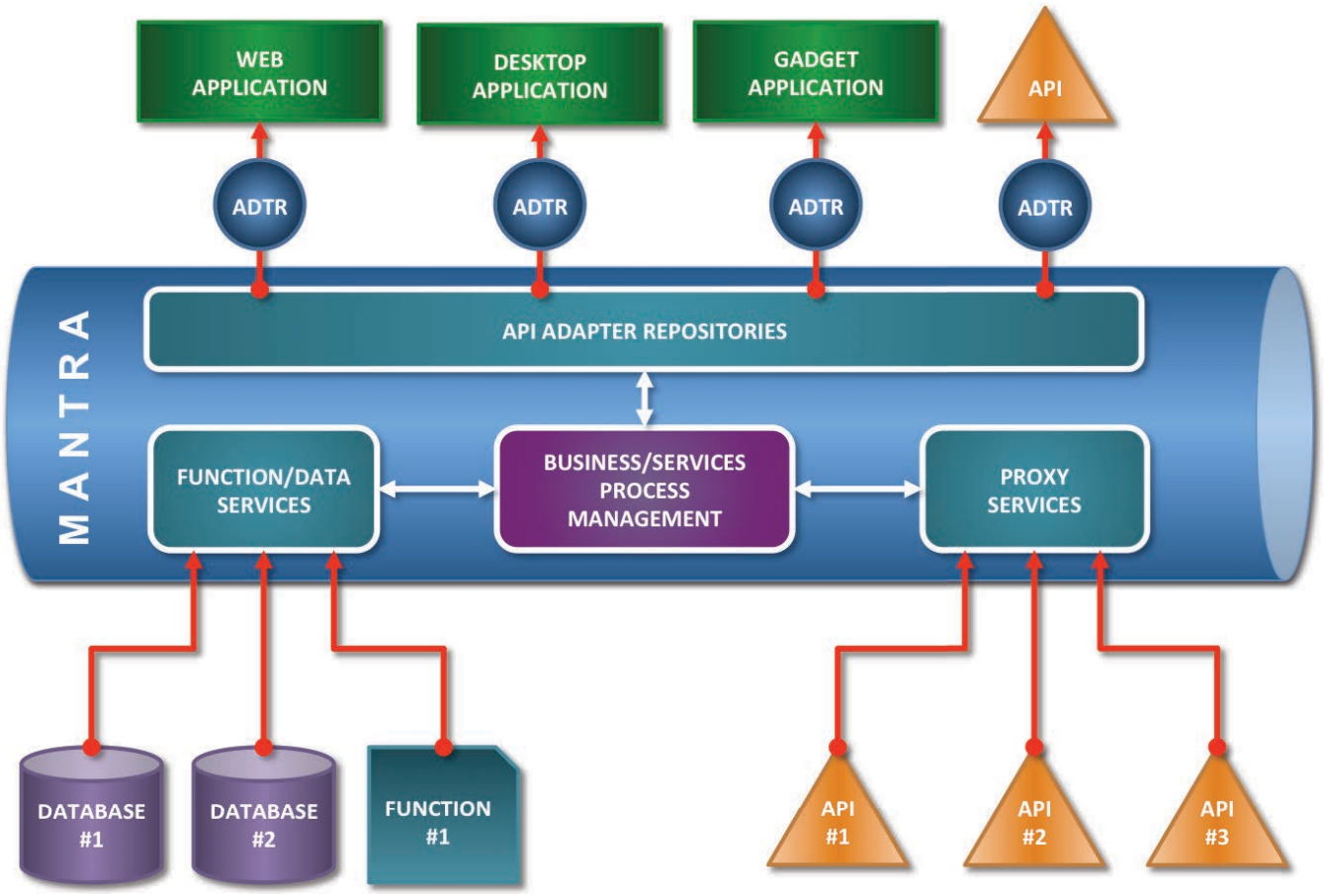
BUKU PEDOMAN

MANTRA

MANAJEMEN INTEGRASI INFORMASI DAN PERTUKARAN DATA



GOVERNMENT SERVICE BUS

DIREKTORAT E-GOVERNMENT
DIREKTORAT JENDERAL APLIKASI INFORMATIKA
KEMENTERIAN KOMUNIKASI DAN INFORMATIKA R.I.

2011

MANTRA

MANAJEMEN INTEGRASI INFORMASI DAN PERTUKARAN DATA
(GOVERNMENT SERVICE BUS)

DIREKTORAT E-GOVERNMENT
DIREKTORAT JENDERAL APLIKASI INFORMATIKA
KEMENTERIAN KOMUNIKASI DAN INFORMATIKA
2016

ISBN:

978-602-18117-5-7

PEMBINA

Ashwin Sasongko

PENGARAH

Herry Abdul Aziz

Pancat Setyantana

TIM TEKNIS

Didi Sukyadi

Agung Basuki

Alamat : Jl. Medan Merdeka Barat No.9 Jakarta

Website : http://www.aptika.kominfo.go.id/

Cetakan Tahun 2016

KATA PENGANTAR

Pembangunan dan pengembangan e-Government khususnya Perangkat Lunak Aplikasi yang berfungsi sebagai pendukung transformasi layanan pemerintah berbasis elektronik dalam wujud pengelohan data digital melalui Sistem Informasi Elektronik telah mendorong upaya peningkatan kinerja berdasarkan kebutuhan internal maupun eksternal. Namun penyediaan Sistem Elektronik pada setiap instansi pemerintah yang dilaksanakan secara lokal bahkan sporadis, mengakibatkan banyak duplikasi data dan keragaman platform teknologi serta format metadata. Keadaan ini dipicu pula dengan tidak adanya kebijakan/pedoman yang menentukan sumber data instansi mana yang ditetapkan sebagai rujukan untuk validasi atau verifikasi.

Kondisi demikian akan terus terjadi karena adanya dukungan proses pengadaan yang dilaksanakan secara lokal oleh masing-masing instansi pemerintah. Akan tetapi untuk melakukan integrasi antar sistem informasi elektronik pemerintahan, harus ada solusi dalam mengatasi masalah keragaman data dan platform teknologi, yaitu dengan menerapkan metode interoperabilitas sistem elektronik untuk berbagi pakai data dengan memanfaatkan metadata dalam format standar terbuka (XML/JSON).

Buku ini disusun sebagai pedoman untuk menjelaskan bagaimana interoperabilitas antar sistem informasi elektronik dapat diterapkan dengan menggunakan aplikasi berbasis layanan sebagai antarmuka aplikasi yang umumnya disebut Application Programming Interface (API) berbasis teknologi Web (Webservice), serta bagaimana cara mengatur integrasi berbagai API Webservice yang terhubung melalui suatu HUB dengan teknologi Government Service Bus (GSB) menjadi sebuah Orkestra layanan berbagi pakai data/informasi. API Webservice dan GSB diwujudkan menjadi aplikasi dengan nama Manajemen Integrasi Informasi dan Pertukaran Data dengan singkatan MANTRA.

Tidak lupa pula kami menyampaikan banyak terima kasih kepada semua pihak yang telah membantu dalam penyusunan buku pedoman ini, semoga bermanfaat bagi seluruh instansi pemerintah di Indonesia.

Direktur e-Government,

Herry Abdul Aziz

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

1

PENDAHULUAN

Perkembangan Aplikasi Teknologi Informasi dan Komunikasi berupa Perangkat Lunak saat ini telah dirasakan manfaatnya bagi masyarakat luas yang tersedia pada peralatan elektronik seperti Komputer, Laptop, Tablet/Pad, Smartphone, GPS dan lainnya. Peran Perangkat Lunak Aplikasi sangat penting dalam pengolahan data layanan sistem elektronik terutama bagi peningkatan manfaat dan mutu suatu sistem elektronik itu sendiri. Semakin banyak aplikasi yang dapat memenuhi kebutuhan penggunanya maka akan semakin tinggi nilai manfaat suatu sistem elektronik tersebut, contohnya pada Smartphone saat ini telah dapat menyediakan aplikasi lengkap secara user friendly bagi penggunanya berupa aplikasi perkantoran, jejaring sosial, GPS, kompas, permainan dan lain-lain.

Pemanfaatan Aplikasi pada layanan sistem elektronik pemerintahpun saat ini sudah dapat digunakan oleh masyarakat umum melalui jaringan internet dengan teknologi web, beberapa diantaranya adalah Layanan Pelaporan Pajak (SPT), Pendaftaran Paspor Imigrasi, Pendaftaran Siswa Sekolah (SD, SMP, SMA), dan Pendaftaran Pencari Kerja. Tersedianya layanan elektronik seperti ini dapat membangun hubungan baik secara transparan antara pemerintah dan masyarakat. Namun sampai sejauh ini hampir seluruh aplikasi yang disediakan belum dapat melakukan komunikasi dan transaksi dengan data yang diolah oleh aplikasi lainnya di instansi terkait. Padahal seringkali suatu aplikasi membutuhkan dua atau lebih data dari instansi lain, karena tidak bisa memperoleh data yang dibutuhkan dari instansi lain akibatnya data yang sama akan dibuat dan disediakan berulang kali secara lokal. Contohnya adalah pemanfaatan data kependudukan yang

dibutuhkan untuk referensi dan verifikasi data penduduk akan selalu ada di setiap sistem elektronik instansi pemerintah.

Hal yang paling penting dari suatu pengolahan data pada Sistem Elektronik adalah proses untuk mendapatkan referensi data, verifikasi data dan validasi data. Oleh sebab itu suatu Sistem Elektronik dikatakan sudah terintegrasi apabila dapat memperoleh beragam informasi dari Sistem Elektronik lainnya yang terhubung dalam suatu jaringan dengan beragam platform teknologi dan format data.

Bab ini akan menjelaskan suatu konsep dan arsitektur yang digunakan untuk melakukan Integrasi Informasi dan Pertukaran Data antar Sistem Elektronik melalui Antarmuka Aplikasi (Application Programming Interface/API) berbasis Web yang umumnya disebut API Webservices. Disamping itu akan dijelaskan pula konsep integrasi berbagai API Webservices melalui suatu HUB dengan teknologi Government Service Bus sebagai Orkestra Layanan Berbagi Pakai Data/Informasi.

1.1 Konsep Integrasi Sistem Elektronik

Integrasi antar Sistem Elektronik merupakan solusi umum yang selalu hadir ketika hendak menyatukan beragam Sistem Elektronik yang tersebar di beberapa lokasi. Namun dibalik alasan integrasi tersebut adalah sulitnya melakukan berbagi pakai data/informasi antar Sistem Elektronik, sehingga integrasi selalu dijadikan solusi pamungkas dalam mengatasi masalah sharing data/informasi. Pendekatan integrasi antar Sistem Elektronik pada umumya dilakukan hanya berorientasi proyek mulai dari penyediaan infrastruktur jaringan yang saling terhubung sampai rebuilding

Sistem Elektronik dengan menyatukan semua bisnis proses re-engineering dalam satu aplikasi dan mengganti Sistem Elektronik yang lama karena dianggap berdiri sendiri. Namun solusi integrasi seperti ini banyak yang gagal dilakukan karena tidak memahami permasalahan utama yang dihadapi, apalagi dengan modal yang pas-pasan.

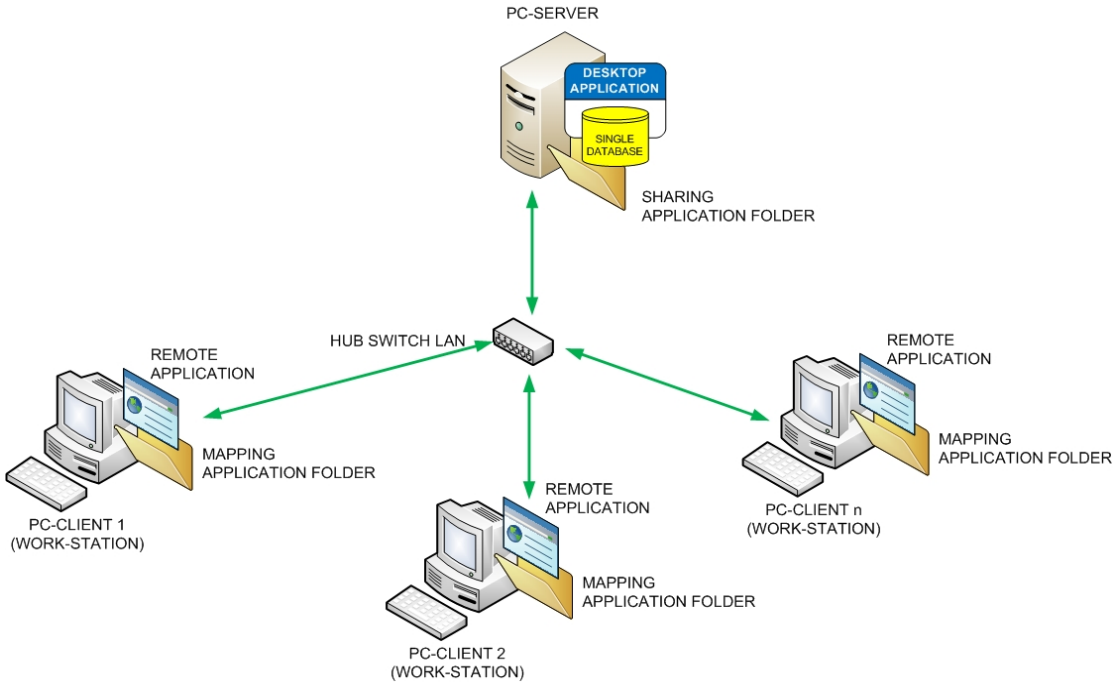
Latar belakang kondisi komunikasi data bahkan transaksi data antar Sistem Elektronik yang tidak dapat dilakukan secara terintegrasi adalah karena awalnya Sistem Elektronik dibangun hanya untuk kebutuhan pengolahan data secara lokal. Keadaan ini didukung pula dengan keberadaan teknologi yang awalnya hanya mampu untuk menangani area lokal pada saat Sistem Elektronik dibangun dan teknologi File Sharing atau berbagi berkas yang disediakan melalui suatu Directory adalah layanan yang paling populer pada Sistem Operasi Server disamping message (pesan).

Integrasi antar Sistem Elektronik sebaiknya tidak dipahami hanya untuk pengadaan infrastruktur jaringan dan penyatuan bisnis proses re-engineering saja, akan tetapi proses komputasi terdistribusi adalah hal yang perlu diperhatikan. Dengan adanya Distrubuted Computing (komputasi terdistribusi) maka akan mengurangi beban kerja server dan infrastruktur jaringan.

Berdasarkan perkembangan teknologi, komputasi terdistribusi dibagi menjadi beberapa fase berikut ini:

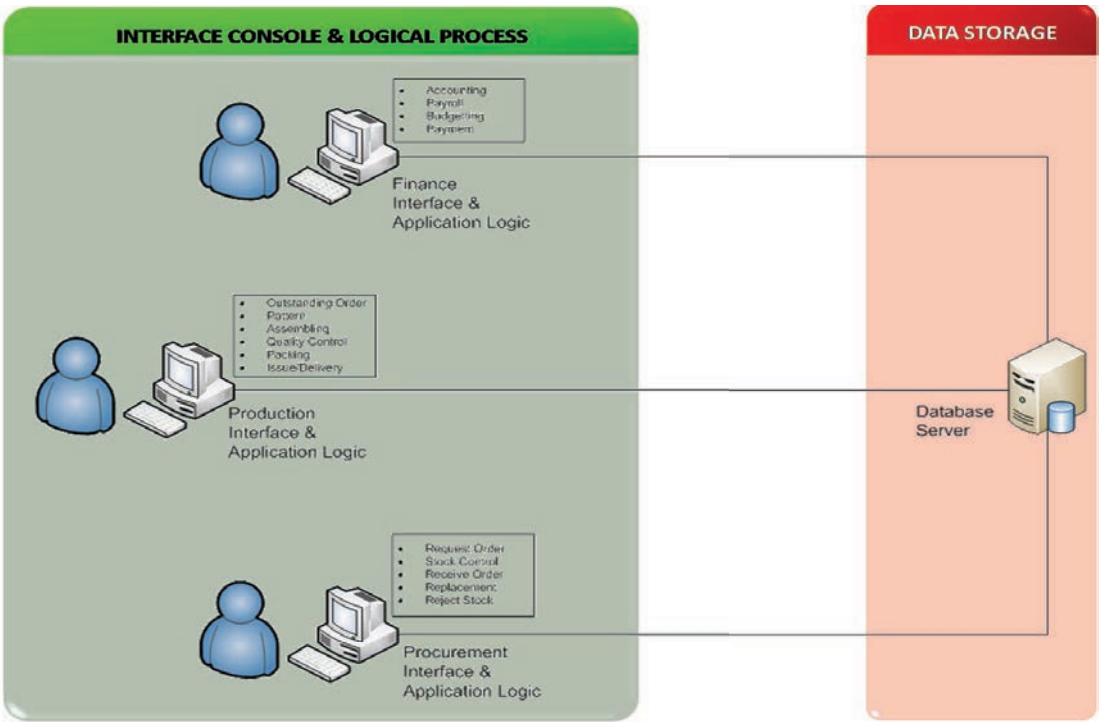
a. Fase berbagi pakai aplikasi, pada fase ini Sistem Elektronik yang dikembangkan hanya mampu menangani jaringan lokal (LAN), sehingga solusi yang dibuat saat itu adalah menyediakan infrastruktur jaringan secara lokal pada ruang kantor pengolah data dengan memanfaatkan aplikasi pada

satu server secara bersama-sama untuk mengolah data terpusat melalui aplikasi dimana suatu interface console yang berada pada sebuah server dapat diakses bersama-sama oleh pc-client (workstation) sementara logical process dan database ditempatkan pada satu lokasi yang sama dengan aplikasi. Pada fase ini tidak membutuhkan pengamanan serius, karena hanya dikelola untuk satu area atau segmen jaringan dan diterapkan pada satu platform sistem operasi dengan cara memetakan alamat lokasi aplikasi (drive path). Dikarenakan keterbatasan kemampuan operasional seperti memakan banyak sumberdaya memori kerja baik di lokasi server maupun pc-client (workstation), mengakibatkan sering terjadinya deadlock transaksi data saat terjadinya antrian proses dari beberapa pc-client ke server. Topologi yang digunakan pada fase ini tergambar dalam ilustrasi berikut ini.



Gambar 1.1 Topologi berbagi pakai aplikasi

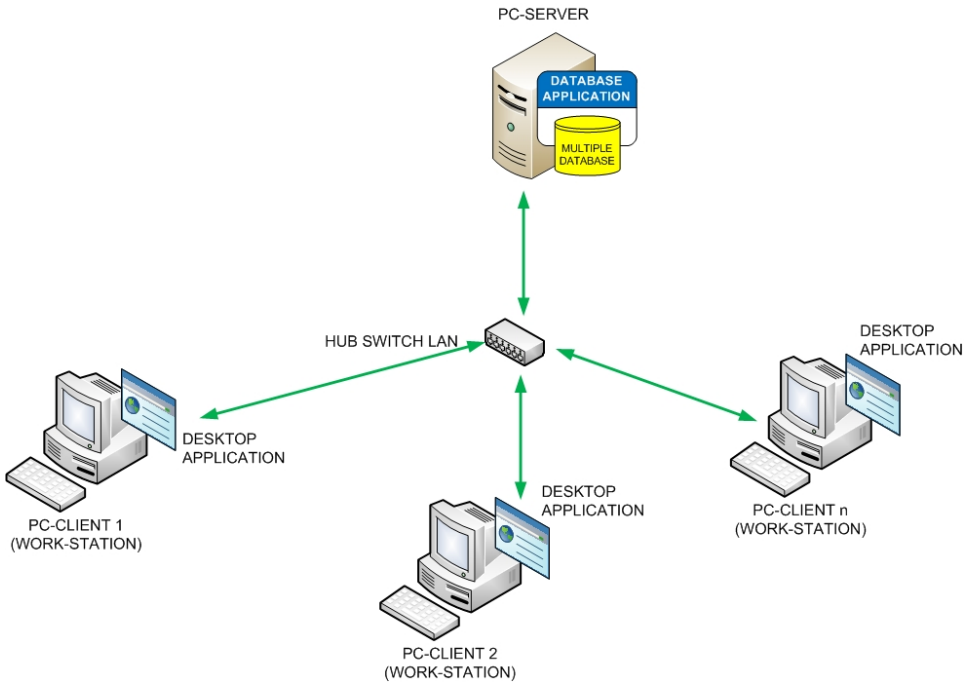
b. Fase berbagi pakai database, pada fase ini Sistem Elektronik dikembangkan untuk mengurangi beban kerja server, sehingga bagian interface console dan logical process dipisahkan dan didistribusikan ke pc-client agar sumberdaya memori logika pengolahan data (logical process) dapat optimal dan server hanya bekerja untuk pengolahan manajemen database saja. Konsep ini umumnya dinamakan konsep Distributed Computing model 2 (Two)-tier seperti diagram berikut ini.



Gambar 1.2 Konsep Distrubuted Computing Model 2 (Two)-tier

Penerapan konsep Distributed Computing model 2 (Two)-tier memiliki hambatan ketika melakukan perbaikan atau pengembangan interface console dan logical process. Apabila ada perbaikan atau pengembangan logical process maupun interface console maka versi terbaru harus dipasang dan diupdate ke semua pc-client (workstation), hal ini akan

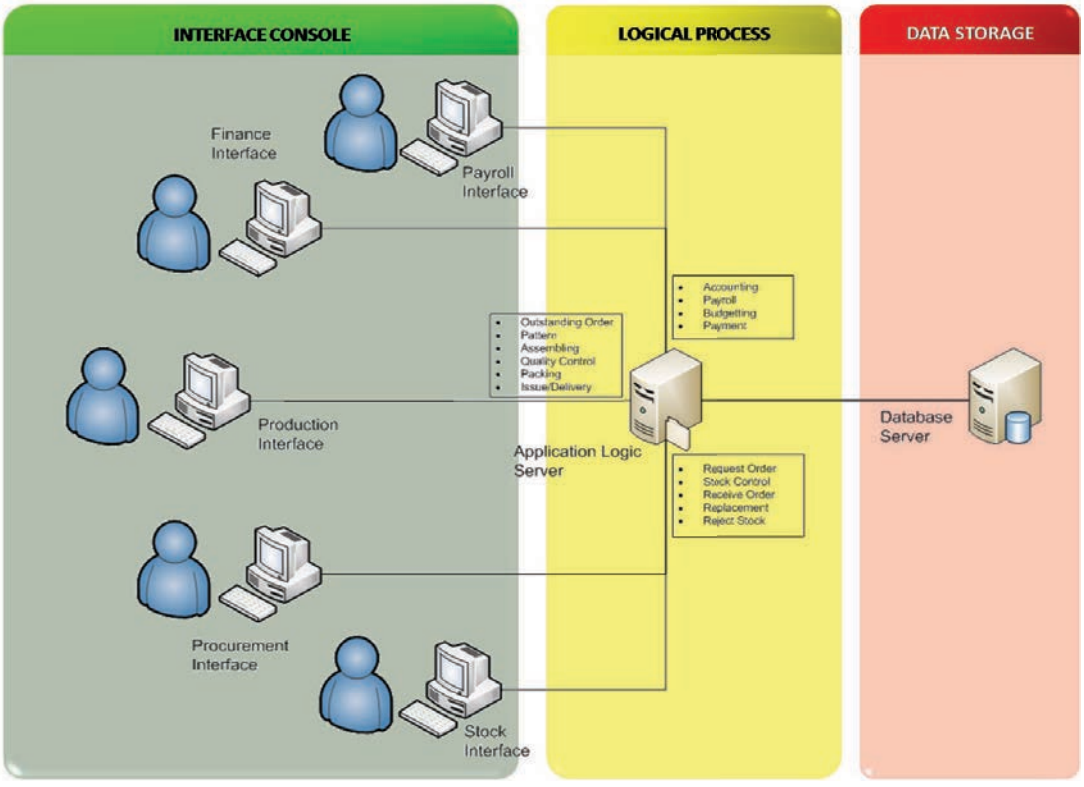
memakan waktu pemutakhiran pada jumlah workstation yang banyak. Terkadang proses update interface console maupun logical process mengalami kegagalan karena versi apikasi harus lebih baru dari sebelumnya.



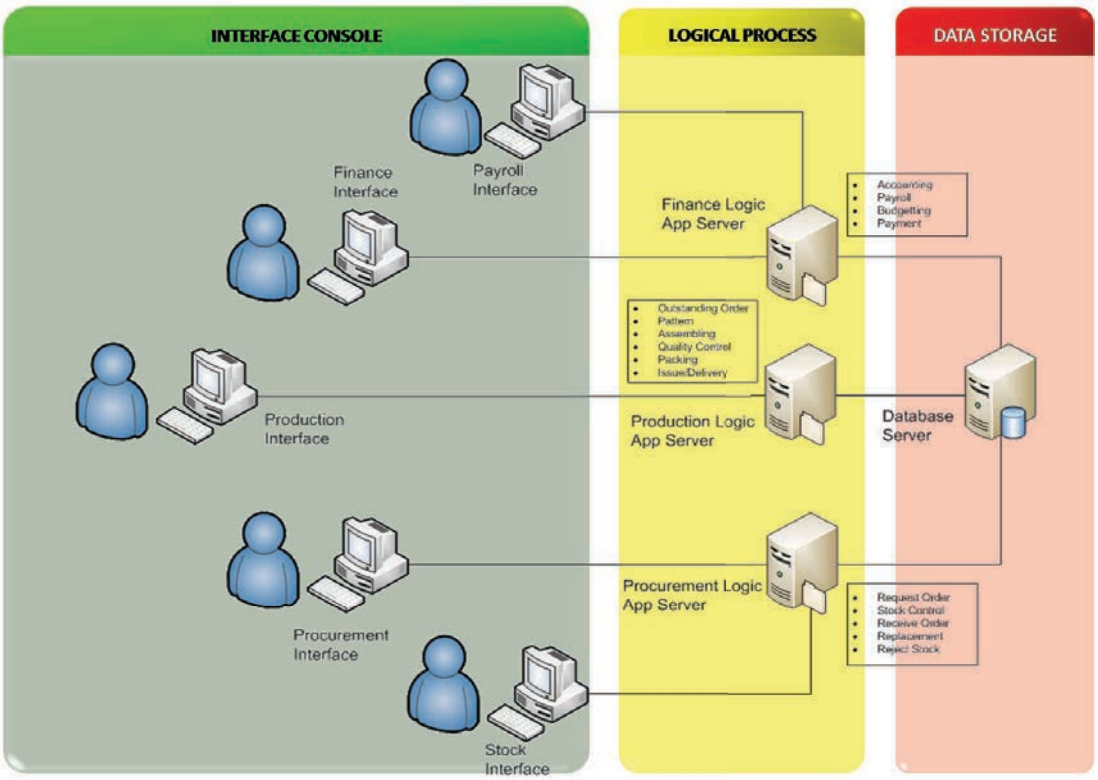
Gambar 1.3 Topologi berbagi pakai database

c. Fase berbagi pakai layanan aplikasi/data, pada fase ini suatu logical process dipisah dari lokasi interface console agar logical process dapat digunakan bersama oleh interface console lainnya yang berbeda modul. Konsep ini disebut model 3 (Three)-tier dimana pengolahan data dibagi menjadi 3 bagian terpisah, yaitu aplikasi antarmuka pengguna (interface console) yang umumnya berbasis grafik berada di lokasi workstation, sedangkan logika pengolahan data (logical process) terpisah pada server aplikasi. Sementara itu database bisa saja berada pada lokasi yang sama atau berbeda dengan server aplikasi. Pemisahan logical process dari interface console menjadi middleware dapat

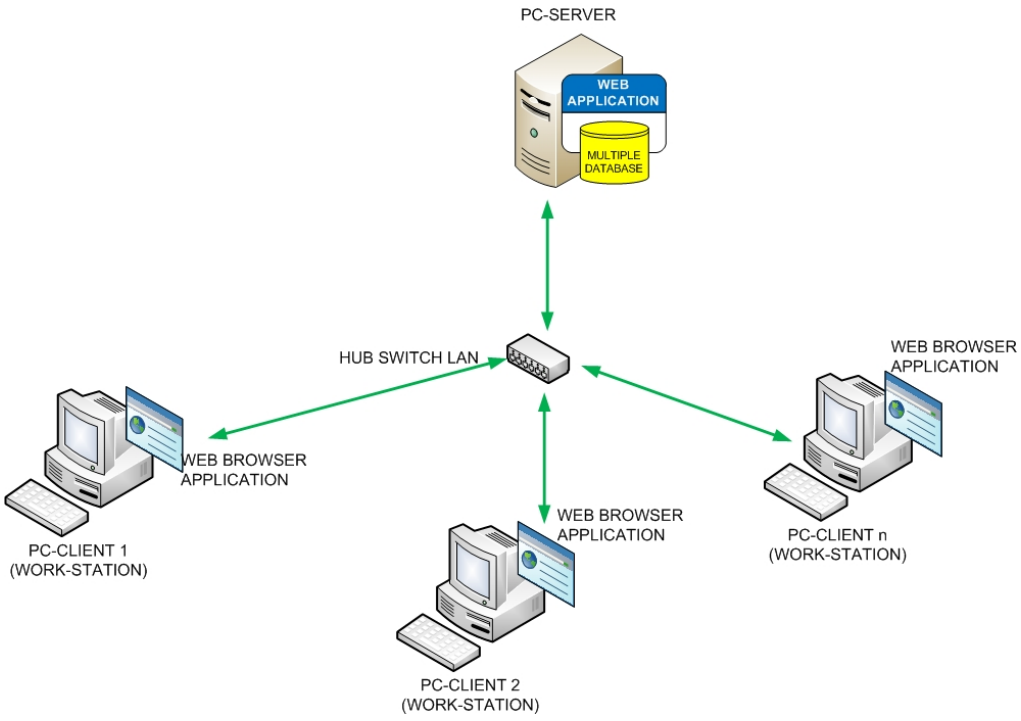
dikembangkan menjadi konsep Multi-tier (N-tier), dimana logical process dapat dibuat lebih dari satu sesuai kebutuhan sektoral misalnya untuk bidang perencanaan, keuangan, produksi dll. Logical process ini secara utuh menjadi aplikasi berbasis layanan yang dapat digunakan berulang kali secara bersama-sama. Dampak dari adanya aplikasi berbasis layanan adalah dapat mengurangi duplikasi fungsi untuk setiap sektor pengolahan data. Perubahan, perbaikan dan pengembangan logical process sebagai aplikasi berbasis layanan dapat dilakukan tanpa mengganggu transaksi data pada sektor lainnya. Berikut adalah diagram yang membedakan model Three-tier dengan Multi-tier.



Gambar 1.4 Konsep Distrubuted Computing Model 3 (Three)-tier



Gambar 1.5 Konsep Distrubuted Computing Model N (Multi)-tier



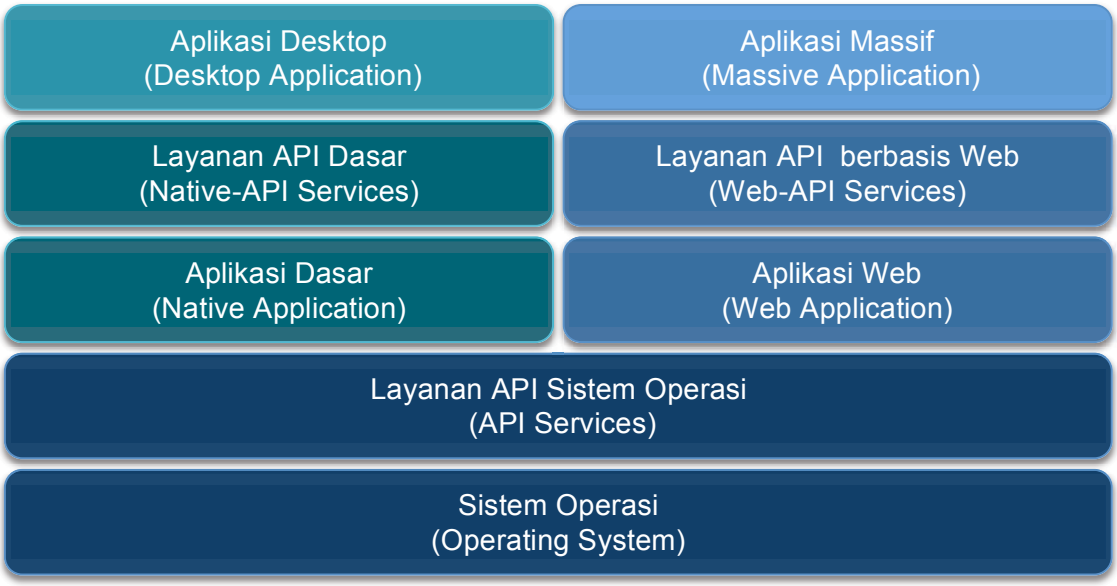
Gambar 1.6 Topologi berbagi pakai layanan aplikasi/data

Saat ini fase Komputasi Terdistribusi yang paling relevan digunakan adalah fase ke-3 yaitu fase berbagi pakai layanan aplikasi/data dengan mengedepankan aplikasi yang berbasis layanan. Beberapa teknologi pendukung interkoneksi dan komunikasi data pada aplikasi berbasis layanan dengan konsep Distributed Computing model 3-tier atau N-tier diantaranya adalah CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model), Java-RMI (Remote Method Invocation), dan dotNET Remoting. Teknologi tersebut menggunakan konsep Distributed Object yang berfungsi sebagai media pendukung aplikasi berbasis layanan melalui sharing logical process yang didefinisikan dalam aplikasi berbasis layanan dengan sebutan Antramuka Pemrograman Aplikasi (Application Programming Interfaces/API).

API yang dibuat sebagai pintu gerbang logical process dapat memanfaatkan platform teknologi yang disediakan setiap vendor, namun pada implementasinya API tersebut memiliki ketergantungan penuh terhadap platform dari vendor tertentu saja, sehingga sulit untuk diterapkan pada platform yang berbeda sebagai pendukung terlaksananya interoperabilitas antar sistem informasi elektronik yang memiliki keragaman informasi secara terbuka.

API pada dasarnya sudah tersedia mulai pada lapisan Sistem Operasi, sampai kepada lapisan Desktop/Window Management. Masing-masing lapisan memiliki ruang lingkup layanan yang diberikan kepada aplikasi pengguna yang ada di lapisan atasnya, dimana setiap lapisan memiliki logical process yang saling terkait dengan fungsi-fungsi yang disediakan aplikasi tertentu agar dapat digunakan berulang secara bersama-sama. Contohnya untuk melakukan koneksi internet dari suatu aplikasi di lapisan Desktop maka cukup dengan memanfaatkan API konektor internet yang ada di lapisan Desktop saja koneksi bisa dilakukan, karena proses

selanjutnya akan dilakukan oleh logical process di lapisan bawahnya sampai ke layanan Sistem Operasi dan Perangkat Keras sebagaimana gambaran berikut:



Gambar 1.7 Lapisan Layanan API dalam Sistem Komputasi

Pada setiap sistem komputasi memiliki lapisan layanan yang merupakan fasilitas berbagi fungsi dan operasi yang disediakan aplikasi tertentu untuk dapat dipergunakan aplikasi lainnya melalui akses layanan API. API yang berada di tiap lapisan terdiri dari:

a. Layanan API (API Services) pada Sistem Operasi disediakan oleh Sistem Operasi agar dapat dimanfaatkan oleh aplikasi yang berjalan diatas Sistem Operasi, misalnya aplikasi pengolah gambar akan menggunakan layanan API Sistem Operasi untuk melakukan Scan Gambar dan Cetak Gambar. Contoh lainnya adalah layanan untuk Web (Apache HTTP Server, IIS, BOA, dan lain-lain) yang bertugas mengelola dan mengatur pengolahan data Web untuk didistribusikan pada jaringan komputasi yang dilengkapi dengan modul praprosesor (server-side script) seperti PHP, ASP, JSP, dan sebagainya.

b. Layanan API pada Aplikasi Dasar disediakan oleh Aplikasi Dasar yang berjalan diatas Sistem Operasi untuk digunakan oleh Aplikasi Dasar lainnya bahkan aplikasi berbasis desktop grafis di lapisan atasnya, misalnya Aplikasi Dasar yang menyediakan fungsi grafis seperti manajemen window, manajemen warna, manajemen animasi dan sebagainya.
c. Layanan API pada Aplikasi berbasis Web disediakan oleh Aplikasi Web yang berjalan diatas Layanan Web suatu Sistem Operasi. Layanan API ini berupa server-side script yang dibentuk oleh praprosesor PHP, ASP atau JSP.

Sejalan dengan tuntutan kebutuhan pengembangan aplikasi perangkat lunak untuk jaringan internet, maka teknologi Web menjadi pilihan teknologi aplikasi berbasis layanan yang mudah dikembangkan menjadi API. Hadirnya teknologi Web membawa perubahan pada konsep Distributed Computing, dimana ketergantungan terhadap suatu platform dapat diatasi melalui penggunaan teknologi standar terbuka sehingga Distributed Computing dapat dilakukan pada berbagai platform. Konsep Distributed Computing model Three-tier dan Multi-tier dapat dikombinasikan dengan model Web, dimana Aplikasi Web Browser (interface console) sebagai presentation-tier, Aplikasi Web Server (logical process) sebagai application-tier dan DBMS (database) sebagai data-tier. Melalui pemanfaatan teknologi Web dalam Distributed Computng diperoleh beberapa keunggulan dibandingkan teknologi dasar, diantaranya:

a. Aplikasi (interface console) dapat disediakan pada berbagai platform.
b. Interkoneksi data menggunakan protokol standar terbuka yaitu HTTP (Hyper Text Transport Protocol).

c. Logical process disediakan secara dinamis dan dapat diakses oleh interface console pada berbagai platform sistem (multi-platform).
d. Data yang diperoleh dari beragam bentuk, format dan struktur dapat dikonversi dalam format standar terbuka yang tidak memiliki ketergantungan pada vendor tertentu seperti HTML.
e. Informasi disajikan dalam format standar yang dikenal oleh semua aplikasi Web Server dan Web Browser.

Bentuk kongkrit dari API adalah suatu program aplikasi yang berfungsi sebagai media perantara atau penghubung (antarmuka) yang bertugas menerima permintaan data dan memberikan informasi/data dengan format akses dan informasi/data yang disepakati kedua belah pihak (pemohon dan penyedia). Pada prinsipnya API merupakan program perantara yang berisi subrutin/subproses pengolahan data dan digunakan untuk memperoleh data dari jarak jauh atau diluar area aplikasi yang digunakan, awalnya cara ini disebut Remote Procedure Call (RPC). Dikarenakan subrutin/subproses program pengolah data yang disediakan dalam jumlah banyak dan tersebar dalam berkas aplikasi yang berbeda maka RPC dibedakan penamaanya menjadi API. Agar aplikasi berbasis layanan dapat memenuhi kriteria interoperabilitas maka API harus dibuat dengan kode sumber terbuka (open source), sehingga dapat mendefinisikan konektor dan adapter yang bisa disisipkan pada program aplikasi penggunanya.

Jadi API tidak hanya dapat ditempatkan dalam Layanan Sistem Operasi saja, melainkan bisa ditempatkan sebagai Layanan Web sehingga disebut API berbasis Web atau API Webservices (Web-API). Penamaan Web-API terkadang lebih mudah disebut Webservice, namun sering menemui kerancuan dengan penamaan

Website karena Website juga bagian dari Webservice. Untuk itu perlu diketahui bahwa Website dan Webservice adalah dua hal yang berbeda seperti dapat diterangkan dalam tabel 1.1 berikut ini.

Tabel 1.1 Perbedaan Website dan Webservice

Website	Webservice
Diakses oleh manusia melalui Aplikasi Browser	Diakses oleh program aplikasi pengolah data
Tanggapan berupa data format HTML, Plain Text, dll	Tanggapan berupa data format XML, JSON, RSS, dll
Digunakan untuk menampilkan data/informasi ke layar aplikasi browser	Digunakan untuk komunikasi data antar beragam platform sistem aplikasi

Menurut W3C definisi Webservice adalah sebuah piranti lunak aplikasi yang dapat teridentifikasi oleh URI (Uniform Resource Identifie) dan memiliki interface yang didefiniskan, dideskripsikan, dan dimengerti dalam format XML (eXtensible Markup Language) dan juga mendukung interaksi langsung dengan software aplikasi yang lain dengan menggunakan pesan (message) berbasis XML melalui protokol internet. Webservice adalah perangkat lunak yang tidak terpengaruh oleh platform, ia akan menyediakan fungsi-fungsi (method-method) yang dapat diakses dalam jaringan (network) dan juga akan menggunakan format terbuka XML untuk pertukaran data, khususnya pada dua entitas yang berbeda.

Webservice memiliki beberapa karakteristik sebagai berikut:

- Message-based, informasi/data yang dilewatkan melalui Webservice adalah berbasis pesan.
- Standard-based, komunikasi data dalam Webservice menggunakan terknologi standar terbuka.

- Programming language independent, bahasa pemrograman yang disediakan Webservice dapat diakses dimana saja.
- Platform-neutral, Platform teknologi memiliki daya dukung dimana saja.

Konsep Integrasi Informasi dan Pertukaran Data yang akan dijelaskan dalam buku pedoman ini dilandasi pada kenyataan layanan sistem elektronik yang ada dengan membedakannya pada interaksi yang dilakukan antar pengguna dan sistemnya. Interaksi dalam sistem pelayanan e-Government diklasifikasikan berdasarkan tingkat kemampuannya, yaitu:

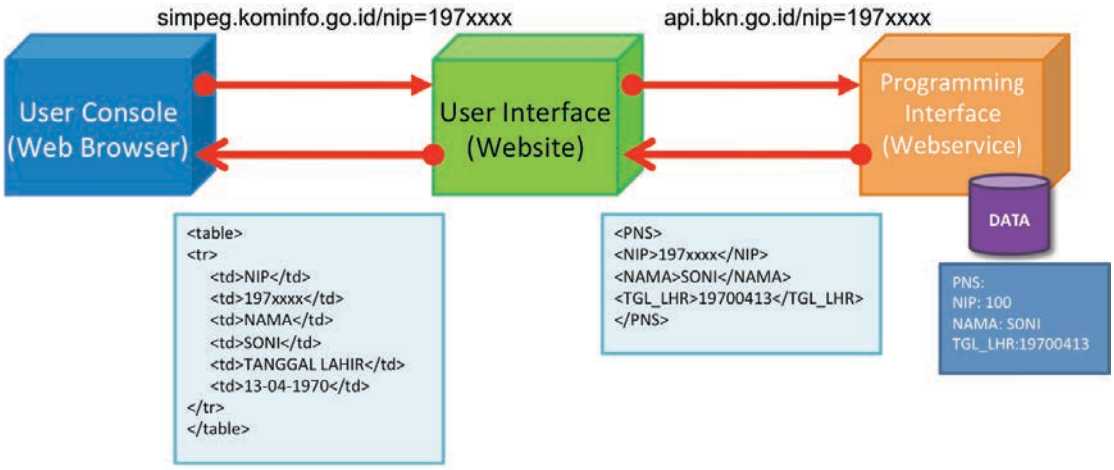
- Level 1, pelayanan online hanya memberikan informasi, pengguna cukup membaca informasi secara online atau mengunduhnya.
- Level 2, tersedianya formulir secara online yang dapat diunduh, kemudian dikembalikan melalui pos, fax atau e-mail.
- Level 3, adanya transaksi individu antara pengguna dan pemberi layanan sehingga formulir dapat diisi dan disubmit secara online.
- Level 4, multiple transaksi yang mungkin dilakukan karena pelayanan sudah terintegrasi antar berbagai lembaga pemerintah dan menunjang pengolahan data otomatis antar mesin atau antar aplikasi.

Secara garis besarnya dapat diumpamakan bahwa level 1 dan 2 menitik beratkan pada pelayanan loket yang belum memiliki proses elektronik, sedangkan pada level 3 dan 4 sudah didukung oleh proses pengolahan data elektronik secara langsung. Penerapatan Integrasi Informasi dan Pertukaran Data adalah peluang yang dapat diberikan bila sudah ada bentuk interaksi layanan e-Government pada level 3 dan 4.

Pentingnya Penerapan Integrasi Informasi dan Pertukaran Data, antara lain:

- Integrasi Informasi dan Pertukaran Data diperlukan karena adanya saling keterkaitan secara rutin antar suatu sistem atau sub sistem pengolahan data digital dengan sistem atau sub sistem pengolahan data digital lainnya.
- Upaya Integrasi Informasi dan Pertukaran Data antar instansi pemerintah perlu dilakukan melalui proses koordinasi antar instansi pemerintah, dengan mengedepankan satu visi bahwa setiap data pemerintah yang saling terkait dapat dimanfaatkan bersama antar instansi pemerintah dengan menyediakan layanan berbagi pakai data/informasi melalui Antarmuka Program Aplikasi (API) di masing-masing instansi.

Bila dikaitkan dengan penerapan Integrasi Informasi dan Pertukaran Data, API Webservice digunakan saat akan mentransformasikan suatu bisnis-logic/class/object/data yang terpisah dan berjauhan lokasi ke dalam 1 (satu) jaringan terintegrasi agar mudah memperoleh referensi data/informasi bagi kepentingan verifikasi dan validasi data. Dengan adanya Webservice maka untuk melakukan daur ulang (upgrade) dan distribusi data (deployment) tidak diperlukan lagi registrasi ke dalam sistem operasi, Webservice cukup diunggah (upload) ke direktori Web Engine/Server kemudian siap diakses oleh aplikasi lainnya yang telah diberi otorisasi melalui koneksi protokol standar HTTP pada port 80. Penggunaan protokol HTTP port 80 akan mengurangi resiko terblokir oleh firewall, kendala ini akan terjadi bila menggunakan teknologi CORBA, COM/DCOM, atau Socket Server lainnya yang sangat tergantung pada platform teknologi tertentu. Contoh proses verifikasi data melalui Webservice dari Aplikasi Website dapat digambarkan pada ilustrasi berikut ini.



Gambar 1.8 Layanan Berbagi Pakai Data melalui Webservice

Interaksi yang ditunjukan pada gambaran tersebut adalah proses verifikasi data PNS yang tersedia di Server BKN diakses dari Aplikasi Simpeg Kemkominfo sesuai permintaan Pengguna melalui Web Browsernya. Antara Data dan API yang digunakan pada layanan tersebut memiliki platform yang berbeda dengan Aplikasi Web yang memanfaatkannya namun dapat terhubung dan mengakses data melalui teknologi Webservice. Kenyataan ini menunjukan bahwa Webservice juga menawarkan banyak kelebihan dan fleksibilitas antara lain:

- Lintas platform, memungkinkan komputer-komputer yang berbeda sistem operasi dapat saling bertukar data.
- Language independent, dapat diakses menggunakan bahasa pemrograman apa saja.
- Jembatan penghubung dengan Database, pada umumnya sebuah aplikasi memerlukan driver database agar bisa melakukan koneksi ke sebuah database. Webservice dapat dijadikan sebagai jembatan penghubung antara aplikasi dengan database tanpa memerlukan driver database dan tidak perlu tahu apa jenis

DBMS yang digunakan pihak penyedia data. Aplikasi tersebut cukup mengetahui fungsi apa saja yang disediakan Webservice.

- Mempermudah proses pertukaran data, penggunaan Webservice dapat mempermudah dan mempercepat pertukaran data daripada harus menyesuaikan platform aplikasi atau database.
- Penggunaan kembali (daur ulang) komponen aplikasi, hal ini disebabkan beberapa aplikasi yang berbeda bisa saja memerlukan sebuah fungsi/subrutin/subproses yang sama.

Dengan demikian maka Webservice merupakan pilihan teknologi yang paling ideal dan mudah diterapkan dalam proses Integrasi Informasi dan Pertukaran Data secara heterogen sebagai upaya memberikan Layanan Berbagi Pakai Data/Informasi antar Sistem Elektronik. Dalam prakteknya Webservice memiliki 2 (dua) konsep teknologi, yaitu SOAP dan REST. Masing-masing memiliki keunggulan dan kelemahan sesuai keperuntukannya.

1.1.1 Konsep SOAP

SOAP (Simple Object Access Protocol) merupakan suatu protokol standar format dokumen terbuka berupa XML yang digunakan untuk berbagi pakai data atau pertukaran data (request dan response) antar Aplikasi berbasis Web. Konsep dengan metode SOAP awalnya dikembangkan untuk mendukung aplikasi desktop yang lama (legacy system) dengan akses berbagai macam protokol seperti CORBA, RMI, COM/DCOM agar dapat menyediakan layanan berbagi pakai data melalui jaringan internet. SOAP juga merupakan protokol substitusi secara logika dari protokol komunikasi secara fisik yang tersedia untuk komunkasi data antar jaringan. Banyak pendapat yang menganggap bahwa SOAP adalah Webservice untuk

skala besar, namun kenyataanya SOAP sendiri memiliki kemampuan terbatas pada Memory.

Berikut adalah gambaran mekanisme akses Webservice dengan konsep SOAP.

Gambar 1.9 Layanan Berbagi Pakai Data dengan konsep SOAP

SOAP berperan sebagai protokol berupa aplikasi yang mengatur dokumen pada API berbasis Web. Dokumen SOAP yang digunakan untuk request ke Webservice disebut SOAP-Request, sedangkan dokumen SOAP yang diperoleh dari Webservice disebut SOAP-Response. Setiap layanan berbagi pakai data diatur oleh SOAP yang mengelola dokumen XML, dimana format XML berperan sebagai bahasa yang digunakan, sementara format SOAP-XML berperan sebagai tata bahasa yang dibungkus dalam dokumen dan diberi amplop (SOAP-Envelope) untuk pengamanan pengiriman. Pada prinsipnya Webservice SOAP menggunakan dokumen XML dalam layanan berbagi pakai data yang berfungsi sebagai informasi tentang data yang diminta (request) dan data yang dihasilkan (response).

Struktur standar dokumen SOAP-XML terdiri dari SOAP-Envelope (Amplop) untuk identifikasi dokumen XML sebagai sebuah pesan, di dalam SOAP-Envelope memiliki 2 elemen dasar, yaitu:

- SOAP-Header yang berisi informasi referensi dokumen
- SOAP-Body berisi informasi tentang request dan response satu atau lebih method.

Webservice SOAP memandang suatu fungsi/subproses dalam layanan sebagai method. Informasi tentang isi suatu Webservice SOAP dapat diperoleh melalui akses ke halaman dokumen WSDL (Web Service Description Language) yang merupakan dokumen XML dengan penjelasan informasi rinci sebuah Webservice diantaranya method/fungsi/subrutin/subproses apa saja yang disediakan berikut parameter dan tipe data hasil yang dikembalikan atau dikeluarkan suatu method Webservice. WSDL tersebut akan terbentuk dengan sendirinya pada saat Webservice didefinisikan menggunakan program SOAP, sehingga cukup dilakukan pendefinisian Webservice dan methodnya saja maka pada saat alamat Webservice diakses dari Web Browser akan tampil halaman informasi yang menjelaskan tentang Webservice dalam bentuk tampilan WSDL.

Bentuk Webservice SOAP yang dibuat dalam bahasa pemrograman PHP dengan program pendukung SOAP yaitu NuSOAP adalah sebagai berikut.

File: apisoap.php

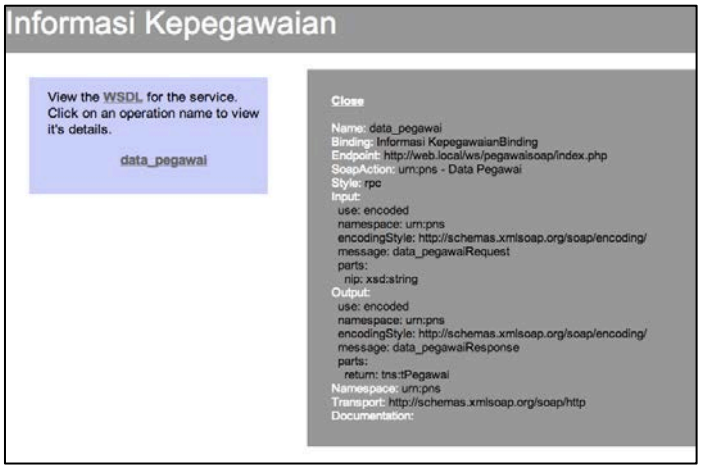
```
<?php // Programmed by: Didi Sukyadi
require_once 'lib/nusoap/nusoap.php';
$api=new nusoap_server();
$api->configureWSDL('Informasi Kepegawaian','urn:pns');
$api->wsdl->addComplexType(
      'tPegawai','complexType','struct','all','',
      array('nip'=>array('desc'=>'Nomor Induk Pegawai',
                         'type'=>'xsd:string'),
            'nama'=>array('desc'=>'Nama Pegawai',
                         'type'=>'xsd:string'),
            'kelahiran'=>array('desc'=>'Tempat Lahir',
                         'type'=>'xsd:string'),
            'tgl_lahir'=>array('desc'=>'Tanggal Lahir',
                         'type'=>'xsd:string')
      )
);

$api->register(
      "data_pegawai",
      array("nip"=>'xsd:string'),
      array("return"=>'tns:tPegawai'),
      'urn:pns','urn:pns - Data Pegawai'
);
```

```
$HTTP_RAW_POST_DATA=isset($HTTP_RAW_POST_DATA)?
                    $HTTP_RAW_POST_DATA:'';
$api->service($HTTP_RAW_POST_DATA);


function data_pegawai($nip){
      $datapegawai=array(
            "0"=>array("nip"=>"197811242009041002",
                        "nama"=>"Siswanto",
                        "kelahiran"=>"Bandung",
                        "tgl_lahir"=>"24/NOP/1978"),
            "1"=>array("nip"=>"197403102010031001",
                        "nama"=>"Linda",
                        "kelahiran"=>"Jakarta",
                        "tgl_lahir"=>"10/MAR/1974"),
            "2"=>array("nip"=>"197506152010031002",
                        "nama"=>"Damar",
                        "kelahiran"=>"Surabaya",
                        "tgl_lahir"=>"15/JUN/1975"),
            "3"=>array("nip"=>"197608182011041001",
                        "nama"=>"Hendri",
                        "kelahiran"=>"Medan",
                        "tgl_lahir"=>"18/AGS/1976")
      );
      foreach($datapegawai as $id=>$data){
            if($data["nip"]==$nip){
                  return $data;
                  break;
            }
      }
}
```

Program apisoap.php membentuk defenisi Webservice dengan method/fungsi untuk mencari data pegawai berdasarkan NIP. Program tersebut cukup ditempatkan dalam Web Server Apache pada direktori /var/www/ws/ dan dipetakan ke dalam alias dari alamat host URL servernya yang didefinisikan dalam berkas konfigurasi Web Server apache2.conf dengan tahapan sebagai berikut:

```
$sudo nano apache2.conf
alias /ws “/var/www/ws/”
<directory “/var/www/ws/”>
      options indexes followsymlinks
      allowoverride none
      order allow,deny
      allow from all
</directory>

$sudo /etc/init.d/apache2 restart
```

Webservice SOAP yang sudah ditempatkan pada Web Server Apache dengan IP 192.168.1.20 tersebut dapat diakses/dibuka dari Web Browser Komputer lain (PC-Client) dengan alamat URL http://192.168.1.20/ws/apisoap.php seperti ilustrasi berikut.



Gambar 1.10 Halaman depan Webservice apisoap.php

Sementara itu informasi WSDL dapat dilihat dari Web Browser dengan alamat akses http://192.168.1.20/ws/apisoap.php?wsdl sebagaimana ilustrasi berikut ini.

Gambar 1.11 Halaman WSDL

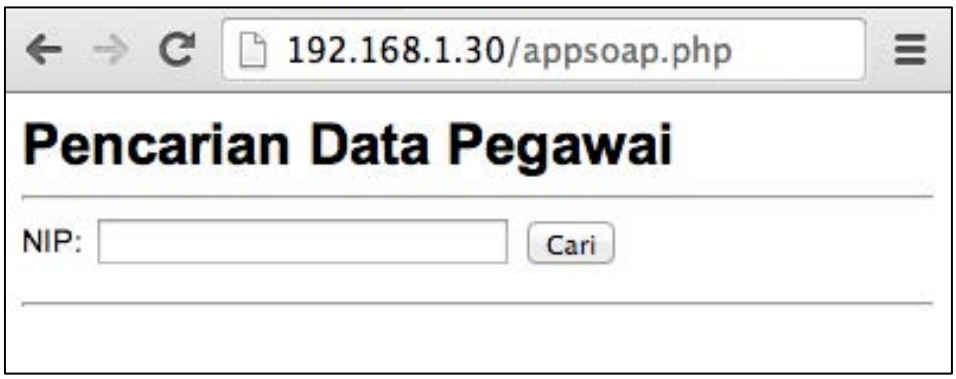
Dari gambaran tersebut dapat dipahami bahwa untuk membuat dan mengakses Webservice SOAP dari suatu Aplikasi membutuhkan program bantu SOAP, dimana dalam membuat Webservicenya menggunakan class SOAP-Server sedangkan untuk koneksi dari Aplikasi ke Webservice menggunakan class SOAP-Client. Tentunya program bantu SOAP akan melakukan pengaturan khusus dalam mengakses dokumen SOAP-Envelope yang tidak disediakan secara langsung oleh aplikasi Web Server seperti IIS, Apache atau NginX. Contoh penerapan untuk mengakses Webservice apisoap.php dari suatu aplikasi appsoap.php adalah sebagai berikut.

File: appsoap.php

```
<?php // Programmed by: Didi Sukyadi

require_once 'lib/nusoap/nusoap.php';

function get_response($code,$message,$data){
  $result['code']=$code;
  $result['message']=$message;
  $result['data']=$data;
  return $result;
}

function writeKeyValue($data){
    echo "<table>";
    foreach($data as $key=>$value):
      echo"<tr><td>".strtoupper($key)."</td>";
      echo "<td>:</td><td>$value</td></tr>";
    endforeach;
    echo "</table>";
}



$url='http://192.168.1.20/ws/apisoap.php?wsdl';
$nip='';
if(isset($_POST['nip'])){
  $nip=$_POST['nip'];
  $api=new nusoap_client($url,true); //inisiasi SOAP Client
  $err=$api->getError();
  if($err){
    $info=get_response(400,"error-response",$err);
  }
  else{
    // Memanggil method data_pegawai dengan parameter nip=id
    $result=$api->call('data_pegawai',array('nip'=>$nip));
    $err=$api->getError();

    if($err){
      $info=get_response(400,"error-response",$err);
    }
```

```
    else{
      if($result!=null){
        $response=$api->response;
        $info=get_response(200,'valid-response',$result);
      }
      else{
        $info=get_response(200,'invalid-response',
                           'Data pegawai tidak ditemukan!');
      }
    }
  }
}
?>
<html>
<head>
  <title>Sistem Kepegawaian</title>
</head>
<body style="margin:8px;font-family:arial;font-size:10pt">
  <strong style='font-size:18pt'>Pencarian Data Pegawai</strong><hr/>
  <form name='dialog' method='post' action=''>
    <label>NIP:</label>
    <input type='text' name='nip' value='<?php echo $nip;?>'/>
    <input type='submit' name='cari' value='Cari'/>
  </form>
<?php
if(is_array($info['data'])){
      writeKeyValue($info['data']); //Tampilkan data pegawai
}
else{
      writeKeyValue($info); //Tampilkan pesan kesalahan
}
?>
</body>
</html>
```

Aplikasi Sistem Kepegawaian appsoap.php yang ditempatkan pada Web Server dengan alamat IP 192.168.1.30 dapat diakses dari Web Browser Komputer lain (PC-Client) dengan alamat URL http://192.168.1.30/appsoap.php seperti ilustrasi berikut.



Gambar 1.12 Tampilan awal appsoap.php

Ketika mencari pegawai dengan cara mengisi NIP kemudian kunci [Enter] ditekan oleh pengguna, Aplikasi Sistem Kepegawaian di lokasi server dengan alamat IP 192.168.1.30 akan mengambil data pegawai yang tersedia di lokasi lain melalui Webservice Informasi Kepegawaian yang berlokasi di server yang berbeda dengan alamat IP 192.168.1.20. Pertama Aplikasi akan melakukan inisiasi koneksi ke Webservice dengan perintah:

```
$url='http://192.168.1.20/ws/apisoap.php?wsdl';
$api=new nusoap_client($url,true);
```

Selanjutnya permintaan data pegawai dengan nilai parameter NIP=197403102010031001 akan disampaikan ke Webservice melalui method yang diminta untuk direspon kembali dalam bentuk format data XML kepada Aplikasi Sistem Kepegawaian. Perintah programnya adalah:

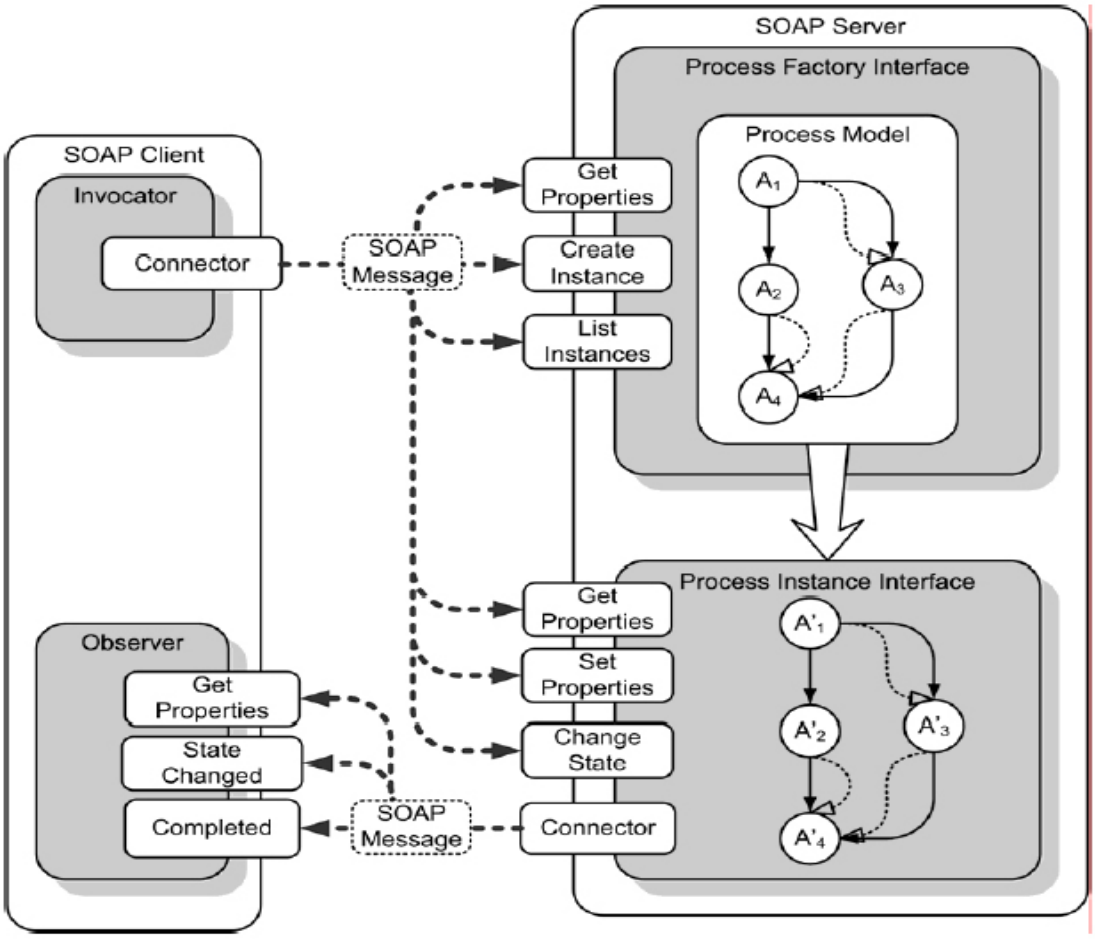
```
$result=$api->call('data_pegawai',array('nip'=>$nip));
```

Apabila data berhasil ditemukan maka data pegawai tersebut akan direspon browser ke layar seperti Gambar 1.13, jika data tidak ditemukan maka akan diberikan pesan seperti Gambar 1.14.

Gambar 1.13 Pencarian pegawai berdasarkan NIP pada appsoap.php

Gambar 1.14 Pencarian pegawai tidak ditemukan pada appsoap.php

Berdasarkan teori yang diterangkan oleh Michael zur Muehlen pada tahun 2004, proses yang terjadi pada Webservice SOAP kurang lebih adalah seperti gambaran berikut.



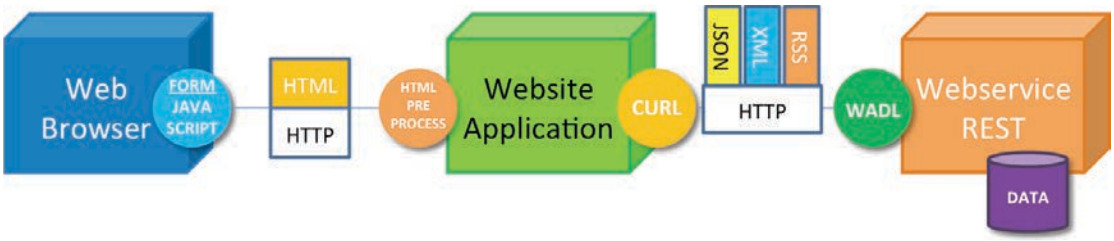
Gambar 1.15 Mekanisme Webservice SOAP

Secara eksplisit proses setiap SOAP-Client yang akan melakukan komunikasi data dengan SOAP-Server terlebih dahulu melewati fungsi invocator atau konektor untuk mengambil nilai hasil fungsi (action) pada Webservice. Pesan (SOAP-Message) berupa nama fungsi dan parameter nilai yang diminta dikirim oleh invocator. Saat fungsi ini sampai di Webservice, maka SOAP-Server akan membentuk jalinan dengan objek fungsi (Create Instance) yang terdapat dalam daftar fungsi (List Instances), kemudian jalinan tersebut akan mengambil properti fungsi fisik (Get Properties) yang terdapat dalam program melalui metode Get (fungsi yang bertugas untuk mengembalikan nilai suatu proses), Set (fungsi yang hanya bertugas melakukan proses tanpa mengembalikan nilai), dan Change State (fungsi even driven yang menunggu kondisi kejadian suatu proses). Hasil proses fungsi akan dikembalikan kepada SOAP-Client dalam bentuk SOAP-Message melalui event-driven State Change yang berfungsi sebagai observer/listener.

1.1.2 Konsep REST

REST (Representational State Transfer) adalah konsep pengembangan Webservice dengan memanfaatkan kemampuan prinsip dan protokol yang sudah tersedia dalam aplikasi Web Server, filosofi REST yang dikenal dengan nama RESTful menganggap teknologi Web sudah cukup untuk mendukung penerapan API Webservice yang kuat (robust). Hal ini dibuktikan bahwa seorang programmer yang mengerti HTTP dan XML akan mampu membuat API Webservice tanpa membutuhkan alat bantu (toolkit) di belakang apa yang biasanya digunakan dalam pengembangan aplikasi internet. Webservice REST pada dasarnya memanfaatkan URL (Uniform Resource Locator) sebagai representasi dari resource Web yang ingin digunakan melalui perintah protokol HTTP (GET, POST, PUT dan DELETE), namun

dalam prakteknya yang sering digunakan adalah GET dan POST. Sehingga suatu fungsi/subproses layanan dianggap sebagai bagian dari resource (URI). Berikut adalah gambaran mekanisme akses Webservice dengan konsep REST.



Gambar 1.16 Layanan Berbagi Pakai Data dengan konsep REST

Keuntungan utama konsep REST adalah:

- Lightweight (akses cepat dan ringan), tidak membutuhkan XML markup tambahan.
- Hasilnya dapat dibaca dengan mudah (human readable result).
- Mudah dikembangkan dan digunakan, tidak membutuhkan alat bantu khusus.

Kelebihan menggunakan konsep REST adalah:

- Bahasa dan platform agnostic
- Penggunaannya lebih sederhana/simple
- Mudah dipelajari
- Ringkas, tidak membutuhkan lapisan pertukaran pesan (messaging) tambahan.
- Desain dan filosofi berbasis Web.

Kelemahan menggunakan konsep REST adalah:

- Tidak mendukung akses multi protokol.
- Kurangnya dukungan standar untuk keandalan pesan.
- Informasi sangat terbuka.

Menurut Pautasso pada tahun 2008, metode REST didasari oleh 4 (empat) prinsip utama teknologi, yaitu:

- *Resource identifier through Uniform Resource Identifier (URI)*, REST Webservice mencari sekumpulan sumberdaya yang mengidentifikasi interaksi antar client.
- *Uniform interface,* sumberdaya yang dimanipulasi (CRUD: Create, Read, Update, Delete) menggunakan operasi PUT, GET, POST, dan DELETE.
- *Self-descriptive messages,* sumberdaya informasi tidak terikat, sehingga dapat mengakses berbagai format konten (HTML, XML, PDF, JPEG, Plain text dan lainnya). Metadata pun dapat digunakan.
- *Stateful interactions through hyperlinks,* setiap interaksi dengan suatu sumberdaya bersifat stateless, yaitu request messages tergantung jenis kontennya.

Webservice dengan konsep REST memiliki cara kerja yang hampir sama dengan Website dinamis, yaitu dengan memanfaatkan aplikasi Web Server sebagai REST-Server dan Web Browser sebagai REST-Client melalui akses URL. Namun pada saat diimplementasikan ke dalam aplikasi maka interkoneksi antara REST-Client ke REST-Server membutuhkan program pendukung bawaan Web Server untuk koneksi ke Webservice dan proses pengambilan data dari Webservice. Program pendukung bawaan tersebut berperan untuk konektor komunikasi data antar program aplikasi yang tersedia dalam banyak pilihan seperti CURL, Telnet, HTTP-Connector, dll. Sementara itu konten hasil proses Webservice REST berupa teks sederhana dapat diolah sesuai kebutuhan melalui pembentukan struktur data dengan berbagai format, seperti XML, JSON, CSV, Notasi atau HTML. Hal ini dapat dilakukan melalui program pendukung konversi format data.

Berikut adalah contoh sederhana yang membuktikan bahwa konsep REST mampu memanfaatkan Web Server menjadi Webservice untuk sebuah API dalam pemrograman PHP.

File: simpleapirest.php

```
<?php
// Programmed by: Didi Sukyadi

if(isset($_GET['nip'])){
  $data=data_pegawai($_GET['nip']);
  if($data){
      echo json_encode($data); //Konversi data ke format JSON
  }
  else{
      echo json_encode(array('pesan'=>'Data tidak ditemukan!'));
  }
}

else{
echo "API Webservice: <strong>Informasi Kepegawaian</strong><hr/>";
echo "<pre>Akses: GET ".currentURL()."?nip=.....</pre>";
}

function data_pegawai($nip){
      $datapegawai=array(
            "0"=>array("nip"=>"197811242009041002",
                       "nama"=>"Siswanto",
                       "kelahiran"=>"Bandung",
                       "tgl_lahir"=>"24/NOP/1978"),
            "1"=>array("nip"=>"197403102010031001",
                       "nama"=>"Linda",
                       "kelahiran"=>"Jakarta",
                       "tgl_lahir"=>"10/MAR/1974"),
            "2"=>array("nip"=>"197506152010031002",
                       "nama"=>"Damar",
                       "kelahiran"=>"Surabaya",
                       "tgl_lahir"=>"15/JUN/1975"),
            "3"=>array("nip"=>"197608182011041001",
                       "nama"=>"Hendri",
                       "kelahiran"=>"Medan",
                       "tgl_lahir"=>"18/AGS/1976")
      );
      foreach($datapegawai as $id=>$data){
            if($data["nip"]==$nip){
                  return $data;
                  break;
            }
      }
}

function currentURL(){
  $s = &$_SERVER;
  $ssl = (!empty($s['HTTPS']) && $s['HTTPS']=='on')?true:false;
  $sp = strtolower($s['SERVER_PROTOCOL']);
  $protocols = substr($sp,0,strpos($sp,'/')).(($ssl)?'s':'');
  $port = $s['SERVER_PORT'];
  $port = ((!$ssl && $port=='80') || ($ssl && $port=='443')) ?
          '' : ':'.$port;
```

```
  $host = isset($s['HTTP_X_FORWARDED_HOST']) ?
          $s['HTTP_X_FORWARDED_HOST']:isset($s['HTTP_HOST']) ?
          $s['HTTP_HOST'] : $s['SERVER_NAME'];

  $uri = $protocols.'://'.$host.$port.$s['REQUEST_URI'];
  $segments = explode('?', $uri, 2);
  $url = $segments[0];
  return $url;
}
?>
```

Program 'simpleapirest.php' tersebut kemudian ditempatkan dalam Web Server pada direktori /var/www/ws/ dan dipetakan ke dalam alias dari alamat host URL servernya yang didefinisikan dalam Web Engine apache2.conf dengan tahapan sebagai berikut:

```
$sudo nano /etc/apache2/apache2.conf
alias /ws "/var/www/ws/"
<directory "/var/www/ws/">
      options indexes followsymlinks
      allowoverride none
      order allow,deny
      allow from all
</directory>
$sudo /etc/init.d/apache2 restart
```

Aplikasi simpleapirest.php tersebut sudah bisa dianggap sebagai API berbasis Web (API Webservice) yang diakses melalui URL http://192.168.1.20/ws/simpleapirest.php dengan memanfaatkan aplikasi Web Browser dari komputer lain yang terhubung dalam satu jaringan seperti gambaran berikut ini.

Gambar 1.17 Tampilan awal Webservice simpleapirest.php

Informasi awal Webservice tersebut akan memberitahukan cara mengakses informasi guna mendapatkan data dari fungsi yang disediakan dalam aplikasi simpleapirest.php, yaitu melalui URL http://192.168.1.20/ws/simpleapirest.php. Dengan menambah parameter '?nip=197403102010031001' dari URL tersebut melalui aplikasi Web Browser maka informasi pegawai berdasarkan nip yang dicari melalui Webservice dalam fungsi data_pegawai akan dikembalikan ke Web Browser dalam tampilan format JSON seperti gambaran berikut.

Gambar 1.18 Hasil akses langsung Webservice simpleapirest.php

Langkah-langkah tersebut telah menggambarkan bahwa aplikasi Web Server dapat berperan sebagai REST-Server yang diterapkan melalui aplikasi simpleapirest.php dan aplikasi Web Browser berperan sebagai REST-Client. Hal ini membuktikan bahwa sebenarnya membuat suatu Webservice tidaklah sulit, hanya tinggal bagaimana mengatur pengelolaannya.

Selanjutnya untuk memanfaatkan Webservice tersebut diperlukan Aplikasi yang berada di lokasi lain. Salah satunya dengan cara sederhana melalui perintah dalam tag FORM yang sudah disediakan pada platform Web dengan format HTML sebagai berikut.

File: simpleapprest.php

```
<?php
// Programmed by: Didi Sukyadi

$apiurl="http://192.168.1.20/ws/simpleapirest.php";
?>

<div style='font-family:arial;font-size:10pt'>
<strong style='font-size:18pt'>Pencarian Data Pegawai</strong>
```

```
<hr/>

<form name='dialog' method='get' action='<?php echo $apiurl;?>'>
  <label>NIP:</label>
  <input type='text' name='nip' size='30' value=''/>
</form>

</div>
<hr/>
```

Aplikasi simpleapprest.php dapat dianggap sebagai REST-Client yang akan memanfaatkan Webservice simpleapirest.php sebagai REST-Server dengan menyisipkan URL Webservice ke dalam atribut ACTION dalam elemen FORM pada dokumen HTML. Saat aplikasi simpleapprest.php diakses maka yang perlu dilakukan adalah mengisi data nip yang akan dicari seperti gambaran berikut.

Gambar 1.19 Tampilan simpleapprest.php

Setelah mengisi data NIP dilanjutkan dengan menekan enter pada elemen kotak input tersebut, program aplikasi akan merespon serta melakukan pengalihan akses ke Webservice melalui URL yang sudah didefinisikan dalam atribut ACTION. Sehingga akan tampil hasil pencarian data berupa halaman Webservice berikut ini.

Gambar 1.20 Hasil akses ke halaman Webservice simpleapirest.php

Bila diperhatikan sebenarnya langkah tersebut bukanlah cara yang diharapkan suatu Aplikasi untuk mendapatkan data dari Webservice, karena yang terjadi adalah hasil pencarian data justru mengalihkan akses aplikasi ke halaman Webservice sehingga respon data hanya dapat ditampilkan dan tidak dapat diperoleh secara langsung di dalam Aplikasi yang mengakses Webservice.

Cara lain untuk memanfaatkan Webservice REST secara sederhana dalah dengan mengubah ACTION form suatu Aplikasi diarahkan ke dalam program lokal saja atau penyedia form tersebut (dikosongkan), kemudian proses pencarian dan pengambilan data dari Webservice dilakukan melalui perintah program PHP seperti 'file_get_contents', dengan memastikan bahwa konfigurasi PHP (php.ini) untuk membuka berkas via URL telah diaktifkan (*allow_url_fopen 1*). Berikut ini program Aplikasi simpleapprest.php yang periu diperbaiki agar dapat mengakses dan mengolah data Webservice simpleapirest.php.

File: simpleapprest.php

```
<?php
// Programmed by: Didi Sukyadi

$apiurl="http://192.168.1.20/ws/simpleapirest.php";
$nip= isset($_GET["nip"])?$_GET["nip"]:'';
?>

<div style='font-family:arial;font-size:10pt'>
<strong style='font-size:18pt'>Pencarian Data Pegawai</strong>

<hr/>
<form name='dialog' method='get' action=''>
  <label>NIP:</label>
  <input type='text' name='nip' size='30'
         value='<?php echo $nip;?>'/>
</form>
</div>
<hr/>

<?php

if( !empty($nip) ){

      $result=file_get_contents($apiurl."?nip=".$nip);
```

```
        if($result){
                echo "<pre>"; //Konversi data JSON ke format Array PHP
                $data=json_decode($result);
                foreach($data as $key=>$value){
                        echo $key.": ".$value."<br/>";
                }
                echo "</pre>";
        }
        else{
                echo "Webservice gagal diakses!";
        }

}
?>
```

Dengan diperbaikinya Aplikasi simpleapprest.php akan diperoleh hasil yang berbeda setelah pencarian data dilakukan sebagaimana gambaran berikut ini.

Gambar 1.21 Hasil pencarian data di halaman simpleapprest.php

Berdasarkan contoh sederhana tersebut dapat disimpulkan bahwa untuk mengakses Webservice diperlukan perintah pendukung seperti perintah 'FORM' atau fungsi 'file_get_contents' yang sudah tersedia dalam HTML atau PHP. Namun untuk penanganan yang lebih lengkap maka diperlukan program pendukung yang lengkap untuk menangani masalah komunikasi data.

Salah satu program pendukung yang lengkap dalam menangani komunikasi data antar aplikasi berbasis Web adalah Client-URL (CURL). Program bantu (Utility Program) CURL dapat memenuhi penanganan komunikasi data di beberapa protokol, seperti HTTP,

HTTPS, FTP, GOPHER, TELNET, FILE, dan LDAP. Selain banyak opsi parameter yang disediakan, CURL juga mampu melakukan sub-koneksi dalam waktu yang bersamaan. Dan yang paling penting adalah penanganan tanggapan akses dari CURL tersusun secara sistematis. Agar dapat menggunakan CURL dalam PHP, diperlukan modul pustaka lib_curl dan php_curl yang terpasang pada Web Server Apache. Informasi tentang persiapan dan cara penggunaan CURL dalam PHP dapat dibaca pada berkas dokumentasi PHP.

Sebelum melangkah pada pemanfaatan CURL untuk koneksi ke Webservice, perlu diperhatikan bahwa Webservice simpleapirest.php adalah contoh sederhana sebuah Webservice yang masih menggunakan parameter dasar URI dalam menangani permintaan data. Sebenarnya konsep RESTful menekankan pada penamaan URL yang bersifat semantik, dalam hal ini setiap URL dalam konsep RESTful minimal memiliki informasi nama method/fungsi dan parameter yang diminta misalnya 'http://192.168.1.20/ws/apirest.php/pegawai/nip=19800212.....'. Jadi suatu API Webservice dapat memiliki lebih dari satu method yang dapat diakses melalui URL. Untuk menangani masalah ini maka perlu dibuat Class program khusus untuk sebuah API Webservice dengan konsep RESTful sebagai berikut.

File: restserver.php

```
<?php // Programmed by: Didi Sukyadi
require_once 'Array2XML.php'

class rest_server{
  private $uriPath='/';
  private $basePath='/';
  private $rootPath='/';
  private $requestURI='';
  private $requestValues=array();
  private $requestKeys=array('method','parameter');
  private $request=array();
  private $requestInput=array();
  private $serviceName=''; private $infoName='';
  private $methods=array();
```

```
  public function currentURL(){
      $s=&$_SERVER;
      $ssl=(!empty($s['HTTPS']) && $s['HTTPS']=='on')?true:false;
      $sp = strtolower($s['SERVER_PROTOCOL']);
      $protocols=substr($sp,0,strpos($sp,'/')).(($ssl)?'s':'');
      $port = $s['SERVER_PORT'];
      $port=((!$ssl && $port=='80') || ($ssl && $port=='443')) ?
      '' : ':'.$port;

      $host = isset($s['HTTP_X_FORWARDED_HOST']) ?
      $s['HTTP_X_FORWARDED_HOST']:isset($s['HTTP_HOST']) ?
      $s['HTTP_HOST'] : $s['SERVER_NAME'];

      $uri=$protocols.'://'.$host.$port.$s['REQUEST_URI'];
      $segments = explode('?', $uri, 2); $url = $segments[0];
      return $url;
  }

  private function initPath(){
      $this->uriPath=
      pathinfo($_SERVER['REQUEST_URI'],PATHINFO_DIRNAME);

      $this->uriPath.=substr($this->uriPath,-1)=='/'?'':'/';

      $this->basePath=
      pathinfo($_SERVER['SCRIPT_NAME'],PATHINFO_DIRNAME);

      $this->basePath.=substr($this->basePath,-1)=='/'?'':'/';

      $this->rootPath=
      pathinfo($_SERVER['SCRIPT_FILENAME'],PATHINFO_DIRNAME);

      $this->rootPath.=substr($this->rootPath,-1)=='/'?'':'/';

      $many=0;
      $baseURI=str_replace($_SERVER['SCRIPT_NAME'],'',
                           $_SERVER['REQUEST_URI'],$many);
      if($many==0){
         $many=0;
         $baseURI=str_replace($this->basePath,'',
                              $_SERVER['REQUEST_URI'],$many);
      }
      else{
         $this->basePath=$_SERVER['SCRIPT_NAME'];
         $this->basePath.=substr($this->basePath,-1)=='/'?'':'/';
      }
      if($many>0){
         $baseURI=
         substr($baseURI,0,1)=='/'?substr($baseURI,1):$baseURI;
         $this->requestURI=$baseURI;
      }
      else{
         $this->requestURI='';
      }
      $this->requestURI.=substr($this->requestURI,-1)=='/'?'':'/';
  }

  private function initRequest(){
      $this->requestValues=explode("/",$this->requestURI);
      if(count($this->requestValues)>count($this->requestKeys)){
        $this->requestValues=array_slice($this->requestValues,
        0,count($this->requestKeys));
      }
```

```
      else{
        $this->requestValues=array_pad($this->requestValues,
        count($this->requestKeys),'');
      }
      $this->request=array_combine($this->requestKeys,
      $this->requestValues);
      if(empty($this->request['parameter'])){
        if(!empty($_POST)){
          // Konversi array POST ke parameter string URI
          $this->request['parameter']=http_build_query($_POST);
        }
      }
      // Konversi string URI ke array PHP
      parse_str($this->request['parameter'],$this->requestInput);
  }


  private function response($status,$messages,$data){
      header("content-type:text/xml; charset=ISO-8859-1");
      $xml=Array2XML::createXML($messages,$data);
      echo $xml->saveXML();
  }

  private function descript(){
      $action=$this->currentURL();
  ?>
  <body style='font-family: arial;'>
  <p>URL: <?php echo $action;?></p>
  <p>Query: <?php echo $_SERVER['REQUEST_METHOD'];?></p>
  <p>Method:</p>
  <ul>
  <?php
      foreach($this->methods as $method=>$elements){
        $par="";$fpar="";
        echo "<li><strong><u>".$method."</u></strong><br/>";
        echo "<div style='background-color:#ccccff;
                          padding:8px;
                          margin-top:8px;
                          margin-right:32px;'>";
        echo "Input:";
        echo "<br/>";
        echo "<ol>";
        foreach($elements->inputs as $part=>$item){
          echo "<li>".$part.":".$item['type']."  ".
          (empty($item['desc'])?"":"(".$item['desc'].")")."</li>";
          $par.=empty($par)?"":"&";
          $par.=$part."=xxxx";
          $fpar.=empty($fpar)?"":",";
          $fpar.=$part;
        }
        echo "</ol>";
        echo "<br/>";
        echo "Output:";
        echo "<br/>";
        echo "<ol>";
        foreach($elements->outputs as $part=>$item){
          echo "<li>".$part.": ".$item['type']."  ".
          (empty($item['desc'])?"":"(".$item['desc'].")")."</li>";
        }
        echo "</ol>";
        echo "<br/>";
        echo "Function: ".$elements->functionName.
        (empty($fpar)?'':'('.$fpar.')')."<br/>";
```

```
        echo "<br/>";
        echo "Action: GET|POST <strong>".
             $action.$elements->selfName.
             "/[".$par."]</strong><br/>";
        echo "</li><br/>";
        echo "</div>";
      }
  ?>
  </ul>
  <p>Back to the <a href="<?php echo $action;?>">front page</a></p>
  </body>
  <?php
  }

  private function home(){
    $title=$this->infoName;
  ?>
  <html>
  <head>
    <title><?php echo 'REST '.$title;?></title>
    <style type="text/css">
      body{
        font-family: arial; color: #000000;
        background-color: #ffffff; margin: 0px 0px 0px 0px;
      }
      p{
        font-family: arial; color: #000000;
        margin-top: 0px; margin-bottom: 12px;
      }
      pre{
        background-color: silver; padding: 5px;
        font-family: Courier New; font-size: x-small;
        color: #000000;
      }
      ul{margin-top: 10px; margin-left: 20px; }
      li{list-style-type: none; margin-top: 10px; color: #000000;}
      .title {
        font-family: arial; font-size: 26px; color: #ffffff;
        background-color: #999999; width: 100%;
        margin-left: 0px; margin-right: 0px;
        padding-top: 10px; padding-bottom: 10px;
      }
      a,a:active  { color: charcoal; font-weight: bold; }
      a:visited   { color: #666666; font-weight: bold; }
      a:hover     { color: cc3300; font-weight: bold; }
    </style>
  </head>
  <body>
    <div><br/>
      <div class=title><?php echo $title;?></div><br/>
      <div style='padding:8px'>
        <p>Action:  GET|POST<strong>
           http://host/service/method/[part1=xx&part2=xx&partN=xxx]
        </strong></p><ul>
        <li><strong>host</strong>= domain name or URL base</li>
        <li><strong>service</strong>= service name</li>
        <li><strong>method</strong>= sub-process or function</li>
        <li>
        <strong>part1...partN</strong>= request parameter and value
        </li></ul>
        <p>View the
        <a href="<?php echo $this->basePath;?>?description">
        description</a>.</p>
```

```
      </div>
    </div>
  </body>
  </html>
  <?php
  }

  public function __construct($serviceName,$infoName){
      $this->initPath(); $this->initRequest();
      $this->serviceName=$serviceName;
      $this->infoName=$infoName;
  }


  public function addMethod($methodName){
    $this->methods[$methodName]=new rest_method($methodName);
    return $this->methods[$methodName];
  }


  public function handle(){
    if(isset($_GET['description']) &&
      ($this->uriPath==$this->basePath)){
      $this->descript();
    }
    elseif(array_key_exists($this->request['method'],
           $this->methods)){
      $method=$this->methods[$this->request['method']];
      $pars='';
      if(count($this->requestInput)>0){
      // Bandingkan parameter URI dengan parameter resource
      $arrdiff=array_diff_key($this->requestInput,$method->inputs);
        if(count($arrdiff)>0){
            $this->response(400,'error-response',
                            'Parameter is not associated');
        }
        else{
          $valpar=array();
          foreach($method->inputs as $keypar=>$despar){
              if(array_key_exists($keypar,$this->requestInput)){
               // Petakan nama parameter dan ambil nilai parameter
                $valpar[$keypar]=$this->requestInput[$keypar];
              }
          }
          // Konversi array nilai parameter ke string
          $pars=implode(',',$valpar);
          $result=call_user_func($method->functionName,$pars);
          if(!empty($result)){
            $this->response(200,'valid response',
                           array('pegawai'=>$result));
          }
          else{
            $this->response(200,'invalid-response','');
          }
        }
      }
      else {
        $this->response(400,'error-response','Missing parameter');
      }
    }
    elseif(!empty($this->request['method'])){
      $this->response(400,'error-response','Undefined method');
    }
```

```
    else{
      $this->home();
    }
  } // --- end of handle() function
} // --- end of rest_server class


class rest_method{
  public $selfName='';
  public $inputs=array();
  public $outputs=array();
  public $functionName='';


  public function __construct($name){
      $this->selfName=$name;
  }

  public function __destruct(){
      $this->selfName='';
      $this->inputs=array();
      $this->outputs=array();
      $this->functionName='';
  }

  public function addInput($name,$desc='',$type='string'){
      $this->inputs[$name]=array('desc'=>$desc,'type'=>$type);
  }

  public function addOutput($name,$desc='',$type='string'){
      $this->outputs[$name]=array('desc'=>$desc,'type'=>$type);
  }

  function setFunctionName($name){
      $this->functionName=$name;
  }
}

?>
```

Dalam penerapannya Class program restserver.php menghasilkan data dalam format XML yang membutuhkan Class program untuk konversi data Array PHP ke dalam format XML seperti berikut ini.

File: Array2XML.php

```
<?php
/**
* Author : Lalit Patel
* Website: http://www.lalit.org
* License: Apache License 2.0
*          http://www.apache.org/licenses/LICENSE-2.0
*/
class Array2XML {
  private static $xml = null;
  private static $encoding = 'UTF-8';
```

```
public static function init($version = '1.0',
$encoding = 'UTF-8', $format_output = true) {
    self::$xml = new DomDocument($version, $encoding);
    self::$xml->formatOutput = $format_output;
    self::$encoding = $encoding;
}

public static function &createXML($node_name, $arr=array()) {
    $xml = self::getXMLRoot();
    $xml->appendChild(self::convert($node_name, $arr));
    self::$xml = null;
    return $xml;
}


private static function &convert($node_name, $arr=array()) {
    $xml = self::getXMLRoot();
    $node = $xml->createElement($node_name);
    if(is_array($arr)){
       if(isset($arr['@attributes'])) {
          foreach($arr['@attributes'] as $key => $value) {
             if(!self::isValidTagName($key)) {
             throw new Exception('[Array2XML] Illegal character'.
             ' in attribute name. attribute: '.$key.
             ' in node: '.$node_name);
             }
             $node->setAttribute($key, self::bool2str($value));
          }
          unset($arr['@attributes']);
       }

       if(isset($arr['@value'])) {
         $node->appendChild($xml->createTextNode(
                              self::bool2str($arr['@value'])));
         unset($arr['@value']);
         return $node;
       }
       elseif(isset($arr['@cdata'])) {
          $node->appendChild($xml->createCDATASection(
                               self::bool2str($arr['@cdata'])));
          unset($arr['@cdata']);
          return $node;
       }
    }
      //create subnodes using recursion
    if(is_array($arr)){
      foreach($arr as $key=>$value){
         if(!self::isValidTagName($key)) {
          throw new Exception('[Array2XML] Illegal character'.
          ' in tag name. tag: '.$key.' in node: '.$node_name);
         }
         if(is_array($value) && is_numeric(key($value))) {
            foreach($value as $k=>$v){
               $node->appendChild(self::convert($key, $v));
            }
         }
         else{
            $node->appendChild(self::convert($key, $value));
         }
         unset($arr[$key]);
      }
    }
```

```
      if(!is_array($arr)) {
         $node->appendChild($xml->createTextNode(
                              self::bool2str($arr)));
      }
      return $node;
  }


  private static function getXMLRoot(){
      if(empty(self::$xml)) {
            self::init();
      }
      return self::$xml;
  }

  private static function bool2str($v){
      //convert boolean to text value.
      $v = $v === true ? 'true' : $v;
      $v = $v === false ? 'false' : $v;
      return $v;
  }

  private static function isValidTagName($tag){
    $pattern = '/^[a-z_]+[a-z0-9\:\-\.\_]*[^:]*$/i';
  return preg_match($pattern,$tag,$matches) && $matches[0]== $tag;
  }

} //--- end of Array2XML class

?>
```

Dengan menggunakan Class program restserver.php maka pembuatan Webservice dapat dilakukan dengan mudah dan mampu menangani banyak method seperti program berikut ini.

File: apirest.php

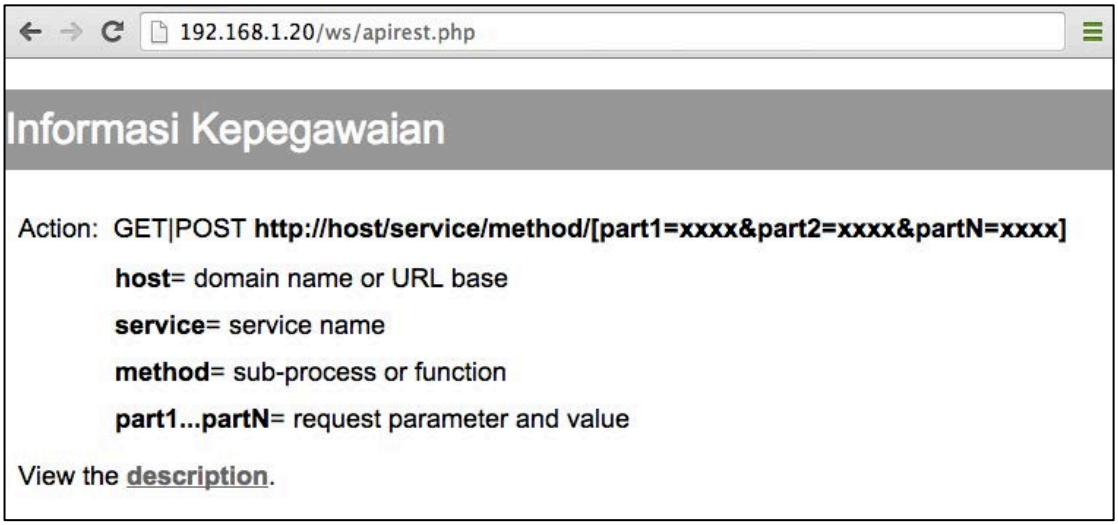
```
<?php // Programmed by: Didi Sukyadi
require_once 'restserver.php'

$api=new rest_server('infopegawai','Informasi Kepegawaian');
$pegawai=$api->addMethod('pegawai');
$pegawai->addInput('nip','Nomor Induk Pegawai');
$pegawai->addOutput('nip');
$pegawai->addOutput('nama');
$pegawai->addOutput('kelahiran','Tempat Lahir');
$pegawai->addOutput('tgl_lahir','Tanggal Lahir','date');
$pegawai->setFunctionName('data_pegawai');
$api->handle();


function data_pegawai($nip){
      $datapegawai=array(
            "0"=>array("nip"=>"197811242009041002",
                       "nama"=>"Siswanto",
                       "kelahiran"=>"Bandung",
                       "tgl_lahir"=>"24/NOP/1978"),
```

```
            "1"=>array("nip"=>"197403102010031001",
                       "nama"=>"Linda",
                       "kelahiran"=>"Jakarta",
                       "tgl_lahir"=>"10/MAR/1974"),
            "2"=>array("nip"=>"197506152010031002",
                       "nama"=>"Damar",
                       "kelahiran"=>"Surabaya",
                       "tgl_lahir"=>"15/JUN/1975")
      );
      foreach($datapegawai as $id=>$data)
      if($data["nip"]==$nip){
            return $data; break;
      }
}
?>
```

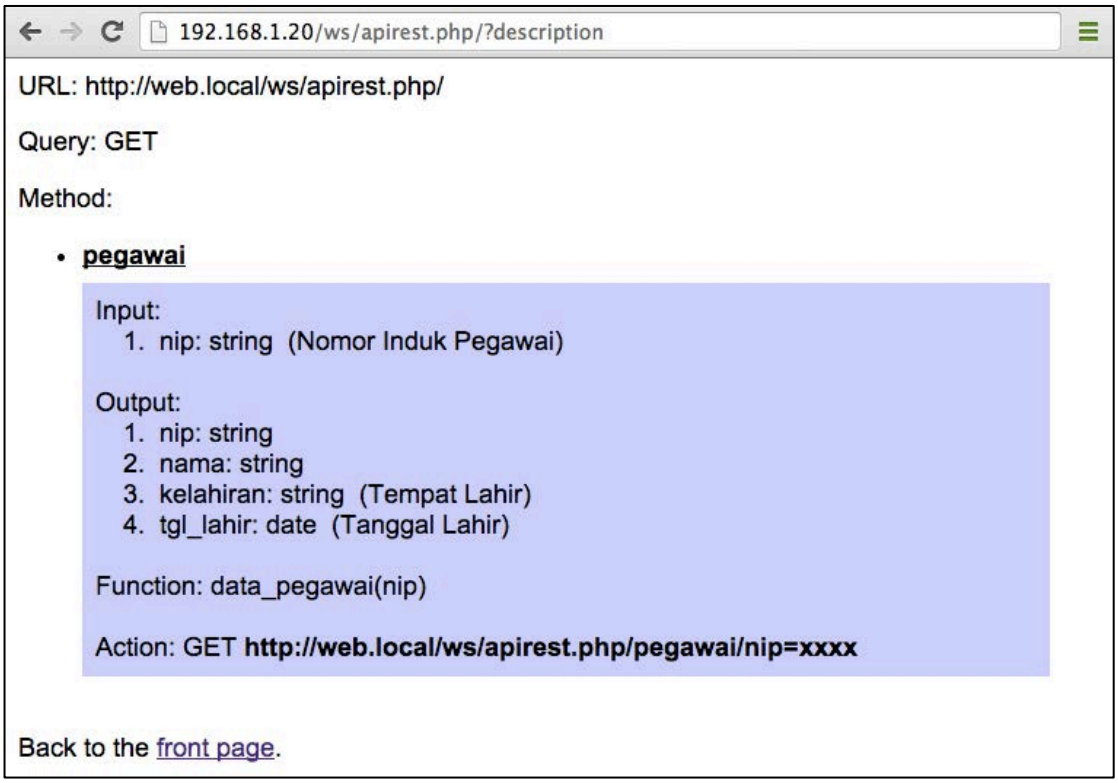
Berkas program apirest.php, restserver.php, dan Array2XML.php kemudian ditempatkan dalam Web Server Apache pada direktori /var/www/ws/ dan dipetakan ke dalam alias dari alamat host URL servernya yang didefinisikan dalam Web Server apache2.conf dengan tahapan sebagai berikut:

```
$sudo nano /etc/apache2/apache2.conf
alias /ws "/var/www/ws/"
<directory "/var/www/ws/">
      options followsymlinks
      allowoverride all
      order allow,deny
      allow from all
</directory>
Simpan dengan Ctrl+KD
Keluar dengan Ctrl+KQ
$sudo a2enmod rewrite
$sudo /etc/init.d/apache2 restart
$nano /var/www/ws/.htaccess
<IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteBase /
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteRule ^ index.php [L]
</IfModule>
```

Parameter FollowSymLinks wajib dipasang pada elemen Options agar RESTful pada URL dapat dikenal oleh Web Server. Berikut ini adalah hasil halaman awal serta deskripsi Webservice apirest.php.

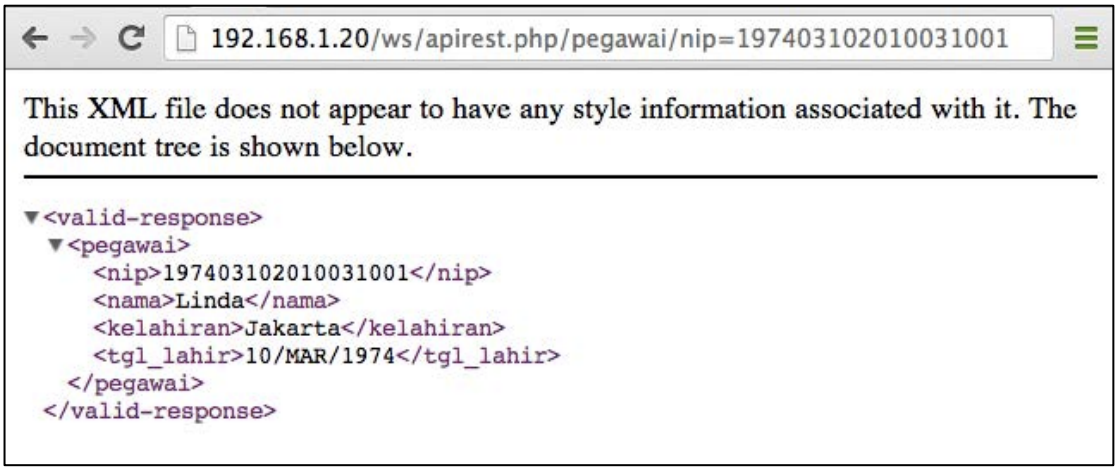


Gambar 1.22 Tampilan depan Webservice apirest.php



Gambar 1.23 Halaman deskripsi Webservice apirest.php

Pratinjau API Webservice dapat diakses melalui Web Browser untuk meyakinkan bahwa hasilnya dapat diperoleh, yaitu dengan URL 192.168.1.20/ws/apirest.php/pegawai/nip=197403102010031001 sebagaimana gambaran berikut ini.



Gambar 1.24 Hasil pencarian data pada Webservice apirest.php

Webservice REST apirest.php dapat diakses dari Aplikasi Web yang memanfaatkan program bantu CURL pada pemrograman PHP dengan konsep RESTful. Untuk mempermudah pemanfaatan program bantu CURL, perlu dibuat Class program sebagai berikut.

File: restclient.php

```
<?php // Programmed by: Didi Sukyadi

require_once 'XML2Array.php';
require_once 'Array2XML.php';

class rest_client{
      private $uriPath='/';
      private $basePath='/';
      private $rootPath='/';
      private $requestURI='';
      private $requestValues=array();
      private $requestKeys=array('method','parameter');
      private $request=array();
      private $requestInput=array();

      private $url='';
      private $connected=false;
      private $status=array('code'=>'','message'=>'','data'=>'');
```

```
private function initPath(){
    $this->uriPath=
    pathinfo($_SERVER['REQUEST_URI'],PATHINFO_DIRNAME);

    $this->uriPath.=substr($this->uriPath,-1)=='/'?'':'/';

    $this->basePath=
    pathinfo($_SERVER['SCRIPT_NAME'],PATHINFO_DIRNAME);

    $this->basePath.=substr($this->basePath,-1)=='/'?'':'/';

    $this->rootPath=
    pathinfo($_SERVER['SCRIPT_FILENAME'],PATHINFO_DIRNAME);

    $this->rootPath.=substr($this->rootPath,-1)=='/'?'':'/';

    $many=0;
    $baseURI=str_replace($_SERVER['SCRIPT_NAME'],'',
                         $_SERVER['REQUEST_URI'],$many);
    if($many==0){
       $many=0;
       $baseURI=str_replace($this->basePath,'',
                            $_SERVER['REQUEST_URI'],$many);
    }
    else{
       $this->basePath=$_SERVER['SCRIPT_NAME'];
       $this->basePath.=substr($this->basePath,-1)=='/'?'':'/';
    }
    if($many>0){
       $baseURI=
       substr($baseURI,0,1)=='/'?substr($baseURI,1):$baseURI;
       $this->requestURI=$baseURI;
    }
    else{
       $this->requestURI='';
    }
    $this->requestURI.=substr($this->requestURI,-1)=='/'?'':'/';
}


private function initRequest(){
    $this->requestValues=explode("/",$this->requestURI);
    if(count($this->requestValues)>count($this->requestKeys)){
      $this->requestValues=array_slice($this->requestValues,
      0,count($this->requestKeys));
    }
    else{
      $this->requestValues=array_pad($this->requestValues,
      count($this->requestKeys),'');
    }
    $this->request=array_combine($this->requestKeys,
    $this->requestValues);
    if(empty($this->request['parameter'])){
      if(!empty($_POST)){
        // Konversi array POST ke parameter string URI
        $this->request['parameter']=http_build_query($_POST);
      }
    }
    // Konversi string URI ke array PHP
    parse_str($this->request['parameter'],$this->requestInput);
}
```

```
	private function response($code,$message,$data){
		$this->status['code']=$code;
		$this->status['message']=$message;
		$this->status['data']=$data;
		return $this->status;
	}

	private function checkAPI($url){
		$result=false;
		$ch = curl_init();
		curl_setopt($ch, CURLOPT_URL, $url);
		curl_setopt($ch, CURLOPT_HEADER, FALSE);
		curl_setopt($ch, CURLOPT_NOBODY, TRUE);
		curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
		curl_setopt($ch, CURLOPT_FAILONERROR, TRUE);
		curl_exec($ch);
		$errno=curl_errno($ch);
		$errmsg=curl_error($ch);
		if ($errno==0){
			$result=true;
		}
		else{
			$this->response(400,'error-response',
			$errmsg.' on '.$url.' (Connection failure!)');
		}
		curl_close($ch);

		return $result;
	}

	public function requestMethod(){
		return $this->request['method'];
	}

	public function requestParameter(){
		return $this->requestInput;
	}

	public function contentXML($message,$data){
		$xml=Array2XML::createXML($message,$data);
		return $xml->saveXML();
	}

	public function status_code(){
		return $this->status['code'];
	}

	public function status_message(){
		return $this->status['message'];
	}

	public function status_data(){
		return $this->status['data'];
	}

	public function __construct($url){
		$this->url=$url;
		$this->connected=$this->checkAPI($this->url);
	}

	public function __destruct(){}
```

```
public function is_connected(){
      return $this->connected;
}

public function call($method_name,
                     $parameter=array(),
                     $callmethod='GET'){
      $result=false;
      $par='';
      $url=$this->url;
      $url.=substr($url,-1)=='/'?'':'/';
      if(!empty($parameter))
        $par=http_build_query($parameter);
      $callmethod=strtoupper($callmethod);
      if($callmethod=='GET'){
            $uri=empty($method_name)?'':
                 $url.$method_name.'/'.$par;
      }
      if($callmethod=='POST'){
            $uri=empty($method_name)?'':
                 $url.$method_name.'/';
      }
      if(empty($uri)){
            $this->response(400,'error-response',
                            'Empty method!');
      }
      else{
            $ch = curl_init();
            // URL target koneksi
            curl_setopt($ch, CURLOPT_URL, $uri);
            // Output dengan header
            curl_setopt($ch, CURLOPT_HEADER, TRUE);
            // Mendapatkan tanggapan
            curl_setopt($ch, CURLOPT_RETURNTRANSFER,TRUE);
            curl_setopt($ch, CURLOPT_BINARYTRANSFER, FALSE);
            curl_setopt($ch, CURLOPT_FAILONERROR, TRUE);
            // Menggunakan metode HTTP GET
            if($callmethod=='GET'){
                  curl_setopt($ch, CURLOPT_HTTPGET, TRUE);
            }
            // Menggunakan metode HTTP POST
            if($callmethod=='POST'){
                  curl_setopt($ch, CURLOPT_POST, TRUE);
                  // Sisipkan parameter
                  curl_setopt($ch, CURLOPT_POSTFIELDS,$par);
            }
            // Buka koneksi dan dapatkan tanggapan
            $rest=curl_exec($ch);
            $errno=curl_errno($ch);
            $errmsg=curl_error($ch);

            // Periksa kesalahan
            if ($errno!=0){
                  $this->response(400,'error-response',
                                  $errmsg.' on '.$uri);
            }
            else{
              $http_code=curl_getinfo($ch,
                  CURLINFO_HTTP_CODE);
              $header_size=curl_getinfo($ch,
                  CURLINFO_HEADER_SIZE);
              $content_length=curl_getinfo($ch,
```

```
                        CURLINFO_CONTENT_LENGTH_DOWNLOAD);
                    $content_type=curl_getinfo($ch,
                        CURLINFO_CONTENT_TYPE);
                    $result=new rest_data($rest,
                        $header_size,
                        $content_type,
                        $content_length);
                    $this->response($result->status_code(),
                                    $result->status_message(),
                                    $result->status_data());
                }
                curl_close($ch);
            }
            return $result;
        }
}

class rest_data{
        private $status=array('code'=>'','message'=>'','data'=>'');
        private $data=NULL;
        private $doc= array('xmldesc'=>'',
                            'xmldata'=>'',
                            'xmldoc'=>'');

        private function response($code,$message,$data){
            $this->status['code']=$code;
            $this->status['message']=$message;
            $this->status['data']=$data;
            return $this->status;
        }

        public function __construct($rest,$header_size,
                                    $content_type,
                                    $content_length){
          $pos=strpos($content_type,'xml');
          if($pos===false){
              $this->response(400,'error-response',
              "Document is not XML format! <br/><pre>".
              substr($rest,$content_length*-1)."</pre>");
          }
          elseif($pos){
              $this->doc['xmldoc']=$rest;
              $this->doc['xmldesc']=substr($rest,0,$header_size);
              $this->doc['xmldata']=substr($rest,$content_length*-1);
              $this->response(200,'invalid_response','No result!');
              if(!empty($this->doc['xmldata'])){
                $xmlpos=strpos($this->doc['xmldata'],'<?xml');
                if($xmlpos!==false){
                 $data=XML2Array::createArray($this->doc['xmldata']);
                 if(isset($data['valid-response'])){
                    if(!empty($data['valid-response'])){
                      $this->data=$data['valid-response'];
                      $this->response(200,'valid-response',
                                      $this->doc['xmldata']);
                    }
                 }
                 if(isset($data['error-response'])){
                   $this->response(400,'error-response',
                                   $this->doc['xmldata']);
                 }
                }
              }
          }
```

```
        else{
            $this->response(400,'error-response',
                                "Unspecified Error");
        }
      }

      public function __destruct(){}

      public function status_code(){
            return $this->status['code'];
      }
      public function status_message(){
            return $this->status['message'];
      }
      public function status_data(){
            return $this->status['data'];
      }

      public function xmldesc(){
            return $this->doc['xmldesc'];
      }

      public function xmldata(){
            return $this->doc['xmldata'];
      }

      public function xmlresponse(){
            return $this->doc['xmldoc'];
      }

      public function rawdata(){
            return $this->data;
      }

      public function jsondata(){
            $result='';
            if(is_array($this->data))
              $result=json_encode($this->data);
            return $result;
      }

      public function serialdata(){
            $result='';
            if(!empty($this->data))
              $result=serialize($this->data);
            return $result;
      }

      public function uridata(){
            $result='';
            if(!empty($this->data))
              $result=rawurldecode(http_build_query($this->data));
            return $result;
      }
}
?>
```

Dalam penerapannya Class program restclient.php akan melakukan konversi data dari format XML ke dalam format Array PHP agar

dapat diolah oleh program Aplikasi yang membacanya, untuk itu perlu disediakan Class program untuk konversi tersebut seperti program berikut ini.

File: XML2Array.php

```
<?php
/**
* Author : Lalit Patel
* Website: http://www.lalit.org
* License: Apache License 2.0
*          http://www.apache.org/licenses/LICENSE-2.0
*/

class XML2Array {
    private static $xml = null;
    private static $encoding = 'UTF-8';
    /**
     * Initialize the root XML node [optional]
     * @param $version
     * @param $encoding
     * @param $format_output
     */
    public static function init($version = '1.0',
                                $encoding = 'UTF-8',
                                $format_output = true) {
        self::$xml = new DOMDocument($version, $encoding);
        self::$xml->formatOutput = $format_output;
        self::$encoding = $encoding;
    }

    /**
     * Convert an XML to Array
     * @param string $node_name - name of the root node
     * @param array $arr - aray to be converterd
     * @return DOMDocument
     */
    public static function &createArray($input_xml) {
       $xml = self::getXMLRoot();
       if(is_string($input_xml)) {
                   $parsed = $xml->loadXML($input_xml);
                   if(!$parsed) {
                     throw new Exception('[XML2Array] '.
                     'Error parsing the XML string.');
                   }
       } else {
                   if(get_class($input_xml) != 'DOMDocument') {
                     throw new Exception('[XML2Array] '.
                     'The input XML object '.
                     'should be of type: DOMDocument.');
                   }
                   $xml = self::$xml = $input_xml;
       }
      $array[$xml->documentElement->tagName] =
             self::convert($xml->documentElement);
       // clear the xml node in the class for 2nd time use.
       self::$xml = null;
       return $array;
    }
```

```
/**
 * Convert an Array to XML
 * @param mixed $node - XML as a string
 * or as an object of DOMDocument
 * @return mixed
 */
private static function &convert($node) {
  $output = array();
  switch ($node->nodeType) {
  case XML_CDATA_SECTION_NODE:
        $output['@cdata']=trim($node->textContent);
        break;
  case XML_TEXT_NODE:
        $output = trim($node->textContent);
        break;
  case XML_ELEMENT_NODE:
  // for each child node, call the covert function recursively
        for ($i=0, $m=$node->childNodes->length; $i<$m; $i++) {
              $child = $node->childNodes->item($i);
              $v = self::convert($child);
              if(isset($child->tagName)) {
                    $t = $child->tagName;
              // assume more nodes of same kind are coming
                    if(!isset($output[$t])) {
                          $output[$t] = array();
                    }
                    $output[$t][] = $v;
              } else {



              //check if it is not an empty text node
                    if($v !== '') {
                          $output = $v;
                    }
              }
        }

        if(is_array($output)) {
        // if only one node of its kind,
        // assign it directly instead if array($value);
          foreach ($output as $t => $v) {
              if(is_array($v) && count($v)==1) {
                    $output[$t] = $v[0];
              }
          }
          if(empty($output)) {
              //for empty nodes
              $output = '';
          }
        }

        // loop through the attributes and collect them
        if($node->attributes->length) {
          $a = array();
          foreach($node->attributes as $attrName=>$attrNode){
              $a[$attrName] = (string) $attrNode->value;
          }
```

```
                // if its an leaf node, store the value in @value
                // instead of directly storing it.
                if(!is_array($output)) {
                    $output = array('@value' => $output);
                }
                $output['@attributes'] = $a;
            }
            break;
        }
        return $output;
    }


    /*
     * Get the root XML node, if there isn't one, create it.
     */
    private static function getXMLRoot(){
        if(empty(self::$xml)) {
            self::init();
        }
        return self::$xml;
    }
}
?>
```

Dengan menggunakan Class program restclient.php maka pengaturan akses Webservice dari Aplikasi Web dapat dilakukan dengan mudah dan mampu menangani banyak method seperti program berikut ini.

File: apprest.php

```
<?php // Programmed by: Didi Sukyadi
require_once 'restclient.php';
function get_response($code,$message,$data){
      $result['code']=$code;
      $result['message']=$message;
      $result['data']=$data;
      return $result;
}
function writeKeyValue($data){
      echo "<table>";
      foreach($data as $key=>$value){
        echo "<tr><td>".strtoupper($key)."</td><td>:</td>".
             "<td>$value</td></tr>";
      }
      echo "</table>";
}
$url='http://192.168.1.20/ws/apirest.php';
$info=NULL;
$desc='';
$nip='';
if(isset($_POST['nip'])){
      $nip=$_POST['nip'];
      $api=new rest_client($url);
      $desc=$api->status_data();
```

```
      if($api->is_connected()){
        $data=$api->call('pegawai',array('nip'=>$nip),'post');
        $desc=$api->status_data();
        if($api->status_code()==200)
          $desc=($data->rawdata()==''')?
                "Data pegawai  tidak ditemukan!":$data->rawdata();
      }
      $info=get_response($api->status_code(),
                         $api->status_message(),$desc);
}
?>
<html>
<head><title>Sistem Kepegawaian</title></head>
<body style="margin:8px;font-family:arial;font-size:10pt">
      <strong style='font-size:18pt'>Pencarian Data Pegawai
      </strong>

      <hr/>
      <form name='dialog' method='post' action=''>
            <label>NIP:</label>
            <input type='text' name='nip'
                   value='<?php echo $nip;?>' size='30'/>
            <input type='submit' name='cari' value='Cari'/>
      </form>
      <hr/>

      <pre>
<?php
if(is_array($info['data'])){
      writeKeyValue($info['data']);}
else{
      writeKeyValue($info);
}
?>
      </pre>

</body>
</html>
```

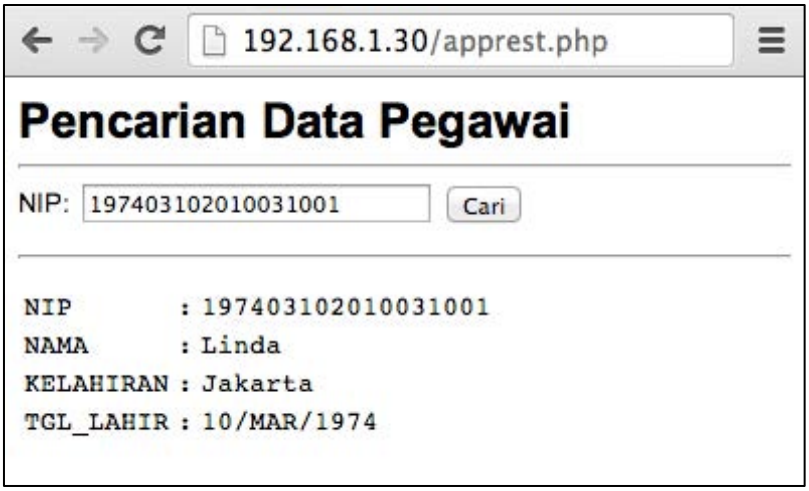
Aplikasi apprest.php dapat mengakses Webservice Informasi Kepegawaian dengan konsep REST melalui perintah berikut.

```
      $url='http://192.168.1.20/ws/apirest.php';

      $api=new rest_client($url);
```

Sedangkan fungsi yang terdapat dalam Webservice akan diakses dengan perintah:

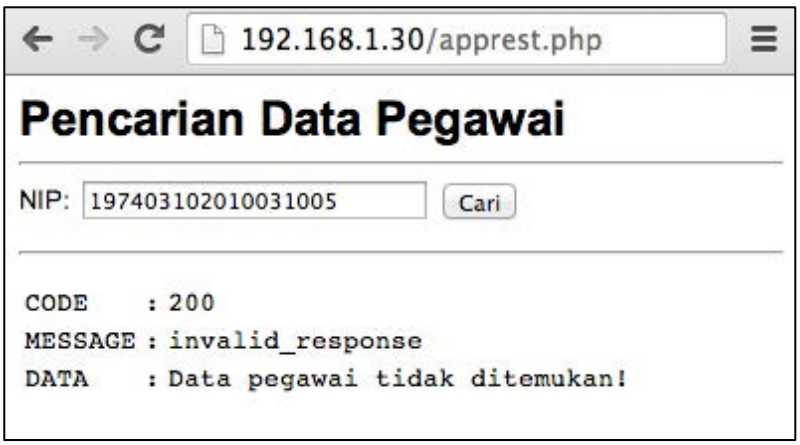
```
      $data=$api->call('pegawai',array('nip'=>$nip),'post');
```

Hasil dari akses fungsi yang terdapat pada Webservice akan ditampilkan kembali ke layar aplikasi seperti gambaran berikut ini.



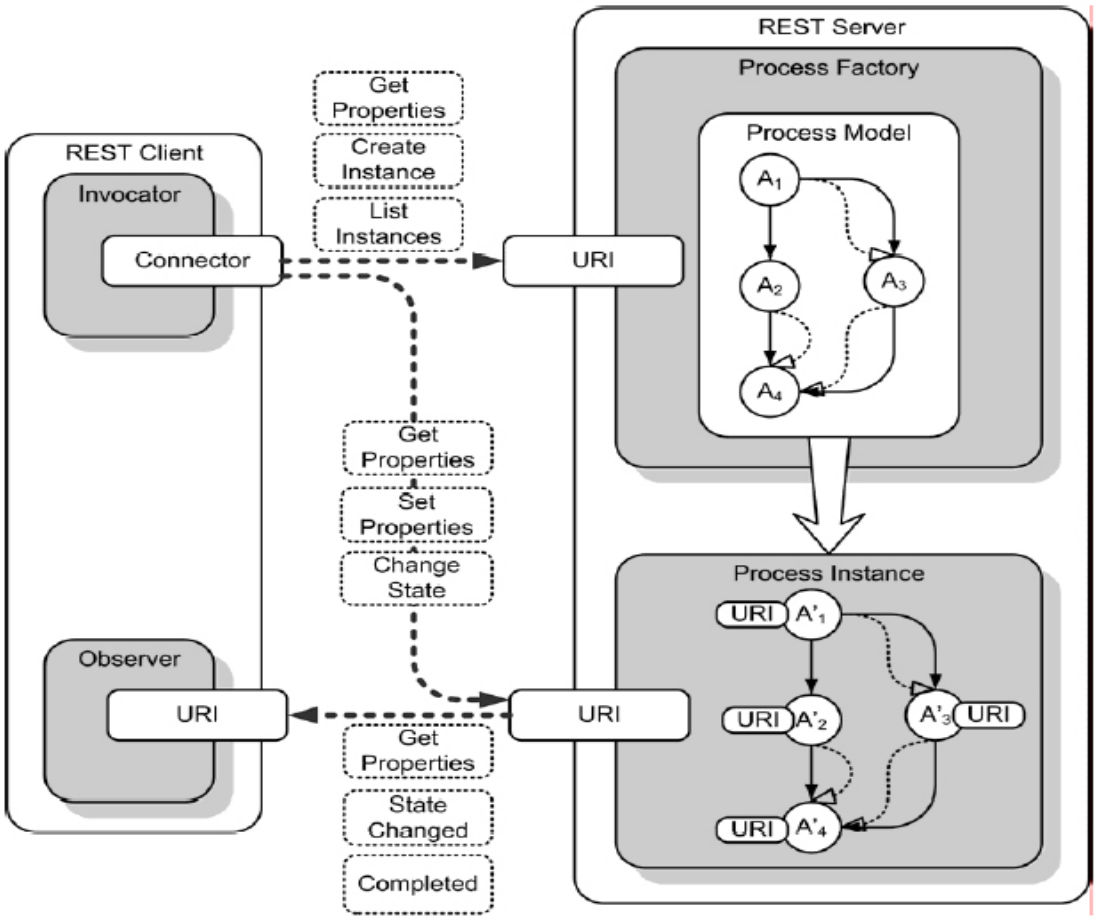
Gambar 1.25 Data ditemukan pada Webservice apirest.php

Apabila akses fungsi tidak menemukan data yang dicari akan menghasilkan tampilan sebagai berikut.



Gambar 1.26 Data tidak ditemukan pada Webservice apirest.php

Berdasarkan teori yang diterangkan oleh Michael zur Muehlen pada tahun 2004, proses yang terjadi pada metode REST kurang lebih adalah seperti gambaran berikut.



Gambar 1.27 Mekanisme Webservice REST

Secara eksplisit mekanisme Webservice REST dibandingkan SOAP memiliki kesamaan yang hampir mirip, namun karena REST tidak menggunakan encapsulated data seperti SOAP-Envelope, REST-Client hanya perlu menyertakan parameter berupa nama method/fungsi/subproses yang akan diproses dengan nilai parameter yang diminta secara langsung dalam bentuk URI. Akses URI ini dapat dilakukan dengan dua cara umum, yaitu POST dan GET. Perbedaannya adalah jika POST dilakukan secara implisit pada URL, sedangkan GET dilakukan secara eksplisit pada URL. Melalui

cara ini Webservice REST cukup membaca permintaan yang dilewatkan kedalam URI, kemudian menterjemahkan seluruh isi parameter berupa nama layanan, nama method (fungsi) yang dipetakan ke dalam fungsi sebenarnya dalam aplikasi berdasarkan nilai parameter permintaan. Konsep REST menganggap URL dari suatu Webservice adalah sebuah resource (sumberdaya).

Akses ke suatu Webservice SOAP maupun REST tidak hanya dapat dilakukan dari Aplikasi Web yang berada pada suatu Server akan tetapi juga dapat diakses dari Web Browse (Client) secara langsung. Hal ini sudah dibuktikan dengan melewatkan URL Webservice pada atribut ACTION elemen FORM dokumen HTML. Selain itu juga dapat dilakukan dengan konsep AJAX (Asynchronous Javascript And XML) dengan cara melewatkan alamat URL Webservice pada object dari class XMLHTTRequest seperti contoh program berikut ini.

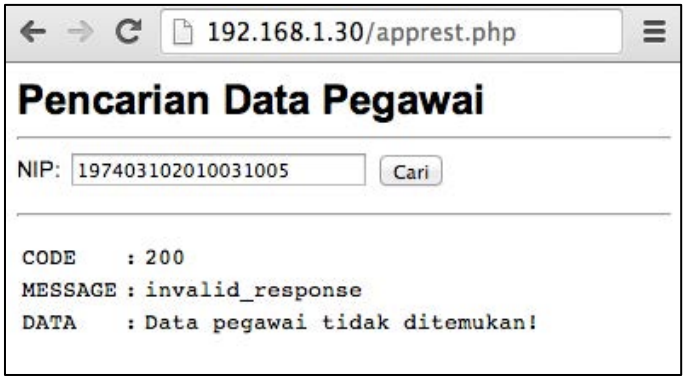
File: appajax.php

```
<?php // Programmed by: Didi Sukyadi ?>
<html>
<head>
      <title>Sistem Kepegawaian</title>
      <script>
      function initAjax(){
            var Ajax;
            if(window.ActiveXObject){
              var msxmlhttp = new Array(
                        'Msxml2.XMLHTTP.5.0',
                        'Msxml2.XMLHTTP.4.0',
                        'Msxml2.XMLHTTP.3.0',
                        'Msxml2.XMLHTTP',
                        'Microsoft.XMLHTTP');
              for (var i = 0; i < msxmlhttp.length; i++) {
                  try {
                        Ajax = new ActiveXObject(msxmlhttp[i]);
                  } catch (e) {
                        Ajax = null;
                  }
              }
            }
            if(window.XMLHttpRequest){
              if(!Ajax && typeof XMLHttpRequest != "undefined")
                 Ajax = new XMLHttpRequest();
            }
            return Ajax;
      }
```

```
      function callAjax(elid, mode, url, methodName, parameters){
            var uri; var post_data; var Ajax=initAjax();
            if(Ajax==null){
                  alert('Null Ajax');
            }
            else{
                  uri=url;
                  slasher=uri.substr(-1,1)=="/"?"":"/";
                  uri=uri+slasher+escape(methodName);
                  mode=mode.toUpperCase();
                  if(mode=='GET'){
                        uri+="/nip=" + escape(parameters);
                        post_data = null;
                  }
                  if(mode=='POST'){
                        post_data = "nip=" + escape(parameters);
                  }
                  Ajax.open(mode,uri);
                  if (mode=="POST"){
                        Ajax.setRequestHeader("Method",
                        "POST "+uri+" HTTP/1.1");
                        Ajax.setRequestHeader("Content-Type",
                        "application/x-www-form-urlencoded");
                  }
                  Ajax.onreadystatechange=function(){
                  if(Ajax.readyState == 4 && Ajax.status==200){
                     var result=Ajax.responseXML;
                     if(result){
                        var data=result.firstChild;
                        var info='<table>';
                        if(data.nodeName=='valid-response'){
                              var item=data.children;
                              if(item.length>0)
                              for(var i=0;i<item.length;i++){
              info+='<tr><td>'+item[i].nodeName+'</td><td>:</td>';
              info+='<td>'+item[i].textContent+'</td></tr>';
                              }
                        }
                        else if(data.nodeName=='invalid-response'){
              info+='<tr><td>code</td><td>:</td><td>200</td></tr>';
              info+='<tr><td>message</td><td>:</td><td>'+
                       data.nodeName+'</td></tr>';
              info+='<tr><td>code</td><td>:</td><td>'+
                       data.textContent+'</td></tr>';
                        }
                        else if(data.nodeName=='error-response'){
              info+='<tr><td>code</td><td>:</td><td>400</td></tr>';
              info+='<tr><td>message</td><td>:</td><td>'+
                       data.nodeName+'</td></tr>';
              info+='<tr><td>code</td><td>:</td><td>'+
                       data.textContent+'</td></tr>';
                        }
                        info+='</table>';
              document.getElementById(elid).innerHTML = info;
                     }
                  }
                  }
            }
            Ajax.send(post_data); delete Ajax;
            return true;
      }
      </script>
</head>
```

```
<body style="margin:8px;font-family:arial;font-size:10pt">
      <strong style='font-size:18pt'>
            Pencarian Data Pegawai
      </strong>
      <hr/>
      <label>NIP:</label>
      <input type='text' name='nip' value='' size='30'/>
      <input type='button' name='cari' value='Cari'
           onclick="document.all.nip?
                    callAjax(elid='result',
                        mode='get',
                        url='http://192.168.1.20/ws/apirest.php',
                        methodName='pegawai',
                        parameters=document.all.nip.value):
                    alert('Unset nip document');"
      />
      <hr/>
      <pre id='result'>
      </pre>
</body>
</html>
```

Hasil dari proses Aplikasi dengan konsep AJAX-Webservice secara langsung adalah sebagai berikut.



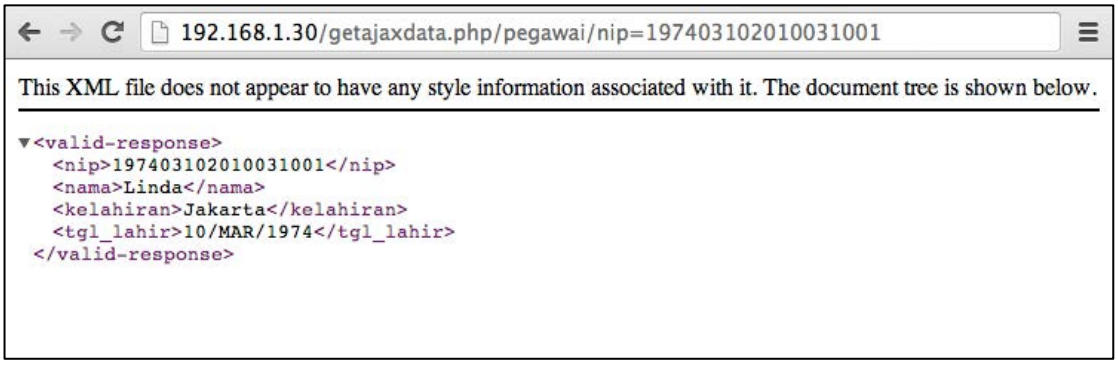
Gambar 1.28 Hasil akses AJAX-Webservice secara langsung

Perlu diperhatikan bahwa mengakses Webservice secara langsung dari Web Browser sangat tidak dianjurkan karena alasan keamanan terhadap Webservice. Idealnya Webservice tidak perlu diketahui oleh Web Browser karena Webservice hanya diperuntukan antar Aplikasi Server (Web Server) yang menggunakannya. Dengan demikian akses ke Webservice pada konsep AJAX sebaiknya dilakukan secara tidak langsung melalui program Adapter di Server seperti contoh program berikut ini.

File: getajaxdata.php

```
<?php
// Programmed by: Didi Sukyadi

require_once 'restclient.php';

function get_response($code,$message,$data){
      $result['code']=$code;
      $result['message']=$message;
      $result['data']=$data;
      return $result;
}

$info=NULL;
$desc='';
$url='http://192.168.1.20/ws/apirest.php';

// Bust cache in the head
header ("Expires: Mon, 07 Jan 1980 01:00:00 GMT");
// Date in the past
header ("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
// always modified
header ("Cache-Control: no-cache, must-revalidate");  // HTTP/1.1
header ("Pragma: no-cache");                          // HTTP/1.0
header ("content-type:text/xml; charset=ISO-8859-1");


$api=new rest_client($url);
$method=$api->requestMethod();
$param=$api->requestParameter();
$desc=$api->status_data();
$info='';
if($api->is_connected()){
      $data=$api->call($method,$param);
      $desc=$api->status_data();
      if($api->status_code()==200){
        $desc=($api->status_message()=='invalid-response')?
              "Data ".$method." tidak ditemukan!":$desc;
      }
}
$info=get_response($api->status_code(),
                  $api->status_message(),$desc);

if($info['message']=='valid-response'){
      echo $info['data'];
}
else{
      echo $api->contentXML($info['message'],$info['data']);
}

?>
```

Jika program getajaxdata.php diakses secara langsung dari Web Browser dengan URL http://192.168.1.30/getajaxdata.php maka hasilnya akan tampak seperti gambaran berikut.

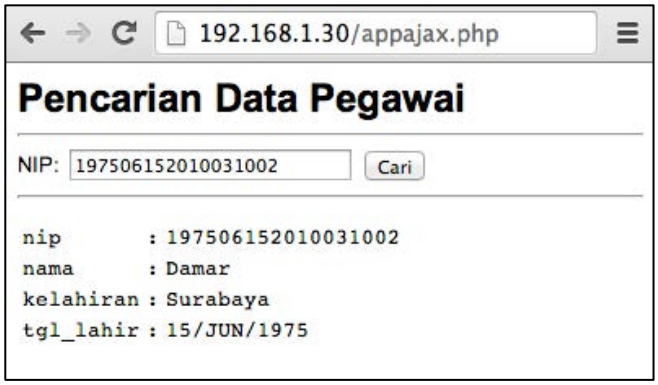


Gambar 1.29 Hasil akses Adapter getajaxdata.php

Program getajaxdata.php yang berfungsi sebagai Adapter dari Webservice digunakan dalam Aplikasi yang menerapkan konsep AJAX dari sisi Client untuk mengakses Webservice di sisi Server. Dengan demikian perubahan yang perlu dilakukan pada program appajax.php adalah mengganti URL Webservice apires.php dengan URL Adapter getajaxdata.php seperti potongan program berikut.

```
<body style="margin:8px;font-family:arial;font-size:10pt">
      <strong style='font-size:18pt'>
            Pencarian Data Pegawai
      </strong>
      <hr/>
      <label>NIP:</label>
      <input type='text' name='nip' value='' size='30'/>
      <input type='button' name='cari' value='Cari'
           onclick="document.all.nip?
                    callAjax(elid='result',
                         mode='get',
                         url='http://192.168.1.30/getajaxdata.php',
                         methodName='pegawai',
                         parameters=document.all.nip.value):
                    alert('Unset nip document');"
      />
      <hr/>
      <pre id='result'>
      </pre>
</body>
```

Hasil dari proses Aplikasi appajax.php dengan memanfaatkan Adapter getajaxdata.php melalui konsep AJAX-Webservice secara tidak langsung akan tampak sebagai berikut.



Gambar 1.30 Hasil akses AJAX-Webservice secara tidak langsung

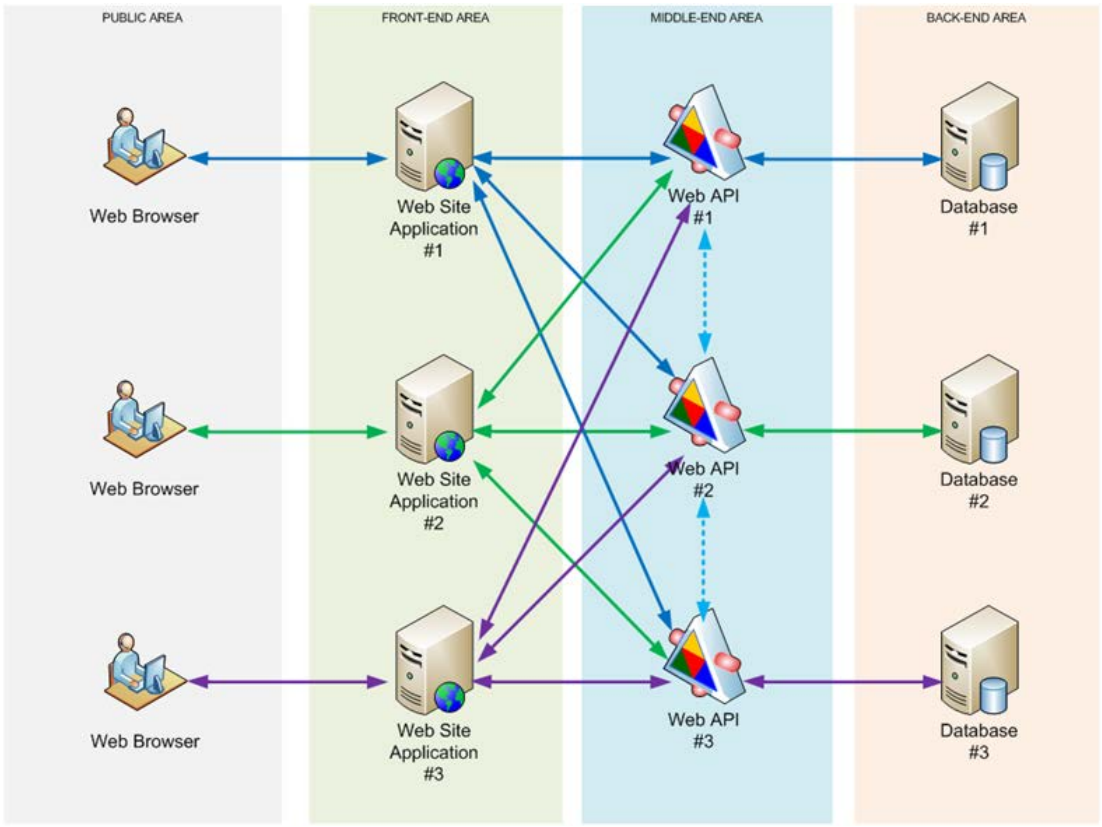
Berdasarkan implementasi konsep AJAX tersebut dapat digambarkan bahwa mekanisme yang direkomendasikan dan tidak direkomendasikan dalam mengakses Webservice dari suatu Aplikasi berbasis Web adalah seperti ilustrasi berikut ini.

Gambar 1.31 Interaksi AJAX-Webservice secara langsung
(Tidak direkomendasikan)

Gambar 1.32 Interaksi AJAX-Webservice secara tidak langsung
(Direkomendasikan)

1.2 Arsitektur Integrasi Sistem Elektronik

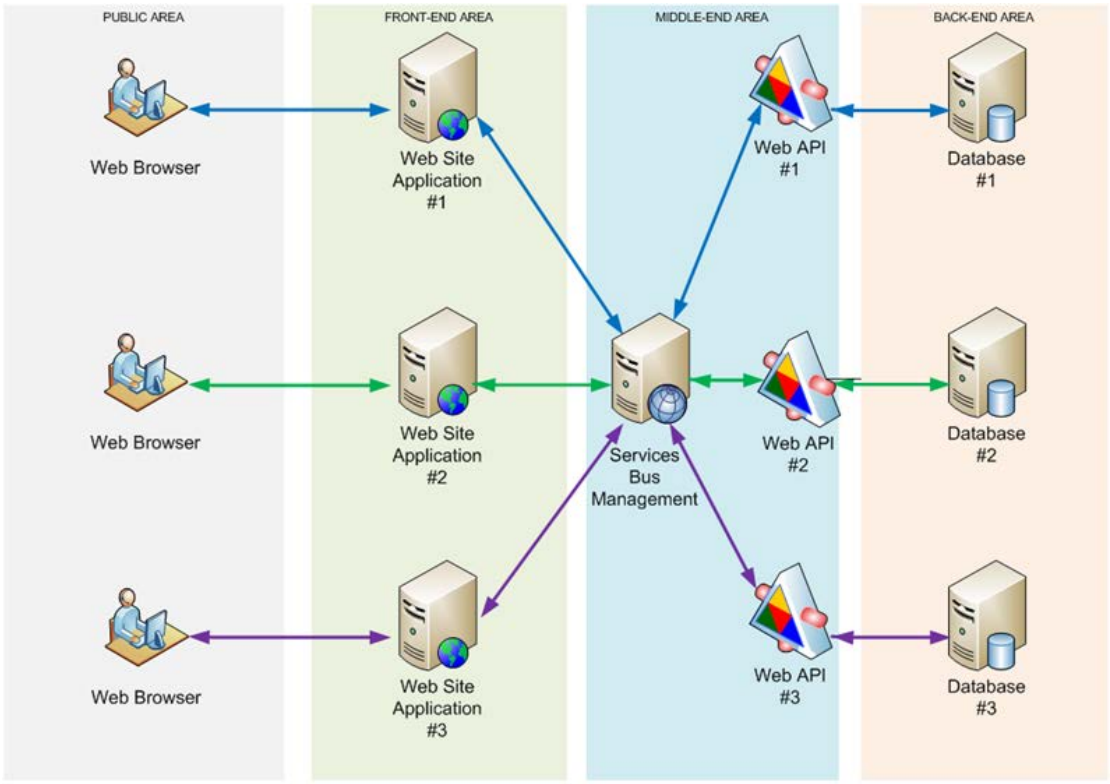
Pemanfaatan akses layanan berbagi pakai data/informasi dari beberapa API Webservice mengakibatkan terbentuknya koneksi jaringan dan lalu lintas komunikasi data antar server dengan topologi yang cukup kompleks seperti ilustrasi berikut ini.



Gambar 1.33 Topologi Layanan API/Webservice Poit To Point (P2P)

Akses suatu Aplikasi ke API secara langsung seperti gambaran di atas akan menyebabkan hilangnya ketersediaan layanan apabila terjadi perpindahan alamat atau perubahan nama domain layanan API Webservice, hal ini akan menyebabkan tidak konsistennya layanan yang diberikan kepada pengguna. Untuk itu sebaiknya akses dari Aplikasi ke layanan API Webservice dimediasikan oleh sebuah HUB/penghubung layanan API dengan teknologi Government Service Bus (GSB), dimana dalam pengelolaannya GSB

berfungsi sebagai Katalog alamat layanan API Webservice dan menjadi Proxy dari layanan API Webservice yang didaftarkan ke dalam GSB oleh Penyedia Layanan API Webservice. Selanjutnya Pihak dari Aplikasi yang memanfaatkan Layanan API Webservice dapat mendaftarkannya di dalam GSB. Dengan demikian perbaikan topologi akses layanan API Webservice melalui GSB tergambar dalam ilustrasi berikut ini.

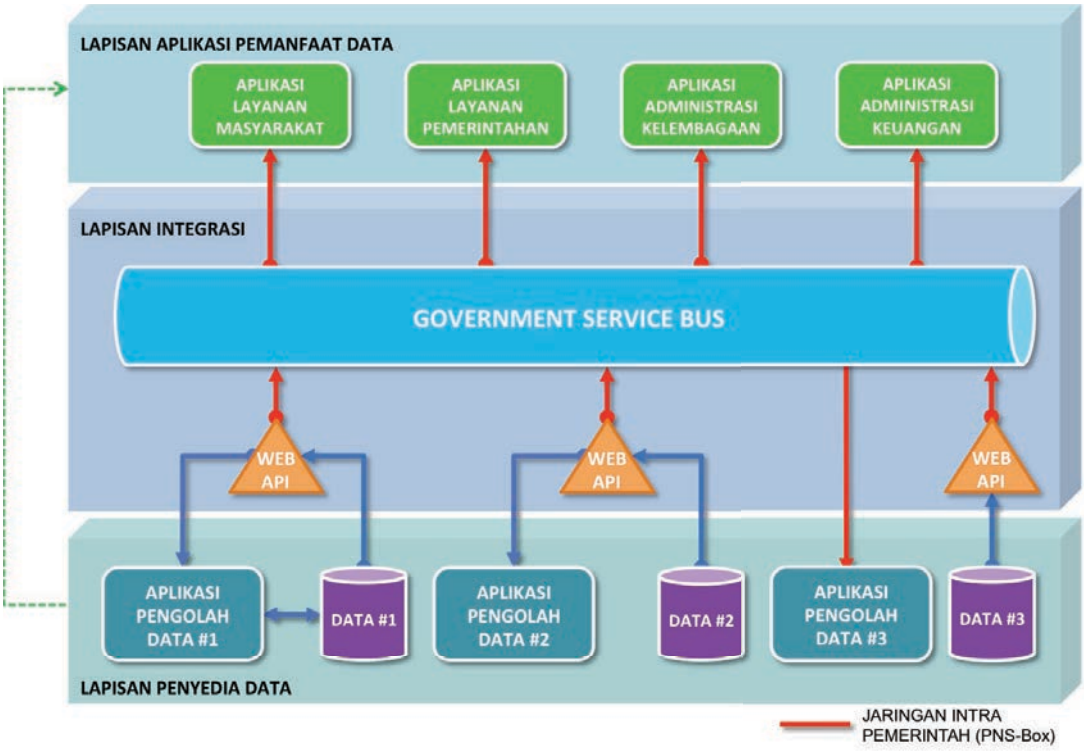


Gambar 1.34 Topologi Layanan API/Webservice melalui GSB

Berdasarkan topologi tersebut dapat disimpulkan akses layanan API Webservice melalui GSB dibagi menjadi 4 lapisan pengelolaan yaitu:

- Penyedia Layanan API dan Data
- Pengelola Integrasi
- Pemanfaat Layanan

Masing-masing pengelola bertanggung jawab atas layanan yang disampaikan dan digunakan dalam GSB, sebagaimana diilustrasikan dalam diagram berikut ini:



Gambar 1.35 Arsitektur Integrasi Layanan API berbasis GSB

Dengan memanfaatkan Arsitektur Integrasi Layanan API Webservice berbasis GSB dan menerapkan kemampuan Interoperabilitas Sistem Elektronik dalam Komputasi Terdistribusi, maka Integrasi antar Sistem Elektronik dapat terselenggara melalui Layanan Berbagi Pakai Data/Informasi antar Sistem Elektronik.

1.3 Spesifikasi Pendukung

Agar Integrasi antar Sistem Elektronik dapat terselenggara dalam Komputasi Terdistribusi, maka Layanan Berbagi Pakai Data/Informasi antar Sistem Elektronik harus didukung dengan spesifikasi pendukung yang antara lain:

- Menggunakan Platform Teknologi berbasis Web dengan protokol HTTP/s.
- Dapat beroperasi pada berbagai Sistem Operasi yang mendukung Teknologi Web.
- Menerapkan konsep interoperabilitas.
- Menetapkan format dokumen/data berbasis standar terbuka.
- Berbasis kode sumber terbuka.
- Memudahkan Pengelolaan Layanan dengan GUI berbasi Web.
- Memenuhi standar aplikasi berbasis Web.

1.4 Manfaat dan Dampak Integrasi Sistem Elektronik

Beberapa manfaat yang diperoleh dari penerapan Integrasi Sistem Elektronik dalam Komputasi Terdistribusi yang memanfaatkan konsep Interoperabilitas melalui Layanan Berbagi Pakai Data/Informasi antara lain:

- Menghemat biaya pengembangan Aplikasi Pengolah Data
- Mengurangi redudansi data
- Pengembangan sistem elektronik berorientasi pada pemanfaatan reusable-resource
- Menetapan arah proses validasi dan verifikasi data
- Mempermudah pengelolaan integrasi informasi dan pertukaran data
- Mendukung kolaborasi data/informasi antar sistem elektronik.

Di samping manfaat terdapat pula dampak fenomena dari Integrasi Sistem Elektronik, di antaranya:

- Secara tidak langsung memberikan dampak bagi Manajemen Pengolahan Data pada suatu organisasi dalam kaitannya dengan pengelolaan sumberdaya informasi masing-masing instansi.
- Beragam tuntutan dan keinginan untuk membentuk relasi dua atau lebih sistem pengolah data yang akan bermuara pada kebutuhan untuk melakukan upaya berbagi pakai (sharing) data dan informasi yang dimiliki masing-masing instansi.

2

PENERAPAN INTEGRASI INFORMASI DAN PERTUKARAN DATA ANTAR SISTEM ELEKTRONIK

2.1 Latar Belakang

Alasan utama disusunnya buku pedoman ini adalah amanat Inpres No.3 Tahun 2003 kepada Kementrian Kominfo, yaitu:

> *Membuat ketentuan tentang standar kelayakan dan interoperabilitas situs informasi dan pelayanan publik dan standarisasi yang berkaitan dengan interoperabilitas pertukaran data serta transaksi informasi antar situs pelayanan publik yang diselenggarakan pemerintah.*

Interoperabilitas yang dimaksud dalam amanat tersebut adalah mekanisme integrasi untuk proses pertukaran data antar sistem elektronik. Dalam melaksanakan amanat tersebut penerapan ketentuan interoperabilitas dipedomani berdasarkan Panduan Penyusunan Rencana Induk e-Government Tahun 2013, yang menyatakan bahwa konsep pengintegrasian dilakukan dalam 2 (dua) tahap, yaitu:

a. Pengintegrasian sistem informasi yang ada melalui antarmuka (interface) tanpa merubah sistem yang digunakan.
b. Pengintegrasian sistem informasi ke dalam satu kesatuan pada setiap lembaga pemerintah.

Dalam penerapan integrasi sistem informasi terdapat 2 (dua) proses kegiatan yang dapat dilakukan, yaitu:

a. Integrasi informasi yang merupakan suatu upaya pengumpulan subset informasi yang tersebar untuk menjadi satu kesatuan informasi (referensi).
b. Pertukaran data antar sistem informasi untuk verifikasi dan validasi data.

Istilah Pertukaran Data secara teori dikenal dengan nama Interoperabilitas, sesuai dengan definisinya berdasarkan IEEE Glossary, yaitu:

> *Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged.*

Interoperabilitas lebih dikenal sebagai mekanisme atau kerangka kerja pertukaran data antar sistem elektronik yang dilakukan dalam rangka integrasi sistem informasi. Singkatnya Interoperabilitas merupakan kemampuan satu atau lebih komponen sistem untuk berbagi pakai data/informasi.

Pengelolaan Integrasi Sistem Elektronik diperlukan karena:

a. Adanya kebutuhan konstituen antar instansi pemerintah untuk saling bekerja sama.
b. Terjadinya pengolahan data pada tiap instansi pemerintah yang membutuhkan referensi data dari instansi lainnya.
c. Dukungan regulasi/kebijakan pemerintah tentang integrasi data elektronik.

Fenomena lain yang mendorong dilakukannya Integrasi Sistem Informasi antara lain:

- Penggabungan/pemisahan sejumlah organisasi di instansi pemerintah.
- Restrukturisasi organisasi instansi pemerintah.
- Strategi kerjasama antar instansi pemerintah pada lintas sekotral untuk meningkatkan kinerja birokrasi.
- Tuntutan berbagai mitra kerjasama antar lembaga dalam dan luar negeri untuk meningkatkan kualitas aliansi dan kolaborasi.

(Sumber: Richardus Eko Indrajit, 2006)

2.2 Tahapan Pengembangan

Strategi Integrasi Informasi dan Pertukaran Data dapat dilaksanakan dengan 2 pendekatan, meliputi:

- Pendekatan Non Teknis
  Melakukan evolusi kebutuhan Integrasi Informasi dan Pertukaran Data antar Sistem Elektronik
- Pendekatan Teknis
  Melaksanakan tahapan teknis Integrasi Informasi dan Pertukaran Data melalui Teknologi terkini

Evolusi Kebutuhan Penerapan Integrasi Informasi dan Pertukaran Data yang harus dilaksanakan adalah:

- Eksploitasi kapabilitas lokal
  Tujuan dari dilakukannya tahap ini adalah untuk memahami secara sungguh-sungguh batasan maksimal kemampuan sistem informasi dalam menghasilkan kebutuhan manajemen strategis dan operasional organisasi yang bersangkutan, baik dilihat dari segi keunggulannya maupun keterbatasannya.

- Integrasi tersembunyi

  Secara vertikal, masing-masing sistem informasi tetap melayani setiap organisasi terkait, sementara secara horisontal telah dilakukan proses integrasi melalui penambahan komponen-komponen baru hasil diskusi beragam organisasi yang terlibat (misalnya: interface, middleware, application integration system, database clearing house, dsb.).

  Keluaran sesungguhnya dalam tahap ini adalah kepercayaan dan kesadaran akan perlunya kerjasama untuk memecahkan solusi.

- Berbagi Pakai

  Melihat bahwa banyak peluang untuk meningkatkan kinerja solusi yang dihasilkan jika dan hanya jika adanya ”sharing” atau pola berbagi pakai antar sumber daya teknologi informasi yang dimiliki masing-masing organisasi. Dalam konteks inilah mulai terlihat adanya tawaran untuk misalnya menggunakan server dari organisasi A, aplikasi dari organisasi B, database dari organisasi C, jaringan dari organisasi D, dan lain sebagainya.

- Redesain Arsitektur Proses

  Menggunakan kebutuhan pemegang kepentingan akhir (yaitu pelanggan atau publik) sebagai target solusi redesain. Dengan berpegang pada konsep dan teori BPR (Business Process Reengineering) sejumlah usaha untuk melakukan eliminasi, simplifikasi, integrasi, dan otomatisasi proses akan dilakukan.

- Optimalisasi Infrastruktur

  Proses optimalisasi bertujuan untuk memenuhi kebutuhan pemegang kepentingan utama dengan batasan tetap dijaganya kinerja masing-masing sistem informasi untuk melayani organisasi yang ada secara vertikal.

- Transformasi Organisasi

  Tahap terakhir yang akan dicapai sejalan dengan semakin eratnya hubungan antar organisasi adalah transformasi masing-masing organisasi. Transformasi yang dimaksud pada dasarnya merupakan akibat dari dinamika kebutuhan lingkungan eksternal organisasi yang memaksanya untuk menciptakan sebuah sistem organisasi yang adaptif terhadap perubahan apapun.

Terdapat beberapa kategori teknologi yang tersedia untuk penerapan integrasi informasi dan pertukaran data. Teknologi tersebut harus dikaji sesuai kebutuhan, antara lain:

- Teknologi Keterpaduan Data

  Beberapa teknologi yang menyediakan solusi keterpaduan data antara lain teknologi mainframe, server file sederhana seperti Samba atau Network Neighbourhood, server repositori seperti CVS maupun SVN, datamart dan data warehouse juga teknologi RSS.

- Teknologi Keterpaduan Proses

  Beberapa teknologi yang menyediakan solusi keterpaduan proses antara lain Remote Procedure Call, CORBA maupun DCOM. Umumnya solusi keterpaduan proses tergantung atau diimplementasikan pada pemrograman tertentu seperti C++, Java. Teknologi keterpaduan proses dapat berupa pemanggilan antar pemrograman yang masih memiliki teknologi yang mirip seperti RMI pada Java programming untuk mengakses kode native C atau C++ programming.

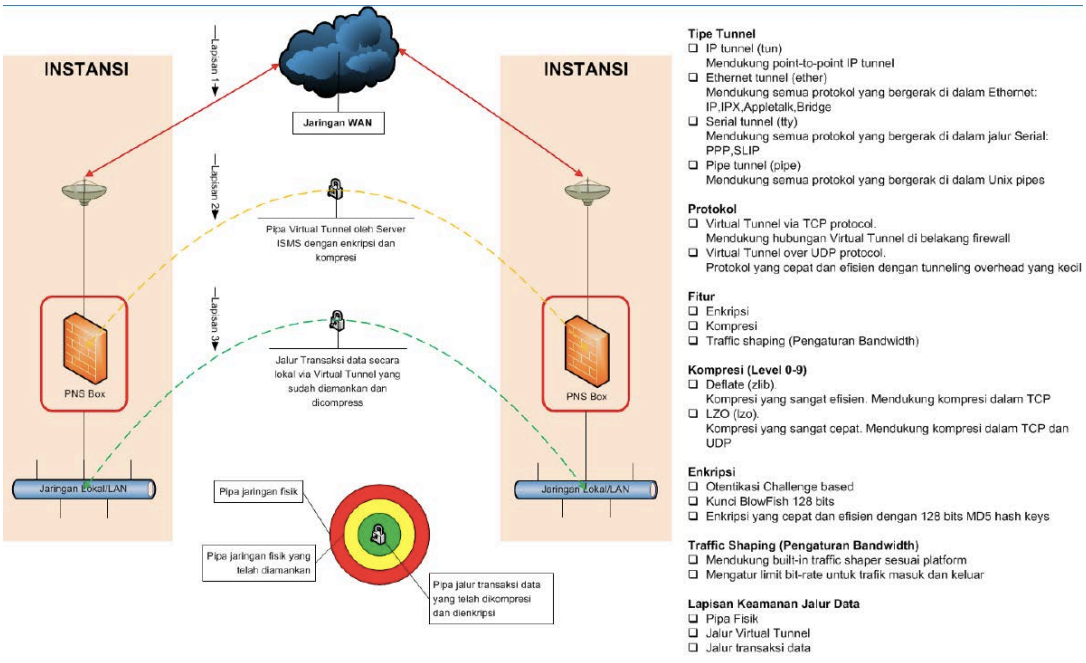
- Teknologi keterpaduan proses dan data

  Teknologi yang menyediakan solusi keterpaduan data dan proses adalah web service. Sebagai penyedia data, web service menyediakan dokumen standar terbuka yang dapat diakses oleh berbagai aplikasi menggunakan berbagai jenis teknologi seperti

database, pemrograman maupun platform sistem. Webservice dapat mengintegrasikan proses menjadi penyedia layanan bagi aplikasi maupun penghubung antar aplikasi menggunakan teknologi middleware.

2.3 Pengamanan Infrastruktur Jaringan

Integrasi Sistem Elektronik sangat membutuhkan keterhubungan melalui infrastruktur jaringan yang aman, hal ini dikarenakan ada banyak data yang harus dilewati menjadi lalu lintas yang perlu diamankan selama perjalanan sampai tujuan akhir. Pengamanan infrastruktur jaringan dilakukan pada jalur internet publik yang diselubungi pengaman jalur khusus sehingga data yang melintas tidak dapat dilihat siapapun diluar jalur khusus tersbut. Cara pengamanan melalui jalur khusus tersebut dikenal dengan nama Virtual Private Network (VPN), dimana untuk memasuki jalur khusus tersebut harus melalui proses verifikasi dan otentikasi terlebih dahulu.

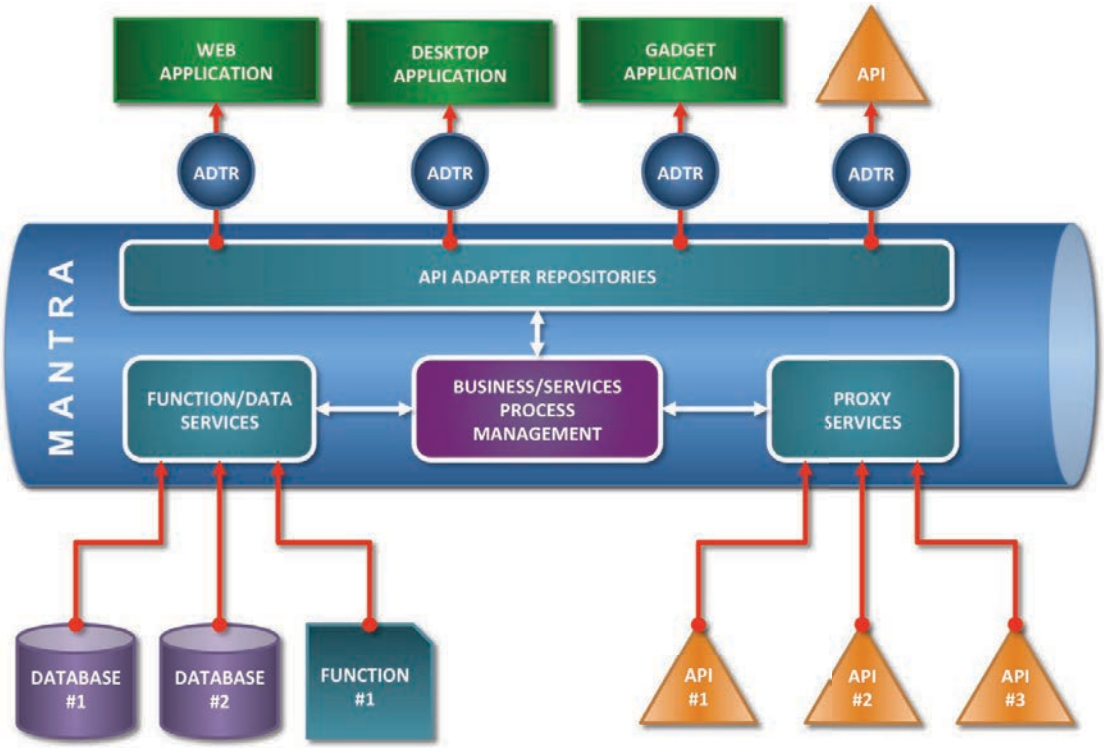


Gambar 2.1 Arsitektur PNS-Box

VPN yang digunakan untuk Layanan Berbagi Pakai Data/Informasi sebagai langkah proses Integrasi Sistem Elektronik dirancang khusus untuk jalur intra pemerintah menggunakan aplikasi PNS-Box (Private Network Security Box) sebagaimana ilustrasi arsitekturnya pada Gambar 2.1.

2.4 Pengelolaan Aplikasi Mantra

Layanan Berbagi Pakai Data/Informasi sebagai proses Integrasi Sistem Elektronik perlu dikelola melalui aplikasi berbasis Kode Sumber Terbuka (Open Source), agar pengembangan teknologi yang dibutuhkan dapat dilakukan secara adaptif. Aplikasi tersebut dinamakan MANTRA, singkatan dari MANAJEMEN INTEGRASI INFORMASI DAN PERTUKARAN DATA. Penamaan yang erat kaitannya dengan Manajemen tersebut dilandaskan pada fungsi Aplikasi tersebut sebagai Pengelolaan Layanan Berbagi Pakai Data/Informasi antar Sistem Elektronik. Aplikasi MANTRA memiliki arsitektur yang diilustrasikan pada gambaran berikut ini.



Gambar 2.2 Arsitektur Aplikasi MANTRA

Secara umum aplikasi MANTRA merupakan perangkat lunak pendukung Kerangka Kerja Interoperabilitas Sistem Informasi Elektronik dalam bentuk Layanan Berbagi Pakai Data/Informasi sebagai proses Integrasi Sistem Elektronik Pemerintahan. Fitur-fitur manajemen yang terdapat dalam aplikasi MANTRA diilustrasikan dalam gambar berikut ini.

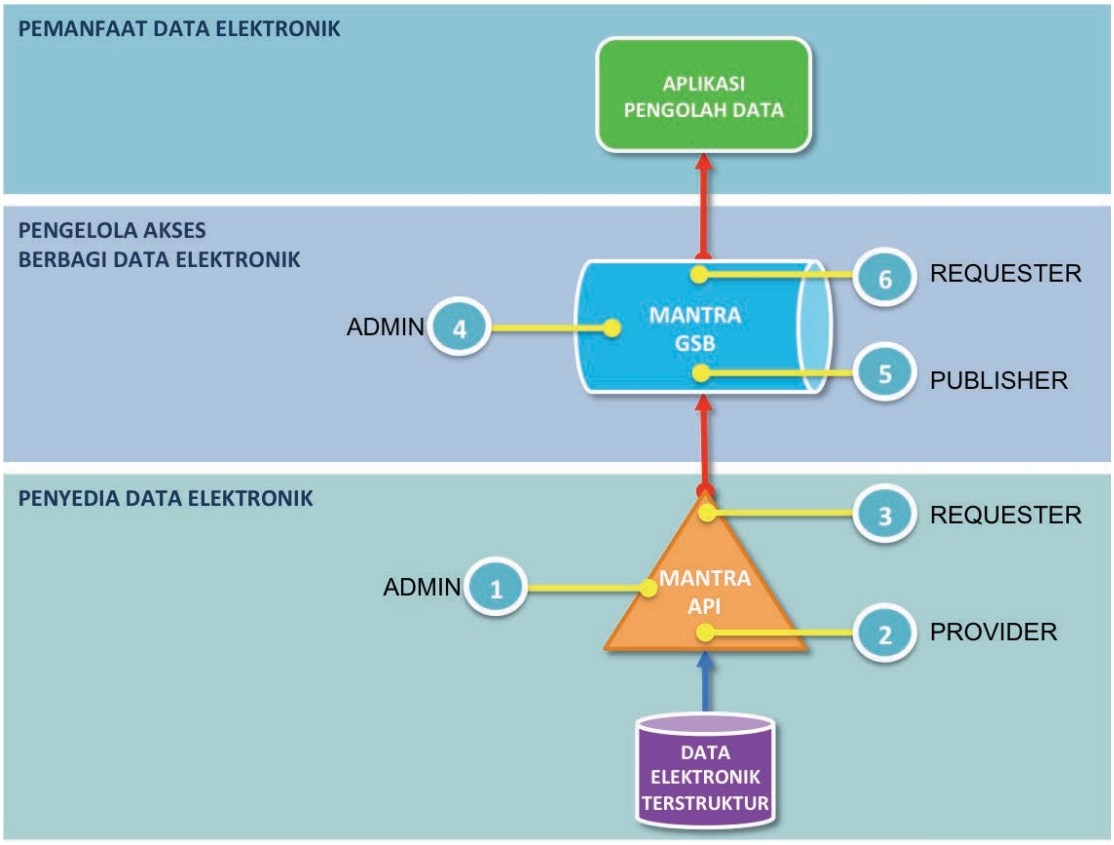
Gambar 2.3 Fitur Aplikasi MANTRA

Dari gambaran arsitektur dan fitur-fitur aplikasi tersebut menunjukan bahwa aplikasi MANTRA dapat dibagi menjadi 2 (dua) fungsi, yaitu:

- Sebagai penyedia layanan data (Data Services) yang terhubung langsung ke Database, dinamakan API Webservice.
- Sebagai penyedia hub layanan (Proxy Services) yang terhubung dengan layanan lain, dinamakan GSB.

Dua fungsi tersebut tersedia secara terpadu dalam aplikasi MANTRA dan dapat dikelola berdasarkan peran pengelola yang diatur oleh Administrator aplikasi MANTRA. Masing-masing peran pengelola

menunjukan peran aplikasi MANTRA ada di lapisan mana seperti yang diilustrasikan dalam gambaran berikut ini.



Gambar 2.4 Pengelolaan Layanan Berbagi Pakai Data/Informasi

Prosedur pengelolaan layanan berbagi pakai data/informasi menggunakan aplikasi MANTRA yang digambarkan pada ilustrasi tersebut adalah sebagai berikut:

1. Administrator (Admin) Web-API menyediakan akses pengguna untuk Penyedia (Provider) dan Pemanfaat (Requester) Layanan Akses Berbagi Pakai Data Elektronik melalui Web-API.
2. Penyedia (Provider) membuat Layanan Operasi Data pada Web-API melalui pendefinisian data dan elemennya.
3. Pemanfaat dari pihak penyedia (Requester) dalam hal ini yang terdaftar sebagai Publisher di GSB memesan Layanan Operasi

Data pada Web-API yang akan didaftarkan ke GSB dan menyalin rute aksesnya setelah disetujui Penyedia (Provider).

4. Administrator (Admin) GSB menyediakan akses pengguna untuk Publisher dan Pemanfaat (Requester) Layanan Akses Berbagi Pakai Data Elektronik melalui GSB.
5. Pempublikasi (Publisher) mendaftarkan rute akses Layanan Operasi Data dari Web-API Penyedia ke dalam layanan GSB.
6. Pemanfaat dari pihak lain (Requester) memesan Layanan Operasi Data pada GSB dan menyalin rute aksesnya (setelah disetujui Publisher pihak penyedia) ke dalam Aplikasi yang akan memanfaatkannya.

Pada dasarnya aplikasi MANTRA memiliki beberapa level/tingkatan peran pengguna berdasarkan instansi/penyedia layanan secara hirarki dari yang terendah sampai yang tertinggi sebagai berikut:

- Requester, berperan sebagai pihak pengguna yang memanfaatkan Layanan Operasi Data yang disediakan oleh Aplikasi MANTRA.
- Publisher, berperan sebagai pihak penyedia yang mendaftarkan rute Akses Layanan Operasi Data dari Web-API yang dimiliki ke dalam MANTRA GSB.
- Provider, berperan sebagai pihak yang membuat Layanan Operasi Data pada Web-API.
- Administrator, berperan sebagai pihak yang mengatur pengelolaan Aplikasi MANTRA secara teknis.
- Supervisor, berperan sebagai pihak yang melakukan monitoring aktivitas integrasi informasi dan pertukaran data pada Aplikasi MANTRA.

3

PANDUAN PENGGUNAAN APLIKASI MANTRA

3.1 Kebutuhan Perangkat

Untuk menggunakan aplikasi MANTRA diperlukan beberapa perangkat pendukung minimal antara lain:

- Perangkat Lunak (Software):

  Web Server Application : Apache versi 2.4

  Web Preprocessing : PHP versi 5.5

  Database Management : MySQL versi 5.x

  Operating System : Linux, Unix, Windows

- Perangkat Keras (Hardware):

  Processor : 1GHz Multicore, Hyperthreading

  RAM : 8 GByte

  Storage : 250 GByte

  NIC : 100 Mbps Ethernet

  Bandwidth : 10 Mbps

Spesifikasi yang dimiliki aplikasi MANTRA adalah sebagai berikut:

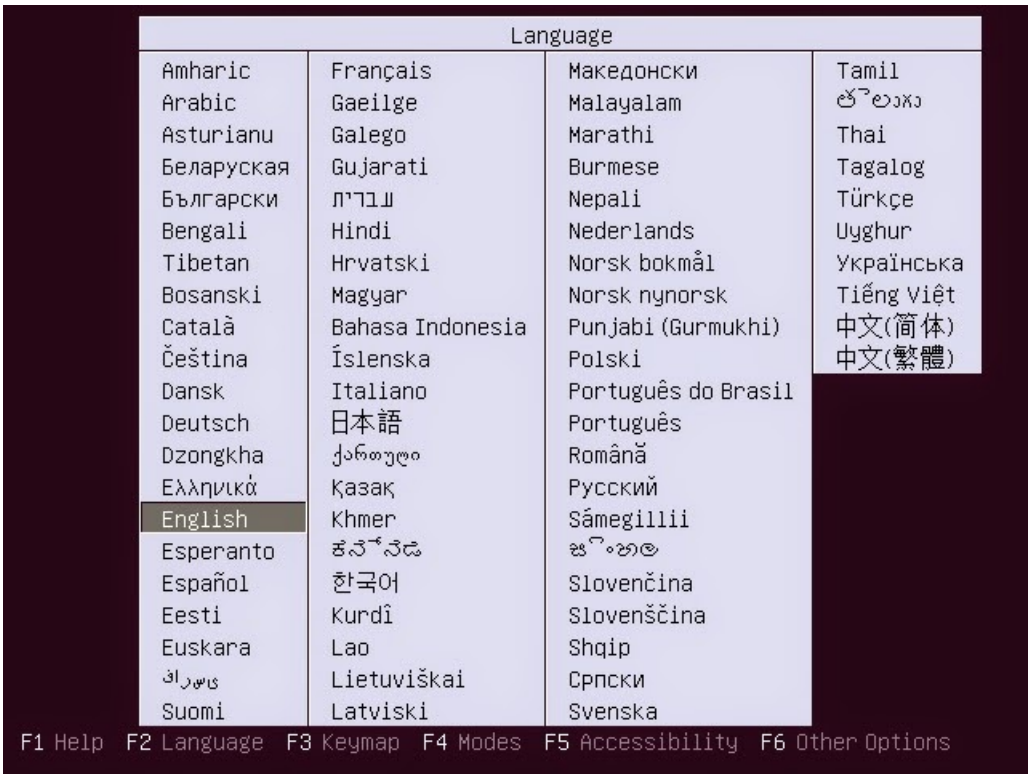
- Menggunakan Teknologi Web dengan protokol HTTP/s yang mendukung Multi-platform.
- Beroperasi pada berbagai Sistem Operasi yang mendukung Teknologi Web.
- Mendukung Konsep Interoperabilitas melalui adapter SOAP dan REST.

- Format data yang digunakan dalam transaksi adalah XML, JSON, PHP-ARRAY, dan PHP-Serialize.
- Menyediakan Akses Layanan Berbagi Data Elektronik (Data Services) dan Akses Berbagi Antar Layanan (Proxy Services).
- Layanan Data Terstruktur yang mendukung DBMS MySQL, PostgreSQL, MS-SQL/Sybase, ORACLE, DBASE/FOXPRO, CSV.
- Memudahkan pengelolaan operasi layanan dengan antarmuka pengguna berbasis GUI dan Web.
- Memberikan peluang pengembangan pemrograman berbasis Kode Sumber Terbuka (Open Source Code)
- Memenuhi standar aplikasi berbasis Web.

3.2 Instalasi Sistem Operasi

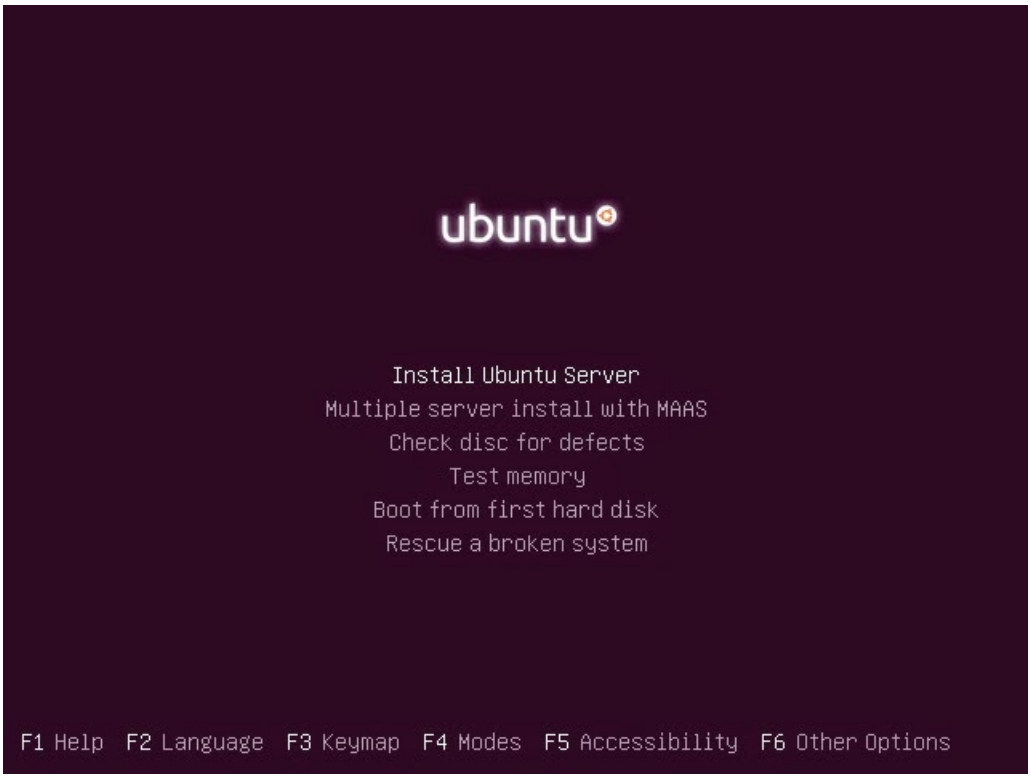
Persiapan pertama yang harus dilakukan sebelum menggunakan aplikasi MANTRA adalah memasang Sistem Operasi pada PC-Server yang akan dijadikan server Layanan Berbagi Pakai Data/Informasi. Sistem Operasi yang digunakan dalam panduan ini adalah Server versi 14 LTS. Langkah-langkah instalasi Ubuntu versi 14 adalah sebagai berikut:

1. Pengaturan prioritas *booting*, untuk instalasi menggunakan CD/DVD silahkan atur CD/DVD Room sebagai prioritas boot utama. Sedangkan untuk instalasi menggunakan USB Flash Drive, silahkan atur USB Removable Media sebagai prioritas boot utama.
2. Masuk *Installation Wizard* Ubuntu. Pilih mode bahasa untuk melakukan instalasi, pilih “English” atau “Bahasa Indonesia”, tekan Enter.



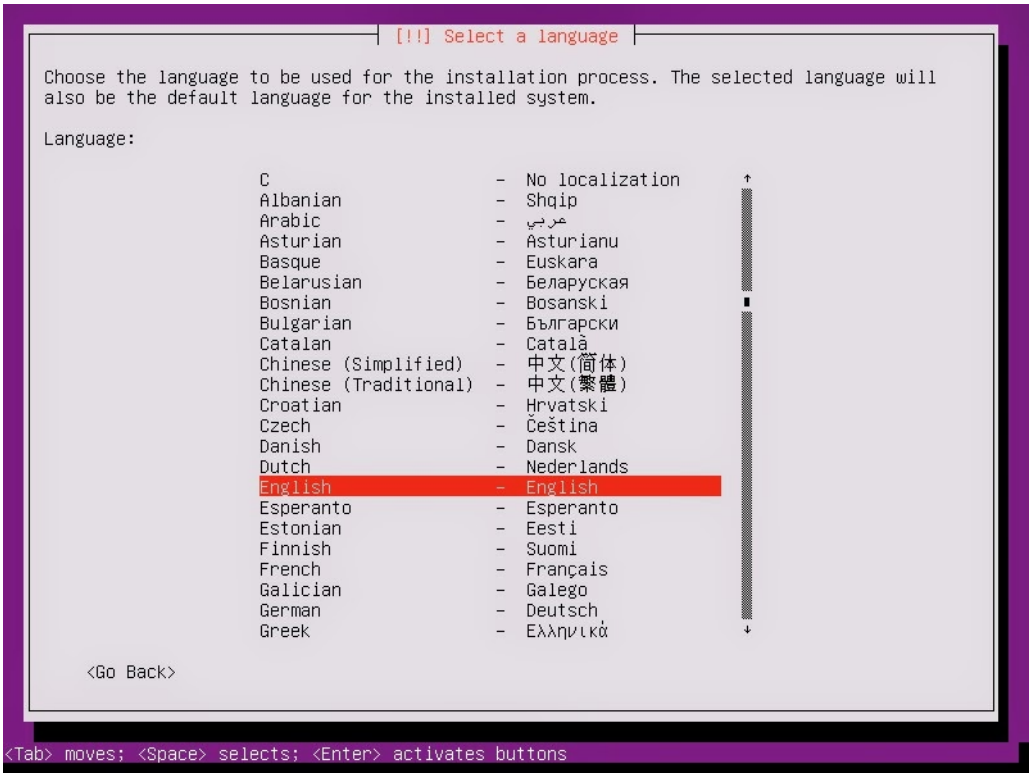
Gambar 3.1 Pemilihan Mode Bahasa

3. Pilih “Install Ubuntu Server”, tekan Enter.



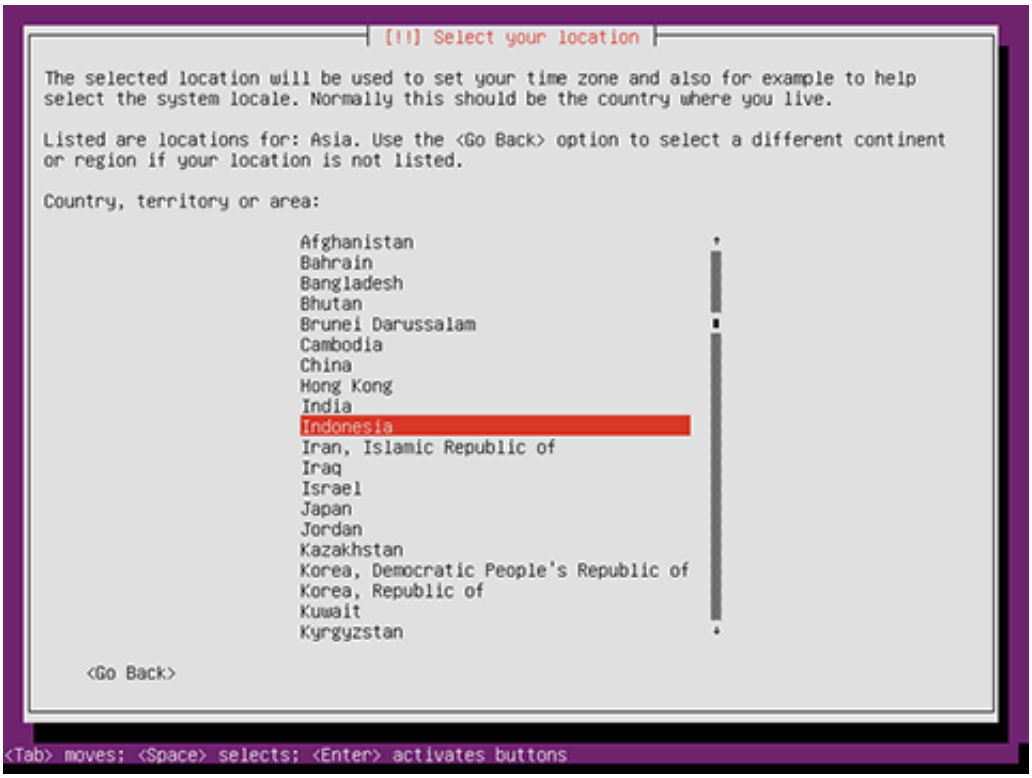
Gambar 3.2 Pemilihan Menu Operasi

4. Pilih mode bahasa standar Sistem Operasi, pilih “English”, tekan Enter.



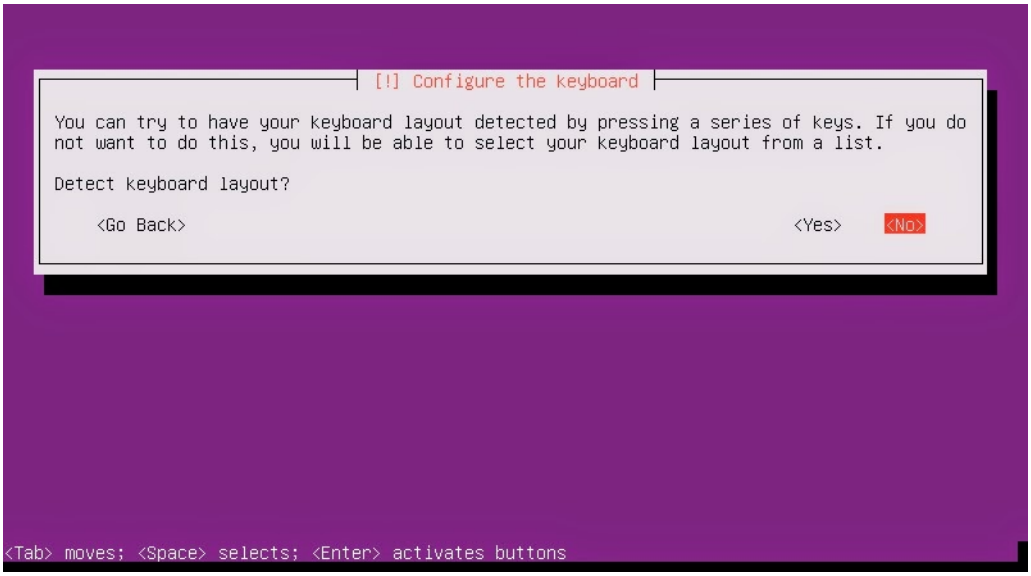
Gambar 3.3 Pemilihan Mode Bahasa Sistem Operasi

5. Pengaturan lokasi server, lokasi tersebut yang akan digunakan untuk menentukan zona waktu. Pilih nama negara tempat kita berada. Untuk memilih “Indonesia” caranya Pilih “other”, pilih region “Asia”, dan kemudian pilih “Indonesia” lalu tekan enter.



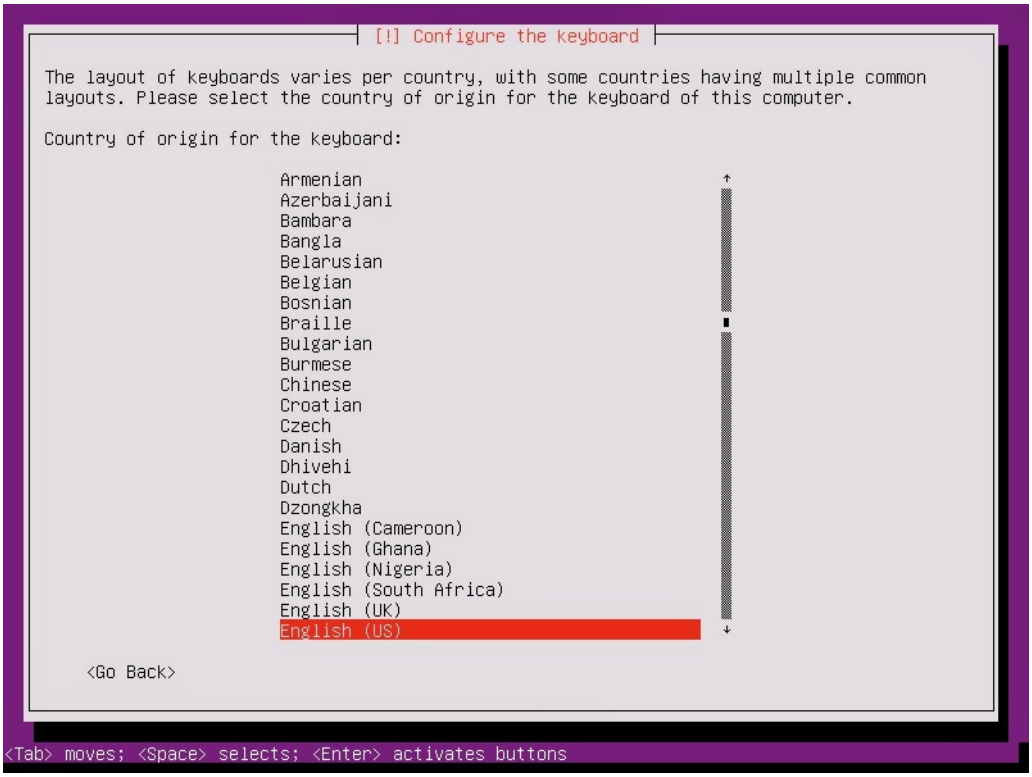
Gambar 3.4 Pemilihan Lokasi Server

6. Pengaturan Layout Keyboard, Pilih “Yes” untuk mengatur ulang keyboard atau pilih “No” untuk melewati langkah ini dan melanjutkan ke langkah selanjutnya.

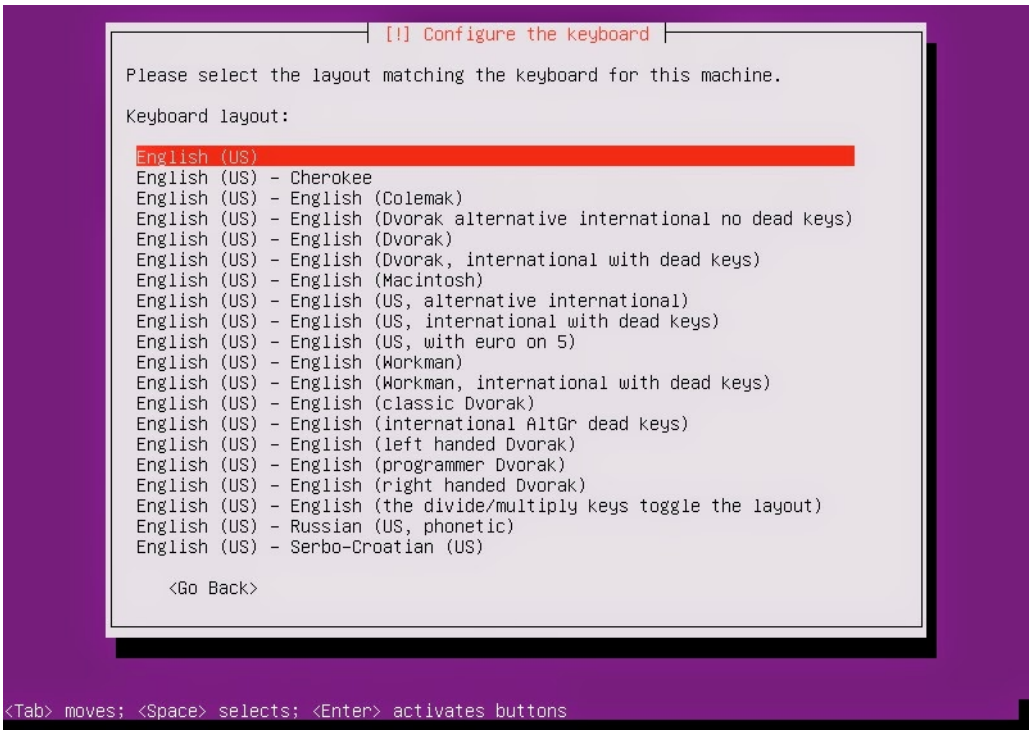


Gambar 3.5 Pemilihan Layout Keyboard

Pilih layout keyboard yang digunakan, untuk layout keyboard yang digunakan di Indonesia adalah English (US).

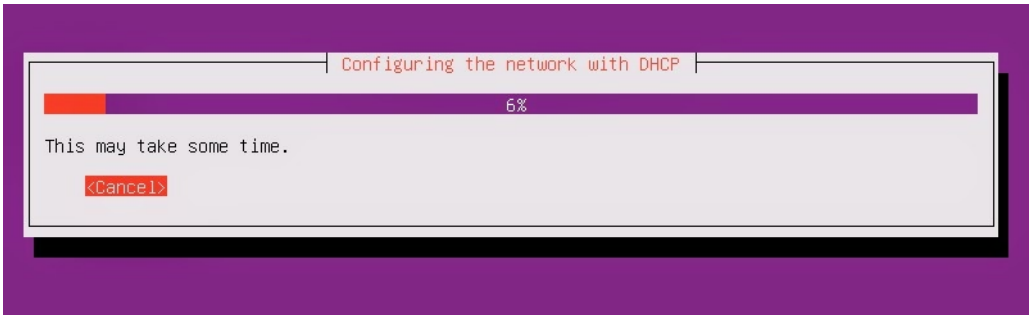


Gambar 3.6 Pemilihan Kategori Layout Keyboard



Gambar 3.7 Pemilihan Tipe Layout Keyboard

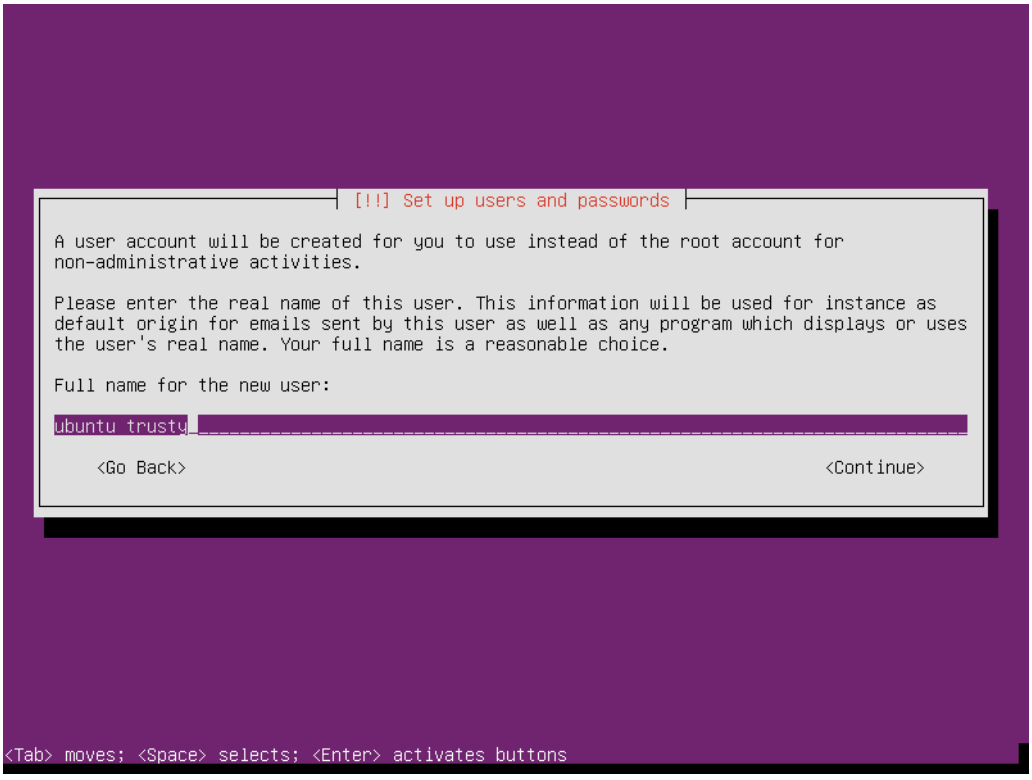
7. Proses ekstraksi dan konfigurasi sistem operasi dimulai.



Gambar 3.8 Proses Instalasi Sistem Operasi

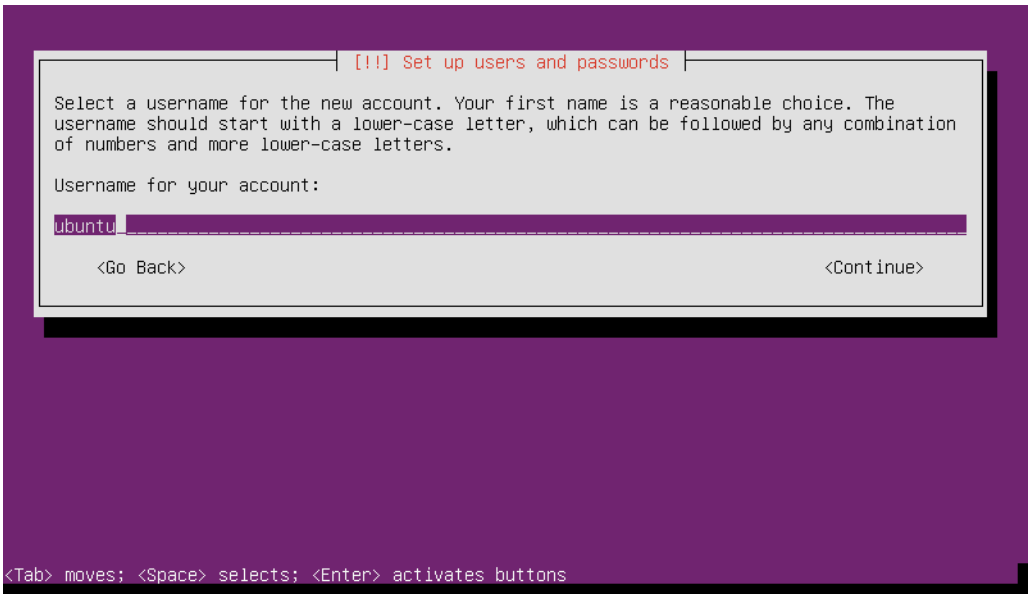
8. Pembuatan akun sistem operasi.

   Isi nama lengkap pemilik akun, tekan tombol "tab" untuk memilih menu "continue", kemudian tekan "enter".



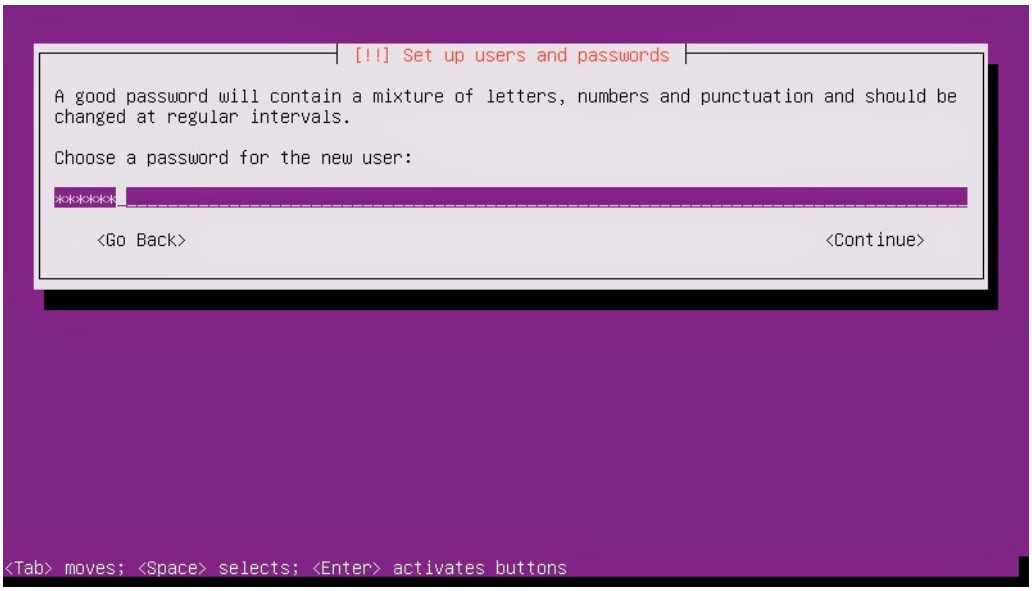
Gambar 3.9 Pengaturan Nama Pemiik Akun Sistem Operasi

Isi nama akun untuk login ke dalam sistem operasi, tekan tombol “tab” untuk memilih menu “continue”, kemudian tekan “enter”.



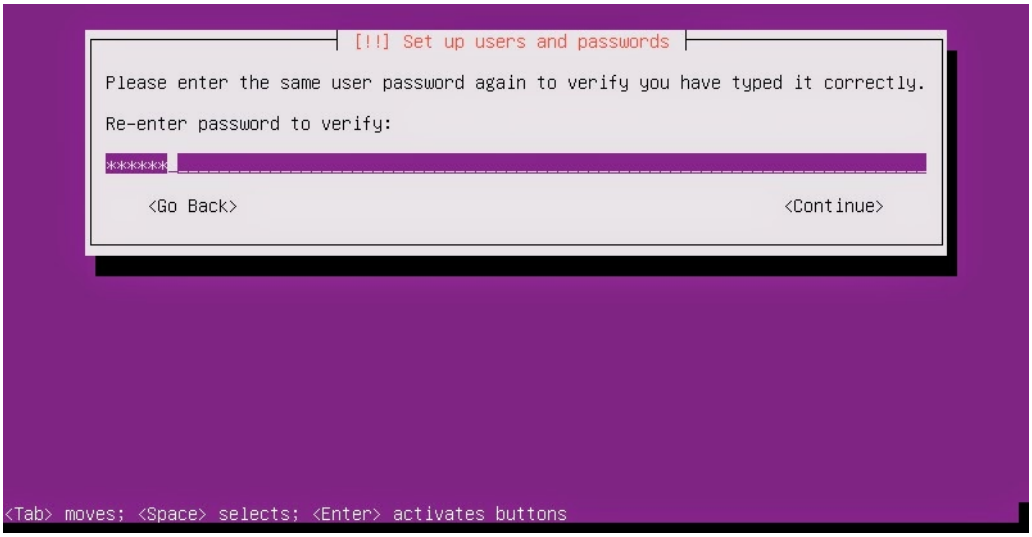
Gambar 3.10 Pengaturan Nama Akun Sistem Operasi

Isi kata kunci untuk akun yang dibuat, tekan tombol “tab” untuk memilih menu “continue”, kemudian tekan “enter”.



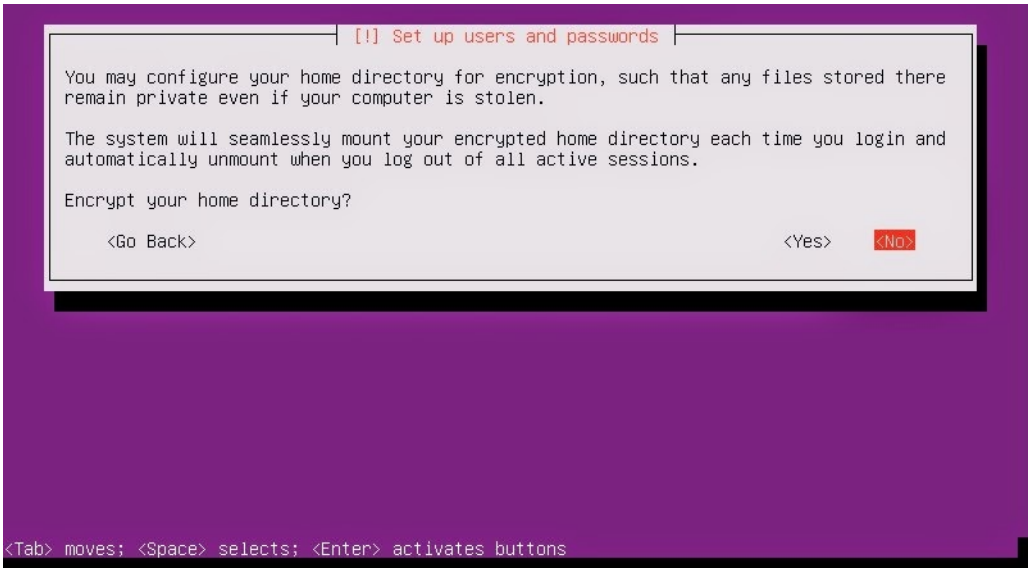
Gambar 3.11 Pengaturan Kata Kunci Akun

Konfirmasi ulang kata kunci untuk akun yang dibuat, tekan tombol “tab” untuk memilih menu “continue”, kemudian tekan “enter”.



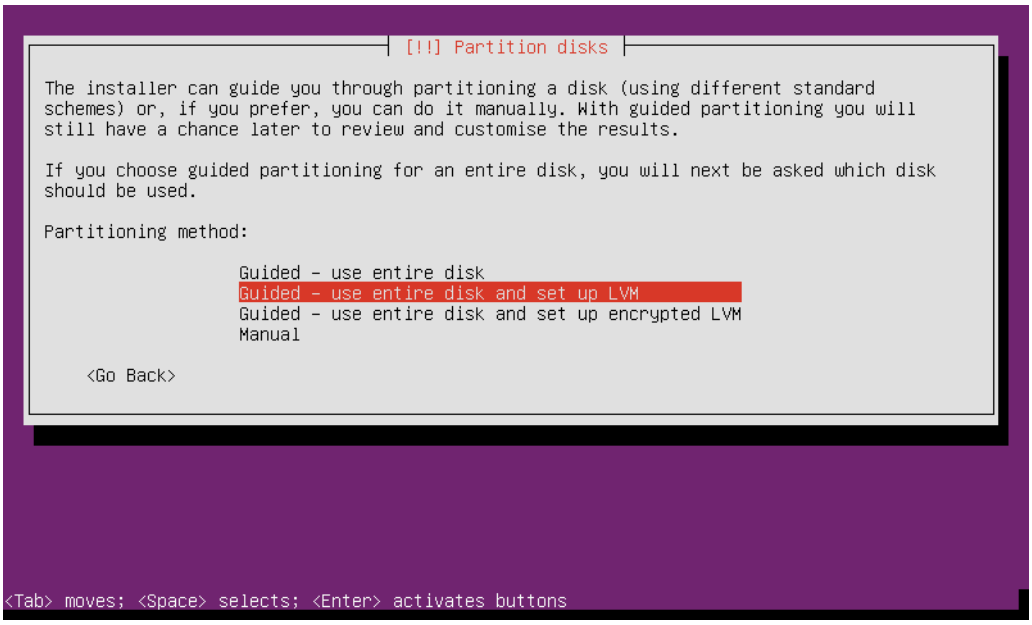
Gambar 3.12 Konfirmasi Ulang Password

Pengaturan enkripsi berkas yang tersimpan dalam home direktori. Pilih “Yes” untuk melakukan penyimpanan berkas dengan enkripsi atau “No” untuk penyimpanan berkas tanpa enkripsi.



Gambar 3.13 Pengaturan Enkripsi Berkas

9. Pembuatan partisi hard disk.



Gambar 3.14 Pembuatan Partisi Penyimpanan Data

Terdapa 4 (empat) pilihan dalam pembuatan partisi hard disk:

- Guided - use entire disk

  Pembuatan partisi hard disk secara otomatis menggunakan seluruh ruang penyimpanan data yang tersedia.
- Guided - use entire disk and set up LVM

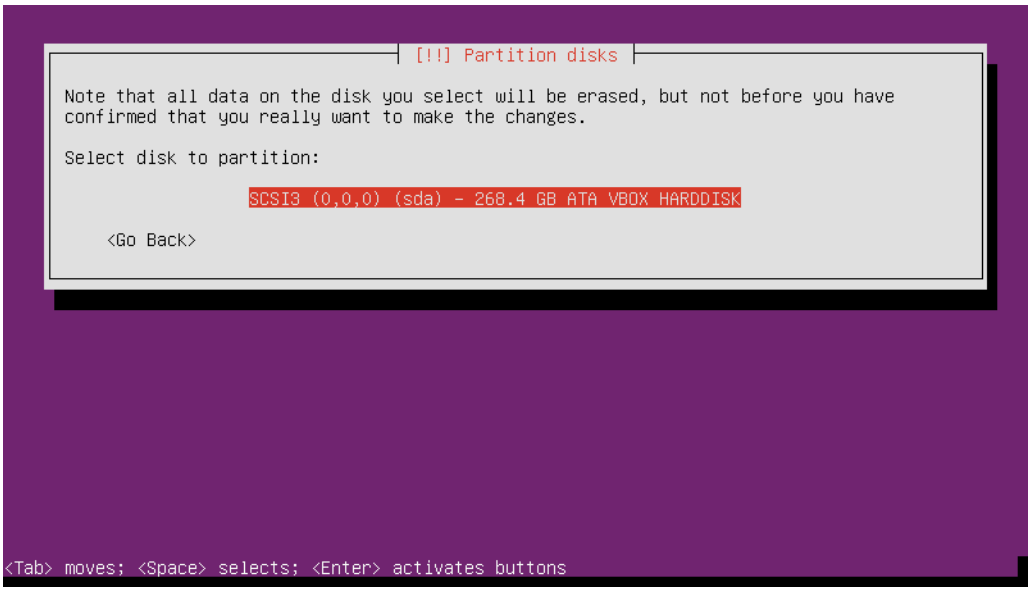
  Pembuatan partisi hard disk secara otomatis menggunakan seluruh ruang penyimpanan data yang tersedia, disertai tambahan fitur LVM (Logical Volume Manager) yang memungkinkan pengaturan hard disk dalam jumlah besar atau banyak. Dengan adanya LVM memungkinkan administrator untuk menambah, mengganti, menyalin dan berbagi isi dari satu hard disk ke hard disk lain tanpa perlu mengganggu operasional sistem operasi.
- Guided - use entire disk and setup encrypt LVM

  Pembuatan partisi hard disk secara otomatis menggunakan seluruh ruang penyimpanan data yang tersedia, disertai tambahan fitur LVM dan Enkripsi LVM.

- Manual

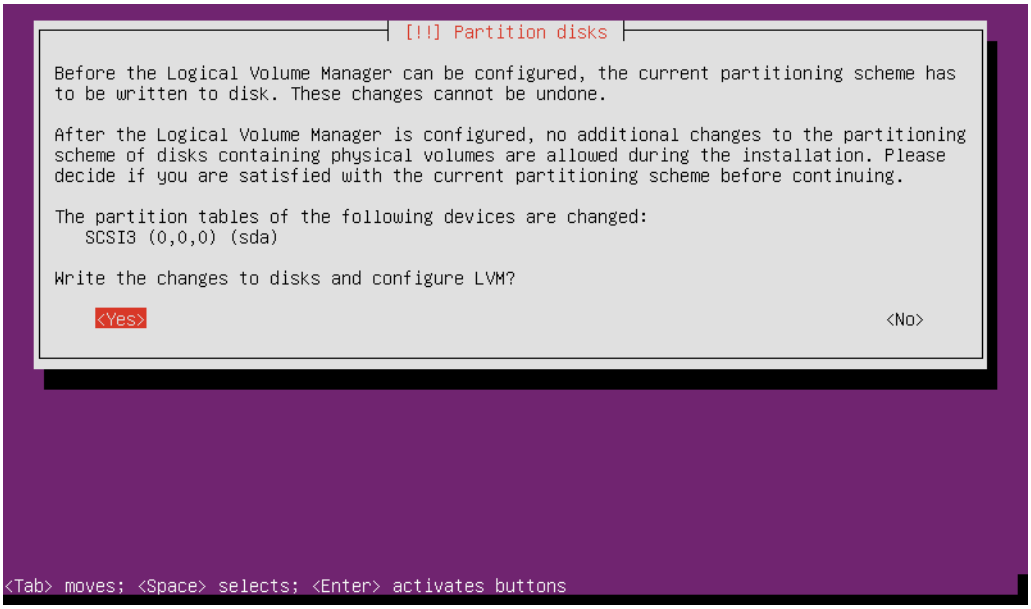
  Pembuatan partisi hard disk secara manual.

Tutorial ini menggunakan pilihan Guided - use entire disk and set up LVM. Tekan tombol “enter” untuk melanjutkan. Kemudian pilih hard disk yang akan dipartisi dan tekan tombol “enter” untuk melanjutkan.



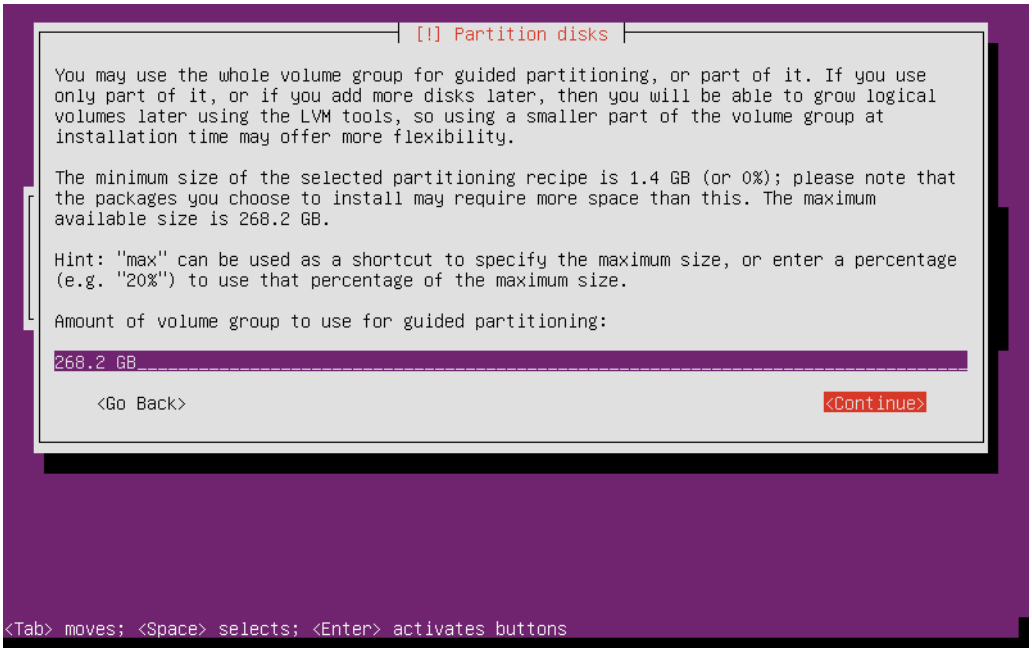
Gambar 3.15 Pemilihan Hard Disk

Pilih “Yes” untuk melakukan konfirmasi perubahan partisi.



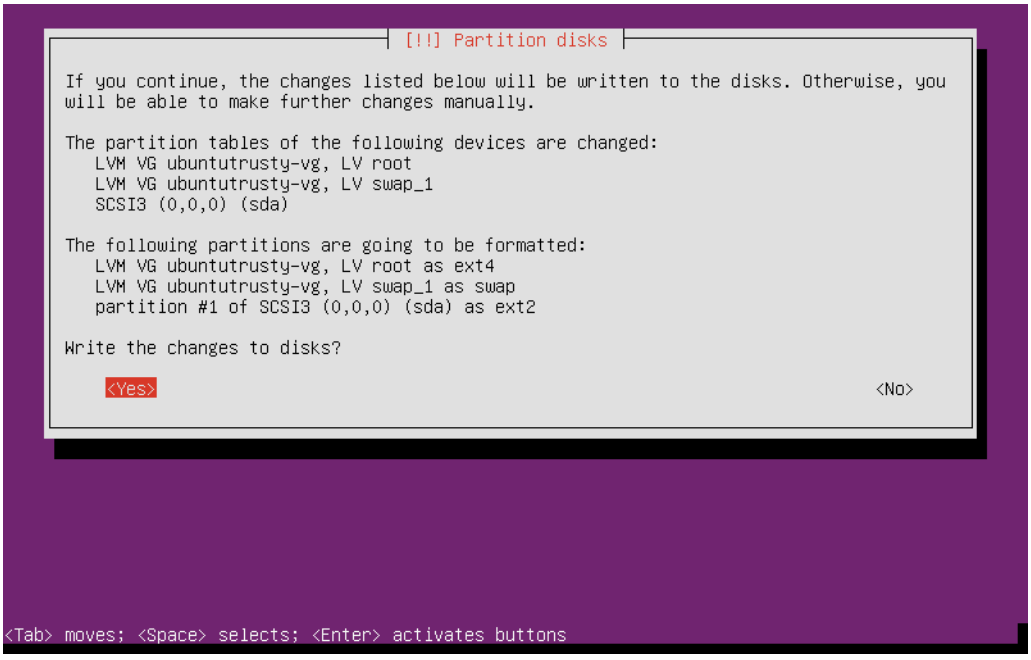
Gambar 3.16 Konfirmasi Perubahan Partisi

Tentukan ukuran partisi, tekan tombol “tab” untuk memilih menu “continue”, kemudian tekan “enter”.



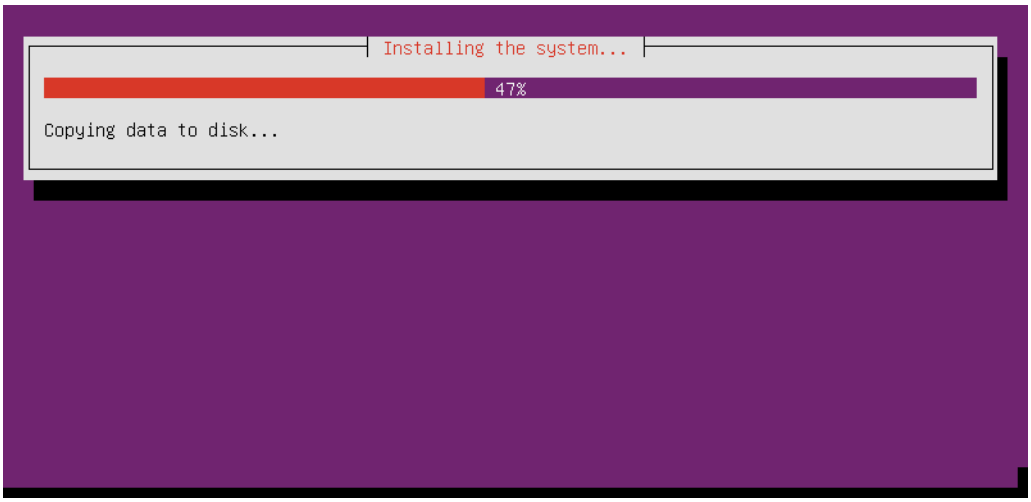
Gambar 3.17 Penenutan Ukuran Partisi

Konfirmasi pembuatan partisi, pilih “No” untuk kembali ke halaman penentuan ukuran partisi, atau pilih “Yes” untuk memulai proses pembuatan partisi.



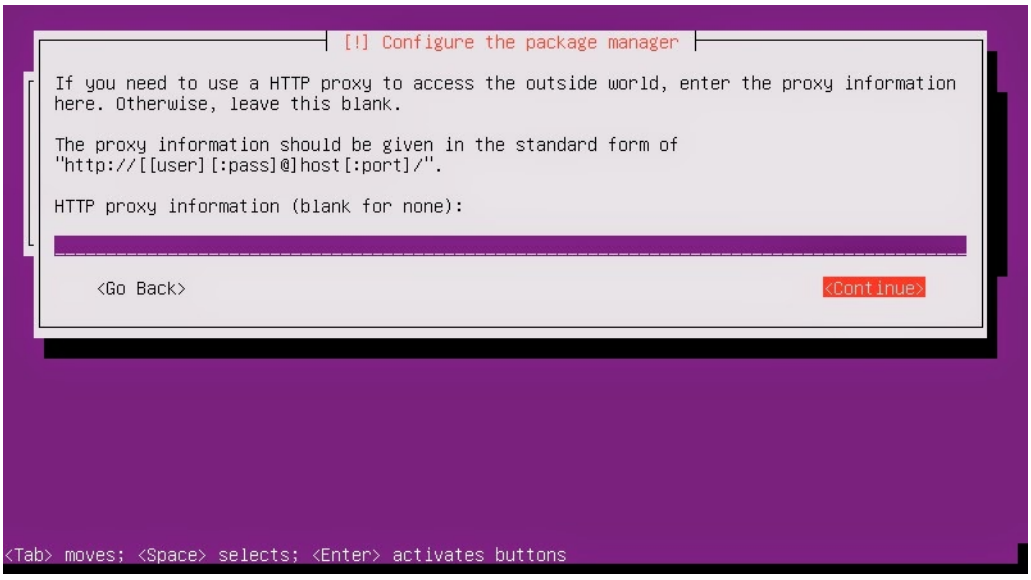
Gambar 3.18 Konfirmasi Pembuatan Partisi

Proses pembuatan partisi dan salin file ke dalam partisi.



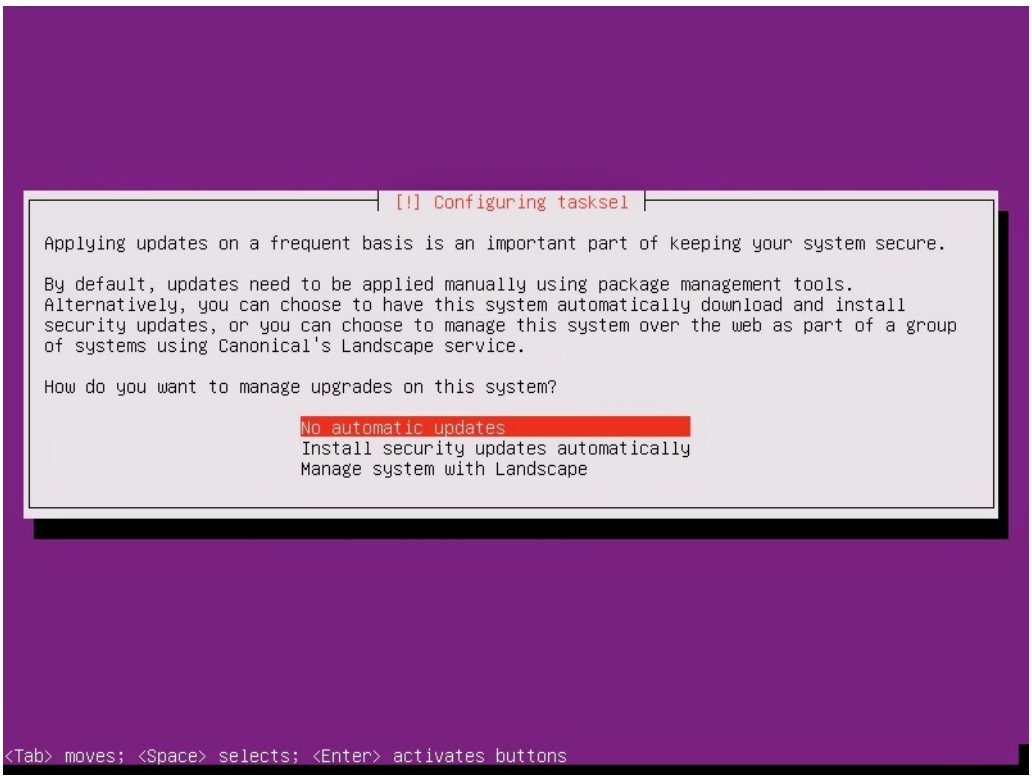
Gambar 3.19 Proses Instalasi Sistem

10. Pengaturan proxy (jika diperlukan) untuk mengakses jaringan internet.



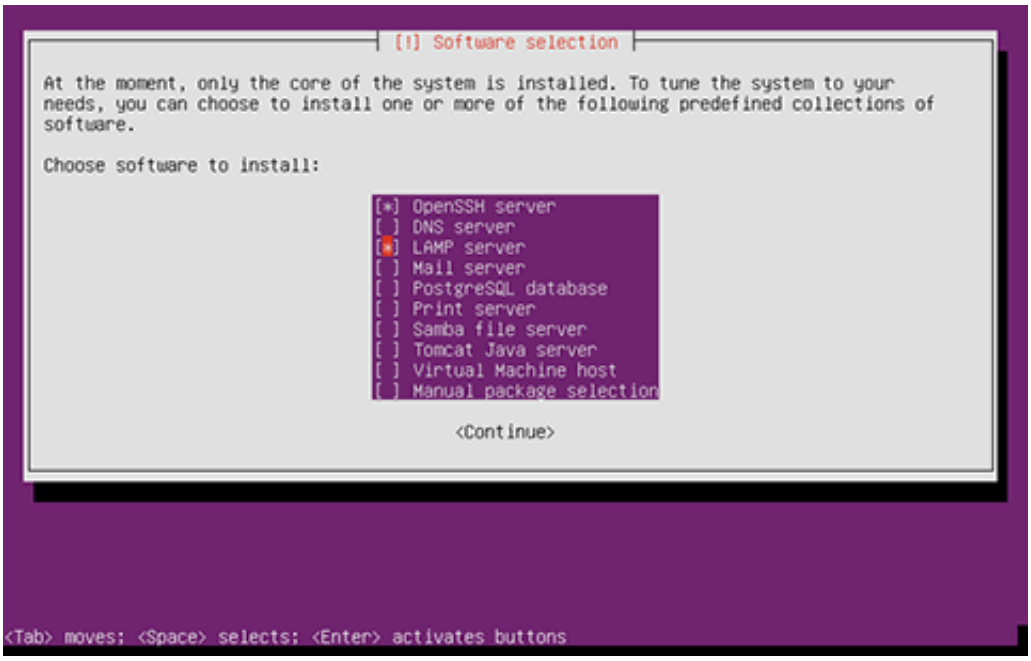
Gambar 3.20 Pengaturan Proxy Internet

11. Pengaturan proses update, direkomendasikan menggunakan mode “install security update automatically”.

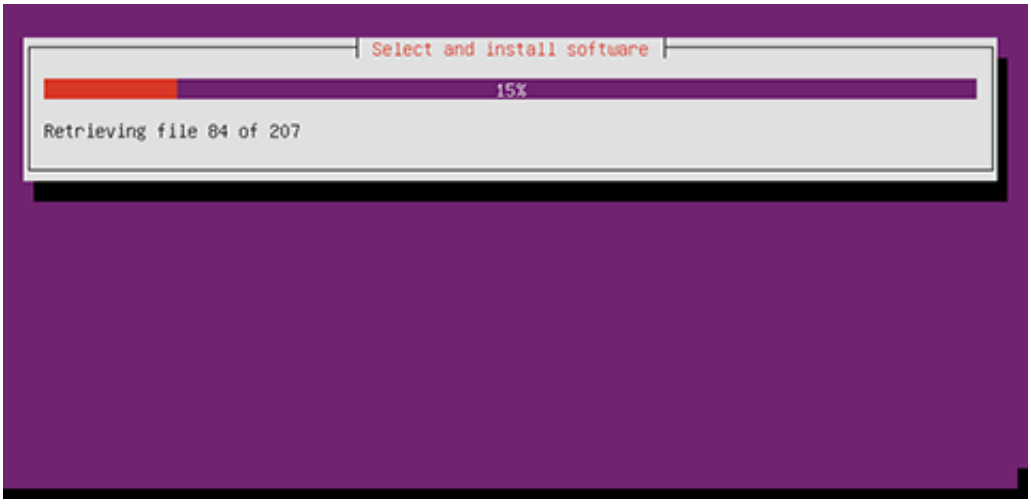


Gambar 3.21 Pengaturan Update Sistem Operasi

12. Pemilihan modul tambahan dalam instalasi sistem operasi. Untuk kebutuhan operasional Aplikasi MANTRA, modul tambahan yang dibutuhkan antara lain OpenSSH Server dan LAMP Server. OpenSSH dibutuhkan agar server dapat diakses secara remote melalui jaringan. LAMP merupakan modul yang berisi Web Server Apache dan Basis Data MySQL. Gunakan tombol “space” untuk menandai modul yang akan dipasang dipasang. Tekan tombol “tab” untuk memilih menu “continue”, kemudian tekan “enter” untuk memulai instalasi modul tambahan.

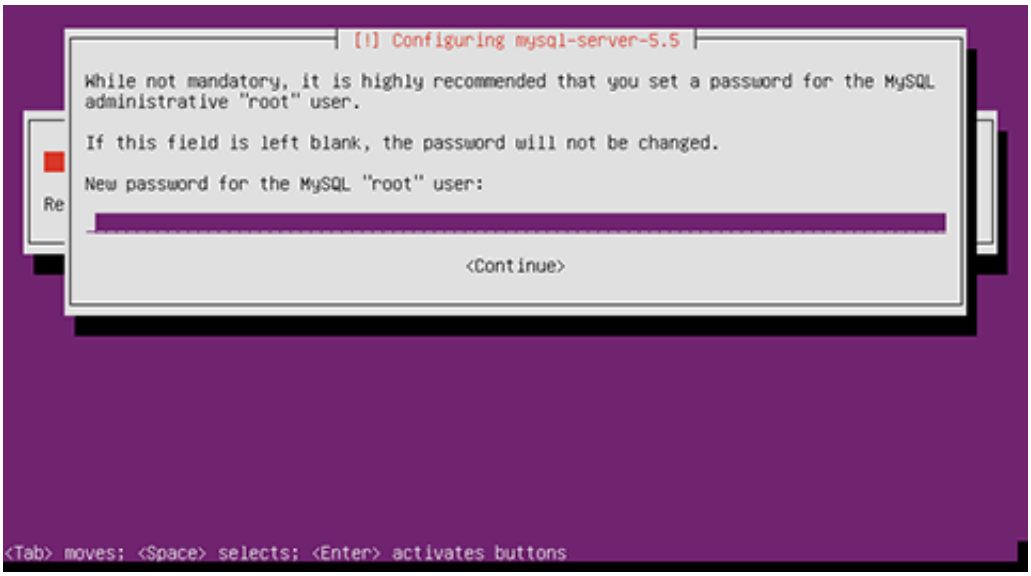


Gambar 3.22 Pemilihan Modul Tambahan



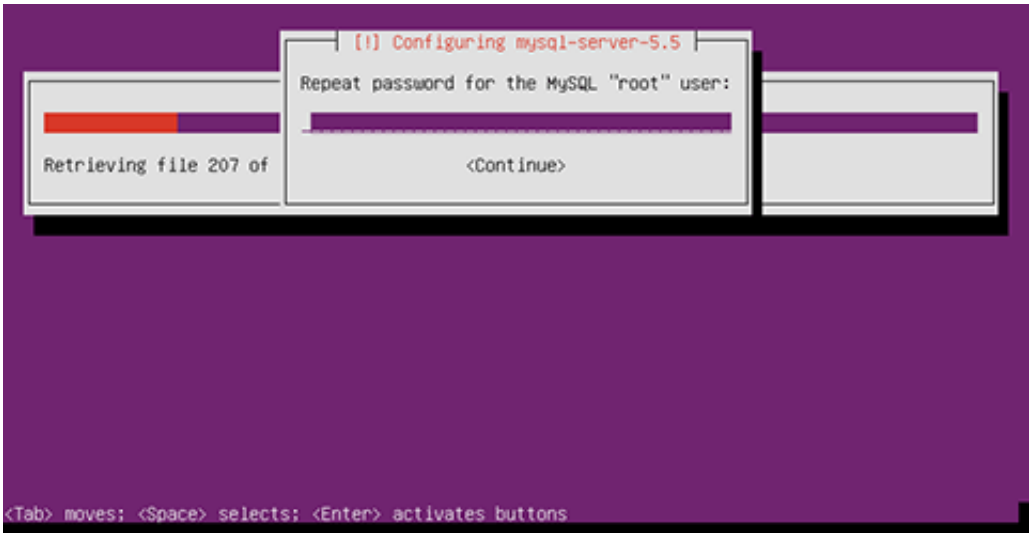
Gambar 3.23 Instalasi Modul Tambahan

13. Pengaturan MySQL password, jika melakukan pemilihan instalasi LAMP Server pada langkah ke 12 (dua belas) di atas, pengguna akan diminta untuk mengisikan password root (administrator) MySQL. Password tersebut digunakan oleh administrator untuk melakukan login ke dalam basis data MySQL.



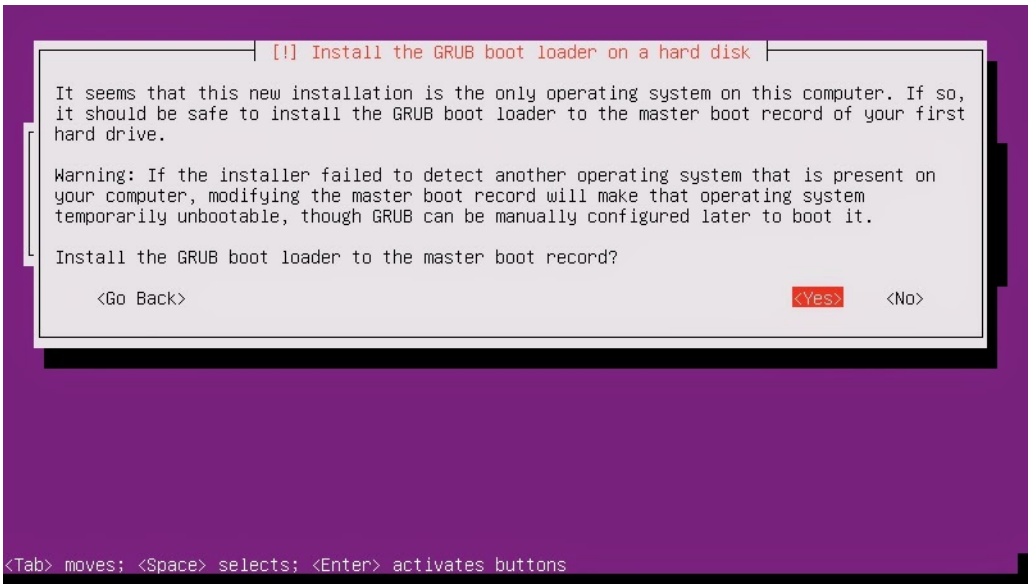
Gambar 3.24 Pengaturan Password MySQL

Konfirmasi ulang password administrator (root) MySQL.



Gambar 3.25 Konfirmasi Password MySQL

14. Instalasi GRUB loader, pilih “yes” kemudian tekan enter untuk melanjutkan proses instalasi.



Gambar 3.26 Instalasi GRUB Loader

15. Penyelesaian instalasi, tekan tombol "tab" pilih "continue" dan tekan tombol "enter". Sistem akan melakukan *reboot* secara otomatis. Silahkan lakukan konfigurasi untuk menyesuaikan prioritas boot menggunakan hard disk sebagai pilihan pertama.
16. Login ke dalam sistem menggunakan akun dan kata kunci yang dibuat pada langkah ke 8 (delapan).

Gambar 3.27 Login Sistem Operasi

3.3 Instalasi Perangkat Lunak dan Modul Pendukung

Sebelum melakukan instalasi Aplikasi MANTRA persiapan awal yang perlu dilakukan adalah melakukan pemasangan Web Server Apache, PHP dan MySQL. Jika pada saat instalasi sistem operasi modul LAMP Server tidak terpasang maka pengguna dapat menggunakan petunjuk ini untuk melakukan instalasi secara manual. Selain instalasi beberapa perangkat lunak tersebut di atas, Aplikasi MANTRA juga membutuhkan beberapa modul PHP antara lain:

1. Modul pendukung aplikasi:
    - php5-curl, modul proses konten Web Client via URL
    - php5-openssl, modul lapisan keamanan socket
    - php5-tidy, modul pengaturan justifikasi format dokumen
    - php5-gd, modul pengaturan grafik di halaman web
    - php5-fileinfo, modul proses informasi berkas

2. Modul pendukung database:
    - php5-mysql, driver untuk Basis Data MySQL
    - php5-pgsql, driver untuk Basis Data Postgre SQL
    - php5-sybase, driver untuk Basis Data Microsoft SQL Server
    - oci8, driver untuk Basis Data Oracle

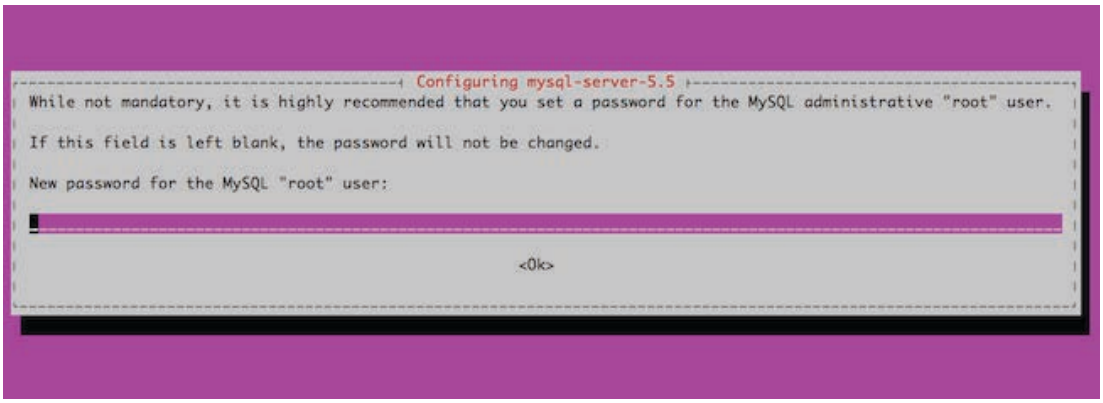
Petunjuk instalasi perangkat lunak dan modul pendukung yang dipaparkan dalam buku ini adalah petunjuk instalasi pada sistem operasi Ubuntu Server versi 14 LTS.

3.3.1 Pemasangan Basis Data MySQL

Perintah instalasi Basis Data MySQL bersi 5 yang direkomendasikan untuk menjalankan Aplikasi MANTRA adalah sebagai berikut:

```
$ sudo apt-get install mysql-server mysql-client
```

Pada saat proses instalasi berlangsung Anda diminta menentukan password user "root" Basis Data MySQL.



Gambar 3.28 Pengisian Password Root MySQL

Setelah proses instalasi selesai dilakukan, MySQL Server akan berjalan secara otomatis pada sistem server. Untuk melihat status operasional MySQL Server gunakan perintah berikut:

```
$ sudo service mysql status
```

Untuk menonaktifkan MySQL Server gunakan perintah berikut:

```
$ sudo service mysql stop
```

Untuk mengaktifkan MySQL Server gunakan perintah berikut:

```
$ sudo service mysql start
```

3.3.2 Pemasangan Web Server Apache

Web Server Apache yang direkomendasikan untuk menjalankan Aplikasi MANTRA adalah Apache Versi 2. Berikut perintah instalasi Web Server Apache pada Ubuntu:

```
$ sudo apt-get install apache2
```

Setelah proses instalasi selesai, lanjutkan dengan mengaktifkan modul rewrite pada apache.

```
$ sudo a2enmod rewrite
$ sudo service apache2 restart
```

3.3.3 Pemasangan PHP dan Modul PHP

Program PHP yang digunakan untuk menjalankan Aplikasi MANTRA adalah PHP Versi 5. Berikut perintah instalasi PHP pada Ubuntu:

```
$ sudo apt-get install php5 libapache2-mod-php5
```

Setelah proses instalasi selesai, lanjutkan dengan instalasi modul-modul PHP yang dibutuhkan.

```
$ sudo apt-get install php5-curl php5-tidy php5-gd
$ sudo apt-get install php5-mysql php5-pgsql php5-sybase
$ sudo service apache2 restart
```

Untuk mendukung konektivitas Aplikasi MANTRA dengan Basis Data Oracle dibutuhkan modul oci8 sebagai driver untuk melakukan koneksi ke dalam Basis Data Oracle. Instalasi modul oci8 dilakukan secara manual karena modul tersebut tidak tersedia dalam repositori. Adapaun petunjuk instalasi driver oci8 dijelaskan di buku ini pada subbab 3.3.4.

Dalam rangka mempermudah pengelolaan Basis Data MySQL, terdapat aplikasi berbasis web yang menggunakan platform PHP untuk membantu pengelolaan basis data. Pemasangan aplikasi tersebut dapat dilakukan menggunakan perintah berikut:

```
$ sudo apt-get install phpmyadmin
```

Pada saat instalasi Anda diminta mengisi data berikut:

Web server to reconfigure automatically: **<--apache2**

Configure database for phpmyadmin with dbconfig-common? **←Yes**

Setelah proses instalasi Aplikasi phpMyAdmin dapat diakses dari browser dengan alamat http://alamat-ip-server/phpmyadmin.

3.3.4 Pemasangan Driver Basis Data Oracle

Untuk melakukan koneksi ke Basis Data Oracle dibutuhkan Oracle Instant Client dan Modul PHP OCI8 yang harus dipasang pada server Aplikasi MANTRA. Berkas Oracle Instant Client Basic dan Oracle Instant Client SDK dapat diunduh pada alamat berikut http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html?ssSourceSiteId=ocomen.

Berikut adalah langkah-langkah instalasi oracle instant client dan modul oci8 pada server ubuntu:

1. Unduh berkas instant client basic dan sdk, salin 2 (dua) berkas tersebut ke dalam root direktori pengguna server contoh: /home/username.
2. Pasang modul PECL, PHP Development, Build Essential, Unzip, dan AIO Library untuk mendukung kompilasi modul oci8 secara manual.

```
$ sudo su atau $su
# apt-get install php-pear php5-dev build-essential unzip libaio1
```

3. Buat direktori untuk keperluan instalasi oracle instant client.

```
# mkdir /opt/oracle
```

4. Pindahkan 2 (dua) berkas instant client (Basic dan SDK) ke direktori yang dibuat pada langkah ke 3 (tiga).

```
# mv /home/username/instantclient-* /opt/oracle
```

5. Masuk ke dalam direktori oracle (/opt/oracle), unzip berkas instant client, dan rename direktori instant client.

```
# cd /opt/oracle
# unzip instantclient-basic*.zip
# unzip instantclient-sdk*.zip
# mv instantclient_11_2 instantclient
```

6. Masuk ke dalam direktori instant client dan buat soft links sebagai berikut:

```
# cd /opt/oracle/instantclient
# ln –s libclntsh.so.11.2 libclntsh.so
# ln –s libocci.so.11.2 libocci.so
```

7. Atur input path direktori instant client pada konfigurasi library dynamic.

```
# echo /opt/oracle/instantclient > /etc/ld.so.conf.d/oracle-
instantclient
```

8. Update library dynamic configuration.

```
# ldconfig
```

9. Pasang modul oci8 untuk php5 menggunakan PECL.

```
# pecl install oci8-2.0.10
```

10. Pada saat proses instalasi berjalan, Anda diminta menentukan path lokasi instant client, isi sesuai keterangan berikut:

```
instantclient,/opt/oracle/instantclient
```

11. Tambah konfigurasi oci8 ke dalam konfigurasi php dan restart apache.

```
# echo extension=oci8.so > /etc/php5/mods-available/oci8.ini
# php5enmod oci8 && service apache2 restart
```

12. Pastikan modul oci8 sudah telah diaktifkan dengan mengakses phpinfo.php dari browser. Buat file phpinfo.php dalam direktori /var/www/html dengan perintah sebagi berikut:

```
# echo "<?php phpinfo(); ?>" > phpinfo.php
# exit
```

3.3.5 Pemasangan Aplikasi MANTRA

Petunjuk pemasangan Aplikasi MANTRA yang dijelaskan dalam buku ini adalah petunjuk pemasangan Aplikasi MANTRA Versi 1.99 Update M menggunakan konfigurasi alias. Adapun langkah-langkah pemasangan aplikasi dijelaskan sebagai berikut:

1. Salin berkas mantra.zip ke dalam home direktori pengguna di server contoh: /home/username. Penyalinan berkas ke dalam server dapat dilakukan secara remote menggunakan aplikasi FileZila, WinSCP, atau dengan perintah SCP:

```
$ scp username@192.168.1.100:/home/username/mantraXXX.zip .
```

2. Login ke server, kemudian unzip berkas mantra.zip menggunakan perintah berikut:

```
$ unzip mantraXXX.zip
```

3. Ubah owner direktori files dan tmp dalam direktori mantra (/home/username/mantra):

```
$ sudo chown –R www-data:www-data /home/username/mantra/files
$ sudo chown –R www-data:www-data /home/username/mantra/tmp
```

4. Tambahkan konfigurasi alias pada Web Server Apache untuk akses aplikasi MANTRA dengan cara sebagai berikut:

```
$ sudo nano /etc/apache2/conf-available/mantra.conf
   Alias /mantra /home/username/mantra/
   <Directory /home/username/mantra/>
        Options FollowSymLinks
        AllowOverride all
        Require all granted
   </Directory >

$ sudo a2enconf mantra

atau

$ sudo nano /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
   DocumentRoot /home/username/mantra/
   <Directory /home/username/mantra/>
        Options FollowSymLinks
        AllowOverride all
        Require all granted
   </Directory >
</VirtualHost>

atau:
<VirtualHost *:80>
   Alias /mantra /home/username/mantra/
   <Directory /home/username/mantra/>
        Options FollowSymLinks
        AllowOverride all
        Require all granted
   </Directory >
</VirtualHost>
```

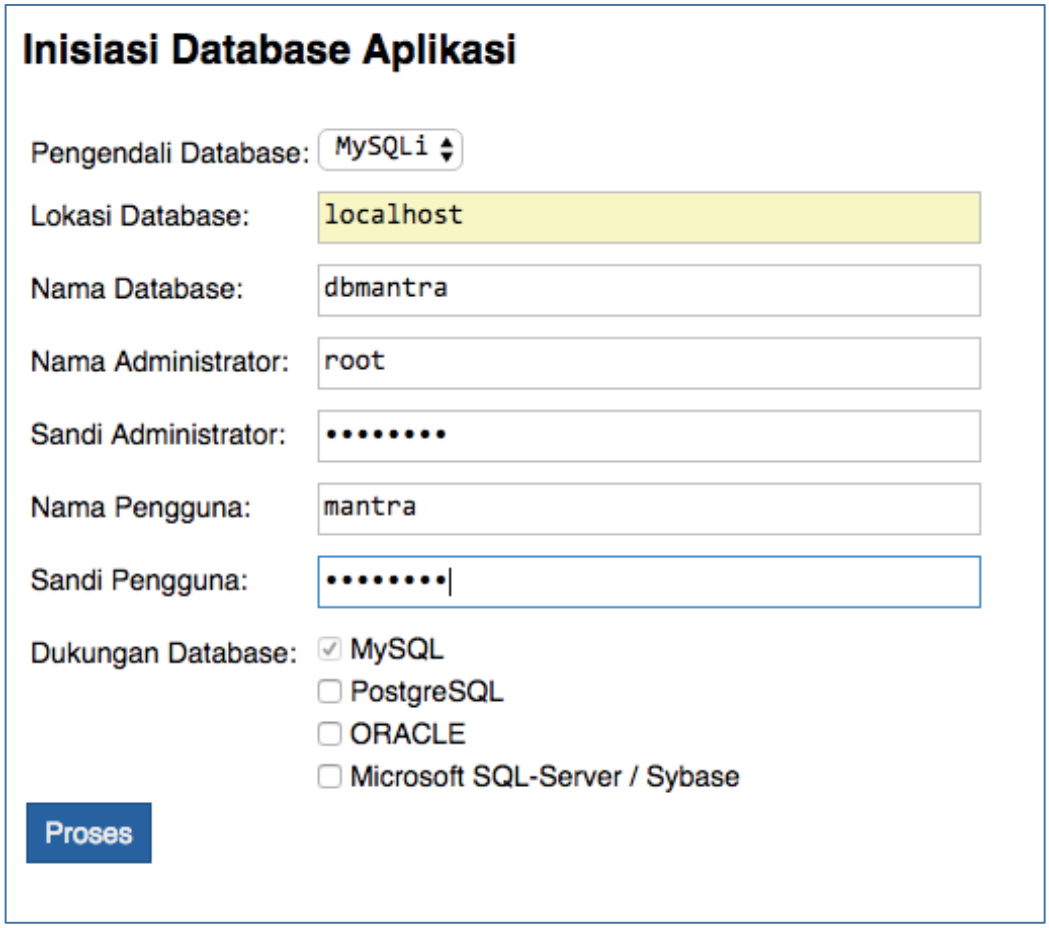
5. Restart Apache untuk mengaktifkan konfigurasi alias yang telah dibuat.

```
$ sudo service apache2 restart
```

6. Sesuaikan konfigurasi RewriteBase pada berkas .htaccess dalam direktori /home/username/mantra sesuai dengan nama alias yang telah dibuat.

```
<IfModule mod_rewrite.c>
    RewriteEngine On

    jika didefinisikan sebagai DocumentRoot:
    RewriteBase /

    jika didefinisikan sebagai Alias:
    RewriteBase /mantra

    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule ^ index.php [L]
</IfModule>
```

7. Akses MANTRA melalui browser dengan alamat http://alamat-ip-server/mantra. Administrator harus mengisi informasi pada kolom-kolom yang ditampilkan sebelum menggunakan Aplikasi MANTRA kemudian menekan tombol Proses.

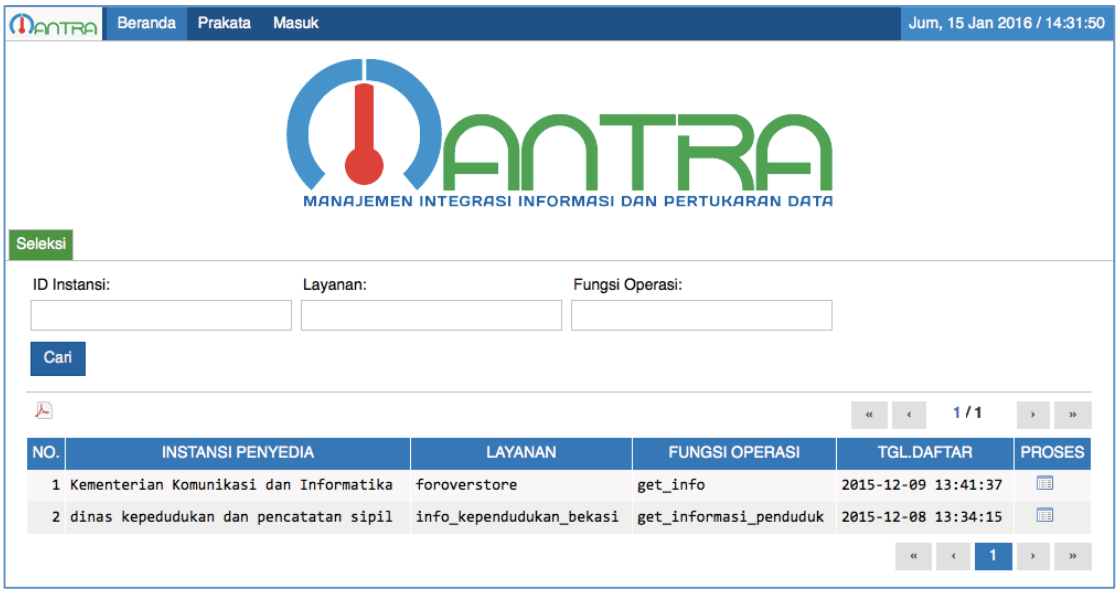


Gambar 3.29 Halaman Inisiasi Database

Pada gambar 3.29 di atas, kolom Nama Administrator menunjukkan nama pengguna dalam Basis Data MySQL yang memiliki hak akses sebagai administrator. Sedangkan kolom Nama Pengguna menunjukkan nama pengguna dalam Basis Data MySQL yang akan melakukan operasi select, insert, update, dan delete pada basis data yang ditulis pada kolom Nama Database. Jika instalasi Aplikasi MANTRA menggunakan basis data yang sudah digunakan sebelumnya, administrator disarankan melakukan backup terlebih dahulu terhadap basis data tersebut.

8. Setelah menekan tombol proses, pembuatan file konfigurasi dan impor basis data akan dilakukan secara otomatis. Jika tidak mengalami kegagalan, administrator dapat melakukan login ke

dalam Aplikasi MANTRA melalui menu "Masuk" pada aplikasi MANTRA menggunakan ID Pengguna "admin" dan Kata Kunci "1234".



Gambar 3.30 Halaman Awal Aplikasi MANTRA

3.3.6 Pemasangan Modul Apache Multi-Proses

Apache dapat dikonfigurasikan untuk berjalan dalam mode *pre-forked* atau mode *multi-process*. Mode standar yang diaktifkan pada saat instalasi Web Server Apache adalah *pre-forked*. Untuk meningkatkan performansi web server dalam menanggapi permintaan akses oleh pengguna salah satu mekanisme yang direkomendasikan adalah menggunakan mode *multi-process*. Berikut ini adalah petunjuk konfigurasi Web Server Apache dalam mode multi-process pada Sistem Operasi Ubuntu Server 14.04:

1. Sebelum memulai pemasangan, pastikan konfigurasi repositori Ubuntu pada server sudah mencantumkan repositori multiverse. Konfigurasi tersebut dapat dilihat dalam berkas /etc/apt/sources.list.

```
deb http://us.archive.ubuntu.com/ubuntu/ trusty main multiverse universe restricted

deb-src http://us.archive.ubuntu.com/ubuntu/ trusty main multiverse universe restricted

deb http://us.archive.ubuntu.com/ubuntu/ trusty-updates main multiverse universe restricted

deb-src http://us.archive.ubuntu.com/ubuntu/ trusty-updates main

multiverse universe restricted
```

2. Pemasangan modul pendukung Apache Multi-Proses (Apache MPM Event).

```
$ sudo apt-get update

$ sudo apt-get upgrade

$ sudo apt-get install apache2-mpm-event apache2-utils

$ sudo apt-get install libapahe2-mod-fastcgi php5-fpm php5-cli
```

3. Non-aktifkan modul prefork dan php pada Web Server Apache.

```
$ sudo a2dismod php5 mpm_prefork
```

4. Aktifkan modul pendukung multi-process.

```
$ sudo a2enmod actions fastcgi alias mpm_event
```

5. Buat berkas PHP fastCGI dan atur mode akses berkas tersebut.

```
$ sudo touch /usr/lib/cgi-bin/php5.fcgi

$ sudo chown www-data:www-data /usr/lib/cgi-bin/php5.fcgi
```

6. Buat berkas konfigurasi php-fpm pada Web Server Apache.

```
$ sudo nano /etc/apache2/conf-available/php5-fpm.conf

<IfModule mod_fastcgi.c>
   AddHandler php5.fcgi .php
   Action php5.fcgi /php5.fcgi
   Alias /php5.fcgi /usr/lib/cgi-bin/php5.fcgi
   FastCgiExternalServer /usr/lib/cgi-bin/php5.fcgi -socket /var/run/php5-
   fpm.sock -pass-header Authorization -idle-timeout 3600
   <Directory /usr/lib/cgi-bin>
      Require all granted
    </Directory>
</IfModule>
```

7. Aktifkan konfigurasi php-fpm pada Web Server Apache dan mcrypt pada PHP.

```
$ sudo a2enconf php5-fpm
$ sudo php5enmod mcrypt
```

8. Restart apache dan php-fpm untuk mengaktifkan konfigurasi yang telah dibuat.

```
$ sudo service apache2 restart
$ sudo service php5-fpm restart
```

9. Pastikan mode apache multi-process sudah aktif dengan perintah berikut:

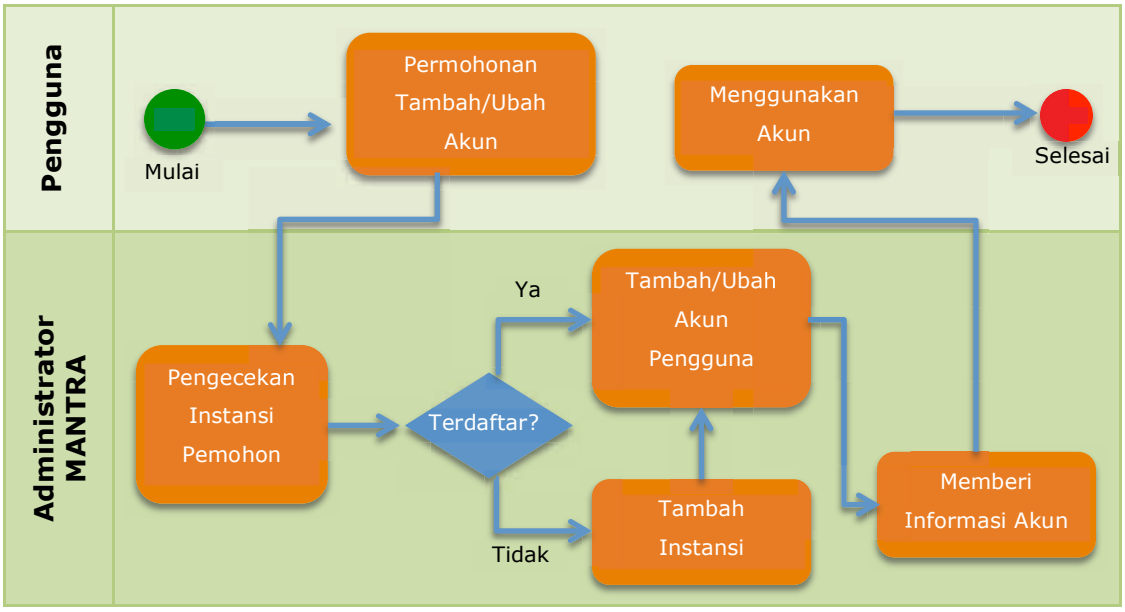
```
$ sudo apache2ctl –V
```

   Jika mode multi-process sudah aktif, perintah di atas menampilkan keterangan "Server MPM: Event".

3.4 Panduan Penggunaan

Pengelolaan Layanan Berbagi Pakai Data/Informasi dengan aplikasi MANTRA disediakan dalam instrumen modul GUI berbasis Web untuk mempermudah pengaturan akses layanan. Hal ini berbeda dengan layanannya yang hanya menyediakan data/informasi hasil olahan saja. Penggunaan aplikasi MANTRA dikelompokan berdasarkan peran yaitu Administrator, Supervisor, Requester, Provider dan Publisher. Masing-masing pengguna memiliki peran untuk mengelola layanan sesuai fungsinya sebagaimana akan dijelaskan pada subbab selanjutnya.

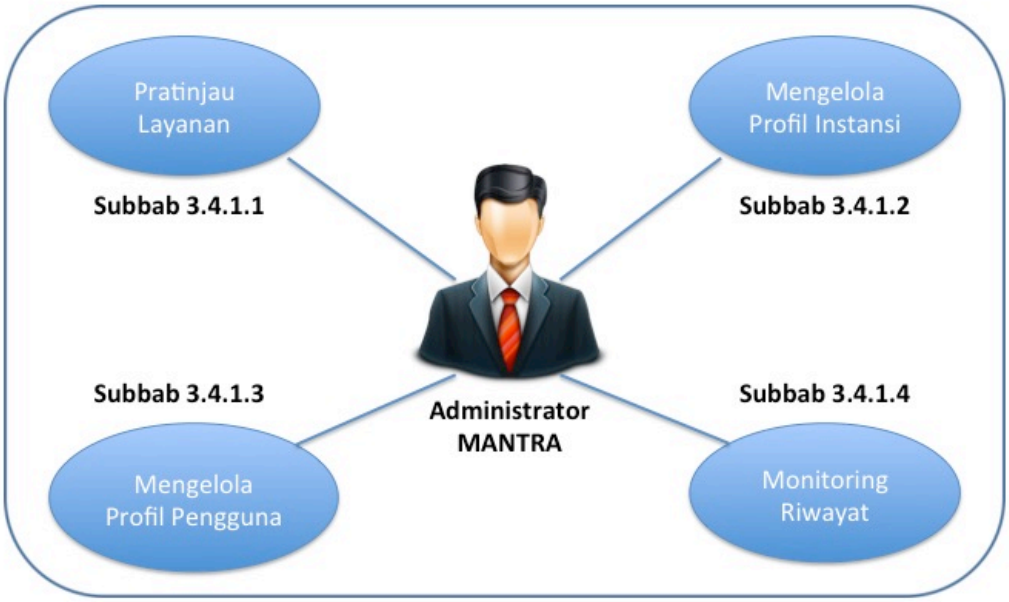
Peran untuk mendaftarkan pengguna Aplikasi MANTRA dijalankan oleh Administrator. Setiap pengguna harus mengajukan permohonan pembuatan atau pengubahan akun kepada Administrator MANTRA. Gambar 3.31 di bawah ini menunjukkan alur kerja penambahan/pengubahan akun pengguna.



Gambar 3.31 Diagram Alur Pembuatan Akun Pengguna

3.4.1 Panduan Administrator

Secara umum fungsi yang dilaksanakan oleh **Administrator** adalah melakukan pengelolaan data pengguna dan pengawasan aplikasi. Adapun penjabaran fungsi tersebut dapat dilihat dalam gambar di bawah ini.

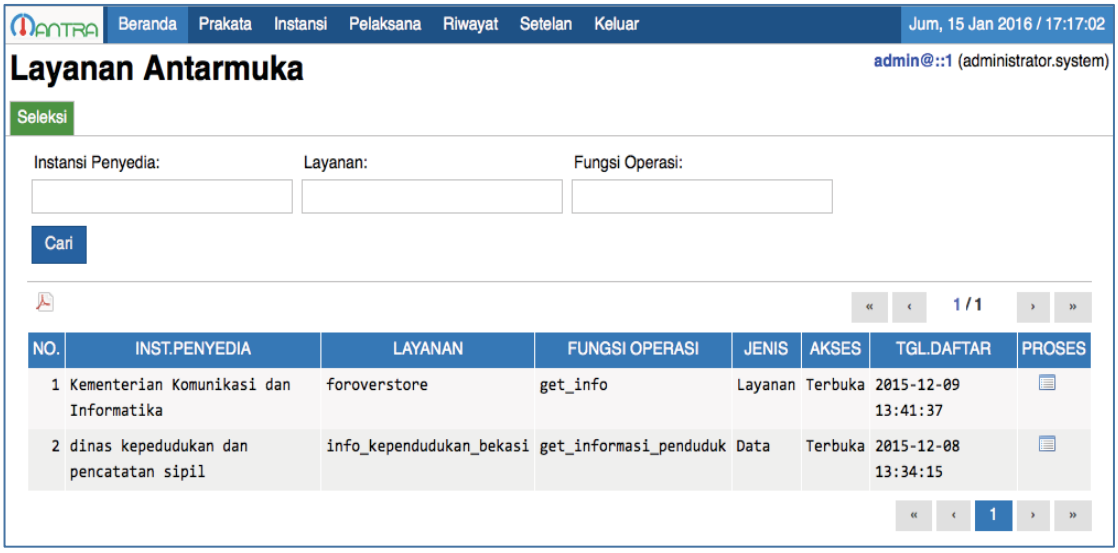


Gambar 3.32 Use Case Administrator MANTRA

Sebagaimana ditampilkan pada Gambar 3.32 di atas, administrator memiliki 4 (empat) fungsi utama. Penjelasan fungsi-fungsi tersebut dapat dipelajari dalam subbab yang ditunjuk pada gambar. Di antara empat fungsi tersebut, fungsi yang memiliki keterkaitan adalah fungsi pengelolaan profil pengguna dan fungsi pengelolaan profil instansi. Keterkaitan yang dimaksud berupa keterkaitan domain. Setiap akun pengguna harus didaftarkan pada instansi induknya (domain) dengan catatan instansi tersebut telah terdaftar dalam MANTRA. Oleh karena itu sebelum seseorang pengguna didaftarkan dalam MANTRA, instansi yang menjadi induk dari pengguna tersebut harus dipastikan telah terdaftar dalam MANTRA.

3.4.1.1 Pratinjau Layanan

Melalui fungsi pratinjau layanan seorang administrator dapat melihat seluruh layanan yang terdaftar dalam MANTRA. Selain itu dengan menggunaan fungsi ini, seorang administrator dapat melakukan uji fungsi layanan untuk mengetahui apakah status suatu layanan berjalan dengan baik atau terkendala.



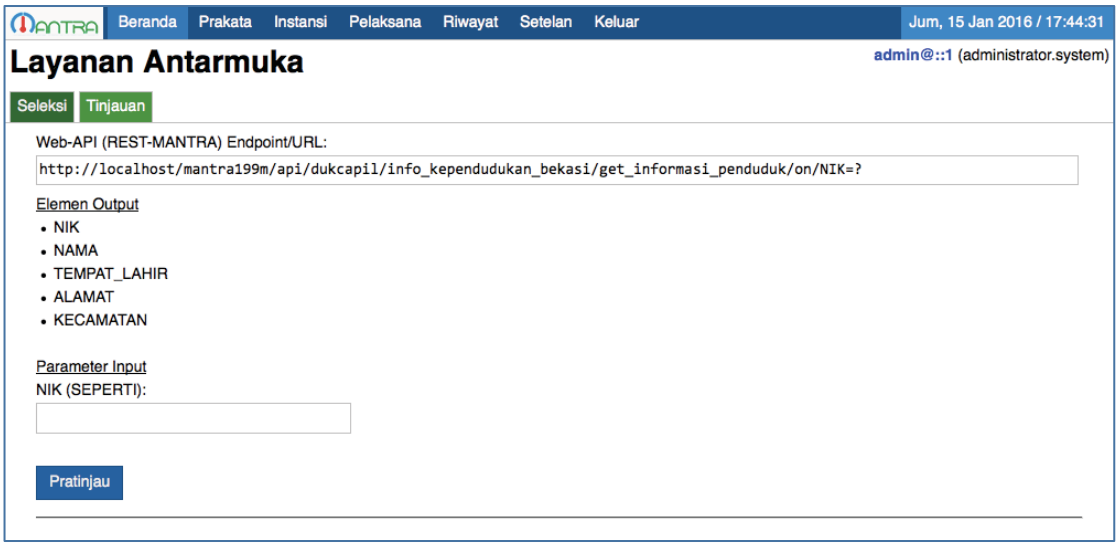
Gambar 3.33 Halaman Pratinjau Layanan

Fungsi Pratinjau Layanan dapat diakses melalui menu "**Beranda**". Pada halaman tersebut akan ditampilkan tabel yang berisi nama-nama layanan yang sudah terdaftar dari masing-masing instansi. Melalui ikon "Tinjauan" pada kolom "**PROSES**", administrator dapat melakukan uji fungsi layanan untuk mengetahui apakah layanan berjalan dengan baik atau terkendala. Sebagaimana ditunjukkan pada Gambar 3.33, dalam kolom "**JENIS**" terdapat 3 (tiga) macam keterangan yaitu Layanan, Data, dan Program.

Keterangan jenis layanan pada Halaman Beranda adalah sebagai berikut:

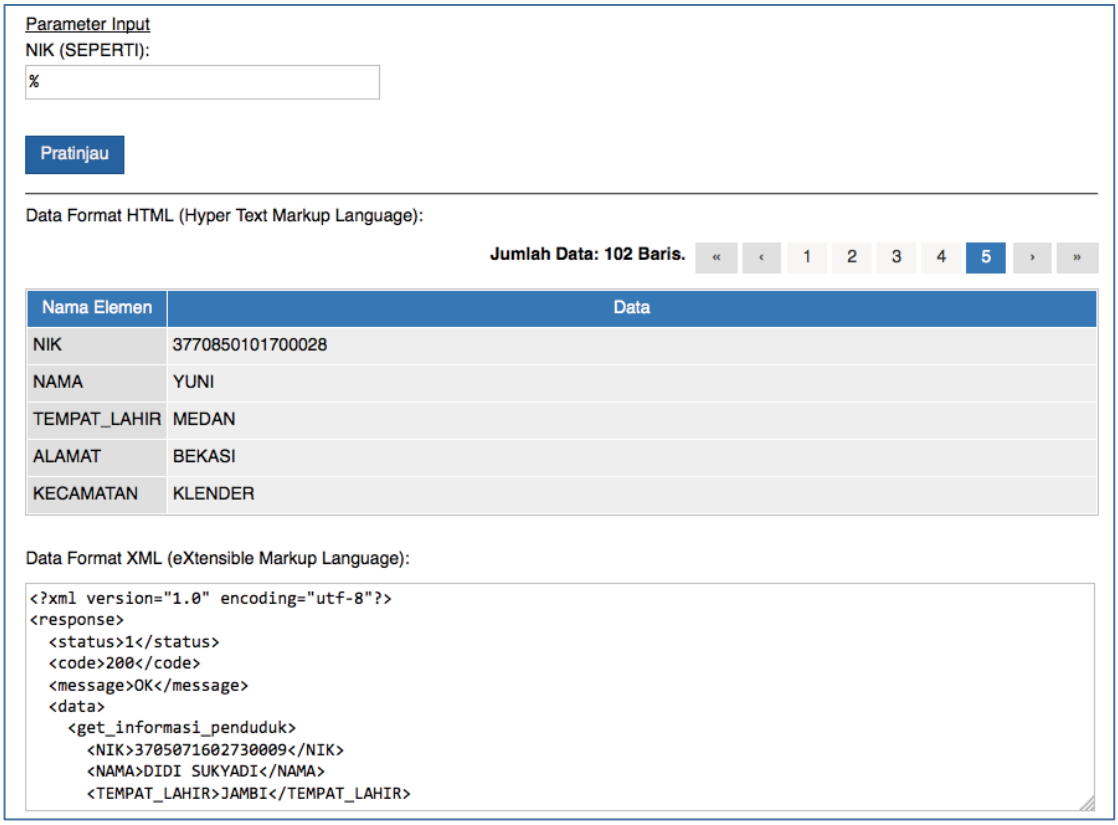
1. **Layanan**, keterangan tersebut menunjukkan layanan yang dimaksud adalah *proxy services* (penjelasan dapat dilihat pada subbab 2.4) yang dibuat oleh pengguna *publisher*.
2. **Data**, keterangan tersebut menunjukkan kategori layanan *data services* (penjelasan dapat dilihat pada subbab 2.4) yang dibuat oleh pengguna *provider*.

3. **Program**, keterangan tersebut menunjukkan kategori layanan terkustomisasi yang dibuat secara manual oleh pengguna *provider* menggunakan kode program PHP.

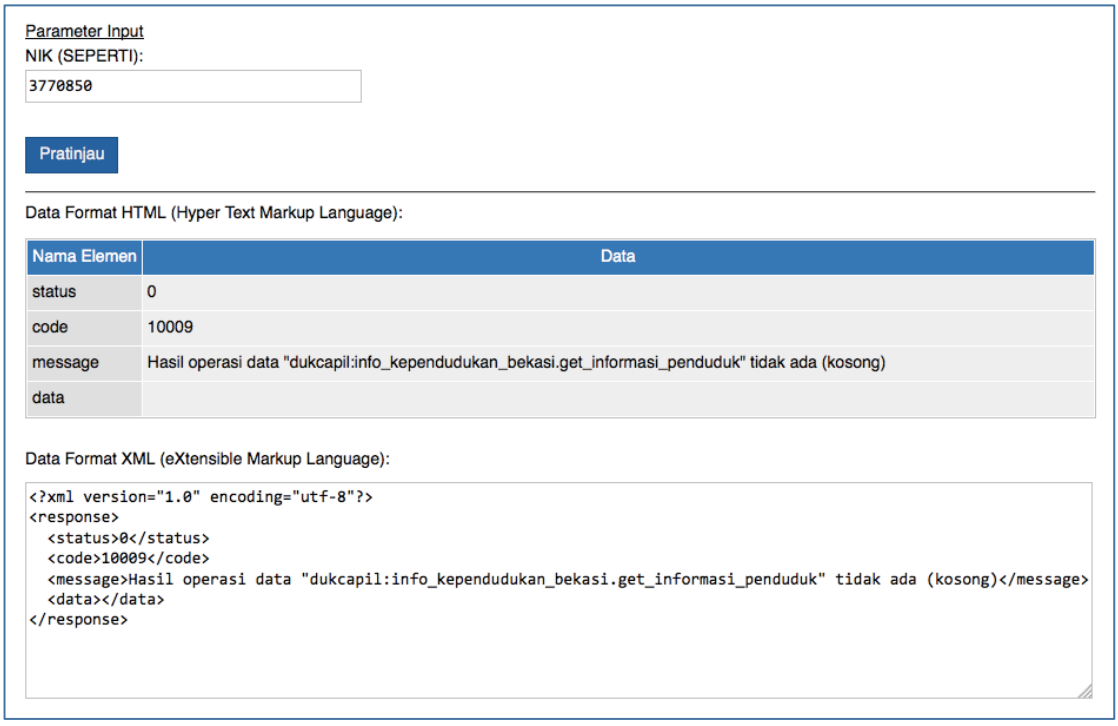


Gambar 3.34 Halaman Tinjauan Layanan

Gambar 3.34 di atas merupakan tampilan fitur tinjauan layanan MANTRA. Pada halaman tersebut, administrator harus mengisi parameter layanan (jika ada) sebelum melakukan eksekusi "Pratinjau". Pada contoh gambar tersebut parameter yang harus diisi adalah "NIK". Keterangan "SEPERTI" pada gambar di atas menunjukkan operasi perbandingan parameter yang dilakukan ketika layanan dijalankan. "Elemen Output" menunjukkan elemen-elemen data yang disediakan oleh layanan pertukaran data. Gambar 3.35 merupakan tampilan tinjauan layanan yang berjalan dengan baik, sedangkan Gambar 3.36 merupakan tampilan tinjauan layanan yang mengalami kendala akses. Secara lengkap kode output layanan MANTRA dapat dilihat dalam Lampiran 1 dan Lampiran 2.



Gambar 3.35 Eksekusi Layanan Berhasil

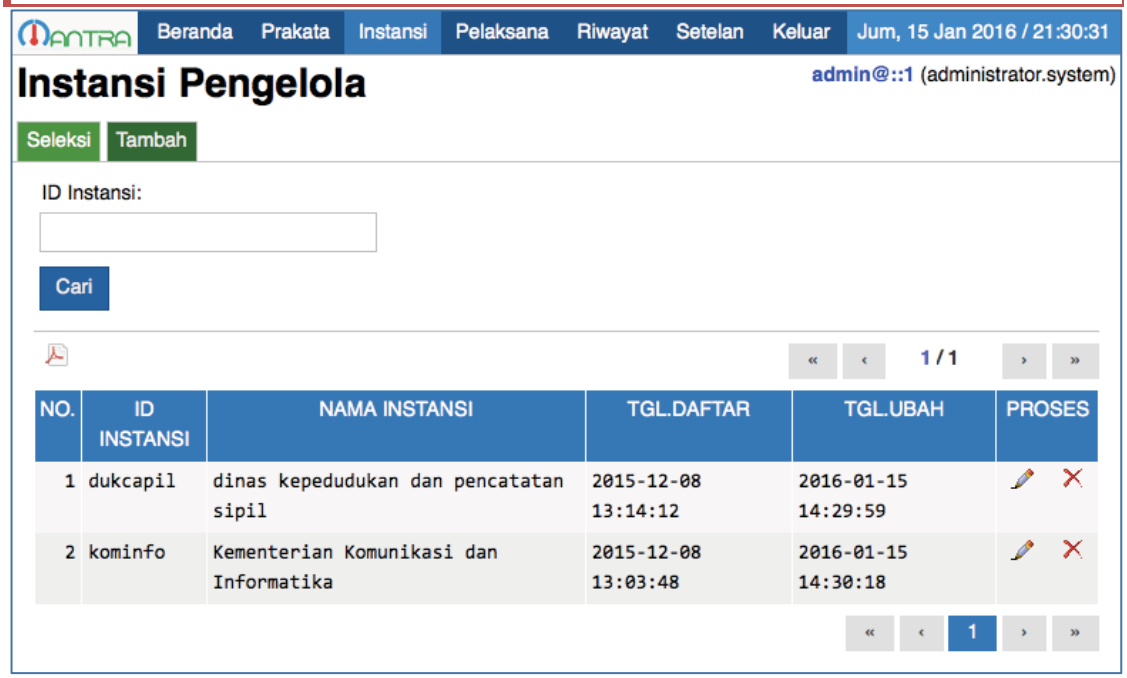


Gambar 3.36 Eksekusi Layanan Gagal

3.4.1.2 Pengelolaan Profil Instansi

Modul pengelolaan profil instansi dapat diakses melalui menu "**Instansi**". Administrator dapat mendaftarkan instansi baru dengan mengakses sub menu "**Tambah**" pada halaman instansi. Selain itu administrator juga dapat mengubah data instansi yang telah terdaftar dengan menekan ikon atau menekan ikon untuk menghapus data instansi yang terdaftar.

Penghapusan profil instansi secara otomatis menghapus semua profil pengguna dan semua layanan yang terdaftar pada instansi tersebut.

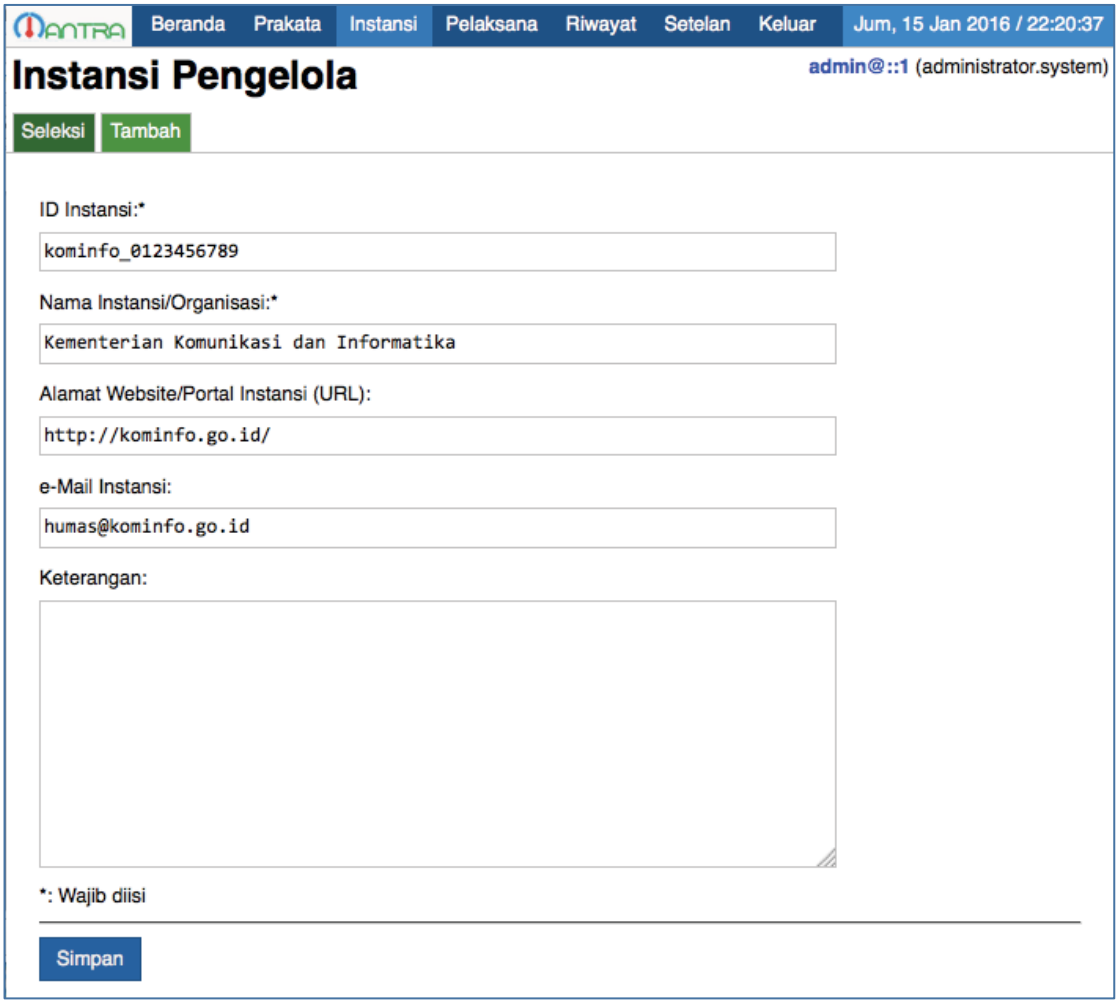


Gambar 3.37 Halaman Daftar Instansi

3.4.1.2.1 Pendaftaran Instansi Baru

Untuk keperluan pendaftaran instansi baru, administrator harus mengakses submenu "**Tambah"** kemudian melengkapi beberapa informasi sesuai dengan formulir yang disediakan. Informasi yang harus dilengkapi antara lain:

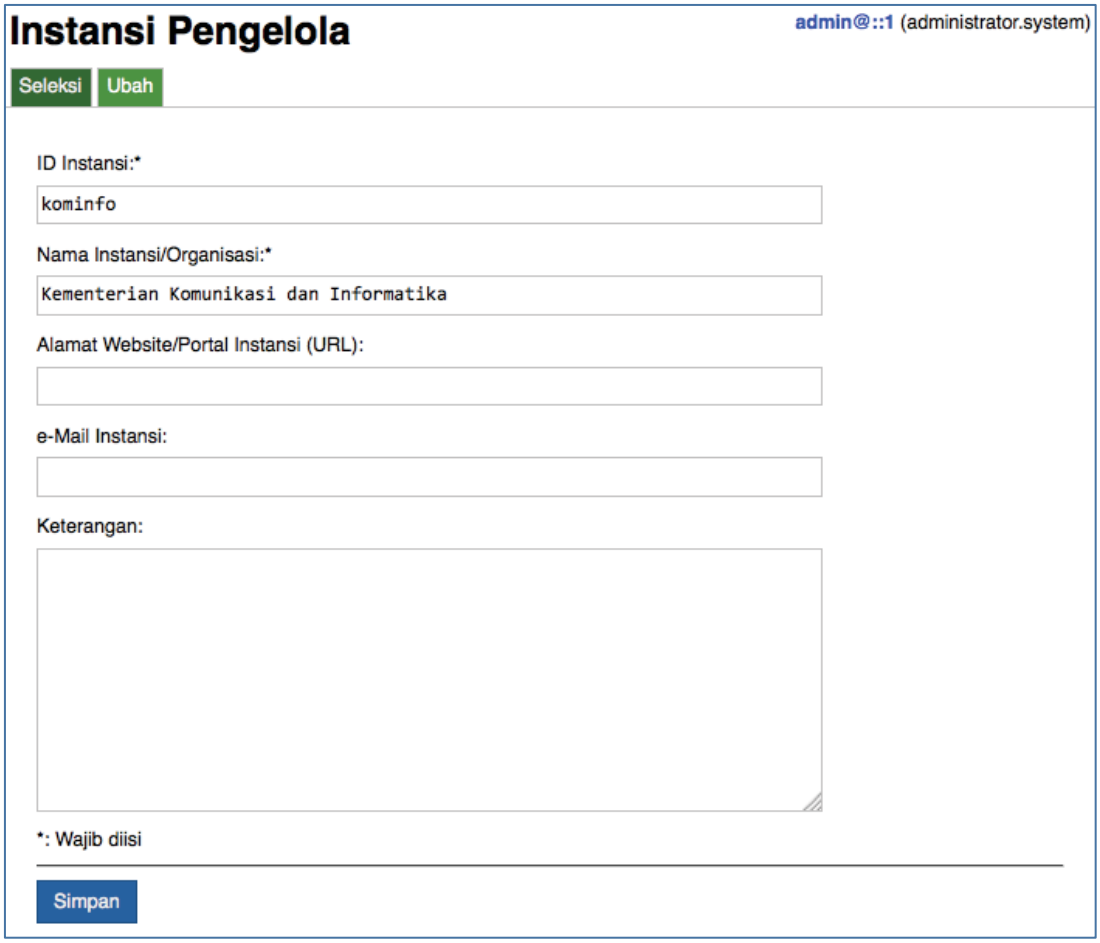
1. **ID Instansi** merupakan kunci perekaman profil instansi (harus diisi). Pengisian kolom ID Instansi hanya dibatasi dengan kombinasi huruf kecil (a-z), angka (0-9), dan karakter "_".
2. **Nama Instansi/Organisasi** adalah nama lengkap instansi/organisasi yang didaftarkan (harus diisi).
3. **Alamat Website/Portal Instansi** adalah alamat url website instansi (optional), contoh: http://kominfo.go.id.
4. **e-Mail Instansi** adalah alamat e-mail utama instansi (optional).
5. **Keterangan** perihal instansi yang didaftarkan (optional).



Gambar 3.38 Halaman Tambah Instansi Baru

3.4.1.2.2 Update Profil Instansi

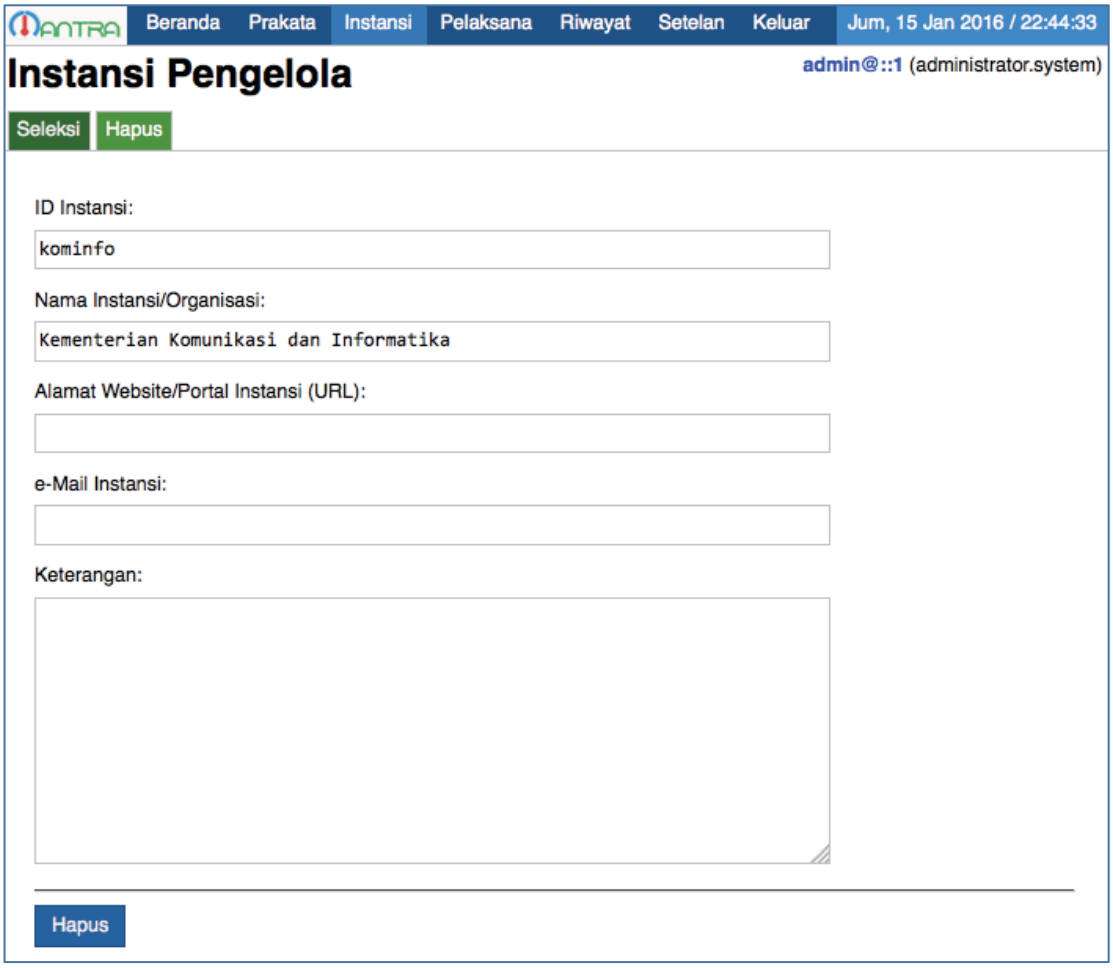
Untuk perbaruan profil instansi, administrator harus menekan ikon yang terletak dalam kolom **"PROSES"** pada halaman **Instansi**. Setelah melakukan perubahan, penyimpanan data dilakukan dengan menekan tombol Simpan **.** Jika data tersebut berhasil disimpan, pada halaman instansi ditampilkan pesan "perubahan data berhasil disimpan".



Gambar 3.39 Halaman Ubah Data Instansi

3.4.1.2.3 Penghapusan Profil Instansi

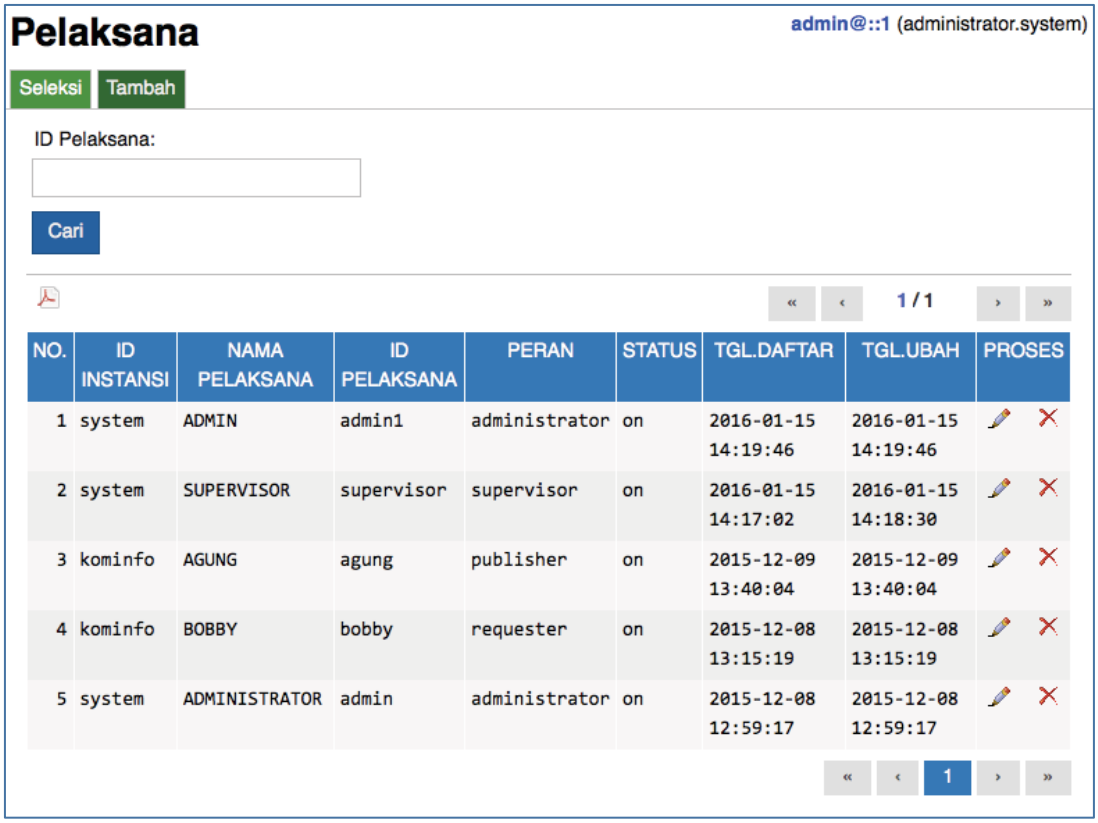
Untuk menghapus profil instansi, Administrator dapat menekan ikon dalam kolom **"PROSES"** pada halaman **Instansi**. Melalui ikon tersebut ditampilkan halaman hapus data sebagaimana ditunjukkan pada Gambar 3.40. Tombol Hapus pada halaman tersebut digunakan untuk melakukan konfirmasi penghapusan data. Jika data berhasil dihapus, pada halaman instansi ditampilkan pesan "Data berhasil dihapus".



Gambar 3.40 Halaman Hapus Data Instansi

3.4.1.3 Pengelolaan Profil Pengguna

Modul pengelolaan profil pengguna dapat diakses melalui menu "**Pelaksana**". Pada halaman tersebut ditampilkan nama-nama pengguna yang telah terdaftar dalam Aplikasi MANTRA. Sebagai tambahan, pada halaman tersebut juga tersedia fitur untuk mendaftarkan, mengubah dan menghapus data pengguna. Gambar 3.41 merupakan tampilan halaman pelaksana bagi administrator.



Gambar 3.41 Halaman Pelaksana Administrator

3.4.1.3.1 Menambah Akun Pengguna

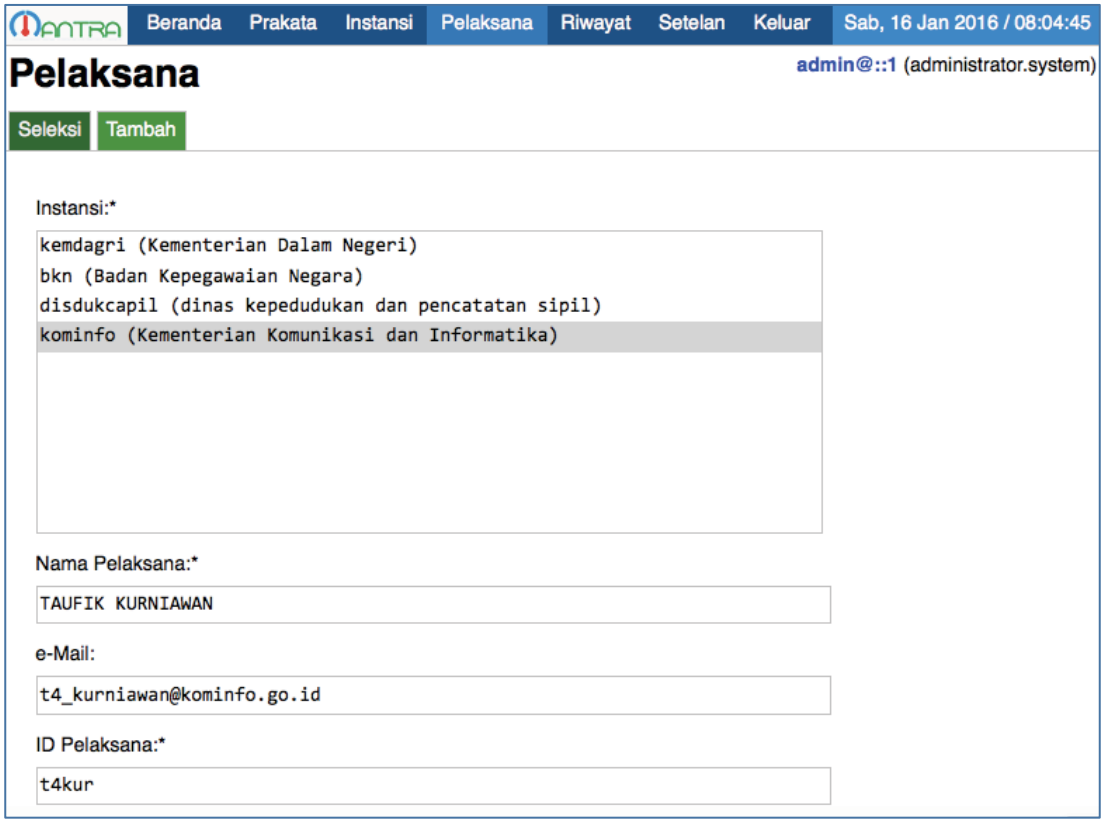
Penambahan pengguna baru dapat dilakukan dengan mengakses submenu **Tambah** pada halaman pelaksana. Administrator kemudian diarahkan menuju halaman penambahan pengguna untuk melengkapi data sesuai dengan kolom-kolom yang telah disediakan, antara lain:

1. **Instansi**, pada kolom tersebut ditampilkan daftar instansi yang telah terdaftar. Administrator harus memilih salah satu instansi yang menjadi induk dari pengguna yang didaftarkan.
2. **Nama Pelaksana**, berisi nama lengkap pengguna, secara otomatis ditulis dalam format huruf kapital oleh sistem (harus diisi).

3. **e-Mail**, kolom tersebut diisi dengan alamat e-mail pribadi yang resmi dari instansi. Untuk pengguna *provider*, *publisher,* dan *requester* e-mail tersebut digunakan oleh sistem untuk pengiriman notifikasi.
4. **ID Pelaksana**, kode unik pengguna yang berfungsi sebagai username ketika melakukan login (harus diisi). Nilainya merupakan kombinasi huruf kecil (a-z), angka (0-9), simbol (“.” “_” “-“).
5. **Kata Kunci (Password),** kode *password* merupakan kombinasi huruf (A-Z, a-z), angka, dan simbol dalam *keyboard* (harus diisi).
6. **Peran**, administrator harus memilih salah satu peran sesuai dengan permintaan pengguna yang didaftarkan.
7. **Status,** akun dengan status **on** dapat digunakan untuk login pada Aplikasi MANTRA. Sebaliknya jika berstatus **off**, akun tersebut tidak dapat digunakan untuk login.

Setelah memastikan kolom-kolom tersebut terisi dengan lengkap, data tersebut kemudian disimpan dengan menekan tombol Simpan. Gambar 3.42 merupakan tampilan halaman penambahan pengguna.

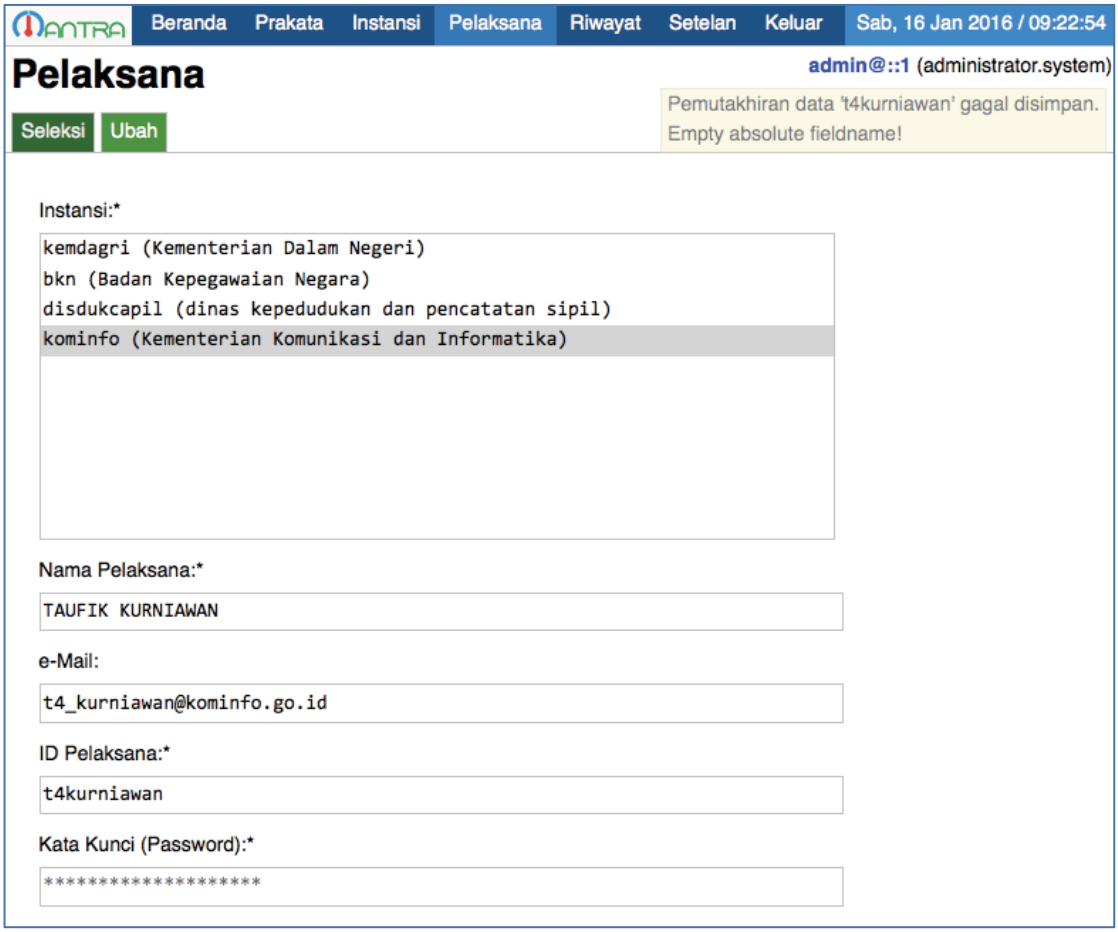
Setiap aktivitas yang dilakukan administrator terekam oleh sistem dan ditampilkan pada halaman Riwayat. Penjelasan dapat dilihat dalam subbab 3.4.1.4



Gambar 3.42 Halaman Tambah Pengguna

3.4.1.3.2 Mengubah Akun Pengguna

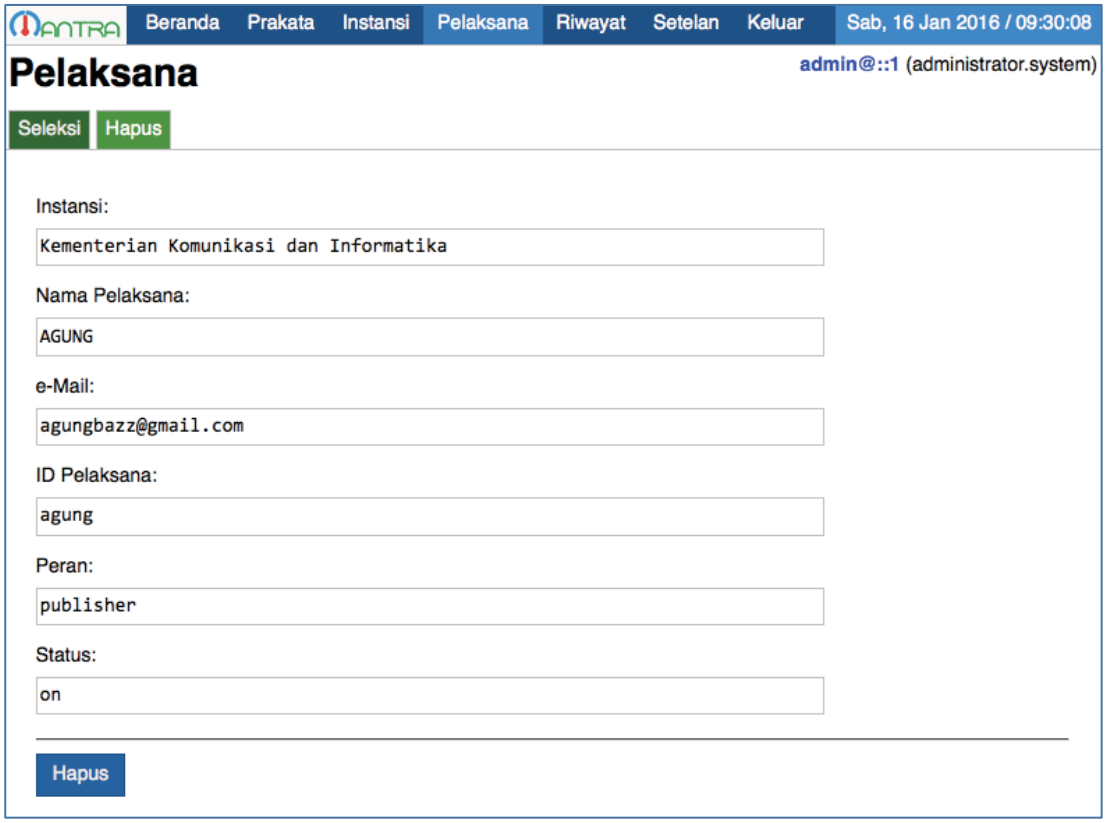
Ikon dalam kolom **PROSES** pada halaman pelaksana merupakan tombol navigasi menuju halaman ubah profil pengguna. Pada halaman tersebut ditampilkan data profil pengguna yang tersimpan. Administrator dapat mengganti data yang tercantum dalam semua kolom yang ditampilkan. Jika perubahan data berhasil disimpan, administrator diarahkan kembali menuju halaman pelaksana dan ditampilkan notifikasi di bagian atas sebelah kanan halaman. Sebaliknya jika perubahan gagal disimpan maka ditampilkan notifikasi gagal di bagian atas sebelah kanan halaman ubah profil pengguna. Notifikasi "empty absolute fieldname" memiliki pengertian terdapat satu atau beberapa data dari kolom yang sifatnya mandatori yang nilainya kosong.



Gambar 3.43 Notifikasi Gagal Proses Ubah Profil Pengguna

3.4.1.3.3 Menghapus Profil Pengguna

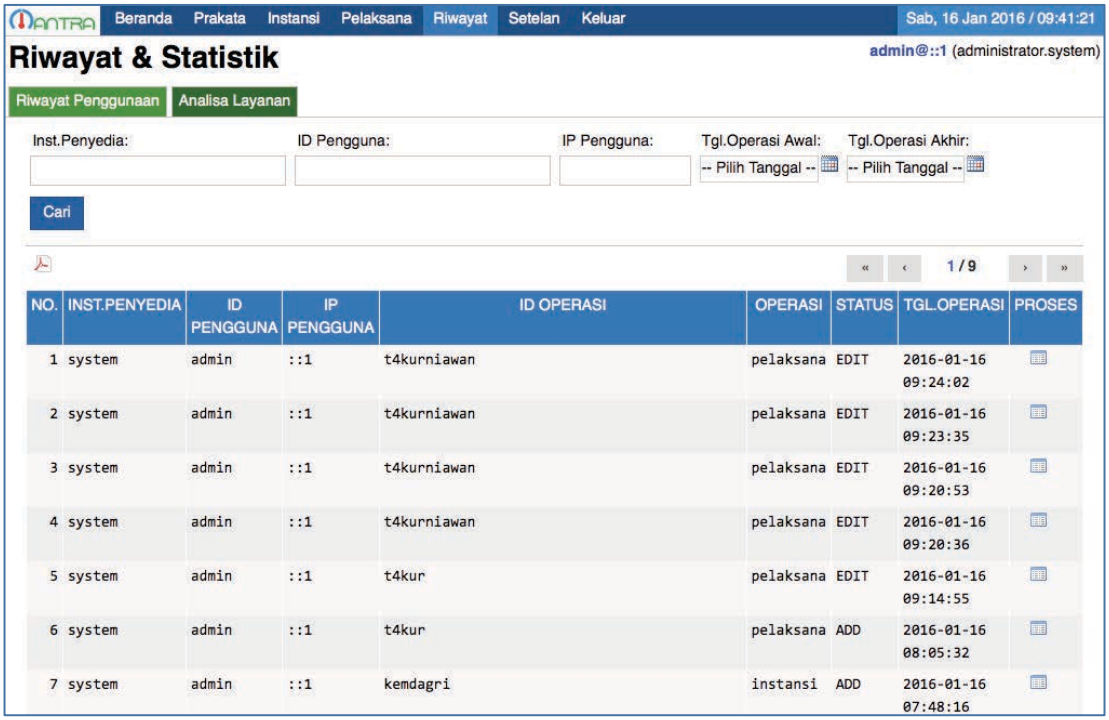
Penghapusan profil pengguna dilakukan dengan menekan ikon dalam kolom **PROSES** pada halaman pelaksana. Aplikasi akan menampilkan detail informasi pengguna dalam halaman hapus data. Tombol **Hapus** di bagian bawah halaman digunakan untuk melakukan konfirmasi penghapusan data secara.



Gambar 3.44 Halaman Hapus Profil Pengguna

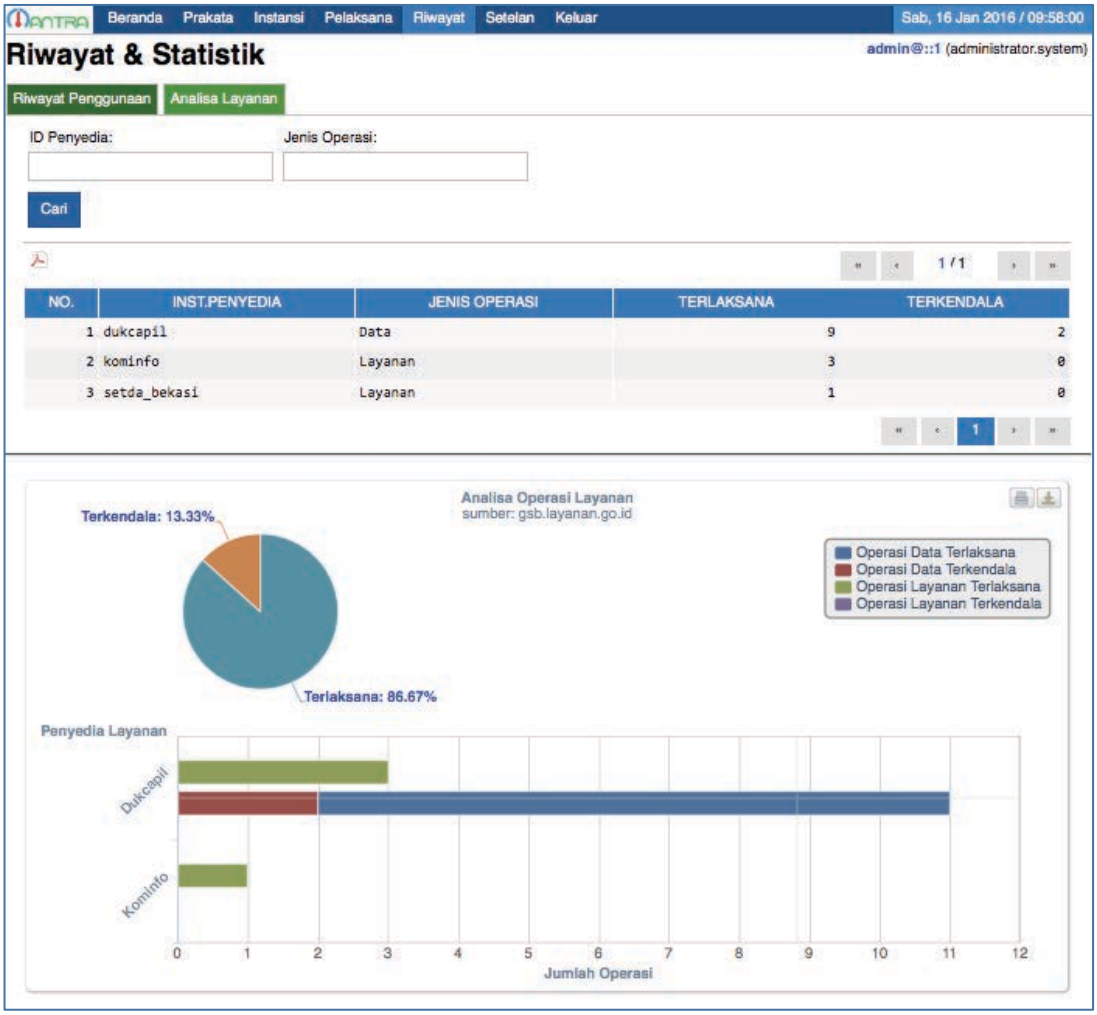
3.4.1.4 Tinjauan Riwayat Operasional

Modul Riwayat Operasional digunakan untuk melakukan monitoring aktivitas pengguna dan aktivitas pemanfaatan layanan. Adapun rekaman-rekaman aktivitas tersebut dapat dilihat melalui menu navigasi "**Riwayat**". Ikon dalam kolom **PROSES** digunakan untuk melihat informasi detail dari salah satu riwayat aktivitas yang ditampilkan. Aktivitas yang direkam adalah seluruh aktivitas pengguna selama menggunakan aplikasi dan seluruh aktivitas pemanfaatan layanan yang terdaftar dalam aplikasi.



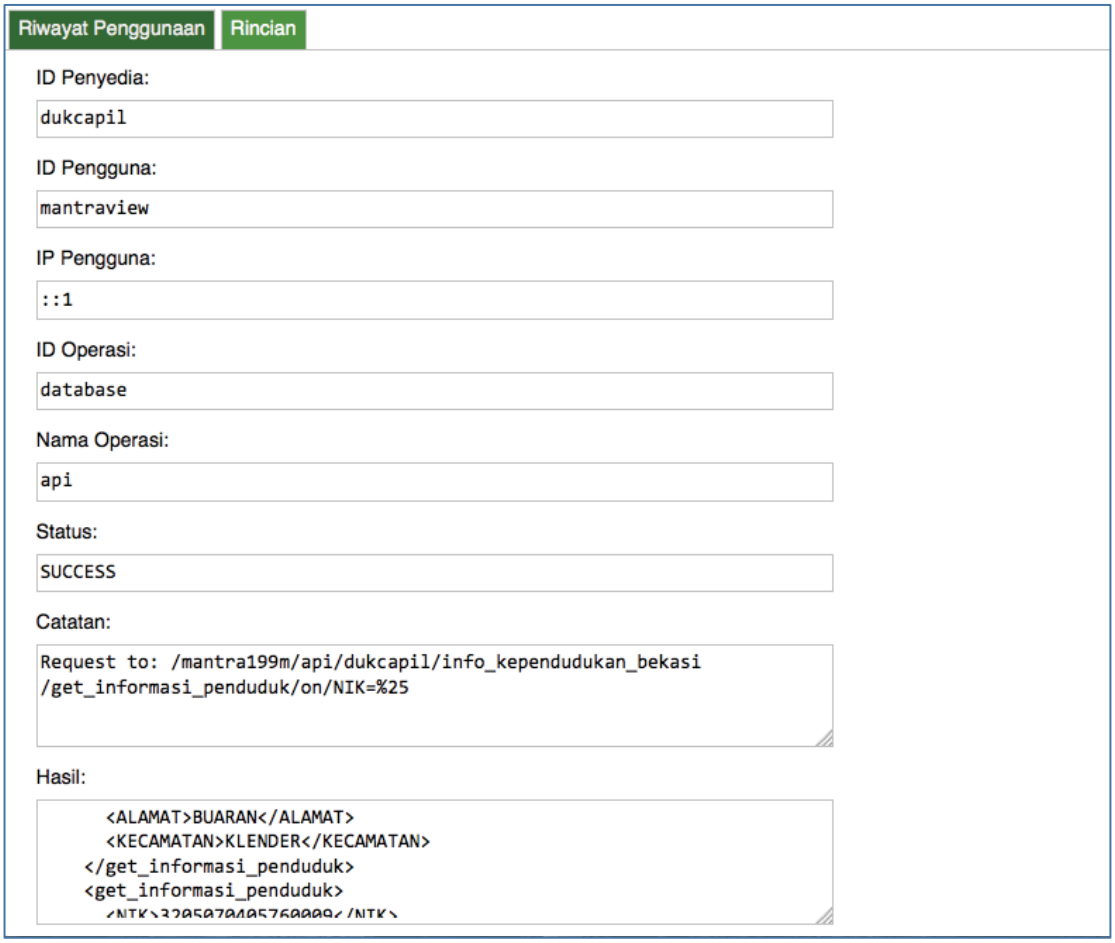
Gambar 3.45 Halaman Riwayat

Untuk mengakses informasi statistik pemanfaatan layanan, pengguna dapat mengakses submenu "**Analisa Layanan**" pada halaman riwayat. Dalam informasi statistik tersebut dicantumkan jumlah akses layanan yang berhasil dilaksanakan (terlaksana) dan jumlah akses layanan yang gagal dilaksanakan (terkendala) berdasarkan pengelompokkan instansi dan jenis operasi layanan. Jumlah terlaksana adalah total akses terhadap layanan yang berhasil direspon oleh server layanan. Jumlah terkendala adalah total akses terhadap layanan dimana server layanan tidak memberikan respon ataupun data yang dibutuhkan tidak ditemukan. Akses terkendela diberikan kode status 0 dengan kode *error* tertentu, kode status dan kode *error* tersebut dapat dilihat dalam Lampiran 1 dan Lampiran 2.



Gambar 3.46 Halaman Statistik Penggunaan Aplikasi

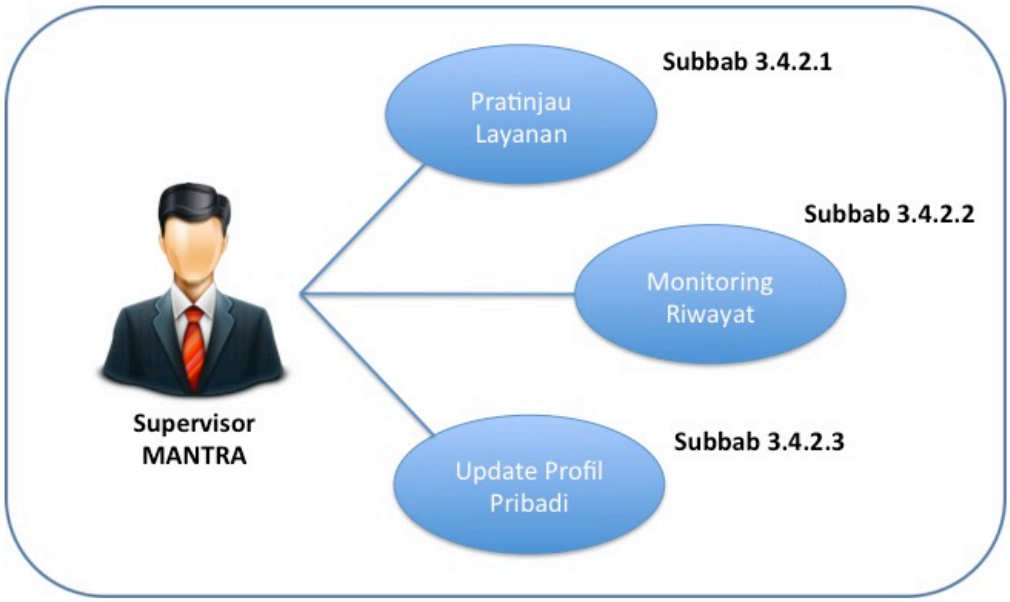
Kolom **JENIS OPERASI** pada halaman statistik penggunaan layanan menunjukkan jenis dari layanan yang diakses. Untuk baris yang bernilai "Layanan" menandakan bahwa layanan tersebut merupakan *Proxy Services* yang dibuat oleh Publisher. Jika bernilai "Data" berarti layanan yang dimaksud merupakan *Data Services* yang dibuat oleh Provider.



Gambar 3.47 Informasi Detail Riwayat Akses Layanan

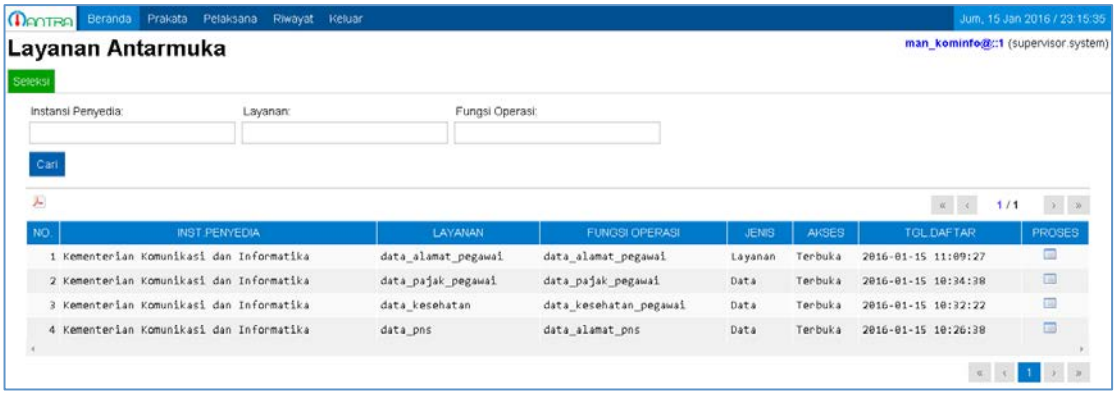
3.4.2 Panduan Supervisor

Peran **Supervisor** dalam Aplikasi MANTRA adalah melakukan monitoring aktivitas berbagi pakai layanan melalui MANTRA. Supervisor tidak terlibat langsung dalam kegiatan berbagi pakai layanan, peran supervisor dapat dianalogikan sebagai wasit yang mengawasi jalannya suatu pertandingan. Modul-modul yang disediakan dalam Aplikasi MANTRA untuk mendukung peran supervisor dalam melakukan monitoring antara lain pratinjau layanan dan monitoring riwayat. Gambar 3.48 di bawah ini merupakan aktivitas yang dilakukan oleh supervisor dalam Aplikasi MANTRA.



Gambar 3.48 Use Case Supervisor

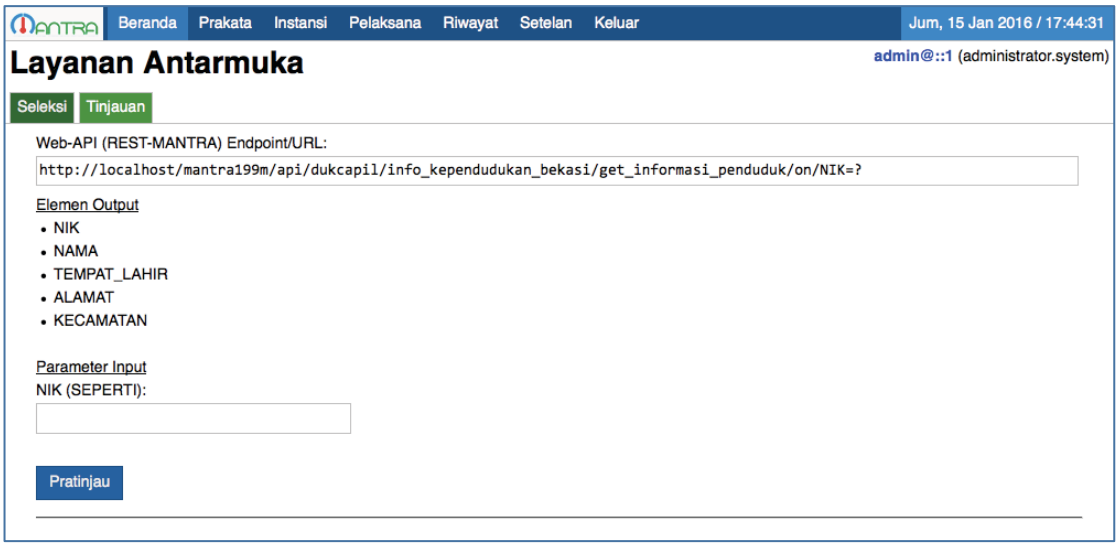
3.4.2.1 Pratinjau Layanan



Gambar 3.49 Halaman Pratinjau Supervisor MANTRA

Halaman Pratinjau dapat diakses melalui menu "**Beranda**", pada halaman tersebut ditampilkan layanan-layanan yang telah terdaftar di dalam Aplikasi MANTRA. Melalui ikon "Tinjauan" dalam kolom "**PROSES**", supervisor dapat melakukan uji fungsi layanan (*web service*). Adapun dalam kolom "**JENIS**" terdapat 3 (tiga) jenis keterangan layanan yaitu Layanan, Data, dan Program. Berikut adalah penjelasan dari masing-masing jenis layanan tersebut:

1. **Layanan** merupakan jenis layanan *proxy services* (penjelasan dapat dilihat pada subbab 2.4) yang dibuat oleh pengguna *publisher*.
2. **Data** merupakan jenis layanan *data services* (penjelasan dapat dilihat pada subbab 2.4) yang dibuat oleh pengguna *provider*.
3. **Program** merupakan jenis layanan terkustomisasi yang dibuat secara manual oleh pengguna *provider* menggunakan kode program PHP.

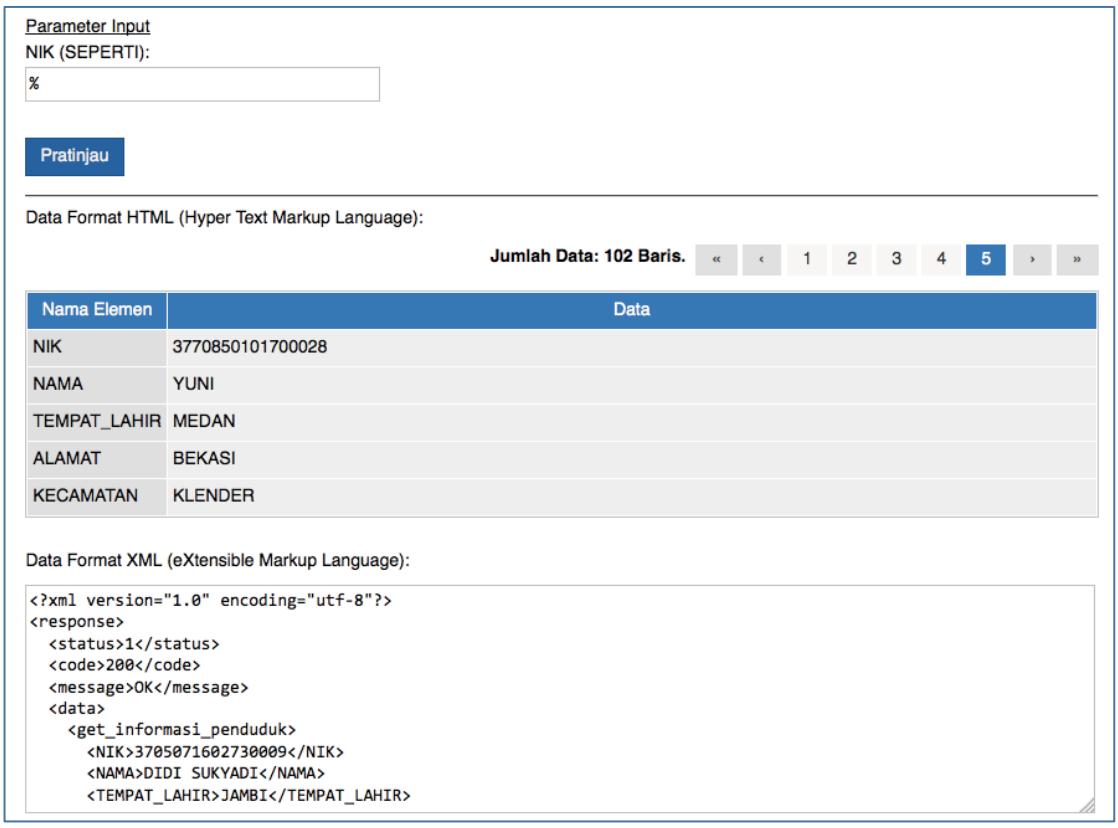


Gambar 3.50 Halaman Tinjauan Layanan

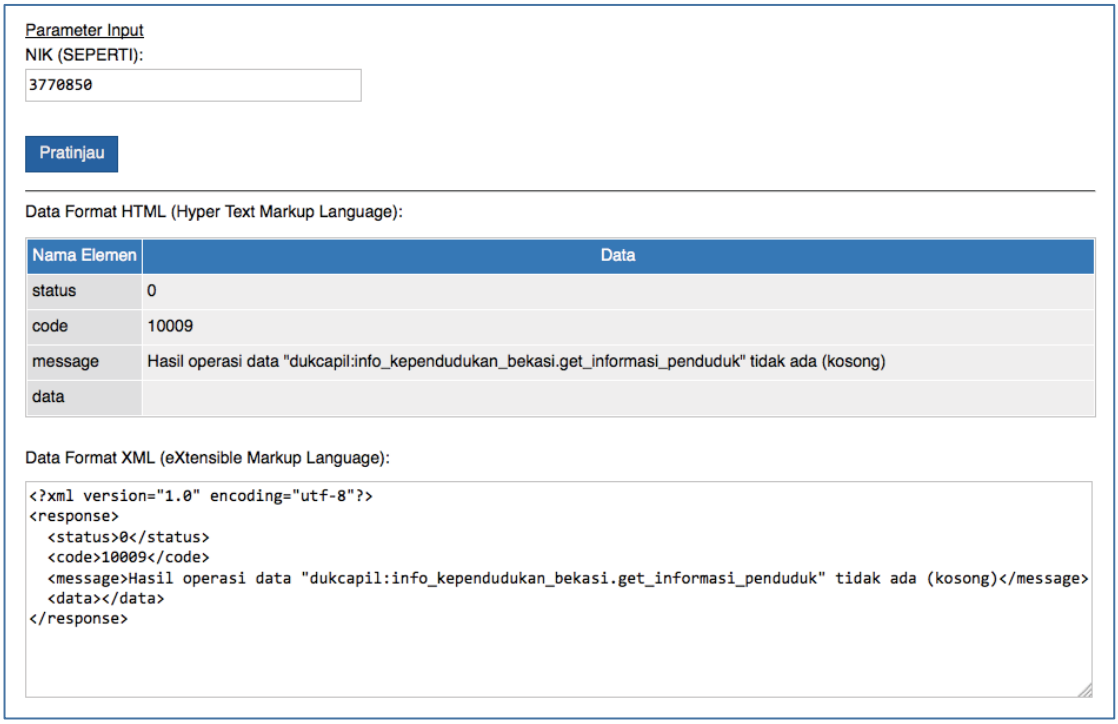
Gambar 3.50 di atas merupakan tampilan fitur tinjauan layanan yang digunakan untuk melakukan uji fungsi layanan. Pada halaman tersebut, supervisor harus mengisi parameter layanan (jika ada) sebelum melakukan eksekusi "Pratinjau". Pada contoh gambar tersebut parameter yang harus diisi adalah "NIK".

Selanjutnya keterangan "SEPERTI" dalam Gambar 3.50 menunjukkan operasi perbandingan parameter yang dilakukan ketika layanan dijalankan. "Elemen Output" menunjukkan elemen-elemen data yang disediakan oleh layanan pertukaran data. Untuk

melakukan uji fungsi layanan, supervisor harus menekan tombol Pratinjau setelah mengisi parameter yang dipersyaratkan (jika ada). Gambar 3.51 merupakan hasil uji fungsi layanan yang berjalan dengan baik, sedangkan Gambar 3.52 merupakan hasil uji fungsi layanan yang mengalami kendala. Masing-masing layanan memberikan output dengan kode tertentu, adapun daftar kode output MANTRA secara lengkap dapat dilihat dalam Lampiran 1 dan Lampiran 2.



Gambar 3.51 Uji Fungsi Layanan Sukses

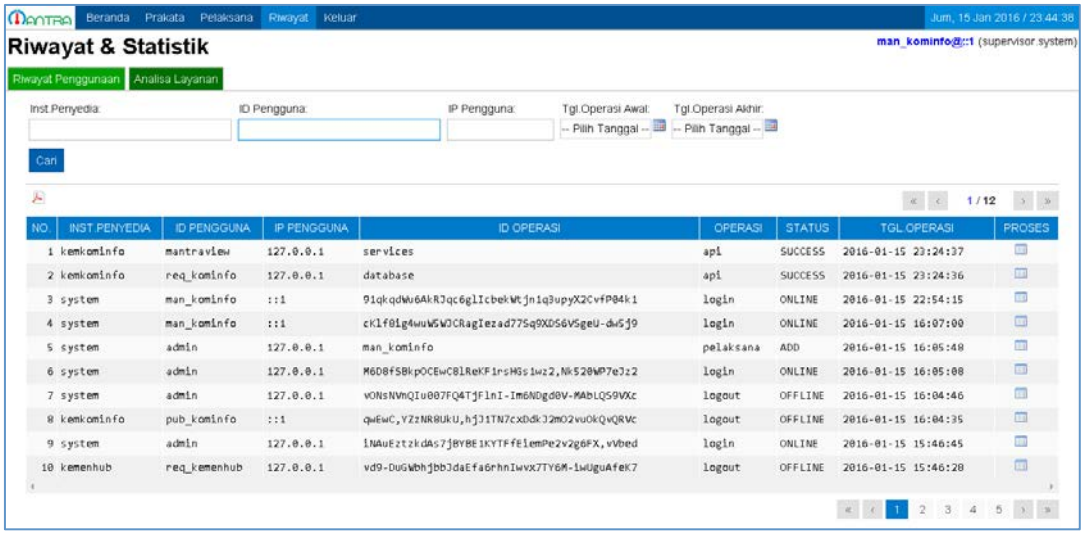


Gambar 3.52 Uji Fungsi Layanan Gagal

3.4.2.2 Monitoring Riwayat Aplikasi

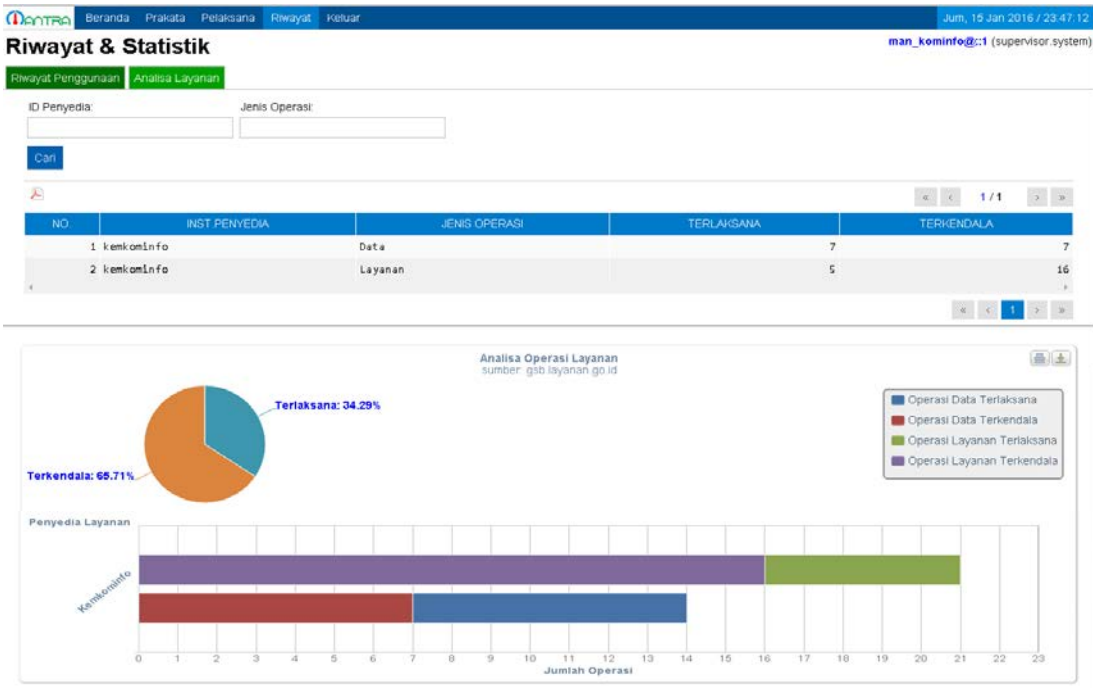
Modul ini digunakan membantu proses monitoring aktivitas pengguna dan aktivitas pemanfaatan layanan. Rekaman aktivitas tersebut ditampilkan pada halaman riwayat yang dapat diakses melalui menu "**Riwayat**". Ikon (Rincian Penggunaan) dalam kolom **PROSES** pada halaman riwayat digunakan untuk melihat detail informasi aktivitas yang dilakukan dalam Aplikasi MANTRA.

Aplikasi MANTRA merekam setiap aktivitas pengguna dan aktivitas penggunaan layanan. Hasil rekaman tersebut disimpan untuk tujuan audit dan monitor.



Gambar 3.53 Halaman Riwayat Pemanfaatan MANTRA

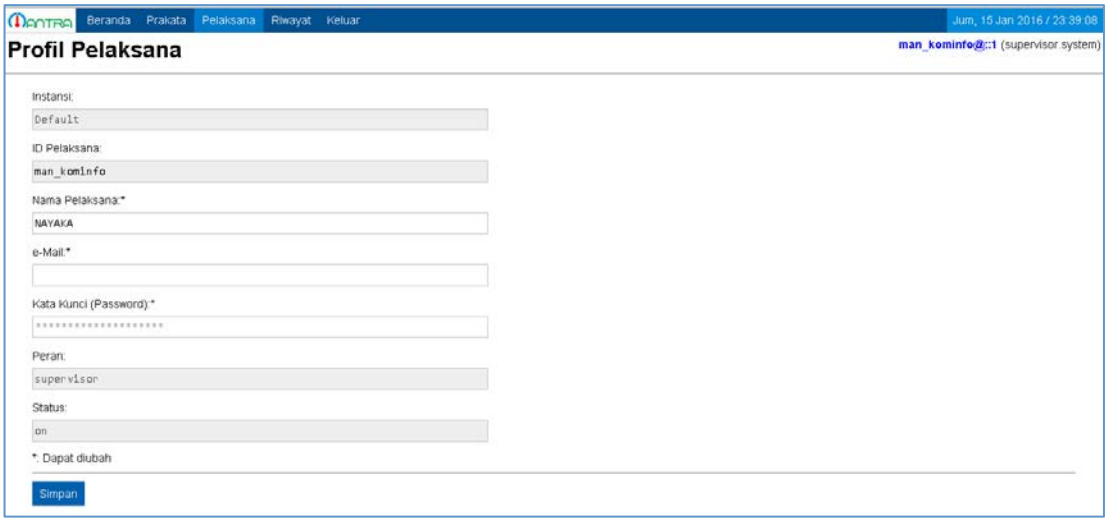
Pada halaman riwayat Aplikasi MANTRA tersedia fitur untuk melihat statistik pemanfaatan layanan yang disajikan dalam bentuk tabel dan grafik. Supervisor dapat melihat informasi statistik tersebut dengan mengakses submenu "**Analisa Layanan**". Informasi statistik yang diberikan dikelompokkan berdasarkan nama instansi dan jenis operasi layanan.



Gambar 3.54 Halaman Riwayat dan Statistik

3.4.2.3 Update Profil Supervisor

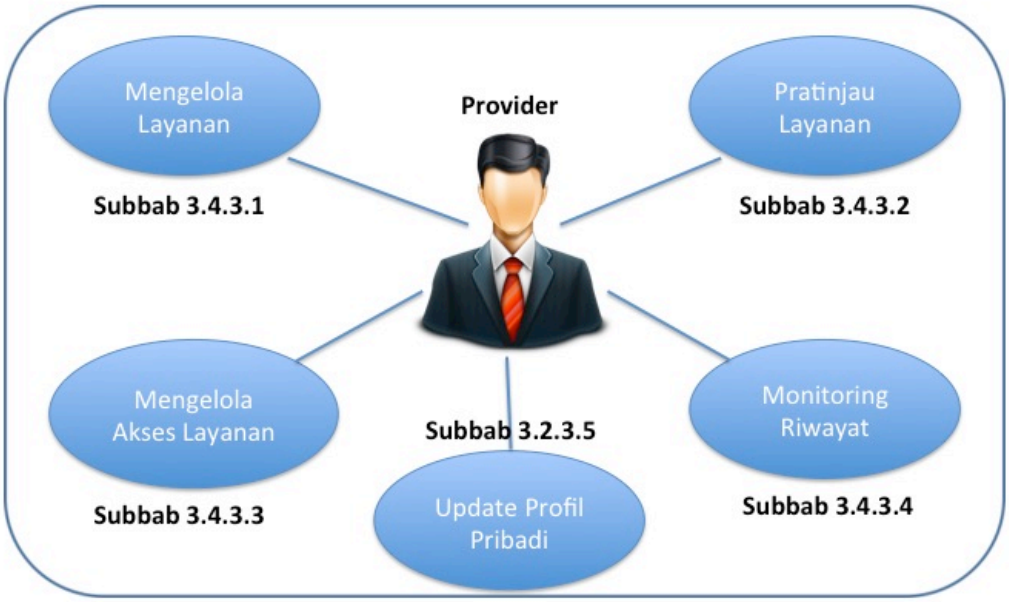
Untuk melakukan perubahan profil pribadi, supervisor dapat mengakses menu "**Pelaksana**". Melalui menu tersebut, supervisor diarahkan menuju halaman pelaksana. Pada halaman tersebut ditampilkan beberapa informasi terkait profil supervisor. Informasi yang dapat diubah antara lain nama pelaksana, e-mail dan kata kunci (password). Jika ingin melakukan perubahan ID Pengguna, supervisor harus mengajukan permohonan ubah data kepada administrator sebagaimana dijelaskan pada subbab 3.4.



Gambar 3.55 Halaman Ubah Profil Supervisor

3.4.3 Panduan Provider

Dalam proses pengelolaan layanan, **provider** adalah pihak pelaksana yang mewakili instansinya dalam membuat layanan dan mengatur hak akses pemanfaat layanan terhadap layanan yang dibuatnya. Ringkasan peran provider dapat dilihat pada Gambar 3.56 dimana penjelasan masing-masing peran dipaparkan dalam subbab-subbab yang tercantum dalam gambar tersebut.

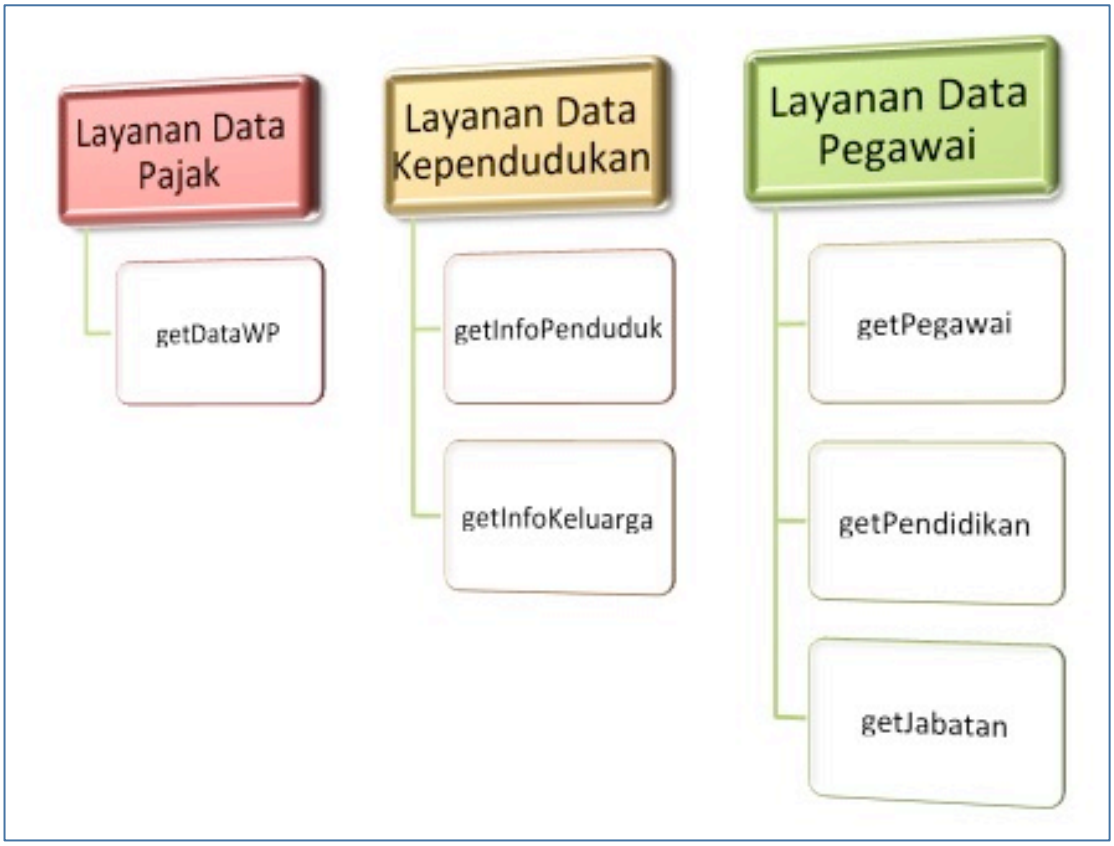


Gambar 3.56 Use Case Provider MANTRA

3.4.3.1 Pengelolaan Layanan

Layanan dalam Aplikasi MANTRA direpresentasikan dalam bentuk hierarki. Struktur hierarki layanan dapat dilihat dalam Gambar 3.57. Sebagaimana ditampilkan dalam gambar tersebut, setiap layanan memiliki satu atau lebih fungsi (API). Layanan Info Penduduk dan Layanan Info Pegawai pada gambar tersebut dapat dianalogikan sebagai direktori layanan, sementara itu fungsi (API) adalah sumber daya yang akan dibagikan kepada para pengguna layanan (requester).

Provider dapat membuat fungsi (API) data atau program dimana fungsi tersebut melekat pada satu direktori layanan. Langkah pertama yang dilakukan provider sebelum membuat fungsi (API) adalah mendefinisikan direktori layanan.

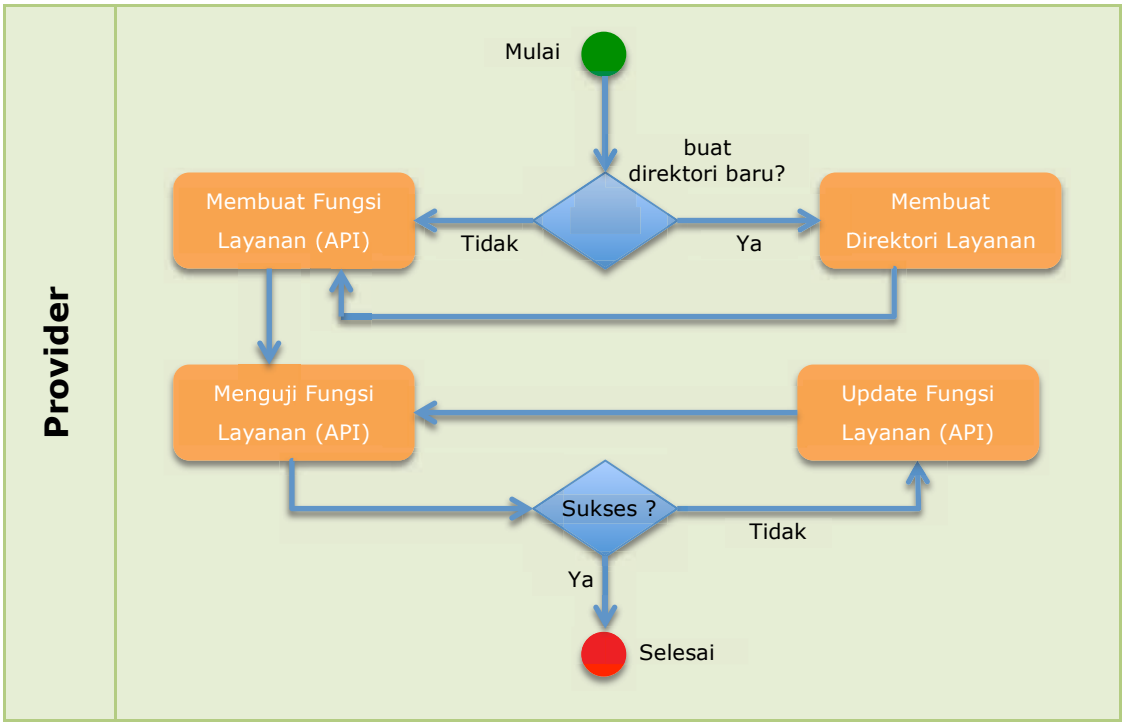


Gambar 3.57 Struktur Layanan MANTRA

Untuk membuat layanan (Web-API), provider harus mendefinisikan direktori layanan dan mendefinisikan fungsi layanan (API). Provider dapat memilih menggunakan direktori layanan yang sudah ada atau membuat direktori layanan baru untuk menempatkan fungsi (API) yang akan dibuat. Jika direktori layanan belum dibuat, maka langkah pertama yang harus dilakukan oleh provider adalah membuat direktori layanan. Alur kerja pembuatan layanan dalam Aplikasi MANTRA dapat dilihat pada gambar 3.57.

Jenis fungsi (API) yang dapat dibuat oleh provider adalah fungsi data (*data services*) dan fungsi program. Penjelasan kedua fungsi tersebut adalah sebagai berikut:

- Fungsi data bertujuan menyediakan antarmuka untuk mengakses data yang tersimpan dalam basis data elektronik (DBMS).
- Fungsi program merupakan antarmuka yang didefinisikan dalam bentuk kode program PHP.

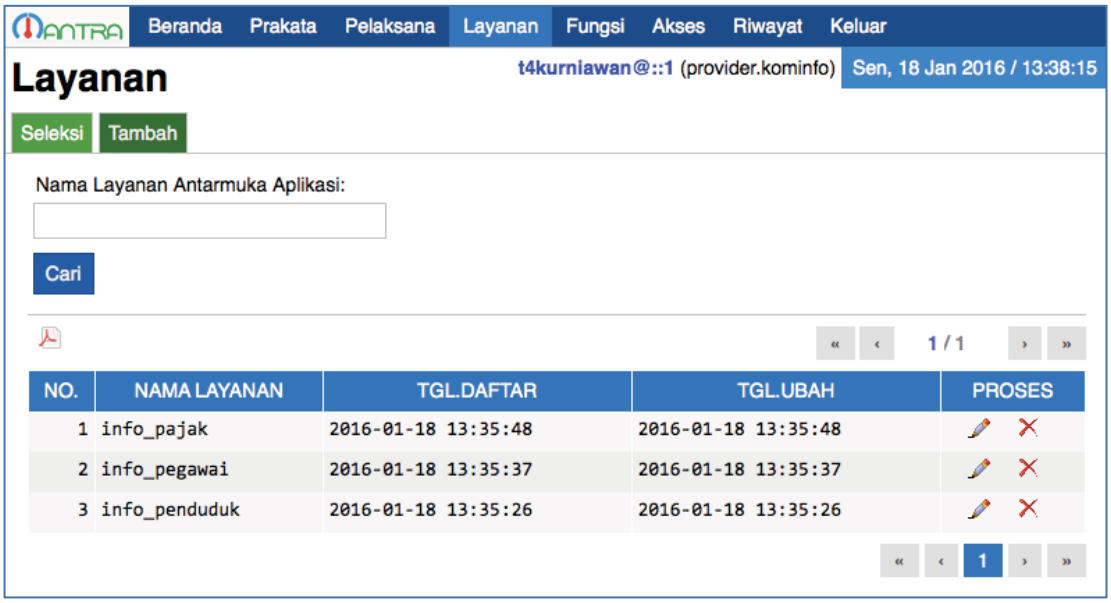


Gambar 3.58 Alur Kerja Pembuatan Layanan

3.4.3.1.1 Pengelolaan Direktori Layanan

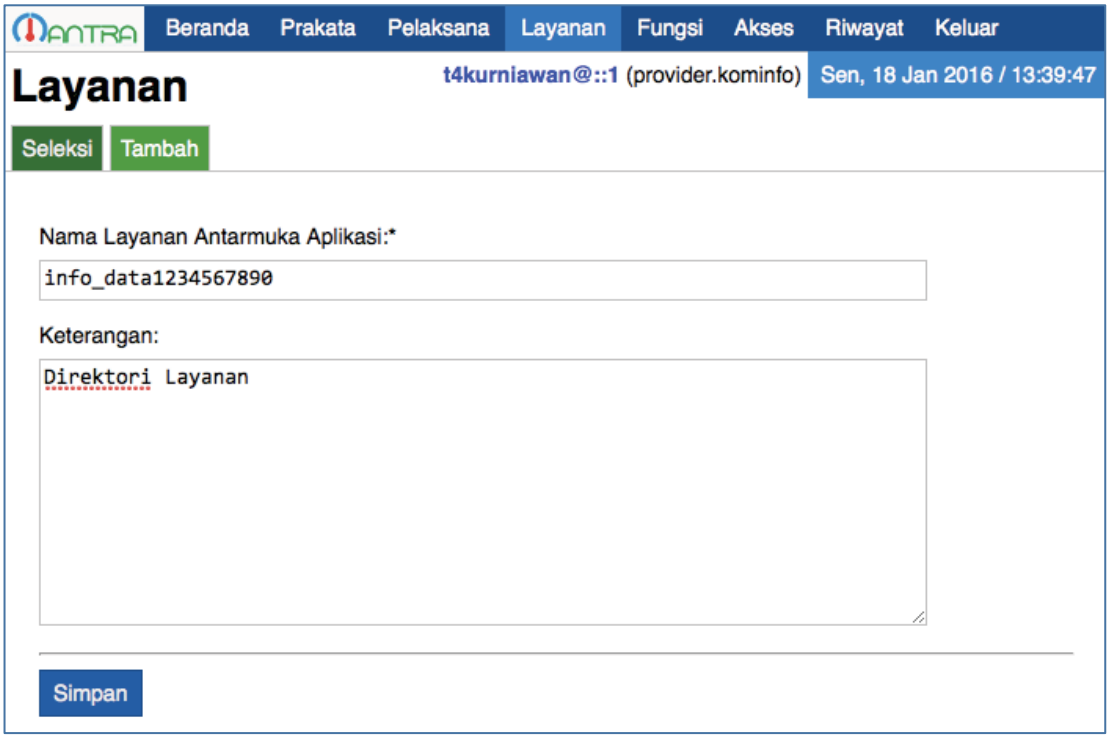
Aktivitas pengelolaan direktori layanan meliputi pembuatan direktori layanan, update nama direktori layanan, dan penghapusan direktori layanan. Sebagaimana dijelaskan dalam subbab 3.4.3.1 layanan tersusun secara hierarki, dimana satu atau lebih fungsi layanan (API) merupakan bagian dari direktori layanan. Oleh karena itu sebelum membuat fungsi layanan, provider harus memastikan tersedianya direktori layanan. Langkah pembuatan direktori layanan adalah sebagai berikut:

1. Akses menu **Layanan**, pada halaman layanan ditampilkan daftar direktori layanan yang telah didaftarkan (Gambar 3.59).
2. Akses submenu **Tambah**, melalui submenu tersebut provider diarahkan menuju halaman tambah direktori layanan (Gambar 3.60).
3. Isi kolom **Nama Layanan Antarmuka Aplikasi** (harus diisi) dan kolom **Keterangan** (optional). Pengisian Nama Layanan hanya dibatasi dengan kombinasi huruf kecil (a-z), angka (0-9), dan karakter "_".
4. Tekan tombol Simpan untuk menambahkan direktori layanan tersebut.



Gambar 3.59 Halaman Layanan Provider

Struktur layanan berupa hierarki, penghapusan direktori layanan secara otomatis akan menghapus semua fungsi layanan (API) yang terdaftar dalam direktori tersebut.

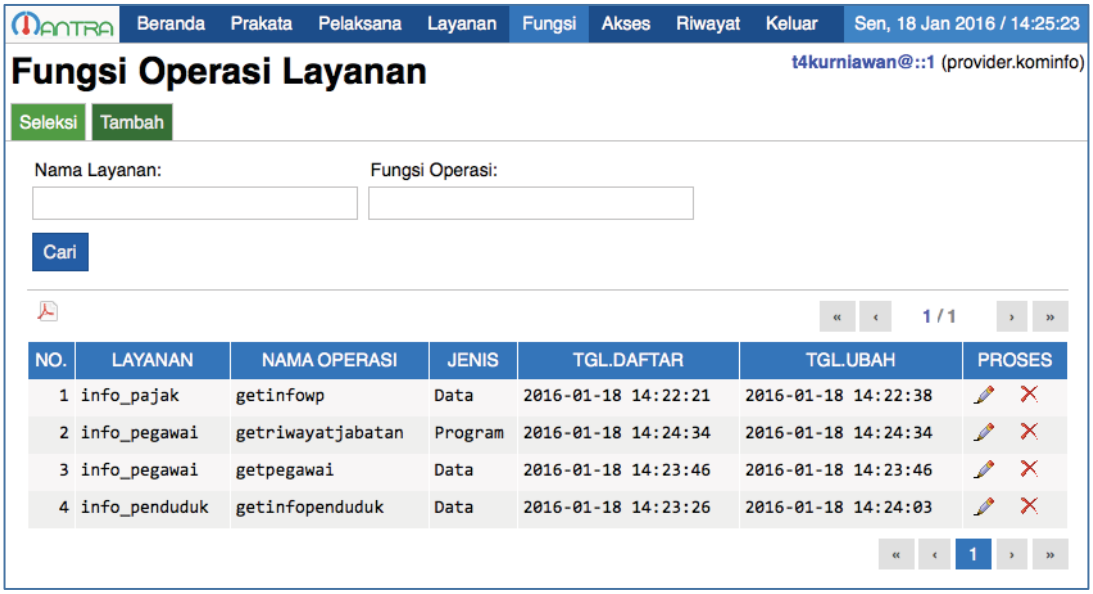


Gambar 3.60 Halaman Tambah Layanan Provider

Selain untuk menambahkan layanan baru, pada halaman layanan juga disediakan fitur untuk melakukan update dan menghapus direktori layanan.

3.4.3.1.2 Pengelolaan Fungsi Layanan

Seperti dijelaskan dalam subbab 3.4.3.1 fungsi (API) provider memiliki 2 (dua) jenis yaitu fungsi data dan fungsi program. Fitur pengelolaan fungsi layanan tersebut dapat diakses melalui menu **Fungsi**. Melalui menu tersebut provider diarahkan menuju halaman fungsi layanan (Gambar 3.61). Pada Halaman Fungsi Layanan ditampilkan seluruh fungsi layanan yang dimiliki oleh instansi dimana provider tersebut didaftarkan. Pada halaman fungsi layanan juga disediakan fitur untuk menambahkan fungsi baru, melakukan update fungsi layanan dan menghapus fungsi layanan.



Gambar 3.61 Halaman Fungsi Layanan Provider

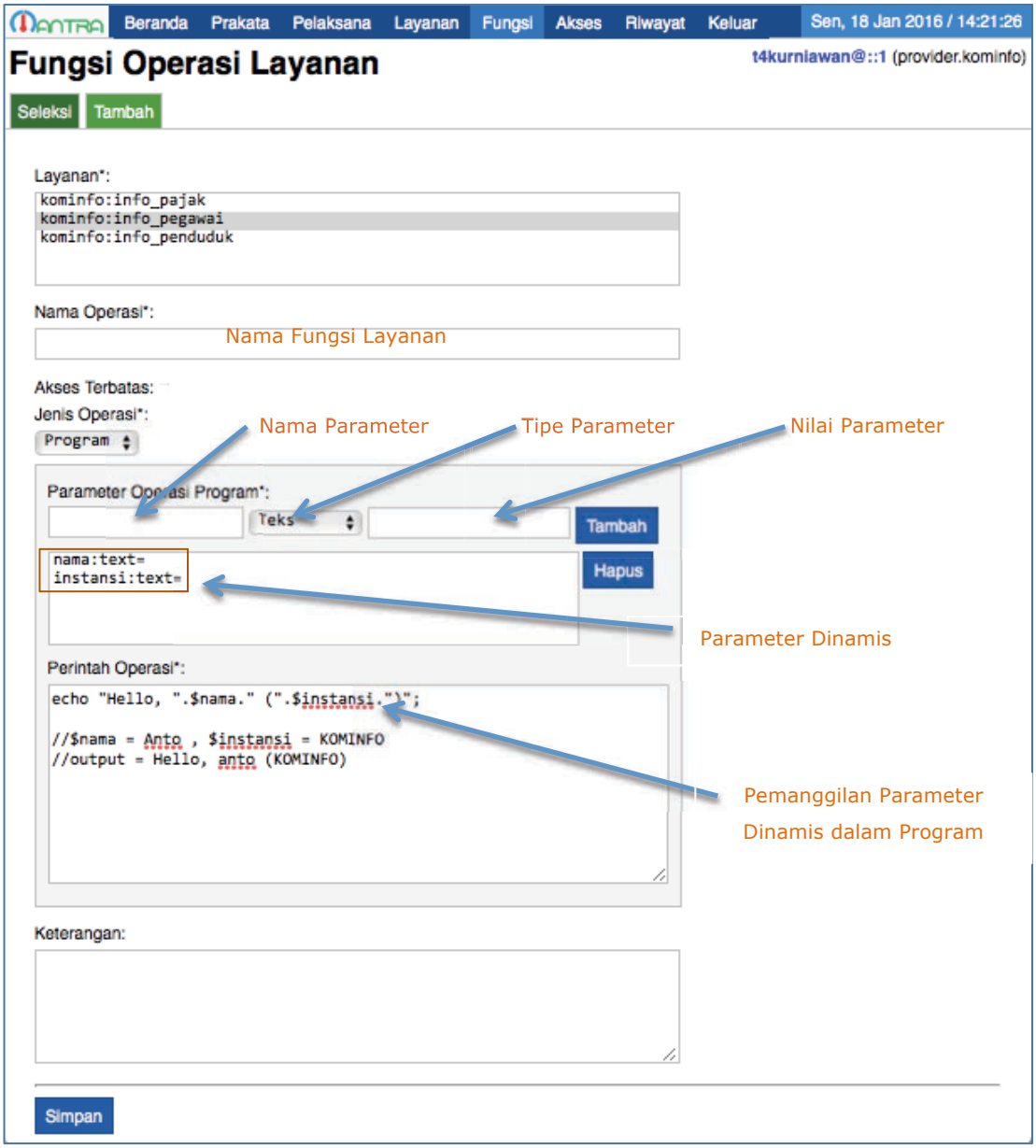
3.4.3.1.3 Pembuatan Fungsi Program

Langkah-langkah pembuatan fungsi dengan jenis program adalah sebagai berikut:

1. Akses menu **Fungsi**, pada halaman fungsi ditampilkan daftar fungsi layanan yang telah didaftarkan (Gambar 3.61).
2. Akses submenu **Tambah**, melalui submenu tersebut provider diarahkan menuju halaman tambah fungsi layanan (Gambar 3.62).

Jenis Parameter:

1. Parameter Statis adalah parameter operasi layanan yang nilainya telah ditentukan pada saat pembuatan fungsi layanan.
2. Parameter Dinamis adalah parameter operasi layanan yang nilainya tidak ditentukan pada saat pembuatan layanan. Pengguna layanan harus mengisi nilai parameter tersebut pada saat melakukan pemanggilan fungsi layanan.

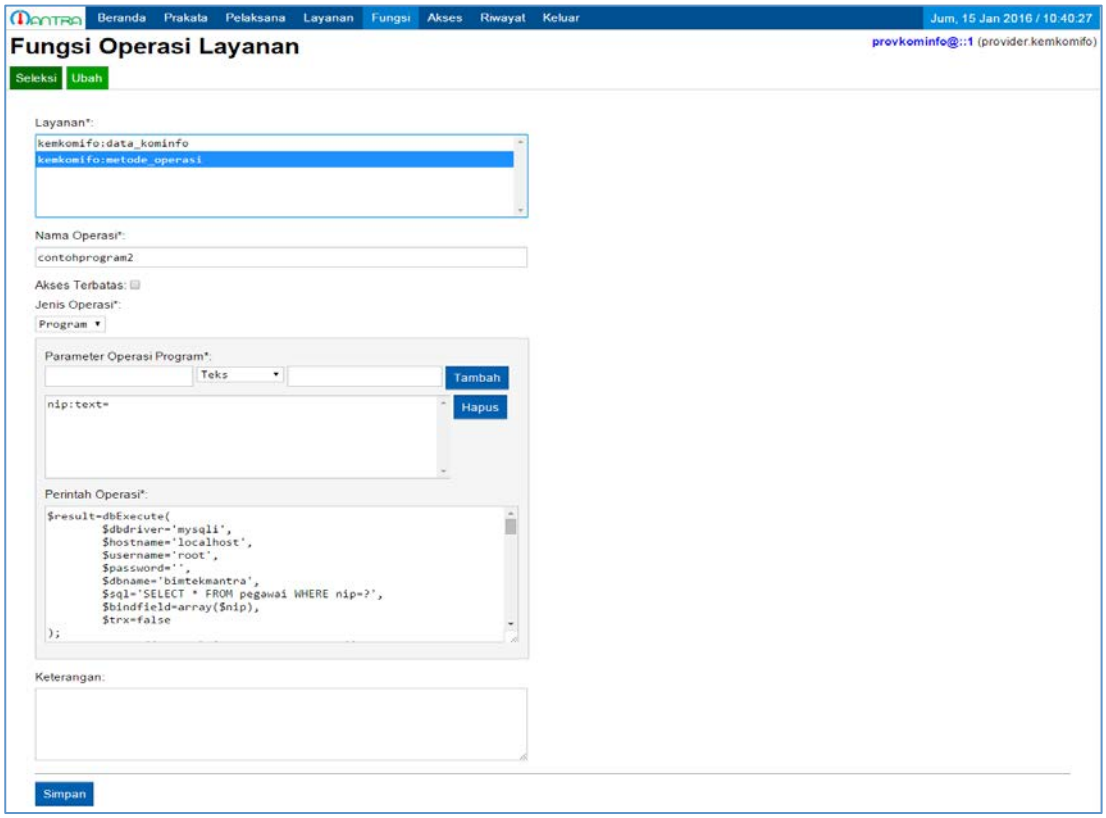


Gambar 3.62 Contoh Fungsi Program

3. Pada kolom **Layanan** pilih salah satu nama direktori layanan yang menjadi induk dari fungsi yang akan dibuat.
4. Isi kolom **Nama Operasi** dengan nama fungsi yang akan dibuat. Pengisian Nama Operasi hanya dibatasi dengan kombinasi huruf kecil (a-z), angka (0-9), dan karakter "_".
5. Pilih **Jenis Operasi** Program.
6. Tentukan **Parameter Operasi Program** (jika dibutuhkan/optional), isikan nama parameter pada kolom nama,

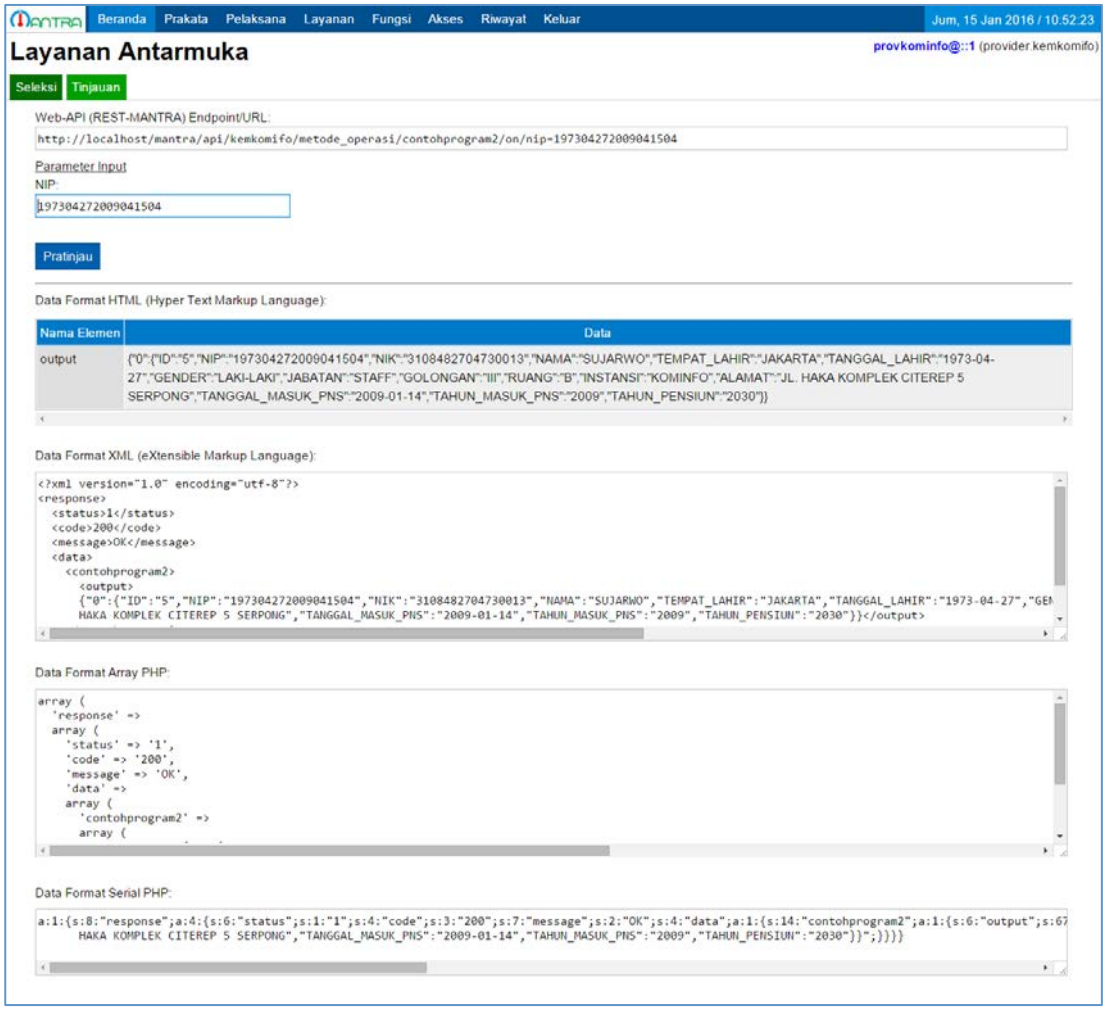
pilih tipe parameter (Teks/Numerik). Di sebelah kolom tipe parameter terdapat kolom nilai parameter. Kosongkan nilai parameter untuk membuat parameter dinamis.

7. Isi kolom **Perintah Operasi** dengan kode program PHP. Contoh sederhana kode program PHP dengan parameter dinamis dapat dilihat pada Gambar 3.62. Aplikasi MANTRA menyediakan template program PHP untuk pengoperasian SELECT, UPDATE, INSERT, dan DELETE sebagaimana dijelaskan dalam Lampiran 3. Contoh penggunaan template program dengan parameter dinamis dapat dilihat pada Gambar 3.63.
8. Isi kolom **Keterangan** dengan deskripsi fungsi (API) yang dibuat. Deskripsi meliputi manfaat dari fungsi tersebut, elemen data yang dibagikan, atau keterangan parameter yang dipersyaratkan.
9. Tekan tombol Simpan untuk menambahkan fungsi (API) tersebut.



Gambar 3.63 Pemanfaatan Template Fungsi Program MANTRA

10. Uji fungsi (API) melalui menu **Beranda**, klik ikon Tinjauan pada kolom "**PROSES**". Setelah tampil halaman tinjauan isi parameter yang dibutuhkan (jika ada), kemudian klik tombol Pratinjau. Output fungsi layanan yang muncul pada halaman tinjauan ditampilkan dalam format tabel, XML, JSON, array, dan serial. Output dengan Status "1" menunjukkan perintah operasi dalam fungsi (API) tersebut berhasil dijalankan. Sedangkan output dengan status "0" menunjukkan perintah operasi pada fungsi (API) tidak berhasil dijalankan. Daftar status dan kode output MANTRA selengkapnya dapat dilihat dalam Lampiran 1 dan Lampiran 2. Contoh output fungsi layanan yang berhasil dijalankan dapat dilihat pada Gambar 3.64.
11. Jika fungsi tidak berhasil dijalankan, periksa kembali perintah operasi fungsi melalui menu **Fungsi** dan klik ikon ubah data pada kolom **PROSES**. Simpan kembali perubahan yang dilakukan dan lakukan kembali uji fungsi sebagaimana ditunjukkan pada langkah ke 10 (sepuluh) di atas.



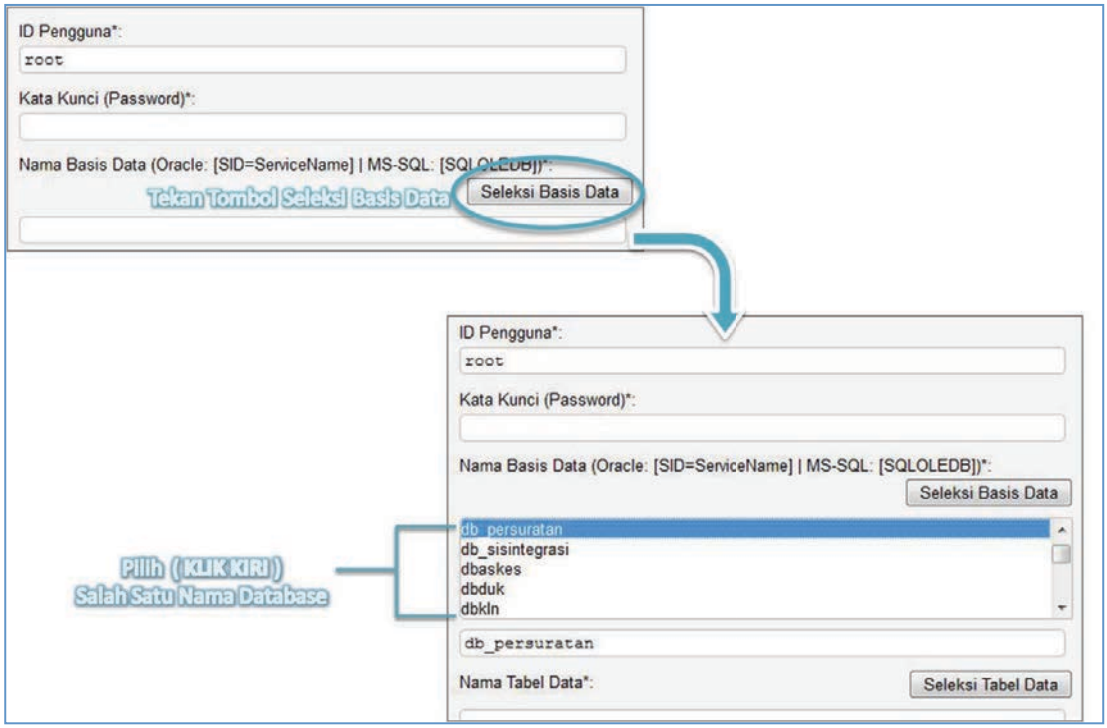
Gambar 3.64 Output Fungsi Layanan Program

3.4.3.1.4 Pembuatan Fungsi Data

Langkah-langkah pembuatan fungsi (API) dengan jenis data adalah sebagai berikut:

1. Akses menu **Fungsi**, pada halaman fungsi ditampilkan daftar fungsi layanan yang telah didaftarkan.
2. Akses sub menu **Tambah**, melalui submenu tersebut provider diarahkan menuju halaman tambah fungsi layanan.
3. Pada kolom **Layanan** pilih salah satu nama direktori layanan yang menjadi induk dari fungsi yang akan dibuat.

4. Isi kolom **Nama Operasi** dengan nama fungsi yang akan dibuat. Pengisian Nama Operasi hanya dibatasi dengan kombinasi huruf kecil (a-z), angka (0-9), dan karakter "_".
5. Tandai (Klik) keterangan **Akses Terbatas** jika tidak menginginkan deskripsi fungsi layanan ditampilkan pada halaman awal MANTRA.
6. Pilih **Jenis Operasi** Data.
7. Pilih **Sistem Basis Data (Database)** dimana data elektronik disimpan. Pilihan basis data meliputi MySQL, Oracle, PostgreSQL, dan Microsoft SQL Server.
8. Isi kolom **Alamat Lokasi Server** dengan alamat IP atau nama alias server basis data.
9. Isi kolom **Nomor Port** sesuai dengan nomor port yang digunakan untuk melakukan koneksi ke dalam basis data.
10. Isi kolom **ID Pengguna** dan **Kata Kunci (Password)** dengan akun user basis data yang memiliki *privilege* untuk melihat struktur basis data.
11. Isi kolom **Nama Basis Data**, khusus untuk basis data Oracle dan basis data MS SQL Server Nama Basis Data diisi secara manual. Pada basis data Oracle isi kolom tersebut dengan *service id* (SID). Sedangkan pada MS SQL Server isi kolom tersebut dengan nama instance basis data yang akan digunakan. Untuk basis data MySQL dan PostgreSQL tekan tombol Seleksi Basis Data, kemudian pilih salah satu dari beberapa nama basis data yang ditampilkan pada kolom tersebut.
12. Pilih **Nama Tabel Data**, tekan tombol Seleksi Tabel Data dan pilih salah satu nama tabel yang ditampilkan.

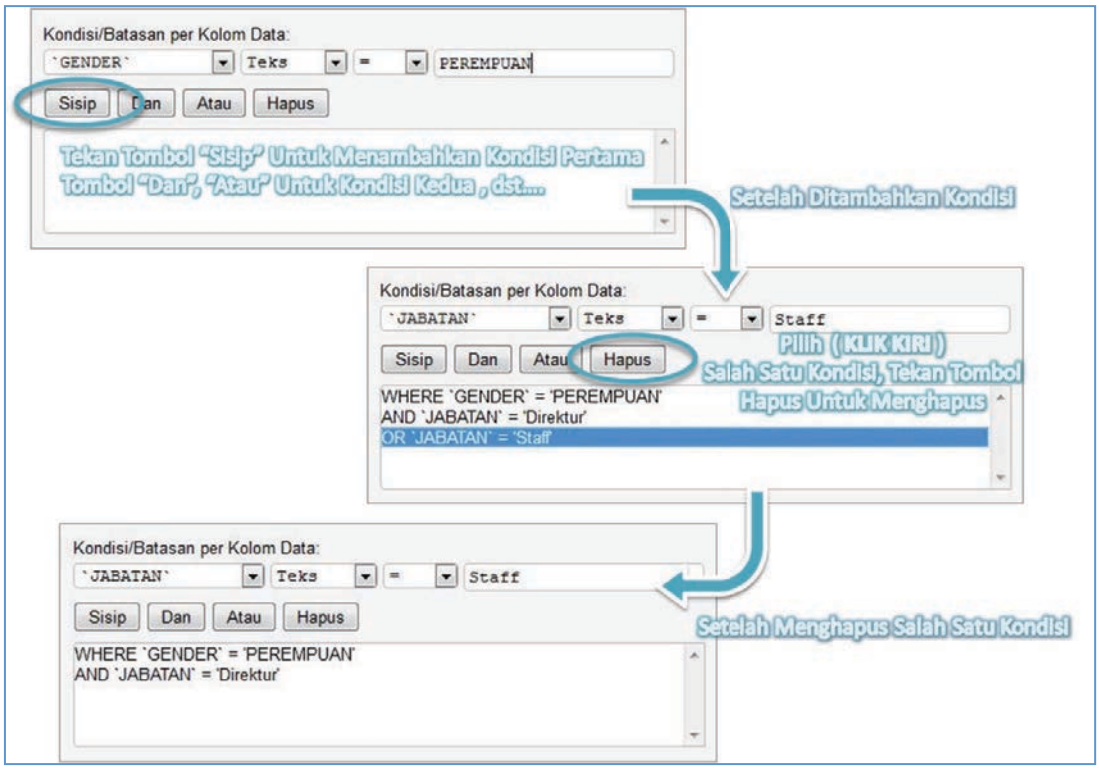


Gambar 3.65 Pemilihan Nama Basis Data

13. Pilih **Nama Kolom Data**, tekan tombol Seleksi Kolom Data dan pilih nama-nama kolom data yang diinginkan. Provider dapat memilih semua kolom data dengan menekan tombol **Semua Kolom** atau sebagian kolom data dengan memilih nama-nama kolom yang diinginkan kemudian tekan tombol Tambah Kolom. Nama kolom data yang terpilih masuk dalam daftar **Susunan Kolom Data**. Pilih salah satu nama kolom data dalam daftar susunan kolom data dan tekan tombol Eliminasi Kolom untuk membatalkan pemilihan kolom.
14. Pilih **Kondisi/Batasan per Kolom Data** (optional), kolom ini mengatur kondisi pencarian data (Query WHERE) berdasarkan kolom yang dipilih. Provider dapat menentukan beberapa kondisi pencarian data jika diperlukan. Kondisi pencarian pertama ditambahkan dengan menekan tombol Sisip. Kondisi pencarian kedua dan seterusnya ditambahkan dengan menekan tombol Dan untuk relasi kondisi "AND" atau menekan tombol Atau untuk relasi kondisi "OR". Untuk menghapus kondisi pencarian

yang sudah ditambahkan, pilih salah kondisi kemudian tekan tombol Hapus.

15. **Urutan Data** (Optional), kolom ini mengatur urutan data output sesuai dengan Nama Kolom Data yang dijadikan sebagai acuan pengurutan.
16. **Jumlah Baris** (Optional), kolom ini mengatur jumlah maksimal data yang terseleksi ketika query data dijalankan. **Awal Baris**, kolom ini digunakan sebagai penanda baris pertama dari basis data yang akan diambil ketika query dijalankan.



Gambar 3.66 Memilih Kondisi/Batasan per Kolom Data

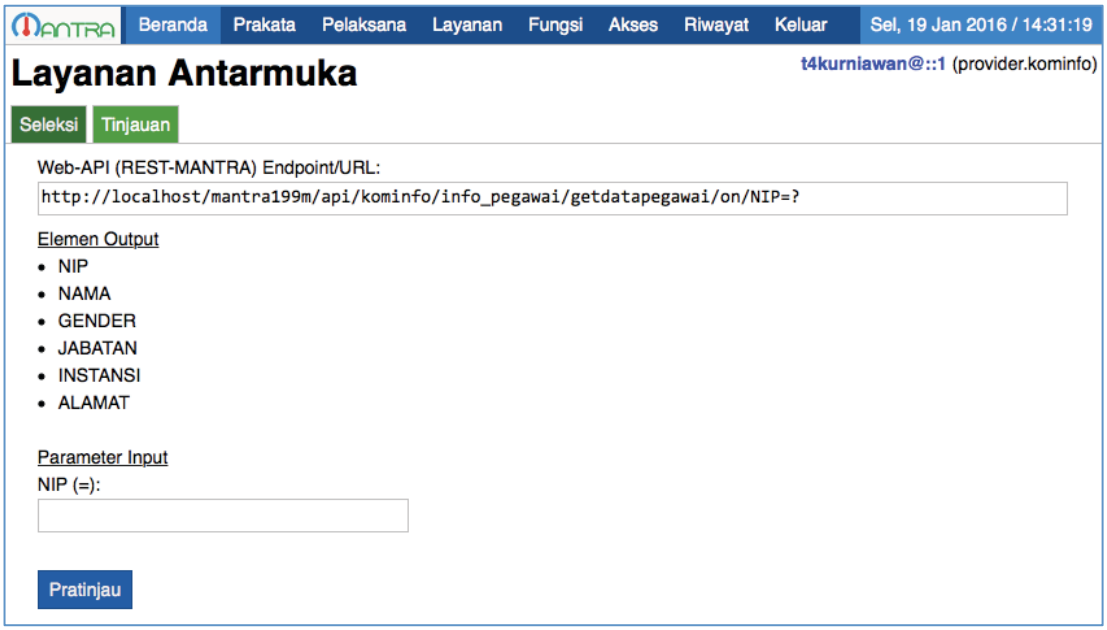
Kondisi pencarian data terdiri dari Nama Kolom Data, Jenis Operasi ("=", "LIKE", "<>", ">", ">=", "<", "<="), dan Nilai Kolom Data. Jika nilai kolom data dibiarkan kosong, maka Nama Kolom Data akan menjadi Parameter dari Fungsi Layanan.

17. Setelah menjalankan langkah 1 s.d 16 maka dihasilkan perintah operasi data sebagaimana yang ditampilkan pada Kolom Perintah Pengolahan Data (SQL).
18. Isi kolom **Keterangan** dengan deskripsi fungsi (API) yang dibuat. Deskripsi meliputi manfaat dari fungsi tersebut, elemen data yang dibagikan, atau keterangan parameter yang dipersyaratkan.
19. Tekan tombol Simpan untuk menambahkan fungsi (API) tersebut.
20. Uji fungsi (API) melalui menu **Beranda**, klik ikon Tinjauan pada kolom "**PROSES**". Setelah tampil halaman tinjauan isi parameter yang dibutuhkan (jika ada), kemudian klik tombol Pratinjau. Output fungsi layanan yang muncul pada halaman tinjauan ditampilkan dalam format tabel, XML, JSON, array, dan serial. Output dengan Status "1" menunjukkan perintah operasi dalam fungsi (API) tersebut berhasil dijalankan. Sedangkan output dengan status "0" menunjukkan perintah operasi pada fungsi (API) tidak berhasil dijalankan. Daftar status dan kode output MANTRA selengkapnya dapat dilihat dalam Lampiran 1 dan Lampiran 2.
21. Jika fungsi tidak berhasil dijalankan, periksa kembali perintah operasi fungsi melalui menu **Fungsi** dan klik ikon ubah data pada kolom **PROSES**. Simpan kembali perubahan yang dilakukan dan lakukan kembali uji fungsi sebagaimana ditunjukkan pada langkah ke 20 (dua puluh) di atas.

3.4.3.2 Pratinjau Layanan

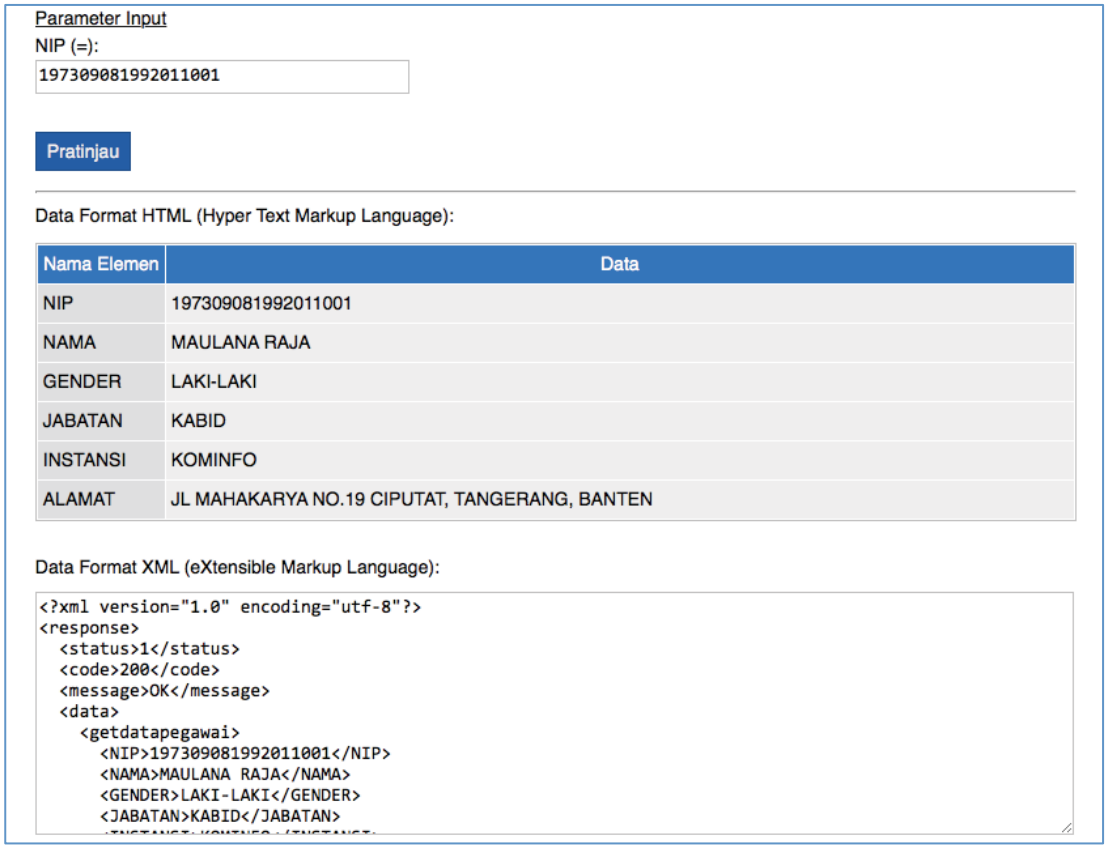
Melalui fungsi pratinjau layanan seorang provider dapat melakukan uji fungsi layanan untuk mengetahui apakah fungsi layanan berjalan dengan baik atau terkendala. Fungsi Pratinjau Layanan dapat

diakses melalui menu "**Beranda**". Pada halaman tersebut akan ditampilkan tabel yang berisi nama-nama fungsi layanan yang dimiliki oleh instansi dimana provider terdaftar. Untuk melakukan uji fungsi layanan tekan ikon "Tinjauan" pada kolom "**PROSES**" di Halaman Beranda.

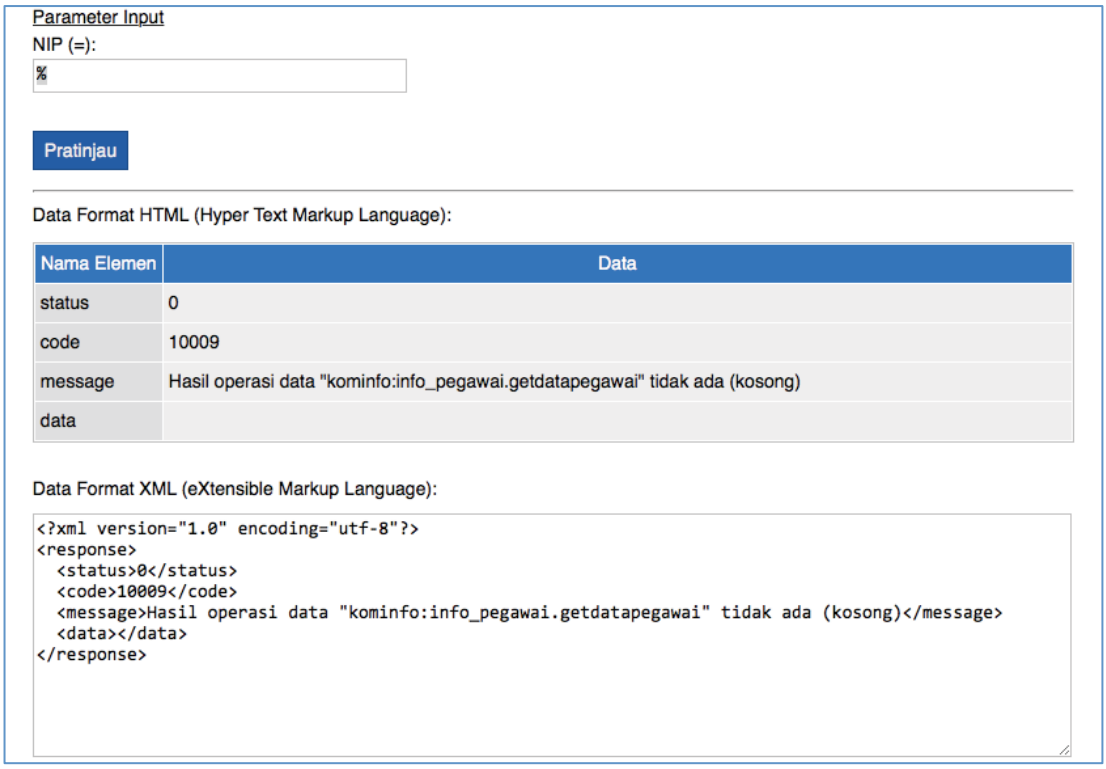


Gambar 3.67 Halaman Tinjauan Layanan Provider

Gambar 3.67 di atas merupakan tampilan fitur tinjauan layanan MANTRA. Provider harus mengisi parameter layanan (jika ada) sebelum melakukan pratinjau. Pada contoh gambar di atas parameter yang harus diisi adalah "NIP". Keterangan "=" pada gambar di atas menunjukkan operasi perbandingan parameter yang dilakukan ketika fungsi layanan dijalankan. "Elemen Output" menunjukkan elemen-elemen data yang dihasilkan oleh fungsi layanan tersebut. Gambar 3.68 menampilkan hasil tinjauan layanan yang berfungsi dengan baik, sedangkan Gambar 3.69 menampilkan hasil tinjauan layanan yang mengalami kendala akses. Adapun daftar kode output MANTRA selengkapnya dapat dilihat dalam Lampiran 1 dan Lampiran 2.



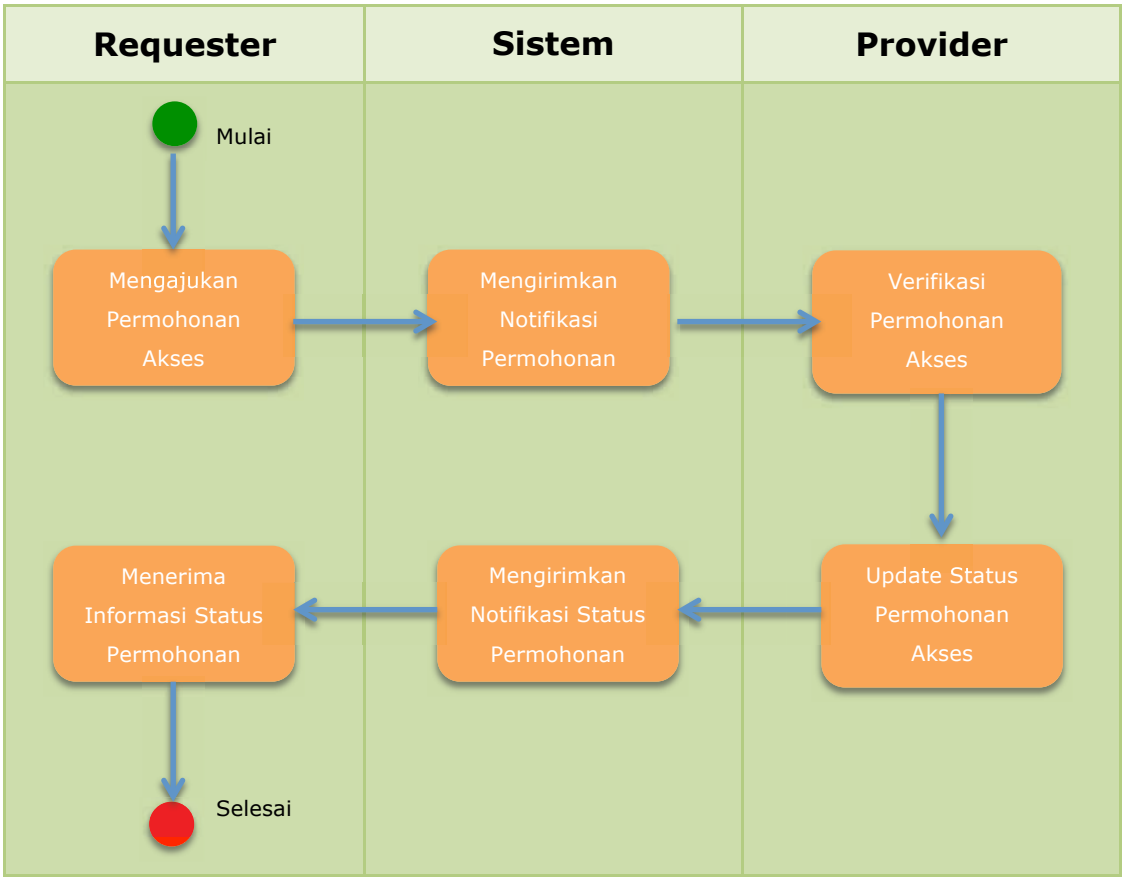
Gambar 3.68 Fungsi Layanan Berjalan dengan Baik



Gambar 3.69 Fungsi Layanan Terkendala

3.4.3.3 Pengelolaan Akses Layanan

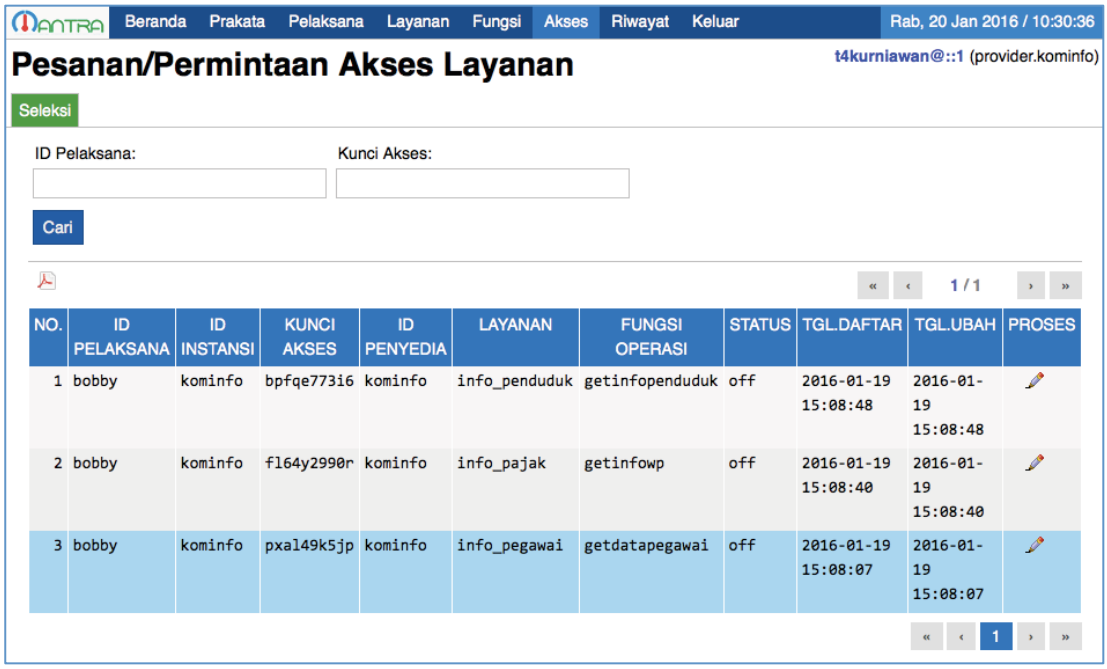
Pengelolaan akses layanan adalah aktivitas provider dalam rangka membuka atau menutup akses pengguna layanan (requester) terhadap layanan yang terdaftar di instansinya. Semua akses terhadap layanan tidak dapat dilakukan tanpa persetujuan dari provider. Gambar 3.70 merupakan alur permohonan akses layanan dari requester kepada provider.



Gambar 3.70 Alur Proses Permohonan Akses Layanan

Setiap permohonan akses layanan ditambahkan, sistem akan mengirimkan notifikasi melalui email kepada provider. Daftar permohonan tersebut akan ditampilkan pada Halaman Permintaan Akses Layanan yang diakses melalui menu "**Akses**". Status awal ketika permohonan ditambahkan adalah "**off**". Untuk membuka

akses requester terhadap layanan, provider harus meng-update status permohonan tersebut menjadi "**on**".



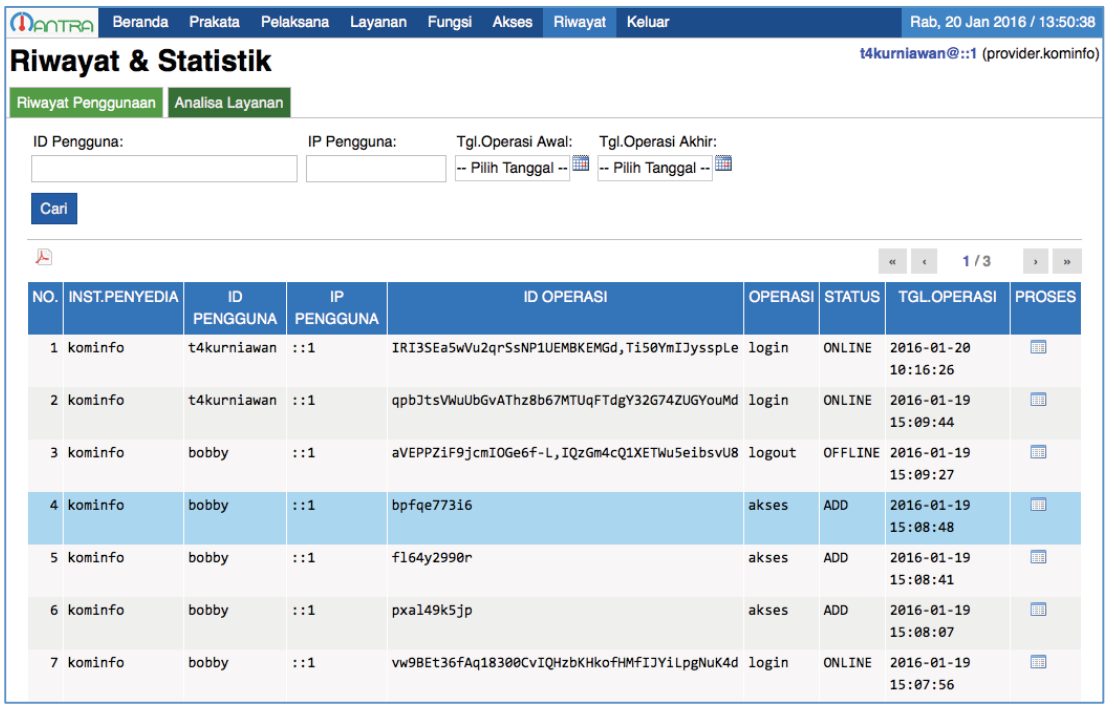
Gambar 3.71 Halaman Permintaan Akses Layanan

Langkah-langkah untuk mengubah status permohonan akses layanan adalah sebagai berikut:

1. Klik ikon ubah data pada Halaman Permintaan Akses Layanan (Gambar 3.71).
2. Pada halaman ubah data update status permintaan menjadi off atau on. Update status menjadi off untuk menutup akses requester terhadap layanan atau update status menjadi on untuk membuka akses requester terhadap layanan.
3. Klik tombol Simpan untuk menyimpan perubahan status tersebut.

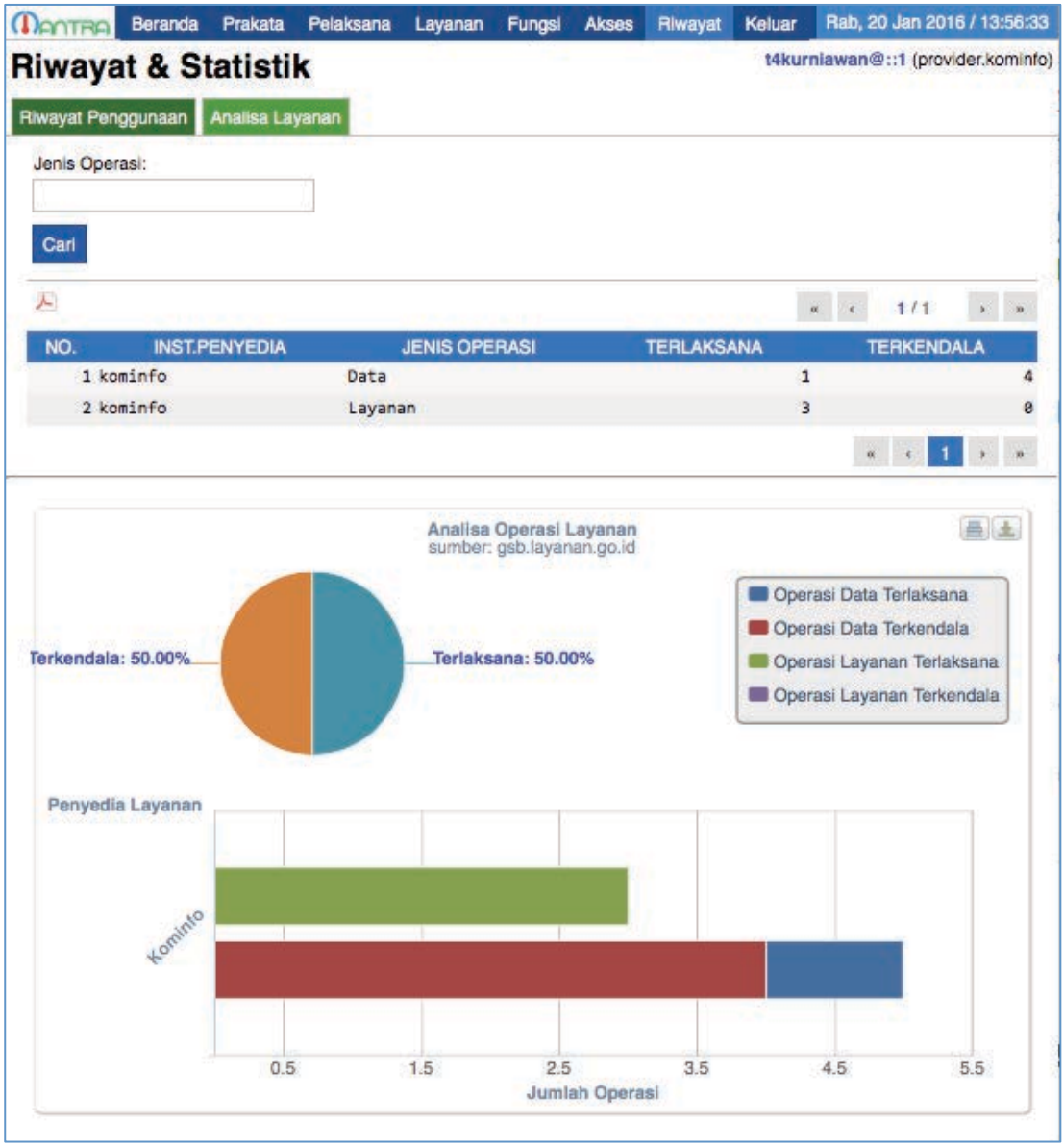
3.4.3.4 Monitoring Riwayat Aplikasi

Modul ini digunakan untuk melakukan monitoring aktivitas pengguna dan aktivitas pemanfaatan layanan pada instansi dimana provider didaftarkan. Aktivitas-aktivitas tersebut direkam dan kemudian hasil rekamannya dapat dilihat melalui menu "**Riwayat**". Ikon  (Rincian Penggunaan) pada kolom **PROSES** halaman riwayat digunakan untuk melihat detail rekaman pemanfaatan MANTRA.



Gambar 3.72 Halaman Riwayat Pemanfaatan Provider

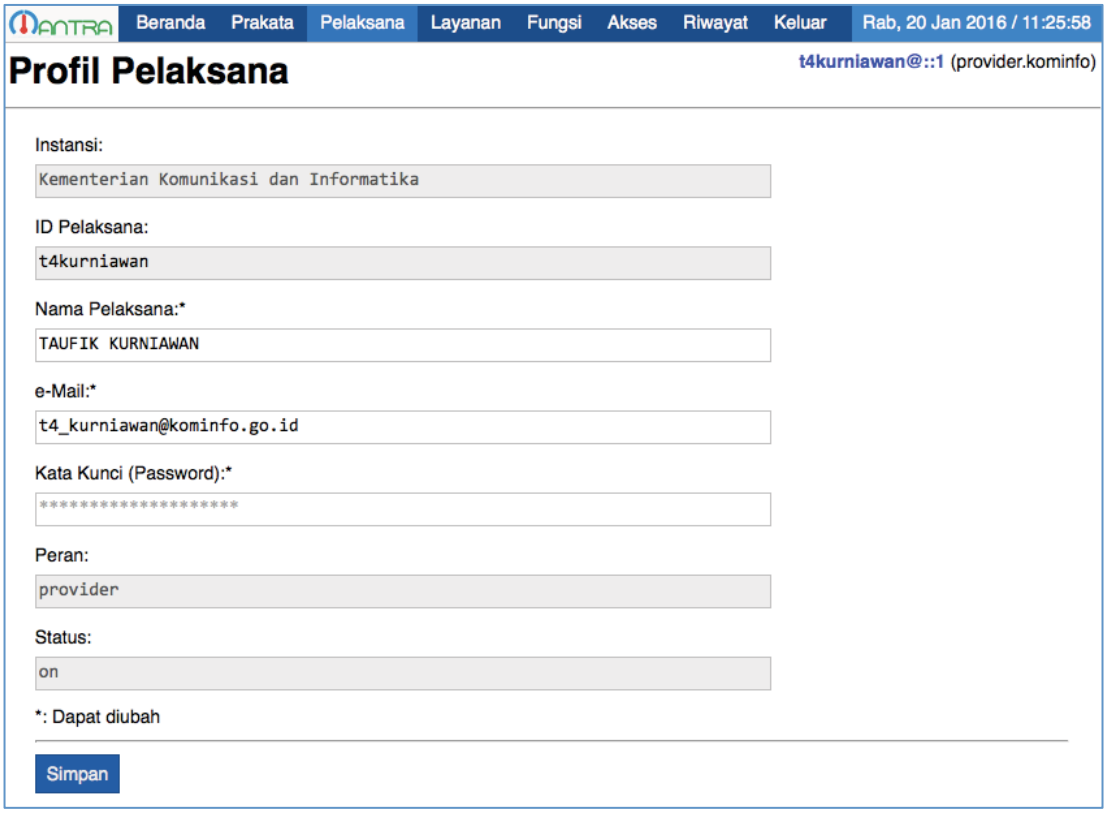
Pada halaman riwayat MANTRA juga tersedia informasi statistik pemanfaatan layanan yang ditampilkan dalam bentuk tabel dan grafik. Untuk melihat informasi tersebut, provider harus mengakses sub-menu "**Analisa Layanan**". Informasi statistik yang ditampilkan akan dikelompokkan berdasarkan jenis operasi layanan yang dimiliki oleh instansi yang bersangkutan.



Gambar 3.73 Halaman Riwayat dan Statistik

3.4.3.5 Update Profil Provider

Provider dapat mengubah informasi terkait profil pribadi dengan mengakses menu "**Pelaksana**". Informasi-informasi yang dapat diubah antara lain nama pelaksana, e-mail, dan kata kunci (password). Jika ingin melakukan perubahan ID Pengguna, provider harus mengajukan permohonan ubah data kepada administrator sebagaimana dijelaskan pada subbab 3.4.



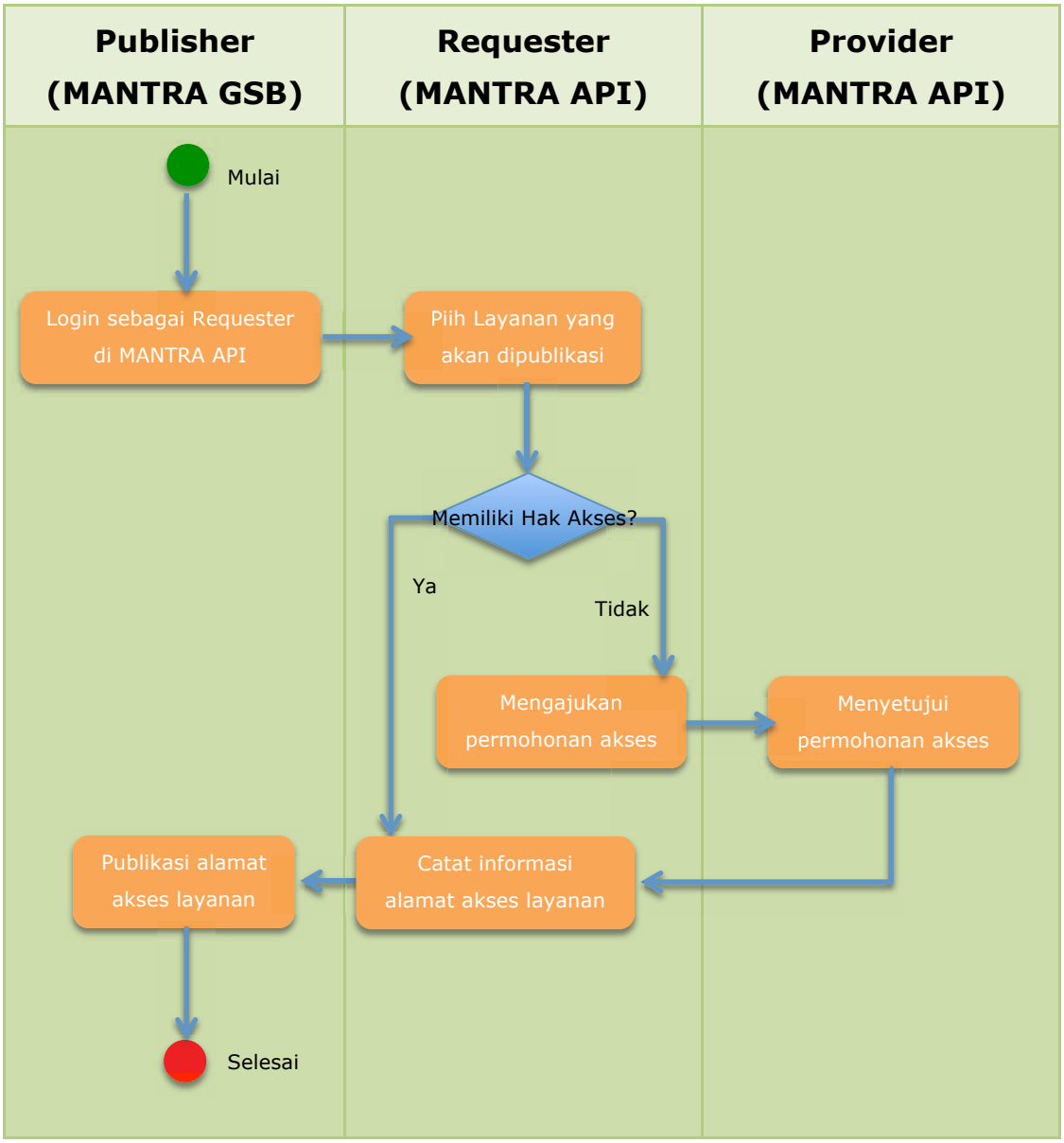
Gambar 3.74 Halaman Ubah Profil Provider

3.4.4 Panduan Publisher

Publisher merupakan pihak penyedia layanan yang mendaftarkan alamat akses layanan (Web-API) yang dimiliki ke dalam MANTRA GSB (Government Service Bus). Adapun kategori layanan (Web-API) yang didaftarkan adalah sebagai berikut:

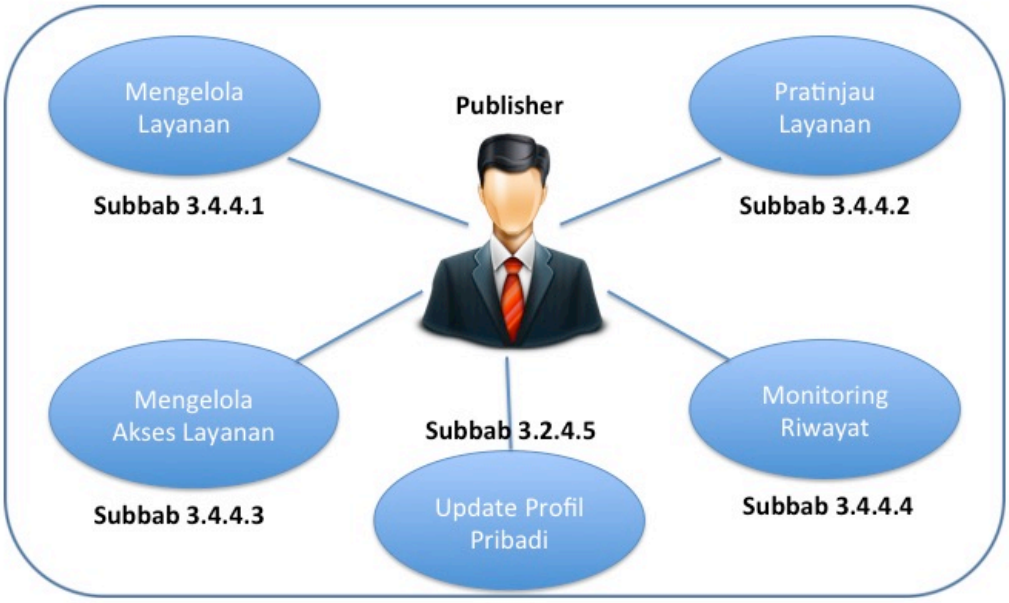
- **Layanan (Web-API) Eksternal** yaitu Web-API yang dibuat oleh penyedia layanan secara mandiri tanpa menggunakan Aplikasi MANTRA. Aplikasi MANTRA mendukung publikasi layanan eksternal baik yang menggunakan metode SOAP, HTTP/GET, HTTP/POST, atau REST.
- **Layanan (Web-API) Internal** yaitu Web-API yang dibuat oleh penyedia layanan menggunakan Aplikasi MANTRA (MANTRA API) dan secara *default* menggunakan metode REST. Publisher layanan internal di MANTRA GSB harus

berperan sebagai requester bagi penyedia layanan di MANTRA API untuk mendapatkan alamat akses layanan (Web-API). Proses pendaftaran layanan internal dapat dilihat dalam Gambar 3.75 di bawah ini.



Gambar 3.75 Alur Proses Publikasi Layanan Internal

Ringkasan peran publisher dapat dilihat dalam Gambar 3.76. Adapun penjelasan dari masing-masing peran tersebut dipaparkan dalam subbab yang ditunjuk dalam gambar tersebut.

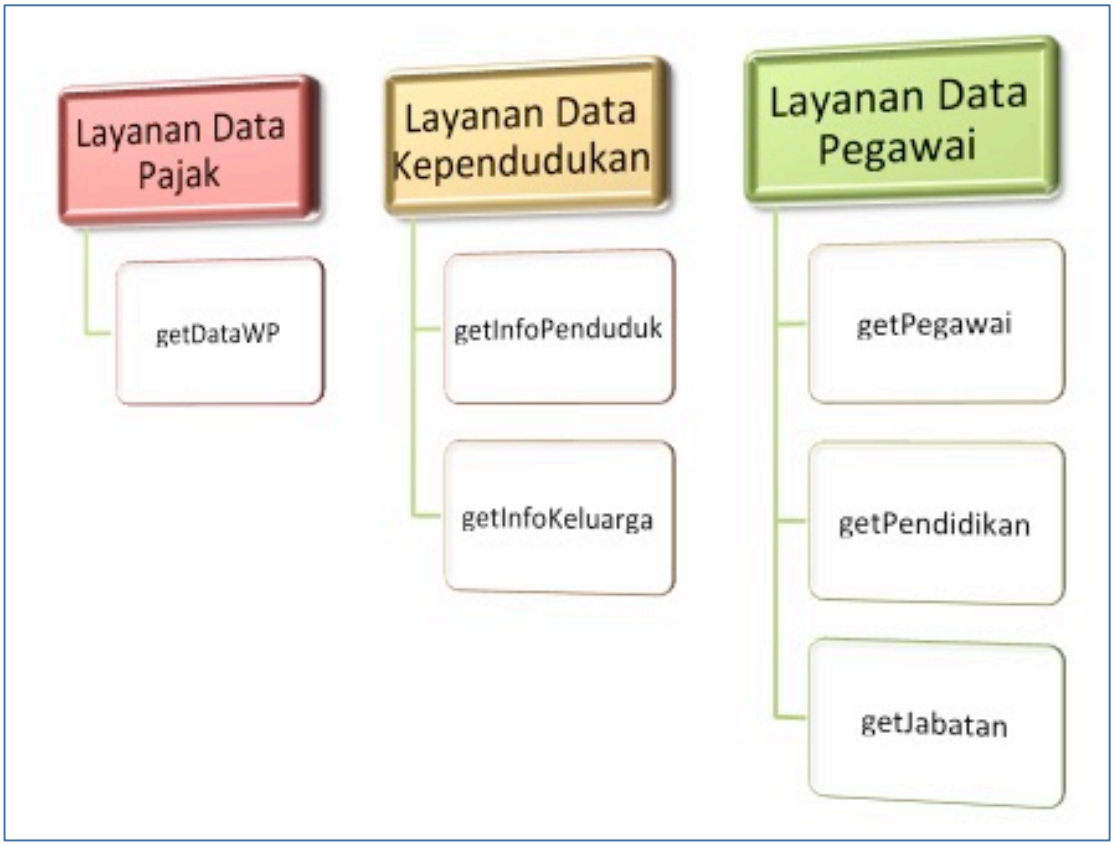


Gambar 3.76 Use Case Publisher MANTRA

3.4.4.1 Publikasi Layanan

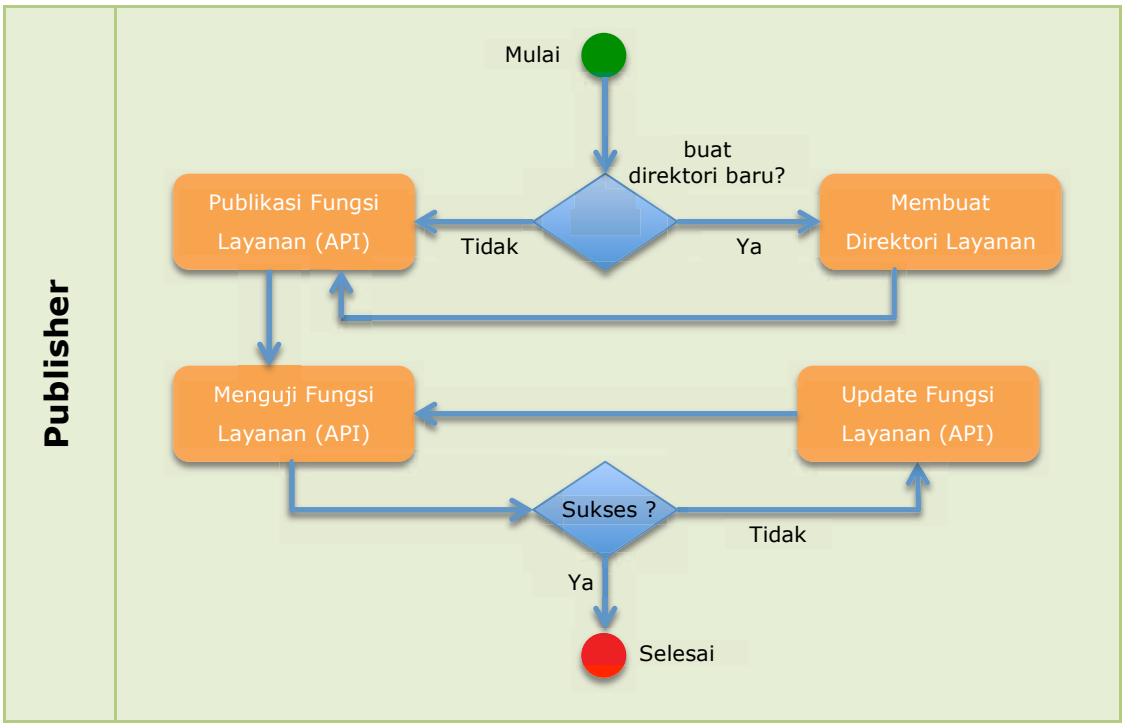
Layanan dalam Aplikasi MANTRA direpresentasikan dalam bentuk hierarki. Struktur hierarki layanan dapat dilihat dalam Gambar 3.77. Sebagaimana ditampilkan dalam gambar tersebut, setiap direktori layanan memiliki satu atau lebih fungsi (API). Layanan Info Penduduk dan Layanan Info Pegawai dalam gambar tersebut dapat dianalogikan sebagai direktori layanan, sementara itu fungsi (API) adalah sumber daya yang akan dibagikan kepada para pengguna layanan (requester).

Layanan terdiri dari direktori layanan dan fungsi layanan (API). Langkah pertama yang dilakukan publisher sebelum membuat fungsi layanan (API) adalah mendefinisikan direktori layanan.



Gambar 3.77 Struktur Layanan MANTRA

Untuk mempublikasikan layanan (Web-API), publisher harus mendefinisikan direktori layanan dan mendefinisikan fungsi layanan (API). Publisher dapat memilih menggunakan direktori layanan yang sudah ada atau membuat direktori layanan baru untuk mempublikasikan fungsi (API) yang akan dibuat. Jika direktori layanan belum dibuat, maka langkah pertama yang harus dilakukan oleh publisher adalah membuat direktori layanan. Alur kerja publikasi layanan dalam Aplikasi MANTRA dapat dilihat pada Gambar 3.78.



Gambar 3.78 Alur Kerja Publikasi Layanan

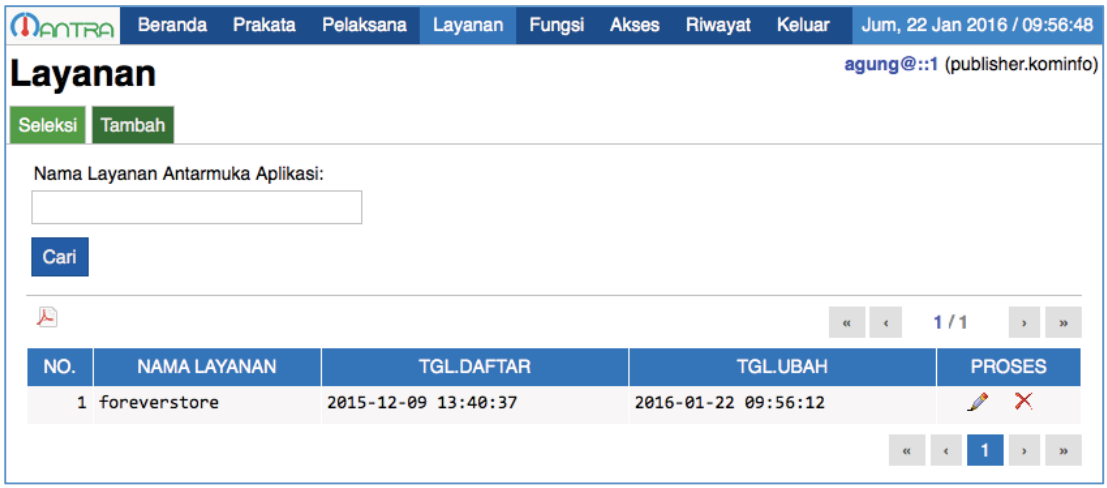
3.4.4.1.1 Pengelolaan Direktori Layanan

Aktivitas pengelolaan direktori layanan meliputi pembuatan direktori layanan, update nama direktori layanan, dan penghapusan direktori layanan. Sebagaimana dijelaskan dalam subbab 3.4.4.1 layanan tersusun secara hierarki, dimana satu atau lebih fungsi layanan (API) merupakan bagian dari direktori layanan. Oleh karena itu sebelum mempublikasikan fungsi layanan, publisher harus memastikan tersedianya direktori layanan.

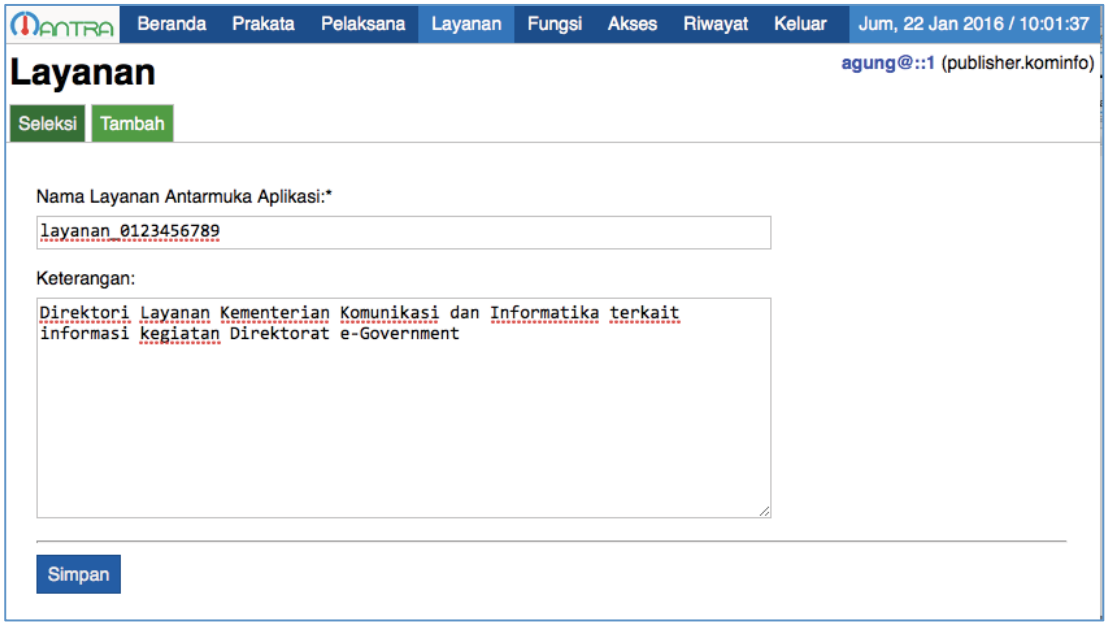
Struktur layanan berupa hierarki, penghapusan direktori layanan secara otomatis akan menghapus semua fungsi layanan (API) yang dipublikasikan di dalam direktori tersebut.

Berikut adalah langkah-langkah pembuatan direktori layanan dalam Aplikasi MANTRA:

1. Akses menu **Layanan**, pada halaman layanan ditampilkan daftar direktori layanan yang telah didaftarkan (Gambar 3.79).
2. Akses submenu **Tambah**, melalui submenu tersebut publisher diarahkan menuju halaman tambah layanan (Gambar 3.80).
3. Isi kolom **Nama Layanan Antarmuka Aplikasi** (harus diisi) dan kolom **Keterangan** (optional). Pengisian Nama Layanan hanya dibatasi dengan kombinasi huruf kecil (a-z), angka (0-9), dan karakter "_".
4. Tekan tombol Simpan untuk menambahkan direktori layanan tersebut.



Gambar 3.79 Halaman Layanan Provider

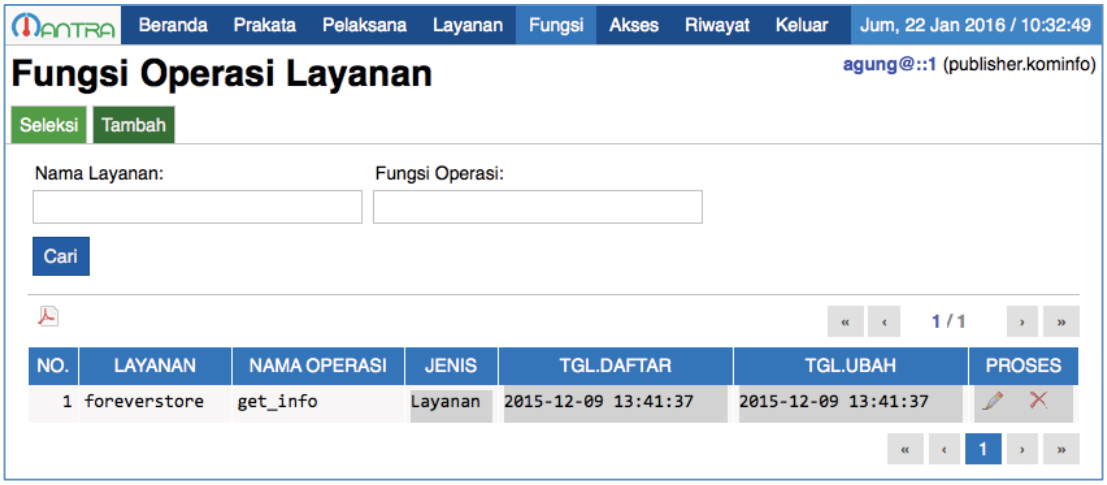


Gambar 3.80 Halaman Tambah Layanan Provider

Selain untuk menambahkan direktori layanan baru, pada halaman layanan juga tersedia fitur untuk melakukan update direktori layanan dan menghapus direktori layanan.

3.4.4.1.2 Publikasi dan Pengelolaan Fungsi Layanan

Fitur pengelolaan fungsi layanan dalam Aplikasi MANTRA dapat diakses oleh publisher melalui menu **Fungsi**. Setelah menu fungsi diakses, publisher akan diarahkan menuju halaman fungsi layanan (Gambar 3.81). Pada halaman tersebut seluruh fungsi layanan yang dipublikasikan oleh publisher akan ditampilkan. Selain itu fitur untuk menambahkan fungsi layanan baru, melakukan update fungsi layanan dan menghapus fungsi layanan juga disediakan pada halaman fungsi layanan.



Gambar 3.81 Halaman Fungsi Layanan Provider

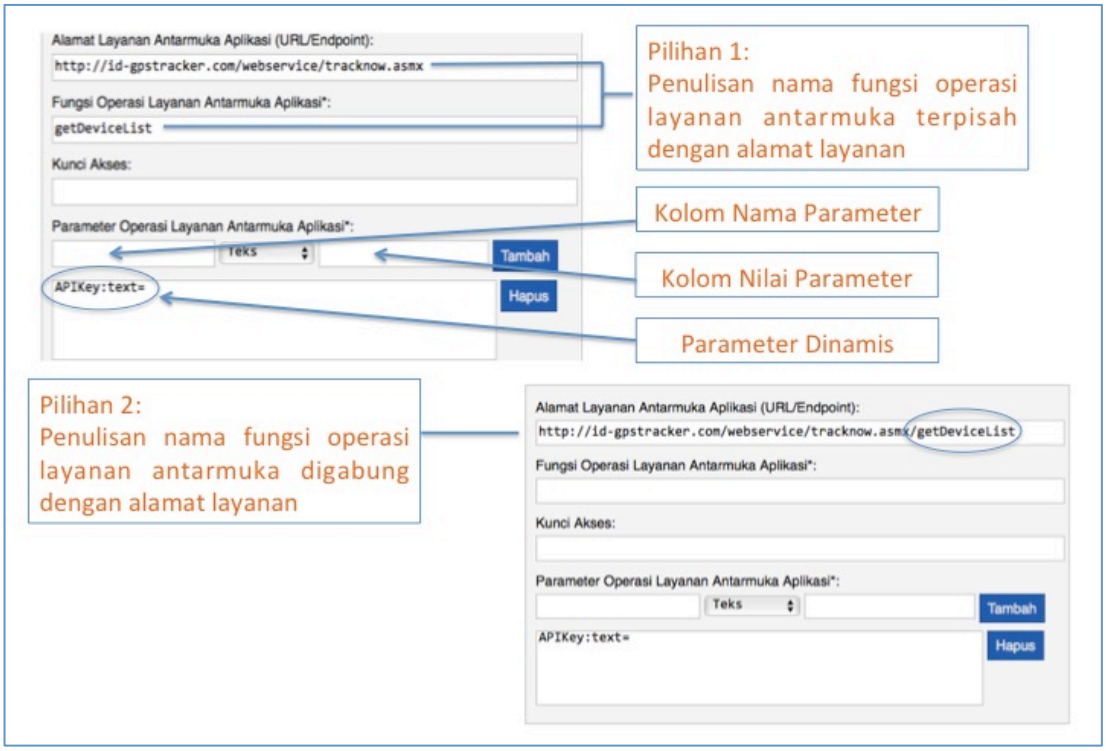
Aktivitas pembuatan fungsi layanan yang dilakukan oleh publisher adalah pemanfaatan fungsi layanan yang sudah dibuat sebelumnya oleh penyedia layanan untuk dipublikasikan sebagai satu fungsi layanan baru di Aplikasi MANTRA. Fungsi layanan baru tersebut dikategorikan sebagai Layanan Proxy (*Proxy Services*) yaitu layanan yang menjembatani akses end-user terhadap layanan lain yang disediakan oleh penyedia layanan.

Langkah-langkah pembuatan fungsi layanan di Aplikasi MANTRA adalah sebagai berikut:

1. Akses menu **Fungsi**, pada halaman fungsi ditampilkan daftar fungsi layanan yang telah didaftarkan (Gambar 3.81).
2. Akses submenu **Tambah**, melalui submenu tersebut publisher diarahkan menuju halaman tambah fungsi layanan.
3. Pada kolom **Layanan** pilih salah satu nama direktori layanan yang menjadi induk dari fungsi layanan yang akan dibuat.
4. Isi kolom **Nama Operasi** dengan nama fungsi yang akan dibuat. Pengisian Nama Operasi hanya dibatasi dengan kombinasi huruf kecil (a-z), angka (0-9), dan karakter "_".
5. Tandai (Klik) keterangan **Akses Terbatas** jika tidak menghendaki deskripsi fungsi layanan ditampilkan pada halaman awal MANTRA.

6. Tentukan isi kolom **Jenis Operasi Layanan Antarmuka Aplikasi**. Jenis operasi merupakan metode akses fungsi layanan internal atau eksternal yang akan dipublikasikan. Untuk layanan internal menggunakan jenis operasi "REST/MANTRA". Sedangkan untuk layanan eksternal pilihan jenis operasi bervariasi, antara lain "REST", "RESTFULL", "SOAP", "HTTP/GET", atau "HTTP/POST".
7. Isi kolom **Alamat Layanan Antarmuka Aplikasi** dengan alamat URL layanan yang akan dipublikasikan. Untuk layanan internal kolom tersebut diisi dengan alamat akses direktori layanan.
8. Isi kolom **Fungsi Operasi Layanan Antarmuka Aplikasi**. Untuk publikasi layanan internal dan layanan eksternal dengan metode SOAP kolom tersebut diisi dengan nama fungsi layanan. Untuk layanan eksternal dengan metode akses REST, HTTP/GET, atau HTTP/POST nama fungsi operasi layanan antarmuka dapat dikosongkan, penulisan nama fungsi tersebut dapat digabungkan dengan alamat layanan antarmuka aplikasi seperti ditunjukkan dalam Gambar 3.82.
9. Khusus untuk publikasi layanan internal, isi kolom **Kunci Akses** dengan kunci akses fungsi layanan yang dimiliki oleh publisher. Kunci akses publisher terdiri dari 10 digit kombinasi huruf (a-z) dan angka (0-9), contoh: *bpfqe773i6*. Kunci akses tersebut didapatkan oleh publisher dengan mengajukan permintaan akses layanan kepada penyedian layanan di MANTRA API melalui akun requester (lihat Gambar 3.75). Panduan permintaan akses layanan tersebut dijelaskan dalam subbab 3.4.5.1.

Format alamat akses layanan internal:
http://{alamat-aplikasi}/{api|xml|json}/{id-instansi-penyedia}/{direktori-layanan}/{fungsi-layanan}/{kunci-akses}/{parameter}



Gambar 3.82 Pilihan Penulisan Nama Fungsi Layanan Eksternal

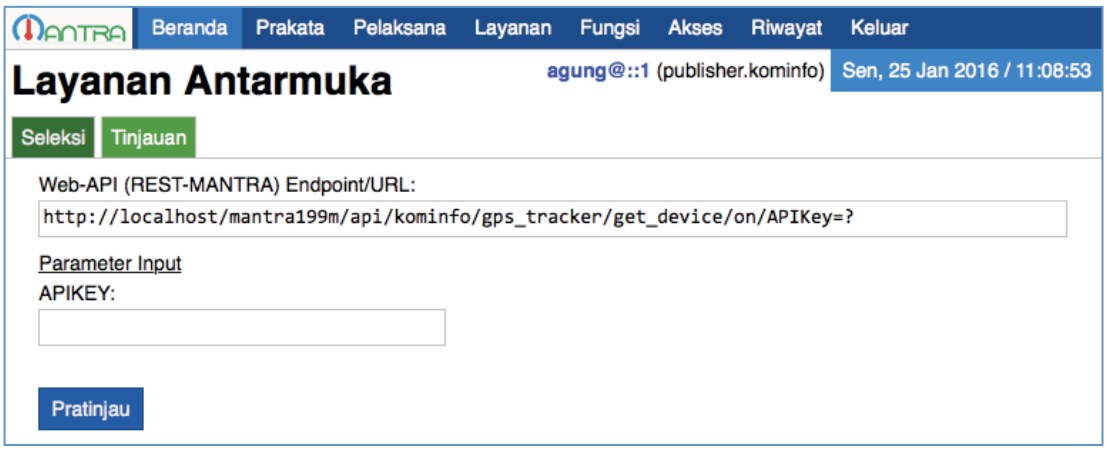
10. Tentukan **Parameter Operasi Program** (jika ada), isikan nama parameter pada kolom nama, pilih tipe parameter (Teks/Numerik). Di sebelah kolom tipe parameter terdapat kolom nilai parameter. Kosongkan nilai parameter untuk membuat parameter dinamis.
11. Isi kolom **Keterangan** dengan deskripsi fungsi (API) yang dibuat. Deskripsi meliputi manfaat dari fungsi tersebut, elemen data yang dibagikan, atau keterangan parameter yang dipersyaratkan.
12. Tekan tombol Simpan untuk menambahkan fungsi (API) tersebut.
13. Uji fungsi (API) melalui menu **Beranda**, klik ikon Tinjauan pada kolom "**PROSES**". Setelah tampil halaman tinjauan isi parameter yang dibutuhkan (jika ada), kemudian klik tombol Pratinjau. Output fungsi layanan yang muncul pada halaman tinjauan ditampilkan dalam format tabel, XML, JSON, array, dan serial. Output dengan Status "1" menunjukkan perintah operasi

dalam fungsi (API) tersebut berhasil dijalankan. Sedangkan output dengan status "0" menunjukkan perintah operasi pada fungsi (API) tidak berhasil dijalankan. Daftar status dan kode output MANTRA selengkapnya dapat dilihat dalam Lampiran 1 dan Lampiran 2.

14. Jika fungsi layanan tidak berhasil dijalankan, periksa kembali fungsi layanan melalui menu **Fungsi** dan klik ikon ubah data dalam kolom **PROSES**. Simpan kembali perubahan yang dilakukan dan lakukan kembali uji fungsi sebagaimana ditunjukkan pada langkah 13 di atas.

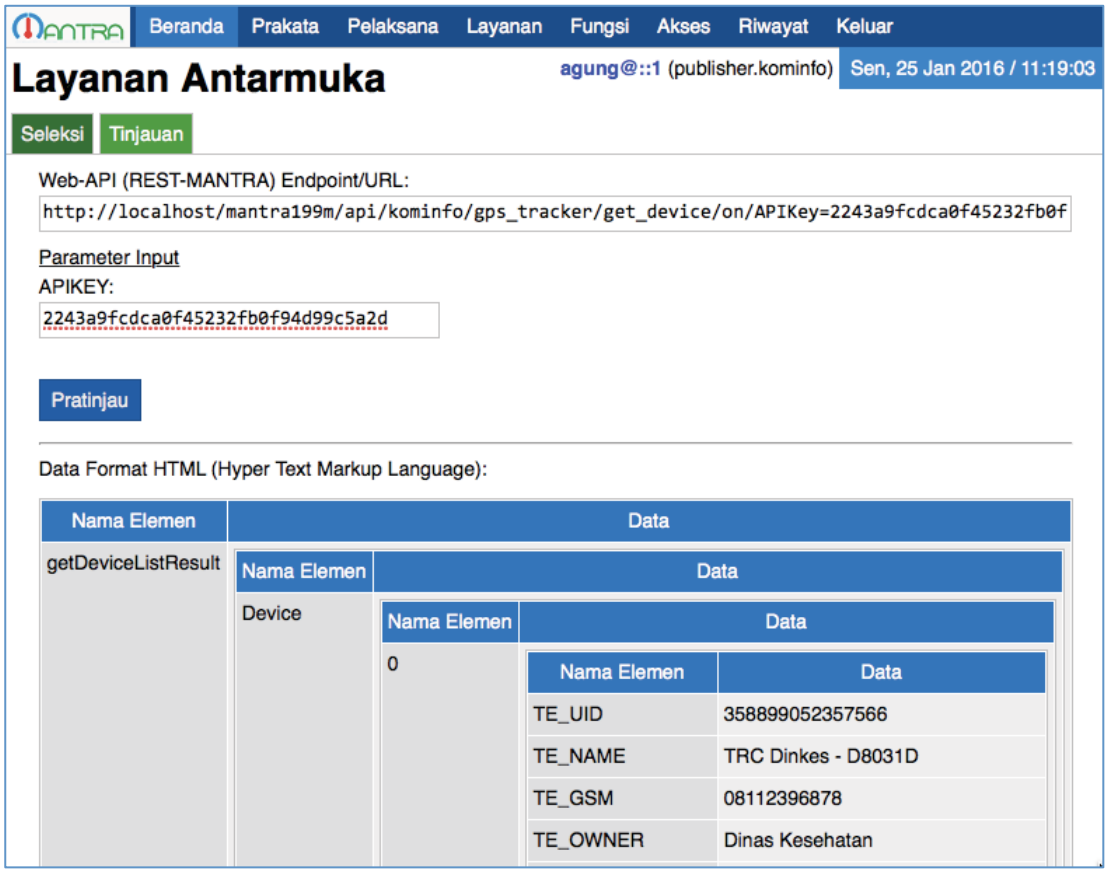
3.4.4.2 Prantinjau Layanan

Melalui fungsi pratinjau layanan, publisher dapat melakukan uji fungsi untuk mengetahui apakah layanan berhasil dijalankan atau terkendala. Fungsi Pratinjau Layanan dapat diakses melalui menu "**Beranda**". Pada halaman tersebut akan ditampilkan tabel yang berisi nama-nama fungsi layanan yang dimiliki oleh instansi dimana publisher terdaftar. Untuk melakukan uji fungsi layanan tekan ikon "Tinjauan" pada kolom "**PROSES**" di Halaman Beranda.

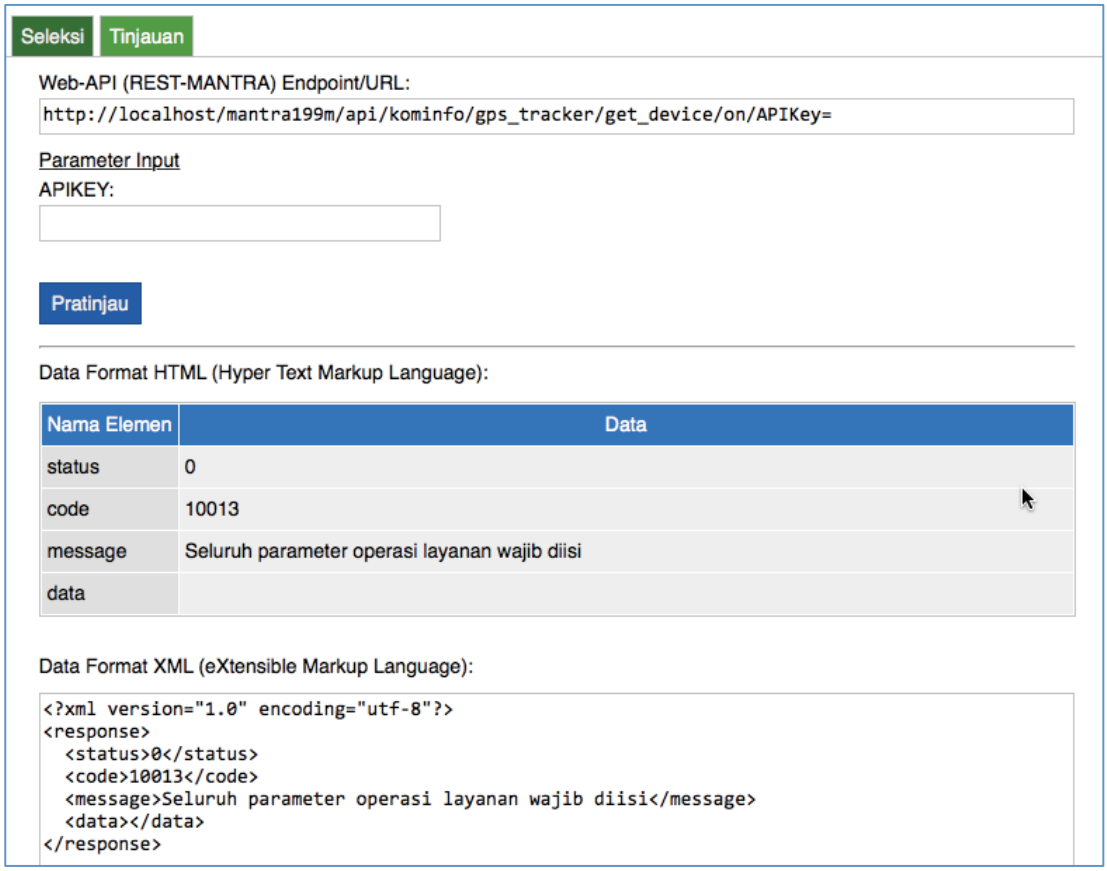


Gambar 3.83 Halaman Tinjauan Layanan Publisher

Gambar 3.83 di atas merupakan tampilan fitur tinjauan layanan MANTRA. Publisher harus mengisi parameter layanan (jika ada) sebelum melakukan pratinjau. Pada contoh gambar di atas parameter yang harus diisi adalah "APIKey". Gambar 3.84 merupakan tampilan tinjauan layanan yang berfungsi dengan baik, sedangkan Gambar 3.85 merupakan tampilan tinjauan layanan yang mengalami kendala akses. Adapun daftar kode output MANTRA dapat dilihat dalam Lampiran 1 dan Lampiran 2.



Gambar 3.84 Fungsi Layanan Berjalan dengan Baik



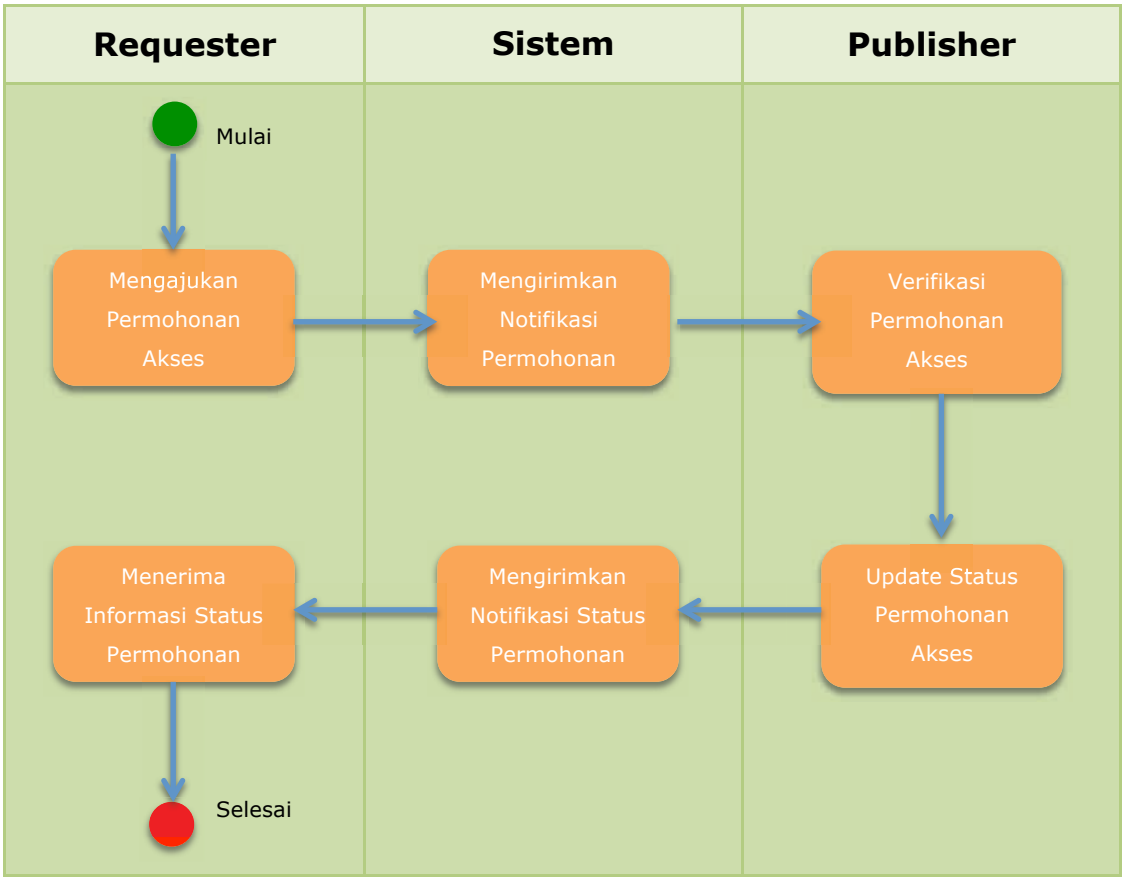
Gambar 3.85 Fungsi Layanan Terkendala

3.4.4.3 Pengelolaan Akses Layanan

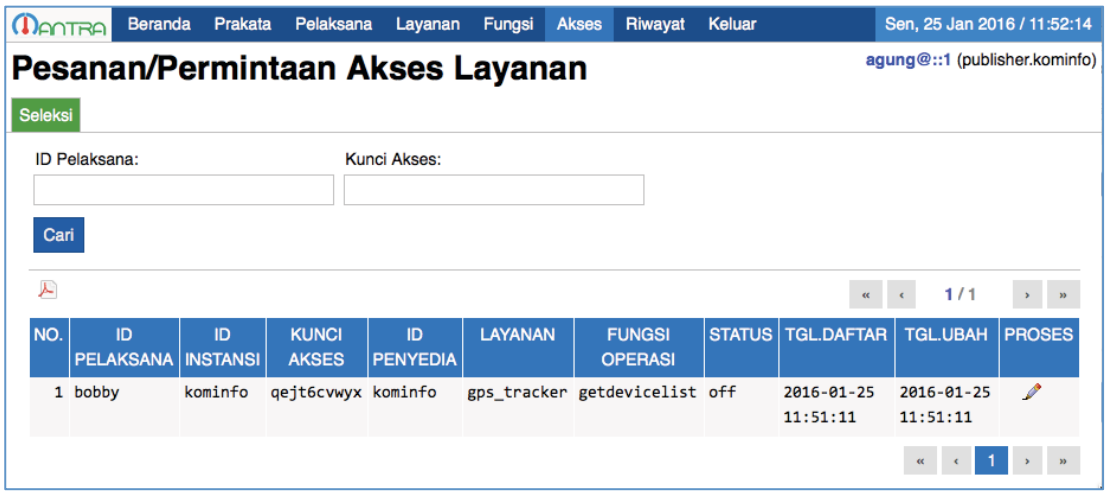
Pengelolaan akses layanan adalah aktivitas publisher dalam rangka membuka atau menutup akses pengguna layanan (requester) terhadap layanan yang telah dipublikasikan. Perlu diketahui bahwa pengguna layanan (requester) tidak dapat memanfaatkan layanan sebelum mendapat persetujuan dari publisher. Gambar 3.86 merupakan alur permohonan akses layanan dari pengguna layanan kepada publisher.

Setiap permohonan akses layanan diajukan oleh requester, sistem akan mengirimkan notifikasi melalui email kepada publisher. Permohonan-permohonan tersebut ditampilkan pada Halaman Permintaan Akses Layanan yang diakses melalui menu "**Akses**". Status awal permohonan yang baru ditambahkan adalah "**off**".

Untuk membuka akses requester terhadap layanan, publisher harus mengubah status permohonan tersebut menjadi "**on**".



Gambar 3.86 Alur Proses Permohonan Akses Layanan



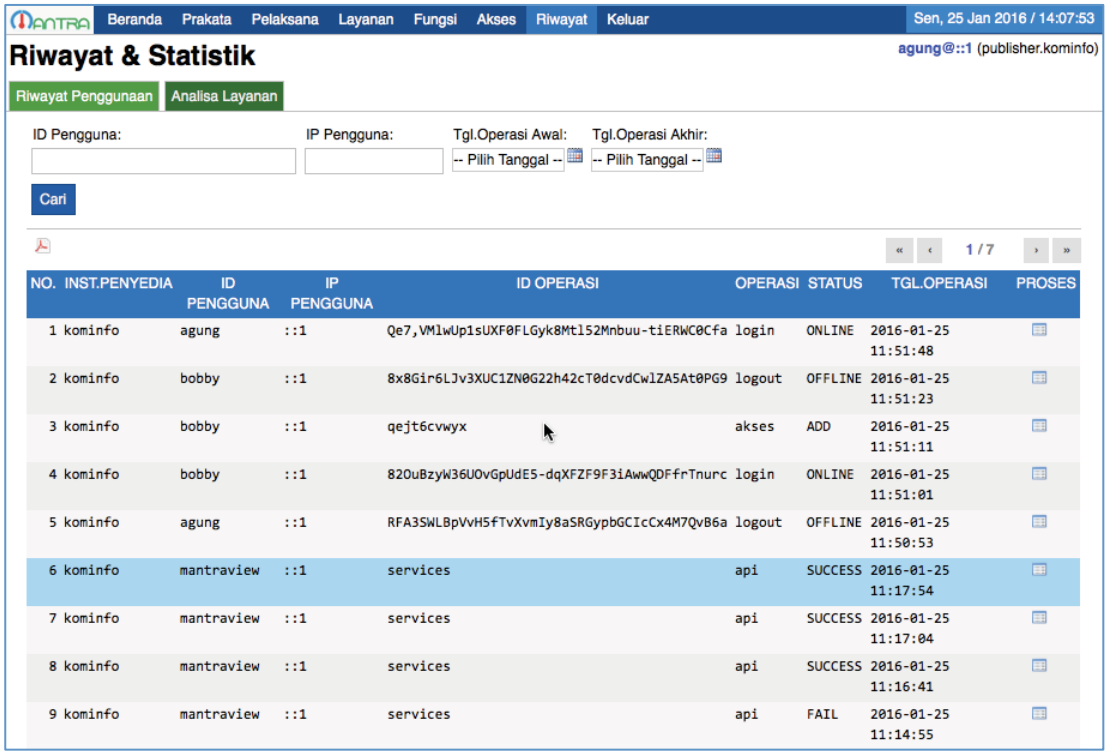
Gambar 3.87 Halaman Permintaan Akses Layanan

Langkah-langkah untuk mengubah status permohonan akses layanan adalah sebagai berikut:

1. Klik ikon ubah data  pada Halaman Permintaan Akses Layanan (Gambar 3.87).
2. Pada halaman ubah data update status permintaan menjadi off atau on. Update status menjadi off untuk menutup akses requester terhadap layanan atau update status menjadi on untuk membuka akses requester terhadap layanan.
3. Klik tombol Simpan untuk menyimpan perubahan status tersebut.

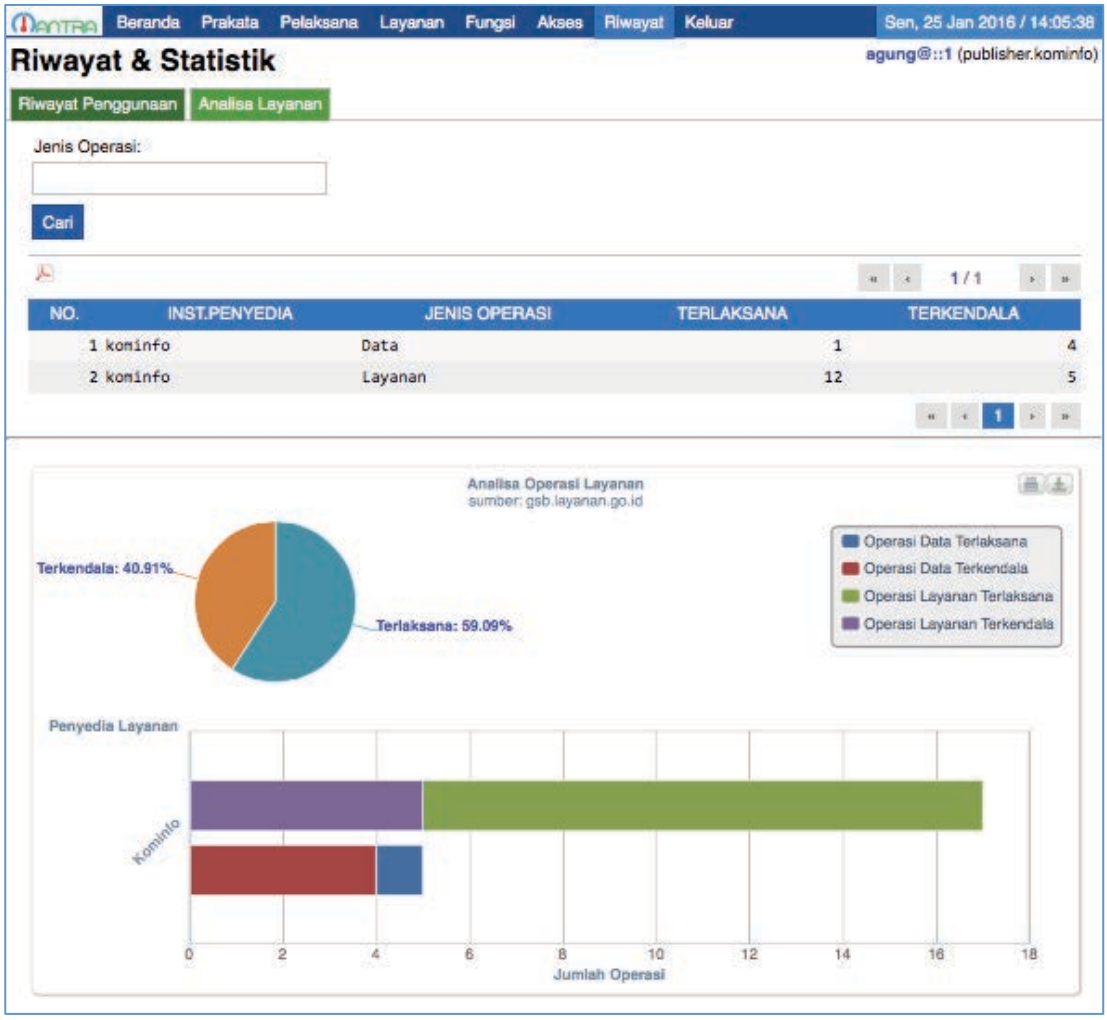
3.4.4.4 Monitoring Riwayat Aplikasi

Modul ini digunakan untuk melakukan monitoring aktivitas pengguna dan aktivitas pemanfaatan layanan yang dimiliki oleh instansi dimana publisher didaftarkan. Aktivitas-aktivitas tersebut direkam dan ditampilkan pada halaman riwayat dan statistik. Halaman tersebut dapat diakses oleh publisher melalui menu "**Riwayat**". Ikon  (Rincian Penggunaan) dalam kolom **PROSES** pada halaman riwayat digunakan untuk melihat detail rekaman aktivitas pemanfaatan Aplikasi MANTRA.



Gambar 3.88 Halaman Riwayat Publisher

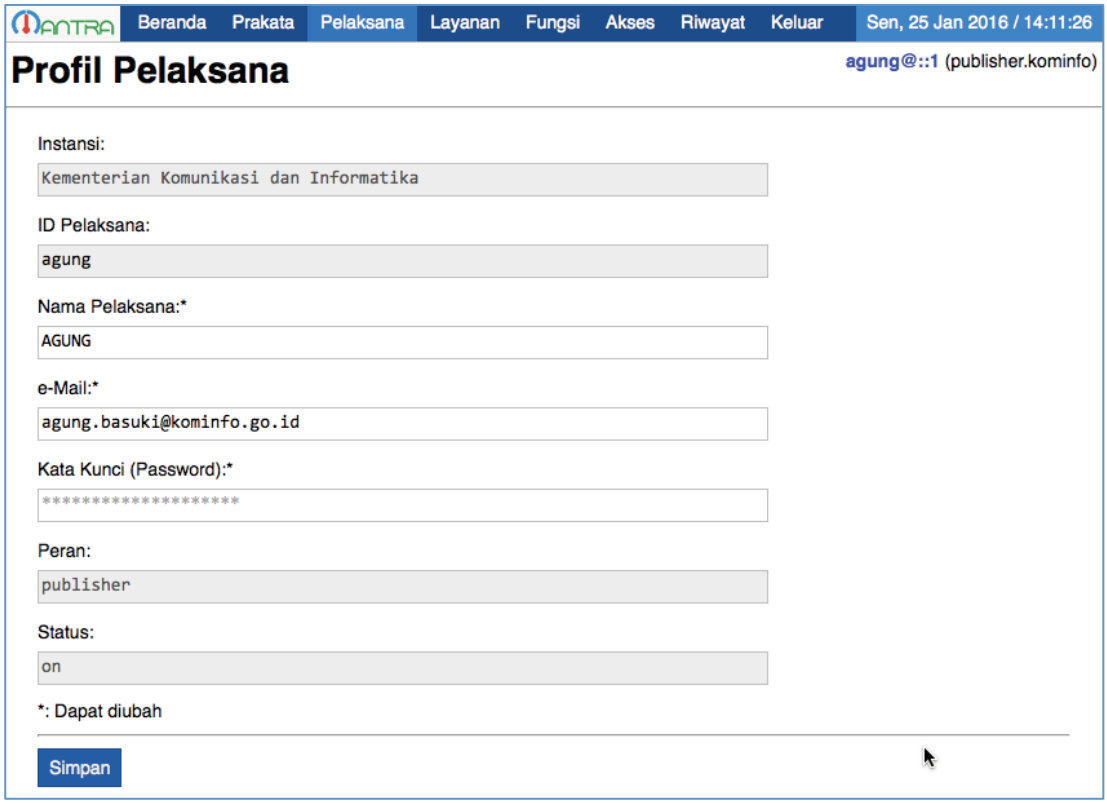
Pada halaman riwayat tersedia fitur untuk melihat statistik pemanfaatan layanan yang ditampilkan dalam bentuk tabel dan grafik. Publisher dapat melihat informasi statistik tersebut dengan mengakses submenu "**Analisa Layanan**". Informasi tersebut dikelompokkan berdasarkan jenis operasi layanan yang dimiliki oleh instansi yang bersangkutan.



Gambar 3.89 Halaman Riwayat dan Statistik

3.4.4.5 Update Profil Provider

Publisher dapat mengubah profil pribadinya dengan mengakses menu "**Pelaksana**". Informasi-informasi yang dapat diubah antara lain nama pelaksana, e-mail, dan kata kunci (password). Jika ingin melakukan perubahan ID Pengguna, publisher harus mengajukan permohonan ubah data kepada administrator sebagaimana dijelaskan dalam subbab 3.4.

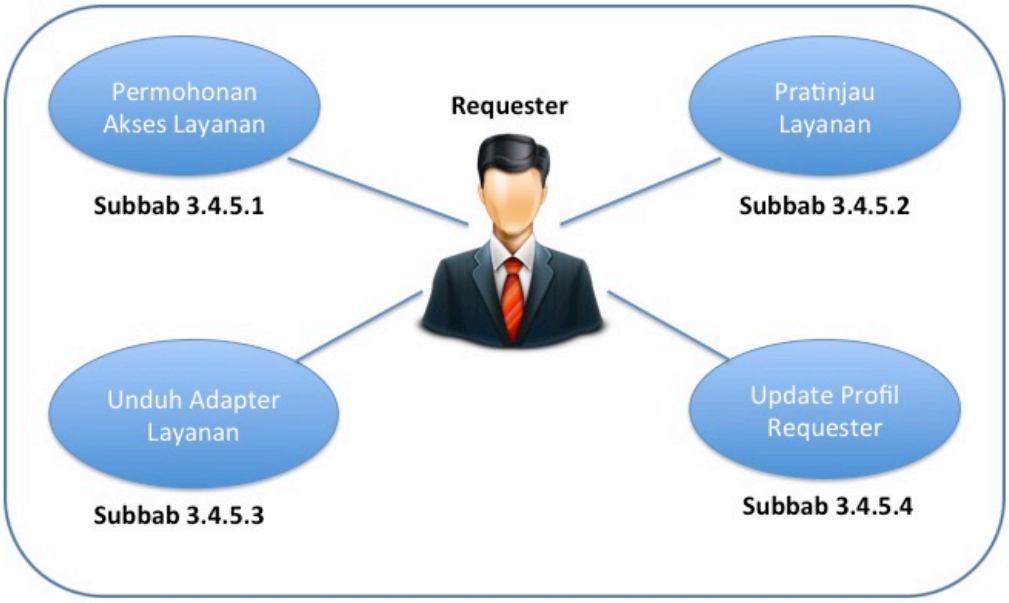


Gambar 3.90 Halaman Ubah Profil Publisher

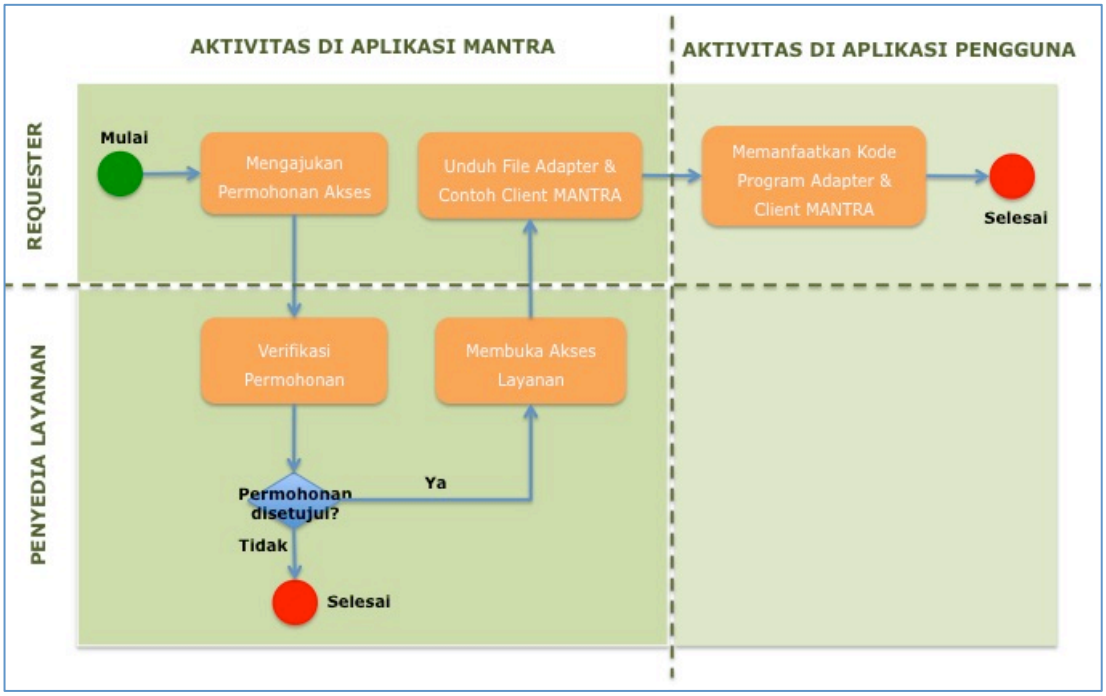
3.4.5 Panduan Requester

Requester adalah pengguna/pemanfaat layanan yang dibuat oleh provider atau layanan yang dipublikasi oleh publisher. Fitur yang disediakan untuk kelompok pengguna Requester dapat dilihat dalam Gambar 3.91. Fitur-fitur tersebut disediakan untuk mempermudah requester dalam rangka menemukan layanan dan mendapatkan hak akses terhadap layanan yang terdaftar di Aplikasi MANTRA. Alur kerja pemanfaatan layanan pertukaran data dapat dilihat pada Gambar 3.92.

Pemanfaatan layanan antar aplikasi dilakukan menggunakan adapter yang diunduh dari Aplikasi MANTRA. Pemanfaat layanan perlu mengintegrasikan kode program pada berkas adapter tersebut ke dalam kode program aplikasi pengguna.



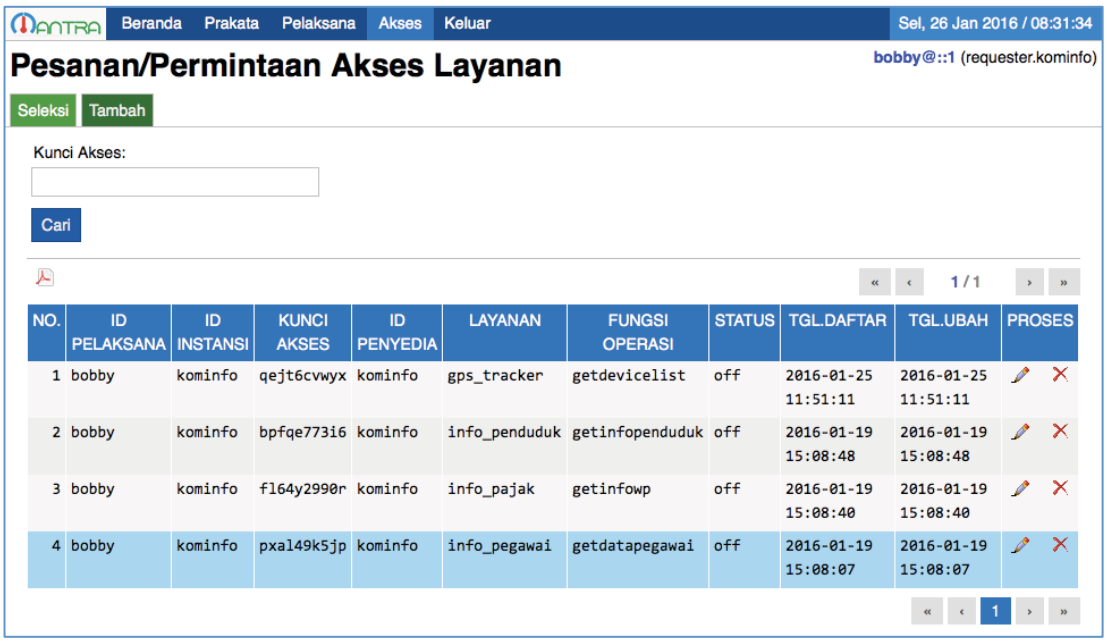
Gambar 3.91 Use Case Pengguna Requester



Gambar 3.92 Alur Kerja Pemanfaatan Layanan

3.4.5.1 Permohonan Akses Layanan

Fitur Permohonan Akses Layanan digunakan oleh requester untuk menambahkan permohonan akses untuk layanan yang disediakan oleh penyedia (provider/publisher). Fitur tersebut dapat digunakan oleh requester dengan mengakses halaman permintaan akses layanan melalui menu **Akses**. Pada halaman tersebut, requester dapat melihat daftar permohonan akses fungsi layanan yang sudah ditambahkan (Gambar 3.93).



Gambar 3.93 Halaman Permintaan Akses Layanan

Adapun proses penambahan akses layanan dilakukan dengan langkah-langkah sebagai berikut:

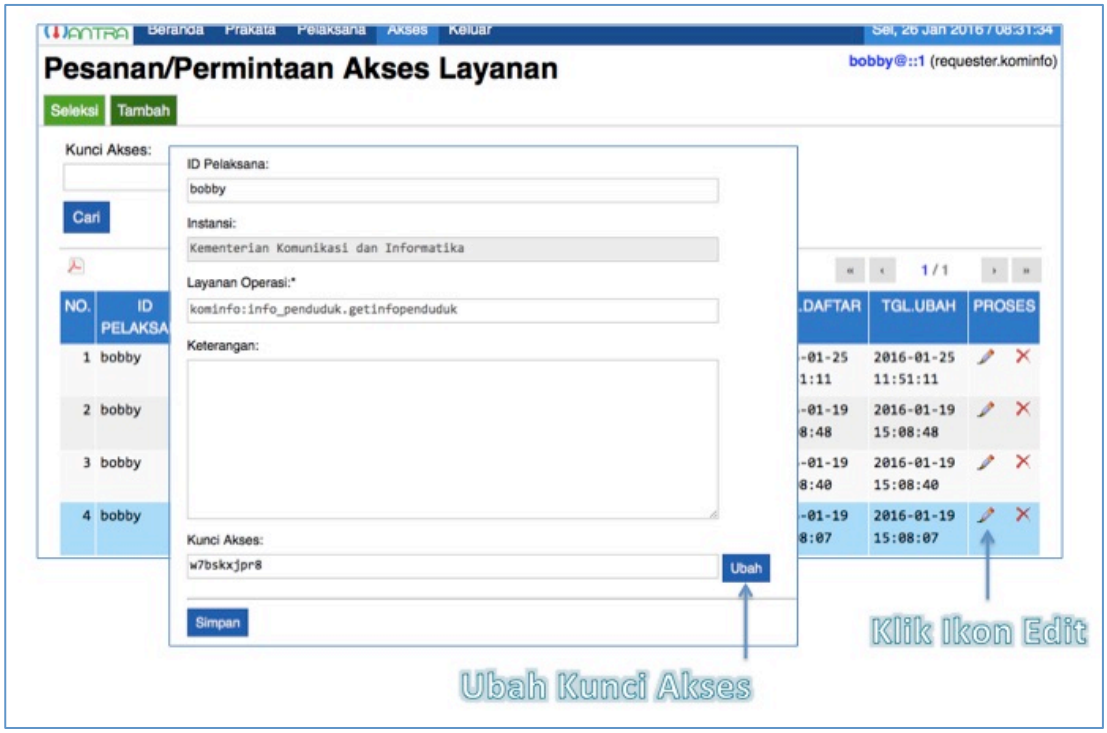
1. Tekan submenu **Tambah** pada Halaman Permintaan Akses Layanan.
2. Pada kolom **Layanan Operasi** ditampilkan daftar fungsi layanan yang ada di Aplikasi MANTRA. Pilih salah satu fungsi layanan yang ingin diakses. Format penulisan layanan pada kolom Layanan Operasi tersusun dari *{id-instansi}*:*{direktori-layanan}*.*{fungsi-layanan}*.

Contoh kominfo:info_pajak.getinfowp, layanan tersebut memiliki nama fungsi layanan getinfowp, nama direktori layanan info_pajak, dan nama instansi penyedia kominfo.

3. Tekan tombol Simpan untuk mengirimkan permintaan akses layanan kepada penyedia layanan.

Informasi permohonan akses yang baru ditambahkan secara otomatis ditampilkan dalam Halaman Permintaan Akses Layanan dengan status **off**. Jika permintaan akses tersebut disetujui oleh penyedia layanan, maka penyedia akan mengubah status tersebut menjadi **on.** Setelah memastikan status akses layanan on, requester dapat mengakses menu Beranda untuk melakukan pratinjau layanan atau mengunduh berkas adapter layanan MANTRA.

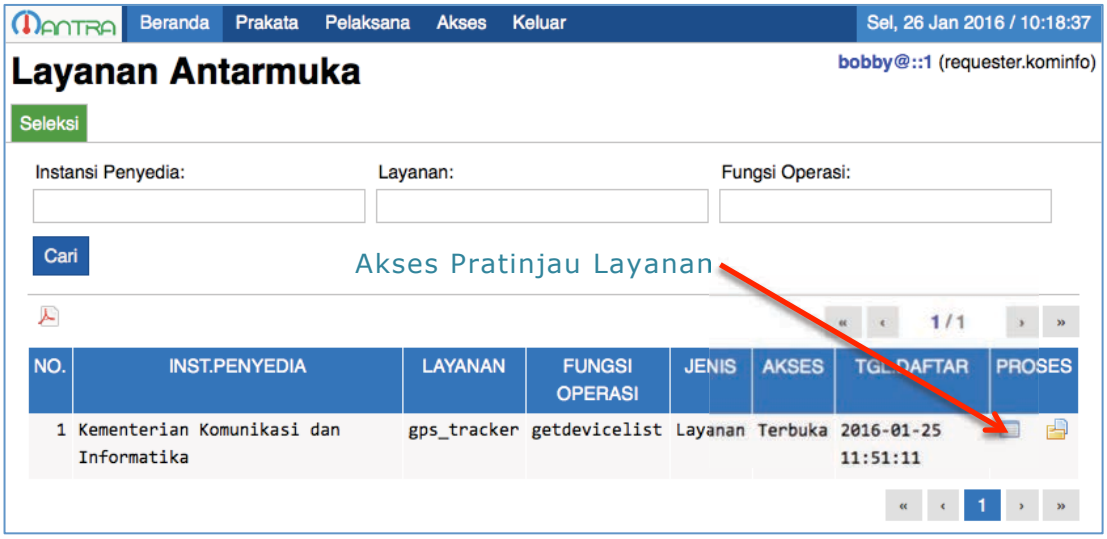
Selain tersedia fitur untuk menambahkan permohonan akses, pada halaman permintaan akses layanan juga tersedia fitur untuk membatalkan permohonan akses layanan (Ikon ) dan fitur untuk mengubah kunci akses layanan (Ikon ). Kunci akses diubah dengan menekan tombol Ubah pada halaman edit pesanan. Untuk menyimpan perubahan kunci akses tersebut requester harus menekan tombol Simpan yang terdapat pada halaman yang sama.



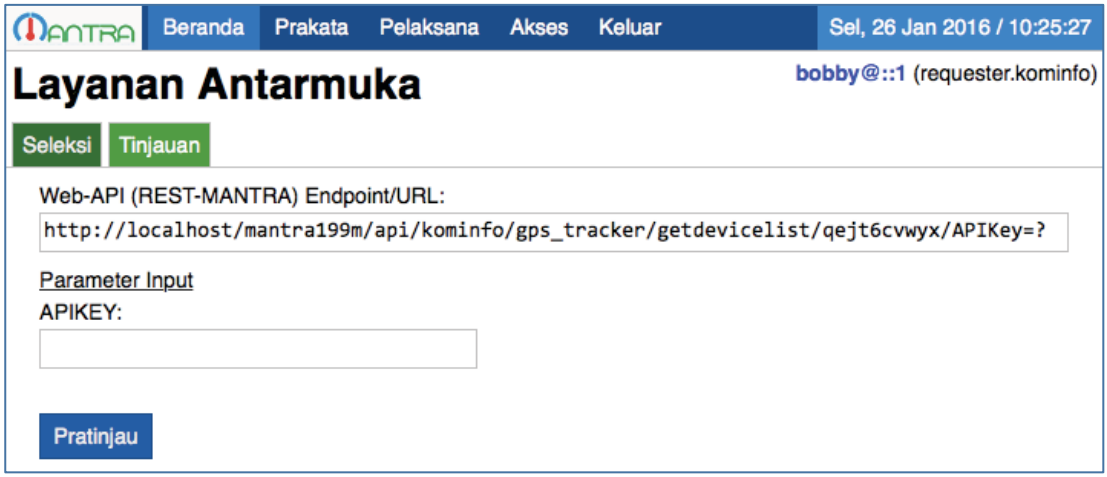
Gambar 3.94 Ubah Kunci Akses Layanan

3.4.5.2 Pratinjau Layanan

Melalui fungsi pratinjau layanan requester dapat melakukan uji fungsi layanan untuk mengetahui apakah fungsi layanan berjalan dengan baik atau terkendala. Fungsi Pratinjau Layanan dapat diakses melalui menu "**Beranda**". Pada halaman tersebut ditampilkan daftar fungsi layanan yang dapat dimanfaatkan oleh requester. Untuk melakukan uji fungsi layanan, requester harus menekan ikon "Tinjauan" dalam kolom "**PROSES**" pada halaman beranda.

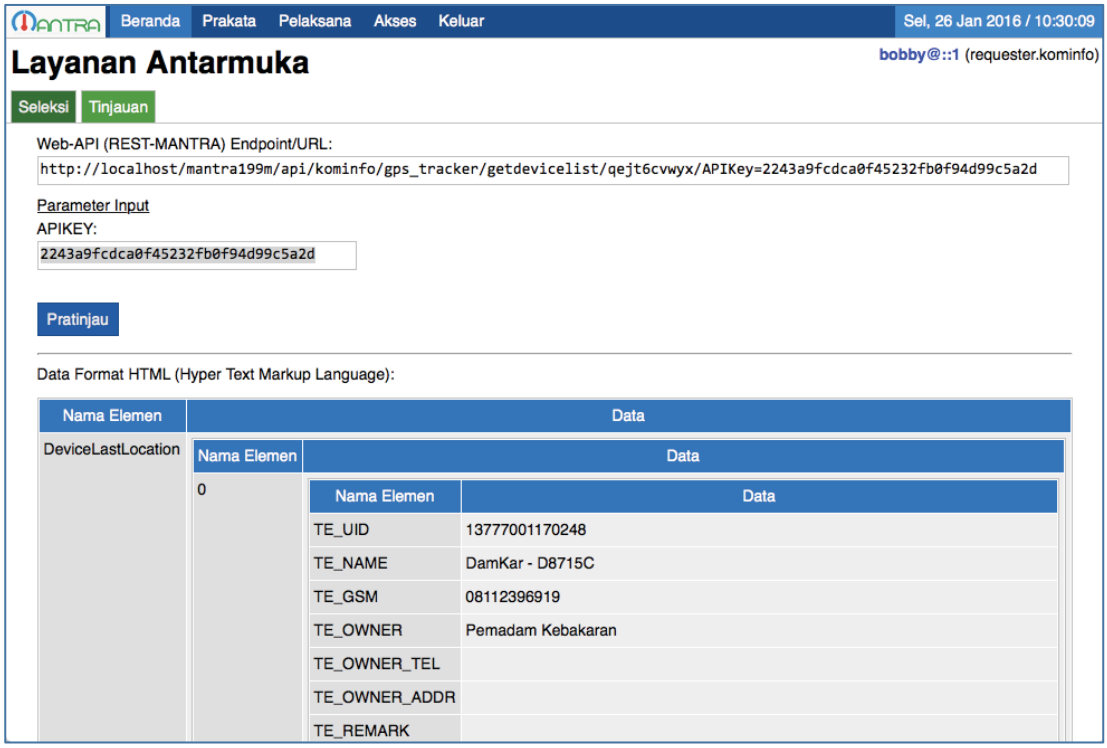


Gambar 3.95 Halaman Beranda Requester

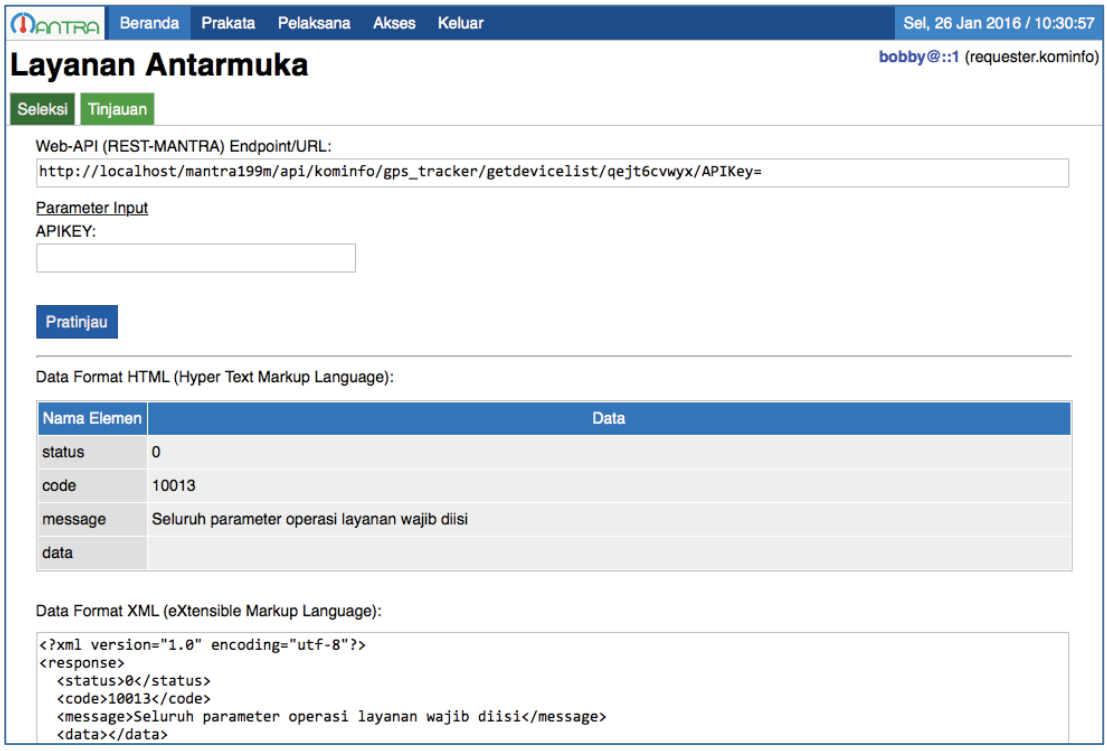


Gambar 3.96 Halaman Tinjauan Layanan Requester

Gambar 3.96 di atas merupakan tampilan fitur tinjauan layanan MANTRA. Requester harus mengisi parameter layanan (jika ada) sebelum melakukan pratinjau. Pada contoh gambar di atas parameter yang harus diisi adalah "APIKey". Gambar 3.97 merupakan tampilan tinjauan layanan yang berfungsi dengan baik, sedangkan Gambar 3.98 merupakan tampilan tinjauan layanan yang terkendala. Adapun daftar kode output MANTRA selengkapnya dapat dilihat dalam Lampiran 1 dan Lampiran 2.



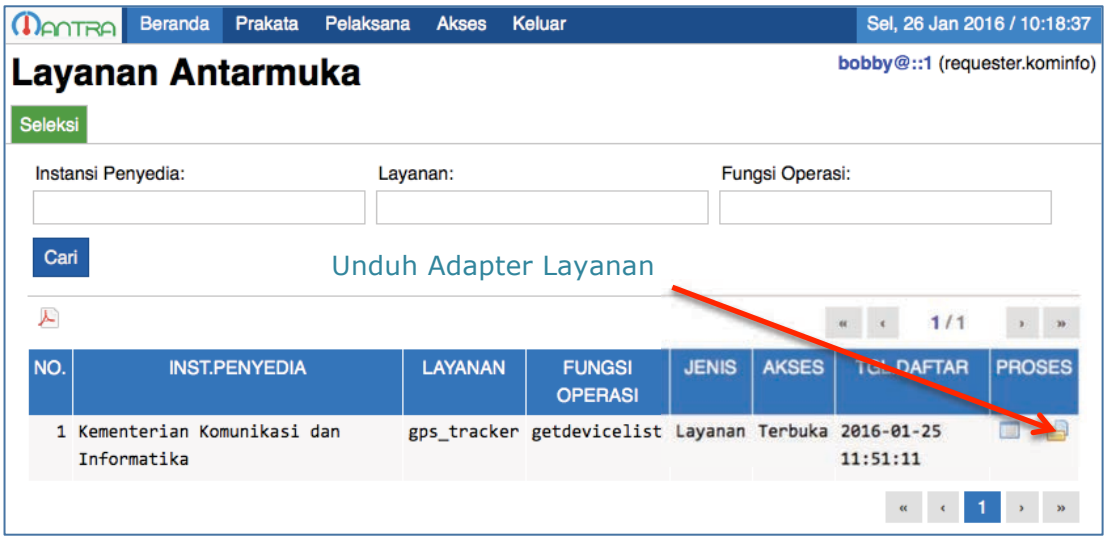
Gambar 3.97 Fungsi Layanan Berjalan dengan Baik



Gambar 3.98 Fungsi Layanan Terkendala

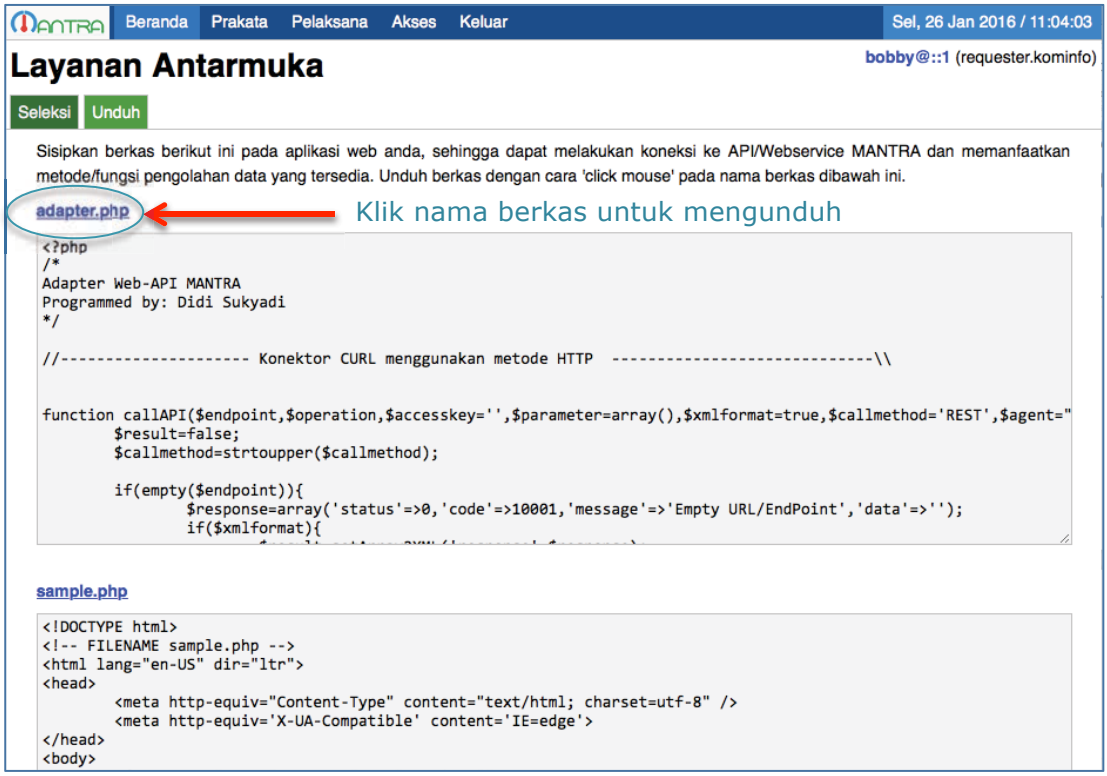
3.4.5.3 Unduh Adapter Layanan

Untuk memanfaatkan fungsi layanan yang telah diizinkan untuk diakses, requester dapat mengunduh berkas adapter yang tersedia dalam repositori adapter MANTRA. Kode program dalam berkas adapter tersebut digunakan pada aplikasi pengguna untuk melakukan pemanggilan fungsi layanan di MANTRA. Berkas adapter tersebut dapat diunduh melalui menu "**Beranda**" dengan menekan ikon unduh dalam kolom **PROSES**.

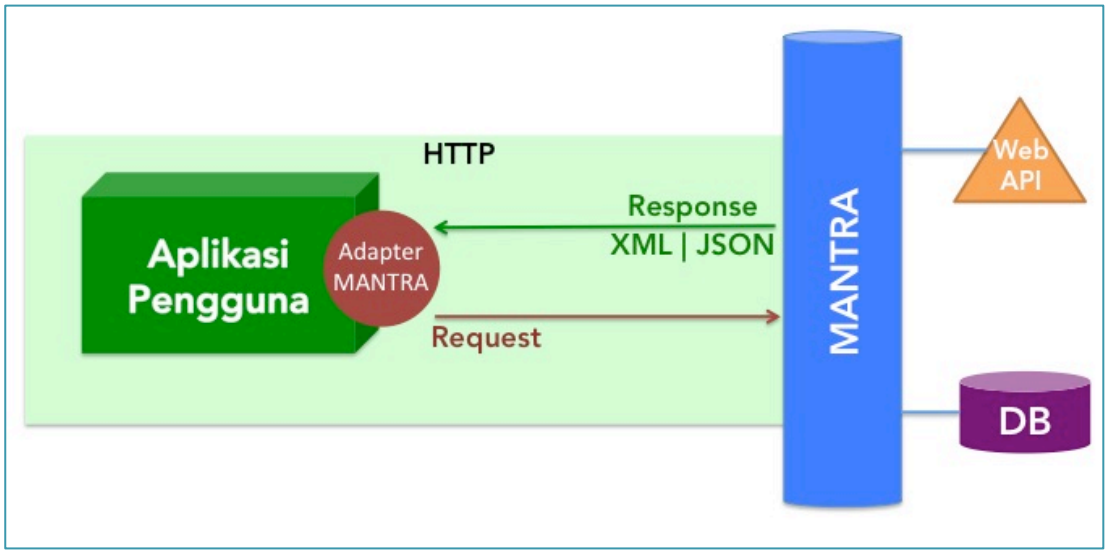


Gambar 3.99 Akses Halaman Unduh Adapter Layanan

Pada halaman unduh adapter layanan disediakan beberapa berkas yang dapat diunduh oleh requester. Berkas tersebut meliputi adapter layanan dan contoh aplikasi pengguna layanan. Saat ini dalam repositori adapter MANTRA telah tersedia adapter dan contoh aplikasi pengguna dengan pemrograman PHP dan Java. Pada pemograman PHP tersedia berkas adapter.php dan contoh aplikasi pengguna (sample.php atau sample-xml.php atau sample-json.php) yang dapat diunduh oleh requester. Sementara itu pada pemrograman Java, tersedia berkas sampleJSON.java yang dapat diunduh oleh requester.



Gambar 3.100 Halaman Unduh Adapter Layanan



Gambar 3.101 Pemanfaatan Adapter Layanan MANTRA

Sebagaimana ditampilkan dalam Gambar 3.101, adapter MANTRA berfungsi sebagai jembatan akses layanan dari Aplikasi Pengguna ke Aplikasi MANTRA. Adapter tersebut menerima respon dari Aplikasi MANTRA dan meneruskan respon ke Aplikasi Pengguna.

Tabel 3.1 di bawah ini merupakan penjabaran fungsi dari berkas-berkas yang tersedia dalam repositori Aplikasi MANTRA.

Tabel 3.1 Keterangan Berkas Adapter Layanan dan Contoh Aplikasi

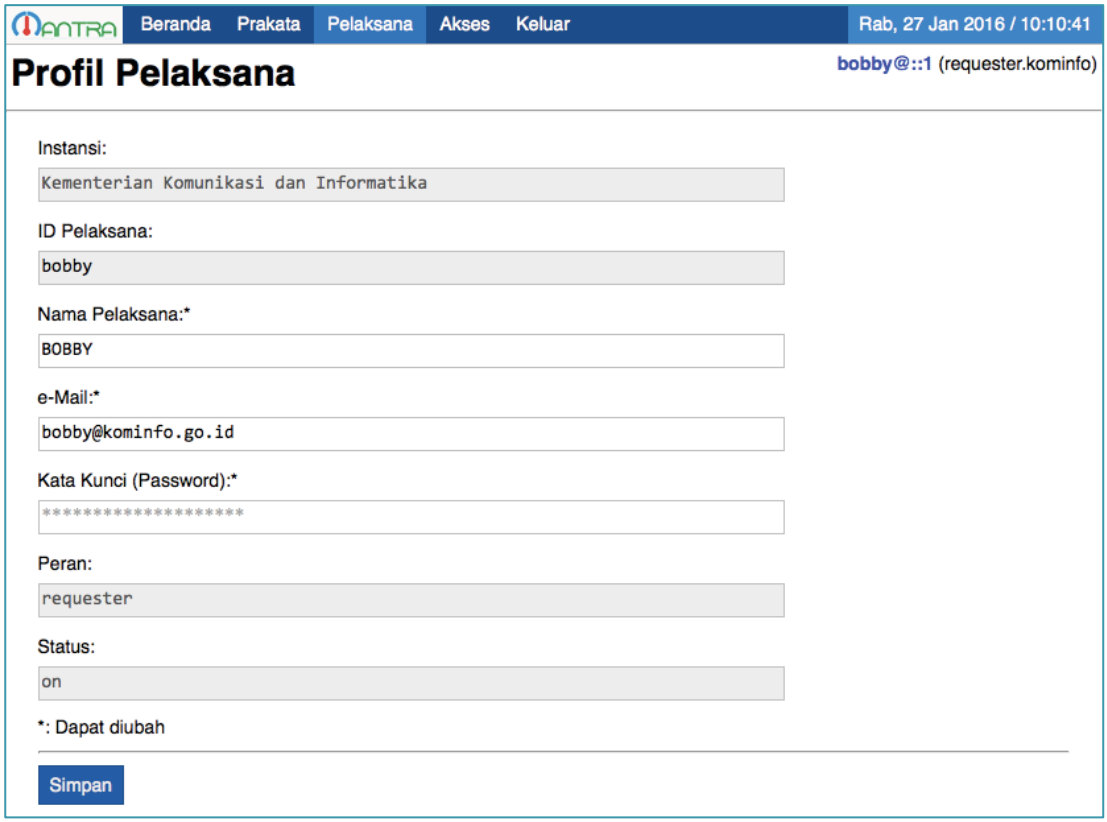
No	Nama Berkas	Keterangan
1	adapter.php	Berkas PHP yang berfungsi menyambungkan aplikasi pengguna dengan layanan di Aplikasi MANTRA.
	Penjelasan Kode Program berkas adapter.php a. Fungsi callAPI(): fungsi untuk memanggil layanan menggunakan CURL melalui request HTTP dengan output XML atau array. Input Parameter callAPI(): 1. endpoint (teks berisi alamat url akses direktori layanan MANTRA) 2. operation (teks berisi nama fungsi layanan MANTRA) 3. accesskey (teks berisi kunci akses fungsi layanan MANTRA) 4. parameter (array berisi parameter-paremeter fungsi layanan) 5. xmlformat (boolen token output MANTRA, "TRUE" jika menginginkan output dalam format XML, "FALSE" jika menginginkan output dalam format Array) 6. callmethod (teks berisi jenis metode pemanggilan fungsi layanan, untuk fungsi layanan dalam Aplikasi MANTRA standarnya menggunakan metode pemanggilan REST/POST) 7. agent (teks berisi identitas agen pemanggilan fungsi layanan MANTRA) b. Fungsi getAPIXML(): fungsi untuk memanggil layanan menggunakan CURL melalui request HTTP dengan output XML. Input Parameter getAPIXML(): 1. endpoint (teks berisi alamat url akses direktori layanan MANTRA) 2. operation (teks berisi nama fungsi layanan MANTRA) 3. accesskey (teks berisi kunci akses fungsi layanan MANTRA) 4. parameter (array berisi parameter-paremeter fungsi layanan) 5. callmethod (text berisi jenis metode pemanggilan fungsi layanan, untuk fungsi layanan dalam Aplikasi MANTRA standarnya menggunakan metode pemanggilan REST/POST) 6. agent (teks berisi identitas agen pemanggilan fungsi layanan MANTRA)	

No	Nama Berkas	Keterangan
	c. Fungsi getAPIJSON(): fungsi untuk memanggil layanan menggunakan CURL melalui request HTTP dengan output JSON. Input Parameter getAPIJSON(): 1. endpoint (teks berisi alamat url akses direktori layanan MANTRA 2. operation (teks berisi nama fungsi layanan MANTRA) 3. accesskey (teks berisi kunci akses fungsi layanan MANTRA) 4. parameter (array berisi parameter-paremeter fungsi layanan) 5. callmethod (text berisi jenis metode pemanggilan fungsi layanan, untuk fungsi layanan dalam Aplikasi MANTRA standarnya menggunakan metode pemanggilan REST/POST) 6. agent (teks berisi identitas agen pemanggilan fungsi layanan MANTRA) d. Fungsi setXML2Array(): fungsi untuk melakukan konversi format data XML menjadi Array. Input Parameter setXML2Array(): 1. xmldata (Teks dalam format XML) e. Fungsi setArray2XML(): fungsi untuk melakukan konversi format data Array menjadi XML. Input Parameter setArray2XML(): 1. nodename (teks berisi index key data array yang akan dikonversi) 2. data (array berisi data yang akan dikonversi) f. Fungsi view_table(): fungsi untuk menampilkan data array dalam bentuk tabel html. Input Parameter view_table(): 1. data (array berisi data yang akan ditampilkan) 2. key (index key dari data array yang akan ditampilkan) 3. rowsperpage (integer berisi jumlah baris data per halaman)	
2	sample.php	Contoh aplikasi php pengguna layanan MANTRA yang menampilkan output layanan dalam format Tabel, XML, Array, JSON. Contoh ini menggunakan fungsi callAPI() pada adapter.php untuk memanggil layanan MANTRA.
3	sample-xml.php	Contoh aplikasi php pengguna layanan MANTRA yang menampilkan output XML dalam format Tabel. Contoh ini menggunakan fungsi getAPIXML() pada adapter.php untuk memanggil layanan MANTRA.

No	Nama Berkas	Keterangan
4	sample-json.php	Contoh aplikasi php pengguna layanan MANTRA yang menampilkan output JSON dalam format Tabel. Contoh ini menggunakan fungsi getAPIJSON() pada adapter.php untuk memanggil layanan MANTRA.
5	sampleJSON.java	Contoh aplikasi java pengguna layanan MANTRA yang menampilkan output JSON.

3.4.5.4 Update Profil Requester

Requster dapat mengubah profil pribadinya dengan mengakses menu "**Pelaksana**". Informasi-informasi yang dapat diubah antara lain nama pelaksana, e-mail, dan kata kunci (password). Jika ingin melakukan perubahan ID Pengguna, requester harus mengajukan permohonan ubah data kepada administrator sebagaimana dijelaskan dalam subbab 3.4.



Gambar 3.102 Halaman Ubah Profil Requester

4

PENUTUP

Kesuksesan Pengelolaan Layanan Berbagi Pakai Data/Informasi antar Sistem Elektronik Pemerintahan ditentukan oleh aturan pelaksanaan dari suatu organisasi/tim pelaksana yang diarahkan melalui Komite sebagaimana gambaran berikut ini.

Gambar 4.1 Rekomendasi Struktur Tata Kelola Berbagi Pakai Data

DAFTAR PUSTAKA

Pautasso, Cesare., Zimmermann, Olaf., & Leymann, Frank. (2008). *RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision***.** WWW 2008, April 21–25, Beijing, China.

Pautasso, Cesare. (2008). *REST vs. SOAP: Making the Right Architectural Decision***.** SOA Symposium. Amsterdam.

Direktorat Sistem Informasi, Perangkat Lunak dan Konten. (2008). *Kerangka Acuan dan Pedoman Interoperabilitas Sistem Informasi Instansi Pemerintahan*, Departemen Komunikasi dan Informatika.

Lucky. (2008). *XML Web Services: Aplikasi Desktop, Internet & Handphone*. Jasakom.

Jakl, Michael. (2006). *REST: Representational State Transfer*. University of Technology Vienna.

zur Muehlen, Michael., Nickerson, J.V., & Swenson, K.D. (2005). *Developing Web Services Choreography Standards – The Case of REST vs. SOAP*. Decision Support System 40(1):9-29.

W3C Working Group. (2004). *Web Services Architecture*. 19 Februari 2009. http://www.w3.org/TR/ws-arch/#whatis

Whitten, J.L., Bentley, L.D., & Dittman, K.C. (2004). *Systems analysis and design methods.* McGraw-Hill Inc, New York, USA.

Vogels, Werner. (2003). *Web Services Are Not Distributed Objects*. IEEE Internet Computing, 7(6):59-66.

Thomas Fielding, Roy. (2000). *Architectural Style and the Design of Network-based Software Architectures*. Ph.D thesis, Dept. of Information and Computer Science, University of California, Irvine, 2000.

LAMPIRAN

Lampiran 1. Keterangan Kode Status pada Output Layanan MANTRA

Nilai	Keterangan
0	Layanan terkendala ketika dijalankan dan tidak memiliki data
1	Layanan berhasil dijalankan dan memiliki data

Lampiran 2. Keterangan Kode pada Output Layanan MANTRA

Nilai	Pesan	Keterangan
200	OK	Layanan memberikan data output yang sesuai
10001	Parameter fungsi operasi tidak dikenal	Terjadi kesalahan penulisan nama parameter atau belum semua parameter dilengkapi
10002	Fungsi operasi tidak terdaftar	Terjadi kesalahan penulisan nama layanan atau nama fungsi
10003	Alamat (URL) akses layanan tidak terdefinisi	Alamat URL layanan (webapi) belum diisi
10004	Nama fungsi operasi tidak terdefinisi	Terjadi kesalahan dalam penulisan nama fungsi operasi pada alamat atau form layanan
10005	Seluruh parameter fungsi operasi wajib diisi	Parameter layanan belum diisi dengan lengkap
10006	SOAP Constructor Error...	Gagal melakukan inisiasi SOAP Client
10007	The request contains an invalid SOAP body...	Terjadi kesalahan struktur pemanggilan SOAP
10008	SOAP Error...	Terjadi kesalahan saat pemanggilan fungsi layanan SOAP

10009	Hasil fungsi operasi SOAP instansi:layanan.fungsi tidak ada (kosong)	Hasil operasi fungsi layanan SOAP tidak ada (kosong)
10010	Nama fungsi operasi data tidak direkomendasikan: {namafungsi}	Terjadi kesalahan pemberian nama fungsi data
10011	Perintah SQL instansi:layanan.fungsi gagal dioperasikan	Terjadi kesalahan penyusulan perintah SQL
10012	Hasil operasi data instansi:layanan.fungsi tidak ada (kosong)	Hasil operasi data tidak ada (kosong)
10013	Hasil operasi data instansi:layanan.fungsi tidak terhimpun	Hasil operasi data yang diperoleh tidak terhimpun
10014	Nama fungsi operasi program tidak direkomendasikan: {nama_fungsi}	Terjadi kesalahan pemberian nama fungsi program
10015	Operasi program instansi:layanan.fungsi gagal	Terjadi kesalahan penyusunan kode sumber program
10016	Hasil operasi program instansi:layanan.fungsi tidak ada (kosong)	Hasil operasi program tidak ada (kosong)
10017	..., Kunci indeks data tidak dapat digunakan sebagai Tag format XML = ...	Terjadi kesalahan nama indeks yang digunakan sebagai tag data dalam format XML pada hasil operasi API
10018	Kunci indeks/tag utama data fungsi operasi tidak dikenal: ...	Nama kunci indeks atau tag utama data hasil operasi tidak dikenal. Default: response.
20001	URL/EndPoint tidak terdefinisi (kosong)	Alamat URL/EndPoint yang akan dioperasikan CallAPI() tidak terisi
20002	URI tidak terdefinisi (kosong)	Hasil penyusunan URI dalam CallAPI() kosong

Lampiran 3. Template Kode Program untuk Fungsi Layanan Program

No	Fungsi	Keterangan
1	Konversi Data Array ke dalam Format JSON	Kode Program: json_encode($array, JSON_FORCE_OBJECT); Input: $array: variabel data dengan tipe array

2	Fungsi Eksekusi Query Basis Data	Kode Program:
		dbExecute($dbdriver, $hostname, $username, $password, $dbname, $sql, $bindfield, $trx);
		
		Input:
		$dbdriver: Nama Driver Koneksi Basis Data
		$hostname: IP atau Nama Alias Basis Data
		$username: ID Pengguna Basis Data
		$password: Kata Kunci untuk ID Pengguna Basis Data
		$dbname: Nama Basis Data
		$sql: Query Pengolahan Data
		$bindfield: Array Parameter Pengolahan Data
		$trx: Token Commit Transaksi ke Basis Data (“true” untuk operasi INSERT, UPDATE, dan DELETE, “false” untuk operasi SELECT)
		
		Contoh Penggunaan:
		Baca data dan tampilkan output dalam format JSON
		$nama=’Abdullah’;
		$result=dbExecute(
		$dbdriver='mysqli',
		$hostname='localhost',
		$username='user',
		$password='password',
		$dbname=’dbName’,
		$sql='SELECT * FROM tbName WHERE nama=? AND jenis_kelamin=?',
		$bindfield=array($nama,’pria’),
		$trx=false
		);
		
		echo
		is_string($result)?$result:json_encode($result->getRows(),JSON_FORCE_OBJECT);

3	Fungsi INSERT Data ke dalam Basis Data	Kode Program: dbInsert($dbdriver, $hostname, $username, $password, $dbname, $tbname, $bindfield);

		Input: $dbdriver: Nama Driver Koneksi Basis Data $hostname: IP atau Nama Alias Basis Data $username: ID Pengguna Basis Data $password: Kata Kunci untuk ID Pengguna Basis Data $dbname: Nama Basis Data $tbname: Nama Tabel dalam Basis Data $bindfield: Array Nama Kolom dan Nilai Data Contoh Penggunaan: Insert data dan tampilkan output dalam format JSON $nip='197909092009011001'; $result=dbInsert($dbdriver='mysqli', $hostname='localhost', $username='user', $password='password', $dbname='dbName', $tbname='pegawai', $bindfield=array('nama'=>'Abdullah', 'id_pegawai'=>$nip)); echo is_string($result)?$result:json_encode($result->getRows(),JSON_FORCE_OBJECT);
4	Fungsi UPDATE Data di dalam Basis Data	Kode Program: dbUpdate($dbdriver, $hostname, $username, $password, $dbname, $tbname, $bindfield, $where); Input: $dbdriver: Nama Driver Koneksi Basis Data $hostname: IP atau Nama Alias Basis Data $username: ID Pengguna Basis Data $password: Kata Kunci untuk ID Pengguna Basis Data $dbname: Nama Basis Data $tbname: Nama Tabel dalam Basis Data $bindfield: Array Nama Kolom dan Nilai Data

		$where: Identifier Data yang akan diupdate Contoh Penggunaan: Update data dan tampilkan output dalam format JSON $name='Abdullah'; $nip='197909092009011001'; $result=dbUpdate($dbdriver='mysqli', $hostname='localhost', $username='user', $password='password', $dbname='dbName', $tbname='pegawai', $bindfield=array('nama'=>$name, 'jabatan'=>'Kepala Bagian') where='nip='.$nip); echo is_string($result)?$result:json_encode($result->getRows(),JSON_FORCE_OBJECT);
5	Fungsi Delete Data di dalam Basis Data	Kode Program: dbDelete($dbdriver, $hostname, $username, $password, $dbname, $tbname, $where); Input: $dbdriver: Nama Driver Koneksi Basis Data $hostname: IP atau Nama Alias Basis Data $username: ID Pengguna Basis Data $password: Kata Kunci untuk ID Pengguna Basis Data $dbname: Nama Basis Data $tbname: Nama Tabel dalam Basis Data $where: Identifier Data yang akan diupdate Contoh Penggunaan: Delete data dan tampilkan output dalam format JSON $nip='197909092009011001'; $result=dbDelete($dbdriver='mysqli', $hostname='localhost', $username='user', $password='password', $dbname='dbName',

		$tbname='pegawai', where='nip='.$nip); echo is_string($result)?$result:json_encode($result->getRows(),JSON_FORCE_OBJECT);

Lampiran 4. Pengaturan Mail Server Fitur Notifikasi MANTRA

User	Keterangan
Administrator	Administrator berperan melakukan pengaturan *mail server*, pengaturan tersebut bertujuan untuk mengaktifkan fitur notifikasi pada Aplikasi MANTRA. Pengaturan tersebut dapat diakses dengan melakukan login sebagai administrator, kemudian akses submenu "Notifikasi Email" pada menu navigasi "Setelan". Parameter pengaturan mail server yaitu: 1. SMTP Server (alamat domain server pengirim email, contoh: mx.kominfo.go.id) 2. Port SMTP (TLS Port Server SMTP, contoh: 25) 3. e-Mail (Akun e-mail yang terdaftar dalam e-mail domain, akun tersebut akan digunakan sebagai pengirim e-mail notifikasi Aplikasi MANTRA, contoh: mantra@kominfo.go.id) 4. Kata Kunci (Password login akun e-mail ke dalam e-mail domain)
Provider/Publisher	Provider/Publisher akan menerima notifikasi melalui pesan e-mail (sesuai dengan alamat e-mail yang disimpan dalam profil pengguna) jika Requester menambahkan permintaan akses layanan-layanan yang disediakan oleh Provider/Publisher.
Requester	Requester akan menerima notifikasi melalui pesan e-mail (sesuai dengan alamat e-mail yang disimpan dalam profil pengguna) untuk setiap perubahan hak akses (on atau off) yang diberikan oleh Provider/Publisher.