RANCANG BANGUN RESTFUL WEB SERVICE UNTUK OPTIMALISASI KECEPATAN AKSES APLIKASI SISTEM PAKAR PERAWATAN CABAI

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Disusun oleh :

IQBAL LUTHFILLAH

NIM. 0910680045

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

PROGRAM STUDI INFORMATIKA

MALANG

2014

LEMBAR PERSETUJUAN

RANCANG BANGUN RESTFUL WEB SERVICE UNTUK OPTIMALISASI KECEPATAN AKSES APLIKASI SISTEM PAKAR PERAWATAN CABAI

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :

IQBAL LUTHFILLAH

NIM. 0910680045

Dosen Pembimbing I	Dosen Pembimbing II
Arief Andy Soebroto, ST., M.Kom	**Budi Darma Setiawan, S.Kom, M.Cs.**
NIP 197204251999031002	**NIP 84101506110090**

LEMBAR PENGESAHAN

RANCANG BANGUN RESTFUL WEB SERVICE UNTUK OPTIMALISASI KECEPATAN AKSES APLIKASI SISTEM PAKAR PERAWATAN CABAI

SKRIPSI
KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :
IQBAL LUTHFILLAH
NIM. 0910680045

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 20 Januari 2014

Penguji I

Ismiarta Aknuranda, ST., M.Sc., Ph.D
NIK 74071906110079

Penguji II

Dr. Eng Herman Tolle, ST., MT.
NIP 19740823 200012 1 001

Penguji III

Agi Putra Kharisma, ST., MT.

Mengetahui,
Ketua Program Studi Informatika

Drs. Marji, MT.
NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 25 Januari 2014

Mahasiswa,

Igbal Luthfillah

0910680045

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, penulis dapat menyelesaikan laporan skripsi yang berjudul ”Rancang bangun restful web service untuk optimalisasi kecepatan akses aplikasi sistem pakar perawatan cabai”. Dalam kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar – besarnya kepada :

1. Bapak Drs. Marji, MT selaku Ketua Program Studi Teknik Informatika Universitas Brawijaya Malang.
2. Arief Andy S, ST, M.Kom selaku dosen pembimbing I selama pelaksanaan skripsi.
3. Bapak Budi Darma Setiawan, S.Kom, M.Cs.selaku dosen pembimbing II selama pelaksanaan skripsi.
4. Orang tua tercinta Bambang Cahyono dan Lilik Safrida.
5. Bapak Dr. Eng Herman Tolle, ST, MT, Bapak Ismiarta Aknuranda ST, M.Sc, Ph.D, dan Bapak Agi Putra Kharisma, ST., MT. selaku penguji skripsi.
6. Seluruh pihak yang telah memberi support penulis dalam penyelesaian skripsi ini yaitu my beloved girl Rindang Mazdarani H S.Kom, Hafidz Rozaq N.J S.Kom, Thierry Rahman A S.Kom. Mustikadewi Prihastuti S.Kom, Yusuf Oktofani S.Kom, seluruh anak TPL angkatan 2009 dan civitas PTIIK yang tidak bisa disebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan laporan ini masih banyak kekurangan. Oleh karena itu, saran dan kritik membangun dari para pembaca senantiasa diharapkan. Semoga laporan ini dapat memberikan manfaat bagi semua pihak, Amin.

Malang, 25 Januari 2014

Penulis

ABSTRAKSI

Iqbal Luthfillah. 2014. : Rancang bangun restful *web service* untuk optimalisasi kecepatan akses aplikasi sistem pakar perawatan cabai. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Arief Andy Soebroto, ST., M.Kom dan Budi Darma Setiawan, S.Kom, M.Cs.

Indonesia menurut data termasuk sebagai negara dengan kecepatan akses internet yang lambat dibandingkan negara lain di Asia Pasifik. Kecepatan akses internet yang lambat menyebabkan kecepatan loading situs menjadi lama terutama pada situs yang memiliki bandwith besar.

Untuk mempercepat kinerja suatu situs penelitian ini akan menggunakan teknologi *web service. Web service* merupakan website yang berfungsi memberikan layanan kepada aplikasi lain yang mengaksesnya. Layanan yang diberikan berbentuk *message* atau pesan *object*. *Representational State Transfer* (REST) merupakan sebuah teknik arsitektur software untuk *web service* yang tidak memerlukan *parsing* XML dan tidak memerlukan sebuah *header* pesan ke dan dari *web service*. Hal ini pada akhirnya mengurangi penggunaan *bandwith.*

Penelitian akan menerapkan teknologi REST *Web service* pada studi kasus sistem pakar perawatan tanaman cabai merah keriting berbasis web. Aplikasi web sistem pakar dibuat menjadi *web service* sistem pakar yang menyediakan layanan *logic* sistem pakar tanpa *user interface* yang memberikan beban pada bandwith. Hasil penelitian menyimpulkan bahwa penggunaan restful webservice mereduksi waktu loading akses sistem pakar perawatan cabai hingga 35.44%.

Kata kunci : ***kecepatan internet, web service, REST, sistem pakar perawatan cabai***

ABSTRACTION

Iqbal Luthfillah . 2014 . : ***Development of restful web service to optimize expert system of chili plant access speed. Ungraduated thesis of Informatics Engineering Program , Information Technology and Computer Science Program , University of*** **Brawijaya , Malang .** ***Advisor*****: Arief Andy Soebroto, ST., M.Kom dan Budi Darma Setiawan, S.Kom, M.Cs.**

Indonesia according to the data including of a country with slow internet acces connection than other countries in asia pacific. The slow connection make loading time to open a site lead so long time especially on sites that have a large space.

To speed up the performance of a site, in this study will be using *web service* technology. *Web service* is a website whose have function to provide service to other application that acces it. Representational State Transfer (REST) is a *web service* architectural for developing *web service* that doesnt required XML parsing and doesnt required a message header from and to *web service*. This ultimately reduces bandwidth usage .

Research will implement REST *Web service* technology on a case study of expert systems red chili plant maintenance web based . Web application expert system be converted to a *web service* that provides a service of expert system logic without an *user interface* that make the load on bandwidth . The research concludes that the use of restful webservice reduce time speed loading system access chili expert care of up to 35.44 % .

Keywords : speed access internet , web services , REST , expert system

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia, menurut data "state of the internet" termasuk kedalam Negara dengan kecepatan akses internet terlambat di kawasan asia pasifik [AKM-12]. Kecepatan rata rata akses internet di Indonesia pada kuartal ketiga tahun 2012 hanya 1.2 Mbps dengan mendapat peringkat 115 dari 120 negara asia pasifik.Kecepatan akses internet mempengaruhi kecepatan *loading* suatu situs. Kecepatan *loading* situs-situs yang diakses melalui *desktop* dan *mobile* di Indonesia rata-rata memakan waktu *loading* 20,3 detik jika diakses melalui *dekstop* computer dan 12,9 detik jika diakses melalui perangkat *mobile* [TMP-12]. Kecepatan *loading* situs merupakan hal yang sangat penting mengingat beberapa hal. Pertama, 40% pengunjung internet membatalkan kunjungannya ke situs tertentu jika loading time lebih dari 3 detik. Kedua, 47% konsumen berharap waktu *loading* itu hanya 2 detik atau kurang. Ketiga, keterlambatan respon halaman situs 1 detik, berarti kehilangan *conversation* sebanyak 7% [SIL-13]. Ini menjadi batasan bagi *developer* aplikasi terutama aplikasi web dalam membangun sebuah perangkat lunak terutama dari segi *bandwith* / ukuran apilkasi yang mempengaruhi kecepatan akses ke aplikasi web.

Web service adalah sebuah aplikasi yang dapat teridentifikasi oleh *Unified Resource Identifier* (URI) dan memiliki interface yang didefiniskan, dideskripsikan, dan dimengerti oleh *Extensible Markup Language* (XML) dan juga mendukung interaksi langsung dengan *software* aplikasi yang lain dengan menggunakan *message* berbasis XML melalui protokol internet [WSC-13]. *Web service* merupakan website yang berfungsi memberikan layanan kepada aplikasi lain yang mengaksesnya. Layanan yang diberikan berbentuk *message* atau pesan *object*. *Web service* tidak meliki *user interface* layaknya sebuah website dan dapat bekerja di semua tipe *client* maupun *operating system* yang berbeda. *Representational State Transfer* (REST) merupakan sebuah teknik arsitektur software untuk *web service*. REST tidak

memerlukan parsing XML dan tidak memerlukan sebuah header pesan ke dan dari *web service*. Hal ini pada akhirnya mengurangi penggunaan bandwith [RST-13]. *Web service* yang menerapkan aristektur REST dinamakan RESTfull webservice.

Penelitian ini akan menerapkan teknologi RESTfull *Web service* pada studi kasus sistem pakar perawatan tanaman cabai merah keriting berbasis web standar [PKL-12]. Web standar yang dimaksud adalah model website dengan user interface, model dan logic terletak pada satu tempat yang sama.User interface yang berisi data html, css dan gambar menyebabkan model website standar memiliki ukuran yang lebih besar dibandingkan dengan *web service*. Aplikasi terdahul sistem pakar perawatan tanaman cabai merah keriting berbasis web standar diubah menjadi *web service* sistem pakar yang menyediakan layanan logic sistem pakar tanpa *user interface* yang memberikan beban pada bandwith. *User interface* nya berada disisi client yang dibuat dalam platform *mobile* web application. *Mobile web application* disebut juga sebagai aplikasi berbasis web. Aplikasi jenis ini dibuat menggunakan html5, javascript dan css. Keunggulan dari aplikasi jenis ini adalah bisa dijalankan pada berbagai sistem operasi (IOS, Android, Symbian).. Hasil yang diharapkan dalam penelitian ini adalah peningkatan kecepatan akses aplikasi sistem pakar dibanding aplikasi web sistem pakar terdahulu.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas maka penulis merumuskan masalah-masalah yang akan dibahas adalah :

1. Bagaimana merancang *web service* untuk sistem pakar perawatan cabai dengan menggunakann arsitektur REST ?
2. Bagaimana implementasi sistem pakar perawatan cabai berbasis REST *web service*?
3. Bagaimana hasil pengujian kecepatan akses aplikasi sistem pakar REST *web service* dibandingkan dengan aplikasi pada penelitian terdahulu yang menggunakan web standar ?

1.3 Batasan Masalah

Penelitan yang akan dibahas dibatasi dengan variable sebagai berikut :

- Studi kasus yang digunakan adalah aplikasi perawatan cabai merah keriting berbasis website.
- *Web service* menggunakan arsitektur REST.
- Bagian yang dibandingkan adalah kecepatan akses.
- Platform yang digunakan dalam penelitian adalah perangkat *mobile* full touch dengan ukuran layar 240x400 px.

1.4 Tujuan Penelitian

Adapun tujuan yang diinginkan dalam penelitian ini sebagai berikut :

1. Sebagai acuan peneliti lain dalam mengembangkan penelitian.
2. Mengimplementasikan Restfull *web service* terhadap aplikasi yang menjadi studi kasus.

1.5 Manfaat Penelitian

Adapun manfaat yang diharapkan dalam penelitian ini sebagai berikut :

- Bagi penulis
  - Mempraktekan ilmu yang didapat selama perkuliahan khusunya dibidang rekayasa perangkat lunak.
  - Membantu masyarakat secara langsung khususnya para petani yang merupakan tulang punggung Negara Indonesia.
- Bagi Pembaca
  - Sebagai bahan dan acuan dalam mengembangkan aplikasi yang memilik keterkaitan dengan penelitian ini.

1.6 Sistematika Penyusunan Laporan

Sistematika penyusunan laporan ditunjukkan untuk memberikan gambaran dan uraian dari laporan skripsi secara garis besar yang meliputi beberapa bab, sebagai berikut.

BAB I : Pendahuluan

Menguraikan latar belakang skripsi, rumusan masalah tentang pengembangan penelitan , tujuan dan manfaat skripsi serta sistematika penyusunan laporan skripsi.

BAB II: Tinjauan Pustaka

Menguraikan teori-teori yang menjadi referensi dalam pelaksanaan skripsi.

BAB III : Metode Penelitian

Menguraikan tentang metode dan langkah kerja yang dilakukan dalam proses perancangan dan implementasi pada pelaksanaan skripsi dan perancangan sistem yang menjadi objek studi kasus skripsi.

BAB IV : Perancangan

Menguraikan proses perancangan perangkat lunak.

BAB V : Implementasi

Menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem.

BAB VI : Pengujian dan Analisis

Menguraikan proses pengujian serta kendala yang dihadapi selama pengujian sistem.

BAB VII : Kesimpulan dan Saran

Menguraikan kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam ini serta saran-saran untuk pengembangan lebih lanjut.

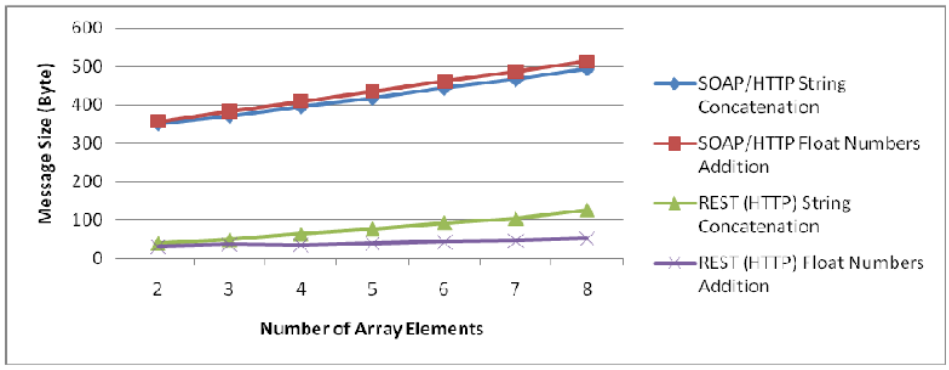
BAB II

TINJAUAN PUSTAKA

Bab tinjauan pustaka terdiri dari kajian pustaka dan dasar teori. Kajian pustaka membahas penelitian terdahulu yang menjadi acuan pengembangan skripsi. Penelitan terdahulu dengan judul "Performance Evaluation of RESTful Web*service"* membuktikan bahwa RESTful *web service* lebih baik performa kecepatan akses datanya dibandingkan dengan SOAP *web service*. Penelitian terdahulu yang berjudul "Perancangan dan Implementasi sistem pakar perawatan cabai keriting berbasis web" menjadi studi kasus skripsi. Dasar teori membahas teori yang dijadikan landasan pengembangan aplikasi RESTful webservice sistem pakar perawatan cabai. Dasar teori yang digunakan meliputi teori *web service*, REST, rekayasa perangkat lunak, HTML dan sistem pakar.

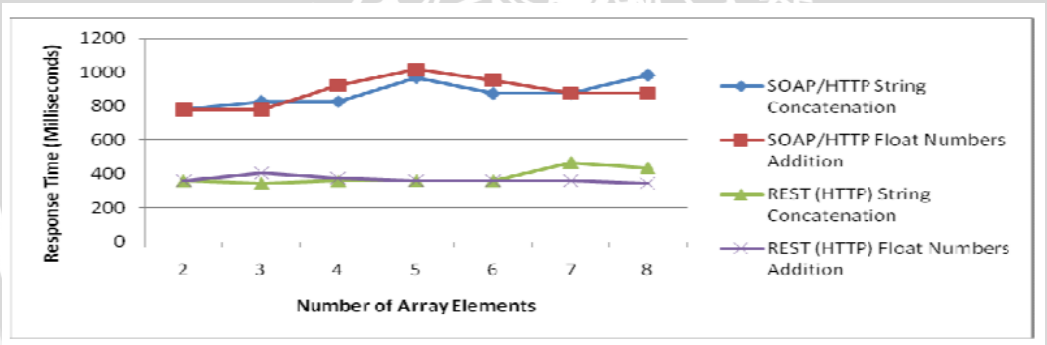
2.1 KAJIAN PUSTAKA

Penelitian terdahulu dengan berjudul "Perancangan dan Implementasi sistem pakar perawatan cabai keriting berbasis web" [PKL-12]. Penelitian tersebut membahas tentang perancangan aplikasi sistem pakar dengan basis web. Penelitian sebelumnya menjadi objek studi kasus dan dikembangkan dalam penelitian ini. Jurnal dengan judul "Performance Evaluation of RESTful Web*service*" menjadi acuan penulis dalam penggunaan arsitektur REST dalam penelitian. Penelitian tersebut mengevaluasi kinerja antara RESTful *web service* dengan *web service Simple object Access Protocol* (SOAP) konvensional. Kasus uji yang digunakan adalah layanan penggabungan string dan kalkulasi angka. Hasilnya menyimpulkan bahwa performa REST lebih baik dibandingkan dengan *web service* SOAP konvensional pada implementasi mobile.



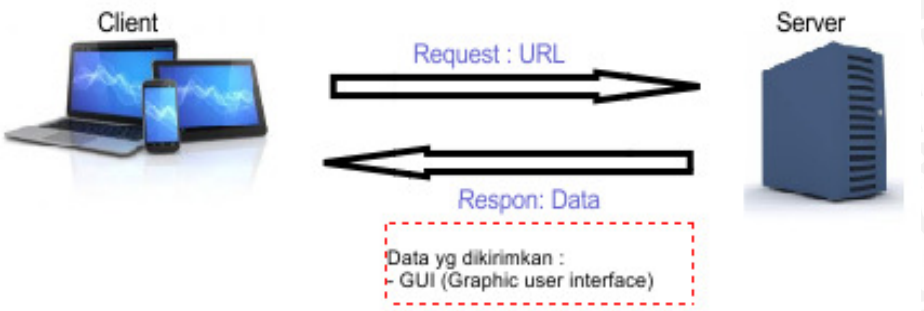
Gambar 2.1 RESTful vs SOAP ukuran *message* dalam kasus uji *service* penggabungan string.
Sumber : [HAH-09]

REST memiliki kelebihan di *message* yang lebih ringan dan waktu respon yang lebih cepat dibandingkan SOAP web*service* konvensional [BBB]. Perbedaan REST dan SOAP akan dijelaskan lebih detail pada bab dasar teori.



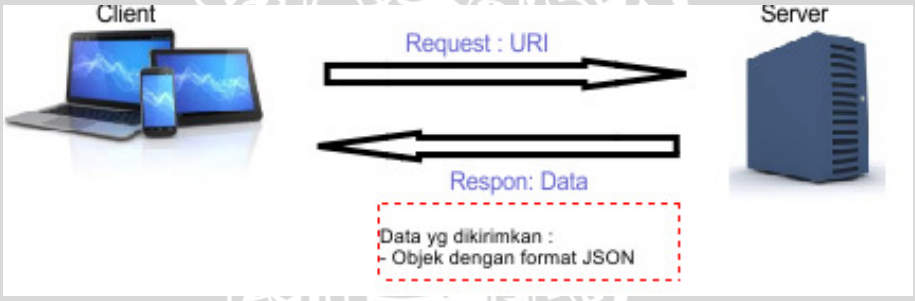
Gambar 2.2 RESTful vs SOAP respon time kasus uji *service* penggabungan string.
Sumber : [HAH-09]

Arsitektur penelitian terdahulu mengenai aplikasi sistem pakar perawatan cabai berbasis web digambarkan di gambar 2.3. *Client* merequest *server* menggunakan *URL*. *Server* aplikasi merespon client dengan mengirimkan GUI dalam bentuk halaman web ke client.



Gambar 2.3 Arsitektur aplikasi web sistem pakar
Sumber : [PKL-12]

Arsitektur aplikasi yang diajukan dengan judul sistem pakar perawatan cabai berbasis rest *web service* digambarkan di gambar 2.4. Client merequest server menggunakan URI. Server merespon dengan mengirimkan data objek dalam format JSON ke client. GUI sudah tertanaman diaplikasi client.



Gambar 2.4 Arsitektur aplikasi web sistem pakar dengan rest *web service*
Sumber : Perancangan

2.2 *WEB SERVICE* SOAP

Web service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. *Web service* digunakan sebagai suatu fasilitas yang disediakan oleh suatu web site untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (*service*) yang disediakan oleh suatu sistem yang menyediakan *web service*. *Web service* menyimpan data informasi dalam format *extensible markup language* (XML),

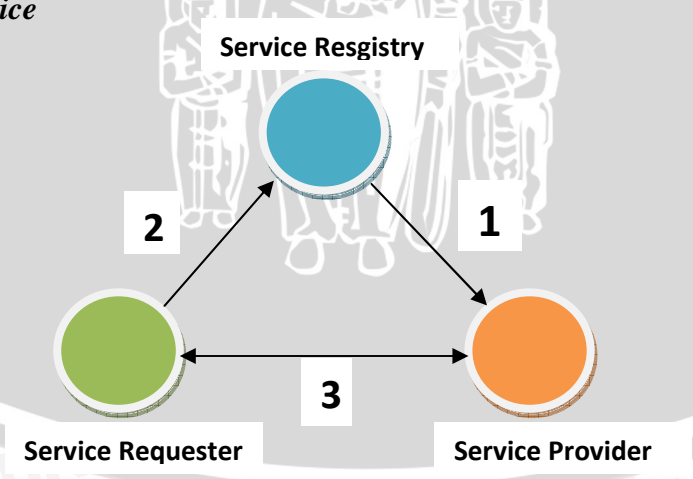
sehingga data ini dapat diakses oleh sistem lain walaupun berbeda platform, sistem operasi, maupun bahasa compiler [YAU-12].

Web service bertujuan untuk meningkatkan kolaborasi antar pemrogram dan perusahaan, yang memungkinkan sebuah fungsi di dalam *Web service* dapat dipinjam oleh perangkat lunak lain tanpa perlu mengetahui detil pemrograman yang terdapat di dalamnya.

Beberapa alasan mengapa digunakannya *web service* adalah sebagai berikut [YAU-12] :

1. *Web service* dapat digunakan untuk mentransformasikan satu atau beberapa bisnis logic atau class dan objek yang terpisah dalam satu ruang lingkup yang menjadi satu, sehingga tingkat keamanan dapat ditangani dengan baik.
2. *Web service* memiliki kemudahan dalam proses deployment-nya, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. *Web service* cukup di-upload ke web server
3. *Web service* berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian *web service* tidak memerlukan konfigurasi khusus di sisi firewall.

Entitas *Web service*



Gambar 2.5 Entitas web*service*
Sumber : [YAU-12]

Web service memiliki tiga entitas dalam arsitekturnya. *Service* Requester sebagai peminta layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut. *Service* Provider berfungsi untuk menyediakan layanan dan membuatnya tersedia. *Service* Registry mendeskripsikan semua layanan yang telah didaftarkan. Proses yang dilakukan antar entitas sebagai berikut :

1. Registry mendeskripsikan semua layanan yang disediakan oleh provider
2. Request mencari layanan ke provider
3. Request dan provider saling bertukar pesan *respond* dan *request*

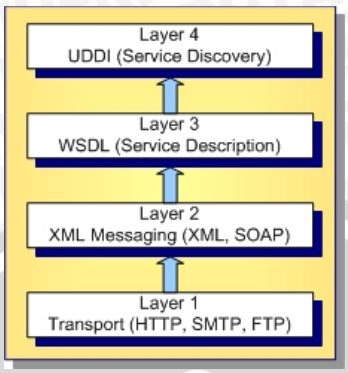
Operasi-Operasi *Web service*

Secara umum, *web service* memiliki tiga operasi yang terlibat di dalamnya, yaitu [YAU-12] :

1. *Publish/Unpublish*: Menerbitkan/menghapus layanan ke dalam atau dari registry.
2. *Find*: *Service* requestor mencari dan menemukan layanan yang dibutuhkan.
3. *Bind*: *Service* requestor setelah menemukan layanan yang dicarinya, kemudian melakukan binding ke *service* provider untuk melakukan interaksi dan mengakses layanan/*service* yang disediakan oleh *service* provider.

SOAP *Web service*

Web service secara keseluruhan memiliki empat layer komponen seperti pada gambar di dibawah, yaitu [YAU-12]:



Gambar 2.6 Layer komponen *Web service*
Sumber : [YAU-12]

Layer 1: Protokol internet standar seperti HTTP, TCP/IP

Layer 2: Simple Object Access Protocol (SOAP), merupakan protokol akses objek berbasis XML yang digunakan untuk proses pertukaran data/informasi antar layanan.

Layer 3: *Web service Definition Language* (WSDL) merupakan suatu standar bahasa dalam format XML yang berfungsi untuk mendeskripsikan seluruh layanan yang tersedia.

Layer 4: Memusatkan seluruh layanan pada registry.

2.3 RESTFUL *WEB SERVICE*

Pada tahun 2000, Roy Fielding menulis disertasi tentang arsitektur dan desain dari arsitektur software berbasis jaringan. Dalam disertasi tersebut, dia menyimpulkan tentang arsitektur web dimana setiap informasi atau konsep dapat diberikan penamaan merujuk kepada resource dan dapat diidentifikasikan dengan resource identifier URI. REST sering dirujuk sebagai sebuah gaya arsitektural ketimbang sebuah arsitektur yang mendefinisikan prinsip-prinsip yang dengannya arsitektur dibuat dan dievaluasi, dengan cara meletakkan konstrain-konstrain pada arsitektur jaringan [SUA-10]. Kunci metodologi REST adalah untuk menulis *web service* menggunakan antarmuka yang sudah tersedia dan banyak digunakan yaitu URI [ADR-12].

Aplikasi client atau server dengan dukungan HTTP dapat dengan mudah memanggil *service* tersebut dengan command HTTP GET. Berdasar pada bagaimana cara penyedia *service* menulis script, hasil respons HTTP kan menjadi lebih simpel seperti beberapa header standar dan string teks yang mengandung harga terkini untuk symbol yang diberikan. Atau, dapat berupa dokumen XML [ADR-12]. Penggunaan metode REST dalam *web service* menggunakan beberapa komponen sebagai berikut :

- XML : format yang merepresentasikan sumber, berfungsi sebagai bahasa pesan antar platform,
- HTTP : metode GET, HEAD, POST, PUT dan DELETE mengindikasikan action yang akan diambil,
- URI : URI adalah resource identifier lokasi dari *web service.*

REST CONSTRAINT

REST sebenarnya bukan merupakan sebuah arsitektur tapi lebih mendekati kumpulanconstraint(batasan) yang ketika diterapkan pada desain sebuah sistem, akan menjadi jenis arsitektur perangkat lunak. RESTful service harusnya memenuhi constraint-contraint seperti [PHI-12]:

1. Resource Identification: Semua Resource (serta statenya) yang berhubungan dengan aplikasi diberikan identifier yang unik dan identifier tersebut harus bersifat global. Konsep resource disini bukan hanya hal statis yang langsung berhubungan dengan aplikasi namun juga termasuk informasi yang dibutuhkan seperti dokumen transaksi. RESTful resource adalah semua hal yang bisa diakses dan ditransfer melalui web antara client dan server. Dan karena protokol yang digunakan untuk berkomunikasi adalah HTTP, berbagai macam tipe file bisa ditransfer, teks file, flash movie,gambar dll. Sehingga dalam RESTful system representasi dari resource tergantung dari tipe yang direquest client (MIME type) yang didefinisikan didalam protokol request.

2. Uniform Interface: Semua interaksi sebaiknya dibangun dengan interface yang seragam. RESTful service menampilkan semua resource dan interaksinya dengan interface yang seragam, tidak seperti RPC yang menampilkan fungsi yang ada melalui method yang bisa dipanggil secara remote. Dalam RESTful *web service* untuk uniform interface ini menggunakan Uniform Resource Identifier(URI). URI pada RESTful webservice berupa hyperlink terhadap resource meskipun RESTful constraint tidak menyatakan URI harus berupa hyperlink, namun karena teknologi yang digunakan pada *web service* adalah web sehingga URI berupa hyperlink. Jika menggunakan teknologi lain, RESTful URI tentu akan berupa hal yang berbeda, namun tetap berupa address terhadap sebuah resource.
3. Self-Describing Message: untuk setiap interaksi dengan resource melalui interface yang seragam, REST membutuhkan representasi dari resource yang menggambarkan semua aspek penting yang dimiliki oleh resource tersebut. Representasi dari resource sendiri adalah semua hal yang dikirim antara cilent dan server. Representasi merupakan state sementara dari data sebenarnya yang terletak di suatu tempat penyimpanan. Dengan kata lain representasi merupakan stream biner besama metadata yang menjelaskan bagaimana stream tersebut digunakan baik untuk client maupun untuk server. Bisa terdapat banyak jenis client yang me-request resource yang ada, oleh karena itu representasi setiap client pun dapat berbeda. Representasinya dapat berupa gambar, text file, stream XML atau stream JSON, tapi kesemua representasi tersebut harus tersedua melalui URI yang sama. Untuk kasus request yang dilakukan oleh manusia (human user) biasanya representasi berupa laman web sehingga menjadi bentuk representasi yang dapat dibaca.

4. Stateless Interaction: Setiap interaksi antara client dan server harus memiliki state sendiri (atau dengan kata lain tidak dipengaruhi session client). Jadi server hanya akan memantau resource state bukan client session.

Semua Constraint tersebut tidak berpengaruh dengan teknologi yang akan digunakan untuk implementasi. Constraint tersebut hanya mendefinisikan bagaimana data ditransfer antar komponen dan keuntungan apa yang didapat. Dan tidak perlu mencari teknologi atau protokol jaringan baru untuk mengimplementasikannya, karena RESTful system dapat diaplikasikan ke infrastruktur jaringan yang sudah ada seperti web, sehingga muncullah RESTful service.

Hypertext Transfer Protocol (HTTP) Methods

HTTP digunakan dalam arsitektur REST. Representasi adalah mapping resource yang ditransfer antara client dan server. Pada *web service* representasi tersebut diajukan dengan komunikasi antara client dan server melalui protokol komunikasi HTTP [ADR-12].

Membangun sebuah RESTful *web service* akan sejalan dengan cara membangun sebuah aplikasi web masa kini. Akan tetapi hal paling dasar yang berbeda untuk membangun aplikasi web yang modern dan tradisional adalah bagaimana memodelkan aksi ke abstraksi data. Modern development menitik beratkan pada pertukaran resource, sedangkan tradisional development menitik beratkan pada aksi remote untuk mengambil data [RER-12].

Pada proses development web modern tingkat ambiguitas design dan implementasi harus dibatasi karena sudah terdapat empat aksi spesifik yang bisa dilakukan terhadap resource, yaitu *Create, Retrieve, Update* dan *Delete* (CRUD). Sedangkan pada development tradisional web banyak sekali aksi yang tidak memiliki strandard penamaan dan implementasi yang dapat dilakukan [RER-12].

Pada REST aksi CRUD tersebut dapat dimappingkan dengan HTTP method sebagai berikut [RER-12]. :

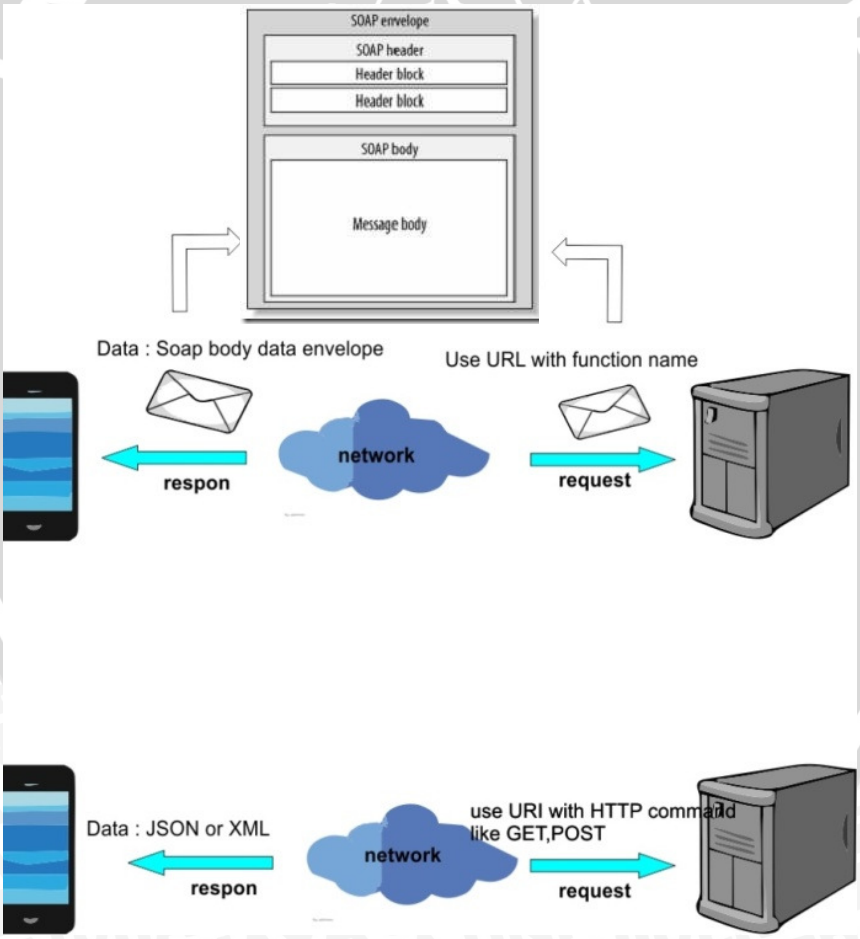
Tabel 2.1 Aksi di REST dan HTTP method yg digunakan

AKSI	HTTP METHOD
CREATE / ADD	POST
RETRIEVE	GET
UPDATE	PUT
DELETE DATA	DELETE

Sumber : Kajian Pustaka

Perbedaan message SOAP dan REST

REST tidak membutuhkan *envelope* layaknya SOAP *web service* dalam akses request dan respon layanan.



Gambar 2.7 Perbedaan arsitektur SOAP dan REST web *servi ce*
Sumber : [TDB-12]

Envelope mendefinisikan seluruh framework untuk menjawab fungsi fungsi yang terdapat didalam message dan elemen elemen yang terlibat. Envelope berisi header (optional) dan body (required). Header berisi blok informasi yang berhubungan dengan bagaimana pesan tersebut diproses. Hal ini meliputi pe-routingan dan delivery setting, authentication atau authorization assertions, and transaction contexts. Body berisi pesan sebenarnya yang dikirim dan diproses. Semua yang dapat ditampilkan dengan sintaks XML dapat dimasukkan dalam pesan body. Tidak adanya envelope membuat data yang terkirim lebih ringan.

2.4 REKAYASA PERANGKAT LUNAK

Istilah Rekayasa Perangkat Lunak (RPL) secara umum disepakati sebagai terjemahan dari istilah Software Engineering. Istilah Software Engineering mulai dipopulerkan tahun 1968 pada Software Engineering Conference yang diselenggarakan oleh NATO. Sebagian orang mengartikan RPL hanya sebatas pada bagaimana membuat program komputer. Padahal ada perbedaan yang mendasar antara perangkat lunak (software) dan program komputer. Perangkat lunak adalah seluruh perintah yang digunakan untuk memproses informasi. Perangkat lunak dapat berupa program atau prosedur. Program adalah kumpulan perintah yang dimengerti oleh komputer sedangkan prosedur adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi. Pengertian RPL sendiri adalah sebagai berikut: Rekayasa perangkat lunak memakai pendekatan yang sistematis dan terorganisir terhadap pekerjaan mereka karena cara ini seringkali paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Rekayasa ini mencakup masalah pemilihan metode yang paling sesuai untuk satu set keadaan dan pendekatan yang lebih kreatif, informal terhadap pengembangan yang mungkin efektif pada beberapa keadaan[SOM-03:7].

***System Development Life Cycle* (SDLC)**

SDLC adalah keseluruhan proses dalam membangun sistem melalui beberapa langkah. Ada beberapa model SDLC. Model yang cukup populer dan banyak digunakan adalah waterfall. Beberapa model lain SDLC misalnya fountain, spiral, rapid, prototyping, incremental, build & fix, dan synchronize & stabilize. Dengan siklus SDLC, proses membangun sistem dibagi menjadi beberapa langkah dan pada sistem yang besar, masing-masing langkah dikerjakan oleh tim yang berbeda.

Dalam sebuah siklus SDLC, terdapat enam langkah. Jumlah langkah SDLC pada referensi lain mungkin berbeda, namun secara umum adalah sama. Langkah tersebut adalah [YGP-13].

- Analisis sistem, yaitu membuat analisis aliran kerja manajemen yang sedang berjalan
- Spesifikasi kebutuhan sistem, yaitu melakukan perincian mengenai apa saja yang dibutuhkan dalam pengembangan sistem dan membuat perencanaan yang berkaitan dengan proyek sistem
- Desain sistem, yaitu membuat desain aliran kerja manajemen dan desain pemrograman yang diperlukan untuk pengembangan sistem informasi
- Pengembangan sistem, yaitu tahap pengembangan sistem informasi dengan menulis program yang diperlukan
- Pengujian sistem, yaitu melakukan pengujian terhadap sistem yang telah dibuat
- Implementasi dan pemeliharaan sistem, yaitu menerapkan dan memelihara sistem yang telah dibuat

Gambar 2.8 SDLC life cycle
Sumber : [RPL-08]

Siklus SDLC dijalankan secara berurutan, mulai dari langkah pertama hingga langkah keenam. Setiap langkah yang telah selesai harus dikaji ulang, kadang-kadang bersama expert user, terutama dalam langkah spesifikasi kebutuhan dan perancangan sistem untuk memastikan bahwa langkah telah dikerjakan dengan benar dan sesuai harapan. Jika tidak maka langkah tersebut perlu diulangi lagi atau kembali ke langkah sebelumnya.

2.5 Hyper Teks Markup Language (HTML)

HTML adalah sebuah bahasa markup yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah Penjelajah web Internet dan formating hypertext sederhana yang ditulis kedalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi. Dengan kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan kedalam format ASCII normal sehingga menjadi home page dengan perintah-perintah HTML. HTML inilah yang memungkinkan Anda menjelajah internet dan melihat halaman web yang menarik [KKN-07:1].

HTML dokumen tersebut mirip dengan dokumen teks biasa, hanya dalam dokumen ini sebuah teks bisa memuat instruksi yang ditandai dengan kode atau lebih

dikenal dengan TAG tertentu. Sebagai contoh jika ingin membuat teks ditampilkan menjadi tebal sepserti: **TAMPIL TEBAL**, maka penulisannya dilakukan dengan cara: TAMPIL TEBAL. Tanda digunakan untuk mengaktifkan instruksi cetak tebal, diikuti oleh teks yang ingin ditebalkan, dan diakhiri dengan tanda untuk menonaktifkan cetak tebal tersebut.

Secara garis besar, terdapat 4 jenis elemen dari HTML:

- *structural*. tanda yang menentukan level atau tingkatan dari sebuah teks (contoh, `<h1>Golf</h1>` akan memerintahkan browser untuk menampilkan "Golf" sebagai teks tebal besar yang menunjukkan sebagai Heading 1,
- *presentational*. tanda yang menentukan tampilan dari sebuah teks tidak peduli dengan level dari teks tersebut (contoh, `<b>boldface</b>` akan menampilkan **bold**. Tanda presentational saat ini sudah mulai digantikan oleh CSS dan tidak direkomendasikan untuk mengatur tampilan teks,
- *hypertext. Tanda yang menghubungkan dengan halaman atau sumber lain*
- Elemen *widget* yang membuat objek-objek lain seperti tombol (<button>), list (<li>), dan garis horizontal (<hr>).

2.6 Standart Query Language (SQL)

SQL digunakan untuk berkomunikasi dengan database. Menurut ANSI (American National Standards Institute), SQL merupakan bahasa standar untuk sistem manajemen database relasional. Perintah SQL digunakan untuk melakukan tugas-tugas seperti update data, atau mengambil data dari database. Beberapa sistem manajemen database relasional umum yang menggunakan SQL adalah: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, dan lain-lain. Meskipun sebagian besar sistem database menggunakan SQL, kebanyakan dari mereka juga memiliki ekstensi tambahan milik mereka sendiri yang biasanya hanya digunakan pada sistem mereka

Terdapat 2 jenis perintah SQL

1. Data definition language (DDL)

   DDL merupakan perintah SQL yang berhubungan langsung dengan pendefinisian struktur database. Beberapa perintah DDL antara lain :

   - CREATE
   - ALTER
   - RENAME
   - DROP

2. Data manipulation Language (DML)

   DML merupakan perintah sql yang berhubungan dengan manipulasi ,pengolahan data dalam tabel. Perintah SQL yang termasuk DML antara lain :

   - SELECT
   - INSERT
   - UPDATE
   - DELETE

2.7 SISTEM PAKAR

Sistem pakar (*expert sistem*) secara umum adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Atau dengan kata lain sistem pakar adalah sistem yang didesain dan diimplementasikan dengan bantuan bahasa pemrograman tertentu untuk dapat menyelesaikan masalah seperti yang dilakukan oleh para ahli. Diharapkan dengan sistem ini, orang awam dapat menyelesaikan masalah tertentu baik ‘sedikit’ rumit ataupun rumit sekalipun ‘tanpa’ bantuan para ahli dalam bidang tersebut. Sedangkan bagi para ahli, sistem ini dapat digunakan sebagai asisten yang berpengalaman [KUS-07].

Sistem pakar merupakan cabang dari *Artificial Intelligence* (AI) yang cukup tua karena sistem ini telah mulai dikembangkan pada pertengahan tahun 1960. Sistem

pakar yang muncul pertama kali adalah *General-purpose problem solver* (GPS) yang dikembangkan oleh Newl dan Simon. Sampai saat ini sudah banyak sistem pakar. yang dibuat, seperti MYCIN, DENDRAL, XCON & XSEL, SOPHIE, Prospector, FOLIO, DELTA, dan sebagainya [KUS-07].

Berikut perbandingan antara sistem konvensional dengan sistem pakar [KUS-07].:

a. Sistem Konvensional
    1. Informasi dan pemrosesan umumnya digabung dalam satu program *sequential.*
    2. Program tidak pernah salah (kecuali pemrogramnya yang salah).
    3. Tidak menjelaskan mengapa *input* dibutuhkan atau bagaimana hasil diperoleh.
    4. Data harus lengkap.
    5. Perubahan pada program merepotkan.
    6. Sistem bekerja jika sudah lengkap.
b. Sistem Pakar
    1. *Knowledge base* terpisah dari mekanisme pemrosesan *(inference)*
    2. Program bisa melakukan kesalahan
    3. Penjelasan *(explanation)* merupakan bagian dari ES
    4. Data tidak harus lengkap
    5. Perubahan pada *rules* dapat dilakukan dengan mudah
    6. Sistem bekerja secara heuristik dan logic.

Suatu sistem dikatakan sistem pakar apabila memiliki ciri-ciri sebagai berikut [KUS-07]:

1. Terbatas pada *domain* keahlian tertentu
2. Dapat memberikan penalaran untuk data-data yang tidak pasti
3. Dapat mengemukakan rangkaian alasan-alasan yang diberikannya dengan cara yang dapat dipahami
4. Berdasarkan pada kaidah atau *rule* tertentu

5. Dirancang untuk dikembangkan sacara bertahap

Adapun banyak manfaat yang dapat diperoleh dengan mengembangkan sistem pakar, antara lain [KUS-07]:

1. Masyarakat awam non-pakar dapat memanfaatkan keahlian di dalam bidang tertentu tanpa kesadaran langsung seorang pakar
2. Meningkatkan produktivitas kerja, yaitu bertambahnya *efisiensi* pekerjaan tertentu serta hasil solusi kerja.
3. Penghematan waktu dalam menyelesaikan masalah yang kompleks
4. Memberikan penyederhanaan solusi untuk kasus-kasus yang kompleks dan berulang-ulang
5. Pengetahuan dari seorang pakar dapat dikombinasikan tanpa ada batas waktu
6. Memungkinkan penggabungan berbagai bidang pengetahuan dari berbagai pakar untuk dikombinasikan.

Tujuan pengembangan sistem pakar sebenarnya bukan untuk menggantikan peran manusia, tetapi untuk mensubstitusikan pengetahuan manusia ke dalam bentuk sistem, sehingga dapat digunakan oleh orang banyak. Selain banyak manfaat yang diperoleh, ada juga kelemahan pengembangan sistem pakar, yaitu [KUS-07]:

1. Daya kerja dan produktivitas manusia menjadi berkurang karena semuanya dilakukan secara otomatis oleh sistem
2. Pengembangan perangkat lunak sistem pakar lebih sulit dibandingkan dengan perangkat lunak konvensional.

2.8 JavaScript Object Notation (JSON)

JSON adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 Desember 1999. JSON merupakan format teks yang tidak

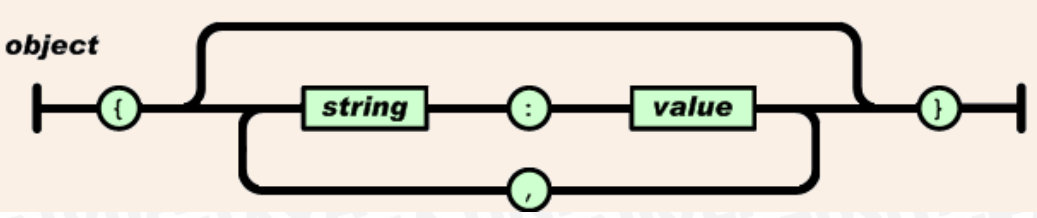
bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. Beberapa orang lebih suka JSON, karena paling mudah untuk mem-parse-nya, hanya menempatkan sebuah eval dan selesai sudah. JSON terbuat dari dua struktur [JSO-13]:

- Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (object), rekaman (record), struktur (struct), kamus (dictionary), tabel hash (hash table), daftar berkunci (keyed list), atau associative array.
- Daftar nilai terurutkan (an ordered list of values). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (array), vektor (vector), daftar (list), atau urutan (sequence).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini [JSO-13] .

Format Penulisan Object JSON

JSON menggunakan bentuk sebagai berikut: Object adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).



Gambar 2.9 Format penulisan object JSON
Sumber : [JSO-13]

Kelebihan penggunaan JSON :

Karena kesederhanaan penulisan JSON, ukuran file yang dihasilkanpun menjadi lebih ramping dibandingkan XML (karena terus mengulangi kata yang sama dalam tag-tag yang digunakan). Hal ini berimbas pada kecepatan loading transfer data, pertukaran data JSON lebih cepat jika dibandingkan dengan XML.

2.9 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan elemen yang kritis dari SQA dan merepresentasikan tinjauan ulang yang menyeluruh terhadap spesifikasi, desain dan pengkodean. Ujicoba merepresentasikan ketidak normalan yang terjadi pada pengembangan software. Selama definisi awal dan fase pembangunan, pengembang berusaha untuk membangun software dari konsep yang abstrak sampai dengan implementasi yang memungkinkan. Para pengembang membuat serangkaian uji kasus yang bertujuan untuk ”membongkar” software yang mereka bangun. Pengembang software secara alami merupakan orang konstruktif. Ujicoba yang diperlukan oleh pengembang adalah untuk melihat kebenaran dari software yang dibuat dan konflik yang akan terjadi bila kesalahan tidak ditemukan. Dari sebuah buku, Glen Myers menetapkan beberapa aturan yang dapat dilihat sebagai tujuan dari ujicoba [AYL-11:1]:

- Ujicoba merupakan proses eksekusi program dengan tujuan untuk menemukan kesalahan.
- Sebuah ujicoba kasus yang baik adalah yang memiliki probabilitas yang tinggi dalam menemukan kesalahan-kesalahan yang belum terungkap.
- Ujicoba yang berhasil adalah yang mengungkap kesalahan yang belum ditemukan.

Blackbox Testing

Metode ujicoba blackbox memfokuskan pada keperluan fungsional dari software. Karena itu ujicoba blackbox memungkinkan pengembang software untuk

membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba blackbox bukan merupakan alternatif dari ujicoba whitebox, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode whitebox. Ujicoba blackbox berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya[ALY-11:1] :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan performa
5. kesalahan inisialisasi dan terminasi

Whitebox Testing

Ujicoba Whitebox merupakan metode desain uji kasus yang menggunakan struktur kontrol dari desain prosedural untuk menghasilkan kasus-kasus uji. Dengan menggunakan metode ujicoba whitebox, para pengembang software dapat menghasilkan kasus-kasus uji yang [AYL-11:3] :

- Menjamin bahwa seluruh independent paths dalam modul telah dilakukan sedikitnya satu kali,
- Melakukan seluruh keputusan logikal baik dari sisi benar maupun salah,
- Melakukan seluruh perulangan sesuai batasannya dan dalam batasan operasionalnya
- Menguji struktur data internal untuk memastikan validitasnya

Basis Path Testing

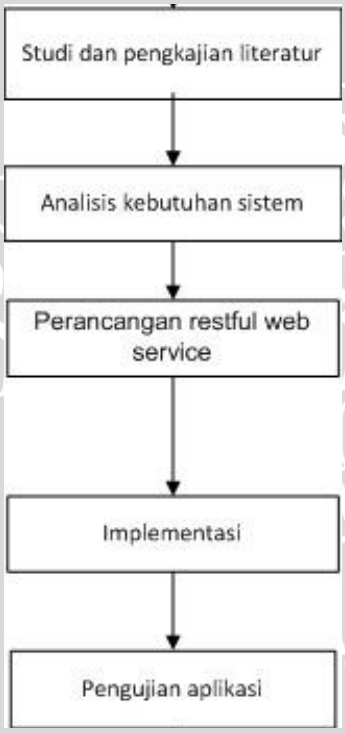
Pengujian basis path merupakan teknik ujicoba whitebox pertama yang diusulkan oleh Tom McCabe. Metode berbasis alur memungkinkan perancang kasus uji untuk menghasilkan ukuran kompleksitas logikal dari desain prosedural dan menggunakan ukuran ini untuk mendefinisikan himpunan basis dari alur eksekusi.

Kasus uji dihasilkan untuk melakukan sekumpulan basis yang dijamin untuk mengeksekusi setiap perintah dalam program, sedikitnya satu kali selama ujicoba.

1) Rancangan atau kode program digunakan sebagai dasar, kemudian buat *flow graphnya*.
2) Tentukan kompleksitas siklomatik dari *flow graph* yang dibuat. *Cyclomatic complexity* adalah metriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metriks ini digunakan dalam konteks metode pengujian *Basis Path*, maka nilai yang terhitung untuk *cyclomatic complexity* menentukan jumlah jalur independen (*independent path*) dalam *basis set* suatu program dan memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua *statement* telah dieksekusi sedikitnya satu kali. Jalur independen adalah jalur yang melalui program yang mengenalkan sedikitnya satu rangkaian *statement* proses baru atau suatu kondisi baru. Untuk menentukan *cyclomatic complexity* bisa dilakukan dengan beberapa cara, diantaranya [AYL-11]:
   a) Jumlah *region* pada *flow graph* sesuai dengan *cyclomatic complexity*.
   b) *Cyclomatic complexity* V(G), untuk grafik G adalah V(G) = E – N + 2, dimana E adalah jumlah *edge*, dan N adalah jumlah *node.* Tentukan himpunan basis dari jalur – jalur yang independen secara linear.
3) Membuat independent path, Independent path adalah alur manapun dalam program yang memperkenalkan sedikitnya satu kumpulan perintah pemrosesan atau kondisi baru. Contoh independent path:
   a. Path 1 : 1 – 11
   b. Path 2 : 1 – 2 – 3 – 4 – 5 – 10 – 1 – 11
   c. Path 3 : 1 – 2 – 3 – 6 – 8 – 9 – 10 – 1 – 11
   d. Path 4 : 1 – 2 – 3 – 6 – 7 – 9 – 10 – 1 – 11
4) Membuat test case.

BAB III
METODOLOGI PENELITIAN

Metode penelitian membahas metode dan tahapan yang digunakan dalam pengembangan aplikasi RESTful *web service* sistem pakar perawatan cabai. Metode metode yang digunakan meliputi studi literatur, analisis kebutuhan sistem, perancangan data, perancangan perangkat lunak, implementasi, pengujian dan pengambilan kesimpulan.



Gambar 3.1 *Flowchart* metode penelitian
Sumber : Perancangan

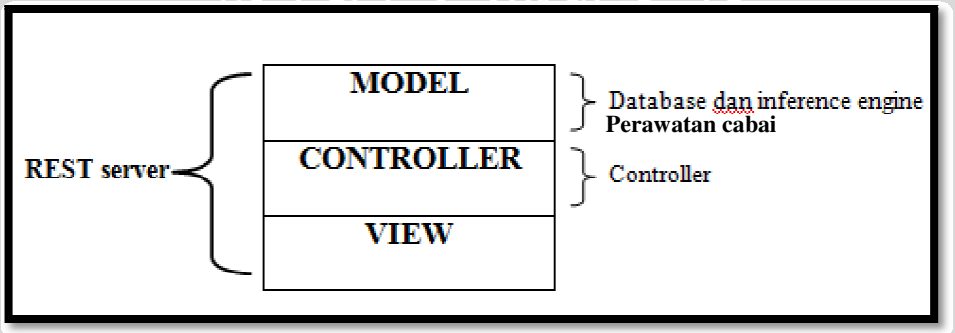
3.1 Studi Literatur

Dalam perancangan dan implementasi penelitian ini, perlu diadakan studi literatur. Literatur digunakan sebagai teori penguat dan landasan dasar dalam penelitian. Literatur yang akan digunakan meliputi hal berikut :

a. Penelitian terdahulu
b. Sistem Pakar
c. *Web service*
d. Arsitektur REST
e. HTML
f. MySQL
g. JSON
h. Rekayasa perangkat Lunak

3.2 Analisis Kebutuhan Sistem

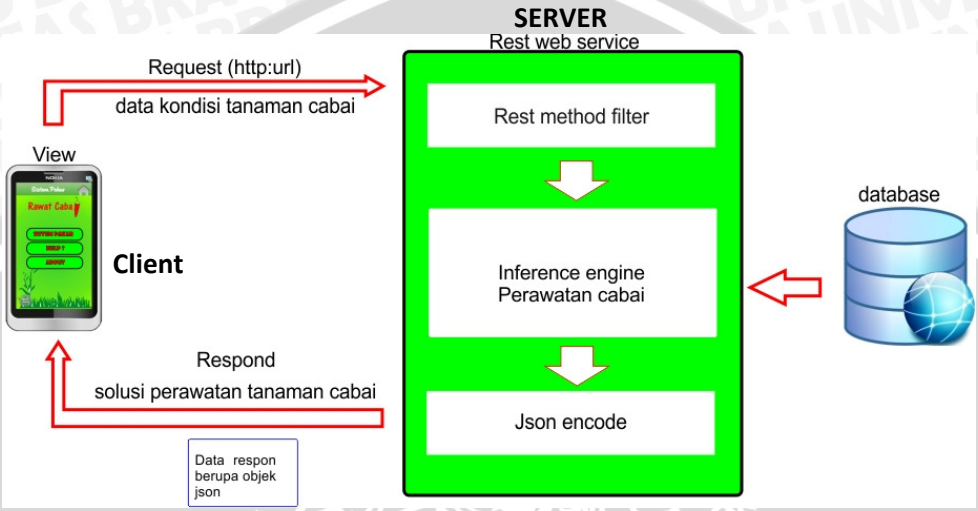
Metode analisis kebutuhan sistem dalam aplikasi ini menggunakan permodelan DFD (Data Flow Diagram). DFD menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data baik secara manual maupun komputerisasi.



Gambar 3.2 Arsitektur layer Rest Server aplikasi *webservice* perawatan cabai
Sumber : Perancangan

REST *server* terdiri dari tiga layer, yaitu model, controller dan view. Layer Controller berfungsi untuk menerima dan merespon request yang diterima dari *client* serta sebagai pengontrol aliran data. Layer model berisi interaksi *server* dengan database serta inferensi engine sistem pakar. View berfungsi untuk meletakkan tampilan, namun dalam aplikasi yang dibangun layer view tidak terlalu digunakan dalam *server* namun diletakkan di sisi *client* karena orientasi dari *web service* hanya layanan. View di *server* hanya menampilkan data data dalam bentuk objek JSON.

Perangkat lunak *web service* diuji dengan menggunakan *client*. *Client* merupakan aplikasi yang merequest layanan dari *web service*. *Client* sebagai platform yang berinteraksi dengan pengguna aplikasi secara langsung melalui user interface. Arsitektur hubungan antara *web service* dan *client* sebagai berikut.



Gambar 3.3 Arsitektur sistem pakar dengan rest *web service* dengan client
Sumber : Perancangan

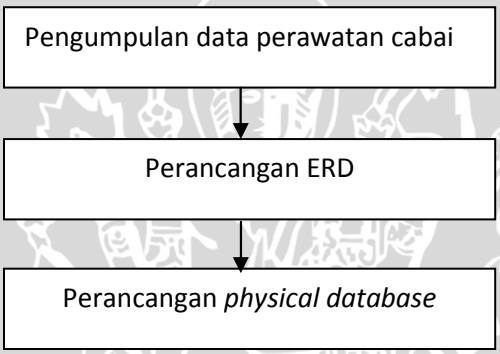
Alur komunikasi antara client dengan server akan dijelaskan sebagai berikut :

1. User / pengguna aplikasi yaitu penyuluh memasukkan kondisi tanaman cabai sesuai pertanyaan yang ditampilkan oleh aplikasi klien.
2. Hasil input kondisi tanaman cabai oleh klien disatukan menjadi sebuah *string*.
3. Klien membuka komunikasi dengan server menggunakan AJAX.
4. String kondisi tanaman cabai dikirimkan klien ke server menggunakan AJAX dengan HTTP method GET.
5. Data string kondisi tanaman cabai diterima server.
6. Server mengolah data string kondisi tanaman cabai di mesin inferensi menjadi sebuah solusi.
7. Solusi di *encode* menjadi objek JSON.
8. Server mengirimkan respon berupa solusi dalam bentuk objek JSON kembali ke klien

9. Klien menangkap objek JSON dari server.
10. Klien menerjemahkan objek json menjadi string solusi.
11. String solusi ditampilkan oleh klien dalam bentuk user interaface sehingga dapat dilihat oleh user.

3.3 Perancangan Data

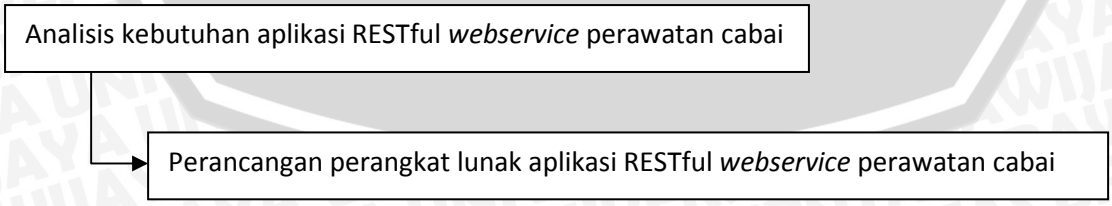
Data yang dibutuhkan dalam penelitian ini adalah data tentang perawatan tanaman cabai atau disebut knowledge. Sumber data didapat dari penelitian terdahulu tentang sistem pakar perawatan caba. Data dirancang menngunakan perancangan Entity Relationship Diagram (ERD) dan diagram psycal database.



Gambar 3.4 Gambaran umum perancangan data perawatan cabai
Sumber : Perancangan

3.4 Perancangan Perangkat Lunak

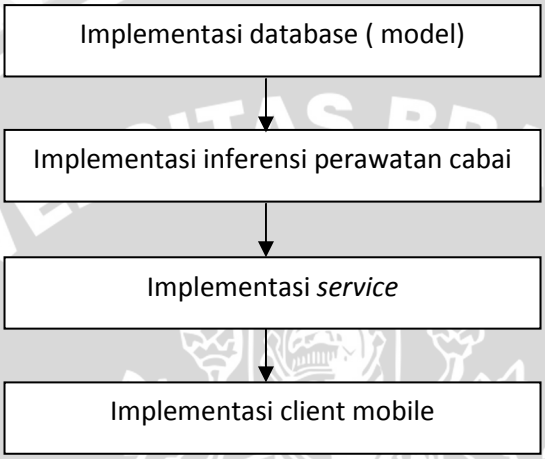
Perancangan sistem berfungsi untuk mendefenisikan secara rinci kebutuhan dari sistem aplikasi sistem pakar perawatan cabai berbasis restful *web service* sebagai acuan pembangunan aplikasi.



Gambar 3.5 Gambaran umum perancangan perangkat lunak Restful *webservice* perawatan cabai
Sumber : Perancangan

3.5 Implementasi

Implementasi merupakan pembangunan aplikasi berdasarkan pada rancangan perangkat lunak yang telah dibuat pada tahap perancangan.



Gambar 3.6 Gambaran umum implementasi perangkat lunak Restful *webservice* perawatan cabai
Sumber : Perancangan

3.6 Metode Pengujian dan Analisis

Pengujian sistem dilakukan untuk menguji fungsionalitas, pengujian REST *web service* dan perbandingan kecepatan akses antara *web* aplikasi sistem pakar dengan *web* aplikasi sistem pakar berbasis *web service*.

Strategi pengujian fungsionalitas yang digunakan adalah pengujian validasi dan unit. Uji validasi berfungsi untuk menguji kerja setiap fitur aplikasi. Uji validasi menggunakan pendekatan *blackbox*. *Black box* bertujuan untuk menguji masukan dan keluaran data apakah sesuai dengan harapan. Pengujian unit dilakukan untuk mengukur kopleksitas suatu *method*. Pengujian unit menggunakan pendekatan white box.

Pengujian kecepatan digunakan untuk membuktikan apakah aplikasi optimalisasi kecepatan aplikasi *web* dengan sistem pakar bekerja atau tidak.

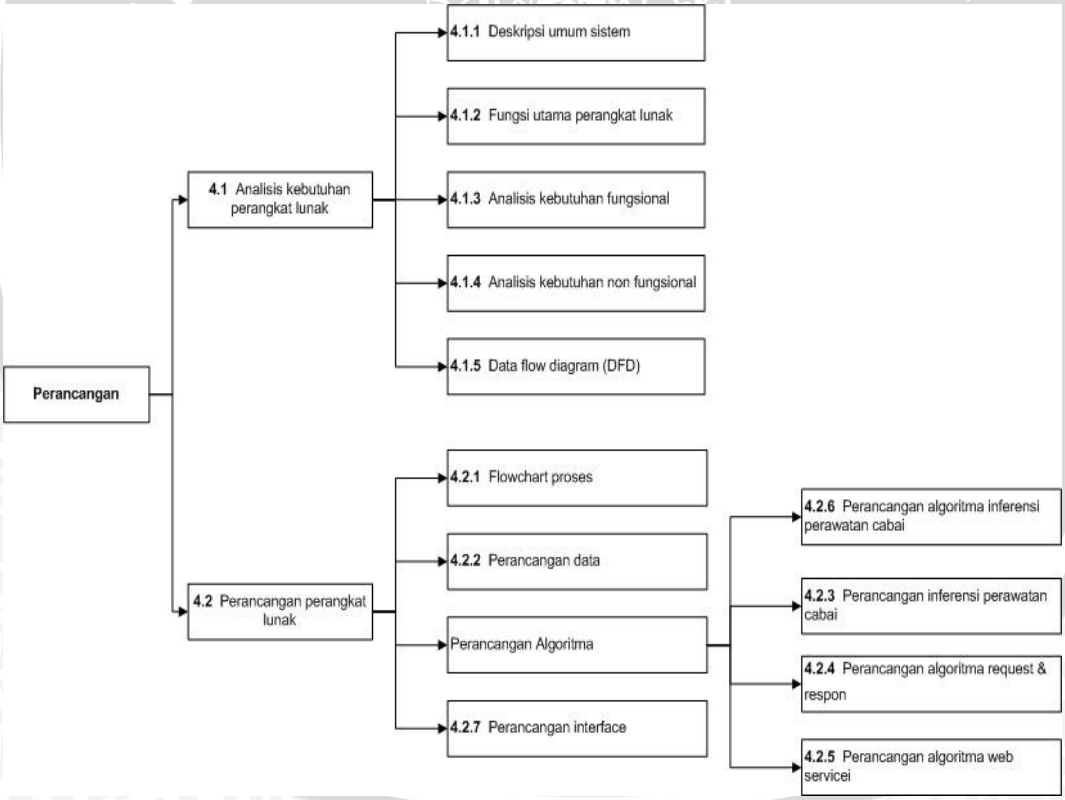
3.7 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Kesimpulan yang diambil untuk membuktikan apakah aplikasi yang dibangun dengan *web service* memiliki lebih cepat diakses dibandingkan dengan *web* standar.

Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dalam pembuatana RESTfull *web service* untuk aplikasi sistem pakar perawatan cabai dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

BAB IV
PERANCANGAN

Perancangan aplikasi RESTful *web service* sistem pakar perawatan cabai. Perancangan terdiri dari dua tahap. Tahap pertama analisis kebutuhan perangkat lunak terdiri dari deskripsi dan tujuan dari aplikasi, analisis kebutuhan fungsional, analisis kebutuhan non fungsional dan data flow diagram. Tahap kedua perancangan perangkat lunak terdiri dari struktur aliran proses sistem, perancangan data, perancangan inferensi sistem pakar perawatan cabai, perancangan algoritma dan perancangan *user interface*.



Gambar 4.1 Pohon perancangan
Sumber : Perancangan

4.1 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak pada perangkat lunak dimaksudkan untuk memenuhi kebutuhan perangkat lunak agar dapat memenuhi kebutuhan pengguna aplikasi. Kebutuhan lingkungan yang digunakan dalam pembuatan penelitian ini meliputi :

- Kebutuhan Hardware :
  - Mobile sebagai client (simulator)
- Kebutuhan Software :
  - Linux sebagai sistem operasi
  - MySQL sebagai server Database Management System
  - Web hosting sebagai hosting service.
- Data yang dibutuhkan :
  - Data knowledge tentang perawatan tanaman cabai keriting

4.1.1 Deskripsi Umum Sistem

Aplikasi yang akan dibangun dibagi menjadi dua sisi / bagian. Bagian pertama dan yang utama yaitu sisi *server,* akan dibangun menjadi sebuah aplikasi *RESTfull web service* dengan layanan sistem pakar perawatan cabai. Web service dibangun dengan pemograman php dan menggunakan framework code igniter.

Sisi kedua yaitu *client*, dibangun sebagai aplikasi yang menggunakan *service* dari sisi *server*. *Client* dibangun menjadi sebuah aplikasi *mobile web apps*. Client dibangun menggunakan html,css dan javascript.

4.1.2 Fungsi Utama Perangkat Lunak

Fungsi Server / Webservice :

- Service menerima dan merespon request dari client
- Service dapat memberikan solusi / inferenseial dari fakta yang diberilan client

Fungsi Client

- Menerima inputan fakta kondisi tanaman dari user
- Mengirim fakta dan menerima solusi perawatan cabai dari *web service*

4.1.3 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional digambarkan dalam bentuk tabel berikut.

Tabel 4.1 Data kebutuhan fungsional aplikasi

Kode	Kebutuhan fungsional	Keterangan	Sisi
SRS_001	User memilih menu yang disediakan aplikasi sistem pakar	Merupakan tampilan awal	Client
SRS_002	User menginputkan data fakta tentang kondisi tanaman cabai	Sistem memberikan pertanyaan terkait kondisi tanaman cabai dan dijawab oleh user	Client
IRS_001	Perangkat lunak menerima input pilihan kondisi tanaman cabai		Client
SRS_003	Aplikasi client mengakses dan mengirim data request yang berisi data fakta kondisi tanaman ke server	Data fakta kondisi tanaman cabai yang telah diinputkan oleh user pada SRS_002	Client
SRS_004	Client dapat Binding Sever	Bind melalui URI (sesuai kaidah REST)	
SRS_005	Server dapat menerima request dari client dan merespon kembali ke client	Request di seleksi berdasarkan method request (sesuai kaidah arsitektur REST)	Server
SRS_006	Server dapat mengolah data request	Data request SRS_003 diolah oleh mesin inferensi menjadi sebuah solusi	Server
SRS_007	Client dapat menerima respon dari server	Respon berupa objek yang dikirimkan oleh server	Client
IRS_002	Aplikasi menampilkan solusi perawatan tanaman cabai		Client

Sumber : Perancangan

User atau pengguna aplikasi adalah penyuluh pertanian yang dijadikan target pengguna aplikasi sistem pakar REST *web service* .

4.1.4 Analisis kebutuhan non fungsional

Daftar kebutuhan non fungsional dijabarkan pada Tabel 4.2

Tabel 4.2 Data kebutuhan non fungsional

Parameter	Deskripsi Kebutuhan
Avaliability	Aplikasi ini harus dapat beroperasi selama waktu yang ditentukan.
Response Time	Aplikasi ini harus cepat dalam melakukan proses pencarian solusi perawatan tanaman cabai. Ini menjadi acuan keberhasilan sesuai dengan tujuan penelitian
Security	Aplikasi ini harus aman,User hanya bisa mengakses aplikasi client yang hanya berisi tampilan.
Memory	Bandwith yang dibutuhkan dalam mengakses aplikasi harus lebih ringan dibandingkan studi kasus awal

Sumber : Perancangan

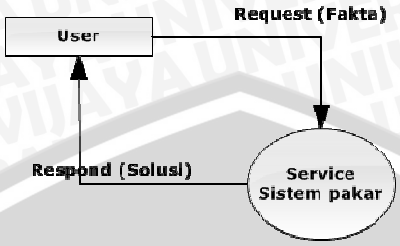
Fokus utama dari kebutuhan non fungsional yang menjadi parameter dalam penelitian ini adalah *response time* dan *memory.* Parameter Respon time berkaitan dengan kecepatan akses aplikasi. *Respon time* aplikasi yang dibangun harus lebih cepat dibandingkan dengan aplikasi terdahulu yang telah dibahas dalam bab latar belakang penelitian. Parameter memory berkaitan dengan ukuran total ukuran request yang dilakukan aplikasi klien ke *web service*. Untuk mengetahui apakah parameter non fungsional tersebut terpenuhi, dilakukan dengan pengujian kecepatan akses yang akan dibahas pada bab pengujian perangkat lunak.

4.1.5 Data flow diagram

Data flow diagram digunakan untuk menggambarkan aliran data aplikasi sistem pakar perawatan cabai berbasis RESTfull *web service*. DFD aplikasi terdiri dari *context diagram* , diagram level 0 dan level 1. Diagram *context* menggambarkan secara garis besar proses, aliran dan entitas aplikasi. Diagram level 0 dan 1 menggambarkan detail dari diagram *context.*

Dibawah ini digambarkan diagram *context* dari aplikasi sistem pakar perawatan cabai berbasis RESTfull webservice.

Diagram context

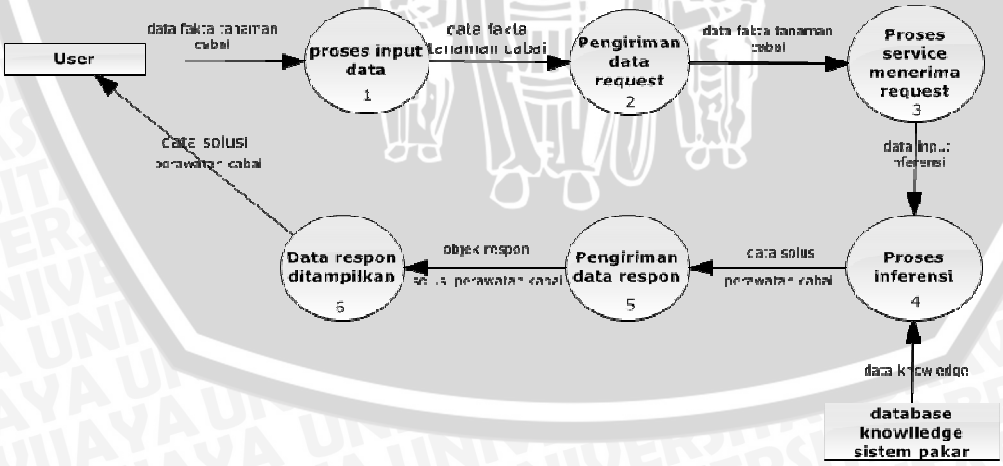


Gambar 4.2 DFD context diagram
Sumber : Perancangan

Gambar 4.2 menggambarkan diagram context aplikasi. Entitas yang berhubungan hanya satu yaitu user. User adalah pengguna aplikasi yang membutuhkan solusi tentang perawatan cabai keriting. User mengirimkan request melalui aplikasi client ke service sistem pakar. Service sistem pakar akan mengolah request dari user dan mengirimkan respon berupa solusi perawatan cabai ke aplikasi client dan ditampilkan oleh client agar dapat dilihat oleh user.

Data Flow Diagram Level 0

Data Flow Diagram Level 0 merupakan perincian dari context diagram. Dalam DFD level 0 penelitian ini, terdapat 6 proses yang akan dilakukan oleh sistem.



Gambar 4.3 Data flow digram level 0 aplikasi
Sumber : Perancangan

Penjelasan dari enam proses pada gambar 4.3 akan dijabarkan pada tabel dibawah.

Tabel 4.3 Penjelasan DFD level 0

No	Nama Proses	Keterangan
1	Proses data input	User memasukkan data inputan berupa fakta kondisi tanaman cabai. Prosedur input fakta yaitu sistem memberikan pertanyaan seputar kondisi tanamn cabai dan user menjawabnya sesuai kondisi tanaman cabainya.
2	Pengiriman data request	Data inputan yang telah diinput oleh user sebelumnya akan dikirimkan menuju *web service* dalam bentuk request.
3	Proses penerimaan data request	*Web service* menerima request yang berisi data inputan fakta kondisi tanaman cabai
4	Proses inferensi	Data kondisi tanaman cabai dimasukkan kedalam mesin inferensi untuk dicarikan solusinya. Proses inferensi membutuhkan data dari database knowledge sistem pakar.
5	Proses pengiriman data respon	Hasil solusi dari proses inferensi akan dikirimkan dalam bentuk respon objek kembali ke sistem client
6	Proses tampil data respon	Client menampilkan data respon dengan user interface agar dapat dipahami oleh user

Sumber : Perancangan

Data Flow Diagram (DFD) level 1

Secara lebih mendetail, masing-masing Data Flow Diagram (DFD) level 0 akan dijabarkan dalam DFD level 1. Di dalam DFD Level 1, akan diberikan uraian nama DFD, input dari proses , output dari proses dan keterangan bagaimana proses berlangsung.

DFD level 1–1

Gambar 4.4 DFD level 1-1
Sumber : Perancangan

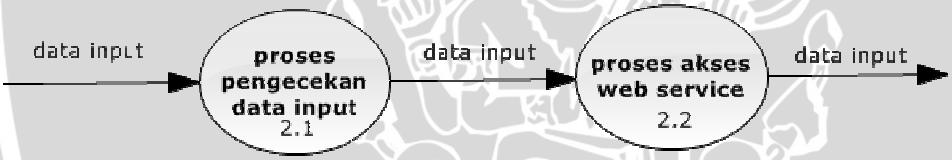
Client mobile menampilkan *user interface* data fakta tanaman cabai. Data fakta tanaman cabai ditampilkan kepada user dalam bentuk pertanyaaan. Pertanyaan dijawab oleh user. Jawaban kemudian dikirim ke service server sebagai data input.

Tabel 4.4 Penjelasan DFD level 1-1

No	Proses	Input	Keterangan proses	Output
1.1	Tampil pertanyaan	Data fakta tanaman cabai	Sistem akan menampilkan pertanyaan seputar kondisi tanaman cabai mulai dari kondisi tanaman, daun, buah dan cirri lainnya.	Jawaban
1.2	Tampil jawaban	Jawaban	Sistem menampilkan pilihan jawaban yang berhubungan dengan pertanyaan	Data input

Sumber : Perancangan

DFD Level 1-2



Gambar 4.5 DFD Level 1-2
Sumber : Perancangan

Data input akan dicek terlebih dahulu sebelum diteruskan menuju server. Pengecekan dilakukan untuk mengetahui apakah data input yang akan dikirim ke server sudah lengkap.Data dikirimkan ke serverdalam bentuk string.

Tabel 4.5 Penjelasan DFD level 1-2

No	Proses	Input	Keterangan proses	Output
2.1	Proses pengecekan data input	Data input	Sistem akan mengecek data input apakah data input sudah terisi semua atau ada yang terlewat. Jika ada data yang tidak diisi maka diset default nilai 0	Data input
2.2	Proses akses *web*	Data input	Client akan mengakses	Data input

	service		URI dari *web service* kemudian mengirimkan data input dengan method GET	

Sumber : Perancangan

DFD Level 1-3

Gambar 4.6 DFD Level 1-3
Sumber : Perancangan

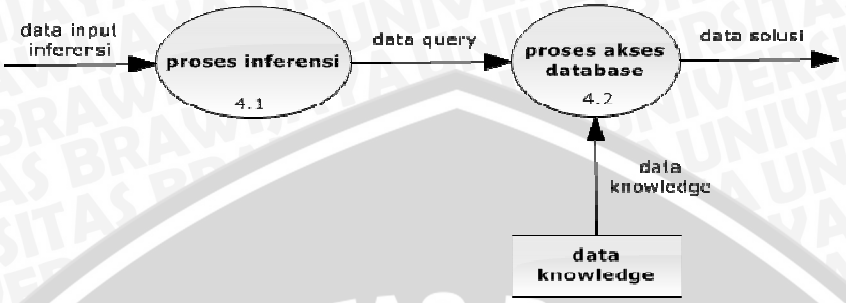
Data input client diterima di server. Server kemudian memfilter method input sesuai *http method* yang digunakan oleh client. Jika method yang digunakan GET maka data input diarahkan menuju inferensi. Data yang telah difilter kemudian dipecah dari string menjadi beberapa bagian charater.

Tabel 4.6 Penjelasan DFD level 1-2

No	Proses	Input	Keterangan proses	Output
3.1	Proses pemfilteran method request	Data input	Sistem akan memfilter tipe method request yang dilakukan oleh client dan mengarahkannya sesuai method terpilih	Data input
3.2	Proses persiapan data	Data input	Data akan dipecah dari bentuk objek string menjadi unit unit kecil sesuai parameter fungsi inferensi	Data input inferensi

Sumber : Perancangan

DFD Level 1-4



Gambar 4.7 DFD Level 1-4
Sumber : Perancangan

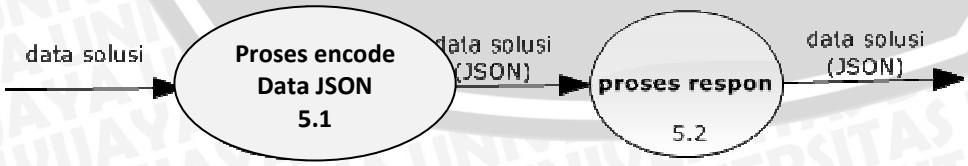
Data input digunakan oleh inferensi sebagai argumen method. Inferensi akan melakukan proses pencarian solusi dengan method forward chaining. Data yang dibutuhkan sebagai knowledge diambil dari database. Hasil inferensi berupa data solusi perawatan cabai.

Tabel 4.7 Penjelasan DFD level 1-4

No	Proses	Input	Keterangan proses	Output
4.1	Proses Inferensi	Data input inferensi	Proses inferensi akan menghasilkan sebuah solusi namun dalam bentuk object query	Data query
4.2	Proses akses database	Data query	Data query dieksekusi ke database knowledge dan menghasilkan data solusi berbentuk object array	Data solusi

Sumber : Perancangan

DFD Level 5-1



Gambar 4.8 DFD Level 1-5
Sumber : Perancangan

Data solusi yang berupa string di encode menjadi objek JSON. Objek JSON solusi perawatan cabai dikirimkan kembal ke client sebagai respon.

Tabel 4.8 Penjelasan DFD level 1-5

No	Proses	Input	Keterangan proses	Output
5.1	Proses encode data solusi	Data solusi	Proses mengkonversi data solusi array menjadi object JSON	Data solusi (JSON)
5.2	Proses Respon	Data solusi (JSON)	Data solusi dikirimkan dalam bentuk objek JSON sebagai respon dari *web service* ke client	Data solusi (JSON)

Sumber : Perancangan

DFD Level 6-1

Gambar 4.9 DFD Level 1-6
Sumber : Perancangan

Respon dari server di terima oleh client. Respon yang berupa objek JSON perawatan cabai didecode menjadi bentuk array. Data array perawatan cabai ditampilkan sesuai user interface ke user.

Tabel 4.9 Penjelasan DFD level 1-6

No	Proses	Input	Keterangan proses	Output
6.1	Proses decode data solusi	Data solusi	Proses mengkonversi data solusi object JSON menjadi array	Data solusi
6.2	Proses tampil	Data solusi	Data solusi ditampilkan dengan user interface yang dipahami oleh user	Data solusi

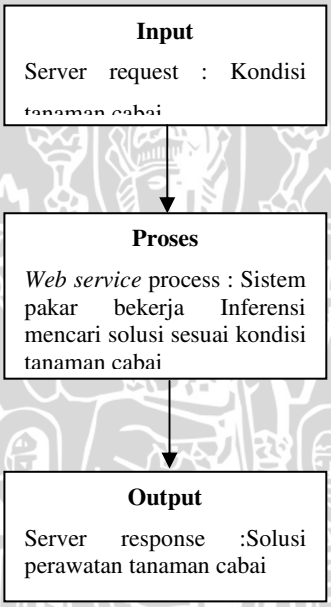
Sumber : Perancangan

4.2 Perancangan Perangkat Lunak

Bagian perancangan akan menjelaskan perancangan perangkat lunak sesuai kebutuhan yang sudah dideskripsikan pada bab analisis kebutuhan sistem.

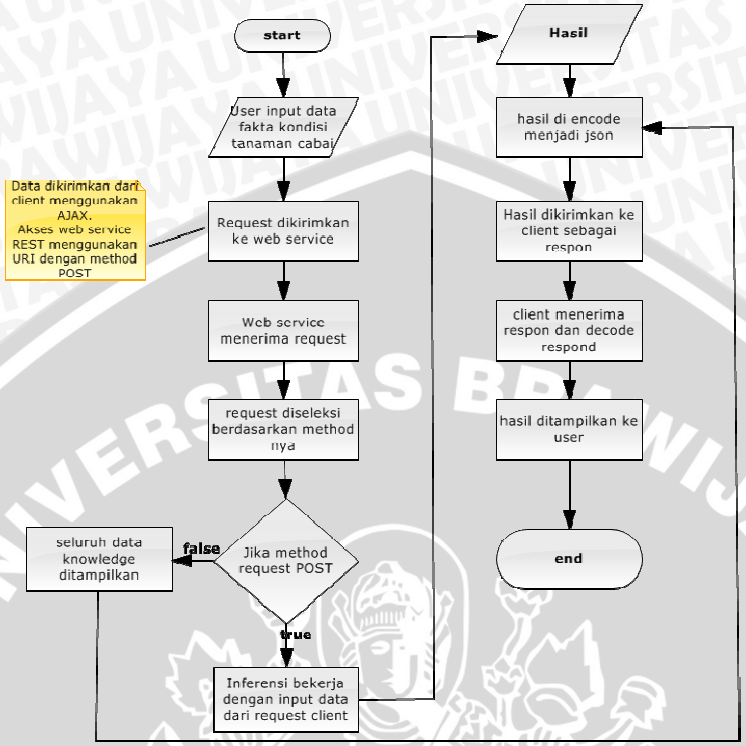
4.2.1 Flowchart proses

Blok diagram sistem pada gambar dibawah, menggambarkan bagian bagian dari fungsi sistem yang saling berinteraksi antara satu blok dengan blok lain. Fungsi utama sistem adalah mendapatkan solusi perawatan cabai denga sistem pakar dengan menggunakan *web service*.



Gambar 4.10 Input dan output sistem
Sumber : Perancangan

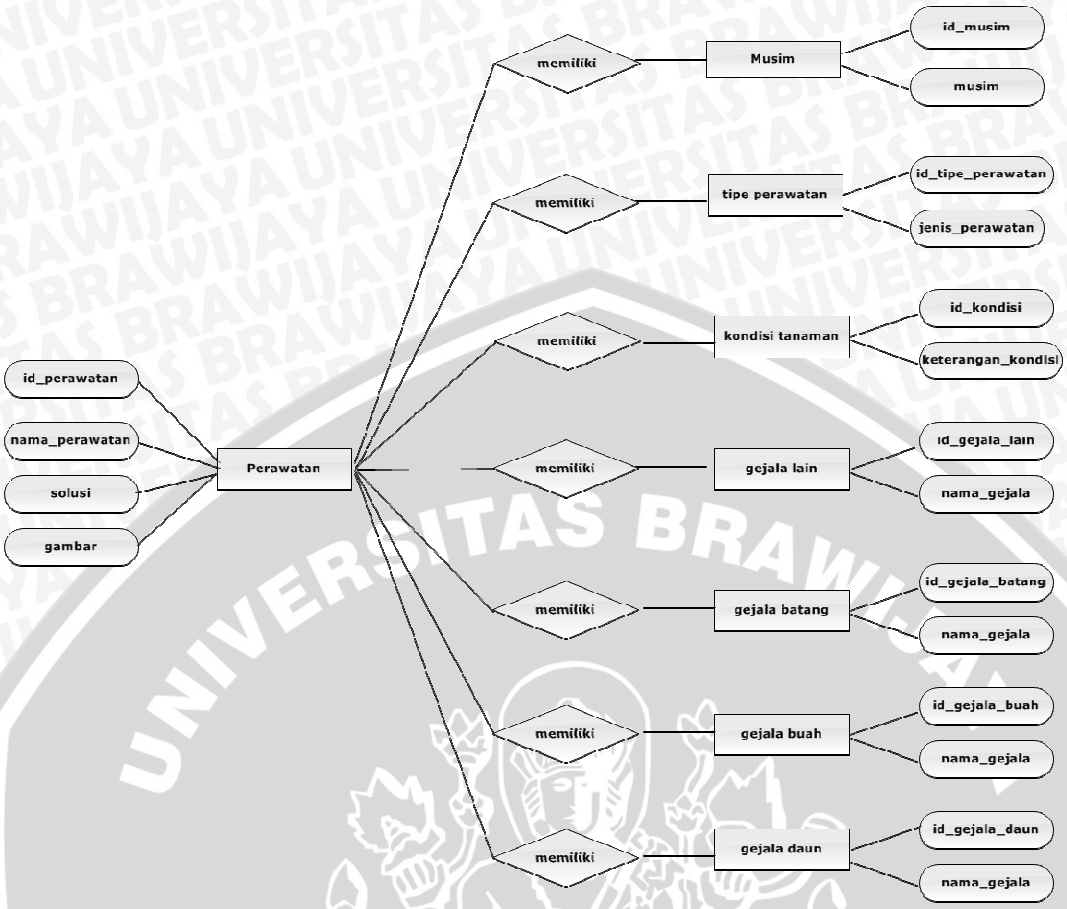
Dari blok diagram diatas, aliran proses dari setiap fungsi dijelaskan menggunakan permodelan flowchart proses berikut.



Gambar 4.11 Flowchart proses sistem
Sumber : Perancangan

4.2.2 Perancangan Data

Data merupakan bagian terpenting dalam pembangunan perangkat lunak sistem pakar, karena data adalah otak bagi sistem pakar. Didalam aplikasi sistem pakar perawatan cabai ini, data digambarkan sebagai sebuah pengetahuan (*knowledge)* mengenai perawatan cabai. Data ini diperoleh melalui dua cara, pertama data primer yang didapat dari pakar dan kedua data sekunder yang didapat dari referensi buku dan situs pertanian. Namun, dalam penelitian ini data didapat dari penelitian sebelumnya yang menjadi studi kasus. Data pengetahuan kemudian disimpan dalam database. Pemetaan data didalam database direpresentasikan dalam bentuk permodelan ERD (entity relationship diagram). Struktur data dari aplikasi sistem pakar ditunjukan pada ERD berikut :



Gambar 4.12 Entity relationship diagram sistem
Sumber : Perancangan

Tabel yang dibuat terdiri dari :

a). Tabel gejala batang

b). Tabel gejala buah

c). Tabel gejala daun

d). Tabel gejala lain

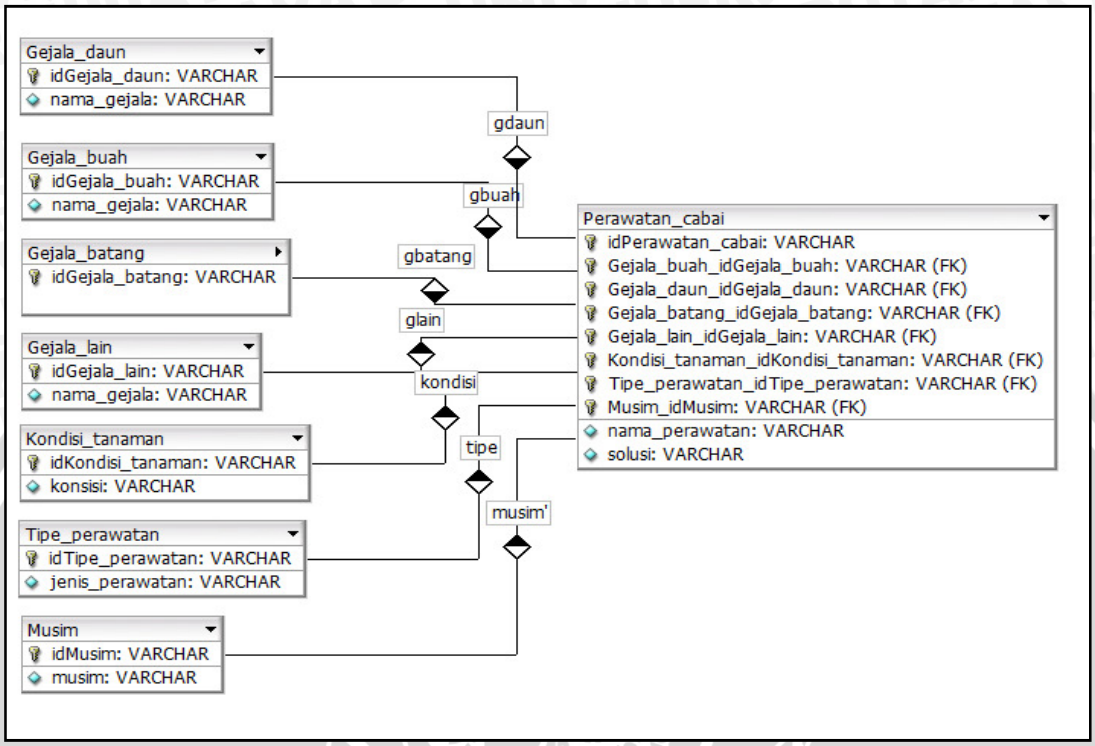
e). Tabel kondisi tanaman

f). Tabel musim

g). Tabel perawatan cabai

h). Tabel tipe perawatan cabai

Physical Diagram dirancang berdasarkan ERD pada gambar 4.12



Gambar 4.13 *Physical diagram* sistem
Sumber : Perancangan

4.2.3 Perancangan Inferensi Sistem pakar

Inferensi dibangun berdasarakan pengetahuan yang direpresentasikan menggunakan permodelan tree. Inferensi didapat dari studi kasus penelitian sebelumnya. Aturan inferensi dijabarkan pada tabel berikut :

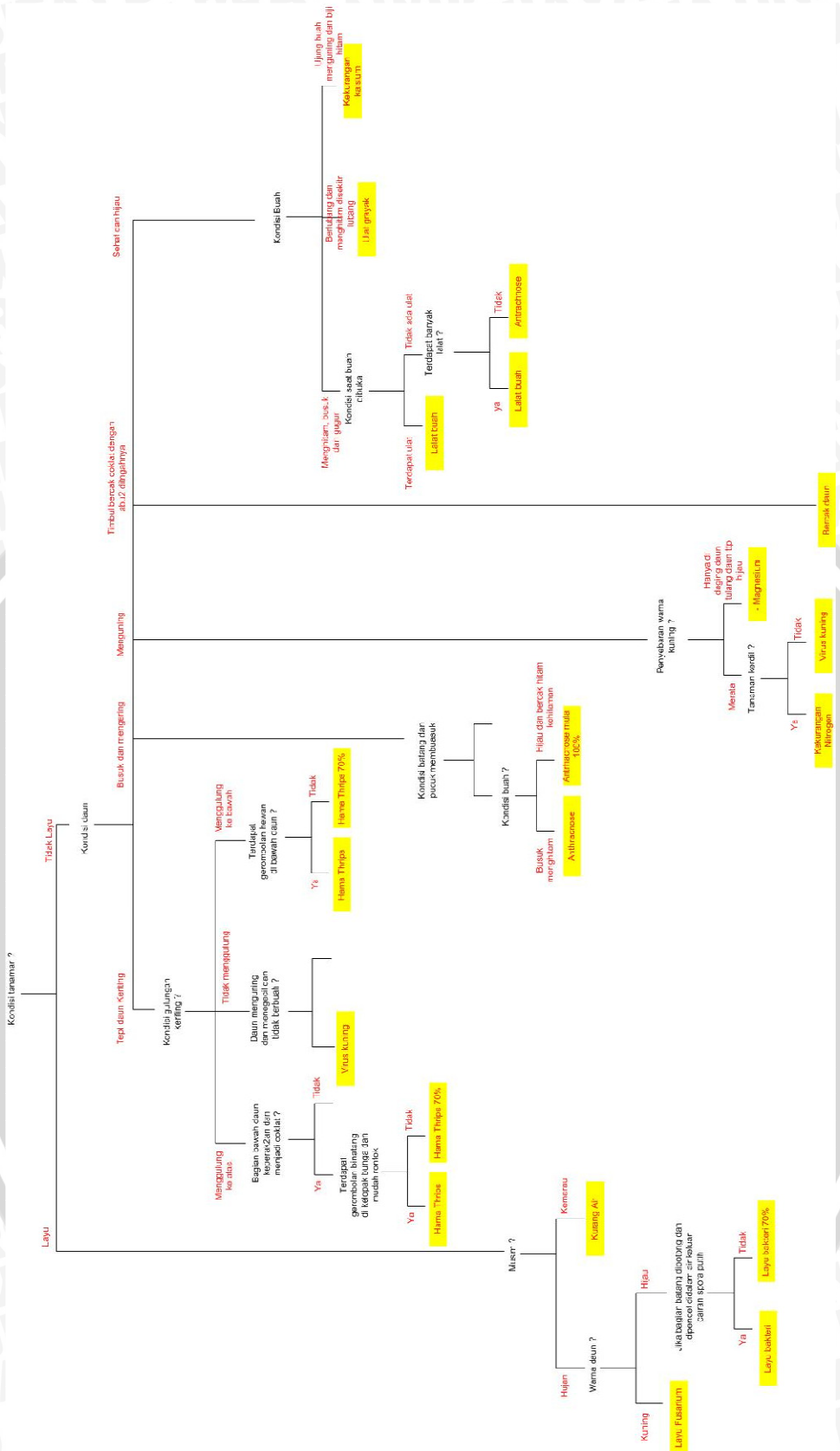
Tabel 4.10 Tabel aturan inferensi sistem pakar perawatan cabai

No aturan	Aturan	Kesimpulan
R01	IF tanaman layu AND musim hujan AND daun berwarna kuning	Layu Fusarium
R02	IF tanaman layu AND musim hujan AND daun berwarna hijau	Layu Bakteri
R03	IF tanaman rusak AND tepi daun keriting AND menggulung keatas	Hama Thrips
R04	IF tanaman rusak AND tepi daun keriting AND menggulung kebawah	Hama Tungau
R05	IF tanaman rusak AND tepi daun	Virus Kuning

	keriting AND menguning tidak menggulung	
R06	IF tanaman rusak AND daun membusuk mengering AND batang buah busuk	Antrachnose
R07	IF tanaman rusak AND daun menguning AND kuning hanya ditulang daun	Kekurangan Magnesium
R08	IF tanaman rusak AND daun menguning AND kuning merata AND tanaman kerdil	Kekurangan Nitrogen
R09	IF tanaman rusak AND timbul bercak coklat didaun	Bercak daun
R10	IF tanaman rusak AND daun masih hijau AND buah merah menghitam busuk	Lalat buah
R11	IF tanaman rusak AND daun masih hijau AND buah berlubang menghitam	Ulat grayak
R12	IF tanaman rusak AND daun masih hijau AND ujung buah menguning dan biji hitam	Kekurangan kalsium

Sumber : Perancangan

Aturan kemudian direpresentasikan dengan metode forward chaining atau runut maju menjadi sebuah *decision tree. Decision tree* adalah pemetaan mengenai alternatif-alternatif pemecahan masalah yang dapat diambil dari masalah tersebut. Pohon tersebut juga memperlihatkan faktor-faktor kemungkinan/probablitas yang akan mempengaruhi alternatif-alternatif keputusan tersebut, disertai dengan estimasi hasil akhir yang akan didapat bila kita mengambil alternatif keputusan tersebut. Decision tree dirancang berdasarkan tabel aturan 4.10.



Gambar 4.14 Representasi pengetahuan dengan permodelan tree

Sumber : Perancangan

4.2.4 Perancangan algoritma request & respon

Algoritma request memproses request dari *client* dan mengirimkan ke server. Request yang diproses adalah data hasil pertanyaan tentang kondisi tanaman cabai yang diinputkan oleh user. Algoritma request bekerja dengan cara menerima input parameter kondisi tanaman. Parameter input yang diterima kemudian dicek apakah sudah diisi oleh user. Algoritma request kemudian membuka jalur koneksi dengan server dan mengirimkan data kondisi tanaman cabai ke server menggunakan metode GET. Jalur komunikasi menggunakan teknologi AJAX.

Client mengecek keberhasilan koneksi dengan server. Jika koneksi dengan server berhasil maka data hasil inferensi dari server dapat dikirimkan ke client dan diproses melalui algoritma respon. Data hasil inferensi berupa objek JSON. Objek JSON kemudian di ubah menjadi array dan ditampilkan dalam bentuk user interface.

```
Data masukan : kondisi tanaman cabai

Deskripsi    :
1    Proses pengecekan data masukan
2    Proses membuka koneksi dengan server
3    Proses pembungkusan data masukan
4    Proses pengiriman data masukan ke server (request)

5    Proses seleksi pengecekan keberhasilan koneksi ke server

6          Jika berhasil maka proses penerimaan data respon dari
           server dilakukan

7          Proses decode data respon dari JSON menjadi array

8          Data respon dalam bentuk array ditampilkan


Keluaran     : data solusi perawatan cabai
```

Gambar 4.15 Algoritma fungsi request & respon
Sumber : Perancangan

4.2.5 Perancangan algoritma fungsi *web service*

Algoritma *web service* berfungsi untuk menerima request dari client, memprosesnya dalam mesin inferensi perawatan cabai dan mengirimkan hasilnya client. Algoritma *web service* menerima inputan berupa data kondisi tanaman cabai

dari client dan HTTP method yang digunakan oleh client. Algoritma *web service* menyeleksi fungsi yang akan digunakan berdasarkan HTTP method yang direquest oleh client. Jika HTTP method yang digunakan adalah GET dan fungsi yang dipilih adalah fungsi inferensi maka data kondisi tanaman akan diproses menuju fungsi inferensi sistem pakar perawatan cabai. Hasil dari fungsi inferensi berupa data solusi perawatan tanaman cabai diubah menjadi objek JSON dan ditampilkan.

```
Data masukan   : kondisi tanaman cabai, HTTP Method


Deskripsi      :
1              Data input kondisi tanamn cabai dipisahkan sesuai
               bagian tanaman.

2              Proses seleksi HTTP method
3              Jika HTTP method GET maka
4               Jika menu = tampilall
                 Menuju proses menampilkan data seluruh knowledge
                 perawatan cabai
5              Jika menu == inferensi
6                Menuju proses inferensi dengan parameter masukan
7                kondisi tanaman cabai
                 Hasil inferensi berupa data solusi
8                Proses decode data solusi dari array menjadi objek
                 JSON

9              Menampilkan objek JSON solusi perawatan cabai


Data Keluaran :  Tampilan Objek JSON
```

Gambar 4.16 Algoritma fungsi service
Sumber : Perancangan

4.2.6 Perancangan algoritma Fungsi Inferensi

Algoritma inferensi berfungsi sebagai mesin pencari solusi sistem pakar perawatan cabai. Inferensi menggunakan metode forward chaining [PKL-12]. Algoritma inferensi dirancang sesuai pohon aturan sistem pakar perawatan cabai pada Sub Bab **4.2.3.**

```
Data masukan   : kondisi tanamn cabai sesuai bagian tanaman
```

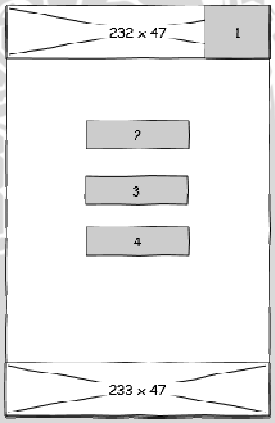
```
Deskripsi     :
1             Jika kondisitanaman == idtanamn maka
2             Query = get query kondisitanaman with idtanam

3              Jika kondisidaun == iddaun maka
4              Query += get query kondisitanaman with iddaun

5                Jika kondisidaunlain == iddaunlain maka
6                Query += get query kondisitanaman with daunlain

7                  jika kondisibuah == idbuah maka
8                     Query += get query kondisitanaman with idbuah

               Proses eksekusi query ke database

Keluaran :   Data array solusi perawatan cabai
```

Gambar 4.17 Perancangan algoritma fungsi inferensi perawatan cabai
Sumber : perancangan

4.2.7 Perancangan *User interface*

User interface atau view berada pada sisi client, yaitu berupa mobile aplication. *User interface* dirancang sesuai platform client dengan luas layar 240 x 400 pixel.

a. Desain halaman home



Gambar 4.18 Perancangan *user interface* home
Sumber : Perancangan

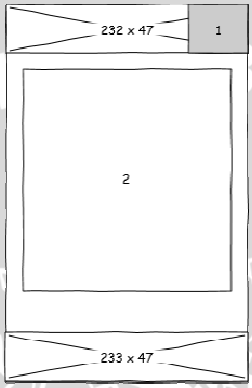
Keterangan :

1. Tombol home untuk mengarah kembali ke halaman home
2. Tombol sistem pakar untuk mengarah ke halaman sistem pakar

3. Tombol help untuk mengarah kehalaman help
4. Tombol about untuk mengarah ke halaman about

b. Desain halaman help

Halaman help berisi panduan penggunaan aplikasi. Panduan menuntun user dalam menggunakan aplikasi.



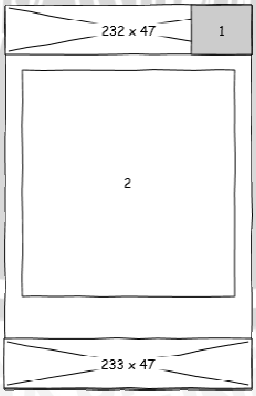
Gambar 4.19 Perancangan *user interface* help
Sumber : Perancangan

Keterangan :

1. Tombol home untuk mengarah kembali ke halaman home.
2. Penjelasan tentang penggunaan aplikasi.

c. Desain Halaman about

Halaman about berisi tentang informasi *developer* aplikasi



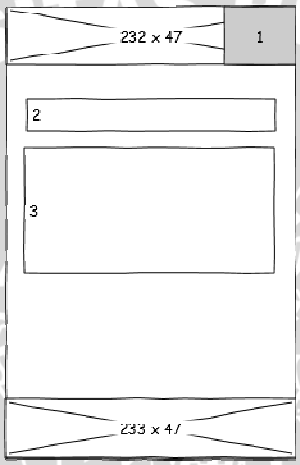
Gambar 4.20 Perancangan *user interface* about
Sumber : Perancangan

Keterangan :

1. Tombol home untuk mengarah kembali ke halaman home.
2. Penjelasan tentang developer aplikasi.

d. Desain halaman sistem pakar

Halaman sistem pakar berisi tentang pertanyaan dan input jawaban tentang kondisi tanaman cabai user.



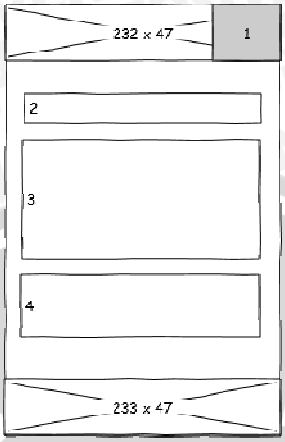
Gambar 4.21 Perancangan *user interface* sistem pakar
Sumber : Perancangan

Keterangan :

1. Tombol home untuk mengarah kembali ke halaman home.
2. Kolom pertanyaan seputar kondisi cabai.
3. Kolom jawaban.

e. Desain halaman solusi

Halaman solusi berisi tentang solusi perawatan tanaman cabai sesuai hasil dari inferensi sistem pakar



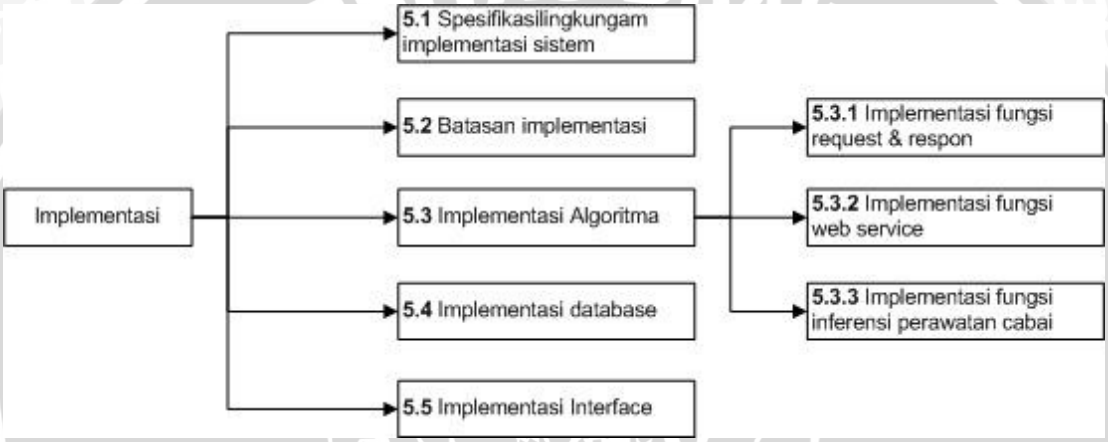
Gambar 4.22 Perancangan *user interface* solusi
Sumber : Perancangan

Keterangan :

1. Tombol home untuk mengarah kembali ke halaman home.
2. Kolom gejala perawatan
3. Kolom gambar tanaman terjangkit.
4. Solusi atas gejala

BAB V
IMPLEMENTASI

Pada bab implementasi akan dibahas implementasi perangkat lunak berdasarkan perancangan pada bab perancangan. Bab implementasi mengulas tentang spesifikasi sistem, batasan batasan dalam implementasi, implementasi algoritma, implementasi database dan implementasi *user interface*.



Gambar 5.1 Pohon Implementasi Aplikasi sistem pakar perawatan cabai berbasis RSET *web service*
Sumber : Implementasi

5.1 Spesifikasi lingkungan implementasi sistem

Aplikasi ini dibangun berdasarkan spesifikasi hardware dan software berikut

Notebook Asus N 46 V	
Prosessor	: Intel i-7 2.30ghz
RAM	: 4 Gb
Hardisk	: 500 Gb
Graphic card	: Nvdia Geforce 630M

Operating System Windows 7	
SDK client	: Nokia web tool builder 2.30 : Language : HTML, Java Script
SDK server	: Netbeans 7.1 , code igniter : Language : php
DBMS	: MySQL
Server host	: Apache

Gambar 5.2 Spesifikasi hardware software implementasi sistem
Sumber : Implementasi

5.2 Batasan implementasi

Batasan dalam implementasi aplikasi sebagai berikut :

- Aplikasi client di debug menggunakan simulator nokia 305 dengan layar 240 x 400.
- Server dihosting dengan alamat : http://sistempakarcabai.besaba.com.

5.3 Implementasi algoritma

Aplikasi RESTfull webservice sistem pakarpada penelitian ini memiliki beberapa method atau proses. Pada Sub Bab implementasi algoritma ini, akan dijelaskan beberapa algoritma dari method atau proses utama dari aplikasi. Algoritma di representasikan menggunakan pseudocode.

5.3.1 Implementasi fungsi request respon

Fungsi request diimplementasikan berdasarkan perancangan algoritma fungsi request pada bab perancangan Sub Bab 4.2.4. Fungsi request diimplementasikan menggunakan bahasa pemograman javascript.

```
FUNCTION : request PARAMETER kondisitanaman,
kondisidaun, kondisidaunlain, kondisi buah IS var
```

OUTPUT : respon catch

```
1  function init()
2  {

3      var xmlhttp;
4      var array;
5      var data;
6      xmlhttp=new XMLHttpRequest();
7      var kondisi = "0";
8      var daun = "0";
9      var buah = "0";
10     var daunLain = "0";

11     var kon = document.getElementsByName("kon");
12     for (var i = 0, length = kon.length; i < length; i++) {
13         if (kon[i].checked) {
14             kondisi = kon[i].value;
15         }
16     }

17     var da = document.getElementsByName("daun");
18     for (var i = 0, length = da.length; i < length; i++) {
19         if (da[i].checked) {
20             daun = da[i].value;
21         }
22     }

23     var dain = document.getElementsByName("daunLain");
24     for (var i = 0, length = dain.length; i < length; i++) {
25         if (dain[i].checked) {
26             daunLain = dain[i].value;
27         }
28     }

29     var bu = document.getElementsByName("buah");
30     for (var i = 0, length = bu.length; i < length; i++) {
31         if (bu[i].checked) {
32             buah = bu[i].value;
33         }
34     }

35     data = kondisi+"_"+daun+"_"+daunLain+"_"+buah+"_0";

36
xmlhttp.open("GET",http://sistempakarcabai.besaba.com/index.p
hp/service/apiinferensi/+data,true);
37     xmlhttp.setRequestHeader("Content-type",  "application/x-
```

```
www-form-urlencoded");
38    xmlhttp.send(data);


39    xmlhttp.onreadystatechange=function()
40      {
41      if (xmlhttp.readyState==4 && xmlhttp.status==200)
42        {
43          array = $.parseJSON(xmlhttp.responseText);
44        document.getElementById("hasil").innerHTML=
array[0]["nama_perawatan"];
45        document.getElementById("gambar").innerHTML  =  "<img
src=\"image/"+array[0]["gambar_unsurhara"]+"\"    style=\"max-
width:100%\" align=\"middle\"/>";
46        document.getElementById("solusi").innerHTML=
array[0]["solusi"];
                }
        };}
```

Gambar 5.3 Implementasi algoritma request dan menerima respon
Sumber : Implementasi

Penjelasan algoritma request pada gambar 5.3 sebagai berikut :

- Baris 3-10, deklarasi variable
- Baris 11-34, Proses mengambil data jawaban yang dikirim dari user interface pertanyaan kondisi cabai.
- Baris 35, Variabel data diberikan sebuah nilai string yang merupakan string penggabungan dari parameter kondisi tanaman, kondisi daun, kondisi daun lain dan kondisi buah
- Baris 36, variable xmlhttp memanggil method *openConnection* yang berfungsi untuk membuka koneksi dengan *web service* dengan mengirimkan argument yang berisi URI dari *web service* dengan http method GET.
- Baris 37, variable xmlhttp memanggil method *requestHeader* yang berfungsi untuk memberikan nilai request header.
- Baris 38 , variable xmlhttp memanggil method *send* yang berfungsi untuk mengirimkan nilai request ke *web service*. Nilairequest berisi dengan variable data.

- Baris 39-46, pengecekan apabila method request berhasil dieksekusi maka akan mendapatkan respon dari *web service*. Kemudian respon yang berupa objek JSON dikonversi menjadi array dan ditampilkan ke dokumen.

5.3.2 Implementasi fungsi service

Fungsi service diimplementasikan berdasarkan perancangan algoritma fungsi service pada bab perancangan Sub Bab 4.2.5. Fungsi request diimplementasikan menggunakan bahasa pemograman php.

```
    FUNCTION    : service API
    OUTPUT      : respond
```

```
1 <?php if(!defined('BASEPATH'))
2         exit('No direct script access allowed');

3     class Service extends CI_Controller{

4           function __construct()
            {
5                parent::__construct();
6                $this->load->helper('servis');
7                $this->load->model('database_model');
            }

8           function index()
            {
9                echo  'Halo  selamat  datang  di  rest  web
service sistempakar' ;
            }

            //fungsi API
10          function apiinferensi(){


            //check URI yang dikirim , apakah ada parameter
11           $path = $_SERVER['PATH_INFO'];
12               if($path != null){
13                    $path_params = explode("/",$path);
                      }

            //method request get
14            if($_SERVER['REQUEST_METHOD'] == 'GET'){
```

```
15              $inputan = file_get_contents("php://input");
16              $a = 0;
17              $pecah = explode("_",$path_params[3]);
18               while($a < 5){
19                   $inputarray[$a] = $pecah[$a];
20                   $a+=1;
                }

21              $query                                    =
inference($inputarray[0],$inputarray[1],$inputarray[2],$input
array[3],$inputarray[4]);
22              $data          =        $this->database_model-
>getData2($query);
23              $jason = json_encode($data);
                 Echo $json;
             }
            }

              Function getalldata{
           //Method Request untuk GET
24              if($_SERVER['REQUEST_METHOD'] == 'GET'){

25              if($path_params[3] != null){
26                   $data      =     $this->database_model-
>getData1($path_params[3]);
                    }
27              else{
28                   $data      =     $this->database_model-
>getData();}

           // menjadikan objek menjadi JSON
29            $jason = json_encode($data);
                 }

30          echo $jason;
            }
                 }

/* End of file kontrol.php */
/* Location: ./application/controllers/kontrol.php */
```

Gambar 5.4 Implementasi algoritma service
Sumber : Implementasi

- Baris 11-13, proses untuk mendapatkan *server path* yaitu http method yang digunakan oleh client ketika mengakses *web service*.

- Baris 14, Seleksi method http yang direquest dari client dengan parameter GET
- Baris 15, Proses untuk menangkap data input request dari client
- Baris 16-20, Proses untuk memecah data request menjadi array
- Baris 14, Proses memanggil fungsi inferensi dengan mengirimkan data argument yang didapat dari datarequest yang telah dipecah pecah.Hasil dari inferensi berupa query
- Baris 21, proses untuk mengeksekusi data query ke dalam fungsi database. Data yang dihasilkan merupakan solusi perawatan cabai dalam bentuk arry
- Baris 22, proses konversi solusi dari tipe array menjadi objek JSON
- Baris 23, Proses menampilakn solusi JSON

5.3.3 Implementasi fungsi inference

Fungsi inference diimplementasikan berdasarkan perancangan algoritma fungsi inference pada bab perancangan Sub Bab 4.2.6. Fungsi inference diimplementasikan menggunakan bahasa pemograman php. Inference yang digunakan menggunakan metode forward chaining yang direpresentasikan oleh pohon keputusan pada bab perancangan Sub Bab 4.2.3.

```
    FUNCTION                                    :       Inference
PARAMETER(kondisitanaman,kondisidaun,       kondisidaunlain,
kondisibuah

    OUTPUT      : query solusi perawatan cabai
```

```
    <?php  if  (  !  defined('BASEPATH'))  exit('No  direct
script access allowed');

    if ( ! function_exists('inference'))
    {

    }

    function
inference($kondisitanaman,$kondisidaun,$kondisilaindaun,$kond
isibuah,$kondisilain)
    {
        if($kondisitanaman == 1){
```

```
                //kondisi layu
            $query = "select * from perawatan_cabai where
kondisi_tanaman = 'KO001'";

                if($kondisilaindaun == 1){
                    //daun kuning
                    $query      =      $query."      "."and
kondisi_lain_daun = 'GD007'";    //fusarium
                }
                if($kondisilaindaun == 2){
                    //potongan batang nampak cairan putih
                    $query      =      $query."      "."and
kondisi_lain_daun != 'GD007'";    //bakteri
                }
        }

         if($kondisitanaman == 2){
                //kondisi rusak
            $query = "select * from perawatan_cabai where
kondisi_tanaman = 'KO002'";

                if($kondisidaun == 1 ){
                    //hijau
                    $query = $query." "."and kondisi_daun =
'GD012'";

                    if($kondisibuah == 1){
                        //bauh merah membusuk dan gugur
                        $query    =    $query."    "."and
kondisi_buah ='GB002'";

                    }

                    if($kondisibuah == 2){
                        // buah hijau membusuk dan  gugur
                        $query    =    $query."    "."and
kondisi_buah = 'GB003'"; // ulat grayak
                    }

                    if($kondisibuah == 3){
                        // ujung buah menguning dan gugur
                        $query    =    $query."    "."and
kondisi_buah = 'GB005'"; // kekurangan KALSIUM
                    }
                }

                if($kondisidaun ==2){
                    //menguning
```

```
                    $query = $query." "."and kondisi_daun
='GD007'";

                    if($kondisilaindaun == 3){
                         //kuning di bagian tertentu
                         $query    =    $query."    "."and
kondisi_lain_daun = 'GD010'";
                    }
                    if($kondisilaindaun == 4 ){
                         //kuning merata
                         $query    =    $query."    "."and
kondisi_lain_daun = 'GD014' ";
                    }
               }
               if($kondisidaun ==3){
                    //bercak daun
                    $query = $query." "."and kondisi_daun =
'GD006'";
               }

               if($kondisidaun ==4){
                    //busuk
                    $query = $query." "."and kondisi_daun =
'GD005' and kondisi_buah='GB002'";

               }

               if ($kondisidaun == 5) {
                    //keriting
                    $query = $query." "."and kondisi_daun =
'GD009'";
                    if ($kondisilaindaun == 5) {
                         //keriting menguning
                         $query      =$query."       "."and
kondisi_lain_daun = 'GD014'";
                    }
                    if ($kondisilaindaun == 6) {
                        //keriting menggulung keatas
                        $query       =$query."       "."and
kondisi_lain_daun = 'GD001'";
                    }
                    if ($kondisilaindaun == 7) {
                         //keriting menggulung kebawah
                         $query       =$query."       "."and
kondisi_lain_daun = 'GD002'";
                    }
          }
     }
```

```
        //return
        return $query;
}
?>
```

Gambar 5.5 Implementasi algoritma inferensi sistem pakar perawatan cabai
Sumber : Implementasi

Fungsi inference merupakan proses inferensi / mencari solusi sistem pakar dengan metode runut maju atau dikenal sebagai forward chaining. Data parameter fungsi berupa data tanaman diseleksi mulai dari kondisi tanaman, daun hingga buah. Hasil dari fungsi ini berupa kembalian solusi namun dalam bentuk query.

5.4 Implementasi database

Implementasi database dilakukan berdasarkan perancangan data pada bab 4 Sub Bab 4.2.2 menggunakan MySQL. Dibawah ini merupakan struktur tabel-tabel yang ada dalam database pengetahuan tentang cabai merah.

1. Tabel gejala_batang

Tabel 5.1 Tabel gejala_batang

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_gejala_batang	varchar(5)	latin1_swedish_ci		No	None		
	nama_gejala_batang	varchar(100)	latin1_swedish_ci		No	None		

Sumber : Implementasi

Tabel gejala_batang berisi tentang data kondisi yang mungkin terjadi pada batang tanaman cabai.

2. Tabel gejala_buah

Tabel 5.2 Tabel gejala_buah

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_gejala_buah	varchar(5)	latin1_swedish_ci		No	None		
	nama_gejala_buah	varchar(100)	latin1_swedish_ci		No	None		

Sumber : Implementasi

Tabel gejala_buah berisi tentang data kondisi yang mungkin terjadi pada buah cabai seperti layu atau rusak.

3. Tabel gejala_daun

Tabel 5.3 Tabel gejala_daun

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_gejala_daun	varchar(5)	latin1_swedish_ci		No	None		
	nama_gejala_daun	varchar(100)	latin1_swedish_ci		No	None		

Sumber : Implementasi

Tabel gejala_daun berisi tentang data kondisi yang mungkin terjadi pada daun tanaman cabai.

4. Tabel gejala_lain

5.4 Tabel gejala_lain

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_gejala_lain	varchar(5)	latin1_swedish_ci		No	None		
	nama_gejala_lain	varchar(100)	latin1_swedish_ci		No	None		

Sumber : Implementasi

Tabel gejala_lain berisi tentang data kondisi lain yang mungkin terjadi pada tanaman cabai

5. Tabel kondisi_tanaman

Tabel 5.5 Tabel kondisi_tanaman

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_kondisitanaman	varchar(5)	latin1_swedish_ci		No	None		
	kondisi	varchar(10)	latin1_swedish_ci		No	None		

Sumber : Implementasi

Tabel kondisi_tanaman berisi tentang data kondisi fisik tanaman yang mungkin terjadi pada tanaman cabai

6. Tabel musim

Tabel 5.6 Tabel musim

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_musim	varchar(5)	latin1_swedish_ci		No	None		
	musim	varchar(10)	latin1_swedish_ci		No	None		

Sumber : Impelementasi

Tabel musim berisi tentang macam musim di Indonesia

7. Tabel tipe_perawatan

Tabel 5.7 Tabel tipe_perawatan

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_perawatan_tanaman	varchar(5)	latin1_swedish_ci		No	None		
	jenis_perawatan	varchar(20)	latin1_swedish_ci		No	None		

Sumber : Implementasi

Tabel tipe_perawatan berisi attribute tipe perawatan cabai

8. Tabel perawatan_cabai

Tabel 5.8 Tabel perawatan_cabai

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id_perawatan	varchar(5)	latin1_swedish_ci		No	None		
	nama_perawatan	varchar(20)	latin1_swedish_ci		No	None		
	jenis_perawatan	varchar(5)	latin1_swedish_ci		No	None		
	kondisi_tanaman	varchar(5)	latin1_swedish_ci		No	None		
	musim	varchar(5)	latin1_swedish_ci		No	None		
	kondisi_daun	varchar(250)	latin1_swedish_ci		No	None		
	kondisi_lain_daun	varchar(100)	latin1_swedish_ci		No	None		
	kondisi_batang	varchar(250)	latin1_swedish_ci		No	None		
	kondisi_buah	varchar(250)	latin1_swedish_ci		No	None		
	kondisi_lain	varchar(250)	latin1_swedish_ci		No	None		
	solusi	varchar(10000)	latin1_swedish_ci		No	None		
	gambar_unsurhara	varchar(50)	latin1_swedish_ci		No	None		

Sumber : Implementasi

Tabel perawatan_cabai berisi data tentang penyakit, hama dan defisiensi unsur hara yang mencakup tentang ciri ciri tanaman terjangkit dan solusinya.

5.5 Implementasi *user interface*

Implementasi *user interface* dibangun berdasarkan pada perancangan *user interface* pada bab perancang Sub Bab 4.2.7 menggunakan bahasa pemograman html dan java script. Implementasi *user interface* terdiri atas *user interface* home, about, help, sistem pakar dan solusi.

a. Implementasi *user interface* home

Halaman home merupakan halaman yang pertama kali diakses ketika aplikasi client dijalankan. Didalam aplikasi home terdapat menu sistem pakar, about, help dan exit aplikasi.

Gambar 5.6 Implementasi *user interface* home
Sumber : Implementasi

b. Implementasi *user interface* help

Halaman help berisikan panduan tentang tata cara penggunaan aplikasi client *web service* sistem pakar perawatan cabai.

Gambar 5.7 Implementasi *user interface* help
Sumber : Implementasi

c. Implementasi *user interface* about

Halaman about berisikan tentang informasi developer aplikasi.

Gambar 5.8 Implementasi *user interface* about
Sumber : Implementasi

d. Implementasi *user interface* sistem pakar

Halaman sistem pakar berisi pertanyaan dan pilihan jawaban seputar kondisi tanaman cabai yang disediakan sistem sebagai inputan user.

Gambar 5.9 Implementasi *user interface* sistem pakar
Sumber : Implementasi

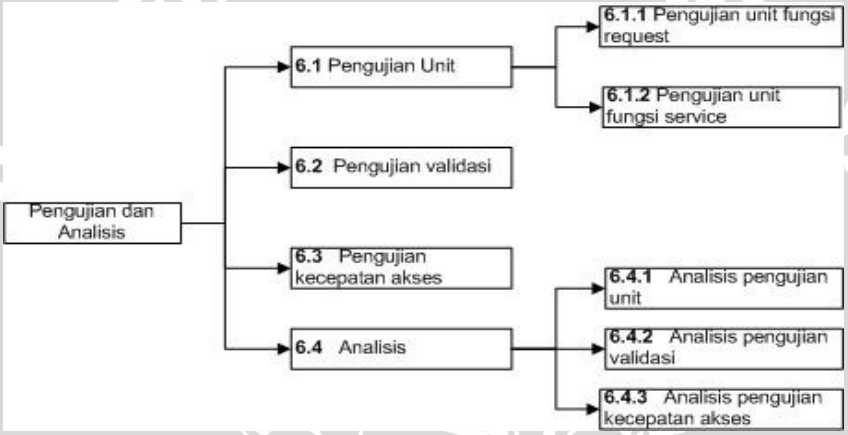
e. Implementasi *user interface* solusi

Halaman solusi berisi solusi yang diberikan sistem pakar sesuai permasalahan yang telah diinputkan user pada halaman sistem pakar.

Gambar 5.10 Implementasi *user interface* solusi
Sumber : Implementasi

BAB VI
PENGUJIAN & ANALISIS

Bab pengujian akan membahas prosedur pengujian aplikasi. Pengujian dilakukan meliputi tiga tahap,yaitu pengujian unit, pengujian validasi dan pengujian performance. Pengujian unit menggunakan pendekatan *white box*. Pengujian validasi menggunakan pendekatan *black box*. Pengujian performance akan digunakan analisis kecepatan akses *web service* sesuai tujuan penelitian.



Gambar 6.1 Diagram pengujian dan analisis
Sumber : Pengujian dan analisis

6.1 Pengujian Unit

Pengujian unit pada penelitian ini menggunakan pendekatan *white box* dengan metode yang akan digunakan adalah teknik BEST PATH TESTING (BPT). Teknik BPT menggunakan diagram flow sebagai permodelan algoritma.dari diagram flow algoritma tersebut dihitung *cyclometic complexcity* dan diberikan studi kasus untuk menguji kemampuan eksekusi tiap tiap path. Perhitungannya menggunakan rumus V(G) = E – N + 2. E adalah edge dan N adalah Node.

6.1.1 Pengujian Unit Algoritma *Request*

Tabel berikut merupakan tabel pengujian unit algoritma *request*.

Pseudocode

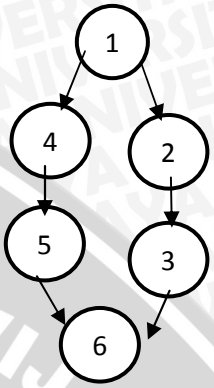
```
FUNCTION      : request PARAMETER kondisitanaman,
                kondisidaun, kondisidaunlain, kondisi
                buah IS var
OUTPUT        : request

DECLARATION :
              xmlhttp IS var
              kondisi tanaman IS var
              Kondisi daun IS var
              Kondisi lain daun IS var
              Kondisi buah IS var
              arraydata IS array
              data request IS var

DESCRIPTION :
1             data <- kondisi tanaman + kondisi daun +
              kondisi daun lain + kondisi buah
2
3             CALL xmlhttp.openConnection(arguments:url)
4
              CALL
5             xmlhttp.requestHeader(arguments:header)
6
7             CALL xmlhttp.send(arguments:data)
8
9
10
              IF    xmlhttp.readystate    ==    4    AND
11            xmlhttp.status== 200 THEN
12
13              array   <-   CALL   parseJSON(arguments:
                xmlhttp.responseText );

                PRINT array[0]

                PRINT array[1]
                PRINT array[2]
```

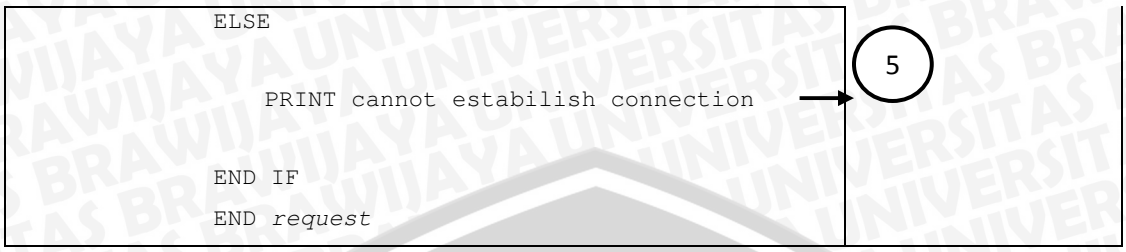
1
2
3
4
6

Flow Graph

Jumlah node : 6

Jumlah Edge : 6





Gambar 6.2 Flow graphic fungsi *request*
Sumber : Pengujian dan Analisis

Flow tabel diatas menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan

V(G) = 2 – 2 + 2 = 2

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

- Jalur 1: 1-2-3-6
- Jalur 2: 1-4-5-6

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan tabel berikut.

Tabel 6.1 *test case* fungsi *request*

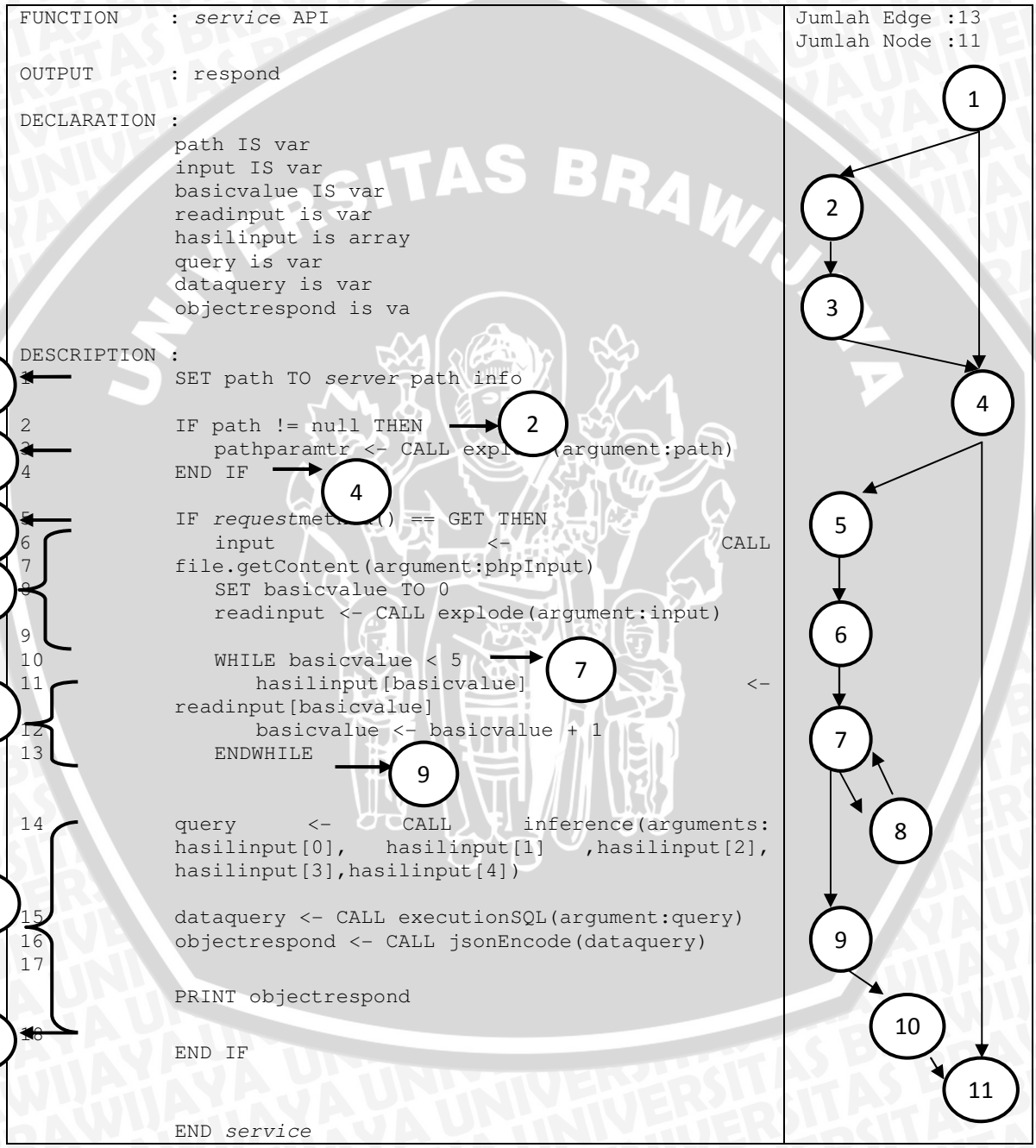
Jalur	Kasus Uji	Hasil yang diharap	Hasil yang didapat
1	*Request* success dengan code HTTP 200	*Request* diterima,dipahami dan dimengerti oleh *server* dan *server* mengirimkan respon	*Request* diterima,dipahami dan dimengerti oleh *server* dan *server* mengirimkan respon
2	*Request* gagal karena *server* mati	*Request* tidak dieksekusi oleh *server*	*Request* tidak dieksekusi oleh *server*

Sumber : Pengujian dan analisis

Kesimpulannya adalah terdapat dua statement / jalur yang mungkin dilewati didalam fungsi *request*. Statement fungsi *request* telah teruji dan berjalan dengan baik antara hasil yang diharapkan dengan yang didapat.

6.1.2 Pengujian Unit Algoritma *Service*

Tabel dibawah merupakan tabel pengujian unit algoritma *service*. Berikut aliran pengujian dari algoritma tersebut.



Gambar 6.3 Flow graphic fungsi *service*
Sumber : Pengujian dan Analisis

Flow tabel diatas menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan

V(G) = 13 – 11 + 2 = 4

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

- Jalur 1: 1-2-3-4-5-6-7-8-….
- Jalur 2: 1-5-6-7-9-10-11
- Jalur 3: 1-2-3-4-11
- Jalur 4: 1-4-11

Tabel 6.2 *Test case* fungsi *service*

Jalur	Kasus Uji	Hasil yang diharap	Hasil yang didapat
1	Basic value lebih dari 5	Infinite loop	Infinite loop
2	*Request* dengan method GET	Respon berupa solusi perawatan cabai dalam bentuk objek JSON	Respon berupa solusi perawatan cabai dalam bentuk objek JSON
3	Data dikirim dengan argument dengan method GET	Pengembalian data knowledge sesuai argumen	Pengembalian data knowledge sesuai argumen
4	Data dikirim tanpa argument dengan method GET	Pengembalian seluruh data knowledge sesuai argumen	Pengembalian seluruh data knowledge sesuai argumen

Sumber : pengujian dan analisis

Kesimpulannya adalah terdapat empat statement / jalur yang mungkin dilewati didalam fungsi *service*. Statement fungsi *service* telah teruji dan berjalan dengan baik dilihat dari hasil yang diharapkan dengan yang didapat.

6.2 Pengujian Validasi

Pengujian validasi digunakan untuk menguji fungsional kerja dari sistem. Pengujian validasi menggunakan pendekatan blackbox karena lebih mengutamakan hasil kerja sistem. Daftar kebutuhan fungional yang telah disusun pada bab perancangan menjadi target pengujian.

Tabel 6.3 Pengujian validasi

Kasus Uji	Kode	Prosedur Input	Hasil Yang Diharapkan	Hasil
Masuk kehalaman home	**F01**	Program *client* dijalankan	Muncul tampilan halaman home menu	valid
Memilih menu help	**F02**	*User* Mengklik menu help pada home menu	Muncul tampilan halaman help	valid
Memlilih menu about	**F03**	*User* Mengklik menu about pada home menu	Muncul tampilan halaman about	valid
Memilih menu sistem pakar	**F04**	*User* Mengklik menu sistem pakar pada home menu	Muncul tampilan halaman pertanyaan kondisi tanaman cabai beserta pilihan jawaban	valid
Menjawab pertanyaan	**F05**	*User* memilih jawaban dengan mengklik radio button	Muncul tombol solusi	valid
	F06	*User* memilih jawaban dengan mengklik label jawaban	Tombol solusi tidak muncul	tidak

Menampilkan solusi	**F07**	*User* menekan tombol solusi	Muncul tampilan halaman hasil solusi sistem pakar	valid

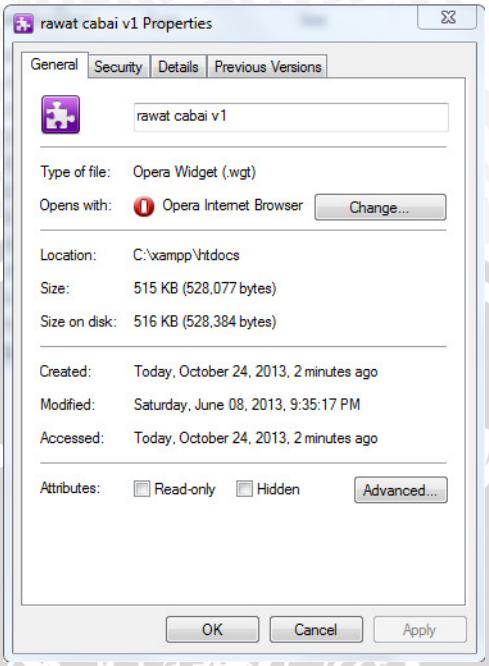
Sumber : pengujian dan analisis

Kesimpulan yang dapat diambil dari pengujian adalah sebagian besar fungsi perangkat lunak dapat berjalan sesuai dengan kebutuhan. Namun pada penggujian F06 hasil yang didapat tidak valid karena kemungkinan kesalahan *user* dalam memilih zona radio button. Radio button harus diklik / dipilih tepat dibulatan.

6.3 Pengujian Kecepatan Akses

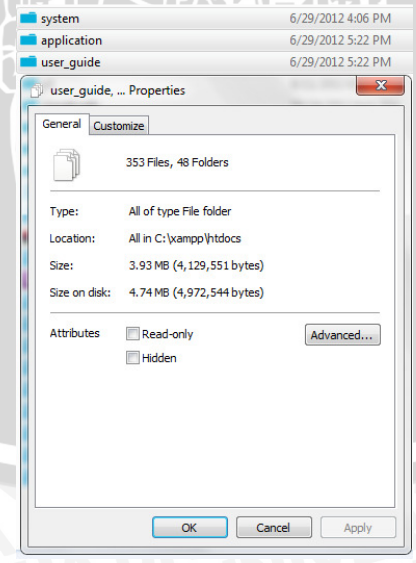
Pengujian kecepatan akses dilakukan untuk mengetahui tujuan penelitian yaitu mengoptimalisasi kecepatan akses sistem pakar. Kecepatan yang diuji adalah perbandingan kecepatan akses antara sistem pakar *web* standard dan sistem pakar berbasis RESTfull *web service*. Batasan batasan yang dilaukan pada pengujian kecepatan yaitu :

- Pengujian kecepatan meliputi uji load time, page size dan total *request*.
- Kecepatan browsing internet distandarisasikan sesuai kecepatan maksimal jaringan GPRS yaitu 115 kbps
- Aplikasi yang digunakan sebagai pengujian adalah *"Pingdom website speed test"*.
- *Test case* yang diberikan adalah mencari solusi perawatan cabai dengan gejala terserang lalat buah
- Ukuran file aplikasi client sistem pakar perawatan cabai adalah 516 KB.



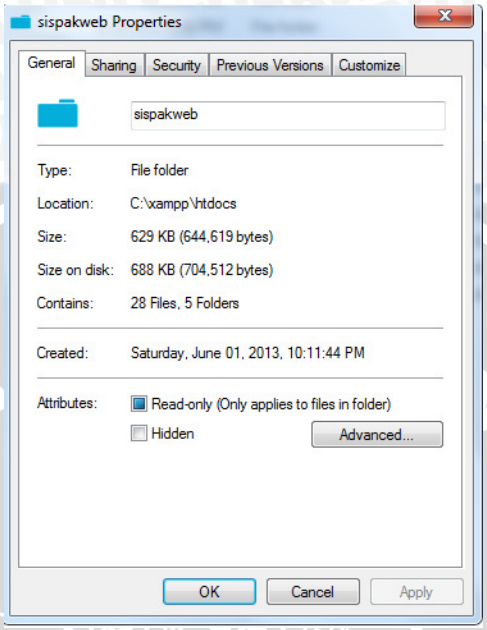
Gambar 6.4 Ukuran file aplikasi client sistem pakar perawatan cabai
Sumber : implementasi

- Ukuran file *web service* sistem pakar perawatan cabai adalah 3.93 Mb. Web service dibuat menggunakan framework CI yang memeiliki tiga folder utama, yaitu system, application dan user guide



Gambar 6.5 Ukuran file aplikasi *web service* sistem pakar perawatan cabai
Sumber : implementasi

- Ukuran file *web standard* sistem pakar perawatan cabai adalah 629 KB.



Gambar 6.6 Ukuran file web standar service sistem pakar perawatan cabai
Sumber : implementasi

Skenario pengujian

Skenario pengujian kecepatan akses *web service* perawatan cabai disusun berdasarkan langkah langkah sebagai berikut :

1. Setting kecepatan internet pada computer pengujian. Kecepatan disetting sebesar 115 kbps dengan menggunakan aplikasi "internet-sehat" (lampiran 1).
2. Membuka aplikasi pingdom *web*site speed *test*.
3. Menguji kecepatan *web service* perawatan cabai dengan cara mengirimkan *request* ke *server web service* melalui aplikasi pingdom *web*site speed *test* (lampiran 2).
4. Menguji keceptan *web* perawatan cabai standar dengan mengirimkan *request* ke*server web* melalui aplikasi pingdom *web*site speed *test*. (lampiran 2).
5. Membandingkan hasil pengujian kecepatan yang didapat.
6. Pengambilan kesimpulan.

Hasil pengujian kecepatan akses dapat dilihat pada Tabel 6.4 :

Tabel 6.4 Tabel hasil pengujian kecepatan akses

No Aturan	Web standar (aplikasi terdahulu)			REST web service		
	Load time (millisecond)	Page size (Byte)	Total request	Load time (millisecond)	Page size (Byte)	Total request
R01	1050	9548.8	7	351	473	1
R02	950	9100.8	7	395	800	1
R03	945	12377.6	7	326	955	1
R04	168	8089.6	7	324	815	1
R05	787	8102.4	7	307	876	1
R06	209	11033.6	7	315	855	1
R07	744	9625.6	7	320	610	1
R08	314	9625.6	7	100	758	1
R09	934	8947.2	7	98	1010	1
R10	804	9740.8	7	104	946	1
R11	836	9600	7	321	764	1
R12	913	8806.4	7	106	931	1
Total	**8654**	**115598.4**		**3067**	**9793**	
Rata rata	**8654 / 12 = 721.1 ms**			**3067 / 12 = 255.58 ms**		

Sumber : Pengujian dan Analsis

1. Total *request web service* hanya satu yaitu objek JSON dibandingkan dengan *web* standar yang mengirimkan tujuh *request*
2. Dari 7 *request web* standar berupa *user interface* dan data solusi perawatan cabai.

Hasil dari pengujian kecepatan akses sesuai dengan Tabel 6.4 dapat dirangkum menjadi tabel berikut :

Tabel 6.5 Rangkuman speed test sistem pakar perawatan cabai

	Total load time	Total page size
Sistem pakar perawatan cbai web standar	8654 ms	115598,4 B
Sistem pakar perawatan cabai berbasis *web service*	3067 ms	9793 B

Sumber : pengujian

6.4 Analisis

Analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian *Aplikasi RESTfull web service yang* telah dilakukan. Analisis didapat dari hasil pengujian unit, validasi dan pengujian kecepatan akses.

6.4.1 Analisis Pengujian Unit

Hasil analisa yang didapat dari pengujian unit sebagai yaitu :

1. Seluruh kemungkian statement/ jalur fungsi telah dihitung menggunakan rumus *cyclomatic complexity*.
2. Jalur yang didapat pada setiap fungsi diuji menggunakan *test case* dan menghasilkan output yang sesuai dengan hasil yang diharapkan.

6.4.2 Analisis Pengujian Validasi

Berdasarkan hasil pengujian validasi, hampir semua fungsi berjalan sesuai dengan kebutuhan yang diharapkan. Dengan demikian dapat disimpulkan bahwa implementasi sistem sesuai dengan kebutuhan awal sistem.

6.4.3 Analisis Pengujian Kecepatan Akses

Pengujian kecepatan akses membuktikan bahwa kecepatan akses dengan *web service* memiliki kecepatan yang lebih di banding dengan *web* standar. Ukuran file instalasi web service adalah 3,93 MB lebih besar daripada total size web service

standar sebesar 629 KB namun tidak mempengaruhi hasil kecepatan pengujian. Ini dikarenakan data yang dikirm dari *webservice* berupa object JSON lebih ringan dibandingkan dengan *web* standar yang harus mengirim seluruh objek termasuk GUI.

Total waktu request yang dibutuhkan dalam mengakses sistem pakar web standar adalah 8654 ms. Total waktu request yang dibutuhkan untuk mengakses sistem pakar *web service* adalah 3067 ms.

Selisih waktu request = 3067 / 8654 * 100 = 35.44 %

Dapat disimpulkan bahwa tujuan penelitian yaitu optimalisasi kecepatan akses sistem pakar telah terpenuhi.

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Dari hasil pengamatan selama perancangan, implementasi, dan proses pengujian perangkat lunak RESTful *web service* sistem pakar perawatan cabai yang dilakukan, diambil kesimpulan sebagai berikut :

1. Perangkat lunak yang dibangun adalah *web service* sistem pakar perawatan cabai dan client yang menggunakan service dari web service berupa aplikasi mobile web app.
2. *Web service* dibangun dengan menggunakan arsitektur REST sehingga penyebutannya menjadi RESTfull *web service*.
3. Perangkat lunak RESTfull *web service* sistem pakar perawatan cabai untuk mempercepat kecepatan akses dibanding penelitian sebelumnya yang menggunakan web standar.
4. Dari hasil pengujian kecepatan akses terbukti bahwa penggunaan *web service* mempercepat kecepatan akses aplikasi sistem pakar dibandingkan dengan web standar dengan selisih waktu akses hingga 35.44 %.

7.2. Saran

Saran untuk pengembangan aplikasi RESTful *web service* sistem pakar perawatan cabai lebih lanjut antara lain :

1. Untuk aplikasi yang tidak memerlukan kecepatan update data sebaiknya dibuat menjadi aplikasi mobile *native* dengan data dan proses sistem pakar dilakukan pada sisi aplikasi client.
2. Web service diaplikasikan ketika sebuah sistem membutuhkan pertukaran data ,update data dan pengguna yang banyak dalam waktu yang cepat.

3. Dapat dilakukan pengembangan aplikasi sehingga nanti knowledge sistem pakar dapat ditambahkan melalui mobile client.
4. Dapat dioptimalkan kinerja RESTful *web service* dengan digunakannya seluruh HTTP method yang tersedia,yaitu : GET, POST, DELETE, UPDATE

DAFTAR PUSTAKA

[AKM-12] Belson, David. 2013. "The state of the Internet".Penerbit : akamai.

[TMP-12] Widiastuti, Rina. 2013 "Akses internet diIndonesia paling lambat".penerbit : www.tempo.co

[SIL-13] www.kabaronline.com . 2013."Situs Indonesia paling lamban sedunia".

[WSC-13] www.w3schools.com.2013. "*web service*".

[RST-13] www.belajarandroid.com. 2013. "Restfull web API"

[PKL-12] Luthfillah, Iqbal. 2013."Sistem pakar perawatan cabai merah keriting berbasis *web service*". Universitas brawijaya malang.

[HAH-09] Hamad, Hatem. 2009. Performance Evaluation of Restfull webserivce for mobile device". penerbit : Islamic University of Gaza. Palestina

[YAU-12] Utama, Yadi. "Teknik Pemrograman *Web service* PHP Dengan Menggunakan SOAP dan WSDL"

[SUA-10] Allamaraju, Subbu. 2010. "Restfull *web service* cookbook".Penerbit : O'rielly

[ADR-13] wisnurdi. 2013. "Rest vs Soap".

[RER-12] Ramadhan, reymar. 2012. "Security di *web service* Soap dan Rest". Penerbit : Universitas Maranatha. Bandung.

[SOM-03] Sommerville, Ian. 2003. Software Engineering (Rekayasa Perangkat Lunak) / Edisi 6 / Jilid 1, Jakarta: Penerbit Erlangga.

[YGP-13] Yulia, group. 2013. "System Development aLife Cycle". www.blogspot.com/yuliagroup

[RPL-08] R. Mulyanto, Aunur. 2008. "Rekayasa Perangkat Lunak". Direktorat Pembinaan Sekolah menengah kejuruan.

[KKN-07] klikkanan.com. 2007. “Tutorial Belajar Dasar-dasar Hypertext Markup Language” Penerbit : DuniaPustaka.com

[KUS-07] Kusumadewi, Sri, 2007, Artificial Intelligence: Teknik dan Aplikasinya, Yogyakarta, Graha Ilmu.

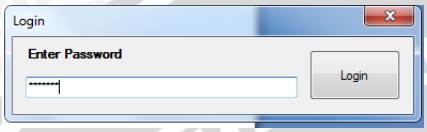
[HAH-12] 2014.”Sekilas tentang REST”. http://ismanprahadi.wordpress.com/

LAMPIRAN 1

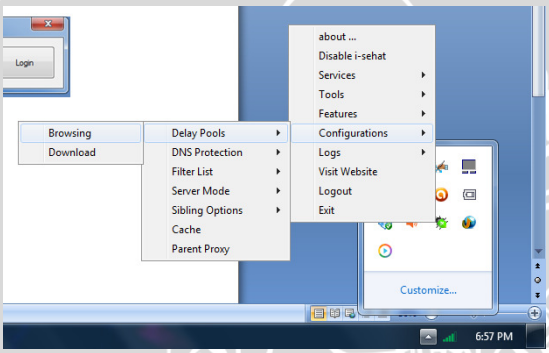
SETTING KECEPATAN AKSES INTERNET

1. Jalankan aplikasi “i-sehat”.

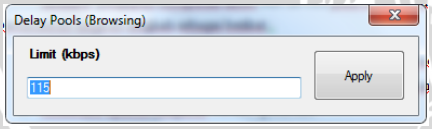
2. Login aplikasi “i-sehat” untuk mendapatkan akses.



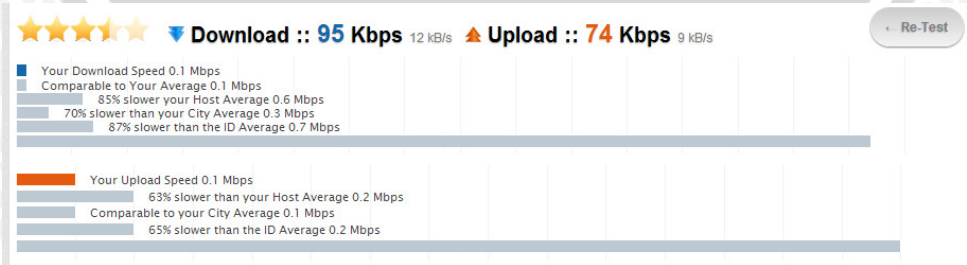
3. Klik kanan pada icon isehat, pilih menu configuration - delay pools - browsing.



4. Isi batasan kecepatan browsing dengan nilai “115”.



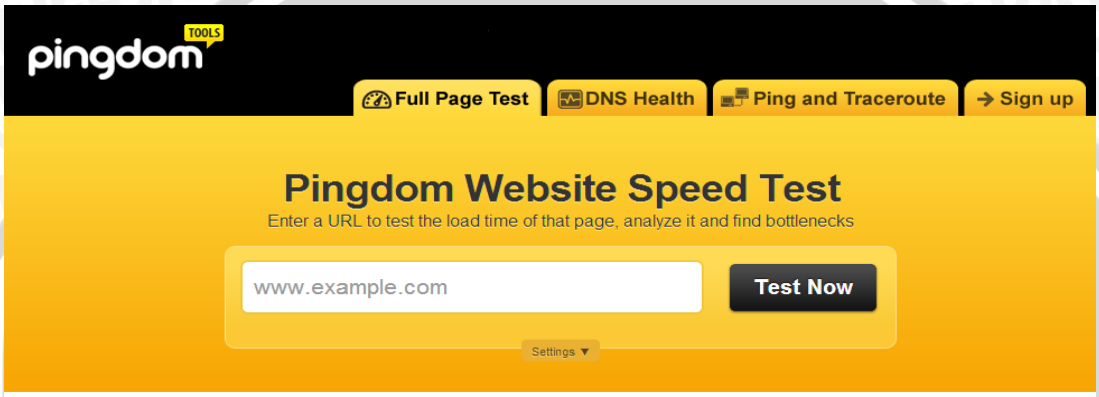
5. Melakukan testing kecepatan dengan mengakses web “http://testmy.net/ “.



LAMPIRAN 2

PROSEDUR PENGUJIAN WEBSITE DAN *WEB SERVICE* SISTEM PAKAR PERAWATAN TANAMAN CABAI

1. Buka aplikasi "pingdom test tool"



2. Menguji web sistem pakar perawatan cabai standar dengan cara memasukkan url dengan parameter cabai terkena layu fusarium "http://sistempakarcabai.besaba.com/index.php/service/api/11" . Kemudian klik tombol test now dan hasil test kecepatan akan muncul.

3. Menguji *web service* sistem pakar perawatan cabai standar dengan cara memasukkan url dengan parameter cabai terkena layu fusarium "http://sistempakarweb.besaba.com/inference.php?kon=1&daunLain=1&x=33&y=11". Kemudian klik tombol test now dan hasil test kecepatan akan muncul.