TUGAS AKHIR - CF 1380

PEMBANGUNAN MODUL APOTEK DAN PENGINTEGRASIAN DALAM PURWARUPA SISTEM INFORMASI RUMAH SAKIT TERPADU BERBASIS SOA

ARIYANTO ADI NUGROHO
NRP 5205 100 044

Dosen Pembimbing
Ir. A. Holil N.A, M.Kom
Faizal Mahananto, S.Kom

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2009

ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT - CF 1380

THE DEVELOPMENT OF PHARMACY MODULE AND ITS INTEGRATION IN THE PROTOTYPE OF INTEGRATED HEALTH INFORMATION SYSTEM BASED ON SOA

ARIYANTO ADI NUGROHO
NRP 5205 100 044

Dosen Pembimbing
Ir. A. Holil N.A, M.Kom
Faizal Mahananto, S.Kom

INFORMATION SYSTEM OF DEPARTMENT
Information Technology Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2009

PEMBANGUNAN MODUL APOTEK DAN PENGINTEGRASIAN DALAM PURWARUPA SISTEM INFORMASI RUMAH SAKIT TERPADU BERBASIS SOA

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

ARIYANTO ADI NUGROHO
NRP 5205 100 044

Surabaya, 10 Agustus 2009

KETUA
JURUSAN SISTEM INFORMASI

DEPARTEMEN PENDIDIKAN NASIONAL INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI INFORMASI JURUSAN SISTEM INFORMASI

Ir. A. HOLIL NOOR ALI, M.KOM
NIP 131 996 150

PEMBANGUNAN MODUL APOTEK DAN PENGINTEGRASIAN DALAM PURWARUPA SISTEM INFORMASI RUMAH SAKIT TERPADU BERBASIS SOA

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

ARIYANTO ADI NUGROHO
NRP 5205 100 044

Disetujui Tim Penguji : **Tanggal Ujian : 5 Agustus 2009**
Periode Wisuda : Oktober 2009

Ir. A. Holil N.A, M.Kom **(Pembimbing I)**

Faizal Mahananto, S.Kom **(Pembimbing II)**

Bekti Cahyo, S.Si, M.Kom **(Penguji I)**

Danu Prananthe, S.T, M.Sc **(Penguji II)**

PEMBANGUNAN MODUL APOTEK DAN PENGINTEGRASIAN DALAM PURWARUPA SISTEM INFORMASI RUMAH SAKIT TERPADU BERBASIS SOA

Nama Mahasiswa	: Ariyanto Adi Nugroho
NRP	: 5205 100 044
Program Studi	: Sistem Informasi FTIf – ITS
Dosen Pembimbing	: Ir. A. Holil N.A, M.Kom
	Faizal Mahananto, S.Kom

ABSTRAKSI

Desain SIRST(Sistem Informasi Rumah Sakit Terpadu) berbasis SOA merupakan hasil restrukturisasi SIRST versi 1 dengan menambahkan SOA sebagai arsitektur aplikasi. SOA dapat memberikan standard komunikasi data dan informasi modul-modul dalam SIRST berbasis SOA. Akibatnya, dalam pembangunan SIRST berbasis SOA menjadi kompleks, karena harus menciptakan standard komunikasi yang memiliki daya interoperabilitas dan skalabilitas. Kemampuan ini dibutuhkan untuk menciptakan Sistem Informasi Rumah Sakit yang memiliki sifat tumbuh kembang.

Pembangunan SIRST berbasisi SOA modul Apotek diawali dengan melakukan pemahaman terhadap dokumen SKPL, DPPL dan framework dari SIRST versi 2. Kemudian masuk ke tahap pembangunan modul Apotek, dimulai dengan melakukan code convention,. Selanjutnya dilakukan pembuatan database, pembuatan form(antarmuka) dan pembuatan servis (ambil, ubah dan tambah data). Pada saat yang sama juga dilakukan pengujian terhadap database dan fungsi-fungsinya. Setelah itu dilakukan integrasi dan validasi dengan modul-modul SIRST versi 2 lainnya. Dilanjutkan dengan uji coba purwarupa dan diakhiri dengan pendokumentasian kode program.

Hasil yang diharapkan dari tugas akhir ini adalah mendemonstrasikan SIRST berbasis SOA yang dibangun

dengan arsitektur SOA dan memiliki sifat tumbuh kembang serta terintegrasi dengan modul-modul SIRST berbasis SOA lainnya.

Kata kunci: Apotek, sistem informasi rumah sakit terpadu, web service,PHP,SOA

THE DEVELOPMENT OF PHARMACY MODULE AND ITS INTEGRATION IN THE PROTOTYPE OF INTEGRATED HEALTH INFORMATION SYSTEM BASED ON SOA

Student Name	: Ariyanto Adi Nugroho
NRP	: 5205 100 044
Departement	: Sistem Informasi FTIf – ITS
Supervisors	: Ir. A. Holil N.A, M.Kom
	Faizal Mahananto, S.Kom

ABSTRACT

The Integrated Health Information System design based on SOA is the result of the reconstruction / synthesis of the first version of Integrated Health Information System by adding SOA as an application architecture The SOA can contribute the standard of data communication and module information in the Integrated Health Information System based on the SOA. As a result, the deveplopment of the Integrated Health Information System based on the SOA becomes complex, because it needs the creation of communication standart which possesses the power of interoperability and scalability. This kind of power is required to create a Integrated Hospital Information System which is growing and developing.

The development of the Integrated Health Information System based on the SOA in the pharmacy module begins with the understanding of SKPL documents, DPPL and the framework of the second version of SIRST. Then it goes on to the stage of the construction of pharmacy module, starting with doing convention codes. Then it continues to make database, to construct interface forms, and to establish services (to collect, to change, to add data) using PHP. At the same time the testing of database and their functions is carried out. After that the integration and validation with the other Integrated Health Information System based on SOA

module is done. Finally the try out of prototype is carried out and it ends with the documentation of program code.

The expected result of this final assignment is to exhibit / describe the Integrated Health Information System which is developed with / through the SOA architecture that possesses growing and developing characteristics and integrated with the other SIRST based on SOA modules.

Kata kunci: pharmacy, integrated health information system, web service,PHP,SOA

KATA PENGANTAR

Alhamdulillahirabbilalamiin atas segala karunia dan kasih sayang-NYA, sehingga tugas akhir berjudul "PEMBANGUNAN MODUL APOTEK DAN PENGINTEGRASIAN DALAM PURWARUPA SISTEM INFORMASI RUMAH SAKIT TERPADU BERBASIS SOA " dapat terselesaikan dan menghantarkan penulis menjadi sarjana komputer dari Program Studi Sistem Informasi, Fakutas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Terima kasih dan penghargaan setinggi-tingginya juga penulis sampaikan kepada:

1. Bpk. Ir. Ahmad Holil Noor Ali, M.Kom dan Bpk. Faizal Mahananto, S.Kom selaku dosen pembimbing yang telah memberikan bimbingan dan motivasi kepada penulis.
2. Bu Mahendrawathi selaku Ketua Predict.
3. Semua Bapak dan Ibu Dosen pengajar di Program Studi Sistem Informasi ITS yang telah memberikan ilmu yang berharga kepada penulis.
4. Seluruh staf karyawan TU Program Studi Sistem Informasi dan karyawan Fakultas Teknologi Informasi atas dukungannya sehingga tugas akhir ini dapat terselesaikan.
5. Mas Didit, terima kasih atas framework OHIS yang dibuat serta kerjasama dan pengalaman baru yang didapat, hal itu yang sangat berarti bagi penulis.
6. Teman-teman senasib seperjuangan *predict*, Amna, Ekawati, Danu, Yusuf, Galuh. Terima kasih atas dukungan moril dan dorongan semangat yang diberikan.

7. Teman-teman sesama bimbingan pak Holil, Venty, Rista, Mbak Ima, dan Anif (yang membuat pulang bimbingan selalu malam)
8. Semua teman-temanku SI 2005, yang tidak dapat disebutkan satu per satu, penulis tidak akan melupakan persaudaraan yang telah terjalin selama ini.
9. Semua teman-teman di Sistem Informasi, SI'06, dan SI'07, terima kasih telah menjadi bagian dari SI.
10. Berbagai pihak yang belum sempat penulis sebutkan jasa-jasanya dalam mendukung penyusunan tugas akhir ini.

Penulis sangat menyadari bahwa tugas akhir ini masih jauh dari sempurna. Oleh karena itu penulis mengharapkan komentar, kritik, dan saran dari berbagai pihak.

Akhirnya, penulis berharap semoga keberadaan tugas akhir ini bermanfaat banyak bagi ilmu pengetahuan dan berbagai pihak.

Surabaya, Juli 2009

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan.

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan.

BAB I
PENDAHULUAN

Pada pendahuluan dituliskan hal-hal yang menitikberatkan kepentingan diadakan penelitian. Dalam pendahuluan dikemukakan proses-proses dalam mengidentifikasi masalah penelitian. Komponen-modul dalam bab ini diantaranya adalah : (1) Latar belakang masalah ; (2) Perumusan masalah ; (3) Batasan masalah ; (4) Tujuan Tugas Akhir dan (5) Relevansi atau Manfaat Kegiatan Tugas Akhir.

1.1 Latar Belakang

Sistem informasi rumah sakit mulai dikembangkan seiring dengan kebutuhan rumah sakit akan pengelolaan informasi yang terautomatisasi. Salah satunya adalah SIRST (Sistem Informasi Rumah Sakit Terpadu). Saat ini, SIRST telah mengalami dua versi pengembangan, SIRST versi 1 dan SIRST berbasis SOA. SIRST berbasis SOA merupakan hasil restrukturisasi dari SIRST versi 1 yang masih bersifat tradisional memiliki beberapa kelemahan, yaitu kesulitan dalam hal menambah dan mengurangi servis atau modul dan interoperabilitas.

Proses pengembangan SIRST versi 1 menggunakan metode pendekatan independen dimana setiap modul dikerjakan dengan konsep yang sama oleh masing-masing *programmer*, sedangkan pada SIRST berbasis SOA pengembangan programnya sudah digeneralisasi. Fungsi dari generalisasi adalah memudahkan penggunaan-penggunaan fitur dasar dari sebuah sistem informasi, misalnya tata letak menu, visualisasi data, pemetaan jenis data, registrasi fungsi servis-servis sehingga sistem informasi yang dihasilkan memiliki interface yang sama untuk semua servisnya walaupun berbeda dari segi *platform* perangkat lunak. Kelebihan lainnya dilihat dari sudut pandang *programmer*

ialah *programmer* hanya fokus pada fungsi-fungsi apa saja yang terdapat dalam suatu modul atau servis tanpa harus memikirkan desain tampilan dan visualisasi data karena telah disediakan oleh *framework*

Arsitektur SOA yang diimplementasikan dalam SIRST berbasis SOA diharapkan dapat mengeliminasi kelemahan-kelemahan yang terdapat pada SIRST versi 1. SOA digunakan karena memiliki kemampuan untuk melepaskan hambatan dalam hal interoperabilitas dan tumbuh kembangnya sistem informasi. Hal ini dikarenakan SOA memberikan suatu standard komunikasi antar data dan informasi, sehingga dalam pengembangannya memiliki desain yang cukup kompleks. Maka diperlukan sebuah purwarupa untuk menunjukkan kemampuan SOA yang memiliki sifat tumbuh kembang.

Sistem Informasi Rumah Sakit memiliki modul-modul inti diantaranya manajemen, apotek, rawat inap dan rawat jalan. Tugas Akhir ini penulis akan mengembangkan modul apotek.

1.2 Rumusan Permasalahan

Permasalahan yang akan diangkat di dalam penyusunan tugas akhir ini, adalah:

1. Bagaimana mengubah rancangan SIRST berbasis SOA menjadi kode pemrograman PHP yang berorientasi SOA?
2. Bagaimana fungsi-fungsi dalam modul Apotek dapat berjalan pada *framework* SIRST berbasis SOA?

1.3 Batasan Masalah

Berdasarkan permasalahan di atas, maka batasan dalam tugas akhir ini adalah sebagai berikut:

1. Pembangunan modul Apotek dibuat sesuai dengan *framework* SIRST berbasis SOA.
2. Uji coba purwarupa dilakukan dengan metode uji fungsionalitas

1.4 Tujuan

Tujuan dari penyusunan tugas akhir ini adalah untuk membangun modul Apotek dan mengintegrasikannya dengan modul lain berdasarkan SKPL dan DPPL SIRST berbasis SOA.

1.5 Manfaat

Manfaat yang diberikan dalam penyusunan tugas akhir ini adalah sebagai berikut:

1. Memperoleh tips dan trik pengimplementasian sistem informasi dengan menggunakan SOA.
2. Modul Apotek yang dapat terintegrasi pada purwarupa SIRST berbasis SOA sehingga siap untuk diimplementasikan.

1.6. Sistematika Pembahasan

Secara garis besar Penulisan dalam Tugas Akhir ini terbagi dalam lima Bab, dimana materi dari setiap bab dapat dituliskan sebagai berikut:

BAB I : Pendahuluan

Bab ini berisi uraian mengenai latar belakang permasalahan, tujuan dari Tugas Akhir, manfaat Tugas Akhir, perumusan masalah, batasan masalah serta sistematika yang digunakan dalam pembahasan masalah ini.

BAB II : Tinjauan Pustaka

Pada bab ini akan membahas mengenai teori-teori yang mendukung pembuatan tugas akhir (TA), yaitu tentang pembangunan Sistem Informasi Rumah Sakit Terpadu berbasis SOA menggunakan bahasa pemrograman PHP.

BAB III : Metodologi

Bab ini menerangkan mengenai metodologi yang digunakan dalam pengerjaan tugas akhir.

Mulai dari pemahaman informasi, pembangunan modul apotek, pengintegrasian dan validasi modul apotek, uji coba modul apotek yang terintegrasi, hingga melakukan dokumentasi aplikasi modul apotek.

BAB IV : Analisa dan Desain Sistem

Bab ini akan menjelaskan proses pemahaman informasi yang terdiri dari *review* DPPL yang dibuat oleh Amna Shifia Nisafani dan Adi Fitroni.

Pemahaman *framework* OHIS yang digunakan dalam membangun modul-modul dasar apotek. Kemudian *code convention* yang menjadikan dasar penulisan program. Pembuatan tabel dan relasi dari modul, kemudian pembuatan servis apotek.

BAB V : Uji Coba

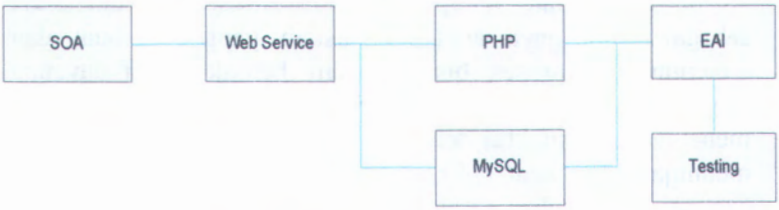
Bab uji coba akan merangkum hasil-hasil uji coba modul apotek dari segi uji integritas dan uji fungsionalitas. Uji integrasi meliputi uji coba *cross-functional module*, instalasi servis, dan penghapusan servis. Uji coba fungsional yang menggunakan metode *black box*.

BAB VI : Penutup

Bab ini merangkum hasil akhir dari pembuatan Tugas Akhir menjadi sebuah simpulan dan dilengkapi dengan saran-saran untuk perbaikan ataupun penelitian lanjutan. Simpulan merupakan rangkuman dari hasil pembuatan sistem informasi. Sedangkan saran merupakan usulan atau rekomendasi dari peneliti terhadap hasil Tugas Akhir untuk perbaikan ataupun penelitian lanjutan sehingga hasil Tugas Akhir ini dapat diimplementasikan dengan baik.

BAB II
TINJAUAN PUSTAKA

Tinjauan pustaka menguraikan teori, temuan, dan bahan penelitian lain yang diperoleh dari acuan yang akan dijadikan landasan untuk melakukan kegiatan penelitian yang akan dijadikan tugas akhir, yaitu tentang SOA, *Web Service*, PHP, MySQL, EAI, *Code Test*. Berikut ini adalah bagan hubungan teori-teori yang digunakan dalam mendukung pembuatan TA ini.



Gambar 2.1 Bagan Hubungan Teori

Keterangan :

1. SOA (*Servis Oriented Architecture)* adalah arsitektur yang digunakan dalam pembangunan modul apotek yang dikerjakan dalam tugas akhir ini.
2. Teknologi *web service* digunakan untuk membangun *framework* oleh karena itu dibutuhkan untuk pemahaman struktur dan cara kerja dari *framework.*
3. PHP sebagai bahasa pemrograman yang digunakan dalam pembangunan modul apotek.
4. MySQL adalah basis data yang tidak berbayar, yang digunakan dalam pembangunan modul apotek.
5. EAI (Entreprise Application Integration) sebagai dasar pengintegrasian modul apotek dengan modul SIRST berbasis SOA lainnya.

6. *Testing* sebagai cara pengujian terhadap aplikasi modul apotek berbasis SOA.

2.1 *Service Oriented Achitecture (SOA)*

Rosen, Mike (2008) mendefinisikan SOA sebagai suatu gaya arsitektur untuk membangun suatu solusi bagi *enterprise* dengan berdasarkan servis atau layanan[8]. Lebih spesifik lagi, beliau juga menekankan bahwa SOA lebih terkait dengan konstruksi independen dari suatu layanan yang sesuai dengan bisnis, yang kemudian dapat dikombinasikan menjadi proses bisnis tingkat tinggi dan memberikan solusi dalam konteks *enterprise.*

Sedangkan Wikipedia (2009) mendefinisikan SOA sebagai suatu gaya arsitektur sistem yang membuat dan menggunakan proses bisnis dalam bentuk paket layanan sepanjang siklus hidupnya[10]. SOA juga mendefinisikan dan menentukan arsitektur teknologi informasi (TI) yang dapat menunjang berbagai aplikasi untuk saling bertukar data dan berpartisipasi dalam proses bisnis. Fungsi-fungsi ini tidak terikat dengan sistem operasi dan bahasa pemrograman yang mendasari aplikasi-aplikasi tersebut.

Dapat disimpulkan dari definisi-definisi SOA di atas, bahwa SOA adalah gaya arsitektur sistem yang berbasis servis atau layanan yang mengutamakan layanan yang dapat independen dan interoperabilitas antar layanan tanpa bergantung pada *platform*nya.

SOA membagi fungsi-fungsi menjadi unit-unit yang berbeda (layanan), yang dapat didistribusikan melalui suatu jaringan dan dikombinasikan serta digunakan ulang untuk membentuk aplikasi bisnis. Layanan-layanan ini saling berkomunikasi dengan mempertukarkan data antar mereka atau dengan mengkoordinasikan aktivitas antara dua atau lebih layanan. Konsep SOA sering dianggap didasari atau berkembang dari konsep-konsep yang lebih lama dari komputasi terdistribusi dan pemrograman modular.

2.2 *Web Service*

Perdebatan mengenai *web service* sering terjadi. Sekalipun W3C yang menciptakan *Web Service Architecture Working Group* untuk menciptakan arsitektur dokumen *web service*, mengeluarkan beberapa statemen definisi *web service*. Pada akhirnya perdebatan itu merajuk pada satu hal mengenai definisi *web service* pada 11 Februari 2004, yang mengatakan bahwa, *Web service* adalah sebuah sistem perangkat lunak yang dibuat untuk mendukung interoperabilitas dari komputer ke komputer dalam jaringan. Antarmuka yang didefinisikan sebagai format yang dikenali komputer (khususnya WSDL). Sistem lain yang berinteraksi dengan *web service* dengan memberikan deskripsinya menggunakan SOAP *messages*, kemudian disampaikan menggunakan HTTP dengan XML bersama dengan standar yang berkaitan dengan *web*. Jadi, *web service* merupakan suatu sistem perangkat lunak yang dapat diakses melalui *web* baik secara internet maupun intranet. Web *service* sepenuhnya berdasarkan standart *web* dan XML, sehingga dalam penggunaannya, *web service* memungkinkan interaksi dan komunikasi antar sistem operasi maupun bahasa pemrograman yang berbeda. Beberapa perusahaan besar telah mengimplementasikan *web service*, seperti Yahoo, Google, eBay dan Amazon.

Berbagai keuntungan dari penggunaan *web service* antara lain:

- Interoperabilitas yang tinggi
- Format penggunaan terbuka untuk semua *platform*
- Mudah di mengerti dan mudah men-*debug*
- Dukungan *interface* yang stabil
- Untuk mengimplementasikannya tidak terlalu mahal
- Untuk mengakses layanan tidak diperlukan komputer berspesifikasi tinggi
- Dapat diakses dari manapun, asalkan masih terhubung jaringan

2.2.1 Simple Object Access Protocol (SOAP)

SOAP adalah salah satu protokol pesan XML-*web service* yang digunakan untuk pertukaran informasi dalam lingkungan komputer terdistribusi[12]. SOAP mendefinisikan mekanisme untuk transfer pesan antar sistem terdistribusi dan prosedur panggilan jarak jauh atau *Remote Procedure Call* (RPC). SOAP dapat digunakan di *platform* baik perangkat keras maupun sistem operasinya. SOAP hanya menggunakan pesan XML yang dapat dimengerti oleh semua bahasa pemrograman. SOAP adalah metode antara *Client* dan *Server* di mana client mengirimkan *request service* dan *server* mengirimkannya kembali. Banyak orang mengetahui tentang protokol HTTP, tetapi HTTP kurang aman untuk melakukan transfer pesan. Jadi, SOAP dapat memberikan keamanan yang lebih baik dalam transfer pesan.

Fitur-fitur dari SOAP yaitu :

- *A messaging framework*, mendefinisikan *framework* secara keseluruhan untuk mengekspresikan isi dari pesan; siapa yang mempunyai autoritas dan apakah pesan optional atau mandatori.
- *Encoding/serialization standard*, mendifinisikan urutan mekanisme yang dapat digunakan untuk melakukan pertukaran instance dari aplikasi.
- *Remote Procedure Call* (RPC) mekanisme, digunakan oleh program yang meminta sevis dari program yang lainnya dimana program-program berada dalam 1 jaringan tetapi tidak ada dalam 1 komputer.

Pesan SOAP tersidiri dari tiga bagian yaitu:

- *Envelope*, mendifinisikan awal dan akhir dari pesan
- *Header*, termasuk semua kondisi yang dibutuhkan pesan
- *Body*, berisi pesan dan XML data yang akan ditransimisikan dalam jaringan.

Keuntungan penggunaan SOAP dibandingkan dengan metode *web service* yang lainnya [1]:

- Mendukung keamanan tingkat *transport* dan *message*
- Mendukung beberapa *protocol bindings* (Tidak hanya HTTP)
- Pesan SOAP yang berada dalam format XML yang sederhana dan mudah dimengerti.
- Komunitas *open source* mendukung SOAP. Jadi terdapat lebih banyak dari kesempatan pembangunan.
- Untuk pengiriman pesan, SOAP menggunakan HTTP protokol yang memungkinkan skalabilitas lebih tinggi.
- Ideal untuk perusahaan menerapkan sistem rumit karena susunan SOAP diperuntukkan bagi keamanan, keandalan, transaksi dll.

2.4 PHP

PHP adalah bahasa pemrograman *script* yang paling banyak dipakai saat ini. PHP banyak dipakai untuk memrogram situs *web* dinamis, walaupun tidak tertutup kemungkinan digunakan untuk pemakaian lain.

Contoh terkenal dari aplikasi PHP adalah phpBB dan MediaWiki (software di belakang Wikipedia). PHP juga dapat dilihat sebagai pilihan lain dari ASP.NET/C#/VB.NET Microsoft, ColdFusion Macromedia, JSP/Java Sun Microsystems, dan CGI/Perl. Contoh aplikasi lain yang lebih kompleks berupa CMS yang dibangun menggunakan PHP adalah Mambo, Joomla!, Postnuke, Xaraya, dan lain-lain.

Sejarah PHP

PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama FI (Form Interpreted), yang wujudnya berupa sekumpulan *script* yang digunakan untuk mengolah data *form* dari *web*.

Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FI, kependekan dari

Hypertext Preprocessing'/Form Interpreter. Dengan perilisan kode sumber ini menjadi *open source*, maka banyak *programmer* yang tertarik untuk ikut mengembangkan PHP.

Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini interpreter PHP sudah diimplementasikan dalam program C. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP/FI secara signifikan.

Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang interpreter PHP menjadi lebih bersih, lebih baik, dan lebih cepat. Kemudian pada Juni 1998, perusahaan tersebut merilis interpreter baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0.

Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi *web* kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi.

Pada Juni 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek.

Kelebihan PHP dari bahasa pemrograman lain

- Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaanya.
- *Web Server* yang mendukung PHP dapat ditemukan dimana - mana dari mulai apache, IIS, Lighttpd, hingga Xitami dengan configurasi yang relatif mudah.
- Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan pengembang yang siap membantu dalam pengembangan.
- Dalam sisi pemahamanan, PHP adalah bahasa *scripting* yang paling mudah karena memiliki referensi yang banyak.

- PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat menjalankan perintah-perintah sistem.

Berikut ini adalah contoh penulisan PHP

- Hello World

  Program Hello World yang ditulis menggunakan PHP adalah sebagai berikut:

```
<?php
   echo "<b>Hello World</b>" ;
   echo "<b>Halo Dunia</b>" ;
?>
```

Berikut ini adalah contoh program yang relatif lebih kompleks yang ditulis dengan menggunakan PHP. Contoh program ini adalah program untuk menampilkan barisan bilangan Fibonacci.

```
<?php

$now = 1;
$prev = 0;

$jumlah = 8;

while ($i < $jumlah)
{
  $temp = $prev;

  $prev = $now;
  $now = $now + $temp;
```

```
    echo($now . ", ");
    $i++;
}

?>
```

Tipe Data

PHP memiliki 8 (delapan) tipe data yaitu :

1. Integer
2. Double
3. Boolean
4. String
5. Object
6. Array
7. Null
8. Nill

2.5 MySQL

MySQL adalah sebuah perangkat lunak sistem IRNA basis data SQL atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB

adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat *closed source* atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basis data sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. Keandalan suatu sistem basis data (DBMS) dapat diketahui dari cara kerja *optimizer*-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh user maupun program-program aplikasinya. Sebagai *server* basis data, MySQL dapat dikatakan lebih unggul dibandingkan basis data *server* lainnya dalam *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh pengguna tunggal, kecepatan *query* MySQL bisa sepuluh kali lebih cepat dari PostgreSQL dan lima kali lebih cepat dibandingkan Interbase. Selain itu MySQL juga memiliki beberapa keistimewaan, antara lain :

1. *Portability*

   MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.

2. *Open Source*

   MySQL didistribusikan secara *open source* (gratis), dibawah lisensi GPL sehingga dapat digunakan secara cuma-cuma.

3. *Multiuser*

   MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

4. *Performance tuning*

   MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.

5. *Column types*

   MySQL memiliki tipe kolom yang sangat kompleks, seperti signed / unsigned integer, float, double, char, text, date, timestamp, dan lain-lain.

6. *Command* dan *functions*

   MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam *query*.

7. *Security*

   MySQL memiliki beberapa lapisan sekuritas seperti *level subnetmask*, nama *host*, dan izin akses user dengan sistem perizinan yang mendetail serta password terenkripsi.

8. *Scalability* dan *limits*

   MySQL mampu menangani basis data dalam skala besar, dengan jumlah *records* lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

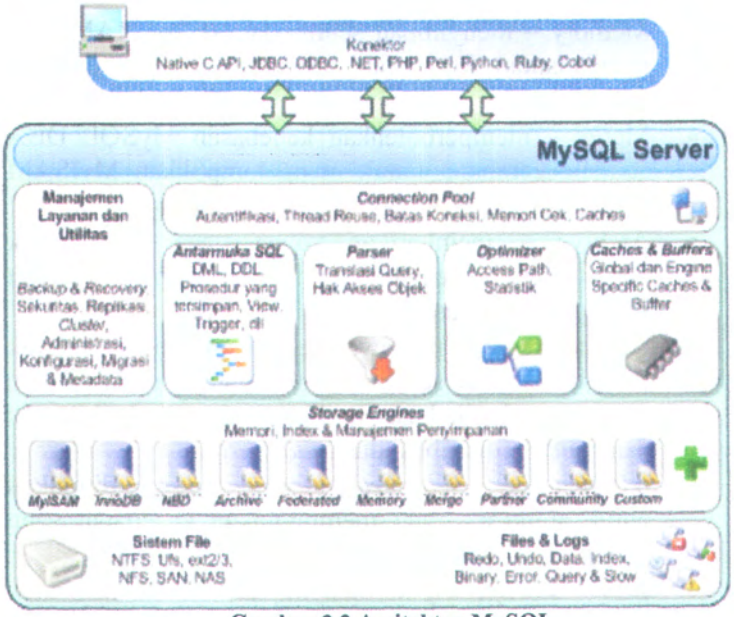
9. *Connectivity*

   MySQL dapat melakukan koneksi dengan *client* menggunakan protokol TCP/IP, Unix soket (UNIX), atau *Named Pipes* (NT).

10. *Localization*

    MySQL dapat mendeteksi pesan kesalahan pada *client* dengan menggunakan lebih dari dua puluh bahasa. Meskipun demikian, bahasa Indonesia belum termasuk didalamnya.

11. Interface



Gambar 2.2 Arsitektur MySQL

Gambar 2.2 di atas merupakan *high-level* diagram dari arsitektur server MySQL. Storage engine dalam MySQL bersifat *pluggable* sehingga dapat disesuaikan dengan kebutuhan penggguna. Pengembang dapat dengan mudah mengganti status aktif *storage engine* maupun menambah *storage engine* yang belum disediakan MySQL secara *default*. Ada 10 pilihan *storage engine* yang ditawarkan MySQL yaitu:

- MyISAM – *default storage engine* dari MySQL dan paling sering digunakan untuk *website* dan *data wearhouse*.MyISAM mendukung semua MySQL konfigurasi dan menjadi *default storage engine* kecuali jika MySQL telah dikonfigurasi untuk menggunakan *storage engine* laiinya.
- InnoDB – *transaction-safe* (sesuai dengan ACID) *storage engine* untuk MySQL .

- Memory – menyimpan semua data dalam RAM agar mendapatkan kecepatan maksimal. Seringkali, tipe *storage engine* ini disebut HEAP *engine*.
- Merge – memperbolehkan kebebasan MySQL DBA atau pengembang untuk mengelompokkan MyISAM tabel dan kunci referensi dijadikan satu objek.
- Archive – memberikan solusi yang baik untuk penyimpanan, pengambilan data, arsip dan audit kemanan informasi.
- Federated – menawarkan kemampuan untuk menghubungkan server MySQL yang terpisah dalam pembuatan logik basis data. Solusi yang baik untuk data yang terdistribusi dan *data mart*.
- NBDCluster – *clustered storage engine* ditujukan untuk aplikasi yang membutuhkan kecepatan yang paling cepat dan ketersediaan.
- CSV – *storage engine* menyimpan data dalam bentuk *text file* menggunakan koma sebagai pemisah. CSV *engine* secaara mudah bertukar data antara perangkat lunak dan aplikasi yang dapat melakukan *import* dan *export* dalam format CSV.
- Blackhole – menerima tetapi tidak menyimpan maupun mengambil data, selalu mengembalikan set kosong. Fungsionalitas ini dapat digunakan dalam desain basis data terdistribusi dimana data secara otomatis tereplikasi, tetapi tidak disimpan dalam lokal.
- Example – *storage engine* ini mempunyai fungsionalitas yang mirip dengan Blackhole karena data tak bisa disimpan maupun diambil meskipun tabel dapat dibuat. Tujuan dari Example adalah sebagai contoh dalam *source code* MySQL yang dapat mengilustrasikan bagaimana memulai menulis *storage engine* yang baru.

Perbandingan fitur yang dimiliki oleh beberapa *storage engine* yang disediakan MySQL, dijelaskan pada tabel 2.1.

Tabel 2.1 perbandingan fitur storage engine MySQL

Feature	MyISAM	Memory	InnoDB	Archive	NBD
Storage	256TB	RAM	64TB	None	384EB
Transaction	No	No	Yes	No	Yes
Penguncian granulity	Table	table	row	row	Row
Clustered index	Tidak	Tidak	Ya	Tidak	Tidak
Compressed data	Ya	Tidak	Ya	Ya	Tidak
Encrypted data	Ya	Ya	Ya	Ya	Ya
Back-up/point-in-time recovery	Ya	Ya	Ya	Ya	Ya
Foreign key	Tidak	Tidak	Ya	Tidak	Tidak
Cluster database	Tidak	Tidak	Tidak	Tidak	Ya

*keterangan : Ya = mempunyai fitur tersebut, tidak = tidak mempunyai fitur tersebut

2.6 *Entreprise Application Integration* (EAI)

Enterprise Application Integration merupakan penggunaan prinsip arsitektur perangkat lunak dan sistem komputer untuk mengintegrasikan kumpulan aplikasi pada organisasi.

Pada aplikasi perusahaan yang tidak menggunakan aplikasi EAI, integrasi secara umum menggunakan metode *point to point integration*, yaitu menggunakan aplikasi

penengah (*middleware*) khusus untuk melakukan integrasi pada bagian-bagian tertentu dalam aplikasi, tetapi jika aplikasi lain ingin dihubungkan dengan metode ini, maka proses rekayasa ulangnya membutuhkan waktu yang lama, sedangkan pada EAI aplikasi dari semua bahasa pemrograman dapat dihubungkan dengan *Enterprise Service Bus (*ESB).

Tujuan diterapkannya EAI adalah

1. Integrasi data/informasi
2. Integrasi proses
3. Memungkinkan aplikasi yang berjalan *multi-vendor*
4. *Interface* yang mirip

2.7 *Testing*

Testing adalah proses pemeriksaan program dengan tujuan tertentu dalam menemukan kesalahan sebelum diserahkan ke pengguna. Hasil testing menentukan ketahanan sebuah program. Ada 2 teknik testing yang digunakan dalam pengembangan perangkat lunak yaitu

- **White Box**

White box testing secara umum dideskripsikan sebagai pengujian yang dilakukan seseorang yang mengetahui proses berdasarkan *Source Code* atau desain UML. Kasus uji dipilih untuk menguji bagian-bagian dari kode tersebut untuk menemukan masalah-masalah yang ada seperti pada cabang-cabang, perulangan dan batasan-batasan.

- **Black Box**

Black Box testing juga diikenal sebagai uji spesifikasi dan fungsionalitas. Suatu unit atau program diuji tanpa adanya pengetahuan tentang struktur internal dari *source codenya*. Maka itu pengujian ini bisa dan harus dilakukan oleh orang lain selain *programmer* yang membuat program

tersebut. Penguji memperlakukan kode program sebagai kotak hitam yang tidak bisa kita lihat dalamnya. Kasus uji dipilih berdasarkan sifat-sifat kode yang tampak dari luar yang dispesifikasikan pada dokumentasi program. Penguji hanya tertarik dengan apa yang dilakukan oleh kode tersebut, bukan bagaimana kode tersebut bekerja. Kode diuji dengan menginputkan data ke dalam kotak hitam dan memeriksa apakah outputnya sesuai dengan apa yang diperkirakan.

Pada proses code testing, dibedakan menjadi 3 fase:

1. *Unit Testing*
   Unit Testing berfokus pada usaha verifikasi pada unit terkecil dari program. *Unit Testing* dapat dilakukan dengan sesuatu yang sederhana seperti langkah melalui kode debugger; modern aplikasi termasuk penggunaan tes kerangka seperti SimpleTEST.

2. *Integration Testing*
   Integration Testing menguji unit yang telah diuji secara tunggal bekerja secara baik pula setelah digabungkan pada sistem. Saat sebuah unit telah berhasil menjalani tahap *Unit Testing*, mereka akan digabungkan ke dalam suatu grup logis yang koheren untuk diuji kembali. Misalnya beberapa unit digabung untuk membuat sebuah subsistem untuk diuji. Saat subsistem tersebut berhasil bekerja dengan baik maka akan dilanjutkan dengan menggabungkannya dengan subsistem yang lain dan seterusnya sampai membentuk suatu sistem utuh yang teruji.
   Hal yang diperhatikan pada *Integration Testing* yaitu memeriksa semua unit untuk dapat bekerja bersama dengan baik. Penguji lebih mengkonsentrasikan pada interaksi unit daripada fungsionalitasnya.

3. *System Testing*

Pengujian yang lengkap yang dilakukan pada perangkat lunak atau perangkat keras, untuk mengevaluasi sistem yang terintegrasi dengan tujuan tertentu. *System Testing* masuk ke dalam ruang lingkup black-box-testing. Oleh karena itu, tidak diperlukan pengetahuan tentang desain kode atau logika. Tester hanya perlu mengetahui *input* dan *output* yang bagaimana yang dikehendaki dari sistem.

BAB III
METODOLOGI

Metodologi atau tahapan pengerjaan merupakan hal yang sangat diperlukan dalam melakukan suatu penelitian, hal ini berlaku juga dalam pengerjaan tugas akhir. Metodologi diperlukan sebagai kerangka dan panduan proses pengerjaan tugas akhir, sehingga rangkaian pengerjaan tugas akhir dapat dilakukan secara terarah, teratur, dan sistematis. Adapun langkah-langkah pengerjaan tugas akhir yang dilakukan oleh penulis dapat dilihat pada gambar 3.1 dengan penjelasan sebagai berikut.

3.1 Pemahaman Informasi

Pada tahapan pertama, dilakukan pemahaman informasi untuk mengumpulkan informasi yang dibutuhkan dalam pembangunan modul Apotek. Hal yang perlu dilakukan peemahaman informasi adalah:

3.1.1 *Review* dokumen SIRST berbasis SOA

Mempelajari dan memahami informasi SKPL dan DPPL untuk modul Apotek mengenai spesifikasi kebutuhannya, desain UML, desain basis data, desain antarmuka dan fungsi-fungsi yang tercakup. Dalam review dokumen SIRST versi 2, juga termasuk di dalamnya evaluasi

3.1.2 *Review framework* SIRST berbasis SOA

Mempelajari *framework* mempelajari struktur dan proses dari *framework*. Informasi yang diperoleh dari review dokumen ini sebagai bekal dalam pembangunan modul Apotek SIRST.

3.2 Pembangunan Modul Apotek

Tahapan selanjutnya, langsung beranjak pada pembangunan modul Apotek sesuai dengan hasil dari review dokumen SIRST berbasis SOA.

3.2.1 *Code Convention*

Code Convention dilakukan untuk menyamakan persepsi dan aturan-aturan penulisan kode antara *programmer framework* OHIS dengan *programmer* modul SIRST berbasis SOA, agar penulisan memiliki standar yang baku untuk semua modul sehingga proses pengintegrasiannya menjadi lebih mudah.

3.2.2 Pembuatan tabel dan relasi basis data

Sebelum masuk pada tahap pembangunan modul apotek, pembuatan tabel dan relasi antar tabel yang disesuaikan dengan desain basis data pada DPPL SIRST berbasis SOA. Relasi antar tabel diperlukan untuk menjaga keintegritasan sistem dan konsistensi data.

3.2.3 Pembuatan *form* (antarmuka)

Form antarmuka dibuat terlebih dahulu, sebelum pembuatan servis. *Form* dibuat sesuai dengan spesifikasi yang telah ditentukan dalam DPPL SIRST berbasis SOA

3.2.4 Pembuatan servis dasar ambil data

Servis ambil data adalah servis-servis yang menampilkan hasil *query SELECT* dari basis data. Pembuatan seluruh servis ambil data dari basis data diperuntukkan bagi modul irna. Servis-servis ini akan disesuaikan dengan SKPL dan DPPL berbasis SOA.

3.2.5 Pembuatan servis dasar ubah data

Servis ubah data adalah servis-servis yang menjalankan hasil *query UPDATE* yang perubahannya tersimpan di basis data. Pembuatan seluruh servis ubah data dari basis data diperuntukkan bagi modul irna. Servis-servis ini akan disesuaikan dengan SKPL dan DPPL berbasis SOA.

3.2.6 Pembuatan servis dasar tambah data

Servis tambah data adalah servis-servis yang menjalankan hasil *query INSERT INTO* di mana penambahan data tersimpan di basis data. Pembuatan semua fungsi untuk menambah data dari basis data diperuntukkan bagi modul irna. Servis-servis ini akan disesuaikan dengan SKPL dan DPPL berbasis SOA.

3.2.7 *Testing*

Testing dilakukan secara bersamaan dengan semua langkah-langkah pembangunan yang disebutkan sebelumnya. Bagian ini dibagi menjadi 3 macam *testing*, yaitu :

1. *Testing* basis data
   Testing basis data menggunakan tools yaitu Power Designer untuk membantu evaluasi dan validasi desainnya.
2. *Testing form*
   Sedangkan *testing form* dilakukan secara manual dengan cara membandingkan langsung *form* aplikasi dengan spesifikasi desainnya.
3. *Testing* fungsi
   Testing fungsi dilakukan setiap fungsi selesai ditulis ke dalam bentuk kode program, agar setiap kegagalan fungsi langsung terdeteksi dan dapat diperbaiki secepatnya. *Testing* fungsi diuji menggunakan aplikasi kode debugger disertai *testing* manual.

3.3 Pengintegrasian dan Validasi Modul Apotek

Agar SIRST berbasis SOA dapat seutuhnya siap diimplementasikan, perlu adanya pengintegrasian modul Apotek dengan modul lainnya yaitu modul Manajemen dan IRNA.

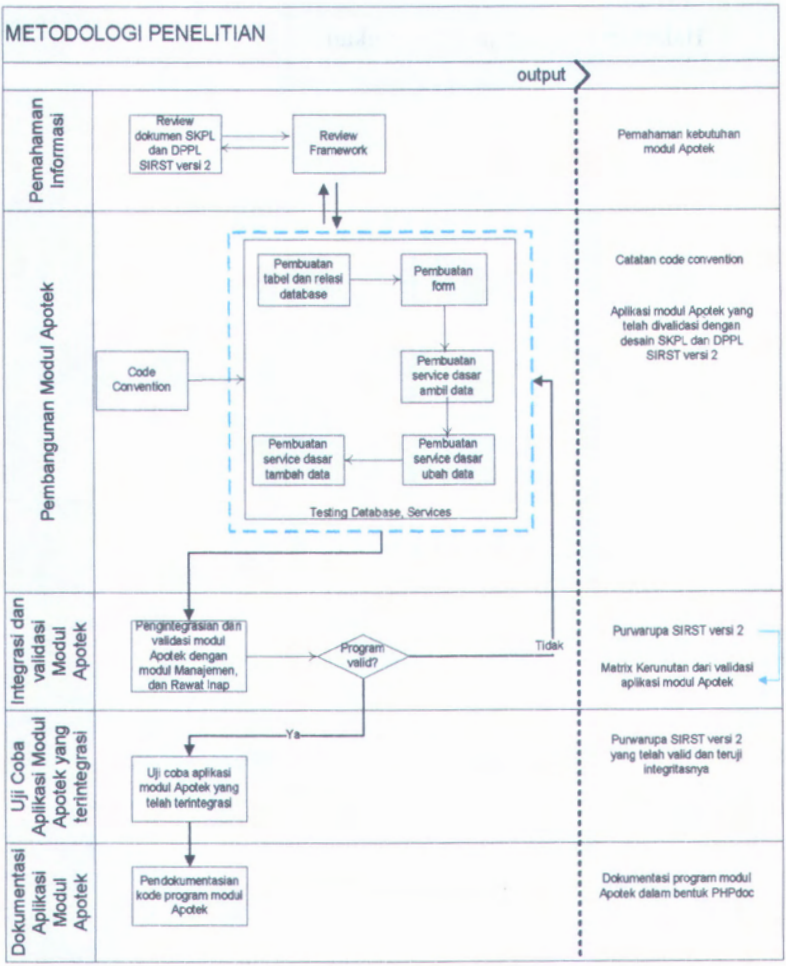
Validasi program dilaksanakan oleh perancang sistem dan *programmer* setelah program modul Apotek telah selesai dan terintegrasi dengan modul lainnya. Validasi bertujuan untuk mengetahui kebenaran aplikasi apakah telah mencakup semua spesifikasi kebutuhan yang tertulis dalam SKPL dan DPPL SIRST berbasis SOA. Matriks kerunutan disusun sebagai bukti validasi aplikasi dengan desainnya.

3.4 Uji Coba Modul Apotek yang Terintegrasi

Uji Coba pada hasil program ini akan menggunakan metode uji fungsional. Uji coba hanya meliputi uji coba terhadap *cross-modul service*, untuk membuktikan proses integrasi yang telah dilakukan berhasil.

3.5 Dokumentasi Aplikasi Modul Apotek

Dokumentasi aplikasi modul Apotek dibuat untuk mempermudah mempelajari kode aplikasinya. Dokumentasi ini ditujukan untuk *programmer* maupun pengguna akhir. Dokumentasi kode program untuk PHP dilakukan dengan membuat PHPDoc.



Gambar 3.1 Metodologi pengerjaan tugas akhir

Halaman ini sengaja dikosongkan.

BAB IV
ANALISA DAN PEMROGRAMAN

Sebelum dilakukan pembuatan servis modul manajemen, perlu dilakukan pemahaman dan anlisis mengenai informasi yang didapat dari dokumen dan *framework* OHIS agar servis yang dibuat benar-benar sesuai kebtuhan dan kelayakan. Bab ini akan membahas pemahaman informasi dari dokumen dan *framework*, kemudian hasil *code convention*, hasil perbaikan desain basis data dari dokumen, dan pembuatan servis manajemen.

4.1 Pemahaman Informasi

Pada sub bab ini akan dijabarkan hasil dari pemahaman informasi yang terdiri dari pemahaman DPPL SIRST rilis 2 dan framework OHIS yang menjadi dasar dalam pembuatan servis modul apotek.

4.1.1 *Review* dokumen SIRST berbasis SOA

Penggalian informasi dalam pembangunan modul Apotek diawali dengan review dokumen SKPL dan DPPL SIRST versi 2. Dalam apotek terdapat empat proses bisnis yang utama, yaitu

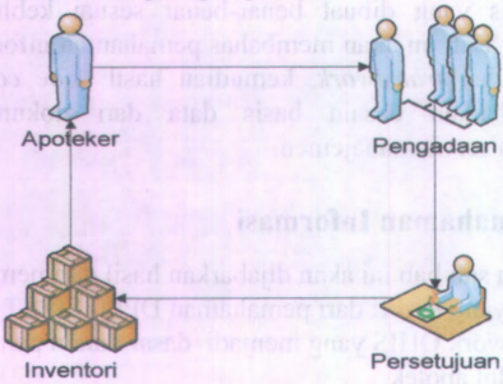
a. Pembelian obat

Gambar 4.1 Model proses pembelian obat

Gambar 4.1 diatas menunjukkan proses pembelian obat di apotek. Pertama-tama konsumen yang ingin membeli obat

akan memberikan resep maupun daftar obat yang akan dibeli kepada apoteker dan dilanjutkan dengan melakukan pembayaran di kasir.Setelah obat selesai diracik konsumen menunjukkan bukti pembayaran untuk mengambil pesanan obat tersebut.

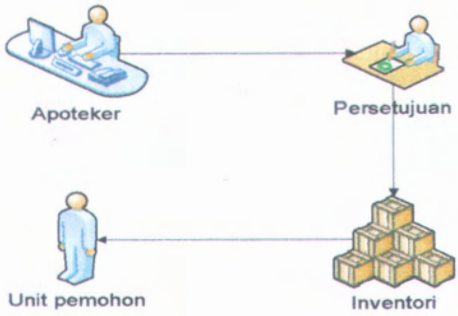
b. Permohonan barang ke pengadaan



Gambar 4.2 Model proses permohonan barang ke pengadaan

Gambar 4.2 menggambarkan proses permohonan barang ke pengadaan. Dimulai dengan pegawai apotek melakukan permohonan barang ke pengadaan. Kemudian apabila permohonan barang tersebut disetujui maka barang tersebut akan segera dikirimkan ke apotek.

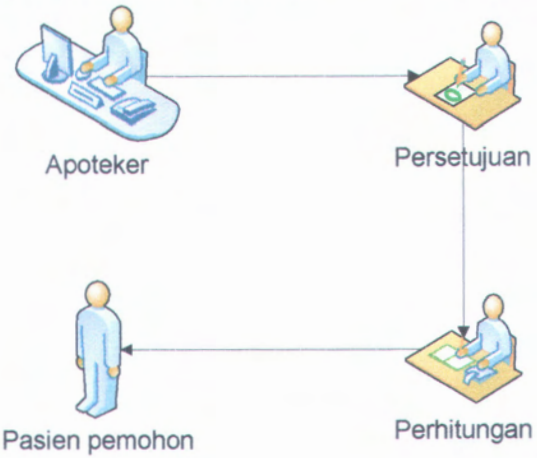
c. Permohonan mutasi barang



Gambar 4.3 Model proses permohonan mutasi obat

Gambar 4.3 menggambarkan proses permohonan mutasi barang oleh unit lain. Diawali dengan apoteker akan melihat apakah ada permohonan mutasi obat dari unit lain. Jika ada permohonan, maka permohonan tersebut akan dipertimbangkan dan apabila permohonan tersebut disetujui maka akan dilaksanakan proses mutasi obat ke unit pemohon tersebut

d. Permohonan retur obat pasien



Gambar 4.4 Model proses permohonan retur obat pasien

Gambar 4.4 diatas menunjukkan proses permohonan retur obat pasien oleh pasien. Diawali dengan apoteker akan melihat apakah ada permohonan retur obat dari pasien. Jika ada permohonan dan permohonan tersebut disetujui maka akan dilakukan perhitungan dan mengembalikan sisa pembayaran ke pasien pemohon

Dalam hasil *review* dokumen SKPL dan DPPL didapatkan informasi mengenai servis-servis yang terdapat dalam modul Apotek. Beberapa contoh servis dapat dilihat pada tabel 4.1, selengkapnya dapat dilihat pada lampiran C.

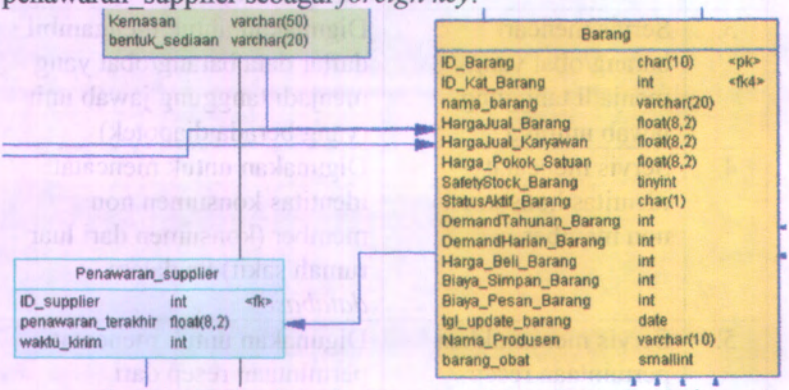
Di dalam DPPL SIRST Rilis 2 terdapat dua jenis form yang digunakan, yaitu simple form dan kompleks form. Simple form yang dimaksud dalam dokumen adalah form yang terdiri dari satu macam fungsi, yaitu input data saja atau edit data saja. Sedangkan Kompleks form terdiri dari lebih dari satu fungsi, misal update, lihat data dan tambah data.

Tabel 4.1 Daftar servis pada modul apotek

No	Nama servis	Fungsi
1.	Servis menampilkan daftar barang dan obat	Digunakan untuk mengambil daftar data barang dan obat
2.	Servis menampilkan daftar barang/obat di unit yang stoknya minimal	Digunakan untuk mengambil daftar data barang/obat di unit yang stoknya minimum (kurang atau sama dengan safety stock)
3.	Servis mencari barang/obat yang menjadi tanggung jawab unitnya	Digunakan untuk mengambil daftar data barang/obat yang menjadi tanggung jawab unit (yang berada diapotek)
4.	Servis mencatat identitas konsumen non member	Digunakan untuk mencatat identitas konsumen non member (konsumen dari luar rumah sakit) ke dalam *database*
5.	Servis memasukkan permintaan resep konsumen yang bukan dari poliklinik dan rawat inap	Digunakan untuk mencatat permintaan resep dari konsumen yang bukan dari poliklinik dan rawat inap rumah sakit
6.	Servis menampilkan resep dari poliklinik dan rawat inap	Digunakan untuk menampilkan resep dari poliklinik dan rawat inap rumah sakit
7.	Servis mengubah-ubah quantity obat yang diminta	Digunakan untuk meng-*edit* (mengubah) data jumlah obat dari resep sesuai dengan permintaan konsumen

Selain didapatkan informasi mengenai proses bisnis dan daftar servis, dalam desain basis data yang terdapat dalam DPPL rilis 2 ditemukan pula kesalahan yang cukup mendasar

sehingga diperlukan untuk dilakukan perbaikan. Kesalahan dalam desain basis data yang terdapat dalam DPPL SIRST rilis 2 diantaranya adalah tabel yang terbentuk akibat relasi *many to many* tidak memiliki *identifier*, kesalahan *cardinality constraint* dan tabel yang belum memenuhi standar normalisasi 3NF. Gambar 4.5 menunjukkan kesalahan dalam melakukan relasi, yaitu tidak adanya *identifier* penghubung tabel. Perbaikan yang dilakukan dari desain tabel tersebut adalah menambahkan id_barang pada tabel penawaran_supplier sebagai *foreign key*.



Gambar 4.5 Desain basis data yang salah pada aspek relasi

Contoh lain perbaikan relasi ditunjukkan oleh tabel pendaftaran_rawat_inap yang seharusnya menyimpan referensi entitas pasien secara unik, tetapi yang dijadikan *foreign key* adalah nama pasien yang bukan *primary key* dari tabel pasien. Perbaikan tabel pendaftaran_rawat_inap dilakukan dengan cara menghapus atribut 'NAMA_PASIEN' menjadi 'ID_PASIEN' seperti ditunjukkan gambar 4.6 dan 4.7

Field	Type	Collation	Attributes	Null	Default	Extra
ID_ANTRIAN_IRNA	int(11)			No	None	
TGL_PENDAFTARA_IRNA	date			No	None	
JAM_KEDATANGAN	time			No	None	
NAMA_PASIEN	varchar(30)	latin1_swedish_ci		No	None	
LOG_DATE	date			No	None	
SHIFT	varchar(30)	latin1_swedish_ci		No	None	
STATUS_LAPORAN	char(1)	latin1_swedish_ci		No	None	

Gambar 4.6 Tabel awal pendaftaran_rawat_inap

Field	Type	Collation	Attributes	Null	Default	Extra
ID_PENDAFTARAN_IRNA	int(11)			No	None	auto_increment
ID_SHIFT	int(11)			No	None	
ID_PASIEN	int(11)			No	None	
TGL_PENDAFTARAN_IRNA	date			Yes	NULL	
JAM_KEDATANGAN	time			Yes	NULL	
LOG_DATE_DAFTAR	varchar(21)	latin1_swedish_ci		Yes	NULL	

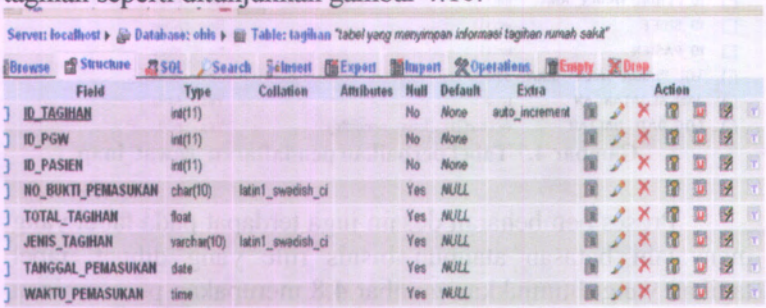
Gambar 4.7 Tabel perbaikan pendaftaran_rawat_inap

Proses pembenaran desain juga terdapat pada tabel yang menyalahi batasan ataupun bisnis rule yang dibuat, tabel tagihan yang ditunjukkan gambar 4.8 merupakan pencerminan aspek tersebut.

Field	Type	Collation	Attributes	Null	Default	Extra
ID_PEMASUKAN_LUNAS	int(11)			No	None	
ID_PGW	int(11)			Yes	NULL	
NOBUKTI_PEMASUKAN	int(11)			Yes	NULL	
NAMA_ITEM_TAGIHAN	varchar(30)	latin1_swedish_ci		No	None	
JUMLAH_ITEM_TAGIHAN	int(11)			No	None	
HARGA_SATUAN	float(8,2)			No	None	
TOTAL_HARGA	float(8,2)			No	None	
JENIS_TAGIHAN	varchar(10)	latin1_swedish_ci		No	None	
STATUS_PEMASUKAN	char(1)	latin1_swedish_ci		Yes	NULL	
ID_IRNA	int(11)			Yes	NULL	
ID_IRJA	int(11)			Yes	NULL	
ID_APOTEK_PENJUALAN	int(11)			Yes	NULL	
ID_APOTEK_RETUR_PASIEN	int(11)			Yes	NULL	
ID_PENDAFTARAN	int(11)			No	None	
TGL_PEMASUKAN	date			No	None	
WT_PEMASUKAN	datetime			Yes	NULL	

Gambar 4.8 Tabel tagihan yang menyalahi business rule

Hubungan antara tabel tagihan dengan tagihan_irna, tagihan_irja, serta tagihan_pendaftaran adalah *many to one*, sehingga bukan *foreign key* dari tabel tagihan_irna, tagihan_irja, serta tagihan_pendaftaran yang dimasukkan ke tabel tagihan melainkan sebaliknya. Tabel tagihan seharusnya tidak mendapatkan atribut referensi dari tabel lain tetapi justru harus menjadikan ID_tagihan referensi bagi tabel lain seperti ditunjukkan gambar 4.9. Tabel tagihan_rawat_inap mendapatkan atribut tambahan, yaitu id_tagihan dari tabel tagihan seperti ditunjukkan gambar 4.10.


Server: localhost ▸ Database: obis ▸ Table: tagihan "tabel yang menyimpan informasi tagihan rumah sakit"

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Empty | Drop

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
]	ID_TAGIHAN	int(11)			No	None	auto_increment	
]	ID_PGW	int(11)			No	None		
]	ID_PASIEN	int(11)			No	None		
]	NO_BUKTI_PEMASUKAN	char(10)	latin1_swedish_ci		Yes	NULL		
]	TOTAL_TAGIHAN	float			Yes	NULL		
]	JENIS_TAGIHAN	varchar(10)	latin1_swedish_ci		Yes	NULL		
]	TANGGAL_PEMASUKAN	date			Yes	NULL		
]	WAKTU_PEMASUKAN	time			Yes	NULL		

Gambar 4.9 Tabel tagihan setelah divalidasi sesuai business rule

Field	Type	Collation	Attributes	Null	Default
ID_TAGIHAN_IRNA	int(11)			No	None
DEPOSIT	float(8,2)			No	None
TOTAL_BIAYA_DOKTER	float(8,2)			No	None
TOTAL_PAKAI_OBAT	float(8,2)			No	None
TOTAL_BIAYA_TINDAKAN	float(8,2)			No	None
TOTAL_BIAYA_PEMAKAIHAN_KAMAR	float(8,2)			No	None
TOTAL_TAGIHAN	float(8,2)			No	None
STATUS_TAGIHAN	varchar(5)	latin1_swedish_ci		No	None

Check All / Uncheck All With selected:

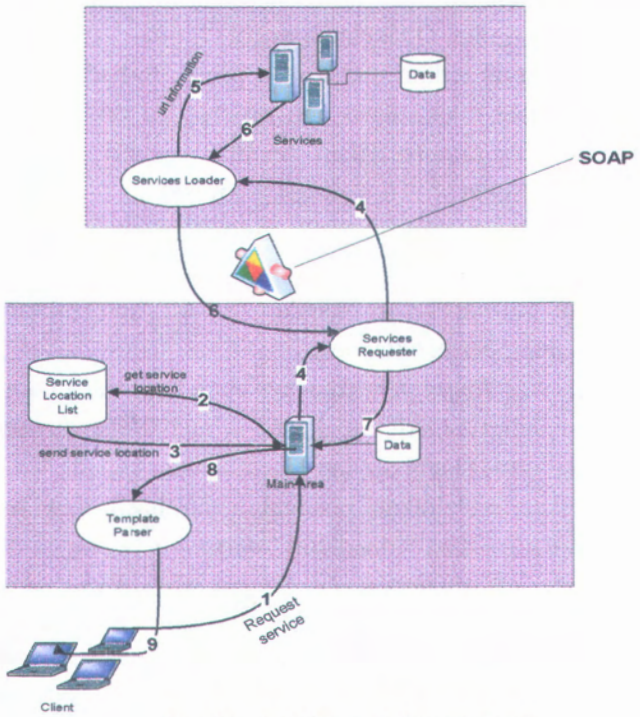
Field	Type	Collation	Attributes	Null	Default	Extra
ID_TAGIHAN_IRNA	int(11)			No	None	auto_increment
ID_TAGIHAN	int(11)			No	None	
DEPOSIT	float(8,2)			No	None	
TOTAL_BIAYA_DOKTER	float(8,2)			No	None	
TOTAL_PAKAI_OBAT	float(8,2)			No	None	
TOTAL_BIAYA_TINDAKAN	float(8,2)			No	None	
TOTAL_BIAYA_PEMAKAIHAN_KAMAR	float(8,2)			No	None	
TOTAL_TAGIHAN	float(8,2)			No	None	
STATUS_TAGIHAN	varchar(5)	latin1_swedish_ci		No	None	

Gambar 4.10 Tabel tagihan_rawat_inap sebelum dan setelah divalidasi

4.1.2 *Review* framework OHIS

Pembangunan modul manajemen pada SIRST berbasis SOA menggunakan *framework* OHIS yang didesain dan dirancang memiliki karakteristik SOA. Penjelasan tentang *framework* OHIS tidak akan detail kepada operasinya karena akan dipatenkan, untuk mengetahui operasi detail dari *framework* akan dijelaskan di OHIS manual yang segera diterbitkan, sedangkan class diagram *framework* bisa dilihat di lampiran G.

Secara dasar, framework terdiri dari dua bagian utama, yaitu bagian *main* dan servis. Pendekatan secara umum framework OHIS ditunjukkan gambar 4.11 sebagaimana berikut,



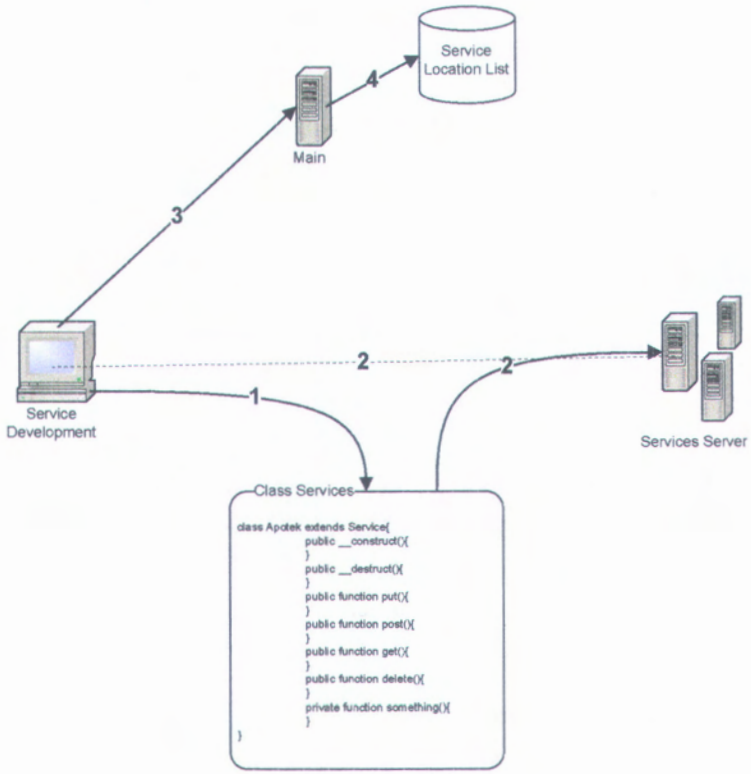
Gambar 4.11 Bagan hubungan teori

Keterangan pada gambar 4.11 :

1. *Client* Melakukan *Request* Halaman (servis) kepada *Main*.
2. *Main* akan meminta lokasi servis kepada unit "*Service Location List*"
3. Unit "*Service Location List*" akan memberikan lokasi servis yang tepat kepada *Main*.
4. *Main* akan memberikan URL dari servis dengan standar SOAP kepada *services requester* dan akan diteruskan kepada *services loader*
5. Setelah *Services Loader* menerima URL, maka URL akan dipecah – pecah menjadi kandidat objek yang akan di *load* (mengacu pada contoh URL sebelumnya, misal : Apotek.class.php, Obat.class.php)
6. Setelah Servis terkait melakukan proses terhadap *request* yang diminta, maka hasil dari process itu dikirimkan ke unit *Services Loader* dalam bentuk Asosiatif Array atau biasa disebut *Dictionary* (misal : $x = (nama=>"didit",pekerjaan=>"ngoding")), lalu unit *Services Loader* akan merubah *dictionary* tersebut kedalam format XML standart SOAP dan dikirmkan kepada unit *Services Requester*.
7. Unit *Service Requester* akan merubah data dalam format XML standart SOAP ke dalam sebuah *data dictionary* (asosiatif array) kembali dan dikembalikan kepada *Main*.
8. *Main* akan melakukan penggabungan hasil dari servis dengan data – data "mandatori" dari *main* sehingga dapat dibentuk *data dictionary* baru yang akan dikirimkan ke dalam unit *template parser*.
9. Unit *Template Parser* akan menggabungkan *data dictionary* ke dalam aturan *template* yang telah

didefinisikan menghasilkan dokumen HTML, dan sehingga HTML dapat dikirimkan ke *Client*.

Sedangkan dari sudut pandang pengembangan modul yang mengikuti framework OHIS ditunjukkan pada gambar 4.12 sebagaimana berikut



Gambar 4.12 Aliran data pada framework

Keterangan pada gambar 4-12 :

1. Pengembang telah membuat kelas servis terhadap satu servis (misal : apotek)
2. File kelas servis tersebut di*upload* dalam *server* servis dengan mengetahui alamat URL dari servis tersebut setelah dilakukan *upload*(garis putus-putus)
3. Melakukan registrasi servis ke dalam main yang akan diteruskan *main* untuk disimpan pada unit *Service Location List.*

Dalam *framework* OHIS, terdapat beberapa fitur-fitur yang belum diselesaikan dalam mencukupi kebutuhan pengembangan modul apotek, diantaranya adalah :

- pembuatan *form* input data,
- pembagian hak akses,
- login,
- *passing parameter*,
- pembuatan *form* ubah data,
- pembuatan *form* hapus data

Sedangkan fitur tampilan data yang bisa digunakan adalah grid yang berupa tabel. Secara fungsional *framework* OHIS dapat melakukan proses tambah data, ubah data, maupun hapus data tetapi tampilan *form*nya masih belum tercakupi.

4.2 *Code Convention*

Pada *code convention* ini, dibuat kesepakatan mengenai penulisan code program, yaitu :

1. Penulisan kelas diawali dengan huruf kapital.
2. Penulisan metode dengan menggunakan huruf kecil semua dan tiap kata dipisah dengan garis bawah.
3. Penulisan nama variabel harus kecil semua.

4. Penulisan nama variabel global dan final harus menggunakan huruf kapital semuanya.

Tujuan *code convention* adalah menyamakan persepsi dan membuat standart yang dapat dimengerti oleh para developer dalam memahami kode program. Gambar 4.14 menunjukkan *skeleton* program yang telah memenuhi *code convention*.

```
class Apotek {

    private $db;

    /**
     * Constructor
     *
     */


    public function __construct(){

    public function _up(){

    public function _down(){

     /**
     * 3.2.2.2 Spesifikasi Proses Form Tampil Data Resep (FRM.APO.02)
     *Form yang menampilkan data resep
     * @param
     * @return array
     */
    public function tampil_data_resep(){

    /**
     * 3.2.2.3 Spesifikasi Proses Form Tambah Resep (FRM.APO.03)
     *Form yang melakukan penambahan data resep
     * @param
     * @return array
```

Gambar 4.13 Contoh kode program sesuai *Code Convention*

4.3 Pembuatan Tabel dan Relasi

Tabel dan relasi pada DPPL rilis 2 seperti yang telah diulas pada sub bab 4.1 masih memiliki banyak kesalahan. Perbaikan pada desain basis data ditunjukkan pada DPPL SIRST rilis 3. Perubahan-perubahan yang dilakukan pada tabel fisik basis data dapat dilihat pada DPPL SIRST rilis 3 dan perbandingan model data konseptual berformat E/R + Merise.

Untuk daftar tabel lengkap yang telah didesain ulang dapat dilihat pada DPPL SIRST Rilis 3. Dari 76 tabel yang terdapat pada DPPL SIRST rilis 2 terjadi penghapusan 6 tabel dan penambahan 10 tabel sehingga jumlah tabel di DPPL rilis 3 sebanyak 80 tabel.

4.4 Pembuatan Servis Apotek

Framework OHIS menggunakan kelas untuk menyatakan modul dan fungsi untuk menyatakan tiap servis. Setiap servis atau layanan yang dibangun direpresentasikan sebagai *class* pada ohis/services/modules. Ada 3 macam fungsi dasar dalam SIRST berbasis SOA ini, yaitu :

1. Ambil data
2. Ubah data
3. Tambah data

Kode fungsi modul apotek secara keseluruhan dapat dilihat pada lampiran B. Pada bab ini akan dijelaskan struktur penulisan kode program beserta penjelasan kode program setiap fungsi dasar.

4.4.1 Struktur Penulisan Kode Program

Pengembangan program dengan menggunakan *framework* OHIS mengharuskan kode program mengikuti struktur yang distandarisasi oleh *framework* tersebut. Sub bab ini akan menjelaskan struktur penulisan kode program yang digunakan untuk membuat servis pada modul manajemen. Sub bab ini juga akan membahas sekilas penulisan kode program tanpa menggunakan *framework*. Struktur penulisan kode

program pada *framework* OHIS dituliskan sebagaimana berikut

```
1  class Apotek {
       private $db;

2  public function __construct()
   {
         $this->db = Factory::get_db();
   }

3  public function _up()
   {
       $resep = Array(
           'tampil_data_resep', 'tambah_resep',
   'ubah_data_resep', 'tambah_konsumen',
   'tampil_data_konsumen', 'penjualan_obat'
       );
       Register::functions_to_activity($resep,
   "Resep");
   }
4  public function _down()
   {
   }

5  public function tampil_data_resep(){
       $sth = $this->db->prepare("SELECT
   po.id_penjualan_obat, po.id_resep, r.id_pasien,
   p.nama_pasien, po.id_pgw, pg.nama_pgw,
   po.total_harga_obat, po.status_flow_resep FROM
   penjualan_obat po, resep r, pasien p, pegawai
   pg where po.id_resep=r.id_resep and r.id_pasien
   = p.id_pasien and po.id_pgw = pg.id_pgw");
       $sth->execute();
       $data = new DataSource();
       $data->set_type("Grid");
       $data->set_comp("Tabel Data Resep");
       $data->set_data($sth->fetchAll());
       return $data->export();
   }
```

Tiap baris kode program dapat dijelaskan sebagai berikut

1. Deklarasi kelas dan variabel $db yang akan digunakan untuk koneksi dengan basis data.

2. *Constructor* pada setiap kelas ditulis di awal. *Constructor* berguna untuk membentuk sebuah objek dari kelasnya. Di dalam *constructor* dipanggil *method Factory::get_db()* untuk memanggil basis datanya.
3. *Method Function_up* akan dibaca ketika sebuah *service* pertama kali di-*install* ke dalam sistem. Di dalam m*ethod* ini juga dilakukan inisialisasi semua fungsi-fungsi yang ada dikelompokkan ke aktifitas-aktifitas tertentu. Dengan menggunakan fungsi *Register::functions_to_activity()* pemetaan fungsi terhadap aktifitas akan disimpan ke dalam basis data.
4. *Method Function_down* akan dibaca ketika sebuah servis di-*uninstall* atau dihapus dari dalam sistem.
5. *Method daftar_inventori* adalah contoh penulisan fungsi untuk setiap servis apotek. Untuk fungsi selanjutnya dijelaskan pada sub bab berikutnya.

Penulisan kode program di atas menyimpulkan bahwa fungsi yang harus ada tiap kelas adalah *_construct*, *_up*, dan *_down*.

Jika dibandingkan dengan cara pembuatan kode secara konvensional yang dilakukan tanpa menggunakan *framework* OHIS, maka gambaran penulisan kodenya diperlihatkan seperti berikut

```
<?php
// Make a MySQL Connection
mysql_connect("localhost", "admin", "") or die(mysql_erro
r());
mysql_select_db("test") or die(mysql_error());

// Create a MySQL table in the selected database
mysql_query("CREATE TABLE products (
  id int(11) NOT NULL default '0',
  name varchar(60) NOT NULL default '',
  type varchar(30) NOT NULL default '',
  price decimal(10,2) NOT NULL default '0.00',
  PRIMARY KEY  (id)
);")or die(mysql_error());

mysql_query("
INSERT INTO products VALUES
```

```
('123451', 'Park''s Great Hits', 'Music', 19.99),
('123452', 'Silly Puddy', 'Toy', 3.99),
('123453', 'Playstation', 'Toy', 89.95),
('123454', 'Men''s T-Shirt', 'Clothing', 32.50),
('123455', 'Blouse', 'Clothing', 34.97),
('123456', 'Electronica 2002', 'Music', 3.99),
('123457', 'Country Tunes', 'Music', 21.55),
('123458', 'Watermelon', 'Food', 8.73);
")or die(mysql_error());

echo "Table Created!";

?>
```

Fungsi di atas tersebut merupakan fungsi pemanggilan data pada basis data MySQL, kemudian untuk menampilkan dalam bentuk tabel fungsi menggunakan JavaScriptnya sebagaimana berikut

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>create DOM table</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">

<style type="text/css">

#newtable{
border:2px solid #999;
font-family:verdana,arial,helvetica,sans-serif;
font-size:18px;
margin:auto;
}
#newtable td{
width:50px;
line-height:50px;
border:1px solid #000;
text-align:center;
}
</style>

<script type="text/javascript">
```

```
window.onload=function() {
makeTable();
}

function makeTable() {

row=new Array();
cell=new Array();

row_num=12; //edit this value to suit
cell_num=12; //edit this value to suit

tab=document.createElement('table');
tab.setAttribute('id','newtable');

tbo=document.createElement('tbody');

for(c=0;c<row_num;c++){
row[c]=document.createElement('tr');

for(k=0;k<cell_num;k++) {
cell[k]=document.createElement('td');
cont=document.createTextNode((c+1)*(k+1))
cell[k].appendChild(cont);
row[c].appendChild(cell[k]);
}
tbo.appendChild(row[c]);
}
tab.appendChild(tbo);
document.getElementById('mytable').appendChild(tab);
}
</script>

</head>
<body>

<div id="mytable"></div>

</body>
</html>
```

Gambaran penulisan kode program dengan menggunakan *framework* dan tanpa menggunakan *framework*. Hasil

perbandingan antara keduanya adalah penulisan kode program mengggunakan *framework* lebih mudah dibandingkan dengan penulisan kode program tanpa menggunakan *framework* OHIS.

4.4.2 Kode Program Servis Ambil Data

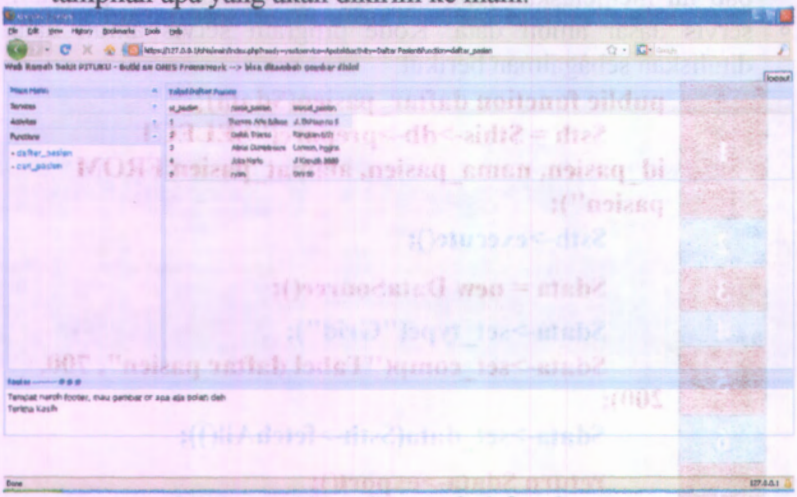
Masing-masing servis ambil data dalam *framework* direpresentasikan dengan fungsi yang berformat sama dan yang membedakan secara utuh adalah perintah SQL-nya. Sub bab ini menjelaskan tentang format penulisan kode program servis dasar ambil data. Kode program servis ambil data dituliskan sebagaiman berikut

```
1  public function daftar_pasien($data){
       $sth = $this->db->prepare("SELECT
   id_pasien, nama_pasien, alamat_pasien FROM
   pasien");
2      $sth->execute();
3      $data = new DataSource();
4      $data->set_type("Grid");
5      $data->set_comp("Tabel daftar pasien", 700,
   200);
6      $data->set_data($sth->fetchAll());
7      return $data->export();
   }
```

Tiap baris kode program dapat dijelaskan sebagai berikut

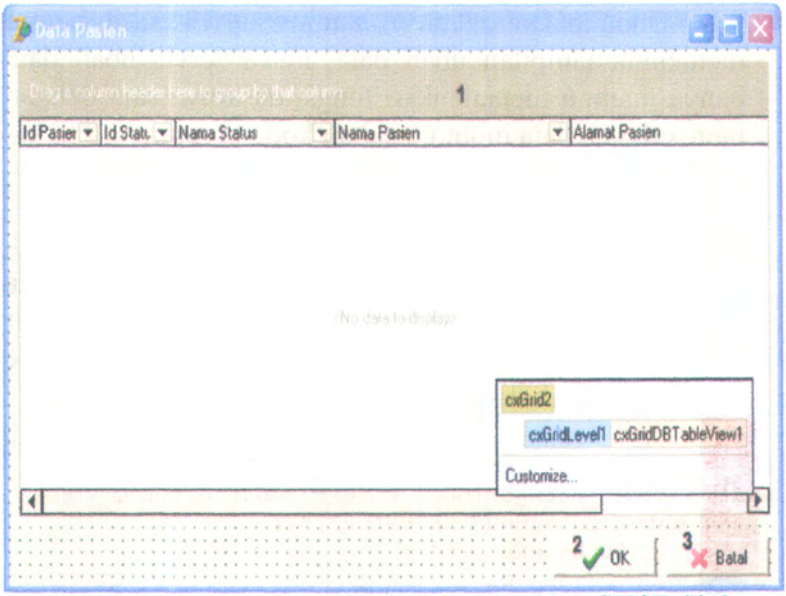
1. Fungsi *prepare()* merupakan fungsi yang berasal dari kelas PDO *built-in* dari PHP. Fungsi ini digunakan untuk mempersiapkan SQL script yang akan dijalankan dengan fungsi *execute()*.
2. Fungsi *execute()* merupakan fungsi yang berasal dari PHP PDO yang berfungsi untuk melakukan ekseskusi pernyataan SQL.
3. Kelas *DataSource* Membuat fungsi untuk melakukan proses pengambilan data dan penambahan data.

4. Fungsi *set_type* berguna untuk memberuikan tipe data yng akan ditampilkan dlm framework. Grid berarti tampilan yang ditunjukkan berupa tabel.
5. Fungsi *set_comp* digunakan untuk memberikan nama tabel dan memberikan ukuran lebar dan panjang dari tabel yang ditampilkan.
6. Fungsi *set_data* dugunakan untuk menampung data dalam array yang kemudian akan disimpan dalam bentuk SOAP.
7. Pengembalian fungsi ini adalah array data beserta template tampilan apa yang akan dikirim ke main.



Gambar 4.14 Tampilan servis daftar_barang

Tampilan dari servis daftar_pasien memang sedikit ada perbedaaan dengan desain yang terdapat dalam DPPL SIRST rilis 2 seperti ditunjukkan pada gambar dalam hal pewarnaan, *layout* dan *style*. Hal tersebut terjadi karena program mengikuti *framework* OHIS yang menggunakan standart tersendiri dalam menampilkan data, tetapi secara fungsi masih sama, yaitu menampilkan data dalam format grid.



Gambar 4.15. Tampilan servis daftar_pasien DPPL SIRST rilis 2

4.4.3 Kode Program Servis Ubah Data

Penulisan kode program servis ubah data tidak jauh berbeda. Secara struktur penulisan hampir sama tetapi berbeda pada perintah SQL-nya saja. Kode program ubah data dituliskan sebagaimana berikut

```
1  public function
   persetujuan_retur_obat_pasien(){
   $sth = $this->db->prepare("UPDATE
   `ohis`.`retur_obat_pasien` SET `STATUS` = 'S'
   WHERE `retur_obat_pasien`.`ID_RETUR_OBAT`
   ='".$data."');
2  $sth->execute();
```

Tiap baris kode program dapat dijelaskan sebagai berikut

1. Spesifikasi perintah SQL untuk proses *update* yang dijalankan.
2. Menjalankan perintah *update* pada pada baris satu menggunakan fungsi *bulit-in* pada PHP PDO.

Dikarenakan dalam *framework* OHIS masih belum memenuhi tampilan ubah data, maka servis ubah data hanya mampu menjalankan fungsi dasarnya saja dengan menyertakan data input ke dalam kode program.

4.4.4 Kode Program Servis Tambah Data

Penulisan kode program servis tambah data tidak jauh berbeda dengan penulisan pada servis ambil data maupun ubah data. Secara struktur penulisan hampir sama tetapi berbeda pada perintah SQL-nya saja. Kode program tambah data dituliskan sebagaimana berikut

```
1  public function tambah_data_kategori(){
       $sth = $this->db->prepare("insert into
   master_kategoriobat values('obat batuk', 'obat
   untuk menyembuhkan batuk')");
       $sth->execute();
2  $sth->execute();
```

Tiap baris kode program dapat dijelaskan sebagai berikut

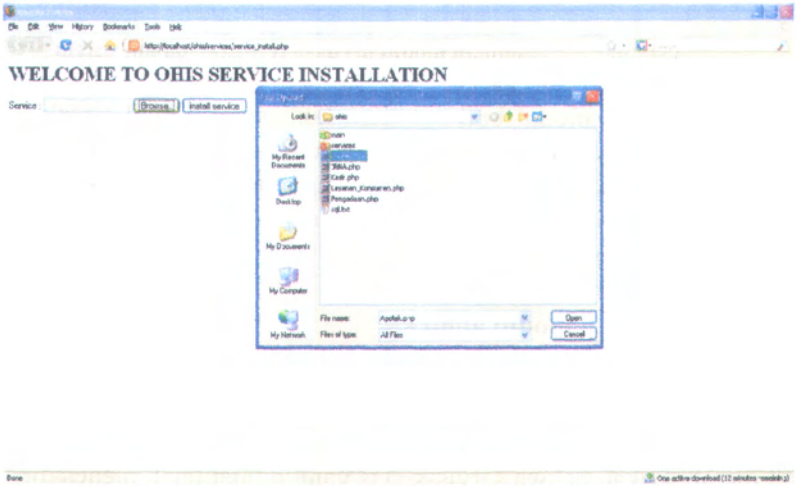
1. Spesifikasi perintah SQL untuk proses *insert* yang dijalankan.
2. Menjalankan perintah *insert* pada pada baris satu menggunakan fungsi *bulit-in* pada PHP PDO.

Dikarenakan dalam *framework* OHIS masih belum memenuhi tampilan tambah data, maka servis tambah data hanya mampu menjalankan fungsi dasarnya saja dengan menyertakan data input ke dalam kode program.

BAB V
VALIDASI DAN UJI COBA

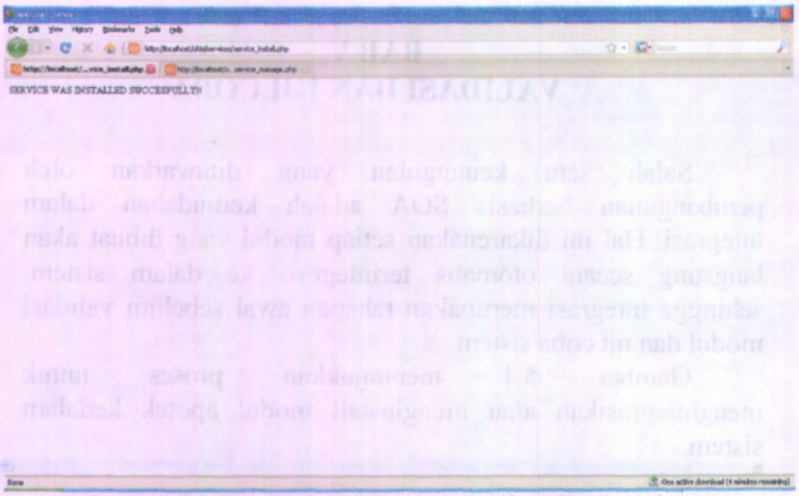
Salah satu keunggulan yang ditawarkan oleh pembangunan berbasis SOA adalah kemudahan dalam integrasi. Hal ini dikarenakan setiap modul yang dibuat akan langsung secara otomatis terintegrasi ke dalam sistem, sehingga integrasi merupakan tahapan awal sebelum validasi modul dan uji coba sistem.

Gambar 5.1 menunjukkan proses untuk mengintegrasikan atau menginstall modul apotek kedalam sistem.



Gambar 5.1 Tampilan proses instalasi modul apotek

Apabila modul berhasil terinstall akan muncul pesan seperti pada gambar 5.2 dibawah ini.



Gambar 5.2 Tampilan modul berhasil terinstall dalam sistem

Setelah modul berhasil terinstall, maka modul tersebut dapat divalidasi dan diuji coba apakah sudah sesuai dengan yang diharapkan. Cara yang digunakan untuk memvalidasi dan menguji coba modul dalam sistem akan dijelaskan pada sub bab-sub bab berikut.

5.1 Validasi modul apotek

Validasi ini bertujuan untuk mengetahui apakah modul yang dibuat telah sesuai dan mencakup semua kebutuhan yang tertulis dalam SKPL dan DPPL SIRST berbasis SOA. Untuk membuktikan bahwa servis-servis yang dibuat telah mencakup keseluruhan kebutuhan pada SKPL dan DPPL maka disusunlah matriks kerunutan yang dapat dilihat pada lampiran D. Sedangkan untuk memvalidasi apakah setiap fungsi sudah berjalan sesuai dengan yang diharapkan, digunakan metode *black box* dengan membuat *test case* untuk setiap servis yang ada. *Test case* dapat dilihat pada lampiran E. Dari *test case* yang telah dibuat, di dapatkan informasi bahwa setiap servis yang dibuat pada dasarnya sudah berjalan seperti yang

diharapkan pada SKPL dan DPPL, akan tetapi dikarenakan adanya fitur yang belum tercakup pada *framework* seperti halnya *form* tambah data, *form* ambil data, dan sebagainya, sebagaimana telah dijelaskan pada bab 4, menyebabkan dari 29 servis yang ada pada modul apotek, 18 servis masih belum dapat terselesaikan sepenuhnya. Untuk daftar besar penyelesaian tiap servis pada modul apotek dapat dilihat pada lampiran F.

5.2 Uji coba integrasi modul apotek

Setiap modul pada SIRST berbasis SOA dilakukan uji coba integrasi melalui tiga cara, yaitu pada saat instalasi, penghapusan dan pemanggilan servis pada modul yang lain.

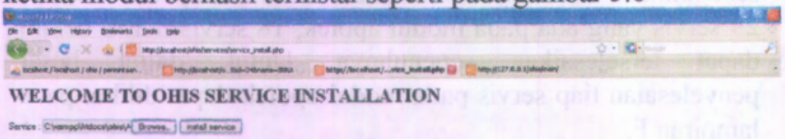
5.2.1 Uji coba integrasi pada saat instalasi service

Seperti yang telah dijelaskan sebelumnya bahwa salah satu keunggulan yang ditawarkan dari pengembangan berbasis SOA yaitu kemudahan integrasi pada saat instalasi modul. Pada sub bab ini akan dilakukan uji coba instalasi seluruh modul beserta penjelasan secara lebih terperinci mengenai proses integrasi pada saat instalasi dan contoh tampilannya pada purwarupa SIRST berbasis SOA.

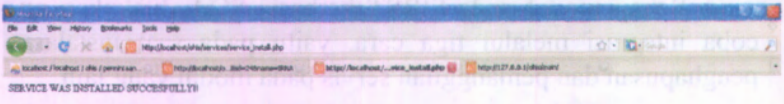
Pada gambar 5.4 dibawah ini menggambarkan kondisi pada saat belum ada modul yang terinstall pada sistem.

Gambar 5.3 Tampilan awal purwarupa SIRST berbasis SOA ketika belum ada modul yang terinstall.

Kemudian dilakukan proses instalasi modul apotek yang ditunjukkan pada gambar 5.5 beserta tampilan keberhasilan ketika modul berhasil terinstal seperti pada gambar 5.6

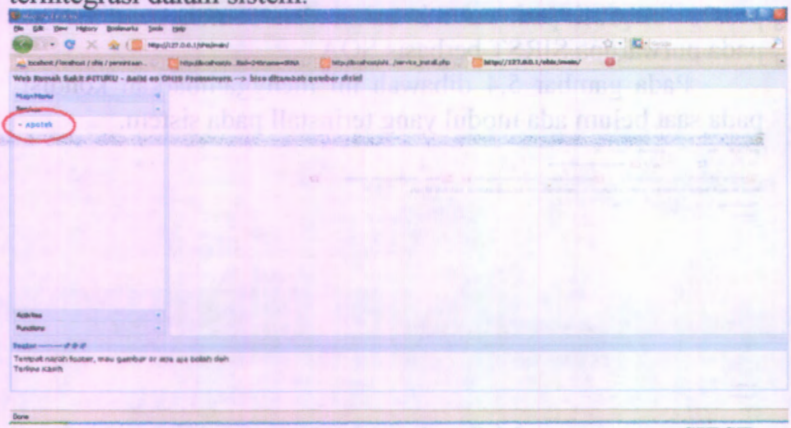


Gambar 5.4 Proses instalasi modul apotek.



Gambar 5.5 Proses instalasi modul apotek berhasil dilakukan.

Setelah modul berhasil terinstall maka secara otomatis modul tersebut telah terintegrasi kedalam sistem, seperti yang ditunjukkan pada gambar 5.6 bahwa modul apotek telah terintegrasi dalam sistem.



Gambar 5.6 Modul apotek telah terinstall dalam purwarupa SIRST berbasis SOA.

Kemudian dilakukan proses instalasi semua modul lainnya dengan cara yang sama sehingga didapatkan purwarupa SIRST berbasis SOA dengan seluruh modul yang telah terintergrasi seperti yang ditunjukkan pada gambar 5.7.

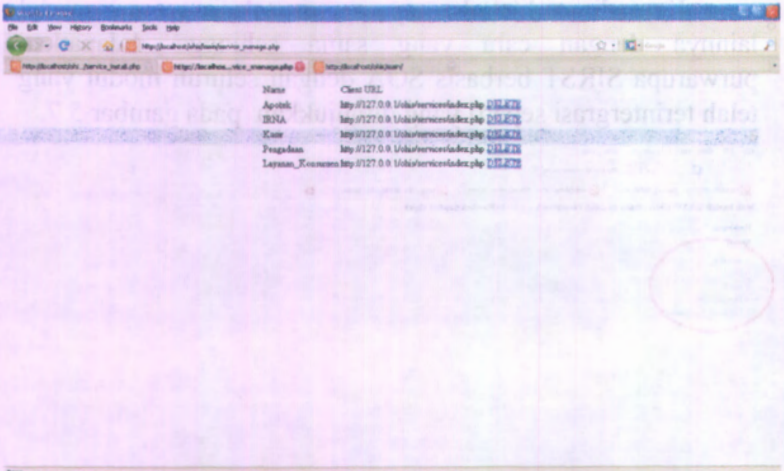
Gambar 5.7 Seluruh modul telah terinstall dalam purwarupa SIRST berbasis SOA

Dari langkah-langkah yang dilakukan diatas dapat disimpulkan bahwa setiap modul pada saat instalasi telah terintegrasi dengan baik.

5.2.2 Uji coba integrasi pada saat penghapusan servis

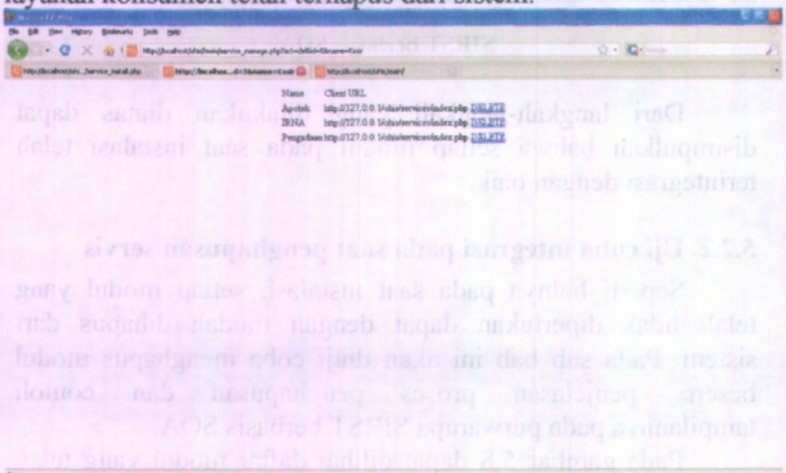
Seperti halnya pada saat instalasi, setiap modul yang telah tidak diperlukan dapat dengan mudah dihapus dari sistem. Pada sub bab ini akan diuji coba menghapus modul beserta penjelasan proses penghapusan dan contoh tampilannya pada purwarupa SIRST berbasis SOA.

Pada gambar 5.8 dapat dilihat daftar modul yang telah terinstall pada sistem dan apabila link 'DELETE' pada modul tertentu diklik maka modul tersebut akan dihapus dari sistem beserta basis data yang disertakan dalam kode programnya



Gambar 5.8 Daftar modul yang terinstal.

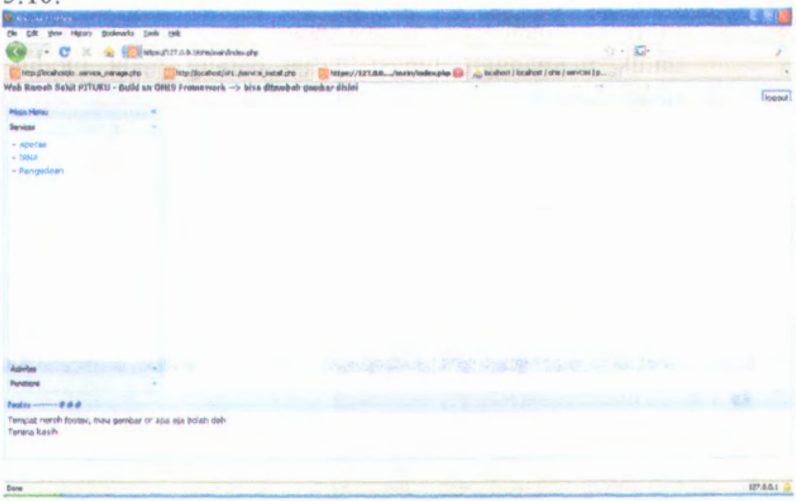
Pada gambar 5.9 dapat dilihat bahwa modul kasir dan layanan konsumen telah terhapus dari sistem.



Gambar 5.9 Modul kasir dan layanan konsumen telah terhapus dari purwarupa SIRST berbasis SOA.

Hal ini akan sama ketika purwarupa SIRST berbasis SOA dijalankan, modul kasir dan layanan konsumen juga

terhapus dari sistem seperti yang ditunjukkan pada gambat 5.10.



Gambar 5.10 Tampilan ketika modul kasir dan layanan konsumen telah terhapus dari purwarupa SIRST berbasis SOA

Dari langkah-langkah yang dilakukan diatas dapat disimpulkan bahwa setiap modul pada saat penghapusan telah terintegrasi dengan baik.

5.2.3 Uji coba integrasi antar modul

Setiap modul pastinya memiliki keterkaitan antara satu dengan yang lainnya. Sehingga apabila antara satu modul dan modul lainnya tidak terintegrasi dengan baik, akan menyebabkan terhambatnya bahkan mungkin terputusnya komunikasi dan interaksi tiap modul. Hal ini menyebabkan perlunya uji coba integrasi antar modul. Dalam melakukan uji coba integrasi antar modul digunakan cara *cross-functional* atau penggunaan kembali servis atau fungsi yang dimiliki oleh modul lain.

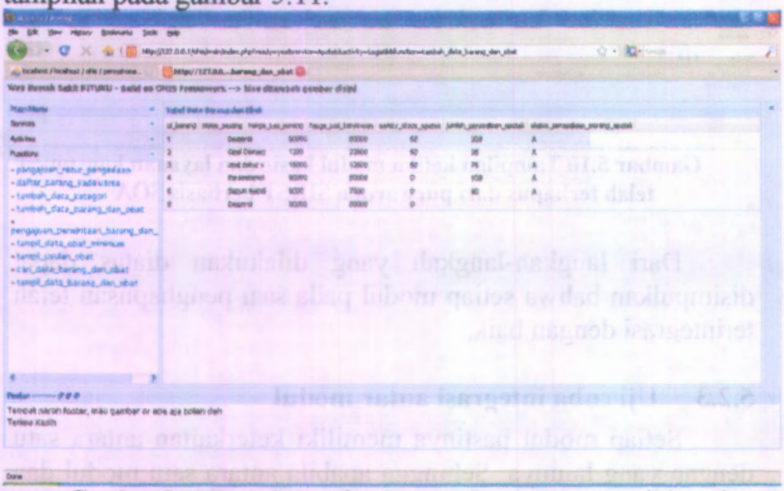
Dalam penguji cobaan dilakukan dengan cara memanggil fungsi yang telah tersedia pada pengadaan maupun IRNA.

a. Memanggil fungsi daftar_barang yang ada pada modul pengadaan

Pertama dibuat suatu fungsi pada apotek yang bertugas untuk memanggil fungsi daftar_barang pada modul pengadaan

```
public function tampil_data_barang_dan_obat(){
        $barang=Factory::get_service("Pengadaan");
        return $barang->daftar_barang($data);
}
```

Kemudian fungsi tersebut dijalankan dan menghasilkan tampilan pada gambar 5.11.



Gambar 5.11 Tampilan daftar barang dengan memanggil fungsi pada pengadaan

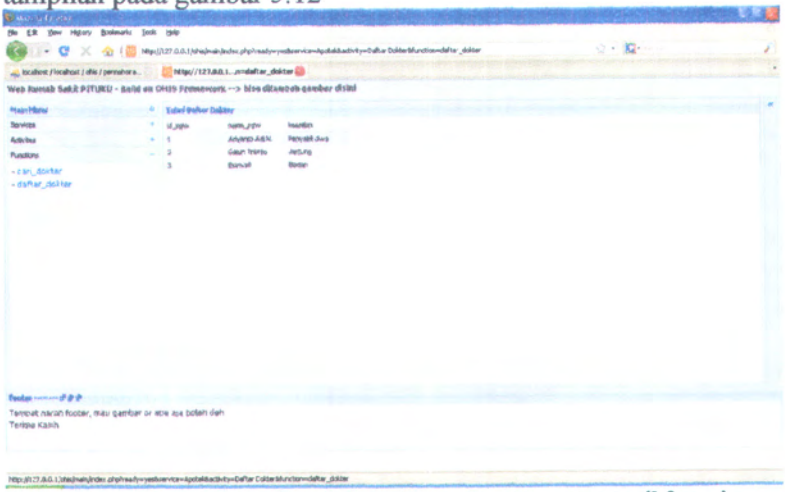
b. Memanggil fungsi daftar_dokter yang ada pada modul IRNA

Pertama dibuat suatu fungsi pada apotek yang bertugas untuk memanggil fungsi daftar_dokter pada modul IRNA

```
public function daftar_dokter(){
        $dokter=Factory::get_service("IRNA");
```

```
        return $barang->daftar_dokter($data);
}
```

Kemudian fungsi tersebut dijalankan dan menghasilkan tampilan pada gambar 5.12



Gambar 5.12 Tampilan daftar dokter dengan memanggil fungsi pada IRNA

Dari kedua tahap diatas dapat disimpulkan bahwa setiap modul telah terintegrasi dengan baik, hal ini ditunjukkan oleh kemampuan tiap modul untuk menggunakan kembali servis atau fungsi yang dimiliki oleh modul lain.

Halaman ini sengaja dikosongkan.

BAB VI
PENUTUP

Bab ini berisi simpulan dari restrukturisasi sistem informasi rumash sakit terpadu yang telah dibuat beserta saran untuk pengembangan sistem ke depan.

6.1 Simpulan

Dari uraian pada bab-bab sebelumnya dapat diperoleh kesimpulan sebagaimana berikut ini :

1. Dari 29 servis yang terdapat pada modul apotek, 11 servis dapat terselesaikan seluruhnya dan 18 servis dapat terselesaikan sebagian. Untuk detailnya dapat dilihat pada lampiran E.
2. Desain basis data pada DPPL SIRST rilis 2 ada yang masih belum memenuhi kaidah-kaidah RDBMS dan tidak sesuai dengan aturan bisnis yang ada.
3. Pembangunan modul apotek telah sesuai dan mencakup seluruh kebutuhan pada SKPL dan DPPL SIRST rilis 2.
4. Modul apotek telah terintegrasi dengan baik dalam purwarupa SIRST berbasis SOA.
5. Proses pengembangan servis mudah bagi pengembang karena tidak perlu memikirkan tampilan dan proses-proses pengiriman data dan pengambilan data. Pengembang lebih dituntut untuk fokus kepada fungsi dasar ambil data, ubah data, dan tambah data.

6.2 Saran

Beberapa hal yang diharapkan dapat dikembangkan di masa mendatang adalah sebagai berikut :

1. Sebelum dilakukan pengembangan modul, sebaiknya pengembang melakukan analisis terhadap desain basis data apakah sesuai dengan *business rule,* kebutuhan normalisasi data dan memenuhi kaidah-kaidah RDBMS.
2. Proses pengembangan modul yang dilakukan dengan menggunakan framework OHIS sebaiknya berdasarkan kemampuan dasar yang sama terlebih dahulu. Proses tersebut dapat membuat tingkat produktivitas pengembang menjadi tinggi dan mempercepat proses pengembangan.
3. Para pengembang yang membangun modul-modul dengan menggunakan framework OHIS diharapkan menyesuaikan desain yang ada dengan desain dan fitur yang ada di framework karena tidak semua desain yang ada sesuai dengan kemampuan framework.
4. Servis-servis dalam modul manajemen ada beberapa yang sama dengan modul apotek dan IRNA, sehingga pemanggilan servis tersebut lebih disarankan dariapada pembuatan ulang servis dengan kemampuan yang sama, agar pengembang tidak perlu memakan waktu lebih banyak.
5. Testing yang dilakukan pada servis yang telah dikembangkan sebaiknya menggunakan *test case* daripada *scenario testing*, karena bisa memperinci hasil dari masing-masing servis.

Daftar Pustaka

[1] Banda, G., Jucyte, K., Kebelaitis K., Park, S. W. (Fall 2006). **Web service implementation with SOAP and REST.** RUC Datalogi, Module 2.

[2] Fithroni, M. A.; Nisafani, A. S. 2009. **Deskripsi Perancangan Perangkat Lunak Sistem Informasi Rumah Sakit Terpadu Release 2**, Jurusan Sistem Informasi ITS.

[3] Nisafani, A. S. 2009. **Restrukturisasi Komponen Sistem Informasi Rumah Sakit Terpadu (SIRST) Pada Bagian Rawat Inap Dan Rawat Jalan,** Jurusan Sistem Informasi ITS.

[4] Mummah, Greg. (2006) **SOA Cures Healthcare Integration Headaches**, Business Integration Journal

[5] MySQL, <Url http://dev.mysql.com/doc/refman/5.1/en/storage-engines.html>, 1 Juli 2009

[6] Ramakrishnan, R. 2003. Basis data **Management Systems-Third Edition,** McGraw-Hill Education.

[7] Richards, R. 2006. **Pro PHP XML and Web Services**. Berkeley, USA: Appres.

[8] Rosen, M., Lublinsky B., Smith K. T., Balcer M. J. (2008). **Service-Oriented Architecture and Design Strategies.,** Wiley Publishing.

[9] W3C, <Url: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>, 28 Maret 2009

[10] Wikipedia, <Url: http://id.wikipedia.org/wiki/SOA>, 28 Maret 2009

[11] Wikipedia, <Url: http://id.wikipedia.org/wiki/SOAP>, 28 Maret 2009

Halaman ini sengaja dikosongkan.

Lampiran A:
Dokumentasi PHP Program Modul Apotek

Halaman ini sengaja dikosongkan.

Class Apotek

Description
Description | Methods (details)

Located in /Apotek.php (line 3)

Method Summary
Description | Methods (details)
Apotek __construct ()
array cari_data_barang_dan_obat (array $data)
array cari_dokter (array $data)
array cari_pasien (array $data)
array daftar_barang_kadaluarsa ()
array daftar_dokter ()
array daftar_pasien ()
array daftar_permintaan_mutasi_obat ()
array daftar_permintaan_retur_obat ()
array pembuatan_laporan_mutasi ()
array pembuatan_laporan_resep ()
array pembuatan_laporan_retur_pasien ()
array pengajuan_permintaan_barang_dan_obat (array $data)
array pengajuan_retur_pengadaan (array $data)
array penjualan_obat (array $data)
array peracikan_obat ()
array persetujuan_mutasi_obat (array $data)
array persetujuan_retur_obat_pasien (array $data)
array racik_resep (array $data)
array selesai_racik_resep (array $data)
array tambah_data_barang_dan_obat ()
array tambah_data_kategori ()
array tambah_konsumen ()
array tambah_resep ()
array tampil_data_barang_dan_obat ()
array tampil_data_konsumen ()
array tampil_data_obat_minimum ()
array tampil_data_resep ()
array tampil_usulan_obat ()
array ubah_data_resep (array $data)
void _down ()
void _up ()
Methods
Description | Methods (details)
Constructor __construct (line 13)

Constructor

- access: public

bersambung

lanjutan

Apotek __construct ()
cari_data_barang_dan_obat (line 237)

3.2.2.8 Spesifikasi Proses Form Tampil Data Barang dan Obat (FRM.APO.08)

Form yang melakukan pencarian data barang dan obat

- access: public

array cari_data_barang_dan_obat (array $data)

- array $data

cari_dokter (line 512)

3.2.2.18 Spesifikasi Proses Form Daftar Dokter (FRM.APO.18)

Form untuk mencari data dokter

- access: public

array cari_dokter (array $data)

- array $data

cari_pasien (line 549)

3.2.2.19 Spesifikasi Proses Form Daftar Pasien (FRM.APO.19)

Form untuk mencari data pasien

- access: public

array cari_pasien (array $data)

- array $data

bersambung

daftar_barang_kadaluarsa (line 363)

3.2.2.12 Spesifikasi Proses Form Pengajuan Retur Pengadaan (FRM.APO.12)

Form yang melakukan menampilkan data barang kadaluarsa

- access: public

array daftar_barang_kadaluarsa ()
daftar_dokter (line 494)

3.2.2.18 Spesifikasi Proses Form Daftar Dokter (FRM.APO.18)

Form untuk menampilkan data dokter

- access: public

array daftar_dokter ()
daftar_pasien (line 533)

3.2.2.19 Spesifikasi Proses Form Daftar Pasien (FRM.APO.19)

Form untuk menampilkan data pasien

- access: public

array daftar_pasien ()
daftar_permintaan_mutasi_obat (line 378)

3.2.2.13 Spesifikasi Proses Form Daftar Permintaan Mutasi Obat (FRM.APO.13)

Form untuk menampilkan data permintaan mutasi obat

- access: public

array daftar_permintaan_mutasi_obat ()

bersambung

lanjutan

daftar_permintaan_retur_obat (line 412)

3.2.2.15 Spesifikasi Proses Form Daftar Permintaan Retur Obat (FRM.APO.15)

Form untuk menampilkan data permintaan retur obat pasien

- access: public

array daftar_permintaan_retur_obat ()
pembuatan_laporan_mutasi (line 462)

3.2.2.17 Spesifikasi Proses Form Pembuatan Laporan (FRM.APO.17)

Form yang digunakan untuk pembuatan laporan mutasi obat

- access: public

array pembuatan_laporan_mutasi ()
pembuatan_laporan_resep (line 446)

3.2.2.17 Spesifikasi Proses Form Pembuatan Laporan (FRM.APO.17)

Form yang digunakan untuk pembuatan laporan resep

- access: public

array pembuatan_laporan_resep ()
pembuatan_laporan_retur_pasien (line 478)

3.2.2.17 Spesifikasi Proses Form Pembuatan Laporan (FRM.APO.17)

Form yang digunakan untuk pembuatan laporan retur obat pasien

- access: public

bersambung

lanjutan

array pembuatan_laporan_retur_pasien ()
pengajuan_permintaan_barang_dan_obat (line 254)

3.2.2.9 Spesifikasi Proses Form Pengajuan Permintaan Barang dan Obat (FRM.APO.09)

Form yang melakukan proses pengajuan permintaan barang dan obat

- access: public

array pengajuan_permintaan_barang_dan_obat (array $data)

- array $data

pengajuan_retur_pengadaan (line 343)

3.2.2.12 Spesifikasi Proses Form Pengajuan Retur Pengadaan (FRM.APO.12)

Form yang melakukan proses pengajuan retur ke pengadaan

- access: public

array pengajuan_retur_pengadaan (array $data)

- array $data

penjualan_obat (line 134)

3.2.2.3 Spesifikasi Proses Form Tambah Resep (FRM.APO.03)

Form yang melakukan proses penjualan obat

- access: public

array penjualan_obat (array $data)

- array $data

bersambung

lanjutan

peracikan_obat (line 164)

3.2.2.5 Spesifikasi Proses Form Peracikan Obat (FRM.APO.05)

Form yang menampilkan data peracikan obat

- access: public

array peracikan_obat ()
persetujuan_mutasi_obat (line 394)

3.2.2.14 Spesifikasi Proses Form Persetujuan Mutasi Obat (FRM.APO.14)

Form yang digunakan untuk menyetujui mutasi obat

- access: public

array persetujuan_mutasi_obat (array $data)

- array $data

persetujuan_retur_obat_pasien (line 428)

3.2.2.16 Spesifikasi Proses Form Persetujuan Retur Obat (FRM.APO.16)

Form yang digunakan untuk menyetujui retur obat pasien

- access: public

array persetujuan_retur_obat_pasien (array $data)

- array $data

racik_resep (line 180)

3.2.2.6 Spesifikasi Proses Form Racik Resep (FRM.APO.06)

bersambung

Form yang melakukan pengubahan status flow resep

- access: public

array racik_resep (array $data)

- array $data

selesai_racik_resep (line 199)

3.2.2.7 Spesifikasi Proses Form Selesai Racik Resep (FRM.APO.07)

Form yang melakukan pengubahan status flow resep

- access: public

array selesai_racik_resep (array $data)

- array $data

tambah_data_barang_dan_obat (line 306)

3.2.2.10 Spesifikasi Proses Form Tambah Data Barang dan Obat (FRM.APO.10)

Form yang melakukan penambahan data barang dan obat

- access: public

array tambah_data_barang_dan_obat ()
tambah_data_kategori (line 325)

3.2.2.11 Spesifikasi Proses Form Tambah Data Kategori (FRM.APO.11)

Form yang melakukan penambahan data kategori obat

- access: public

bersambung

lanjutan

```
array tambah_data_kategori ()
tambah_konsumen (line 116)

3.2.2.3 Spesifikasi Proses Form Tambah Resep (FRM.APO.03)

Form yang melakukan penambahan data konsumen

    • access: public

array tambah_konsumen ()
tambah_resep (line 82)

3.2.2.3 Spesifikasi Proses Form Tambah Resep (FRM.APO.03)

Form yang melakukan penambahan data resep

    • access: public

array tambah_resep ()
tampil_data_barang_dan_obat (line 219)

3.2.2.8 Spesifikasi Proses Form Tampil Data Barang dan
Obat (FRM.APO.08)

Form yang menampilkan data barang dan obat

    • access: public

array tampil_data_barang_dan_obat ()
tampil_data_konsumen (line 100)

3.2.2.3 Spesifikasi Proses Form Tambah Resep (FRM.APO.03)

Form yang menampilkan data konsumen

    • access: public

array tampil_data_konsumen ()
tampil_data_obat_minimum (line 290)
```

bersambung

3.2.2.9 Spesifikasi Proses Form Pengajuan Permintaan Barang dan Obat (FRM.APO.09)

Form yang menampilkan data obat minimum

- access: public

array tampil_data_obat_minimum ()
tampil_data_resep (line 66)

3.2.2.2 Spesifikasi Proses Form Tampil Data Resep (FRM.APO.02)

Form yang menampilkan data resep

- access: public

array tampil_data_resep ()
tampil_usulan_obat (line 274)

3.2.2.9 Spesifikasi Proses Form Pengajuan Permintaan Barang dan Obat (FRM.APO.09)

Form yang menampilkan usulan obat

- access: public

array tampil_usulan_obat ()
ubah_data_resep (line 152)

3.2.2.4 Spesifikasi Proses Form Ubah Data Resep (FRM.APO.04)

Form yang melakukan pengubahan data resep

- access: public

array ubah_data_resep (array $data)

bersambung

lanjutan

```
    • array $data

_down (line 55)

    • access: public

void _down ()
_up (line 17)

    • access: public

void _up ()
```

Lampiran B:
Source Code Program Modul Apotek

Halaman ini sengaja dikosongkan.

Source Code Program Modul Apotek

```
<?php

class Apotek {

    private $db;

    /**
         * Constructor
         *
         */


    public function __construct(){
        $this->db = Factory::get_db();
    }

    public function _up(){
        // disini tempat create table misalnya
        //dipanggil pas install service
                $resep = Array(
        'tampil_data_resep', 'tambah_resep',
'ubah_data_resep', 'tambah_konsumen',
'tampil_data_konsumen', 'penjualan_obat'
        );
                $racik = Array(
        'peracikan_obat', 'racik_resep',
'selesai_racik_resep'
        );
                $logistik = Array(
        'tampil_data_barang_dan_obat',
'cari_data_barang_dan_obat','tampil_usulan_obat',
'tampil_data_obat_minimum','pengajuan_permintaan_barang_dan_
obat', 'tambah_data_barang_dan_obat',
'tambah_data_kategori', 'daftar_barang_kadaluarsa'
,'pengajuan_retur_pengadaan'        );
                $mutasi = Array(
        'daftar_permintaan_mutasi_obat',
'persetujuan_mutasi_obat'
        );
                $retur = Array(
        'daftar_permintaan_retur_obat',
'persetujuan_retur_obat_pasien'
        );
                $laporan = Array(
        'pembuatan_laporan_resep',
'pembuatan_laporan_mutasi', 'pembuatan_laporan_retur_pasien'
        );
```

bersambung

lanjutan

```
            $dokter = Array(
        'daftar_dokter', 'cari_dokter'
        );
            $pasien = Array(
        'daftar_pasien', 'cari_pasien'
        );

        Register::functions_to_activity($resep, "Resep");
            Register::functions_to_activity($racik,
"Racik Obat");
            Register::functions_to_activity($logistik,
"Logistik");
            Register::functions_to_activity($mutasi,
"Mutasi Obat");
            Register::functions_to_activity($retur,
"Retur Obat Pasien");
            Register::functions_to_activity($laporan,
"Laporan");
            Register::functions_to_activity($dokter,
"Daftar Dokter");
            Register::functions_to_activity($pasien,
"Daftar Pasien");

        }

    public function _down(){
        // disini tempat drop table
        // dipanggil pas remove service
    }

        /**
        * 3.2.2.2 Spesifikasi Proses Form Tampil Data Resep
(FRM.APO.02)
        *Form yang menampilkan data resep
        * @param
        * @return array
        */
    public function tampil_data_resep(){
        $sth = $this->db->prepare("SELECT
po.id_penjualan_obat, po.id_resep, r.id_pasien,
p.nama_pasien, po.id_pgw, pg.nama_pgw, po.total_harga_obat,
po.status_flow_resep FROM penjualan_obat po, resep r, pasien
p, pegawai pg where po.id_resep=r.id_resep and r.id_pasien =
p.id_pasien and po.id_pgw = pg.id_pgw");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Resep");
            $data->set_data($sth->fetchAll());
        return $data->export();
```

bersambung

```
    }

        /**
         * 3.2.2.3 Spesifikasi Proses Form Tambah Resep
(FRM.APO.03)
         *Form yang melakukan penambahan data resep
         * @param
         * @return array
         */
        public function tambah_resep() {
         $sth = $this->db->prepare("insert into resep
values(3, 1, 1, 2009-07-01, null)");
         $sth->execute();
         $sth2 = $this->db->prepare("SELECT
po.id_penjualan_obat, po.id_resep, r.id_pasien,
p.nama_pasien, po.id_pgw, pg.nama_pgw, po.total_harga_obat,
po.status_flow_resep FROM penjualan_obat po, resep r, pasien
p, pegawai pg where po.id_resep=r.id_resep and r.id_pasien =
p.id_pasien and po.id_pgw = pg.id_pgw");
         $sth2->execute();
         $data = new DataSource();
         $data->set_type("Grid");
         $data->set_comp("Tabel Data Resep");
                $data->set_data($sth2->fetchAll());
         return $data->export();
        }

        /**
         * 3.2.2.3 Spesifikasi Proses Form Tambah Resep
(FRM.APO.03)
         *Form yang menampilkan data konsumen
         * @param
         * @return array
         */
        public function tampil_data_konsumen() {
         $sth = $this->db->prepare("SELECT id_pasien,
nama_pasien, alamat_pasien FROM pasien where tipe_pasien =
'LUAR'");
                $sth->execute();
         $data = new DataSource();
         $data->set_type("Grid");
         $data->set_comp("Tabel Daftar Pasien");
         $data->set_data($sth->fetchAll());
         return $data->export();
        }

        /**
         * 3.2.2.3 Spesifikasi Proses Form Tambah Resep
(FRM.APO.03)
         *Form yang melakukan penambahan data konsumen
```

bersambung

lanjutan

```
        * @param
        * @return array
        */
       public function tambah_konsumen() {
              $sth = $this->db->prepare("INSERT INTO
`ohis`.`pasien` (`ID_PASIEN` ,`NAMA_PASIEN` ,`ALAMAT_PASIEN`
,`TGLLAHIR_PASIEN` ,`TEMPATLAHIR_PASIEN` ,`JK_PASIEN`
,`AGAMA_PASIEN` ,`NOKTP_PASIEN` ,`NOASURANSI_PASIEN`
,`STATUS_NIKAH_PASIEN` ,`WARGA_NEGARA_PASIEN`
,`PENDIDIKAN_PASIEN` ,`KODEPOS_PASIEN` ,`TELP_PASIEN`
,`HP_PASIEN` ,`EMAIL_PASIEN` ,`INSTANSI_PASIEN`
,`ALAMAT_KANTOR_PASIEN` ,`KODEPOS_KANTOR_PASIEN`
,`TELP_KANTOR_PASIEN` ,`NAMA_IBUKANDUNG_PASIEN`
,`PENANGGUNG_BIAYA_PASIEN` ,`ALAMAT_PENANGGUNG_PASIEN`
,`TELP_PENANGGUNG` ,`FOTO_PASIEN` ,`LOGTIME_PGW_ENTRY`
,`TIPE_PASIEN`)VALUES ('4', 'Joko Karto', 'Jl Keputih 99999
Surabaya', '2009-07-01', 'Surabaya', 'L', 'Islam',
'1290280218', '7687638961', 'N', 'Indonesia', 'SMA', '1245',
'0319998877', NULL , NULL , NULL , NULL , NULL , NULL , '',
NULL , NULL , NULL , NULL , NULL , 'LUAR')");
              $sth->execute();
       $sth2 = $this->db->prepare("SELECT id_pasien,
nama_pasien, alamat_pasien FROM pasien where tipe_pasien =
'LUAR'");
              $sth2->execute();
       $data = new DataSource();
       $data->set_type("Grid");
       $data->set_comp("Tabel Daftar Pasien");
       $data->set_data($sth2->fetchAll());
       return $data->export();
              }

      /**
       * 3.2.2.3 Spesifikasi Proses Form Tambah Resep
(FRM.APO.03)
       *Form yang melakukan proses penjualan obat
       * @param  array $data
       * @return array
       */
      public function penjualan_obat($data) {
              $sth = $this->db->prepare("INSERT INTO
`ohis`.`penjualan_obat` (`ID_PENJUALAN_OBAT`, `ID_RESEP`,
`ID_PGW`, `TOTAL_HARGA_OBAT`, `STATUS_FLOW_RESEP`,
`TGL_PEMBUATAN`, `DATA_KWITANSI`, `NO_ANTRIAN_RESEP`,
`KET_PEMBUATAN`, `TGL_PERACIKAN`, `ASAL_RESEP`,
`LOG_DATE_PENJUALAN`, `LOG_TIME_PENJUALAN`) VALUES ('4',
'3', '3', '120000', 'belum diracik', '2009-07-01', 'A',
'24', NULL, NULL, 'IRNA', NULL, NULL);)");
              $sth->execute();
       $sth2 = $this->db->prepare("SELECT
```

bersambung

```
po.id_penjualan_obat, po.id_resep, r.id_pasien,
p.nama_pasien, po.id_pgw, pg.nama_pgw, po.total_harga_obat,
po.status_flow_resep FROM penjualan_obat po, resep r, pasien
p, pegawai pg where po.id_resep=r.id_resep and r.id_pasien =
p.id_pasien and po.id_pgw = pg.id_pgw");
                $sth2->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel penjualan");
        $data->set_data($sth2->fetchAll());
        return $data->export();
                }

       /**
        * 3.2.2.4 Spesifikasi Proses Form Ubah Data Resep
(FRM.APO.04)
        *Form yang melakukan pengubahan data resep
        * @param  array $data
        * @return array
        */
       public function ubah_data_resep($data) {
                $data = 3;
        $sth = $this->db->prepare("update detail_resep set
id_resep = 2, id_barang = 1, jumlah_obat =24,
harga_jual_satuan = 100, harga_obat = 2400,
aturan_pakai_dokter = '3 kali'  where no_urut_resep =
'".$data."'");
        $sth->execute();
       }

       /**
        * 3.2.2.5 Spesifikasi Proses Form Peracikan Obat
(FRM.APO.05)
        *Form yang menampilkan data peracikan obat
        * @param
        * @return array
        */
    public function peracikan_obat(){
        $sth = $this->db->prepare("SELECT
po.id_penjualan_obat, po.id_resep, r.id_pasien,
p.nama_pasien, po.id_pgw, pg.nama_pgw, po.total_harga_obat,
po.status_flow_resep FROM penjualan_obat po, resep r, pasien
p, pegawai pg where po.id_resep=r.id_resep and r.id_pasien =
p.id_pasien and po.id_pgw = pg.id_pgw");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Peracikan Obat");
        $data->set_data($sth->fetchAll());
        return $data->export();
```

bersambung

lanjutan

```
    }
        /**
         * 3.2.2.6 Spesifikasi Proses Form Racik Resep
(FRM.APO.06)
         *Form yang melakukan pengubahan status flow resep
         * @param  array $data
         * @return array
         */
        public function racik_resep($data) {
                $data=1;
        $sth = $this->db->prepare("update penjualan_obat set
status_flow_resep = 'sedang diracik' where id_penjualan_obat
= '".$data."'");
        $sth->execute();
        $sth2 = $this->db->prepare("SELECT
po.id_penjualan_obat, po.id_resep, r.id_pasien,
p.nama_pasien, po.id_pgw, pg.nama_pgw, po.total_harga_obat,
po.status_flow_resep FROM penjualan_obat po, resep r, pasien
p, pegawai pg where po.id_resep=r.id_resep and r.id_pasien =
p.id_pasien and po.id_pgw = pg.id_pgw");
        $sth2->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Resep");
                $data->set_data($sth2->fetchAll());
        return $data->export();
        }

        /**
         * 3.2.2.7 Spesifikasi Proses Form Selesai Racik
Resep (FRM.APO.07)
         *Form yang melakukan pengubahan status flow resep
         * @param  array $data
         * @return array
         */
        public function selesai_racik_resep($data) {
                $data=1;
        $sth = $this->db->prepare("update penjualan_obat set
status_flow_resep = 'selesai diracik' where
id_penjualan_obat = '".$data."'");
        $sth->execute();
        $sth2 = $this->db->prepare("SELECT
po.id_penjualan_obat, po.id_resep, r.id_pasien,
p.nama_pasien, po.id_pgw, pg.nama_pgw, po.total_harga_obat,
po.status_flow_resep FROM penjualan_obat po, resep r, pasien
p, pegawai pg where po.id_resep=r.id_resep and r.id_pasien =
p.id_pasien and po.id_pgw = pg.id_pgw");
        $sth2->execute();
        $data = new DataSource();
```

bersambung

lanjutan

```
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Resep");
                $data->set_data($sth2->fetchAll());
        return $data->export();
        }


        /**
         * 3.2.2.8 Spesifikasi Proses Form Tampil Data Barang
dan Obat (FRM.APO.08)
         *Form yang menampilkan data barang dan obat
         * @param
         * @return array
         */
    public function tampil_data_barang_dan_obat(){
                $sth = $this->db->prepare("SELECT
p.id_barang, b.nama_barang, b.harga_jual_barang,
b.harga_jual_karyawan, p.safety_stock_apotek,
p.jumlah_persediaan_apotek,
p.status_persediaan_barang_apotek from persediaan_apotek p,
barang b where p.id_barang = b.id_barang");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Barang dan Obat");
        $data->set_data($sth->fetchAll());
        return $data->export();
                //$barang=Factory::get_service("Pengadaan");
                //return $barang->daftar_barang($data);
    }

        /**
         * 3.2.2.8 Spesifikasi Proses Form Tampil Data Barang
dan Obat (FRM.APO.08)
         *Form yang melakukan pencarian data barang dan obat
         * @param  array $data
         * @return array
         */
    public function cari_data_barang_dan_obat($data){
                $data=1;
                $sth = $this->db->prepare("SELECT
p.id_barang, b.nama_barang, b.harga_jual_barang,
b.harga_jual_karyawan, p.safety_stock_apotek,
p.jumlah_persediaan_apotek,
p.status_persediaan_barang_apotek from persediaan_apotek p,
barang b where p.id_barang='".$data."' and p.id_barang =
b.id_barang");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
```

bersambung

lanjutan

```
        $data->set_comp("Tabel Data Barang dan Obat");
        $data->set_data($sth->fetchAll());
        return $data->export();
              }

       /**
        * 3.2.2.9 Spesifikasi Proses Form Pengajuan
Permintaan Barang dan Obat (FRM.APO.09)
        *Form yang melakukan proses pengajuan permintaan
barang dan obat
        * @param  array $data
        * @return array
        */
       public function
pengajuan_permintaan_barang_dan_obat($data) {
              $sth = $this->db->prepare("INSERT INTO
`ohis`.`permintaan_barang` (`ID_PR`, `ID_PGW`,
`TGL_PERMINTAAN_BARANG`, `JUMLAH_REALISASI_PERMINTBARANG`,
`UNIT_PEMOHON`, `TOTAL_HARGA_PERMINTAAN`,
`STATUS_PERMINTAAN`) VALUES ('2', '1', '2009-07-01', '10',
'Apotek', '100000', 'A')");
              $sth->execute();
              $sth2 = $this->db->prepare("INSERT INTO
`ohis`.`detail_permintaan_barang` (`NO_URUT_PERMINTAAN`
,`ID_BARANG` ,`ID_PR` ,`JUMLAH_PERMINTAAN`
,`SPESIFIKASI_BARANG_PERMINTAAN` ,`STATUS_BARANG_PERMINTAAN`
,`HARGA_PERKIRAAN_PERMINTAAN` ,`HARGA_TOTAL_PERMINTAAN`
,`ID_PO_TEMP`)VALUES ('1', '1', '1', '5', 'kualitas tinggi',
'A', '5000', '750000', NULL), ('2', '3', '1', '10',
'kualitas sedang', 'A', '2500', '250000', NULL)");
              $sth2->execute();
              $sth3 = $this->db->prepare("select * from
permintaan_barang where unit_pemohon = 'Apotek'");
        $sth3->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Daftar Pengajuan Barang dan
Obat");
        $data->set_data($sth3->fetchAll());
        return $data->export();
       }

       /**
        * 3.2.2.9 Spesifikasi Proses Form Pengajuan
Permintaan Barang dan Obat (FRM.APO.09)
        *Form yang menampilkan usulan obat
        * @param
        * @return array
        */
        public function tampil usulan obat(){
```

bersambung

```
                $sth = $this->db->prepare("SELECT * from
usulan_obat");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Usulan Obat");
        $data->set_data($sth->fetchAll());
        return $data->export();
    }

       /**
        * 3.2.2.9 Spesifikasi Proses Form Pengajuan
Permintaan Barang dan Obat (FRM.APO.09)
        *Form yang menampilkan data obat minimum
        * @param
        * @return array
        */
    public function tampil_data_obat_minimum(){
                $sth = $this->db->prepare("SELECT
p.id_barang, b.nama_barang, b.harga_jual_barang,
b.harga_jual_karyawan, p.safety_stock_apotek,
p.jumlah_persediaan_apotek,
p.status_persediaan_barang_apotek from persediaan_apotek p,
barang b where p.id_barang = b.id_barang and
p.jumlah_persediaan_apotek <= p.safety_stock_apotek");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Obat Minimum");
        $data->set_data($sth->fetchAll());
        return $data->export();
    }

       /**
        * 3.2.2.10 Spesifikasi Proses Form Tambah Data
Barang dan Obat (FRM.APO.10)
        *Form yang melakukan penambahan data barang dan obat
        * @param
        * @return array
        */
       public function tambah_data_barang_dan_obat() {
        $sth = $this->db->prepare("INSERT INTO
`ohis`.`barang` (`ID_BARANG`, `NAMA_BARANG`,
`HARGA_JUAL_BARANG`, `HARGA_JUAL_KARYAWAN`,
`SAFETY_STOCK_BARANG`, `STATUS_AKTIF_BARANG`,
`DEMAND_TAHUNAN_BARANG`, `HARGA_BELI_BARANG`,
`BIAYA_SIMPAN_BARANG`, `BIAYA_PESAN_BARANG`,
`TGL_UPDATE_BARANG`, `NAMA_PRODUSEN`, `TYPE_BARANG`) VALUES
('3', 'Obat Demam', '5000', '4000', '100', 'A', '150',
'3500', '2000', '3000', '2009-07-14 19:49:12', 'Anti Demam',
```

bersambung

lanjutan

```
'obat')");
        $sth->execute();
                $sth2 = $this->db->prepare("INSERT INTO
`ohis`.`persediaan_apotek` (`ID_BARANG`
,`JUMLAH_PERSEDIAAN_APOTEK`
,`STATUS_PERSEDIAAN_BARANG_APOTEK` ,`SAFETY_STOCK_APOTEK`)
VALUES ('3', '75', 'A', '50')");
                $sth2->execute();
                $sth3 = $this->db->prepare("SELECT
p.id_barang, b.nama_barang, b.harga_jual_barang,
b.harga_jual_karyawan, p.safety_stock_apotek,
p.jumlah_persediaan_apotek,
p.status_persediaan_barang_apotek from persediaan_apotek p,
barang b where p.id_barang = b.id_barang");
        $sth3->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Barang dan Obat");
        $data->set_data($sth3->fetchAll());
        return $data->export();                }

       /**
        * 3.2.2.11 Spesifikasi Proses Form Tambah Data
Kategori (FRM.APO.11)
        *Form yang melakukan penambahan data kategori obat
        * @param
        * @return array
        */
       public function tambah_data_kategori() {
        $sth = $this->db->prepare("insert into
master_kategoriobat values(4, 'obat pusing', null)");
        $sth->execute();
                $sth2 = $this->db->prepare("select * from
master_kategoriobat");
                $sth2->execute();
                $data = new DataSource();
                $data->set_type("Grid");
                $data->set_comp("Tabel data kategori obat");
                $data->set_data($sth2->fetchAll());
                return $data->export();
                }

       /**
        * 3.2.2.12 Spesifikasi Proses Form Pengajuan Retur
Pengadaan (FRM.APO.12)
        *Form yang melakukan proses pengajuan retur ke
pengadaan
        * @param  array $data
        * @return array
        */
```

bersambung

lanjutan

```
        public function pengajuan_retur_pengadaan($data) {
                $sth = $this->db->prepare("INSERT INTO
`ohis`.`permohonan_retur_barang` (`ID_RETUR_BARANG`
,`ID_PGW` ,`TGL_RETUR_BARANG` ,`UNIT_PEMOHON_RETUR`
,`STATUS_RETUR_BARANG`)VALUES ('4', '1', '2009-07-02',
'Apotek', NULL)");
                $sth->execute();
                $sth2 = $this->db->prepare("INSERT INTO
`ohis`.`detail_retur_barang` (`NO_URUT_RETUR_BARANG`,
`ID_RETUR_BARANG`, `ID_BARANG`, `JUMLAH_RETUR_BARANG`,
`STATUS_RETUR`, `STATUS_PELAKSANAAN_RETUR`) VALUES ('4',
'4', '9', '12', NULL, NULL)");
                $sth2->execute();
                $sth3 = $this->db->prepare("select
id_retur_barang, id_pgw, tgl_retur_barang from
permohonan_retur_barang");
        $sth3->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Daftar Pengajuan retur
pengadaan");
        $data->set_data($sth3->fetchAll());
        return $data->export();
        }

        /**
         * 3.2.2.12 Spesifikasi Proses Form Pengajuan Retur
Pengadaan (FRM.APO.12)
         *Form yang melakukan menampilkan data barang
kadaluarsa
         * @param
         * @return array
         */
         public function daftar_barang_kadaluarsa(){
            $sth = $this->db->prepare("SELECT
f.ID_FAKTUR,b.nama_barang, d.`JUMLAH_BARANG_FAKTUR`,
d.`TANGGAL_KADALUARSA_BARANG`
                FROM `detail_faktur` d, faktur_pembelian f,
barang b
                where f.id_faktur= d.id_faktur and
d.`TANGGAL_KADALUARSA_BARANG` < now() and b.id_barang =
d.id_barang");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_data($sth->fetchAll());
        return $data->export();
         }

        /**
```

bersambung

lanjutan

```
        * 3.2.2.13 Spesifikasi Proses Form Daftar Permintaan
Mutasi Obat (FRM.APO.13)
        *Form untuk menampilkan data permintaan mutasi obat
        * @param
        * @return array
        */
    public function daftar_permintaan_mutasi_obat(){
        $sth = $this->db->prepare("SELECT * FROM
permohonan_mutasi_obat");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Resep");
        $data->set_data($sth->fetchAll());
        return $data->export();
    }

       /**
        * 3.2.2.14 Spesifikasi Proses Form Persetujuan
Mutasi Obat (FRM.APO.14)
        *Form yang digunakan untuk menyetujui mutasi obat
        * @param  array $data
        * @return array
        */
       public function persetujuan_mutasi_obat($data) {
            $sth = $this->db->prepare("UPDATE
`ohis`.`permohonan_mutasi_obat` SET
`STATUS_PELAKSANAAN_MUTASI` = 'S' WHERE
`permohonan_mutasi_obat`.`ID_MUTASI` =1");
            $sth->execute();
        $sth2 = $this->db->prepare("SELECT * FROM
permohonan_mutasi_obat");
        $sth2->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Mutasi Obat");
        $data->set_data($sth2->fetchAll());
        return $data->export();
       }

       /**
        * 3.2.2.15 Spesifikasi Proses Form Daftar Permintaan
Retur Obat (FRM.APO.15)
        *Form untuk menampilkan data permintaan retur obat
pasien
        * @param
        * @return array
        */
    public function daftar_permintaan_retur_obat(){
        $sth = $this->db->prepare("SELECT id_retur_obat,
```

bersambung

```
id_pasien, id_pgw, id_resep, tgl_retur_obat, status FROM
retur_obat_pasien");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Retur Obat");
        $data->set_data($sth->fetchAll());
        return $data->export();
    }

        /**
         * 3.2.2.16 Spesifikasi Proses Form Persetujuan Retur
Obat (FRM.APO.16)
         *Form yang digunakan untuk menyetujui retur obat
pasien
         * @param  array $data
         * @return array
         */
        public function persetujuan_retur_obat_pasien($data)
{
                $sth = $this->db->prepare("UPDATE
`ohis`.`retur_obat_pasien` SET `STATUS` = 'S' WHERE
`retur_obat_pasien`.`ID_RETUR_OBAT` =1");
                $sth->execute();
        $sth2 = $this->db->prepare("SELECT id_retur_obat,
id_pasien, id_pgw, id_resep, tgl_retur_obat, status FROM
retur_obat_pasien");
        $sth2->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Retur Obat");
        $data->set_data($sth2->fetchAll());
        return $data->export();
        }

        /**
         * 3.2.2.17 Spesifikasi Proses Form Pembuatan Laporan
(FRM.APO.17)
         *Form yang digunakan untuk pembuatan laporan resep
         * @param
         * @return array
         */
        public function pembuatan_laporan_resep() {
        $sth = $this->db->prepare("SELECT
po.id_penjualan_obat, po.id_resep, r.id_pasien,
p.nama_pasien, po.id_pgw, pg.nama_pgw, po.total_harga_obat,
po.status_flow_resep FROM penjualan_obat po, resep r, pasien
p, pegawai pg where po.id_resep=r.id_resep and r.id_pasien =
p.id_pasien and po.id_pgw = pg.id_pgw");
        $sth->execute();
```

bersambung

lanjutan

```
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Resep");
               $data->set_data($sth->fetchAll());
        return $data->export();
               }

       /**
        * 3.2.2.17 Spesifikasi Proses Form Pembuatan Laporan
(FRM.APO.17)
        *Form yang digunakan untuk pembuatan laporan mutasi
obat
        * @param
        * @return array
        */
       public function pembuatan_laporan_mutasi() {
        $sth = $this->db->prepare("SELECT * FROM
permohonan_mutasi_obat");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Resep");
        $data->set_data($sth->fetchAll());
        return $data->export();
               }

       /**
        * 3.2.2.17 Spesifikasi Proses Form Pembuatan Laporan
(FRM.APO.17)
        *Form yang digunakan untuk pembuatan laporan retur
obat pasien
        * @param
        * @return array
        */
       public function pembuatan_laporan_retur_pasien() {
        $sth = $this->db->prepare("SELECT id_retur_obat,
id_pasien, id_pgw, id_resep, tgl_retur_obat, status FROM
retur_obat_pasien");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Data Retur Obat");
        $data->set_data($sth->fetchAll());
        return $data->export();
               }

       /**
        *3.2.2.18 Spesifikasi Proses Form Daftar Dokter
(FRM.APO.18)
        *Form untuk menampilkan data dokter
```

bersambung

```
         * @param
         * @return array
         */
    public function daftar_dokter(){
        /*$sth = $this->db->prepare("SELECT id_pgw,
nama_pgw, keahlian FROM pegawai where jabatan='Dokter'");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Daftar Dokter");
        $data->set_data($sth->fetchAll());
        return $data->export();*/
               $dokter=Factory::get_service("IRNA");
               return $barang->daftar_dokter($data);
    }

        /**
         *3.2.2.18 Spesifikasi Proses Form Daftar Dokter
(FRM.APO.18)
         *Form untuk mencari data dokter
         * @param  array $data
         * @return array
         */
    public function cari_dokter($data){
               $data=1;
        $sth = $this->db->prepare("SELECT id_pgw, nama_pgw,
keahlian FROM pegawai where id_pgw= '".$data."'and
jabatan='Dokter'");
        $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Daftar Dokter");
        //$colw = array(
               //150, 150
               //);
               //$data->set_col_width($colw);
        $data->set_data($sth->fetchAll());
        return $data->export();
    }

        /**
         *3.2.2.19 Spesifikasi Proses Form Daftar Pasien
(FRM.APO.19)
         *Form untuk menampilkan data pasien
         * @param
         * @return array
         */
    public function daftar_pasien(){
        $sth = $this->db->prepare("SELECT id_pasien,
nama_pasien, alamat_pasien FROM pasien");
```

bersambung

```
            $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Daftar Pasien");
        $data->set_data($sth->fetchAll());
        return $data->export();
        }

        /**
         *3.2.2.19 Spesifikasi Proses Form Daftar Pasien
(FRM.APO.19)
         *Form untuk mencari data pasien
         * @param  array $data
         * @return array
         */
    public function cari_pasien($data){
            $data=1;
        $sth = $this->db->prepare("SELECT id_pasien,
nama_pasien, alamat_pasien FROM pasien where id_pasien =
'".$data."'");
            $sth->execute();
        $data = new DataSource();
        $data->set_type("Grid");
        $data->set_comp("Tabel Daftar Pasien");
        $data->set_data($sth->fetchAll());
        return $data->export();
    }
}
?>
```

Lampiran C:
Daftar Servis Modul Apotek

Halaman ini sengaja dikosongkan.

Daftar Servis Modul Apotek

No. servis	Nama servis	Keterangan fungsi servis	Kode servis dalam kode PHP
1.	Servis menampilkan daftar resep	Digunakan untuk mengambil dan menampilkan daftar data resep	Tampil_data_resep
2.	Servis menambah resep	Digunakan untuk mencatat data resep baru	Tambah_resep
3.	Servis menampilkan daftar konsumen	Digunakan untuk mengambil dan menampilkan daftar data konsumen	Tampil_data_konsumen
4.	Servis menambah konsumen	Digunakan untuk mencatat data konsumen baru	Tambah_konsumen
5.	Servis memasukkan penjualan obat	Digunakan untuk mencatat setiap penjualan obat yang terjadi	Penjualan_obat
6.	Servis mengubah resep	Digunakan untuk mengubah data resep yang ada	Ubah_data_resep
7.	Servis menampilkan daftar peracikan obat	Digunakan untuk mengambil dan menampilkan daftar data obat yang akan diracik	Peracikan_obat
8.	Servis mengubah status resep bahwa resep mulai diracik	Digunakan untuk mengubah status resep apabila obat mulai diracik	Racik_resep
9.	Servis mengubah status resep bahwa obat telah selesai diracik	Digunakan untuk mengubah status resep apabila obat sudah selesai diracik	Selesai_racik_resep

bersambung

lanjutan

No. servis	Nama servis	Keterangan fungsi servis	Kode servis dalam kode PHP
10.	Servis menampikan daftar barang dan obat	Digunakan untuk mengambil dan menampilkan daftar data barang dan obat	Tampil_data_barang_dan_obat
11.	Servis mencari daftar barang dan obat	Digunakan untuk mencari data barang dan obat	Cari_data_barang_dan_obat
12.	Servis mengajukan permintaan barang dan obat ke pengadaan	Digunakan untuk mencatat setiap pengajuan permintaan barang dan obat ke manajemen	Pengajuan_permintaan_barang_dan_obat
13.	Servis menampilkan usulan obat	Digunakan untuk mengambil dan menampilkan daftar data usulan obat	Tampil_usulan_obat
14.	Servis menampilkan daftar obat minimum	Digunakan mengambil dan menampilkan data obat yang sudah sama atau kurang dari safety stock	Tampil_data_obat_minimum
15.	Servis menambah barang dan obat	Digunakan untuk mencatat data barang dan obat baru	Tambah_data_barang_dan_obat
16.	Servis tambah kategori obat	Digunakan untuk mencatat kategori obat baru	Tambah_data_kategori
17.	Servis mengajukan retur barang ke pengadaan	Digunakan untuk mencatat setiap pengajuan retur barang ke pengadaan	Pengajuan_retur_pengadaan
18.	Servis menampilkan daftar barang kadaluarsa	Digunakan untuk mengambil dan menampilkan daftar data barang kadaluarsa	Daftar_barang_kadaluarsa
19.	Servis menampilkan daftar permintaan mutasi obat	Digunakan untuk mengambil dan menampilkan daftar data permintaan mutasi obat dari unit lain	Daftar_permintaan_mutasi_obat
20.	Servis menyetujui mutasi	Digunakan untuk mengubah status mutasi obat	Persetujuan_mutasi_obat

bersambung

lanjutan

No. servis	Nama servis	Keterangan fungsi servis	Kode servis dalam kode PHP
	obat	yang telah disetujui	
21.	Servis menampilkan daftar permintaan retur obat pasien	Digunakan untuk mengambil dan menampilkan daftar data permintaan retur obat pasien	Daftar_permintaan_retur_obat
22.	Servis menyetujui retur obat pasien	Digunakan untuk mengubah status retur obat pasien yang telah disetujui	Persetujuan_retur_obat_pasien
23.	Servis membuat laporan resep	Digunakan untuk membuat laporan resep	Pembuatan_laporan_resep
24.	Servis membuat laporan mutasi obat	Digunakan untuk membuat laporan mutasi obat	Pembuatan_laporan_mutasi
25.	Servis membuat laporan retur obat pasien	Digunakan untuk membuat laporan retur obat pasien	Pembuatan_laporan_retur_pasien
26.	Servis menampilkan daftar dokter rumah sakit	Digunakan untuk mengambil dan menampilkan daftar data dokter di rumah sakit	Daftar_dokter
27.	Servis mencari dokter	Digunakan untuk mencari data dokter di rumah sakit	Cari_dokter
28.	Servis menampilkan daftar pasien	Digunakan untuk mengambil dan menampilkan daftar data pasien	Daftar_pasien
29.	Servis mencari pasien	Digunakan untuk mencari pasien	Cari_pasien

Halaman ini sengaja dikosongkan.

Halaman ini sengaja dikosongkan.

Matriks Kerunutan Modul Apotek

No. servis	Kode servis	No. form DPPL	Tabel	
			Input	Output
1	Tampil_data_resep	FRM.APO.02	-	Penjualan_obat
2	Tambah_resep	FRM.APO.03	Resep Detail_resep	-
3	Tampil_data_konsumen	FRM.APO.03	-	Pasien
4	Tambah_konsumen	FRM.APO.03	Pasien	-
5	Penjualan_obat	FRM.APO.03	Penjualan_obat	-
6	Ubah_data_resep	FRM.APO.04	Detail_resep	-
7	Peracikan_obat	FRM.APO.05	-	Penjualan_obat
8	Racik_resep	FRM.APO.06	Penjualan_obat	-
9	Selesai_racik_resep	FRM.APO.07	Penjualan_obat	-
10	Tampil_data_barang_dan_obat	FRM.APO.08	-	Barang Persediaan_apotek
11	Cari_data_barang_dan_obat	FRM.APO.08	-	Barang Persediaan_apotek
12	Pengajuan_permintaan_b	FRM.APO.09	Permintaan_barang	-

bersambung

lanjutan

No. servis	Kode servis	No. form DPPL	Tabel	
			Input	Output
	arang_dan_obat		Detail_permintaan_barang	
13	Tampil_usulan_obat	FRM.APO.09	-	Usulan_obat
14	Tampil_data_obat_minimum	FRM.APO.09	-	Persediaan_apotek
15	Tambah_data_barang_dan_obat	FRM.APO.10	Persediaan_apotek	-
16	Tambah_data_kategori	FRM.APO.11	Master_kategoriobat	-
17	Pengajuan_retur_pengadaan	FRM.APO.12	Permohonan_retur_barang Detail_retur_barang	-
18	Daftar_barang_kadaluarsa	FRM.APO.12	-	Faktur_pembelian Detail_faktur Barang
19	Daftar_permintaan_mutasi_obat	FRM.APO.13	-	Permohonan_mutasi_obat
20	Persetujuan_mutasi_obat	FRM.APO.14	Permohonan_mutas	-

bersambung

lanjutan

No. servis	Kode servis	No. form DPPL	Tabel	
			Input	Output
			i_obat	
21	Daftar_permintaan_retur_obat	FRM.APO.15	-	Retur_obat_pasien
22	Persetujuan_retur_obat_pasien	FRM.APO.16	Retur_obat_pasien	-
23	Pembuatan_laporan_resep	FRM.APO.17	-	Penjualan_obat
24	Pembuatan_laporan_mutasi	FRM.APO.17	-	Permohonan_mutasi_obat
25	Pembuatan_laporan_retur_pasien	FRM.APO.17	-	Retur_obat_pasein
26	Daftar_dokter	FRM.APO.18	-	Pegawai
27	Cari_dokter	FRM.APO.18	-	Pegawai
28	Daftar_pasien	FRM.APO.19	-	Pasien
29	Cari_pasien	FRM.APO.19	-	Pasien

Halaman ini sengaja dikosongkan.

Halaman ini sengaja dikosongkan.

Test Case Modul Apotek

Prekondisi: User membuka web browser dan menuju alamat ohis lalu menekan link servis Apotek

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
1	Tampil data resep	Untuk melakukan pengujian terhadap fungsi menampilkan data dari tabel penjualan	Menekan link Resep pada activites dan menekan link tampil_data_resep pada functions	Tampil daftar resep yang terdapat dalam tabel penjualan	Tampil daftar resep yang terdapat dalam tabel penjualan	P
2	Tambah Resep	Untuk melakukan pengujian terhadap fungsi penambahan data pada tabel resep dan tabel detail_resep	Menekan link Resep pada activites dan menekan link tambah_resep pada functions	Data yang diinputkan dalam perintah insert masuk ke tabel resep dan tabel detail_resep	Data yang diinputkan dalam perintah insert masuk ke tabel resep dan tabel detail_resep	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
3	Tampil data konsumen	Untuk melakukan pengujian terhadap fungsi menampilkan data konsumen dari tabel pasien	Menekan link Resep pada activites dan menekan link tampil_data_konsumen pada functions	Tampil daftar konsumen yang terdapat dalam tabel pasien	Tampil daftar konsumen yang terdapat dalam tabel pasien	P
4	Tambah Konsumen	Untuk melakukan pengujian terhadap fungsi penambahan data pada tabel pasien	Menekan link Resep pada activites dan menekan link tambah_konsumen pada functions	Data yang diinputkan dalam perintah insert masuk ke tabel pasien	Data yang diinputkan dalam perintah insert masuk ke tabel pasien	P
5	Penjualan Obat	Untuk melakukan pengujian terhadap fungsi penambahan data pada tabel	Menekan link Resep pada activites dan menekan link penjualan_obat pada	Data yang diinputkan dalam perintah insert masuk ke tabel penjualan_obat	Data yang diinputkan dalam perintah insert masuk ke tabel penjualan_obat	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
		penjualan_obat	functions			
6	Ubah Data Resep	Untuk melakukan pengujian terhadap fungsi perubahan data pada tabel detail_resep	Menekan link Resep pada activites dan menekan link ubah_data_resep pada functions	Data yang diinputkan dalam fungsi update merubah data pada tabel detail_resep yang dituju	Data yang diinputkan dalam fungsi update merubah data pada tabel detail_resep yang dituju	P
7	Peracikan Obat	Untuk melakukan pengujian terhadap fungsi menampilkan data dari tabel penjualan_obat	Menekan link Peracikan pada activites dan menekan link peracikan_obat pada functions	Tampil data peracikan obat yang terdapat pada tabel penjualan_obat	Tampil data peracikan obat yang terdapat pada tabel penjualan_obat	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
8	Racik Resep	Untuk melakukan pengujian terhadap fungsi perubahan data status_flow_resep pada tabel penjualan_obat	Menekan link Peracikan pada activites dan menekan link racik_resep pada functions	Data status_flow_resep pada tabel penjualan_obat berubah menjadi "sedang diracik"	Data status_flow_resep pada tabel penjualan_obat berubah menjadi "sedang diracik"	P
9	Selesai Racik Resep	Untuk melakukan pengujian terhadap fungsi perubahan data status_flow_resep pada tabel penjualan_obat	Menekan link Peracikan pada activites dan menekan link selesai_racik_resep pada functions	Data status_flow_resep pada tabel penjualan_obat berubah menjadi "selesai diracik"	Data status_flow_resep pada tabel penjualan_obat berubah menjadi "selesai diracik"	P

bersambung

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
10	Tampil Data Barang Dan Obat	Untuk melakukan pengujian terhadap fungsi menampilkan data dari tabel persediaan_apotek	Menekan link Logistik pada activites dan menekan link tampil_data_barang_dan_obat pada functions	Tampil data barang dan obat yang terdapat pada tabel persediaan_apotek	Tampil data barang dan obat yang terdapat pada tabel persediaan_apotek	P
11	Cari Data Barang Dan Obat	Untuk melakukan pengujian terhadap fungsi pencarian data barang dan obat dari tabel persediaan_apotek	Menekan link Logistik pada activites dan menekan link cari_data_barang_dan_obat pada functions	Tampil data barang dan obat yang terdapat pada tabel persediaan_apotek	Tampil data barang dan obat yang terdapat pada tabel persediaan_apotek	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
12	Pengajuan Permintaan Barang dan Obat	Untuk melakukan pengujian terhadap fungsi penambahan data pada tabel permintaan_barang dan detail_permintaan_barang	Menekan link Logistik pada activites dan menekan link pengajuan_permintaan_barang_dan_obat pada functions	Data yang diinputkan dalam perintah insert masuk ke tabel permintaan_barang dan detail_permintaan_barang	Data yang diinputkan dalam perintah insert masuk ke tabel permintaan_barang dan detail_permintaan_barang	P
13	Tampil Usulan Obat	Untuk melakukan pengujian terhadap fungsi menampilkan data dari tabel usulan_obat	Menekan link Logistik pada activities dan menekan link tampil_usulan_obat pada functions	Tampil data usulan obat yang terdapat pada tabel usulan_obat	Tampil data usulan obat yang terdapat pada tabel usulan_obat	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
14	Tampil Data Obat Minimum	Untuk melakukan pengujian terhadap fungsi menampilkan data obat minimum dari tabel persediaan_apotek	Menekan link Logistik pada activities dan menekan link tampil_data_obat_minimum pada functions	Tampil data obat minimum yang terdapat pada tabel persediaan_apotek	Tampil data obat minimum yang terdapat pada tabel persediaan_apotek	P
15	Tambah Data Barang dan Obat	Untuk melakukan pengujian terhadap fungsi penambahan data pada tabel persediaan_apotek	Menekan link Logistik pada activites dan menekan link tambah_data_barang_dan_obat pada functions	Data yang diinputkan dalam perintah insert masuk ke tabel persediaan_apotek	Data yang diinputkan dalam perintah insert masuk ke tabel persediaan_apotek	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
16	Tambah Data Kategori	Untuk melakukan pengujian terhadap fungsi penambahan data pada tabel master_kategoriobat	Menekan link Logistik pada activites dan menekan link tambah_data_kategori pada functions	Data yang diinputkan dalam perintah insert masuk ke tabel master_kategoriobat	Data yang diinputkan dalam perintah insert masuk ke tabel master_kategoriobat	P
17	Pengajuan Retur Pengadaan	Untuk melakukan pengujian terhadap fungsi penambahan data pada tabel permohonan_retur_barang dan detail_retur_barang	Menekan link Logistik pada activites dan menekan link pengajuan_retur_pengadaan pada functions	Data yang diinputkan dalam perintah insert masuk ke tabel permohonan_retur_barang dan detail_retur_barang	Data yang diinputkan dalam perintah insert masuk ke tabel permohonan_retur_barang dan detail_retur_barang	P

bersambung

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
18	Daftar Barang Kadaluarsa	Untuk melakukan pengujian terhadap fungsi menampilkan data barang kadaluarsa dari tabel faktur_pembelian dan detail faktur	Menekan link Logistik pada activities dan menekan link daftar_barang_kadaluarsa pada functions	Tampil data barang kadaluarsa yang terdapat pada tabel faktur_pembelian dan detail_faktur	Tampil data barang kadaluarsa yang terdapat pada tabel faktur_pembelian dan detail_faktur	P
19	Daftar Permintaan Mutasi Obat	Untuk melakukan pengujian terhadap fungsi menampilkan data dari tabel permohonan_mutasi_obat	Menekan link Mutasi_Obat pada activites dan menekan link daftar_permintaan_mutasi_obat pada functions	Tampil data permintaan mutasi obat yang terdapat pada tabel permohonan_mutasi_obat	Tampil data permintaan mutasi obat yang terdapat pada tabel permohonan_mutasi_obat	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
20	Persetujuan Mutasi Obat	Untuk melakukan pengujian terhadap fungsi perubahan data pada tabel permohonan_mutasi_obat	Menekan link Mutasi_obat pada activites dan menekan link persetujuan_mutasi_obat pada functions	Data status_pelaksanaan_mutasi pada tabel penjualan_obat berubah menjadi "S"	Data status_pelaksanaan_mutasi pada tabel penjualan_obat berubah menjadi "S"	P
21	Daftar Permintaan Retur Obat	Untuk melakukan pengujian terhadap fungsi menampilkan data dari tabel retur_obat_pasien	Menekan link Retur_Obat_Pasien pada activites dan menekan link daftar_permintaan_retur_obat pada functions	Tampil data permintaan retur obat pasien yang terdapat pada tabel retur_obat_pasien	Tampil data permintaan retur obat pasien yang terdapat pada tabel retur_obat_pasien	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
22	Persetujuan Retur Obat Pasien	Untuk melakukan pengujian terhadap fungsi perubahan data pada tabel retur_obat_pasien	Menekan link Retur_Obat_Pasien pada activites dan menekan link daftar_permintaan_retur_obat pada functions	Data status pada tabel retur_obat_pasien berubah menjadi "S"	Data status pada tabel retur_obat_pasien berubah menjadi "S"	P
23	Pembuatan Laporan Resep	Untuk melakukan pengujian terhadap fungsi menampilkan data dari tabel penjualan	Menekan link Laporan pada activites dan menekan link pembuatan_laporan_resep pada functions	Tampil daftar resep yang terdapat dalam tabel penjualan	Tampil daftar resep yang terdapat dalam tabel penjualan	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
24	Pembuatan Laporan Mutasi	Untuk melakukan pengujian terhadap fungsi menampilkan data dari permohonan_ mutasi_obat	Menekan link Laporan pada activites dan menekan link pembuatan_laporan_mutasi pada functions	Tampil daftar mutasi obat yang terdapat dalam tabel permohonan_mutasi_obat	Tampil daftar mutasi obat yang terdapat dalam tabel permohonan_mutasi_obat	P
25	Pembuatan Laporan Retur Pasien	Untuk melakukan pengujian terhadap fungsi menampilkan data dari tabel retur_obat_pasien	Menekan link Laporan pada activites dan menekan link pembuatan_laporan_retur_pasien pada functions	Tampil daftar retur obat pasien yang terdapat dalam tabel retur_obat_pasien	Tampil daftar retur obat pasien yang terdapat dalam tabel retur_obat_pasien	P
26	Daftar Dokter	Untuk melakukan pengujian terhadap fungsi menampilkan	Menekan link Daftar_Dokter pada activites dan menekan link	Tampil data dokter yang terdapat pada tabel pegawai	Tampil data dokter yang terdapat pada tabel pegawai	P

bersambung

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
		data dokter dari tabel pegawai	daftar_dokter pada functions			
27	Cari Dokter	Untuk melakukan pengujian terhadap fungsi pencarian data pada table pegawai	Menekan link Daftar_Dokter pada activities dan menekan link cari_dokter pada functions	Tampil data dokter yang terdapat pada tabel pegawai	Tampil data dokter yang terdapat pada tabel pegawai	P
28	Daftar Pasien	Untuk melakukan pengujian terhadap fungsi menampilkan data pasien dari tabel pasien	Menekan link Daftar_Pasien pada activites dan menekan link daftar_pasien pada functions	Tampil data pasien yang terdapat pada tabel pasien	Tampil data pasien yang terdapat pada tabel pasien	P

bersambung

lanjutan

Test case id	Test case name	test case desc	test steps			test status (P/F)
			Input	Expected output	Actual output	
29	Cari Paien	Untuk melakukan pengujian terhadap fungsi pencarian data pada table pasien	Menekan link Daftar_Pasien pada activities dan menekan link cari_pasien pada functions	Tampil data pasien yang terdapat pada tabel pasien	Tampil data dokter yang terdapat pada tabel pasien	P

Lampiran F:
Daftar Penyelesaian Servis Modul Apotek

Halaman ini sengaja dikosongkan.

Daftar Penyelesaian Servis Modul Apotek

No. servis	Nama servis dalam kode PHP	Besar penyelesaian			No. Form DPPL
		Tidak selesai	Selesai fungsi dasar	Sudah selesai	
1	Tampil_data_resep			✓	FRM.APO.02
2	Tambah_resep		✓		FRM.APO.03
3	Tampil_data_konsumen			✓	FRM.APO.03
4	Tambah_konsumen		✓		FRM.APO.03
5	Penjualan_obat		✓		FRM.APO.03
6	Ubah_data_resep		✓		FRM.APO.04
7	Peracikan_obat			✓	FRM.APO.05
8	Racik_resep		✓		FRM.APO.06
9	Selesai_racik_resep		✓		FRM.APO.07
10	Tampil_data_barang_dan_obat			✓	FRM.APO.08
11	Cari_data_barang_dan_obat		✓		FRM.APO.08
12	Pengajuan_permintaan_barang_dan_obat		✓		FRM.APO.09
13	Tampil_usul			✓	FRM.AP

bersambung

lanjutan

No. servis	Nama servis dalam kode PHP	Besar penyelesaian			No. Form DPPL
		Tidak selesai	Selesai fungsi dasar	Sudah selesai	
	an_obat				O.09
14	Tampil_data_obat_minimum			✓	FRM.APO.09
15	Tambah_data_barang_dan_obat		✓		FRM.APO.10
16	Tambah_data_kategori		✓		FRM.APO.11
17	Pengajuan_retur_pengadaan		✓		FRM.APO.12
18	Daftar_barang_kadaluarsa			✓	FRM.APO.12
19	Daftar_permintaan_mutasi_obat			✓	FRM.APO.13
20	Persetujuan_mutasi_obat		✓		FRM.APO.14
21	Daftar_permintaan_retur_obat			✓	FRM.APO.15
22	Persetujuan_retur_obat_pasien		✓		FRM.APO.16
23	Pembuatan_laporan_resep		✓		FRM.APO.17
24	Pembuatan_laporan_mutasi		✓		FRM.APO.17
25	Pembuatan_		✓		FRM.AP

bersambung

lanjutan

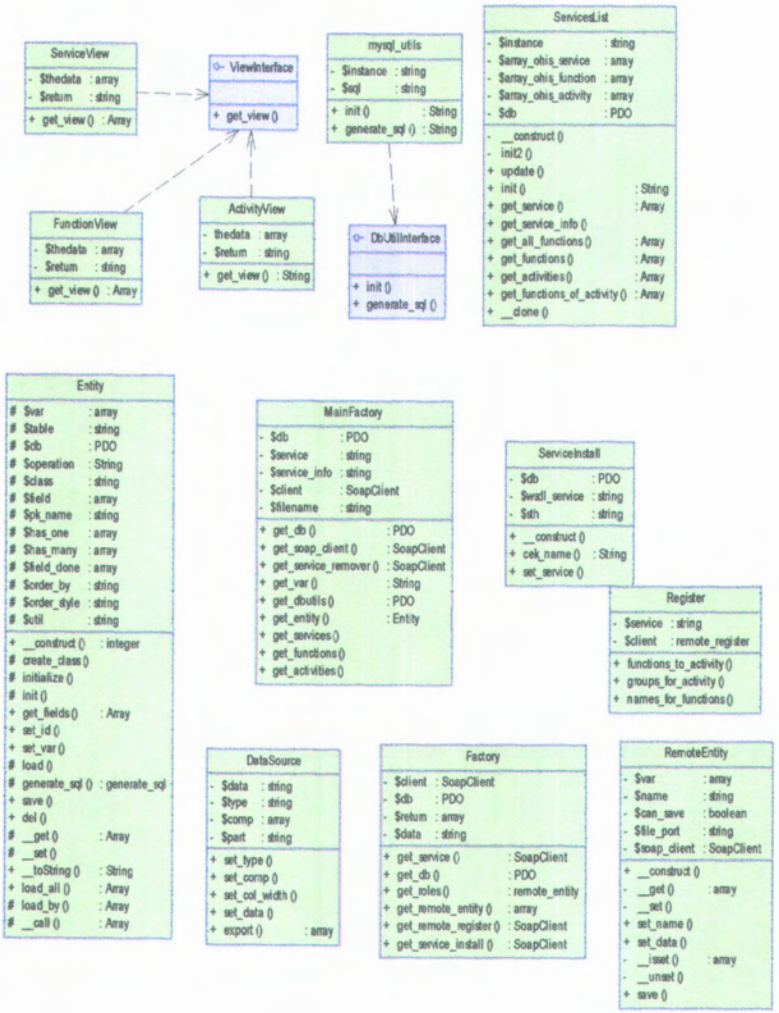
No. servis	Nama servis dalam kode PHP	Besar penyelesaian			No. Form DPPL
		Tidak selesai	Selesai fungsi dasar	Sudah selesai	
	laporan_retur_pasien				O.17
26	Daftar_dokter			✓	FRM.AP O.18
27	Cari_dokter		✓		FRM.AP O.18
28	Daftar_pasien			✓	FRM.AP O.19
29	Cari_pasien		✓		FRM.AP O.19

Halaman ini sengaja dikosongkan.

Lampiran G:
Class Diagram Framework OHIS

Halaman ini sengaja dikosongkan.

Class Diagram Framework OHIS



Halaman ini sengaja dikosongkan.

Lampiran H:
Daftar Jenis Form Modul Apotek

Halaman ini sengaja dikosongkan.

No. servis	Nama servis dalam kode PHP	Jenis Form		No. Form DPPL
		Kompleks Form	Simple Form	
1	Tampil_data_resep		✓	FRM.APO.02
2	Tambah_resep	✓		FRM.APO.03
3	Tampil_data_konsumen		✓	FRM.APO.03
4	Tambah_konsumen	✓		FRM.APO.03
5	Penjualan_obat	✓		FRM.APO.03
6	Ubah_data_resep	✓		FRM.APO.04
7	Peracikan_obat		✓	FRM.APO.05
8	Racik_resep	✓		FRM.APO.06
9	Selesai_racik_resep	✓		FRM.APO.07
10	Tampil_data_barang_dan_obat		✓	FRM.APO.08
11	Cari_data_barang_dan_obat	✓		FRM.APO.08
12	Pengajuan_permintaan_barang_dan_obat	✓		FRM.APO.09
13	Tampil_usulan_obat		✓	FRM.APO.09
14	Tampil_data_obat_minimum		✓	FRM.APO.09
15	Tambah_data_barang_d	✓		FRM.APO.10

bersambung

lanjutan

No. servis	Nama servis dalam kode PHP	Jenis Form		No. Form DPPL
		Kompleks Form	Simple Form	
	an_obat			
16	Tambah_data_kategori	✓		FRM.APO.11
17	Pengajuan_retur_pengadaan	✓		FRM.APO.12
18	Daftar_barang_kadaluarsa		✓	FRM.APO.12
19	Daftar_permintaan_mutasi_obat		✓	FRM.APO.13
20	Persetujuan_mutasi_obat	✓		FRM.APO.14
21	Daftar_permintaan_retur_obat		✓	FRM.APO.15
22	Persetujuan_retur_obat_pasien	✓		FRM.APO.16
23	Pembuatan_laporan_resep	✓		FRM.APO.17
24	Pembuatan_laporan_mutasi	✓		FRM.APO.17
25	Pembuatan_laporan_retur_pasien	✓		FRM.APO.17
26	Daftar_dokter		✓	FRM.APO.18
27	Cari_dokter	✓		FRM.APO.18
28	Daftar_pasien		✓	FRM.APO.19

bersambung

No. servis	Nama servis dalam kode PHP	Jenis Form		No. Form DPPL
		Kompleks Form	Simple Form	
29	Cari_pasien	✓		FRM.APO.19

Halaman ini sengaja dikosongkan.

BIODATA PENULIS

Ariyanto Adi Nugroho lahir di Surabaya, 6 Juni 1988, anak kedua dari tiga bersaudara dari pasangan Herijanto dan Titiek Tedjaningtyas. Penulis telah menempuh pendidikan formal yang dimulai dari TK Kuncup Dian Palembang (1993-1994), SD Negeri 438 Palembang (1994-1997), SD Negeri Bandar Lor II Kediri (1997-1999), SMP Negeri 1 Kediri (1999-2002) dan SMA Negeri 1 Kediri (2002-2005) dan terakhir sebagai mahasiswa Institut Teknologi Sepuluh Nopember Surabaya Jurusan Sistem Informasi (2005-2009).

Penulis memilih bidang minat Perencanaan dan Pengembangan Sistem Informasi sebagai bidang penelitian tugas akhirnya. Komunikasi dengan penulis dapat melalui ryan_itssi@yahoo.com.