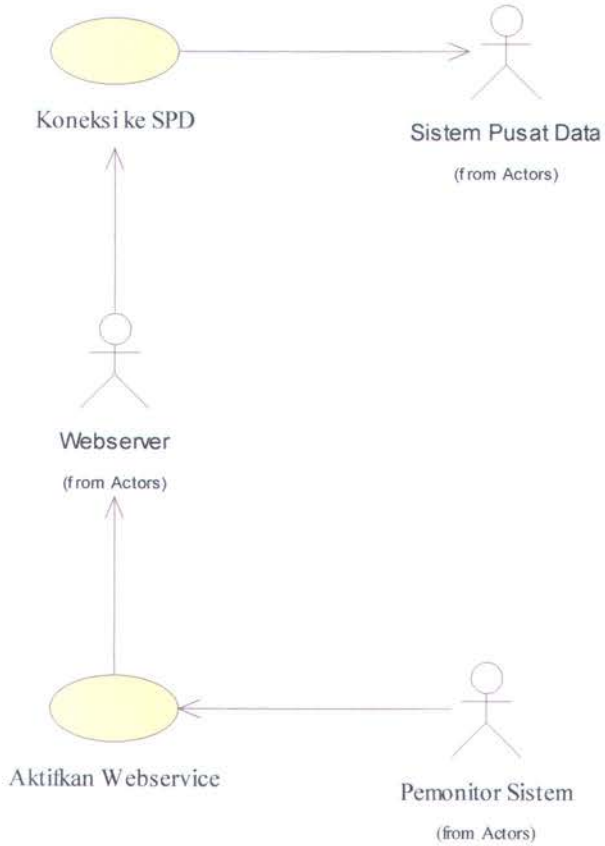
a. Proses aktivasi *webservice*.
b. Proses pendistribusian data.
c. Proses penghitungan rekening

3.3.1 Perancangan Proses Aktivasi *Webservice*

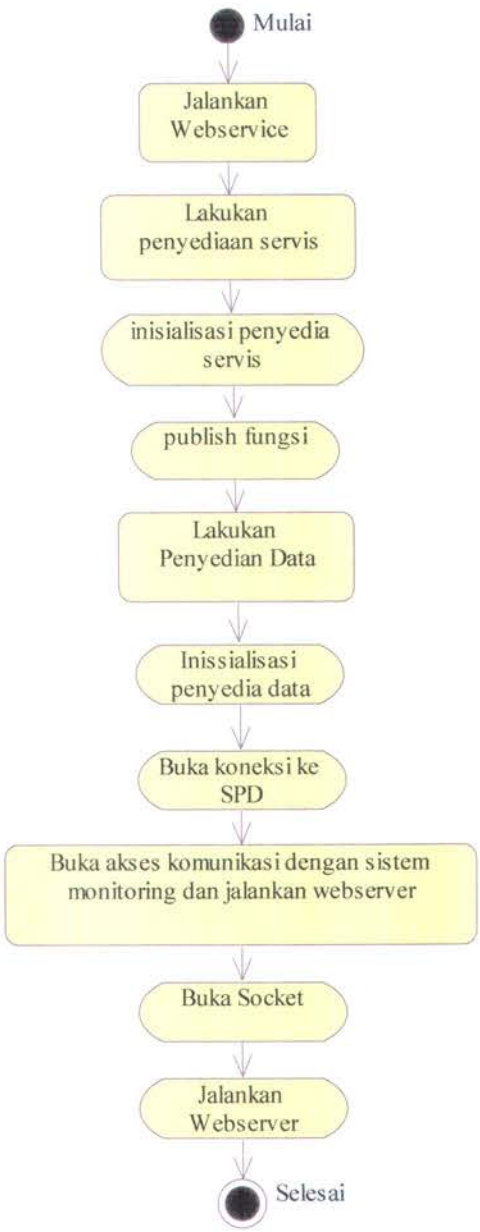
Proses ini menangani kegiatan pengkaktifan *webservice* yang berjalan pada sebuah *Webserver*. *Webserver* yang digunakan untuk keperluan pengadaan *webservice* itu sendiri merupakan sebuah aplkasi sederhana yang dapat melayani pemanggilan fungsi serta publikasi *interface* publik dari *webservice* melalui protokol transportasi HTTP.

Proses aktivasi dimulai dengan mengaktifkan *webservice* melalui aplikasi pemonitor sistem, dimana di dalamnya terdapat proses – proses aktivasi *Webserver*, dan inisialisasi penyediaan servis, serta inisialisasi penyediaan data

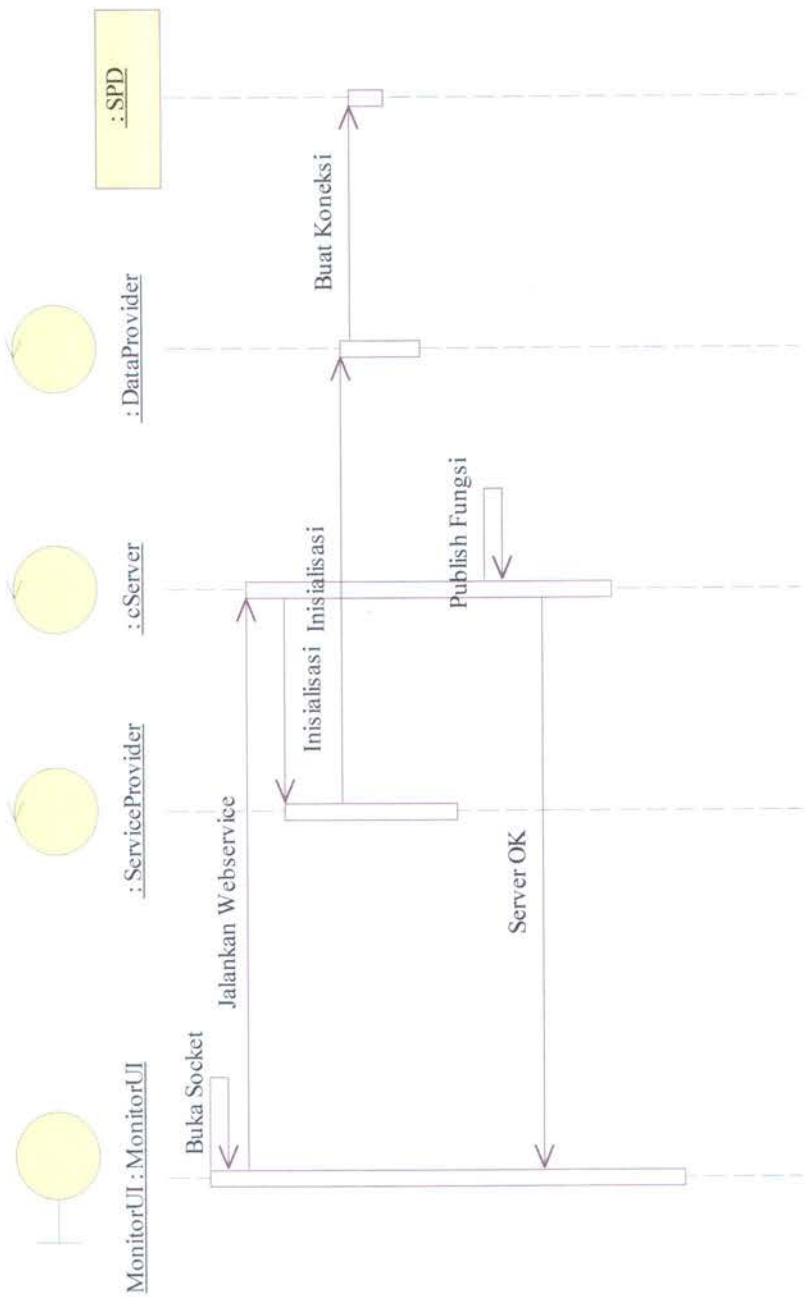
Proses – proses tersebut dapat digambarkan melalui diagram – diagram perancangan berikut ini:



Gambar III-1 Diagram *Use-Case* Proses Aktivasi *Webservice*



Gambar III-2 Diagram Aktifitas Pada Proses Aktivasi *Webservice*



Gambar III-3 Diagram Sekuen dari Proses Aktivasi *Webservice*

Gambar diagram *use-case* diatas merupakan pemodelan *use-case* yang didapatkan dari proses aktivasi *webservice* di dalam sistem pendistribusian data. Berikut ini penjelasannya.

Tabel III-1 Keterangan Diagram *Use-Case* Proses Aktivasi *Webservice*

Nama	**Stereotype**	**Keterangan**
Sistem Pusat Data (SPD)	Aktor	Sistem yang menyimpan data – data operasional PT. PLN, termasuk di dalamnya data tarif dasar listrik dan data pelanggan yang dibutuhkan untuk melakukan proses penghitungan rekening.
Webserver	Aktor	Engine dari sistem pendistribusian data yang mampu menyediakan layanan *webservice*.
Pemonitor Sistem	Aktor	Aplikasi pemonitor sistem yang memantau aktifitas mesin – mesin sistem penghitungan rekening dalam melakukan proses penghitungan rekening.

Koneksi ke SPD	*Use-Case*	*Use-case* yang menunjukkan proses pembukaan koneksi dari sistem pendistribusian data ke SPD.
Aktifkan *webservice*	*Use-Case*	*Use-Case* yang menunjukkan proses pengkaktifan *webservice*.

Setelah aplikasi *webservice* dijalankan oleh aplikasi pemonitor sistem, maka aplikasi *webservice* akan melakukan proses aktivasi dengan melakukan persiapan penyediaan servis dan penyediaan data yang diambil dari SPD.

Proses penyediaan servis meliputi penyediaan protokol pengiriman data XML lewat HTTP (SOAP) dengan menggunkan modul SOAPpy dan publikasi fungsi:

- **getData**, yaitu fungsi yang dapat digunakan oleh sistem penghitungan rekening untuk mendapatkan data tarif dasar listrik dan data pelanggan dari SPD yang dijadikan sumber data untuk proses penghitungan rekening.
- **sendData**, yaitu fungsi yang dapat digunakan oleh sistem penghitungan rekening untuk menyimpan data pelanggan sebagai hasil dari proses penghitungan rekening yang telah dilakukan ke SPD.

Sedangkan proses penyediaan data adalah proses pembukaan koneksi ke SPD yang menggunakan Oracle10g sebagai sistem manajemen databasenya. Pengkaksesan Oracle10g pada SPD menggunakan modul cx_Oracle. Koneksi

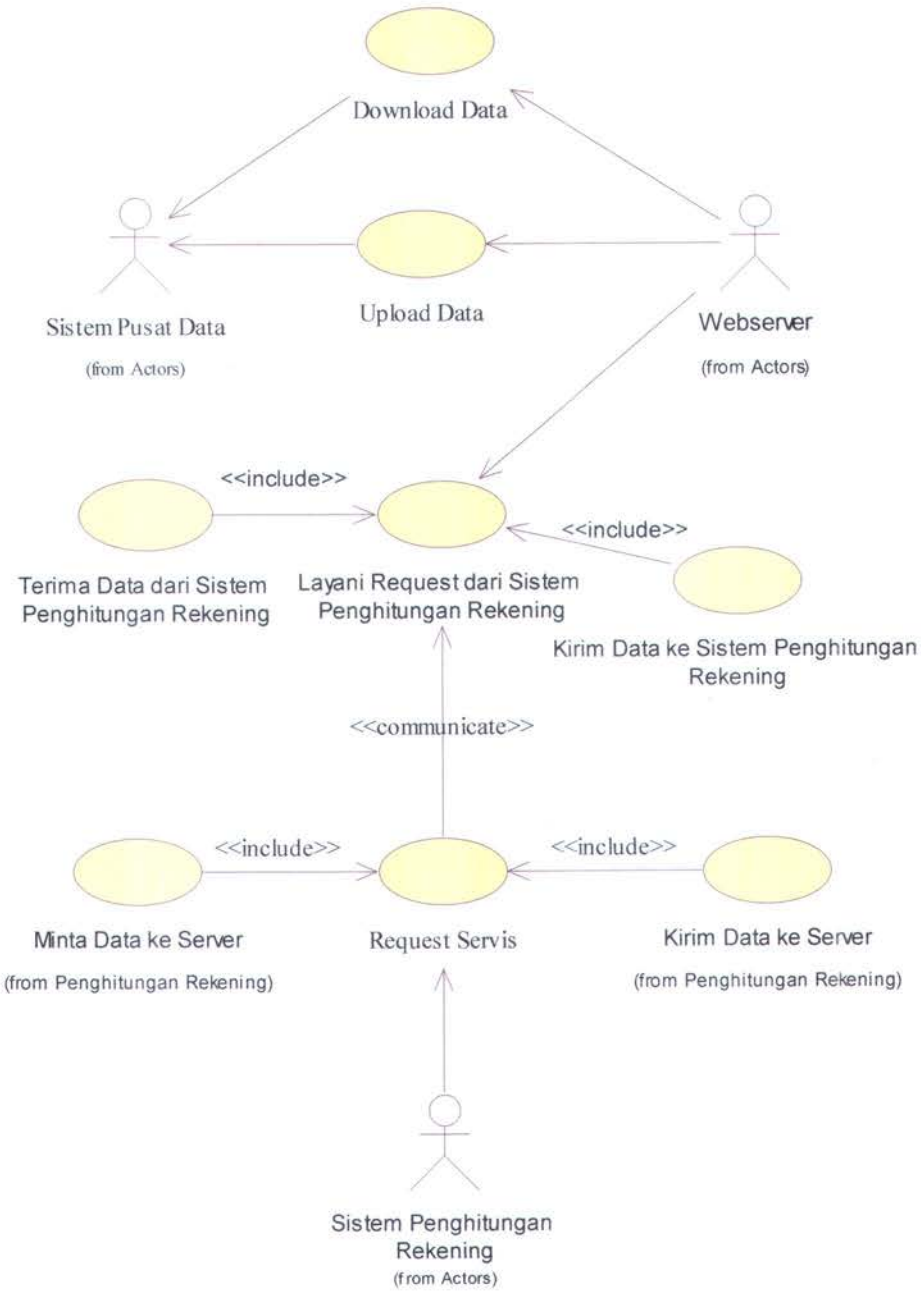
inilah yang akan digunakan untuk pengambilan data dari SPD untuk kemudian dikirimkan ke sistem penghitungan rekening untuk pemrosesan lebih lanjut.

3.3.2 Perancangan Proses Pendistribusian Data

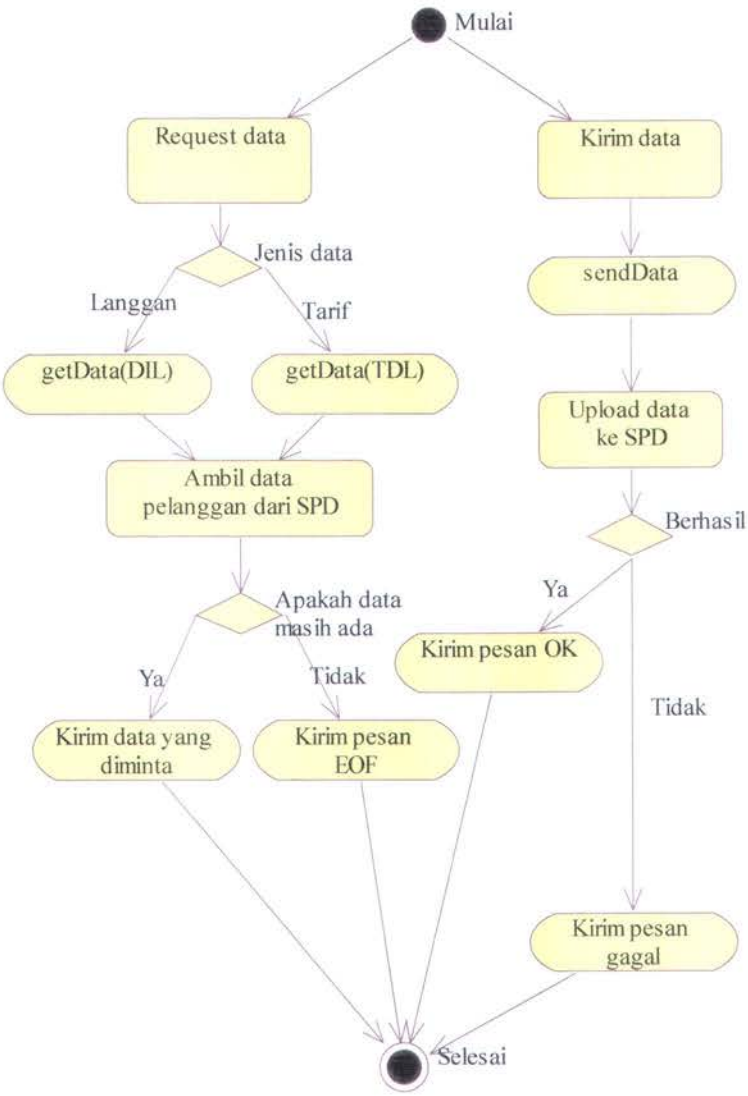
Proses pendistribusian data bertugas menangani permintaan dan pengiriman data yang dilakukan oleh sistem penghitungan rekening. Proses ini adalah realisasi dari teknologi *webservice* yang akan diterapkan dalam membangun sistem ini.

Proses pendisribusian data ini sendiri melputi empat proses utama yakni proses download, dan upload data dari dan ke SPD, serta proses *request* servis oleh sistem penghitungan rekening dan proses layani *request* yang dilakukan oleh sistem pendistribusi data dalam melayani permintaan dan pengiriman data yang dilakukan oleh sistem penghitungan rekening.

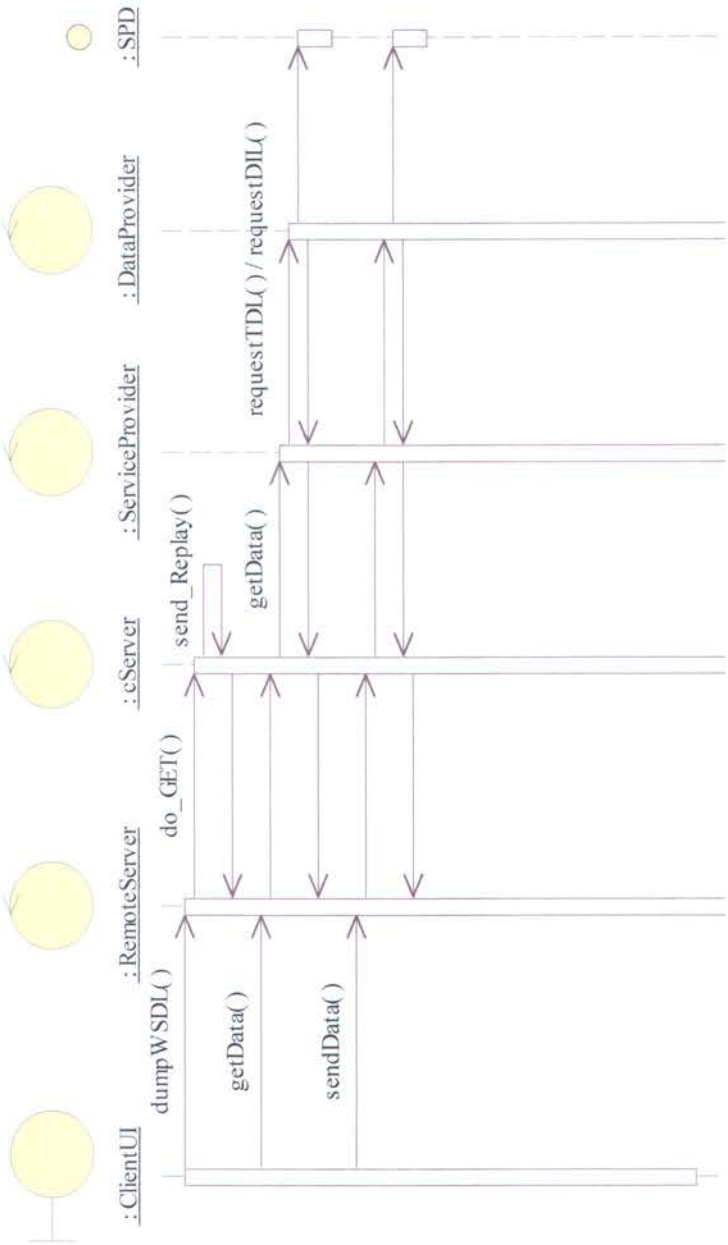
Untuk lebih jelasnya berikut ini disajikan diagram – diagram perancangan untuk menggambarkan proses pendistribusian data:



Gambar III-4 Diagram *Use-Case* Proses Pendistribusian Data



Gambar III-5 Diagram Aktifitas Proses Pendistribusian Data



Gambar III-6 Diagram Sekuensial Proses Pendistribusian Data

Gambar diagram *use-case* diatas merupakan pemodelan *use-case* yang didapatkan dari proses pendistribusian data di dalam sistem pendistribusian data. Berikut ini penjelasannya.

Tabel III-2 Keterangan Diagram *Use-Case* Proses Pendistribusian Data

Nama	**Stereotype**	**Keterangan**
Sistem Pusat Data (SPD)	Aktor	Sistem yang menyimpan data – data operasional PT. PLN, termasuk di dalamnya data tarif dasar listrik dan data pelanggan yang dibutuhkan untuk melakukan proses penghitungan rekening.
Webserver	Aktor	Engine dari sistem pendistribusian data yang mampu menyediakan layanan *webservice*.
Sistem Penghitungan Rekening	Aktor	Gabungan beberapa mesin yang bekerja bersama – sama memproses penghitungan rekening.
Download Data	*Use-Case*	*Use-case* yang menunjukkan proses pengambilan (query) data dari SPD.

Upload Data	*Use-Case*	*Use-case* yang menunjukkan proses penyimpanan data ke SPD.
Terima Data dari Sistem Penghitungan Rekening	*Use-Case*	*Use-case* yang menunjukkan proses penerimaan data yang dikirimkan oleh sistem penghitungan rekening.
Kirim Data ke Sistem Penghitungan Rekening	*Use-Case*	*Use-case* yang menunjukkan proses pengiriman data ke sistem penghitungan rekening.
Layani *Request* dari Sistem penghitungan Rekening	*Use-Case*	*Use-case* yang menunjukkan proses pelayanan permintaan sistem penghitungan rekening. Temasuk di dalamnya adalah proses penerimaan data dari sistem penghitungan rekening serta pengiriman data.ke sistem penghitungan rekening.
Minta Data ke *Server*	*Use-Case*	*Use-case* yang menunjukkan proses permintaan data ke sistem pendistribusian data.

Kirim Data ke *Server*	*Use-Case*	*Use-case* yang menunjukkan proses pengiriman data ke sistem pendistribusian data.
Request Servis	*Use-Case*	*Use-case* yang menunjukkan proses permintaan servis yang terdapat pada *webservice*. Termasuk di dalamnya adalah proses permintaan dan pengiriman data ke sistem pendistribusian data.

Setelah melalui proses aktivasi, *Webserver* siap untuk melayani permintaan servis dari sistem penghitungan rekening. Pada diagram diatas terdapat empat proses utama yang dijalankan oleh sistem, yaitu:

a. *Request* Servis

Proses ini digunakan oleh sistem penghitungan rekening untuk melakukan pemanggilan fungsi untuk mendapatkan data pelanggan ataupun data tarif (**getData**) yang selanjutnya dilakukan penghitungan rekeningnya. Proses ini juga untuk memanggil fungsi untuk mengirimkan data hasil penghitungan rekening (**sendData**) yang akan disimpan ke SPD. Baik proses permintaan dan pengiriman data harus telah disediakan oleh *webservice*.

b. Layani *Request* dari Sistem Penghitungan Rekening

Proses ini digunakan *server* untuk melayani pemanggilan fungsi yang dilakukan oleh mesin – mesin di dalam sistem penghitungan rekening. Proses ini menjadi satu dengan *Webserver* dan akan terus berjalan untuk melayani penggunaan servis hingga *Webserver* dimatikan. Pelayanan *request* ini dilakukan secara sinkron, artinya *request* – *request* yang masuk akan dimasukkan ke dalam antrian dan dilayani dengan sistem FIFO (*First In First Out*) yakni permintaan yang masuk lebih dulu akan dilayani lebih dulu baru kemudian permintaan – permintaan berikutnya sesuai urutan waktu kedatangan permintaan.

Data yang dikirimkan ke sistem penghitungan rekening akan dibungkus dengan pesan SOAP yang secara otomatis dilakukan oleh modul SOAPpy dan dibaca oleh sistem pennghitungan rekening dengan menggunakan *library* MSSOAP.1.

c. Download Data

Proses digunakan untuk mendapatkan data, baik data pelanggan maupun data tarif dasar listrik, yang diminta oleh sistem penghitungan rekening melalui pemanggilan fungsi yang telah disediakan dalam *webservice*. Data – data ini kemudian dikirimkan ke sistem penghitungan rekening.

d. Upload Data

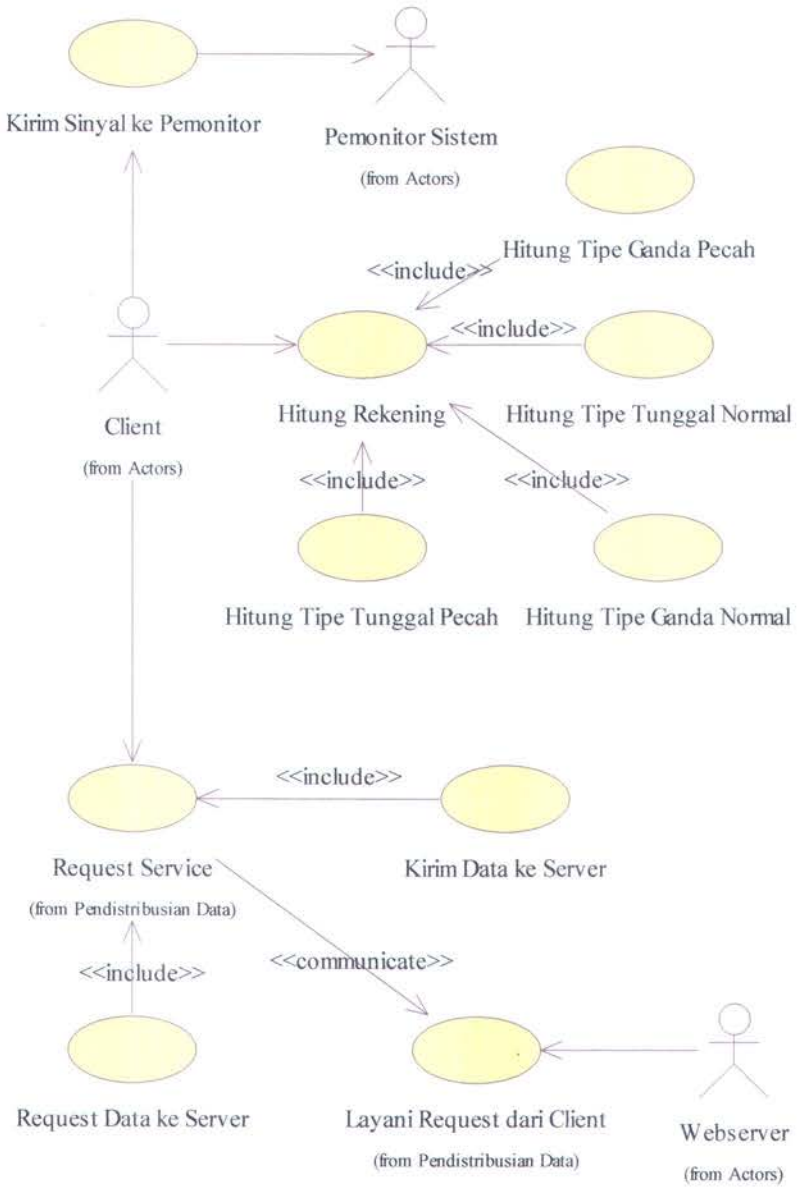
Proses ini digunakan untuk menyimpan data hasil penghitungan rekening yang dilakukan oleh sistem penghitungan rekening yang dikirimkan melalui pemanggilan fungsi **sendData** yang telah disediakan dalam *webservice*.

Semua proses komunikasi berupa pemanggilan dan pengiriman serta pesan – pesan lainnya yang dilakukan antara sistem pendistribusian data dengan sistem penghitungan rekening dilakukan dengan menggunakan protokol SOAP melalui protokol transportasi HTTP.

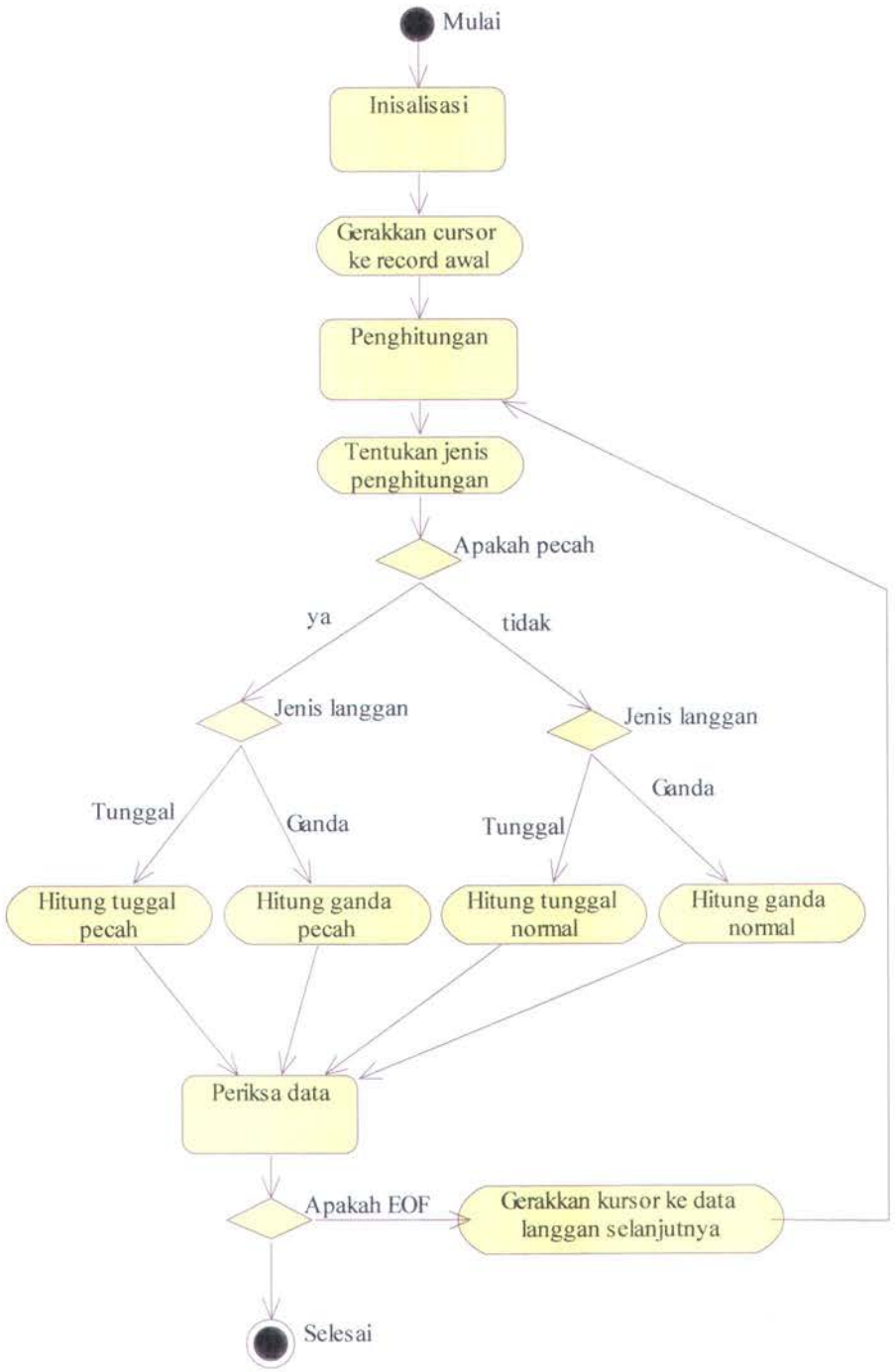
3.3.3 Perancangan Proses Penghitungan Rekening

Proses penghitungan rekening ini merupakan proses yang berjalan di dalam sistem penghitungan rekening. Di dalam proses penghitungan rekening inilah proses bisnis diakomodasi guna mendapatkan tagihan pemakaian listrik tiap – tiap pelanggan pada setiap bulannya.

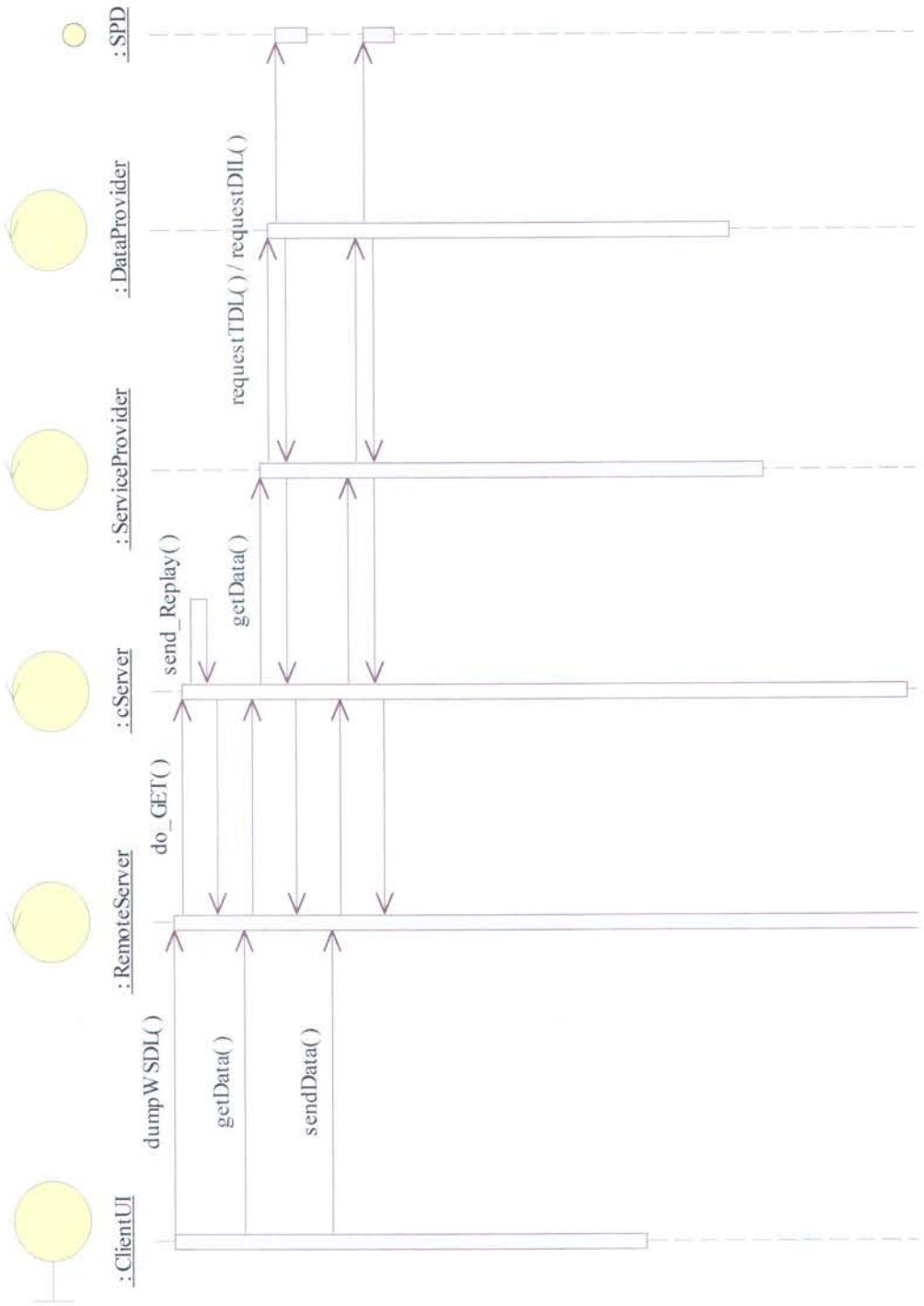
Berikut ini diagram – diagram perancangan untuk proses penghitungan rekening:



Gambar III-7 Diagram *Use Case* Proses Penghitungan Rekening



Gambar III-8 Diagram Aktifitas Proses Penghitungan Rekening



Gambar III-9 Diagram Sekuensial Proses Penghitungan Rekening

Gambar diagram *use-case* diatas merupakan pemodelan *use-case* yang didapatkan dari proses penghitungan rekening. Berikut ini penjelasannya.

Tabel III-3 Keterangan Diagram *Use-Case* Proses Penghitungan Rekening

Nama	**Stereotype**	**Keterangan**
Webserver	Aktor	Engine dari sistem pendistribusian data yang mampu menyediakan layanan *webservice*.
Sistem Penghitungan Rekening	Aktor	Gabungan beberapa mesin yang bekerja bersama – sama memproses penghitungan rekening.
Pemonitor Sistem	Aktor	Aplikasi pemonitor sistem yang memantau aktifitas mesin – mesin sistem penghitungan rekening dalam melakukan proses penghitungan rekening.
Layani *Request* dari Sistem penghitungan	*Use-Case*	*Use-case* yang menunjukkan proses pelayanan permintaan sistem penghitungan rekening. Temasuk di

Rekening		dalamnya adalah proses penerimaan data dari sistem penghitungan rekening serta pengiriman data.ke sistem penghitungan rekening.
Minta Data ke *Server*	*Use-Case*	*Use-case* yang menunjukkan proses permintaan data ke sistem pendistribusian data.
Kirim Data ke *Server*	*Use-Case*	*Use-case* yang menunjukkan proses pengiriman data ke sistem pendistribusian data.
Request Servis	*Use-Case*	*Use-case* yang menunjukkan proses permintaan servis yang terdapat pada *webservice*. Termasuk di dalamnya adalah proses permintaan dan pengiriman data ke sistem pendistribusian data.
Hitung Rekening	*Use-Case*	*Use-case* yang menunjukkan proses penghitungan rekening.
Hitung Tipe	*Use-Case*	*Use-case* yang menunjukkan proses

Tunggal Normal		penghitungan rekening dengan pelanggan tunggal tanpa adanya pecahan yang disebabkan oleh perubahan data.
Hitung Tipe Ganda Normal	*Use-Case*	*Use-case* yang menunjukkan proses penghitungan rekening dengan pelanggan ganda tanpa adanya pecahan yang disebabkan oleh perubahan data.
Hitung Tipe Tunggal Pecah	*Use-Case*	*Use-case* yang menunjukkan proses penghitungan rekening dengan pelanggan tunggal dengan menghitung secara pecahan yang disebabkan oleh perubahan data.
Hitung Tipe Ganda Pecah	*Use-Case*	*Use-case* yang menunjukkan proses penghitungan rekening dengan pelanggan ganda dengan menghitung secara pecahan yang disebabkan oleh perubahan data.

Seperti yang telah dijelaskan di atas, ada dua jenis data yang akan digunakan untuk melakukan proses penghitungan rekening yaitu data tarif dasar listrik dan

data pelanggan. Data – data terlebih dahulu dimintakan ke *webservice* dengan memanggil fungsi **getData** dan kemudian disimpan ke dalam memori untuk pemakaian selanjutnya hingga proses penghitungan selesai. Penyimpanan data ke memori harus dapat diakses dengan mudah, sehingga skema data yang akan disimpan dirancang dengan sedemikian rupa sehingga menyerupai skema *recordset*, dimana dari skema tersebut dapat diperoleh kemampuan sebagai berikut:

- Menandai *record* data yang sekarang di proses, yaitu dengan menggunakan kursor yang dapat digerakkan.
- Menggerakkan kursor ke *record* selanjutnya.
- Menggerakkan kursor ke *record* sebelumnya.
- Menggerakkan kursor ke awal *record* .
- Menggerakkan kursor ke akhir *record.*
- Mendeteksi apakah kursor berada pada awal atau akhir dari *recordset*.

Namun sebelum data disimpan ke memori, data mentah yang berupa pesan SOAP yang diterima dari *webservice* harus diekstraksi terlebih dahulu .

Terdapat dua jenis data pelanggan, yaitu data pelanggan tunggal dan data pelanggan ganda dimana masing – masing data memiliki jumlah field yang berbeda (Lihat spesifikasi tabel pelanggan pada lampiran). Oleh karena itu proses penghitungan diawali dengan menentukan jenis data pelanggan yang akan diproses. Setelah jenis data diketahui maka langkah selanjutnya adalah menentukan jenis penghitungan apakah termasuk dalam jenis tunggal normal,

ganda normal, tunggal pecah atau ganda pecah. Mengapa bisa demikian? Berikut ini penjelasannya.

Seorang pelanggan dapat setiap saat melakukan permohonan perubahan – perubahan atas properti kelistrikan yang diperolehnya dari PT. PLN. Perubahan – perubahan tersebut antara lain perubahan tarif, daya, kode daya, nama, alamat, dan lain – lain yang dalam istilah operasioanl PT. PLN disebut sebagai mutasi. Perubahan – perubahan ini ada yang mempengaruhi penghitungan rekening pelanggan, seperti misalnya perubahan tarif dan daya listrik yang mengharuskan adanya penghitungan yang terpisah antara pemakaian tarif dan daya lama dengan tarif dan daya baru sesuai dengan proporsi penggunaannya apabila perubahan ini terjadi di dalam satu periode operasional. Inilah yang menyebabkan adanya penghitungan pecahan pada proses penghitungan rekening. Indikator dari adanya perlakuan penghitungan pecahan pada data langgan adalah bulan tahun mutasi yang terdapat pada kolom BLTH_MTSI sesuai dengan bulan tahun yang didefinisikan sistem, dan mengandung jenis mutasi D, E, J, K, L, N yang menunjukkan adanya perubahan tarif, daya, kode daya yang menyebabkan data harus diproses secara pecahan.

Langkah selanjutnya adalah melakukan penghitungan rekening sesuai tipe penghitungan yang telah ditentukan dengan rincian proses sebagai berikut:

- Penghitungan biaya abonemen atau biaya beban dari pemakaian jasa penyediaan listrik sesuai dengan tarif dan daya masing – masing.

- Penghitungan pemakaian daya listrik serta biaya yang harus ditanggung atas pemakaian daya listrik tersebut.
- Penghitungan biaya total yang harus dibayarkan dengan menjumlahkan biaya beban dan biaya pemakaian daya listrik.
- Pembaruan data pelanggan sesuai dengan data yang diperoleh dari proses penghitungan yang meliputi, pemakaian pada daya listrik pada blok I,II, dan III, pemakain daya listrik secara keseluruhan, rupiah biaya beban, rupiah biaya pemakaian, dan rupiah total.

Setelah proses – proses diatas selesai maka dilanjutkan dengan data pelanggan selanjutnya hingga data terproses semua. Kemudian data yang hasil penghitungan ini dirubah kembali ke skema awal ketika mengekstraksi dari pesan SOAP yang dikirim oleh *webservice*, dan kemudaian data dikirimkan kembali ke *webservice* dengan memanggil fungsi **sendData** yang akan membungkus data dengan pesan SOAP sebelum ditransportasikan.

Proses penghitungan yang terdiri dari permintaan, penghitungan, dan pengiriman data akan dilakukan terus – menerus hingga data yang ada telah didistribusikan serta diproses seluruhnya.

Untuk dapat mengukur unjuk kerja sistem dalam melakukan proses penghitungan rekening terdistribusi ini, maka diperlukan proses monitoring yang dilakukan oleh sebuah aplikasi tersendiri. Aplikasi ini akan berkomunikasi dengan mesin penghitung rekening pada setiap *state*-nya. Ada beberapa *state* yang mengharuskan mesin mengirimkan pesan ke aplikasi pemonitor sistem, yaitu:

a. Ketika memulai proses penghitungan rekening.
b. Ketika proses penghitungan rekening selesai.
c. Ketika mesin penghitung rekening menerima pesan dari webservice bahwa data telah mencapai EOF.

Di antara poin a dan poin b dianggap ada aktifitas penghitungan rekening pada mesin penghitung rekening, sedangkan di antara poin b dan poin a dianggap sebagai *idle*.

Dari aplikasi ini akan didapatkan waktu komputasi untuk menyelesaikan proses penghitungan rekening secara keseluruhan maupun pada tiap – tiap mesin penghitung yang terdapat dalam sistem penghitungan rekening.

3.4 Perancangan Antarmuka

Antar muka digunakan sebagai sarana interaksi pengguna dengan perangkat lunak yang terlibat di dalam sistem ini.

3.4.1 Perancangan Antarmuka *Webserver*

Antar muka untuk *Webserver* cukup dengan mode *console* yang dapat dipanggil melalui aplikasi pemonitor sistem untuk memudahkan pengoperasian.

3.4.2 Perancangan Antarmuka Sistem penghitungan rekening

Antar muka aplikasi penghitungan rekening harus dapat menyediakan fasilitas – fasilitas sebagai berikut:

a. Tombol atau menu unutk menjalankan proses penghitungan rekening.
b. Tombol atau menu untuk menentukan dan menyimpan URI dari WSDL yang digunakan sebagai *interface* publik.

3.4.3 Perancangan Antarmuka Pemonitor Sistem

Antar muka pemonitor sistem haruslah menyediakan fasilitas – fasilitas sebagai berikut:

a. Tombol atau menu untuk menjalankan dan menghentikan *Webserver*.
b. Tampilan pemantauan aktifitas penghitungan rekening yang dilakukan oleh sistem penghitungan rekening.
c. Tampilan waktu komputasi total dan waktu komputasi tiap – tiap mesin penghitung rekening di dalam sistem penghitungan rekening.

BAB IV

IMPLEMENTASI SISTEM

4.1 Implementasi Arsitektur

Implementasi dari perancangan arsitektur merupakan penyediaan lingkungan sistem yang teridiri dari perangkat keras dan perangkat lunak. penjelasan spesifikasi perangkat keras dan perangkat lunak dalam pembangunan sistem ini adalah sebagai berikut:

Sistem pusat data disimulasikan dengan memanfaatkan *server* database berbasis Oracle10g yang tersedia di laboratorium Rekayasa Perangkat Lunak Teknik Informatika ITS. Mesin ini menggunakan prosesor Intel Pentium IV dengan kecepatan proses 3 GHz. Dengan spesifikasi seperti ini, mesin akan mampu melayani proses pengambilan dan penyimpanan data dengan baik karena telah memnuhi kebutuhan sistem untuk menjalankan Oracle10g.

Sistem pendistribusian data diimplementasikan dengan menggunakan sebuah sebuah komputer HP-Vectra yang menggunakan prosesor Intel Pentium II dengan kecepatan proses 400 MHz. Dengan spesifikasi seperti ini mesin mampu menjalankan proses pendistribusian data dengan baik dengan catatan mesin tidak menjalankan aplikasi lain yang membutuhkan *resource* besar. Unjuk kerja sistem ini akan lebih meningkat dengan spesifikasi yang lebih baik.

Dan di sisi sistem penghitungan rekening diimplementasikan dengan menggunakan lima buah komputer ACER yang menggunakan prosesor Intel Pentium I dengan kecepatan proses 133 MHz. Dengan spesifikasi ini mesin masih mampu melakukan proses penghitungan dengan baik dengan catatan tidak sistem operasi menjalankan aplikasi lain yang memakan *resoruce* besar. Dan unjuk kerja sistem akan semakin meningkat dengan spesifikasi yang lebih besar.

Baik sistem pendisribusi data maupun sistem penghitungan rekening, keduanya berada di dalam Laboratorium Pemrograman Teknik Informatika ITS.

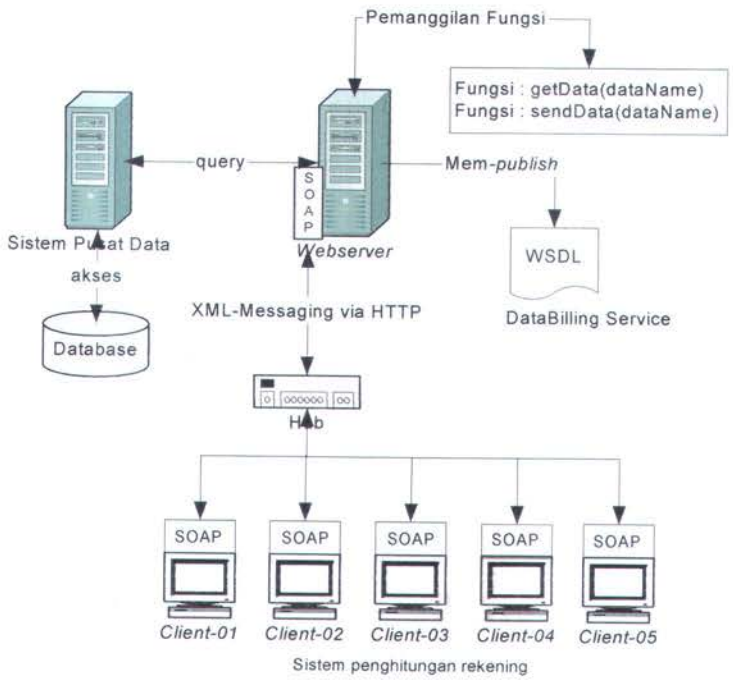
Berikut ini tabel spesifikasi perangkat lunak dan keras secara lebih lengkap:

Tabel IV-1 Spesifikasi Perangkat Keras dan Lunak

Sistem Pusat Data	
Perangkat Keras	Prosesor : Intel Pentium IV 3 GHz RAM : 1534 MB
Perangkat Lunak	Sistem Operasi : Microsoft Windows 2003 *Server* DBMS : Oracle 10g *Server*
***Webserver* (Sistem Pendistribusian Data)**	
Perangkat Keras	Prosesor : Intel Pentium II 400 MHz RAM : 384 MB
Perangkat Lunak	Sistem Operasi : Microsoft Windows XP Professional SP.1

	Compiler & Tools : Python 2.3, Pythonwin, Microsoft Visual Studio 6 Library & Component : cx_Oracle, SOAPpy, VSVlexgrid Pro.
Mesin Penghitung (Sistem Penghitungan Rekening)	
Perangkat Keras	Prosesor : Intel Pentium 133 MHz RAM : 64 MB
Perangkat Lunak	Sistem Operasi : Microsoft Windows 98 Compiler & Tools : Microsoft Visual Studio 6 Library & Component : MSSOAP.1, VSVlexgrid Pro.

Sedangkan arsitektur sistem yang akan dibangun tersebut dapat digambarkan sebagai berikut:



Gambar IV-1 Arsitektur Sistem

4.2 Implementasi Data

Berikut ini akan disajikan implementasi dari perancangan data yang telah dilakukan sebelumnya.

4.2.1 Implementasi Data pada Sistem Pendistribusian

a. Data Masukan

Aplikasi sistem pendistribusian data ini memiliki dua macam data masukan, yakni data hasil *query* dari Oracle10g (SPD) dan data hasil penghitungan rekening yang dikirimkan oleh sistem penghitungan rekening.

Pengambilan data ke SPD ini menggunakan modul cx_Oracle dimana hasilnya berupa *List of tuple*. *List* merupakan tipe data dari Python yang memiliki bentuk sebagai berikut:

```
[x1, x2, x3,...,xn]
```

dimana x1, x2, x3 hingga xn dapat bertipe apapun. Sedangkan *tuple* merupakan tipe data dari Python yang memiliki bentuk sebagai berikut:

```
(x1, x2, x3,...,xn)
```

dimana x1, x2, x3 hingga xn dapat bertipe apapun. Sehingga *List of Tuple* maksudnya adalah himpunan *tuple – tuple* yang disimpan dalam sebuah *list*.

Sedangkan contoh dari hasil pengambilan data menggunakan cx_Oracle dengan query **"SELECT idpel, nama, tarif, daya FROM diltgl WHERE rownum<=5"**adalah sebagai berikut:

```
[('511200003026', 'H CHOLIFAH', 'B1', '000900'),
('511200004364', 'BUDI SOETRISNO', 'B1','001300'),
('511200004372', 'RUSMAWATI', 'R1', '000900'),
('511200005884', 'PAK AS AD', 'R1', '000900'),
('511200005892', 'JEMU', 'R1', '000900')]
```

Sebelum data ini dikirimkan, terlebih dahulu data tersebut siubah ke bentuk *string* dengan sebagai berikut:

- Setiap *record* data dipisahkan dengan string "{-=record=-}".
- Setiap *field* data dipisahkan dengan karakter Tab ("\t").

Sehingga data keluarannya akan berbentuk seperti gambar di bawah ini:

```
511200003026'\t'H CHOLIFAH'\t'B1'\t'000900{-=record=-}
511200004364'\t'BUDI SOETRISNO'\t'B1'\t'001300{-=record=-}
511200004372'\t'RUSMAWATI'\t'R1'\t'000900{-=record=-}
511200005884'\t'PAK AS AD'\t'R1'\t'000900{-=record=-}
511200005892'\t'JEMU'\t'R1'\t'000900
```

Semantara itu data masukan lainnya adalah data hasil penghitungan rekening yang dikirimkan oleh sistem penghitungan rekening. Dalam hal ini sistem penghitungan dipakas untuk mengikuti format pengiriman data sebgaimana yang dijelaskan di atas.

b. Data Keluaran

Data keluaran dari sistem ini adalah data hasil pengambilan dari SPD yang kemudian diubah dalam format yang telah dijelaskan diatas. Kemudian data ini akan dikirimkan dengan menggunakan SOAP melalui protokol transportasi HTTP.

Selain itu sistem harus dilengkapi dengan sebuah dokumen WSDL yang mendeskripsikan serta menjadi *interface* publik *webservice* yang disediakan. Adapun dokumen WSDL yang dibuat adalah sebagai berikut:

```
<?xml version="1.0"?>
<definitions name="databillingService"
targetNamespace="http://webservice:8080/wsdl/databilling.wsdl"
xmlns:tns="http://webservice:8080/wsdl/databilling.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="getDataRequest">
    <part name="dataName" type="xsd:string"/>
  </message>
```

```
<message name="getDataResponse">
  <part name="dataName" type="xsd:string"/>
  <part name="dataResult" type="xsd:string"/>
</message>
<message name="sendDataRequest">
  <part name="dataName" type="xsd:string"/>
  <part name="theData" type="xsd:string"/>
</message>
<message name="sendDataResponse">
  <part name="status" type="xsd:string"/>
</message>
<portType name="databillingPortType">
  <operation name="getData">
    <input message="tns:getDataRequest"/>
    <output message="tns:getDataResponse"/>
  </operation>
  <operation name="sendData">
    <input message="tns:sendDataRequest"/>
    <output message="tns:sendDataResponse"/>
  </operation>
</portType>
<binding name="databillingBinding"
 type="tns:databillingPortType">
<soap:binding style="rpc"
 transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getData">
<soap:operation soapAction=""/>
  <input>
    <soap:body use="encoded" namespace="urn:data-billing"
     encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
    <soap:body use="encoded" namespace="urn:data-billing"
     encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
</operation>
<operation name="sendData">
<soap:operation soapAction=""/>
  <input>
    <soap:body use="encoded" namespace="urn:data-billing"
     encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
    <soap:body use="encoded" namespace="urn:data-billing"
     encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
</operation>
</binding>
<service name="databillingService">
  <documentation>Melayani Permintaan dan Penyimpanan Data-
   Data Penghitungan Rekening Pelanggan</documentation>
  <port name="databillingPort"
```

```
      binding="tns:databillingBinding">
      <soap:address location="http://webservice:8080/"/>
    </port>
  </service>
</definitions>
```

Dokumen di atas menunjukkan dokumen WSDL untuk servis *dataBillingService* yang menyediakan fungsi **getData** dan **sendData**. Servis ini disediakan di alamat http://*webservice*:8080/.

4.2.2 Implementasi Data pada Sistem Penghitungan

a. Data Masukan

Data masukan dari sistem ini akan berupa pesan SOAP yang mengandung data hasil pengambilan dari SPD dengan format seperti yang telah dijelaskan pada bagian 4.2.1 poin a.

Data masukan yang telah diekstraksi tersebut kemudian akan disimpan dalam memori dimana bentuk penyimpanannya nanti harus memiliki kemampuan:

- Menandai *record* data yang sekarang di proses, yaitu dengan menggunakan kursor yang dapat digerakkan.
- Menggerakkan kursor ke *record* selanjutnya.
- Menggerakkan kursor ke *record* sebelumnya.
- Menggerakkan kursor ke awal *record* .
- Menggerakkan kursor ke akhir *record.*
- Mendeteksi apakah kursor berada pada awal atau akhir dari *recordset.*

Dan untuk mengimplementasikannya digunakan kelas DataLanggan dan kelas DataTarif berikut ini:

Kelas DataTarif

Kelas ini merupakan kelas entitas yang memiliki atribut – atribut untuk menyimpan data tarif dasar listrik yang diperoleh dari pemanggilan fungsi permintaan data ke *webservice*. Selain itu kelas ini juga memiliki fungsi – fungsi dan prosedur – prosedur untuk melakukan operasi – operasi pengaksesan data. Kelas ini diimplementasikan di sistem penghitungan rekening dengan menggunakan Visual Basic 6.0. berikut ini abstraksinya:

```
Const deftdl
Private m_recordset() As String
Private m_jmlrecord As Long
Private m_cursor As Long
Private m_dictfld As Scripting.Dictionary
Public Property Get Field(Index) As String
Public Property Let Field(Index, value As String)
Public Sub setData(rawstring As String)
Public Sub getJumlahRecord(ByRef jumlah As Integer)
Public Sub moveFirst()
Public Sub moveLast()
Public Sub moveNext()
Public Sub movePrevious()
Public Function isBOF() As Boolean
Public Function isEOF() As Boolean
Public Function FindTDLFor(tarif As String, daya As String)
As  Boolean
```

Berikut ini penjelasan dari masing – masing bagian dalam kelas DataTarif:

Tabel IV-2 Keterangan Variable Angggota Kelas Data Tarif

Nama Variabel Anggota dan Tipenya	**Keterangan**
Nama : deftdl	Digunakan untuk menyimpan nama-nama field

Tipe : String	dari tabel data tarif dasar listrik (TDL). Variabel ini akan dibaca dan diubah menjadi pemetaan nama-nama field dengan nomor urutnya dalam tabel data tarif dasar listrik. Pemetaan ini akan berupa *dictionary*.
Nama : m_recordset() Tipe : String	Digunakan untuk menyimpan data tarif dasar listrik, dimana sebuah elemen array berisi sebuah baris data.
Nama : m_jmlrecord Tipe : Long	Digunakan untuk menyimpan jumlah baris data yang disimpan.
Nama : m_cursor Tipe : Long	Digunakan untuk menyimpan informasi baris data yang sekarang ditunjuk. Variabel ini berguna untuk penanda pada saat kursor penunjuk baris data digerakkan ke depan maupun ke belakang.
Nama : m_dictfld Tipe : Dictionary	Digunakan untuk memetakan nama-nama field dengan urutan nomor urut field tersebut.

Tabel IV-3 Keterangan Fungsi Anggota Kelas DataTarif

Nama Fungsi Anggota	**Keterangan**
Field(Index)	Merupakan *interface* kelas yang berupa *property* untuk mendapatkan nilai data baris

	yang sekarang ditunjuk pada kolom yang didefinisikan dengan Index.
Field(Index, value As String)	Merupakan *interface* kelas yang berupa *property* untuk menmperbarui nilai data baris yang sekarang ditunjuk pada kolom yang didefinisikan dengan Index dengan nilai baru value.
setData(rawstring As String)	Merupakan *interface* kelas yang digunakan untuk mengisi variabel m_recordset dengan data rawstring.
getJumlahRecord(ByRef jumlah As Integer)	Digunakan untuk mendapatkan jumlah baris data yang disimpan.
moveFirst()	Digunakan untuk menggerakkan kursor ke baris data paling awal.
moveNext()	Digunakan untuk menggerakkan kursor ke baris data selanjutnya.
movePrevious()	Digunakan untuk menggerakkan kursor ke baris data sebelumnya.
moveLast()	Digunakan untuk menggerakkan kursor ke baris data paling akhir.
isBOF()	Digunakan untuk mendeteksi apakan kursor

	menunjukkan penanda awal data. Mengembalikan nilai TRUE jika benar dan FALSE jika tidak.
isEOF()	Digunakan untuk mendeteksi apakan kursor menunjukkan penanda akhir data. Mengembalikan nilai TRUE jika benar dan FALSE jika tidak.
`FindTDLFor(tarif As String, daya As String)`	Digunakan untuk mencari data tarif yang sesuai dengan tarif dan daya yang didefiniskan oleh parameter fungsi. Mengembalikan nilai TRUE jika berhasil menemukan dan FALSE jika tidak berhasil. Kursor akan menunjukkan posisi terakhir penggerakan.

Kelas DataLanggan

Kelas ini merupakan kelas entitas yang memiliki atribut – atribut untuk menyimpan data pelanggan yang diperoleh dari pemanggilan fungsi permintaan data ke *webservice*. Selain itu kelas ini juga memiliki fungsi – fungsi dan prosedur – prosedur untuk melakukan operasi – operasi pengaksesan data. Berikut ini abstraksinya:

```
Const defdiltgl
Const deffilgan
Private m_jmlrecord As Long
Private m_cursor As Long
```

```
Private m_isDilTgl As Boolean
Private m_dictfld As Scripting.Dictionary

Public Property Get Field(Index) As String
Public Property Let Field(Index, value As String)
Public Property Get isDilTgl() As Boolean
Public Sub setDILstatus(isTunggal As Boolean)
Public Sub getAllData(ByRef data As String)
Public Sub setData(ByRef rawstring As String)
Public Sub getJumlahRecord(ByRef jumlah As Long)
Public Sub moveFirst()
Public Sub moveLast()
Public Sub moveNext()
Public Sub movePrevious()
Public Function isBOF() As Boolean
Public Function isEOF() As Boolean
```

Berikut ini penjelasan dari masing – masing bagian dalam kelas Datalanggan:

Tabel IV-4 Variabel Anggota Kelas DataLanggan

Nama Variabel Anggota dan Tipenya	**Keterangan**
Nama : deftdiltgl Tipe : String	Digunakan untuk menyimpan nama-nama field dari tabel data pelanggan tunggal (DILTGL). Variabel ini akan dibaca dan diubah menjadi pemetaan nama-nama field dengan nomor urutnya dalam tabel DILTGL. Pemetaan ini akan berupa *dictionary*.
Nama : deftdilgan Tipe : String	Digunakan untuk menyimpan nama-nama field dari tabel data pelanggan ganda (DILGAN). Variabel ini akan dibaca dan diubah menjadi pemetaan nama-nama field dengan nomor urutnya dalam tabel DILGAN. Pemetaan ini akan berupa

	dictionary.
Nama : m_recordset() Tipe : String	Digunakan untuk menyimpan data tarif dasar listrik, dimana sebuah elemen array berisi sebuah baris data.
Nama : m_jmlrecord Tipe : Long	Digunakan untuk menyimpan jumlah baris data yang disimpan.
Nama : m_cursor Tipe : Long	Digunakan untuk menyimpan informasi baris data yang sekarang ditunjuk. Variabel ini berguna untuk penanda pada saat kursor penunjuk baris data digerakkan ke depan maupun ke belakang.
Nama : m_isDilTgl Tipe : Boolean	Digunakan untuk menyimpan apakah data yang disimpan adalah data pelanggan tunggal atau pelanggan ganda.
Nama : m_dictfld Tipe : Dictionary	Digunakan untuk memetakan nama-nama field dengan urutan nomor urut field tersebut.

Tabel IV-5 Fungsi Anggota Kelas DataLanggan

Nama Fungsi Anggota	**Keterangan**
Field(Index)	Merupakan *interface* kelas yang berupa *property* untuk mendapatkan nilai data

	baris yang sekarang ditunjuk pada kolom yang didefinisikan dengan Index.
Field(Index, value As String)	Merupakan *interface* kelas yang berupa *property* untuk menmperbarui nilai data baris yang sekarang ditunjuk pada kolom yang didefinisikan dengan Index dengan nilai baru value.
setData(rawstring As String)	Merupakan *interface* kelas yang digunakan untuk mengisi variabel m_recordset dengan data rawstring.
isDilTgl()	Merupakan *interface* kelas yang digunakan untuk mendapatkan informasi apakah data yang tersimpan bertipe tunggal atau ganda. Menembalikan TRUE jika tunggal dan FALSE jika ganda.
setDILstatus(isTunggal As Boolean)	Merupakan *interface* kelas yang digunakan untuk memberikan nilai TRUE jika data yang disimpan bertipe tunggal dan FALSE jika ganda.
getAllData(ByRef data As String)	Merupakan *interface* kelas yang digunakan untuk mendapatkan semua data yang

	disimpan. Data kembalian sudah dalam format yang telah ditentukan oleh *webservice* untuk pengiriman data.
getJumlahRecord(ByRef jumlah As Integer)	Digunakan untuk mendapatkan jumlah baris data yang disimpan.
moveFirst()	Digunakan untuk menggerakkan kursor ke baris data paling awal.
moveNext()	Digunakan untuk menggerakkan kursor ke baris data selanjutnya.
movePrevious()	Digunakan untuk menggerakkan kursor ke baris data sebelumnya.
moveLast()	Digunakan untuk menggerakkan kursor ke baris data paling akhir.
isBOF()	Digunakan untuk mendeteksi apakan kursor menunjukkan penanda awal data. Mengembalikan nilai TRUE jika benar dan FALSE jika tidak.
isEOF()	Digunakan untuk mendeteksi apakan kursor menunjukkan penanda akhir data. Mengembalikan nilai TRUE jika benar dan FALSE jika tidak.

b. Data Keluaran

Data keluran dari sistem ini adalah pesan SOAP yang didalamnya mengandung data pelanggan hasil penghitungan rekening dengan format yang telah ditentukan oleh sistem pendistribusian data.

Selain itu sistem akan mengirimkan pesan berisisi state mulai dan selesai kepada aplikasi pemonitor sistem yang memiliki format sebagai berikut:

<namaKomputer><state>

dimana <state> dapat berupa string "@start" untuk state mulai dan "@end" untuk state selesai.

4.2.3 Implementasi Data pada Aplikasi Pemonitor Sistem

Aplikasi ini hanya menerima data masukan dari mesin penghitung rekening melalui soket setiap kali ada perubahan state dalam proses penghitungan. Pesan ini berupa state yang sekarang dilakukan oleh mesin penghitung rekening, yakni state mulai dan state selesai. Adapun format dari pesan tersebut adalah:

<namaKomputer><state>

dimana <state> dapat berupa string "@start" untuk state mulai dan "@end" untuk state selesai. Jeda waktu antar "@start" dan "@end" akan dihitung sebagai waktu komputasi sedangkan jeda waktu antara "@end" dan "@start" dihitung sebgai *idle*. Contoh pesan "CLIENT-01@start", yang menunjukkan pesan state mulai yang dikirimkan oleh CLIENT-01.

4.3 Implementasi Proses

Berikut ini akan dijelaskan implementasi proses dari hasil perancangan proses yang telah dilakukan sebelumnya:

4.3.1 Kelas c*Server*

Kelas ini merupakan kelas yang digunakan untuk mengimplementasikan *Webserver* sebagai mesin yang menjalankan *webservice*. Kelas ini diimplementasikan di sisi *server* dengan menggunakan Python 2.3. Berikut ini abstraksinya:

```
class cServer(SOAPRequestHandler):
    def do_GET(self):
    def    send_Replay(self,    data,content_type='text/html',
cache=1):
```

Berikut ini penjelasan dari masing – masing bagian dalam kelas c*Server*:

Tabel IV-6 Fungsi Anggota Kelas c*Server*

Nama Fungsi Anggota	**Keterangan**
do_GET(self)	Fungsi untuk melayani *request* dari klien dengan metode GET
send_Replay(self, data,content_type='text/html', cache=1)	Fungsi untuk mengirimkan response ke klien.

4.3.2 Kelas ServiceProvider

Kelas ini merupakan kelas untuk mengimplementasikan penyediaan servis yang akan dipublish dalam *webservice*. Kelas ini diimplementasikan di sisi *server* dengan menggunakan Python 2.3. Berikut ini abstraksinya:

```
class ServiceProvider:
    def __init__(self):
        self.dataProvider
        self.dataProvider.start(connectionString)
    def getData(self,dataName):
    def sendData(self,dataName,theData):
```

Berikut ini penjelasan dari masing – masing bagian dalam kelas ServiceProvider:

Tabel IV-7 Variabel Anggota Kelas ServiceProvider

Nama Variabel Anggota dan Tipenya	**Keterangan**
Nama : dataProvider Tipe : obyek kelas dataProvider.	Digunakan untuk mengintansiasi kelas dataProvider.

Tabel IV-8 Fungsi Anggota Kelas ServiceProvider

Nama Fungsi Anggota	**Keterangan**
getData(self,dataName)	Fungsi untuk mendapatkan data dari SPD.

sendData(self,dataName,theData)	Fungsi untuk menyimpan data ke SPD.

4.3.3 Kelas DataProvider

Kelas ini merupakan kelas untuk mengimplementasikan penyediaan data yang akan dipublish dalam *webservice*. Kelas ini diimplementasikan di sisi *server* dengan menggunakan Python 2.3. Berikut ini abstraksinya:

```
class DataProvider:
    def start(self,strConn):
        self.conn
        self.rsDIL
        self.rsTDL
        self.fetchAmount
        self.DILWherestmt
        self.part
        self.strDIL
        self.tabName
        self.isDiltgl
    def requestTDL(self):
    def requestDIL(self):
    def buildMessage(self,data):
    def SendToOracle(self,tabname, data):
```

Berikut ini penjelasan dari masing – masing bagian dalam kelas DataProvider:

Tabel IV-9 Variabel Anggota Kelas DataProvider

Nama Variabel Anggota dan Tipenya	**Keterangan**
Nama : conn Tipe : String	Digunakan untuk menginstansiasi obyek *connection* dari cx_Oracle.
Nama : rsDIL	Digunakan untuk menginstansiasi obyek *cursor*

Tipe : String	dari cx_Oracle untuk pengambilan data pelanggan.
Nama : rsTDL Tipe : String	Digunakan untuk menginstansiasi obyek *cursor* dari cx_Oracle untuk pengambilan data tarif dasar listrik.
Nama : part Tipe : Numerik	Digunakan untuk menyimpan pengambilan data pelanggan yang ke-part.
Nama : fetchAmount Tipe : Numerik	Digunakan untuk mendefinsikan besarnya data pada setiap pengambilan ke SPD.
Nama : DILWherestmt Tipe : String	Digunakan untuk mendefinisikan syarat dari sintaks query pengambilan data ke SPD.
Nama : strDIL Tipe : String	Digunakan untuk menyimpan hasil dari pengambilan data pelanggan dari SPD.
Nama : tabName Tipe : String	Digunakan untuk menentukan data apakah yang sekarang ini akan dikirimkan.

Tabel IV-10 Fungai Anggota Kelas DataProvider

Nama Fungsi Anggota	**Keterangan**
requestTDL(self):	Fungsi untuk melayani servis permintaan data tarif dasar listrik.

def requestDIL(self):	Fungsi untuk melayani servis permintaan data pelanggan.
buildMessage(self,data):	Fungsi untuk membangun data kembalian sesuai dengan format yang telah ditentukan untuk membentuk skema data.
SendToOracle(self,tabname, data):	Fungsi untuk melayani servis penyimpanan data ke SPD.

4.3.4 Kelas orclthread

Kelas ini merupakan kelas yang digunakan untuk mengimplementasikan penyimpanan data ke Oracle10g *server* (SPD). Kelas ini diimplementasikan di sisi *server* dengan menggunakan Python 2.3. Berikut ini abstraksinya:

```
class orclthread(Thread):
    def __init__(self,strinsert,datainsert):
        Thread.__init__(self)
        self.insert
        self.data
    def run(self):
```

Berikut ini penjelasan dari masing – masing bagian dalam kelas orclthread:

Tabel IV-11 Variabel Anggota Kelas orclthread

Nama Variabel Anggota dan Tipenya	**Keterangan**
Nama : insert	Menyimpan string *query* penyimpanan data

Tipe : String	ke SPD.
Nama : data Tipe : String	Digunakan untuk menyimpan data yang akan disimpan ke SPD.

Tabel IV-12 Fungsi Anggota Kelas orclthread

Nama Fungsi Anggota	**Keterangan**
run(self)	Fungsi untuk mengimplementasikan *thread.*

4.3.5 Kelas Remote*Server*

Kelas ini merupakan kelas untuk mengimplementasikan pengaksesan fungsi – fungsi yang dipublish dalam *webservice*. Kelas ini diimplementasikan di sisi sistem penghitungan rekening dengan menggunakan Visual Basic 6.0. Berikut ini abstraksinya:

```
Private m_EnumService As EnumWSDLService
Private m_EnumPort As EnumWSDLPorts
Private m_EnumOperation As EnumWSDLOperations
Private m_EnumMapper As EnumSoapMappers

Public Function DumpWSDL(ByVal WSDLFileName As String) As
Boolean
Private Sub getPortbyName(name As String, ByRef thePort As
WSDLPort)
Private Sub getOperationbyName(name As String, ByRef
theOperation As WSDLOperation)
Public Function getData(dataName As String, ByRef result()
As String) As Boolean
Public Function sendData(params() As String, ByRef status As
Boolean) As Boolean
```

Berikut ini penjelasan dari masing – masing bagian dalam kelas Remote*Server*:

Tabel IV-13 Variabel Anggota Kelas Remote*Server*

Nama Variabel Anggota dan Tipenya	**Keterangan**
Nama : m_EnumService Tipe : EnumWSDLService	Digunakan untuk menyimpan semua elemen *Service* yang terdapat dalam dokumen WSDL yang dibaca.
Nama : m_EnumPort Tipe : EnumWSDLPorts	Digunakan untuk menyimpan semua elemen *Port* yang terdapat dalam dokumen WSDL yang dibaca.
Nama : m_EnumOperation Tipe : EnumWSDLOperations	Digunakan untuk menyimpan semua elemen *Operation* yang terdapat dalam dokumen WSDL yang dibaca.
Nama : m_EnumMapper Tipe : EnumSoapMappers	Digunakan untuk menyimpan semua elemen yang terdapat dalam elemen Message yang terdapat dalam dokumen WSDL yang dibaca.

Tabel IV-14 Fungsi Anggota Kelas Remote*Server*

Nama Fungsi Anggota	**Keterangan**
DumpWSDL(ByVal WSDLFileName As String) As Boolean	Fungsi untuk membaca dan meyimpan semua elemen WSDL.
getPortbyName(name As String, ByRef thePort As WSDLPort)	Fungsi untuk mendapatkan *Port* dengan nama yang didefinisikan.
getOperationbyName(name As String, ByRef WSDLOperation)	Fungsi untuk mendapatkan *Operation* dengan nama yang didefinisikan.
getData(dataName As String, ByRef result() As String) As Boolean	Fungsi untuk melakukan invokasi fungsi getData yang disediakan oleh *webservice*.
sendData(params() As String, ByRef status As Boolean) As Boolean	Fungsi untuk melakukan invokasi fungsi sendData yang disediakan oleh *webservice*.

4.3.6 Kelas Billing

Kelas ini merupakan kelas untuk mengimplementasikan proses penghitungan rekening. Kelas ini diimplementasikan di sisi sistem penghitungan rekening dengan menggunakan Visual Basic 6.0. Berikut ini abstraksinya:

```
Private Declare Function GetComputerName Lib "kernel32"
Alias "GetComputerNameA" (ByVal lpBuffer As String, nSize As
Long) As Long
Private Type RSLT_TGL_TYPE
Private Type RSLT_GDN_TYPE
Private Enum FIXTYPE_ENUM
Private Enum NORMTYPE_ENUM
Private m_oRemServ As cRemoteServer
Private m_oDIL As cDataLanggan
Private m_oTDL As cDataTarif
Private m_blthn As String
Private m_errlog As String
Private m_uri As String

Public Property Let wsdlURI(uri As String)
Public Sub RunBilling(blthn As String)
Private Sub doBilling()
Private Sub tunggalPecah()
Private Sub gandaPecah()
Private Sub tunggalNormal()
Private Sub gandaNormal()
Private Function FixingData(ByRef datanya As String, fixtype
As FIXTYPE_ENUM) As Double
Private Function normField(data As Double, normtype As
NORMTYPE_ENUM) As Double
Private Sub updateErrLog()
Private Sub sendSinyal(log As String)
```

Berikut ini penjelasan dari masing – masing bagian dalam kelas Billing:

Tabel IV-15 Variabel Anggota Kelas Billing

Nama Variabel Anggota dan Tipenya	**Keterangan**
Nama : RSLT_TGL_TYPE Tipe : Type	Digunakan untuk menyimpan hasil penghitungan rekening pelanggan tunggal untuk setiap rutin penghitungan.
Nama : RSLT_GDN_TYPE Tipe : Type	Digunakan untuk menyimpan hasil penghitungan rekening pelanggan ganda untuk setiap rutin penghitungan.

Nama : FIXTYPE_ENUM Tipe : Enum	Digunakan untuk mendefinisikan tipe pembetulan data.
Nama : NORMTYPE_ENUM Tipe : Enum	Digunakan untuk mendefinisikan tipe normalisasi data.
Nama : m_oRemServ Tipe : cRemote*Server*	Digunakan untuk menginstansiasi obyek dari kelas cRemote*Server*.
Nama : m_oDIL Tipe : cDataLanggan	Digunakan untuk menginstansiasi obyek dari kelas cDataLanggan.
Nama : m_blthn As Tipe : String	Digunakan untuk menyimpan bulan tahun operasional PT. PLN.
Nama m_errlog As Tipe : String	Digunakan untuk menyimpan pesan – pesan error.
Nama : m_uri As Tipe : String	Digunakan untuk menyimpan URI dimana servis disediakan oelh *webservice*.

Tabel IV-16 Fungsi Anggota Kelas Remote*Server*

Nama Fungsi Anggota	**Keterangan**
wsdlURI(uri As String)	*Interface* untuk mendefiniskan URI dimana *webservice* disediakan.

RunBilling(blthn As String)	Fungsi untuk menjalankan proses pengitungan rekening secara keseluruhan, termasuk proses permintaan dan pengiriman data dari dan ke *webservice*.
doBilling()	Fungsi untuk menjalakan proses penghitungan rekening.
tunggalPecah()	Fungsi penghitungan rekening dengan tipe tunggal pecah.
gandaPecah()	Fungsi penghitungan rekening dengan tipe ganda pecah.
tunggalNormal()	Fungsi penghitungan rekening dengan tipe tunggal normal.
gandaNormal()	Fungsi penghitungan rekening dengan tipe ganda normal.
FixingData(ByRef datanya As String, fixtype As FIXTYPE_ENUM) As Double	Funsi untuk memperbaiki data yang tidak benar akibat kesalahan pemasukan atau kesalahan pada proses – proses pengolahan data sebelumnya.
normField(data As Double, normtype As NORMTYPE_ENUM) As	Fungsi untuk pemotongan angka di belakang koma agas sesuai dengan panjang field tujuan.

Double	
updateErrLog()	Fungsi untuk menuliskan pesan error.
sendSinyal(log As String)	Fungsi untuk mengirim pesan state ke aplikasi pemonitor sistem.

4.3.7 Implementasi Proses Aktivasi *Webservice*

Proses aktivasi *webservice* dilakukan dengan menjalankan *Webserver* yang diimplementasikan dengan menginstansiasi kelas SOAP*Server* yang disediakan di dalam modul SOAPpy. Pemakaian kelas ini dimaksudkan untuk dapat menangani metode POST dan GET, dimana metode GET digunakan untuk menangani permintaan non-SOAP sedangkan metode POST digunakan untuk menangani permintaan berbasis SOAP. Kelas ini merupakan turunan dari kelas HTTP*Server* pada modul BaseHTTP*Server* yang merupakan modul preinstalasi Python2.3.

Untuk melayani permintaan non-SOAP, misalnya pada pembacaan WSDL yang diidentifikasikan dengan URI, maka perlu dilakukan *overiding* fungsi ***do_GET*** pada kelas turunan c*Server* yang abstraksinya dapat dilihat pada bagian 4.3.1. Sedangkan untuk melayani permintaan berbasis SOAP yang berarti pemanggilan fungsi secara remote oleh sistem penghitungan rekening, maka terlebih dahulu dilakkan pendaftaran fungsi pada obyek ***server*** melalui fungsi ***registerFunction,*** dimana obyek ***server*** yang merupakan hasil instansiasi kelas SOAP*Server*. Fungsi – fungsi yang didaftarkan ke *webservice* tersebut diletakkan

di dalam modul Provider di dalam kelas ServiceProvider yang abstraksinya dapat dilihat pada bagian 4.3.2.

```
BEGIN
    INIT server(IP/Hostname, Port)
    INIT ServiceProvider
    PUBLISH Fungsi getData
    PUBLISH Fungsi sendData
    RUN Webservice
EXCEPTION
    KEYSTROKE ^C
END
```

Dengan menginstansiasi obyek dari kelas ServiceProvider, konstruktorya akan langsung melakukan koneksi ke Oracle10g (SPD) dengan memamggil fungsi **start** pada obyek hasil instansiasi kelas DataProvider yang telah dilakukan sebelumnya.

4.3.8 Implementasi Proses Pendistribusian Data

Proses pendistribusian data baru akan dilaksanakan ketika ada permintaan data atau pemanggilan fungsi – fungsi yang disediakan dalam *webservice* oleh sistem penghitungan rekening. Untuk melakukan proses penghitungan rekening sistem penghitungan rekening harus mendapatakan data tarif dasar listrik dan data pelanggan ke *server* malalui pemanggilan fungsi **getData** dan fungsi **sendData** pada *webservice*.

Berikut ini dua fungsi yang merupakan potongan program dari kelas Remote*Server* yang digunakan untuk memanggil fungsi **getData** dan **sendData** yang disediakan *webservice*.

```
Public Function getData(dataName As String, ByRef result()
As String) As Boolean

    SET sconn ← New HttpConnector
```

```
    CALL getPortbyName("databillingPort", thePort)
    CALL getOperationbyName("getData", theOperation)
    CALL sconn.ConnectWSDL(thePort)

    CALL Serializer.Init(sconn.InputStream)
    CALL sconn.BeginMessageWSDL(theOperation)
    CALL Serializer.startEnvelope
    CALL Serializer.startBody
    CALL Serializer.startElement("getData", "")
    CALL Serializer.startElement("dataName", "")
    CALL Serializer.writeString(dataName)
    CALL Serializer.endElement
    CALL Serializer.endElement
    CALL Serializer.endBody
    CALL Serializer.endEnvelope
    CALL sconn.EndMessage
    CALL sReader.Load sconn.OutputStream
    EXTRACT Response Message
End Function
```

```
Public Function sendData(params() As String, ByRef status As
Boolean) As Boolean

    SET sconn ← New HttpConnector
    CALL getPortbyName("databillingPort", thePort)
    CALL getOperationbyName("sendData", theOperation)
    CALL sconn.ConnectWSDL thePort
    CALL status = False
    CALL Serializer.Init(sconn.InputStream)
    CALL sconn.BeginMessageWSDL(theOperation)
    CALL Serializer.startEnvelope
    CALL Serializer.startBody
    CALL Serializer.startElement("sendData", "")
    CALL Serializer.startElement("dataName")
    CALL Serializer.writeString(params(0))
    CALL Serializer.endElement
    CALL Serializer.startElement("theData")
    CALL Serializer.writeString(params(1))
    CALL Serializer.endElement
    CALL Serializer.endElement
    CALL Serializer.endBody
    CALL Serializer.endEnvelope
    CALL sconn.EndMessage
    CALL sReader.Load sconn.OutputStream
    EXTRACT Response Message
End Function
```

Sebelum dapat menjalankan fungsi – fungsi diatas, terlebih dahulu harus diketahui *interface* publik dari servis yang disediakan *webservice* melalui pembacaan WSDL dengan melakukan *request* via HTTP yang diidentifikasikan

dengan URI dimana dokumen WSDL tersbut disediakan. Berikut ini potongan program untuk membaca WSDL dan menyimpan properti – properti di dalamnya sebagai *interface* untuk pemanggilan fungsi. Berikut ini adalah *pseudo code* untuk mengiplementasikan proses pembacaan dokumen WSDL:

```
Public Function DumpWSDL(ByVal WSDLFileName As String) As
Boolean
    READ WSDLFile
    WHILE Port Masih Ada
        FETCH Port
        WHILE Operasi Masih Ada
            Ambil Semua Operasi
            WHILE Message Masih Ada
                Ambil Semua Message
            ENDWHILE
        ENDWHILE
    ENDWHILE
End Function
```

Dengan menjalankan fungsi **getData** maka pesan berbasis SOAP akan dikirimkan ke *server* tempat dimana servis yang dibutuhkan tersedia, dimana alamat *server* telah ditunjukkan dalam dokumen WSDL. Kemudian setelah sampai di *server* maka *server* akan mengidentifikasi adanya pesan berbasis SOAP yang isinya adalah pemanggilan fungsi **getData,** dan secara otomatis akan memanggil fungsi – fungsi yang ada sesuai dengan parameter yang dikirimkan yaitu parameter *dataName*. Jika parameter berisi "TDL" maka panggil fungsi requestTDL dan jika berisi "DIL" maka panggil fungsi requestDIL. Berikut ini adalah *pseudo code* untuk mengiplementasikan proses untuk melayani permintaan *client* baik untuk data Tarif maupun data Langgan:

```
def requestTDL(self):
    self.tabName ← 'tdl'
    CALL self.rsTDL.execute('select * from TDL')
    data ← self.buildMessage(self.rsTDL.fetchall())
    RETURN self.tabName,data
```

```
def requestDIL(self):
    self.part ← self.part+1
    IF self.strDIL=='' THEN
       CALL self.rsDIL.execute('select * from diltgl')
       self.tabName ← 'diltgl-'
       data ← self.rsDIL.fetchmany(self.fetchAmount)
       IF data==[]THEN
          IF self.isDiltgl==TRUE THEN
             self.part ← 1
             CALL self.rsDIL.execute('select * from _
                                      dilgan')
             self.isDiltgl ← FALSE
             data ← _
                  self.rsDIL.fetchmany(self.fetchAmount)
             self.strDIL ← self.buildMessage(data)
          ELSE
             self.part ← 0
             self.tabName ← 'None'
             self.strDIL ← '{-=EOF=-}'
          ENDIF
    ELSE
       self.strDIL ← self.buildMessage(data)
    ENDIF

    IF self.isDiltgl THEN
       self.tabName ← 'diltgl-'
    ELSE
       self.tabName ← 'dilgan-'
    ENDIF
    RETURN  self.tabName+str(self.part), self.strDIL
```

Masing – masing fungsi di atas akan mengembalikan data hasil query ke Oracle10g *server* (SPD) dalam bentuk string, dimana akan dilakukan proses pembuatan data kembalian dengan format sebagai berikut:

- Setiap *record* data dipisahkan dengan string "{-=record=-}".
- Setiap *field* data dipisahkan dengan karakter Tab ("\t").

Pembuatan data kembalian seperti ini dimaksudkan untuk memberikan kemudahan di sisi sistem penghitungan rekening untuk membangun sebuah skema

recordset sendiri dari data kembalian ini. Berikut ini adalah *pseudo code* untuk membangun data kembalian:

```
def buildMessage(self,data):
    def FilterData(x):
        IF x==None THEN RETURN ''
        RETURN str(x)
    Reccount ← str(len(data))
    strdata ← []
    FOR rec in data
        fieldcount ← str(len(rec))
        temp ← _
        "\t".join([FilterData(field) FOR field in rec])
        strdata ← strdata + temp
        retval = '{-=record=-}'JOIN(strdata)
    ENDFOR
    RETURN retval
```

Data – data kembalian ini kemudian akan dibugkus oleh SOAP dan dikirimkan ke sistem penghitungan rekening yang memintanaya. Poses – proses permintaan dan mendapatkan data yang diminta serta mengirimkan data hasil penghitungan ke *server* diimplementasikan pada fungsi **runBilling** yang dijelaskan pada bagian 4.3.9 di dalam bab ini.

Setelah data selesai diproses oleh sistem penghitungan rekening maka data tersebut akan dikirimkan kembali ke *server* dengan memanggil fungsi **sendData** seperti yang telah dijelaskan di atas pada sub bab yang sama. Fungsi **sendData** ini akan memanggil fungsi **SendToOracle** yang terdapat pada kelas DataProvider yang akan mengirimkan data tersebut ke SPD. Namun sebelum dikirimkan data yang dibungkus oleh pesan berbasis SOAP tersebut harus diekstraksi terlbih dahulu untuk mendapatkan data yang sebenarnya dan direkayasa sedemikian rupa untuk dapat dieksekusi melalui perintah query dengan menggunakan modul cx_Oracle.

Berikut ini adalah *pseudo code* untuk mengiplementasikan proses upload data ke SPD:

```
def SendToOracle(self,tabname, data):
    def ConvListToDict(list):
        i ← 0
        Mydict ← { }
        FOR col in list
            IF col=='' THEN
               mydict[str(i)] ← None
            ELSE
               mydict[str(i)] ← str(col)
               i ← i+1
            ENDIF
        ENDFOR
        RETURN mydict

        IF tabname=='diltgl' THEN
           tabdef ← open('tabdefdiltgl.txt','r').read()
           tabname ← 'DILTGL'
        ELSEIF tabname=='dilgan' THEN
           tabdef ← open('tabdefdilgan.txt','r').read()
           tabname ← 'DILGAN'
        ENDIF
        values ← ''
        FOR i in range(0,len(tabdef.split(',')))
            values ← values+':'+str(i)+','
        ENDFOR

        datanya ← _
          [str(rec).split('\t') _
            FOR rec in  data.split('\n')]
        datainsert ← _
          [ConvListToDict(rec) rec in datanya]
        strinsert ← _
          'INSERT INTO '+tabname+'2 (' + tabdef + _
          ') VALUES ('+values[0:len(values)-1]+')'

        OrclExp ← orclthread(strinsert,datainsert)
        CALL OrclExp.start()
```

Proses ekseskusi query pada ptotngan program di atas dilakukan dengan menciptakan *thread* agar tidak mengganggu proses pendistribusian data yang diminta oleh sistem penghitungan rekening. *Thread* itu sendiri dimplementasikan dengan kelas orclthread yang dapat dilihat pada bagian 4.3.4.

4.3.9 Implementasi Proses Penghitungan Rekening

Seperti yang telah dijelaskan sebelumnya bahwa sebelum melakukan proses penghitungan rekening terlbih dahulu sistem penghitungan rekening harus meminta data data tarif dasar listrik dan data pelanggan. Setelah data – data tersebut didapatkan maka proses penghitungan rekening siap untuk dilakukan.

Berikut ini adalah potongan program yang menunjukkan implementasi proses – proses permintaan data ke *server* dan proses penghitungan rekening yang tuliskan dalam fungsi **RunBilling**:

```
Public Sub RunBilling(blthn As String)
    m_blthn ← blthn
    Set m_oRemServ ← New cRemoteServer
    Set m_oDIL ← New cDataLanggan
    Set m_oTDL ← New cDataTarif

    dwLen ← 32
    strName ← String(dwLen, "X")
    CALL GetComputerName(strName, dwLen)
    strName ← Left(strName, dwLen)

    IF m_oRemServ.DumpWSDL(m_uri)==FALSE THEN eror

    IF m_oRemServ.getData("TDL", rawData)==FALSE THEN _
       eror
    CALL m_oTDL.setData (rawData(1))

    WHILE rawData(1) <> "{-=EOF=-}"
        IF m_oRemServ.getData("DIL",rawData)==FALSE THEN_
           eror
        m_oDIL.setDILstatus ← IIF(InStr(rawData(0),_
        "diltgl") > 0, TRUE, FALSE)
        CALL m_oDIL.setData (rawData(1))
        CALL sendSinyal(strName & "@-=start=-")
        CALL doBilling
        CALL sendSinyal(strName & "@-=end=-")
        rawData(0) ← _
          IIF(m_oDIL.isDilTgl, "diltgl", "dilgan")
        CALL m_oDIL.getAllData(rawData(1))

        IF NOT m_oRemServ.sendData(rawData,status) THEN_
          eror
    ENDWHILE
    CALL sendSinyal(strName & "@-=finish=-")
    EXIT
```

```
eror:
    VIEW pesan eror
End Sub
```

Dalam setiap rutin penghitungan rekening terdapat tiga tahapan proses yaitu tahan preproses meliputi permintaan data dan penyimpanan ke memori, termasuk didalamnya adalah mengekstraksi data yang diterima dari *server* dengan memperhatikan format data yang dikirimkan yang telah dijelaskan pada bagian 4.3.8 di bagian pembuatan data kembalian, serta tahapan penghitungan rekekning sendiri. Proses penghitungan rekening akan dilakukan secara terus – menerus hingga data pelanggan yang ada si SPD telah terproses seluruhnya. Jika semua data pelanggan telah terporoses maka *server* akan mengirimkan pesan "{-=EOF=-}" kepada sistem penghitungan rekening yang masih mencoba untuk meminta data.

Berikut ini adalah potongan program yang berupa fungsi **doBilling** yang mengimplementasikan proses penghitungan rekening setelah dilakukan tahapan preproses:

```
Private Sub doBilling()
    CALL m_oDIL.moveFirst
    i ← 1
    WHILE  NOT m_oDIL.isEOF
        IF isPecah(m_oDIL.Field("JNS_MUTASI"),_
           m_oDIL.Field("BLTH_MTSI")) THEN
            IF m_oDIL.isDilTgl THEN
                CALL tunggalPecah
            ELSE
                CALL gandaPecah
            ENDIF
        ELSE
            IF m_oDIL.isDilTgl THEN
                CALL tunggalNormal
            ELSE
                CALL gandaNormal
            ENDIF
        End If
```

```
        CALL m_oDIL.moveNext
        i ← i + 1
    ENDWHILE
End Sub
```

Rutin penghitungan dimulai dengan memproses data yang pertama dalam data pelanggan yang sekarang disimpan. Proses pertama yang dilalui adalah pemeriksaan pada data pelanggan, apakah data tersebut harus dihitung dengan tipe penghitungan normal atau pecahan (lihat bagian 3.3.3 untuk penjelasan secara lengkap). Kemudian proses dilanjutkan dengan pencarian tarif dasar listrik yang bersesuaian dengan data tarif dan daya yang ada pada data pelanggan. Berikut ini adalah ptotongan program untuk pencarian tarif dasar listrik yang bersesuaian dengan data tarif dan daya pada data pelanggan:

```
Public Function FindTDLFor(tarif As String, daya As String)
As Boolean
    IF Len(tarif) == 0 Or Len(daya) == 0 Then
        FindTDLFor ← False
        EXIT Function
    ENDIF
    daya ← _
         IIF(InStr(1, daya, "K", vbTextCompare), & _
          CDbl(Left(daya, Len(daya) - 1)) * 1000, daya)
    CALL moveFirst
    WHILE NOT isEOF
        IF Field("TARIF") == tarif THEN
            IF CDbl(Field("DAYAREAL")) >= CDbl(daya) THEN
                FindTDLFor ← True
                EXIT Function
            ENDIF
            CALL moveNext
        ELSE
            CALL moveNext
        ENDIF
   ENDWHILE
End Function
```

Apabila tidak diketemukan tarif yang bersesusaian maka proses akan menghentikan proses dan menulisakan log dan meneruskan ke data selanjutnya

dengan menggerakkan kursor ke *record* selanjutnaya. Apabila ditemukan tarif dasar listrik yang bersesuaian maka akan dilakukan lima proses utama , yaitu:

- Penghitungan rupiah beban.
- Penghitungan pemakaian daya listrik.
- Penghitungan rupiah pemakaian daya listrik.
- Penghitungan total tagihan.
- Pembaruan kolom – kolom pada data pelanggan dengan data hasil penghitungan.

Kelima proses diatas akan dijalankan dua kali dengan referensi yang berbeda di dalam satu fungsi apabila dalam pemeriksaan jenis penghitungan menghasilkan keputusan untuk menghitung secara pecahan.

Setelah proses ini selesai dijalankan maka data akan kembali disusun dengan format yang telah ditentukan oleh *server* yang telah dijelaskan pada bagian 4.3.8 di bagian pembuatan data kembalian sebelum dikirimkan ke *server* melalui fungsi **sendData** yang disediakan oleh *webservice*.

Keseluruhan proses pendistribusian data diatas dilakukan dengan menggunakan protokol SOAP yang berjalan pada protokol transport HTTP. Disisi *server* proses diimplementasikan dengan menggunakan modul SOAPpy sedangkan disisi sistem penghitungan rekening dengan menggunakan library MSSOAP.1.

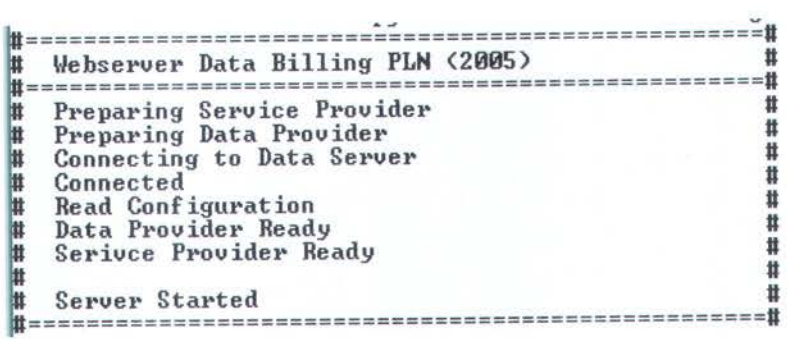
Setiap kali proses penghitungan rekening dimulai maka aplikasi akan mengirim pesan "@-=start=-" yang menandakan *state* mulai ke aplikasi pemonitor sistem, dan setelah penghitungan selesai aplikasi akan mengirimkan pesan "@-=end=-" yang menandakan *state* selesai. Pesan yang didapatkan oleh aplikasi pemonitor sistem ini akan diparsing dan kemudian dinterpretasikan ke dalam bentuk tampilan grafis.

4.4 Implementasi Antarmuka

Berikut ini akan dijelaskan implementasi antarmuka yang akan digunakan untuk membantu pengguna sistem untuk menjalankan aplikasi – aplikasi yang terlibat di dalam sistem.

4.4.1 Implementasi Antarmuka *Webserver*

Sesuai dengan perancangan yang telah dilakukan, antar muka untuk *Webserver* cukup dengan mode console yang dapat dipanggil melalui aplikasi pemonitor sistem untuk memudahkan pengoperasian. Berikut ini antarmuka dari *Webserver*:



Gambar IV-2 Antarmuka *Webservice*

4.4.2 Implementasi Antarmuka Sistem penghitungan rekening

Berikut ini implementasi antarmuka dari perancangan antarmuka sistem penghitungan rekening dengan spesifikasi sebagai berikut:

a. Tombol atau menu unutk menjalankan proses penghitungan rekening.

b. Tombol atau menu untuk menentukan dan menyimpan URI dari WSDL yang digunakan sebagai *interface* publik.

Gambar IV-3 Antarmuka Penghitungan Rekening

4.4.3 Implementasi Antarmuka Pemonitor Sistem.

Berikut ini adalah implementasi antarmuka pemonitor sistem dengan spesifikasi sebagai berikut:

a. Tombol atau menu untuk menjalankan dan menghentikan *Webserver*.

Gambar IV-4 Tombol Pengontrol Sistem

b. Tampilan pemantauan aktifitas penghitungan rekening yang dilakukan oleh sistem penghitungan rekening.

Client History

Proc. ID	STATE	START	END
CLIENT-01	State-7924	20:44:34	20:44:5
CLIENT-01	State-7925	20:44:55	20:45:1
CLIENT-01	State-7926	20:45:16	20:45:3
CLIENT-01	State-7927	20:45:37	20:45:5
CLIENT-01	State-7928	20:45:59	20:46:1
CLIENT-01	State-7929	20:46:20	

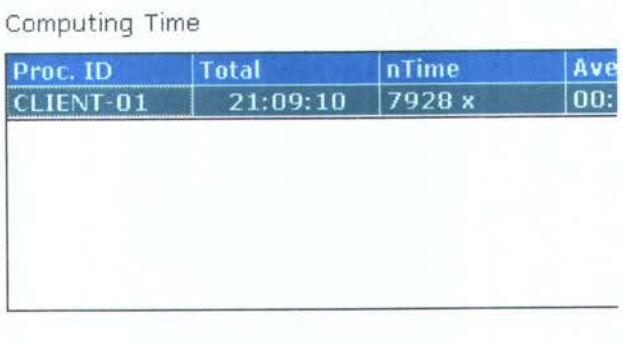
Gambar IV-5 Client History

Processor activity per 0,5 second

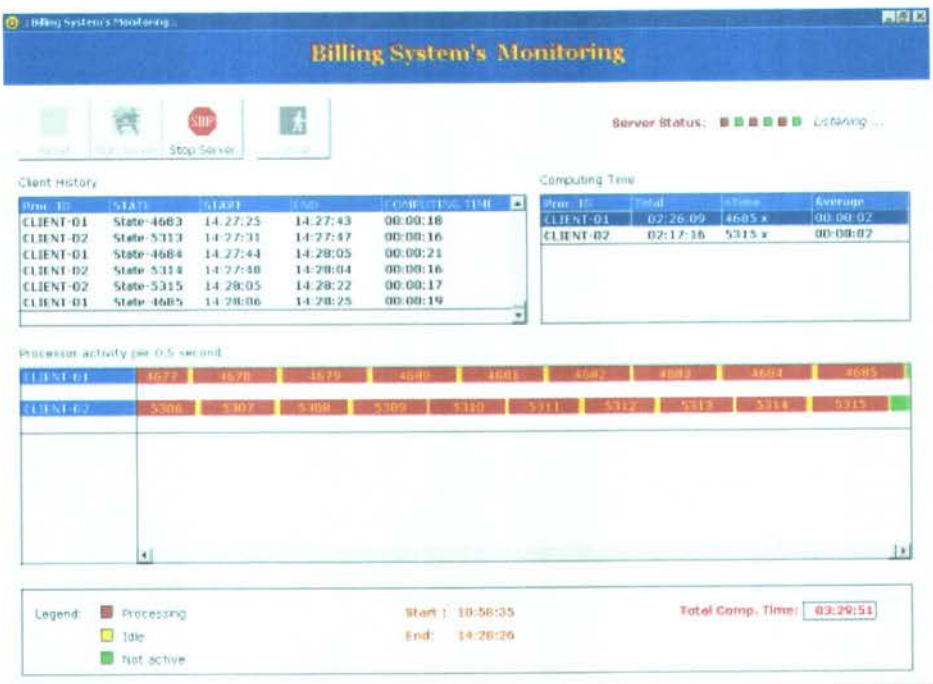
Gambar IV-6 Aktfitas Prosesor

c. Tampilan waktu komputasi total dan waktu komputasi tiap – tiap komputer sistem penghitungan rekening.

Gambar IV-7 Waktu Total



Gambar IV-8 Total Waktu Pneghitungan Tiap Mesin



Gambar IV-9 Antarmuka Aplikasi Pemonitor Sistem

BAB V

UJI COBA DAN EVALUASI

5.1 Skenario Uji Coba

Uji coba sistem penghitungan rekening pelanggan terdistribusi ini dilakukan dengan skenario uji sebagai berikut:

- Menetapkan besarnya data untuk setiap pengambilam data pelanggan ke SPD yaitu sebanyak 50 *record.*
- Adapun jumlah data yang diujikan dan jumlah prosesor yang dipakai adalah sebagai berikut:

Tabel V-1 Skenario Uji Coba

Percobaan ke-	Jumlah Data	Keterangan
1	1.000	Dilakukan pemrosesan data dengan menggunakan satu, dua, tiga, empat dan lima komputer penghitung rekening secara berurutan.Kemudian dicatat waktu komputasi untuk tiap-tiap pemakaian jumlah mesin penghitung yang berbeda.
2	10.000	
3	100.000	

5.2 Hasil Uji Coba

Tabel V-2 Tabel Hasil Uji Coba ke-1

Percobaan ke-1			
Jumlah Data: 1.000			
Jumlah Prosesor	**Waktu Komputasi**	**Satuan Detik**	**Percepatan**
1	6 Menit 36 Detik	396	1
2	3 Menit 14 Detik	194	2,041
3	2 Menit 16 Detik	136	2,911
4	1 Menit 47 Detik	107	3,700
5	1 Menit 34 Detik	94	4,212

Tabel V-3 Tabel Hasil Uji Coba ke-2

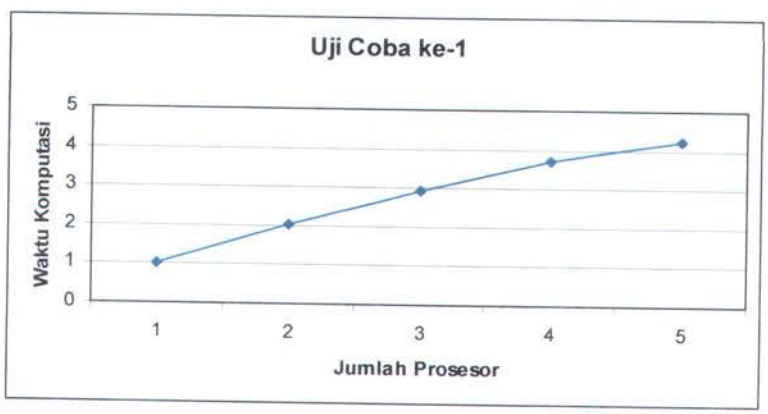
Percobaan ke-2			
Jumlah Data: 10.000			
Jumlah Prosesor	**Waktu Komputasi**	**Satuan Detik**	**Percepatan**
1	1 Jam 4 Menit 39 Detik	3879	1
2	30 Menit 49 Detik	1849	2,097
3	20 Menit 51 Detik	1251	3,100
4	16 Menit 14 Detik	974	3,982
5	13 Menit 7 Detik	874	4,438

Tabel V-4 Tabel Hasil Uji Coba ke-3

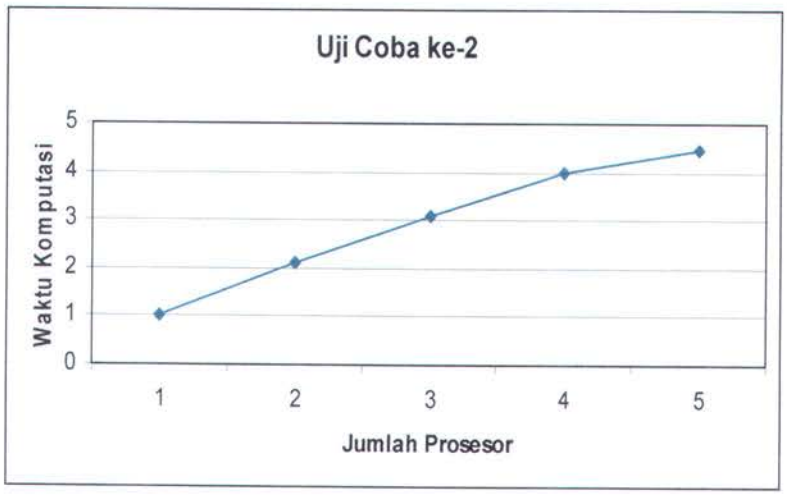
Percobaan ke-3			
Jumlah Data: 100.000			
Jumlah Prosesor	**Waktu Komputasi**	**Satuan Detik**	**Percepatan**
1	11 Jam 38 Menit 1 Detik	39219	1
2	5 Jam 19 Menit 59 Detik	19173	2.045
3	3 Jam 30 Menit 4 Detik	12823	3.058
4	2 Jam 39 Menit 50 Detik	9517	4.120
5	2 Jam 6 Menit 9 Detik	7444	5.268

5.3 Evaluasi

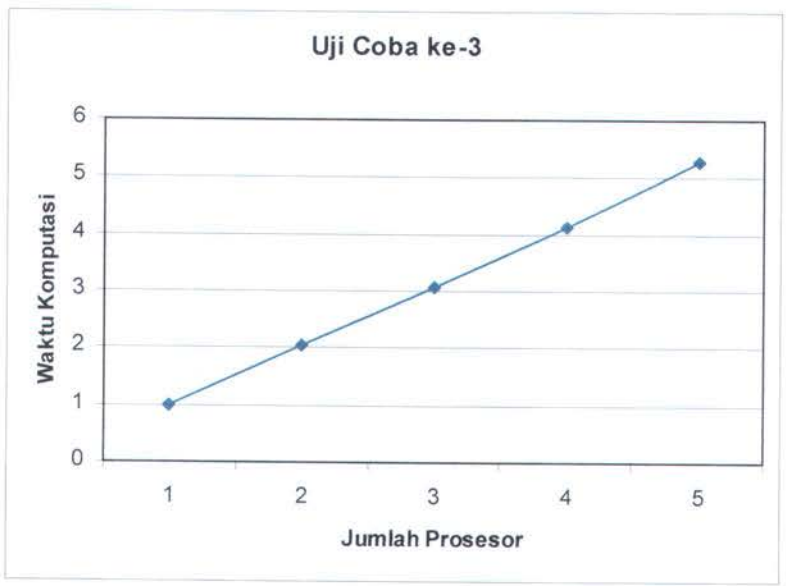
Berikut ini evaluasi dari uji coba yang telah dilakukan sesuai dengan skenario yang telah ditentukan. Untuk mempermudah evaluasi maka data percepatan proses pada tiap – tiap uji coba direpresentasikan ke dalam sebuah grafik.



Gambar V-1 Grafik Percepatan Uji Coba ke-1



Gambar V-2 Grafik Percepatan Uji Coba ke-2



Gambar V-3 Grafik Percepatan Uji Coba ke-3

Grafik – grafik di atas menunjukkan adanya percepatan sistem dalam menyelesaikan penghitungan rekening dengan melakukan penambahan mesin penghitung (prosesor).

BAB VI

KESIMPULAN DAN SARAN

Berikut ini kesimpulan yang dapat diambil dari hasil uji coba sistem yang telah dilakukan serta saran – saran untuk pengembangan selanjutnya.

6.1 Kesimpulan

Dari hasil pengujian dan evaluasi, sistem penghitungan rekening pelanggan yang terdistribusi dengan menggunakan teknologi *Webservice* mampu menunjukan adanya percepatan dalam menyelesaikan penghitungan rekening dengan penambahan mesin penghitung rekening.

Unjuk kerja sistem ini bisa ditingkatkan lagi dengan meningkatkan spesifikasi tiap – tiap mesin yang terlibat dalam sistem penghitungan rekening terdistribusi ini, serta memperbaiki kembali algoritma yang digunakan dalam program.

6.2 Saran

a. Perbaikan kembali Algoritma – algoritma yang digunakan dalam sistem penghitungan rekening, terutama pada teknik perulangan atau pada teknik pencarian data.
b. Penelitian lebih lanjut mengenai keamanan pertukaran data pada *webservice*, sehingga sistem ini dapat diterapkan pada area jaringan komputer yang lebih luas (WAN, MAN, Internet).

DAFTAR PUSTAKA

1. [W3C04] Web Services Architecture Working Group, *Web Services Architecture*, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, W3C Web Services Activity, 2004.

2. [ETH02] Cerami Ethan, Distributed Application with XML-RPC, SOAP, UDDI & WSDL, O'Reilly, 2002.

3. [IBM05] IBM DeveloperWorks, *Standards and Web services*, http://www-128.ibm.com/developerworks/webservices/standards/, IBM Press developerWorks, 2005.

4. [WKW05] Wikipedia, *Web services description language*, http://en.wikipedia.org/wiki/WSDL, The Free Encyclopedia Wikipedia, 2005.

5. [WKS05] Wikipedia, *Simple Object Access Protocol*, http://en.wikipedia.org/wiki/SOAP, The Free Encyclopedia Wikipedia, 2005.

6. [TAN02] Tanenbaum,AS dan Steen,M, *Dystributed System Principle and Paradigm,* Prentice Hall, 2002.

7. [ROS04] Rosenberg,s and Remy,D, Securing Web Services with WS-Security, SAMS, 2004.