

Web Service Sebagai Penyedia Layanan Pada Sistem Informasi Manajemen Jurusan

Wahyu Eko Sulistiono¹, Mardiana¹, Nasrul Fatoni²

Dosen Jurusan Teknik Elektro Universitas Lampung
Jl. S. Brojonegoro No. 1 Gedung Meneng Bandar Lampung 35145
eko@unila.ac.id, mardiana@unila.ac.id

Abstrak – Teknologi *Web Services* yang mulai berkembang saat ini menghadirkan berbagai macam percepatan teknologi informasi di berbagai bidang. Salah satu arsitektur terbaru yang menggunakannya adalah *Service Oriented Architecture (SOA)*. SOA merupakan sebuah pendekatan dalam merancang (arsitek) suatu aplikasi dengan menggunakan kembali (*reuse*) komponen-komponen yang sudah ada sebelumnya. Komponen-komponen tersebut memberikan suatu jenis layanan bisnis (*service*) dan dibangun dan berinteraksi satu sama lain secara bebas dan lepas (*loose coupled*).

Pada penelitian ini, dibuat suatu aplikasi *web* yang menggunakan *web service* sebagai penyedia layanan dengan mengimplementasikan SOA. Aplikasi *web* yang dibuat sebagai studi kasus adalah Sistem Informasi Manajemen Jurusan (Simanjur) yang bertujuan untuk mempermudah proses pencarian dan pencatatan data program studi di Universitas Lampung. Simanjur dibuat dengan menggunakan bahasa pemrograman server side PHP dan metode proses RUP (*Rational Unified Process*), telah diujicoba pada platform Java dan .NET. Hasil pengujian aplikasi Simanjur menunjukkan bahwa penggunaan SOA dalam aplikasi *web* memiliki berbagai keuntungan. Keuntungan yang diperoleh adalah terciptanya sebuah aplikasi *web* yang dapat menyediakan *service* bagi aplikasi lain, sehingga data yang dihasilkan dapat dikonsumsi oleh aplikasi lain walau berbeda *platform* atau *framework*.

Kata Kunci : *web services*, SOA, WSDL, SOAP, XML, Sistem Informasi Manajemen Jurusan.

Abstract – *Web Services technology that currently is developing presents a variety of information technology acceleration in many fields. One of new architecture using it is Service Oriented Architecture (SOA). SOA is an approach in architecting an application by reusing existing components. These components provide a business*

service type and are built and interacted one to each other independently and loose coupled.

In this research, a web application is developed using web service as service provider by implementing SOA. The case study is Departmental Management Information System (Simanjur) that aims to ease searching and recording of study program data in University of Lampung. Simanjur is built using PHP as server side programming language and RUP (Rational Unified Process), as process method, and is tested on Java and .NET platform. The result of testing shows that SOA usage in this application gives some advantages. The advantages is a web application that provides service to other application, therefore data owned can be consumed by other application even from different platform or framework.

Keywords: *web services*, SOA, WSDL, SOAP, XML, Departmental Information System

A. Pendahuluan

Teknologi *web service* menawarkan kemudahan menjembatani pulau-pulau informasi tanpa mempermasalahkan perbedaan teknologi yang digunakan oleh masing-masing sumber [1]. Tidak hanya informasi, *web services* juga dapat dikembangkan ke arah penggunaan aplikasi. Misalnya suatu aplikasi memerlukan perhitungan kurs mata uang dari rupiah ke dollar, aplikasi tersebut dapat mengambil informasinya dari *web* yang menyediakan layanan perhitungan kurs tersebut tanpa membuat aplikasi perhitungan sendiri dengan cara mengirimkan informasi jumlah rupiah yang ingin dihitung, selanjutnya aplikasi itu akan menerima hasil perhitungan tersebut. *Web services* merupakan ‘*big thing*’ berikutnya dari pengembangan software. Ia berupa aplikasi modular yang dapat dipublikasikan, diletakkan, dan

Naskah ini diterima pada tanggal 8 Juni 2009, direvisi pada tanggal 15 Juli 2009 dan disetujui untuk diterbitkan pada tanggal 1 Agustus 2009

dieksekusi antar *web*. *Web services* dapat diibaratkan sebagai pustaka yang menyediakan berbagai macam informasi dan layanan ke aplikasi lain antar *web*. Keuntungan menggunakan teknologi *web services* adalah informasi yang disediakan oleh aplikasi dapat digunakan kembali oleh aplikasi lain yang berbeda *platform* tanpa harus membuat aplikasi baru, lebih efisien dalam mengembangkan aplikasi yang lebih kompleks karena aplikasi dapat dipecah menjadi sub aplikasi yang lebih kecil secara independen sehingga aplikasi akan lebih mudah dikembangkan, lebih menghemat biaya pengembangan aplikasi. *Service Oriented Architecture* (SOA) adalah sebuah metode penyajian informasi berbasiskan *service* yang informasinya dapat diakses oleh berbagai aplikasi atau *client* meskipun berbeda *platform* maupun *framework*. Salah satu cara untuk mengimplementasikan SOA ini adalah menggunakan *web services*.

Informasi data jurusan merupakan suatu hal yang sangat penting dalam suatu universitas. Bukan hanya berfungsi untuk menilai kinerja jurusan dari waktu ke waktu tetapi juga dapat dijadikan sebagai bahan acuan untuk akreditasi suatu jurusan. Untuk itu perlu dibuat suatu software aplikasi yang dapat mencatat, mengatur dan menyediakan informasi data jurusan secara terpusat dan dapat diakses dengan mudah oleh pihak-pihak yang membutuhkan di Universitas Lampung. Selain itu software aplikasi tersebut harus dapat menyediakan *service* untuk aplikasi lain agar lebih mudah untuk dikembangkan dan data yang dihasilkan mudah dikonsumsi oleh aplikasi lain walau berbeda *platform* atau *framework*. Aplikasi *web* Sistem Informasi Manajemen Jurusan (Simanjur) yang berbasiskan SOA adalah solusi yang tepat untuk masalah tersebut. Melalui Simanjur ini diharapkan data jurusan akan mudah diakses oleh pihak-

pihak yang memerlukannya secara efektif, efisien dan tepat waktu.

B. Tinjauan Pustaka

Web Services

Web services adalah sebuah teknik pemrograman di mana sebuah *service* menggunakan standar-standar berbasis XML dalam menjelaskan interface dan protocol yang harus digunakan untuk memanggil *service* tersebut. Standar-standar tersebut adalah [3]:

a. SOAP (*Simple Object Access Protocol*)

Menjelaskan protokol tentang bagaimana sebuah *web service* dapat dipanggil.

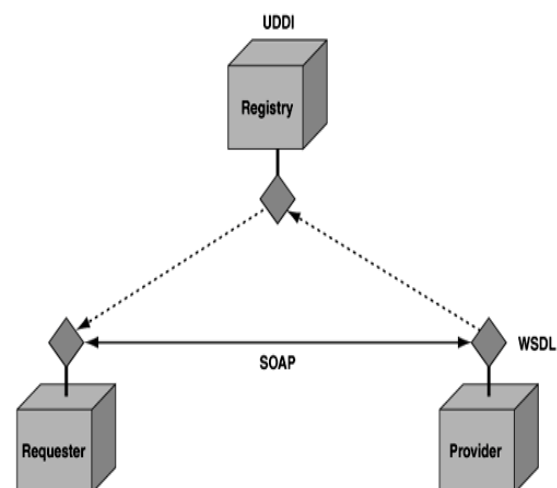
b. WSDL (*Web Services Definition Language*):

Sebuah format XML yang menjelaskan *interface* dari sebuah *web service* (parameter *input* dan *output*).

c. UDDI (*Universal Description, Discovery and Integration*):

Sebuah direktori yang berisi daftar *web service* yang dapat ditemukan dan dipanggil oleh aplikasi lainnya.

Dengan standar di atas, *web services* sangat mendukung implementasi SOA dimana karakteristik SOA seperti *loose coupling* dan *service interface* disediakan oleh teknologi *web services*.



Gambar 1. Komponen-komponen *web service* [3]

Simple Object Access Protocol (SOAP)

SOAP merupakan suatu protokol berbasis XML yang digunakan untuk kebutuhan pertukaran informasi dalam suatu sistem terdistribusi dan terdesentralisasi, seperti halnya IIOP (pada CORBA), ORCP (pada DCOM) dan JRMP (pada RMI). Berbeda dengan RMI, CORBA dan DCOM, SOAP merupakan protokol yang bersifat *independent* terhadap *platform*, model pemrograman, dan *transport* protokol yang digunakan dalam proses pertukaran pesan SOAP. Pesan SOAP dapat dikirimkan melalui HTTP, SMTP maupun FTP.

Pesan SOAP berbentuk sekumpulan XML *Schema* yang mendefinisikan format untuk mentransmisikan pesan XML melalui jaringan, termasuk tipe data dan cara menstrukturkan pesan secara tepat sehingga dapat mudah dipahami oleh server atau *end-point* lainnya.

Pesan SOAP terdiri dari 3 bagian, yaitu :

- a. **Envelope**, suatu kerangka yang mendefinisikan apa yang ada dalam pesan dan bagaimana pesan harus diproses serta menunjukkan *resipien* dari *message* tersebut.
- b. **Header**, berisi informasi yang berhubungan dengan keamanan dan *routing*. Keberadaan *Header* dalam SOAP bersifat *optional*.
- c. **Body**, berisi data yang berhubungan dengan aplikasi tertentu yang sedang dipertukarkan. Data di-*'mark-up'* sebagai XML dan dimasukkan dalam format yang spesifik yang didefinisikan dalam XML *Schema*.

Standard SOAP yang dikembangkan oleh W3C versi terakhir adalah SOAP versi 1.2 yang ditetapkan pada 24 Juni 2003.

Web Service Describe Language (WSDL)

WSDL merupakan bahasa standar yang menyediakan mekanisme untuk mendeskripsikan *Service* yang disediakan oleh sistem (*Web service*), lokasi keberadaan *service* tersebut dan bagaimana cara memperolehnya, secara terstruktur

dalam format XML. WSDL dapat dianalogikan sebagai IDL (*interface definition language*) dalam CORBA dan COM. *Service* dideskripsikan sebagai koleksi dari *entry-point* atau *port* komunikasi. WSDL mendeskripsikan *service* dengan menggunakan elemen sebagai berikut :

- a. **Type**, tipe data yang digunakan sebagai *argument* dan *return type*.
- b. **Message**, merepresentasikan definisi data yang ditransmisikan.
- c. **Port type**, sekumpulan operasi yang didukung oleh satu atau lebih *end-point*.
- d. **Binding**, mendefinisikan protokol dan format pertukaran data untuk operasi yang didefinisikan oleh *Port Type*.
- e. **Port**, menspesifikasikan *end-point* yang digunakan untuk *binding*.
- f. **Service**, koleksi *end-point* yang berkaitan yang disediakan oleh *web service*.
- g. **Operation**, mendefinisikan kemampuan yang didukung oleh *service* tertentu.

Universal Description, Discovery & Integration (UDDI)

UDDI merupakan sekumpulan spesifikasi yang menunjukkan *registry* informasi mengenai *web service*. UDDI menyediakan mekanisme untuk mempublikasikan informasi mengenai bisnis dan *service* pada satu lokasi (*repository*) yang dikelola secara terpusat dan melakukan *query* mengenai informasi tersebut secara dinamis dan programatis. Direktori pada UDDI bertindak seperti '*Yellow Pages*' dimana *service* dikategorikan sesuai tujuan utamanya. *Direktory* UDDI terdiri dari 3 bagian, yaitu:

- a. **White pages**, menyediakan informasi rinci mengenai organisasi yang menawarkan *service*
- b. **Yellow pages**, mencakup pengakategorian jenis industri berdasarkan standar *taxonomi* industri

- c. **Green pages**, mendeskripsikan *interface* dan kebutuhan untuk memperoleh *service*, seperti *return type*.

UDDI merupakan file XML Schema yang mendefinisikan struktur data mengenai karakteristik bisnis dan *service*. Deskripsi *service* didefinisikan menggunakan *Dokumen Type Model* (tModel). Secara umum UDDI berisi informasi mengenai siapa yang menyediakan *service* (*businessEntity*), *Service* apa yang disediakan (*businessService*), di mana lokasi *service* tersedia (*bindingTemplate*), referensi mengenai informasi bagaimana *service* tersebut diperoleh (tModel).

SOA (Service Oriented Architecture)

SOA adalah sebuah paradigma untuk menyatukan dan menggunakan kemampuan terdistribusi yang mungkin dikontrol oleh domain yang berbeda [2].

SOA adalah sebuah pendekatan dalam merancang (arsitek) suatu aplikasi dengan menggunakan kembali (*reuse*) komponen-komponen yang sudah ada sebelumnya. Dalam hal ini, komponen-komponen tersebut memberikan suatu jenis layanan bisnis (*service*) tertentu seperti antara lain: melihat data program studi, mencari data mahasiswa, mencari data dosen, dan lain sebagainya.

Karakteristik SOA

- a. SOA adalah suatu cara perancangan aplikasi dengan menggunakan komponen-komponen atau pelayanan yang sudah ada. Dengan kata lain, suatu aplikasi dibangun secara modular. Sebenarnya pendekatan modular ini bukanlah sesuatu yang baru. Teknik - teknik pemrograman masa kini seperti *object oriented programming*, telah mengedepankan pendekatan modular dalam pembangunan aplikasi. Namun yang membuat SOA berbeda adalah komponen atau *service* tersebut dibangun dan berinteraksi satu sama

lain secara bebas dan lepas (*loose coupled*). Dengan bersifat *loose coupled*, sebuah *service* dapat dipanggil oleh program/*service* lainnya tanpa program pemanggil tersebut perlu memperhatikan dimana lokasi *service* yang dipanggil berada dan *platform/teknologi* apa yang digunakan oleh *service* tersebut. *Loose coupling* sangat penting bagi SOA karena dengan demikian pemanggilan sebuah *service* oleh *service* lainnya dapat dilakukan pada saat *run-time*.

- b. Karakteristik lainnya adalah *service* dalam SOA disusun atas 2 hal: *Service Interface* dan *Service Implementation*. *Service Interface* menyatakan bagaimana *service* tersebut dapat dipanggil seperti parameter *input/output* dan lokasi ia berada. Misalkan, *service interface* untuk pencarian data mahasiswa menyatakan berbagai cara untuk mendapatkan informasi tentang seorang mahasiswa (dari npm atau nama dan sebagainya) dan struktur data mahasiswa yang dikembalikan. *Service Implementation* adalah bagaimana *logic* dari *service* pencarian data mahasiswa tersebut dijalankan. *Service implementation* sangat terkait dengan teknologi pemrograman yang digunakan. SOA tidak perlu memperdulikan bagaimana sebuah *service* diimplementasikan. Entah ditulis dengan bahasa Java, .NET atau PHP, yang penting adalah bagaimana *service* tersebut dapat dipanggil dan memberikan informasi sesuai dengan *Service Interface*-nya.
- c. Karakteristik SOA yang terakhir adalah *service* tersebut harus *business oriented*. Dalam arti, setiap *service* yang didefinisikan harus melakukan suatu aktifitas bisnis tertentu, misalkan mencari data mahasiswa, melihat data program studi, mencari data dosen, dan sebagainya.

C. Metode Penelitian

Aplikasi web dalam penelitian ini dibuat dengan menggunakan bahasa pemrograman server side PHP dan metode proses RUP (*Rational Unified Process*). Untuk pemodelan aplikasi digunakan *Unified Modeling Language* (UML). Sebagai dasar pembuatan aplikasi, mengacu pada requirement dan fungsi fungsi software dari DIKTI (Direktorat Jendral Pendidikan Tinggi) versi 6.0.4 serta bentuk laporan dari *website* DIKTI (<http://www.evaluasi.or.id>). Beberapa fungsi tambahan untuk kebutuhan internal Universitas Lampung diperoleh dari masukan pengguna yang dikumpulkan melalui survey, wawancara dan workshop terkait manajemen jurusan. Aplikasi web kemudian diujicoba pada platform Java dan .NET.

D. Hasil Pembahasan

Aplikasi Simanjur adalah suatu sistem aplikasi berbasis *web* yang digunakan untuk menyimpan, mengatur dan menyediakan data program studi. Batasan yang dapat dilakukan sistem meliputi input data program studi, penyajian data program studi dalam bentuk laporan yang diakses melalui halaman *web*, ada fitur *web service* sebagai implementasi SOA yang dapat menyediakan *service* bagi aplikasi lain.

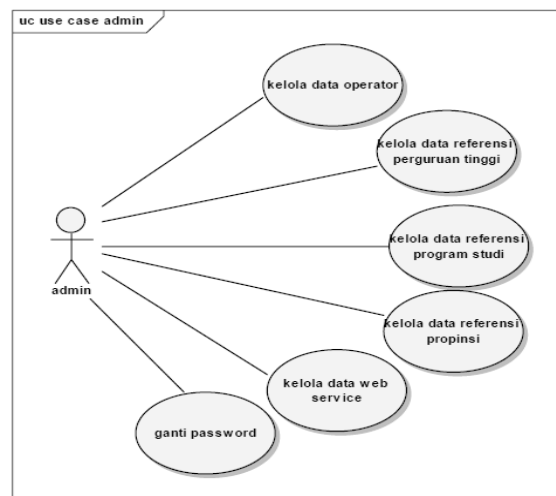
1. Analisis Sistem

Ada 3 tipe aktor dalam sistem Simanjur ini yaitu :

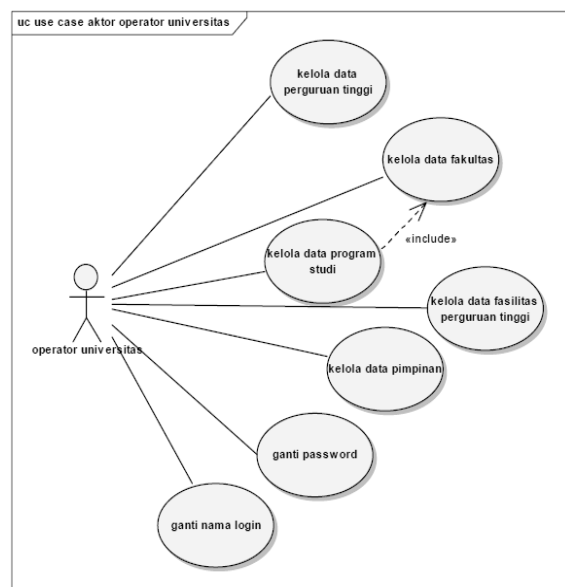
- a. Pihak pengelola
 - i. Admin, yaitu *user* yang mengatur manajemen operator melalui halaman admin dan memantau sistem secara keseluruhan.
 - ii. Operator yang terdiri atas operator universitas dan operator program studi. Operator universitas mengatur manajemen data universitas, sedangkan operator program studi mengatur manajemen data universitas

- b. Pengguna umum, yaitu *user* atau pengunjung yang hanya dapat menggunakan fitur-fitur umum sistem.
- c. Pengguna *web service*, yaitu *user* yang menggunakan fitur *web service*.

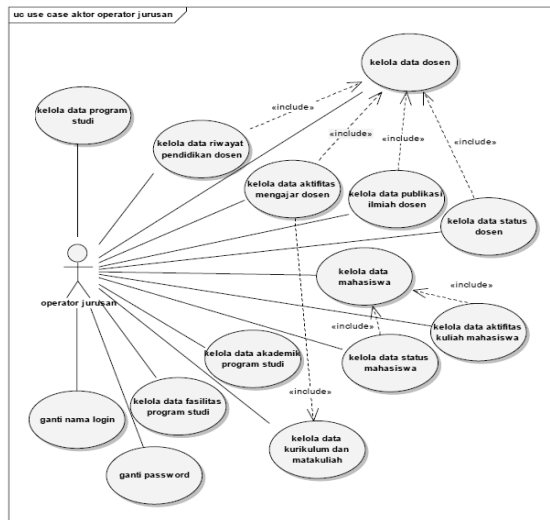
Pihak pengelola harus login terlebih dahulu untuk dapat mengakses fitur-fitur pengelola, sedangkan pengguna umum dapat langsung menggunakan fitur-fitur umum termasuk fitur *web service* tanpa harus login. Untuk fitur *web service* disediakan halaman untuk melihat daftar *web service* beserta *methode-methode* yang dapat digunakan.



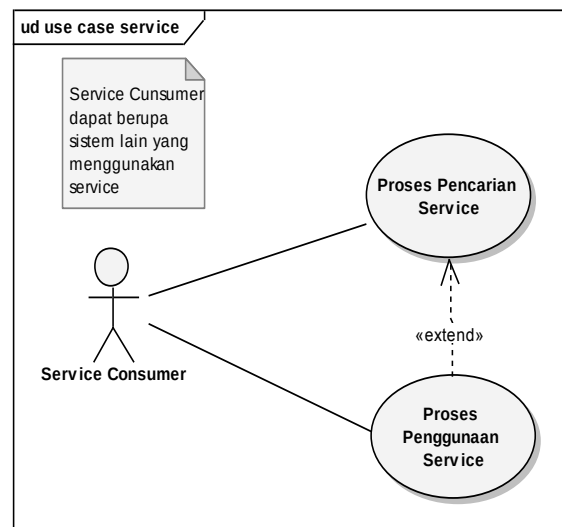
Gambar 2. Diagram *use case* aktor admin



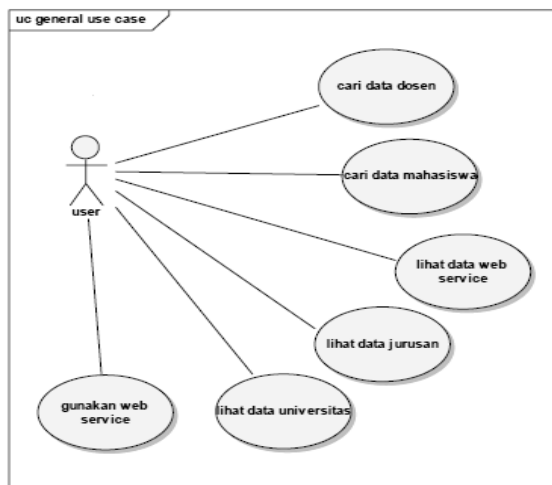
Gambar 3. Diagram *use case* aktor operator universitas



Gambar 4. Diagram use case aktor operator jurusan



Gambar 6. Diagram use case untuk aktor yang menggunakan web service



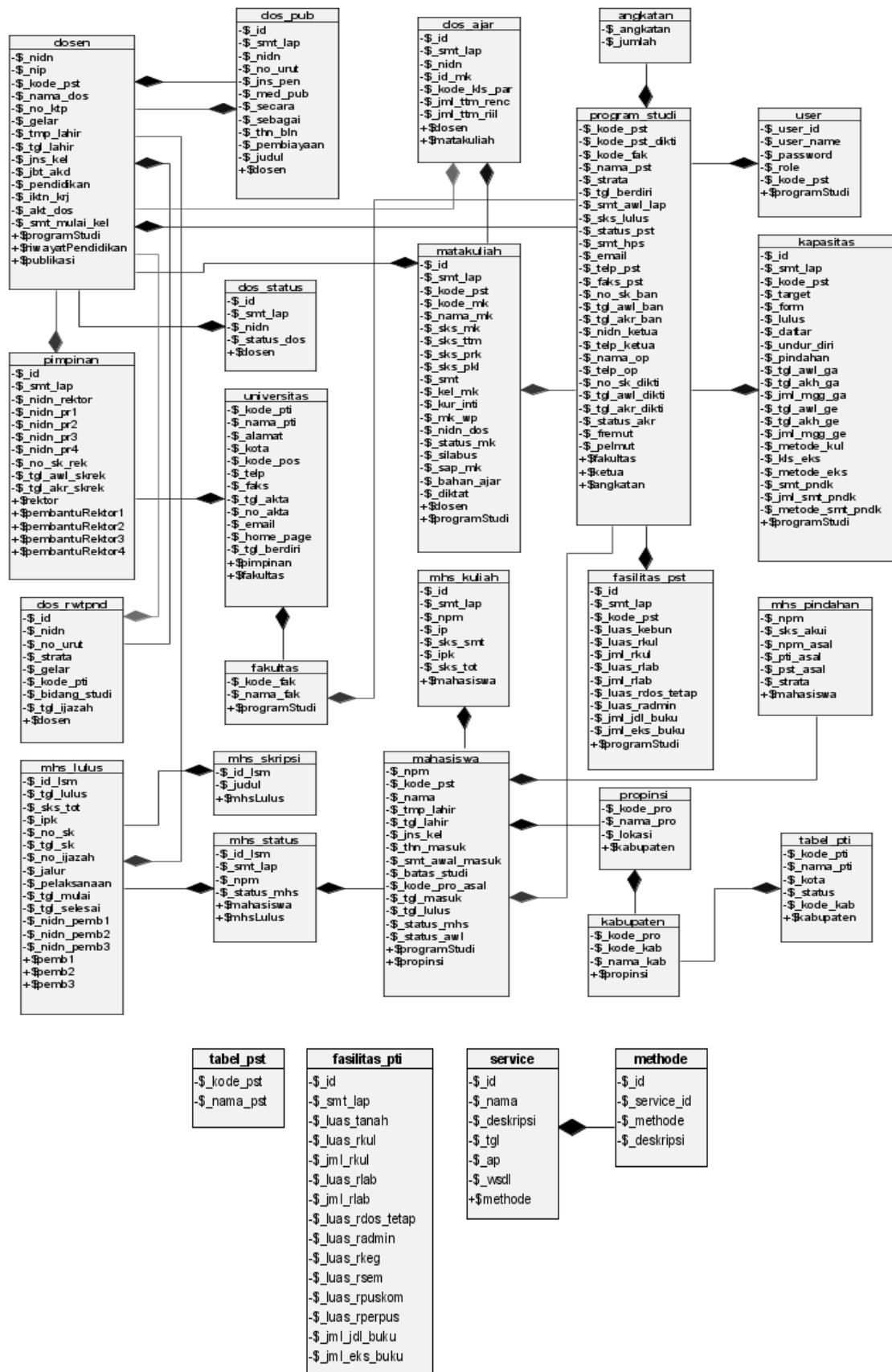
Gambar 5. Diagram use case general user

Skenario Use case

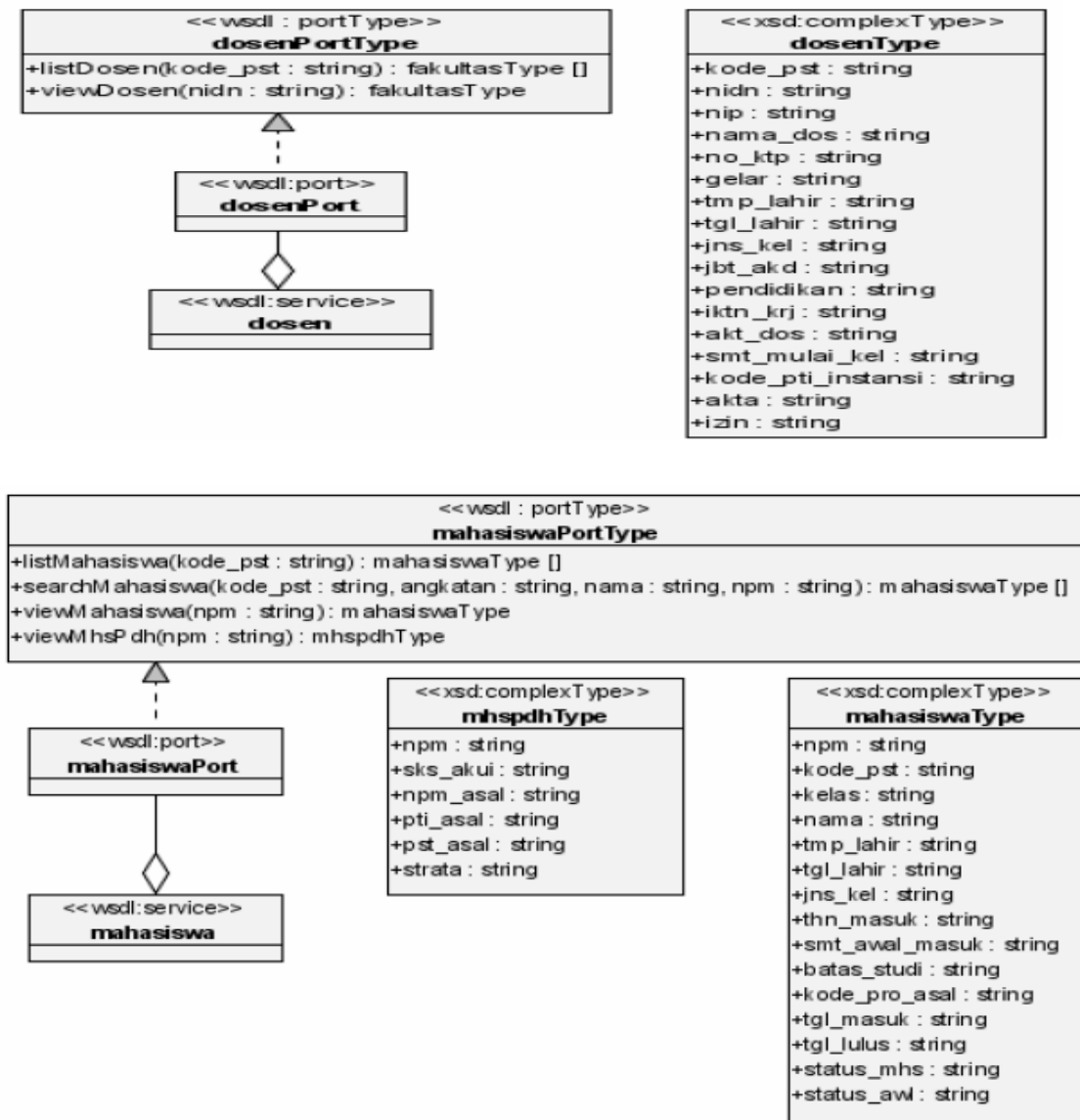
Skenario use case berisi gambaran use case secara detail yang meliputi nama use case, deskripsi use case, aktor, alur kejadian dan hubungan antar use case satu dengan lainnya. Karena keterbatasan halaman skenario tidak dapat ditampilkan semua. Contoh skenario use case kelola perguruan tinggi

Nama Use Case	:	Kelola Data Referensi Perguruan Tinggi
Deskripsi	:	Fungsi yang terdiri dari proses membuat/menambah, mengubah, menghapus data referensi perguruan tinggi
Aktor	:	Admin
Pemicu	:	Aktor memilih menu kelola data referensi perguruan tinggi pada halaman menu utama aktor
Kondisi Sebelumnya	:	Aktor berada di halaman menu utama aktor
Alur kejadian	:	1. Aktor memilih menu kelola data referensi perguruan tinggi dari halaman menu utama aktor 2. Sistem menampilkan daftar perguruan tinggi yang telah ada (dalam bentuk tabel) dan menu <i>form</i> (tambah, ubah, hapus) pada tiap-tiap perguruan tinggi
Pengecualian	:	-
Kondisi sesudahnya	:	• Aktor berada di halaman daftar perguruan tinggi

Gambar 7. Skenario use case kelola perguruan tinggi



Gambar 8. Diagram kelas Simanjur



Gambar 9. Diagram WSDL untuk web service dosen dan mahasiswa

2. Desain Sistem

Diagram Kelas

Sistem Simanjur memiliki 27 kelas entitas, dengan jenis relasi kelas yang digunakan adalah komposisi, Artinya kelas-kelas yang memiliki relasi komposisi tersebut memiliki ketergantungan yang erat dengan kelas lain yang menjadi tujuan relasinya. Contohnya bila pada kelas fakultas objek fakultas dihapus, otomatis akan mempengaruhi kelas program_studi, objek program_studi akan ikut terhapus.

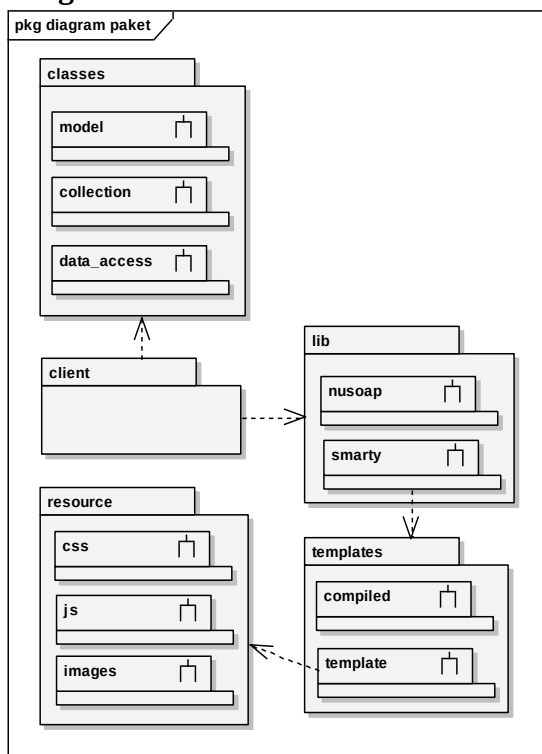
Desain WSDL

Desain WSDL ini menggambarkan desain *service interface* yang menyatakan bagaimana *service* tersebut dapat dipanggil seperti parameter *input* dan struktur data yang dikembalikan. Diagram WSDL yang dihasilkan untuk *web service* : universitas, fakultas, program studi, dosen, riwayat pendidikan dosen, aktifitas mengajar dosen, publikasi ilmiah dosen, status dosen, mahasiswa, aktifitas kuliah, mahasiswa, status mahasiswa, akademik program studi, kurikulum dan matakuliah,

fasilitas program studi, fasilitas perguruan tinggi, propinsi, kabupaten, pimpinan, laboran dan teknisi, tabel dosen, tabel perguruan tinggi, tabel program studi, laporan profil, laporan mahasiswa, laporan data dosen, laporan pendidikan, laporan jabatan akademik dosen, laporan sebaran ip mahasiswa, laporan sebaran ipk mahasiswa

Contoh diagram WSDL yang dihasilkan seperti gambar 9.

Diagram Paket



Gambar 10. Diagram paket Simanjur

Penjelasan diagram paket di atas adalah sebagai berikut:

Paket classes, berisi file-file kelas yang digunakan. Terdiri atas subpaket *model*, *collection* dan *data_access*

- Subpaket *model* berfungsi untuk membuat suatu objek/entity, contohnya pada objek *program_studi*, *fakultas*, *mahasiswa* dan lain sebagainya. Paket ini berisi file-file *entity* sistem yang dapat dilihat pada diagram kelas.

- Subpaket *collection* berisi kelas-kelas *collection* yang berfungsi untuk menampung suatu objek entity pada suatu kumpulan atau koleksi objek entity.
- Subpaket *data_access* berisi kelas-kelas yang berhubungan dengan akses database untuk masing-masing kelas *entity*.

Paket lib, berisi file-file *library* yang sering digunakan. Paket ini terdiri atas subpaket *template engine smarty* untuk mengakses *web template* dan subpaket *nusoap* untuk membuat *web service*. Untuk membuat *web service* menggunakan *nusoap*, penulis menggunakan *methode-methode* khusus yang ada pada *nusoap* seperti:

- configureWSDL*, *methode* ini berguna untuk mengkonfigurasi dokumen WSDL yang akan diterbitkan oleh *web service*.
- addComplexType*, *methode* ini berguna untuk menambah tipe data yang akan digunakan pada parameter input atau nilai kembalian dari proses *web service*.
- register*, *methode* ini berguna untuk meregister atau mempublikasikan *web service* yang telah dikonfigurasi oleh *methode-methode* sebelumnya.

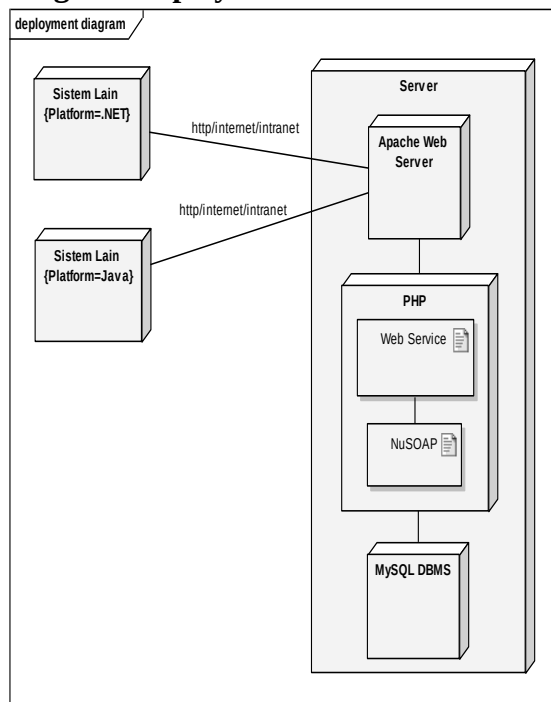
Paket templates, paket ini berisi file *template layout* halaman *web*. Paket ini terdiri atas subpaket *template* yang berisi file *template* dan subpaket *compiler* yang berisi file hasil gabungan antara file *php* dengan *template* yang di *compile smarty*. Paket ini bergantung juga terhadap paket *resource* dalam hal akses *image*, *banner*, *CSS*, dan hal hal yang berhubungan dengan desain *web* lainnya.

Paket resource, berisi subpaket yang berhubungan dengan pengaturan desain halaman *web*. Paket ini terdiri atas subpaket *css* yang berisi file *CSS (Cascading Style Sheet)*, subpaket *image* yang berisi *banner* dan gambar-gambar

lainnya dan subpaket *js* yang berisi fungsi-fungsi *java script* untuk mendukung fungsionalitas *web* seperti menu, tab dan lain sebagainya.

Paket client, merupakan paket umum berisi file-file yang berhubungan dengan pemakaian fungsi-fungsi yang ada di dalam paket *classes* dan paket *lib*. Paket *client* menggunakan objek-objek yang ada pada paket *classes* dan paket *lib* untuk menjalankan *method-method* yang ada didalamnya. Sehingga paket *client* bergantung dengan paket *classes* dan paket *lib*.

Diagram Deployment



Gambar 11. Diagram Deployment

3. Pengujian

Pengujian Unit (Unit Testing)

Pengujian Unit ini diperlukan untuk mengetahui lebih dini dan meminimalisir *bugs* yang dapat terjadi ketika proses *coding* akan di mulai pada suatu unit maupun ketika aplikasi selesai dibuat. Proses pengujian (*testing*) ini adalah bentuk penyesuaian antara suatu bagian unit sistem dengan *requirements* yang sudah dibuat.

Dalam pembuatan Simanjur ini *unit testing* dilakukan untuk menguji kelas yang ada pada paket *classes*. Untuk menguji kelas - kelas tersebut digunakan IDE Eclipse 3.2. Berikut contoh pengujian suatu kelas dengan *entity* bernama universitas:

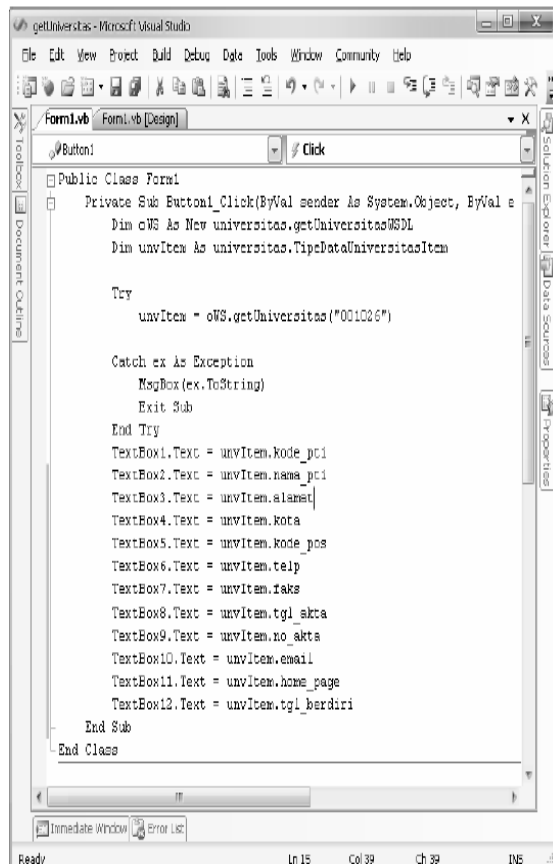
```

1 <?php
2 require_once"/../classes/collection/entity_object.class.php";
3 require_once"/../classes/data_access/universitas_da.class.php";
4
5 try(
6     $objEnt = new entity_object();
7     universitas_da::getUniversitas($objEnt);
8     $objUnv = $objEnt->getObject();
9
10    echo "Nama Perguruan Tinggi " . " : " . $objUnv->getNamaPti() . "<br/>";
11    echo "Alamat " . " : " . $objUnv->getAlamat() . "<br/>";
12    echo "Kota " . " : " . $objUnv->getKota() . "<br/>";
13    echo "Kode Pos " . " : " . $objUnv->getKodePos() . "<br/>";
14    echo "Telp. " . " : " . $objUnv->getTelp() . "<br/>";
15    echo "Faks. " . " : " . $objUnv->getFaks() . "<br/>";
16    echo "Tgl. Akta " . " : " . $objUnv->getTglAkta() . "<br/>";
17    echo "No Akta " . " : " . $objUnv->getNoAkta() . "<br/>";
18    echo "Email " . " : " . $objUnv->getEmail() . "<br/>";
19    echo "Homepage " . " : " . $objUnv->getHomepage() . "<br/>";
20    echo "Tgl. Berdiri " . " : " . $objUnv->getTglBerdiri() . "<br/>";
21
22 )catch (Exception $ex){
23     echo "Terjadi kesalahan : <br/>" . $ex->getMessage();
24 }
25 ?>
  
```

Gambar 12. Unit testing pada kelas universitas

Testing dilakukan secara manual, bertujuan untuk mengambil data universitas pada kelas universitas melalui perantara kelas universitas_da dan kelas entity_object dan menampilkannya pada *web browser*. *Testing* ini dilakukan sebelum kelas universitas digunakan ke dalam aplikasi yang sebenarnya. Pada objek *\$objEnt* dibuat sebuah objek baru dari kelas *entity_object*, kelas ini berfungsi sebagai kelas perantara sebelum kelas universitas yang sebenarnya digunakan, kemudian dilakukan pengambilan data dari database melalui kelas universitas_da -> getUniversitas dan hasilnya ditampung pada objek *\$objEnt*. Hasil yang ada pada objek *\$objEnt* diambil dan ditampung pada objek *\$objUnv* melalui metode *\$objEnt->getObject()*. Objek *\$objUnv* ini merupakan instance dari kelas universitas yang merupakan produk akhir dari pengambilan data dari basis data yang datanya dapat ditampilkan melalui metode *get* Seperti yang dapat dilihat

pada saat pengujian melalui *browser* di bawah ini :



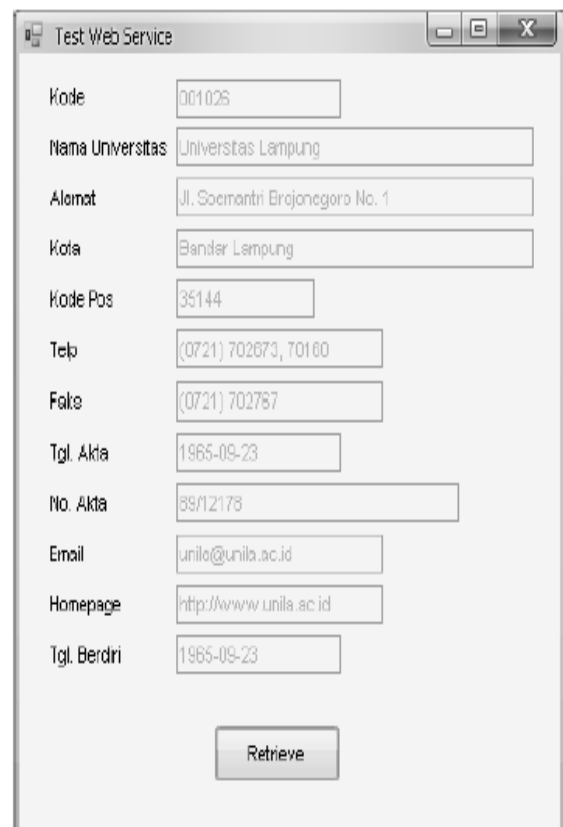
Gambar 13. Pengujian kelas *query* melalui *browser*

Pengujian Web Service

Telah dilakukan uji coba *web service* dengan dua platform yang berbeda, yaitu Java dan .NET.

a. Pengujian dengan platform .NET

Sebelum melakukan *coding* untuk mengakses *web service* dengan .NET perlu ditambahkan referensi *web service* kedalam IDE Visual Studio .NET 2005, sehingga referensi objek tersebut dapat digunakan pada kode program untuk mengakses data yang ada pada *web service* tersebut. Berikut tampilan hasil pengujian :

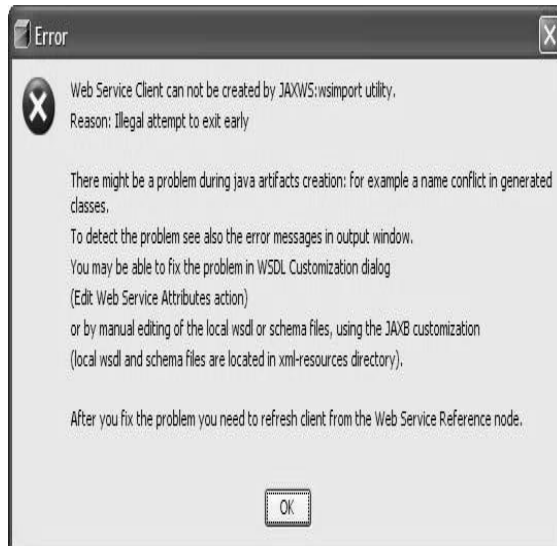


Gambar 14. Pengujian *web service* pada platform .NET

b. Pengujian dengan platform Java

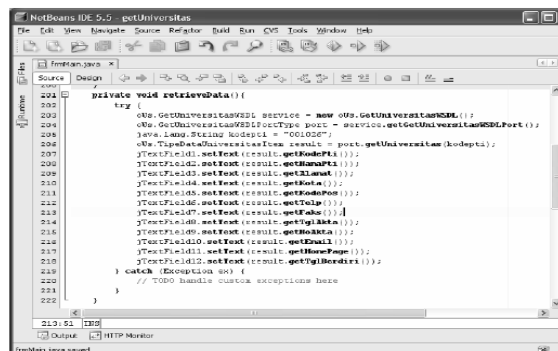
Pengujian menggunakan IDE NetBeans 5.5 dan JDK 1.6, awalnya diperoleh

pesan kesalahan ketika menambahkan referensi *web service*.



Gambar 15. Pesan kesalahan pada IDE NetBeans

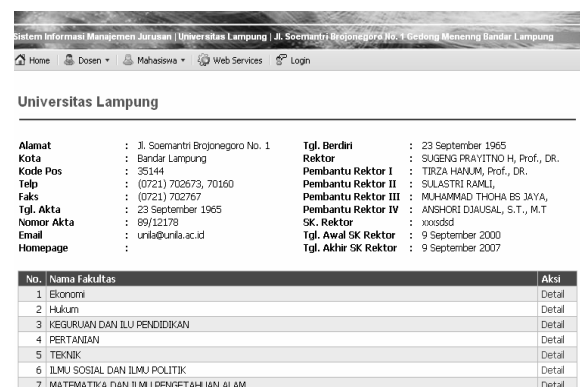
Pesan kesalahan ini disebabkan karena WSDL yang dihasilkan dari web PHP tidak sesuai dengan WSDL yang diinginkan *platform* Java sehingga perlu dilakukan perubahan pada dokumen WSDL pada IDE NetBeans agar WSDL tersebut dapat dimengerti oleh *platform* Java. Setelah dokumen WSDL diubah maka *web service* dapat dijalankan pada *platform* Java, berikut adalah gambar test *web service* pada *platform* Java:



Gambar 16. Pengujian *web service* pada *platform* Java

Pengujian *release beta*

Pada tahap ini sebagian besar fungsi kerja sistem sudah dapat berjalan.



Gambar 17. Tampilan halaman muka versi beta

Keuntungan SOA

Implementasi SOA menggunakan *web service* memiliki beberapa keuntungan yaitu:

a. Decoupling

Dengan adanya *web service* maka ketergantungan *service consumer* terhadap *provider* kecil. Masing-masing dapat dikembangkan secara terpisah. *Web service* dapat dikembangkan tanpa mempengaruhi *consumer*-nya. Pada sistem Simanjur sifat *decoupling* telah ada. Terbukti dengan adanya *web service* masing-masing logika dapat dikembangkan terpisah. Logika untuk pengambilan data dikerjakan oleh *web service* PHP sedangkan untuk *user interface*

dikerjakan oleh sistem lain berupa Java dan .NET.

b. Peningkatan *reusability*

Dari hasil pengujian *web service* dengan Java dan .NET terbukti bahwa dengan *web service* dapat meningkatkan *reusability* sistem yang dibuat. Penulis tidak perlu menambahkan kode program untuk mengakses data atau membuat database baru untuk mendapatkan data yang diinginkan cukup menggunakan *web service* yang sudah ada.

c. Kemudahan integrasi sistem

Oleh karena *web service* dikembangkan dengan standar yang baku (antara lain HTTP, SOAP) maka *web service* dapat dengan mudah diintegrasikan dengan sistem lain tanpa mengganggu logika yang ada pada sistem tersebut. Hal tersebut telah diterapkan pada sistem Simanjur yang menggunakan *web service* sebagai implementasi SOA.

Kekurangan SOA

Ada beberapa jenis aplikasi yang tidak cocok menggunakan konsep SOA diantaranya:

- a. Aplikasi *stand-alone*, yaitu bukan aplikasi terdistribusi yang tidak mengharuskan adanya integrasi komponen aplikasi yang akan disertakan. contohnya aplikasi pengolahan kata (*word processing*) yang tidak membutuhkan fitur berbasis *request* dan *response*.
- b. *Short live application*, yaitu aplikasi yang terbatas pada scope tertentu. contohnya aplikasi yang dibuat untuk solusi sementara yang tidak diharapkan untuk menyediakan fungsi yang maksimal atau dapat digunakan kembali pada aplikasi yang akan datang.
- c. Aplikasi yang hanya membutuhkan komunikasi *asynchronous* satu arah dan dimana *loose coupling* tidak diinginkan dan tidak diperlukan.

d. Aplikasi yang memiliki lingkungan yang homogen. seperti aplikasi yang dibuat dengan menggunakan komponen J2EE. Dalam hal ini lebih baik menggunakan *Java remote method invocation (JRMi)* dibandingkan menggunakan XML pada protokol HTTP untuk komunikasi antar komponen.

e. Aplikasi yang membutuhkan fungsionalitas berbasis GUI, seperti aplikasi yang memanipulasi *map*(peta) yang memiliki banyak manipulasi data geografis dan juga tidak cocok untuk aplikasi yang membutuhkan pertukaran data yang besar.

E. Kesimpulan dan Saran

Kesimpulan

Dari hasil dan pembahasan dapat diambil simpulan sebagai berikut:

1. Telah dibuat dan diuji aplikasi Simanjur menggunakan *web service* yang mengimplementasikan SOA.
2. Mengonsumsi *web service* pada *platform* Java memerlukan perubahan khusus pada dokumen WSDL.
3. Sesuai pengujian, implementasi SOA menggunakan *web service* dalam aplikasi Simanjur memiliki keuntungan seperti *decoupling*, peningkatan *reusability*, kemudahan integrasi sistem.

Saran

Saran yang dapat diberikan adalah sebagai berikut:

1. Bagi *developer* sistem yang akan mengimplementasikan SOA menggunakan *web service* dapat menggunakan bahasa pemrograman PHP dengan memanfaatkan *library* Nusoap dengan menggunakan *methode-methode* seperti: *configureWSDL*, *addComplexType*, *register*.
2. Pembangunan aplikasi menggunakan *web service* dapat menggunakan

platform seperti J2EE dan .NET, karena menyediakan API dan dokumentasi yang lengkap.

3. Dalam aplikasi Simanjur ini baru di implementasikan pada satu DBMS saja. Untuk pengembangannya dapat di buat agar bisa mendukung lebih banyak DBMS lagi.

Daftar Pustaka

- [1] Siswoutomo, W. 2004. *Membangun Web Service Open Source Menggunakan PHP*. Elex Media Komputindo, Jakarta.
- [2] MediaWiki. *Service-oriented architecture*. http://en.wikipedia.org/wiki/Service_Oriented_Architecture
- [3] Newcomer, dkk., 2004. *Understanding SOA with Web Services*. Addison Wesley Professional, USA.
- [4] Krafzig, D., Banke, K., Slama, D., 2004. *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall PTR, USA.
- [5] Kappel, Gerti et al. 2006. *Web Engineering 'The Discipline of Systematic Development of Web Applications'*. John Wiley & Sons Ltd, England.
- [6] Pender, T., 2003, *UML Bible*. John Wiley & Sons, West Sussex, UK.