STUDI PERBANDINGAN KINERJA ANTARA SOAP WEB SERVICE DENGAN REST WEB SERVICE DALAM PERTUKARAN DATA PADA PLATFORM ANDROID DAN WINDOWS PHONE

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:

RIA RIZKI WARDANI

NIM: 115060807111061

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG

2015

PENGESAHAN

STUDI PERBANDINGAN KINERJA SOAP WEB SERVICE DENGAN RESTFUL WEB SERVICE DALAM PERTUKARAN DATA PADA PLATFORM ANDROID DAN WINDOWS PHONE

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Ria Rizki Wardani
NIM. 115060807111061

Skripsi ini telah diuji dan dinyatakan lulus pada
12 November 2015
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Arief Andy Soebroto S.T., M.Kom.	Aryo Pinandito S.T., M.MT.
NIP. 19720425 199903 1 002	NIP. 19830519 201404 1 001

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS

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 didalam 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, 12 November 2015

Ria Rizki Wardani

NIM. 115060800111015

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayah-Nya, penulis dapat menyelesaikan tugas akhir yang berjudul "STUDI PERBANDINGAN KINERJA SOAP WEB SERVICE DENGAN RESTFUL WEB SERVICE DALAM PERTUKARAN DATA PADA PLATFORM ANDROID DAN WINDOWS PHONE".

Dalam pelaksanaan dan penulisan tugas akhir ini penulis mendapatkan banyak bantuan dari berbagai pihak baik secara moril maupun materiil. Dalam kesempatan ini penulis ingin mengucapkan terimakasih yang sebesar–besarnya kepada:

1. Bunda Suwarni, Bapak Slamet Riadi, Adik Nabila Dwi Wahyuni, Adik Dhaniella Ayu Maulidhina dan seluruh keluarga besar, atas segala nasehat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta yang senantiasa tiada henti-hentinya memberikan doa dan semangat demi terselesaikannya skripsi ini.
2. Bapak Arief Andy Soebroto, S.T., M.Kom dan Bapak Aryo Pinandito S.T., M.MT selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
3. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Drs. Marji, M.T dan Bapak Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Informatika Universitas Brawijaya.
5. Seluruh Dosen Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
6. Seluruh Civitas Akademika Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
7. *The one who always listens and cares me*, Dwi Hardyanto, S.Kom *for always be right by my side through my darkest hour and brightest day. No matter how hard my life, he always be by my side*.
8. Anis Fitri Nur Masruriyah, S.Kom., Voni Nurfaizzah, S.Kom., Deby Putri I, S.Kom., Tety Dewi P, S.Kom., Eka Hastian A, S.Kom., Aryani, S.Kom., dan seluruh anggota bimbingan Bapak Arief Andy Soebroto yang selalu bertukar semangat dan pendapat dengan penulis selama menyelesaikan skripsi ini.
9. Himyatul Milah, S.Kom., Arini Indah P., S.Kom., Meidina Rosyadah, S.Kom., Renzy Nuritha S, S.Kom., Luki Puspita Sari, S.Kom., Nadia Previani, S.Kom., Maulida Dwi A, S.Kom., Lutfi Ajeng Karlina, S.Kom., Ihda Mawaddah, S.Kom., Weni Prameswari, S.Kom, dan seluruh teman-teman kelas TIF-G 2011 yang selalu dapat memberi motivasi dan bimbingan serta berbagi pengalaman tentang kehidupan.

10. Farida Ayu Wirawati Kusuma S.Pi, Dr. Nadia Ochang, Dr. Inas Khairunnisa, Dianne Dear S.H., M.Kn., dan seluruh anggota D5 house yang telah menjadi keluarga senantiasa setia dalam mendukung dan memberikan semangat.
11. Teman–teman angkatan 2011 Informatika, terima kasih atas segala bantuannya selama menempuh studi di Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
12. Seluruh pihak yang telah membantu kelancaran penulisan tugas akhir yang tidak dapat kami sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan tugas akhir ini masih banyak kekurangan baik format penulisan maupun isinya. Oleh karena itu, saran dan kritik membangun dari para pembaca senantiasa penulis harapkan guna perbaikan bagi tugas akhir selanjutnya. Semoga tugas akhir ini dapat memberikan manfaat bagi semua pihak, Aamiin.

Malang, 12 November 2015

Penulis

riarizkiwardani@gmail.com

ABSTRAK

Stasiun Klimatologi BMKG Karangploso dalam mewujudkan visi dan misi telah menyediakan *website real-time* sebagai media penyampaian informasi. Seiring dengan perkembangan teknologi penyampaian informasi aplikasi *web* monitoring klimatologi BMKG Karangploso juga tersedia pada aplikasi *mobile*. Android dan Windows Phone merupakan aplikasi *mobile* monitoring klimatologi BMKG Karangploso yang telah dibangun. Optimalisasi kinerja kecepatan akses antara aplikasi Android monitoring klimatologi BMKG Karangploso dan aplikasi Windows Phone monitoring klimatologi BMKG Karangploso belum diketahui. Metode SOAP *web service* dan REST *web service* digunakan sebagai pembanding dalam menentukan optimalisasi kinerja kecepatan akses pengambilan data pada aplikasi Android dan Windows Phone tersebut. Pendekatan pengembangan aplikasi monitoring dengan menggunakan *native mobile application development*. Pengujian terdiri dari komparasi dalam lingkungan dan komparasi antar lingkungan. Hasil pengujian menunjukkan bahwa REST *web service* dalam lingkungan Android mempunyai rata-rata kecepatan akses pengambilan data sebesar 0.4191 *second* dan 3.55x lebih cepat dibandingkan dengan REST *web service* dengan rata-rata kecepatan akses pengambilan data sebesar 1.49035 *second* dalam lingkungan Windows Phone.

Kata Kunci: *Service*, *Native*, *Smartphone*, Komparasi.

ABSTRACT

Climatological Station BMKG Karangploso in realizing the vision and mission has been provide real-time website as a information delivery media. Along with the development delivery of information technology web application monitoring climatology BMKG Karangploso also available on mobile applications. Android and Windows Phone is a mobile application monitoring climatology Karangploso BMKG has been built. The optimization of access speeds performance between android application of monitoring climatological BMKG Karangploso and Windows Phone application of monitoring climatology BMKG Karangploso unknown. Methods of SOAP web service and REST web service are used as a comparison determining performance optimization of access speed a data retrieval on Android and Windows Phone. Approach to monitoring application development using native mobile application development. Testing consisted of comparison in environment and comparison between environment. The results show that REST web service in an environment of Android has an average access speed of data retrieval at 0.4191 second, and 3.55x faster than REST web service with an average access speed of data retrieval 1.49035 second in an environment of Windows Phone.

Keywords: Services, Native, Smartphone, Comparison.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1. Latar Belakang

Teknologi informasi dan komunikasi dari waktu ke waktu terus mengalami perkembangan yang pesat. Aplikasi yang dikembangkan untuk memenuhi kebutuhan manusia akan informasi diantaranya adalah aplikasi perangkat lunak *web* dan aplikasi perangkat *mobile*. Salah satu yang memanfaatkan teknologi tersebut adalah Stasiun Badan Meteorologi Klimatologi dan Geofisika (BMKG) kelas II Karangploso Malang. Stasiun BMKG menangani perubahan cuaca dan iklim secara cepat, tepat, akurat, dan mudah dipahami. Salah satu visi dan misi dari stasiun klimatologi BMKG Karangploso yaitu menyediakan data, informasi dan jasa meteorologi, klimatologi, kualitas udara dan geofisika yang handal dan terpercaya (Geofisika, 2013). Stasiun klimatologi BMKG Karangploso dalam mewujudkan visi dan misi telah menyediakan *website real-time* sebagai media penyampaian informasi. Kinerja stasiun klimatologi BMKG Karangploso dalam kualitas recording dan monitoring telah didukung dengan aplikasi *mobile* yang lebih efektif, efisien, dan memudahkan dalam pekerjaan petugas.

Perkembangan teknologi perangkat *web* dan perangkat *mobile* membuat layanan *web* dapat disediakan secara *machine-to-machine* yang memungkinkan perangkat lunak lain untuk menggunakan layanan atau fitur yang terdapat pada aplikasi *web* yang sudah ada. Teknologi tersebut dikenal sebagai *web services* (Ghifary, 2011). *Developer* dengan adanya *web services* tidak perlu membuat ulang suatu modul yang pernah dibuat sebelumnya. Misal pada suatu perangkat lunak akan dibuat modul *search engine* seperti Google. *Developer* cukup dengan membuat program untuk mengirimkan *request* dan menerima *response* dari Google. Aplikasi-aplikasi *web* yang cukup terkenal seperti Google, Facebook, Twitter, Instagram, dan lain lain, sudah menyediakan layanan yang berupa Aplication Programming Interface (API) agar dapat digunakan oleh aplikasi lainnya. Perangkat lunak seperti Google, Facebook, Twitter, maupun Instagram yang berjalan pada *platform* berbeda-beda khususnya pada *platform mobile* diantaranya Blackberry OS, Android, Windows Phone, iOS, dan sebagainya merupakan bukti adanya keberadaan API tersebut (Ghifary, 2011).

Teknologi dalam sebuah layanan *web service* pasti erat hubungannya dengan permasalahan kualitas layanan atau Quality of Service (QoS). Matrik kualitas layanan *web service* terbagi dalam beberapa hal diantaranya yaitu *accessibility*, *reliability*, *performance* (Ladan, 2011). Pengukuran kinerja layanan *web service* dapat dilakukan berdasarkan *latency* yang merupakan lama waktu yang diperlukan dari pengiriman permintaan oleh klien sampai menerima hasil permintaan tersebut dari *server*. Terdapat dua prinsip utama dalam pengambilan data *web services* diantaranya Simple Object Access Protocol (SOAP) dan Representational State Transfer (REST) (Padiya, 2013). SOAP merupakan hasil dari respon data yang tersusun atas en*velope* yang terdapat *header* dan *body*

didalamnya (Ebert, 2006). SOAP *web service* memberikan respon data dengan format Extensible Markup Language (XML) (Lee, 2014). Prinsip REST merupakan suatu bentuk hubungan *client server* yang mana ketika *client* mengirimkan permintaan pada *server* kemudian permintaan tersebut oleh *server* diproses dan diberikan tanggapan terhadap permintaan yang telah dikirimkan oleh *client*. REST tidak memerlukan format pesan seperti *envelope* yang di dalamnya terdapat *header* dan *body* seperti yang dibutuhkan oleh prinsip SOAP. Aplikasi *web* yang menerapkan prinsip REST dinamakan RESTful *web service* (Padiya, 2013). RESTful *web service* mendukung data dengan format XML dan JSON (Reddy, 2013).

Penyampaian informasi pada aplikasi web Klimatologi BMKG Karangploso saat ini sudah dikembangkan dengan layanan API sehingga dapat berjalan pada *platform* yang berbeda yaitu pada Android dan Windows Phone. Pengembangan dalam pembuatan aplikasi baik Android maupun Windows Phone dapat menggunakan beberapa pendekatan diantaranya *Native Mobile Application Development*, *Hybrid Mobile Application Development*, dan *Web Mobile Application Development* (Tun, 2014). Android merupakan tumpukan perangkat lunak open source yang mencakup sistem operasi, middleware, dan aplikasi utama bersama dengan satu set API untuk menulis aplikasi mobile (Meier, 2009). Aplikasi Android dikembangkan dengan menggunakan bahasa pemrograman Java (Sharma, 2014). Windows Phone atau Microsoft Windows Phone merupakan sistem operasi untuk aplikasi mobile yang dikembangkan oleh Microsoft (Cholli, 2012). Aplikasi Windows Phone dikembangkan dengan menggunakan Bahasa pemrograman .NET (Gronli, 2014).

Penelitian terdahulu yang dilakukan oleh (Luthfillah, 2014) menggunakan pendekatan pengembangan *hybrid mobile application* mendapatkan hasil bahwa REST web service dapat memisahkan tampilan dengan logic sehingga beban web lebih ringan. Hasil penelitian tersebut menyimpulkan bahwa penggunaan RESTful *web service* mereduksi waktu *loading* akses sistem pakar perawatan cabai hingga 35%. Permasalahan pada penelitian ini adalah untuk membandingkan kinerja kecepatan akses pengambilan data SOAP dan REST *web service* yang terdapat pada Android dan Windows Phone dengan menggunakan pendekatan pengembangan *Native Mobile Application Development*. Tujuan dan manfaat yang diharapkan dalam penelitian ini adalah memberi kemudahan bagi *developer*/pengembang aplikasi dalam menentukan *web service* dan pendekatan pengembangan aplikasi yang sesuai dengan kebutuhan serta menjadi literatur/acuan dalam mengkaji prinsip *web services* dalam pengembangan aplikasi *native mobile* khususnya pada platform Android dan Windows Phone. Kinerja penggunaan SOAP *web service* dan REST *web service* pada masing–masing *platform* juga akan dievaluasi. Pengujian perlu dilakukan untuk mendapatkan performa yang lebih baik melalui analisis waktu yang dibutuhkan untuk melakukan *request* layanan *web services*.

1.2. Rumusan Masalah

Penelitian ini merumuskan beberapa permasalahan yang akan dibahas meliputi:

1. Bagaimana perbandingan kinerja dengan menggunakan pendekatan *native mobile application development* antara SOAP *web service* dengan REST *web service* pada Android?
2. Bagaimana perbandingan kinerja dengan menggunakan pendekatan *native mobile application development* antara SOAP *web service* dengan REST *web service* pada Windows Phone?
3. Apakah kinerja SOAP dan REST *web service* pada aplikasi Android dan Windows Phone dipengaruhi oleh banyaknya *line of code*?
4. Bagaimana hasil perbandingan kinerja dari SOAP *web service* dan REST *web service* pada Android dan Windows Phone?

1.3. Batasan Masalah

Penelitian yang dilakukan memiliki batasan–batasan masalah sebagai berikut:

1. Studi kasus yang digunakan adalah aplikasi *mobile* monitoring klimatologi Karangploso Malang dengan menggunakan server *Localhost*. Server *localhost* dipilih sebagai pereduksi untuk akses homogenitas di jaringan.
2. *Platform* aplikasi yang digunakan dalam penelitian ini dibatasi pada Android dan Windows Phone.
3. Pendekatan pengembangan aplikasi menggunakan *native mobile application development*.
4. Pengujian kinerja difokuskan kepada pengambilan data pada aplikasi monitoring klimatologi Karangploso Malang.

1.4. Tujuan

Penelitian yang dilakukan ini memiliki tujuan untuk menganalisis kinerja kecepatan akses pengambilan data pada SOAP dan REST *web services* dari perangkat *mobile* khususnya Android dan Windows Phone menggunakan pendekatan pengembangan *native* dengan studi kasus aplikasi *mobile* Stasiun Klimatologi kelas II Karangploso Malang.

1.5. Manfaat

Adapun manfaat yang diharapkan dalam dokumentasi penelitian ini adalah sebagai berikut:

1. Memberikan kemudahan bagi *developer*/pengembang aplikasi dalam menentukan *web service* yang sesuai dengan kebutuhan dalam pengembangan aplikasi *mobile* khususnya pada *platform* Android dan Windows Phone.

2. Memberikan sebuah literatur/acuan bagi *developer* dalam menentukan pengembangan melalui pendekatan *native mobile application development* dalam mengembangkan aplikasi *mobile*.

1.6. Sistematika Penulisan

Sistematika dalam penulisan dokumentasi pada penelitian ini dapat diuraikan sebagai berikut:

BAB I : Pendahuluan

Bab ini menguraikan tentang latar belakang, rumusan masalah, batasan, tujuan, dan manfaat penelitian dalam Studi Perbandingan Kinerja antara SOAP *web service* dengan REST *web service* dalam Pertukaran Data pada *platform* Android dan Windows Phone.

BAB II : Tinjauan Pustaka

Bab ini menguraikan mengenai kajian pustaka dan teori-teori pendukung yang akan digunakan dalam penelitian Studi Perbandingan Kinerja antara SOAP *web service* dengan REST *web service* dalam Pertukaran Data pada *platform* Android dan Windows Phone. Teori–teori yang berhubungan terdiri dari *web service*, SOAP *Web Service*, REST *Web Service*, Pendekatan Pengembangan Aplikasi *Mobile*, dan Pengujian Perangkat Lunak.

BAB III : Metode Penelitian

Metode penelitian merupakan bagian yang menerangkan mengenai langkah kerja dan metode yang diterapkan dalam Studi Perbandingan Kinerja antara SOAP *web service* dengan REST *web service* dalam Pertukaran Data pada *platform* Android dan Windows Phone.

BAB IV : Perancangan

Menganalisa kebutuhan dan melakukan perancangan dalam Studi Perbandingan Kinerja antara SOAP *web service* dengan REST *web service* dalam Pertukaran Data pada *platform* Android dan Windows Phone.

BAB V : Implementasi

Menjelaskan tahap–tahap implementasi dalam Studi Perbandingan Kinerja antara SOAP *web service* dengan REST *web service* dalam Pertukaran Data pada *platform* Android dan Windows Phone.

BAB VI : Pengujian dan Analisis

Memaparkan mengenai tahap–tahap pengujian yang dilakukan terhadap aplikasi dan melakukan analisis terhadap hasil yang diberikan oleh aplikasi.

BAB VII: Penutup

Memberikan kesimpulan dari penelitian yang telah dilakukan dan saran yang dapat digunakan oleh peneliti selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Di dalam bab ini menguraikan kajian pustaka dan dasar teori yang digunakan dalam melakukan penelitian. Kajian pustaka membahas beberapa penelitian sebelumnya yang digunakan sebagai referensi. Penelitian–penelitian tersebut antara lain penelitian yang dilakukan oleh R. Ramanathan dan T. Korte (Ramanathan, 2014), Novrian Fajar Hidayat dan Ridi Ferdiana (Hidayat, 2012), Iqbal Luthfillah, Arief Andy, dan Budi Darma (Luthfillah, 2014), Jonathan Ledlie, Jeff Shneidman dkk (Ledlie, 2004), Bramwell A. Kasaedja, Rizal Sengkey, dan Oktavian A. Lantang (Kasaedja, 2014), serta penelitian dari Anil Dudhe dan S.S. Sherekar (Dudhe, 2014). Dasar teori menjelaskan mengenai teori–teori yang mendukung dan digunakan dalam melakukan penelitian. Dasar teori yang digunakan dalam penelitian ini antara lain *web service*, SOAP *Web Service*, REST *Web Service*, Pendekatan Pengembangan Aplikasi *Mobile*, dan Pengujian Perangkat Lunak.

2.1. Kajian Pustaka

Pada bagian kajian pustaka ini membahas mengenai beberapa penelitian sebelumnya yang dijadikan referensi untuk melakukan penelitian. Beberapa penelitian tersebut antara lain penelitian yang berjudul "*Software Service Architecture to Access Weather Data Using RESTful Web Services*" (Ramanathan, 2014), "*The Development of Mobile Client Application in Yogyakarta Tourism an Culinary Information System Based on Social Media Integration*" (Hidayat, 2012), "Rancang Bangun Restful *Web Service* untuk Optimalisasi Kecepatan Akses Studi Kasus Aplikasi Sistem Pakar Berbasis *Web*" (Luthfillah, 2014), "*Open Problems in Data Collection Networks"* (Ledlie, 2004), "*Design of Web Service Sam Ratulangi University Library*" (Kasaedja, 2014), dan "*Performance Analysis of SOAP and RESTful Mobile Web Services in Cloud Environment*" (Dudhe, 2014). Penelitian–penelitian yang menjadi referensi disajikan pada **Tabel 2.1**.

Penelitian pertama dengan judul "*Software Service Architecture to Access Weather Data Using RESTful Web Services*" yang dilakukan oleh Ramanathan R. dan Korte T pada tahun 2014. Penelitian ini mengembangkan aplikasi yang diambil langsung dari alat pendeteksi cuaca dalam bentuk *service* dengan tujuan memberikan informasi tentang data cuaca setempat. Pemenuhan data *service* cuaca menggunakan prinsip RESTful *web service*. Permintaan data dari sersor cuaca yang dibutuhkan dipenuhi dengan menggunakan desain RESTful API melalui Uniform Resource Identifier (URI). Transaksi data menggunakan koneksi internet dan untuk data yang digunakan dalam bentuk JSON yang diberikan oleh RESTful *web service* (Ramanathan, 2014).

Tabel 2. 1 Kajian Pustaka

No.	Judul	Obyek (*Input*)	Metode (Proses)	Hasil (*Output*)
1	*Software Service Architecture to Access Weather Data Using RESTful Web Services* (Ramanathan, 2014).	Data Cuaca - Kelembaban - Pencahayaan - Tingi permukaan air - Temperatur - Tekanan udara	RESTful *Web Service*	Aplikasi *service* untuk menampilkan informasi keadaan cuaca lokasi setempat.
2	*The Development of Mobile Client Application in Yogyakarta Tourism an Culinary Information System Based on Social Media Integration* (Hidayat, 2012).	Data wisata dan kuliner di Yogyakarta	RESTful *Web Service,* Geo-Location, Windows Phone.	Aplikasi pencarian untuk mengetahui area wisata dan kuliner yang berada di wilayah Yogyakarta.
3	Rancang Bangun *Restful Web Service* untuk Optimalisasi Kecepatan Akses Studi Kasus Aplikasi Sistem Pakar Berbasis *Web* (Luthfillah, 2014).	Sistem Pakar Perawatan Cabai Merah Keriting	RESTful *Web Service*	Aplikasi perawatan cabai merah keriting dengan menggunakan REST *web service* yang memisahkan tampilan dengan *logic* sehingga beban *web* lebih ringan. Hasil penelitian tersebut menyimpulkan bahwa penggunaan RESTful *web service* mereduksi waktu *loading* akses sistem pakar perawatan cabai hingga 35%.
4	*Open Problems in Data Collection Networks* (Ledlie, 2004).	Data monitoring kesehatan	SOAP dan REST *Web Service*	Masalah penanganan heterogenitas, handling konektivitas berselang, pembatasan kinerja Penempatan layanan

				jaringan dapat diselesaikan dengan Houglass yang merupakan jaringan pengumpulan data .
5	*Design of Web Service Sam Ratulangi University Library* (Kasaedja, 2014).	Data Perpustakaan - Pengelolaan katalog - Pengelolaan peminjaman. - Pengelolaan anggota.	SOAP *Web Service*	Aplikasi *service* dengan fungsi-fungsi yang mendukung proses pengelolaan katalog, peminjaman, dan keanggotaan perpustakaan.
6	*Performance Analysis of SOAP and RESTful Mobile Web Services in Cloud Environment* (Dudhe, 2014).	Sistem SOA dengan server Tomcat Apache dan server Cloud Google App Engine.	SOAP *Web Service,* dan RESTful *Web Service*	Perbandingan layanan web SOAP berdasarkan dengan layanan web REST yang dites pada host lokal menggunakan server Web Apache Tomcat. Penelitian ini menemukan bahwa layanan web REST lebih cepat dari layanan web SOAP. Kemudian pengujian layanan yang sama dilakukan pada server Cloud Google App Engine dan menunjukkan bahwa REST mempunyai kinerja yang lebih baik dibandingkan SOAP.
7	Studi Perbandingan SOAP *Web Service* dengan RESTFUL *Web Service* dalam Pertukaran Data pada *Platform* Android dan Windows Phone.	Data Klimatologi - Wilayah - Kecepatan angin - Arah angin - Curah hujan - Suhu - Kelembaban	REST *Web Service*, SOAP Web Service	Optimasi performa REST *Web Service* pada aplikasi *mobile* dengan menggunakan pendekatan pengembangan *native* pada *platform* Windows Phone dibandingkan SOAP dan REST *Web Service* pada aplikasi *mobile platform* Android.

Sumber: (Ramanathan, 2014) (Hidayat, 2012) (Luthfillah, 2014) (Ledlie, 2004) (Kasaedja, 2014) (Dudhe, 2014)

Penelitian berjudul "*The Development of Mobile Client Application in Yogyakarta Tourism an Culinary Information System Based on Social Media Integration*" dilakukan oleh Hidayat N. Fajar dan Ferdiana Ridi pada tahun 2012. Penelitian ini mengembangkan aplikasi info wisata dan kuliner yang berada di wilayah Yogyakarta. Pengembangan aplikasi ditunjukan pada turis lokal dan mancanegara yang sewaktu–waktu akan berkunjung ke Yogyakarta dan membutuhkan info tentang tempat-tempat wisata maupun tempat kuliner. Prinsip kerja dari aplikasi dengan cara mengirimkan koordinat lokasi wisata dan kuliner dalam mesin pencarian yang telah diketikkan dengan desain RESTful API. Penentuan lokasi pengguna dengan memanfaatkan fasilitas *Geo-Location* dan koneksi internet yang dimiliki oleh *smart phone* windows phone (Hidayat, 2012).

Penelitian dengan judul "*Rancang Bangun Restful Web Service untuk Optimalisasi Kecepatan Akses Studi Kasus Aplikasi Sistem Pakar Berbasis Web*" dilakukan pada tahun 2014 oleh Luthfillah Iqbal, Soebroto A. Andy, dan Darma Budi. Aplikasi sistem pakar perawatan tanaman cabai merah keriting yang dikembangkan pada penelitian ini ditujukan untuk mengetahui bahwa penggunaan RESTful *web service* dapat meringankan *bandwith* aplikasi sistem pakar. Dalam penggunaan REST *web service* dapat memisahkan tampilan dengan *logic* sehingga beban *web* lebih ringan. Penelitian ini menjadi acuan bagi para *developer* lain dalam hal membangun sebuah *website* (Luthfillah, 2014).

Penelitian yang mempunyai judul "*Open Problems in Data Collection Networks*" dilakukan oleh Jonathan Ledlie, Jeff Shneidman, Matt Welsh, Mema Roussopoulos dan Margo Seltzer pada tahun 2004. Penelitian ini didasari pada sensor jaringan, continuous queries (CQ), dan lainnya yang termotivasi oleh aplikasi yang bertujuan untuk gerbang agregat, mangasimilasi dan berinteraksi dengan skor jaringan sensor secara pararel. Penelitian jaringan sensor adalah membangun beberapa komponen dari bawah ke atas, yang berurusan dengan isu-isu seperti konektivitas nirkabel dan data tahan baterai, CQ, peer to peer yang membangun dari atas ke bawah, memeriksa jaringan, penamaan query desentralisasi, dan skala. Masalah penanganan heterogenitas, handling konektivitas berselang, pembatasan kinerja Penempatan layanan jaringan dapat diselesaikan dengan Houglass yangmerupakan jaringan pengumpulan data. (Ledlie, 2004)

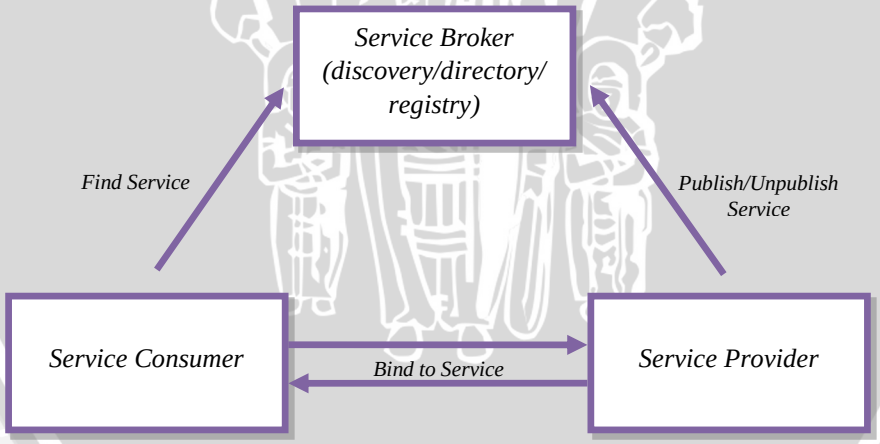
Penelitian yang dilakukan pada tahun 2014 oleh Kasaedja A. Bramwell et al dengan judul "*Design of Web Service Sam Ratulangi University Library*". Penelitian ini mengembangkan aplikasi untuk memberikan kemudahan service dalam mendukung proses pengelolaan katalog, peminjaman, dan keanggotaan perpustakaan. Pemenuhan *service* data perpustakaan menggunakan prinsip SOAP *web service*. Dengan adanya aplikasi yang didesain dengan sesederhana mungkin, tampilan interface yang menarik, mudah dipahami, dan mudah dioperasikan, sehingga proses pengelolaan katalog, data peminjam, dan keanggotaan perpustakaan menjadi lebih mudah (Kasaedja, 2014).

Penelitian yang dilakukan pada tahun 2014 oleh Anil Dudhe dan S.S. Sherekar dengan judul "*Performance Analysis of SOAP and RESTful Mobile Web Services in*

Cloud Environment". Penelitian ini membandingkan antara layanan SOAP *web service* dengan REST *web service*. Penelitian berfokus pada pengembangan kedua *web services* dengan berbeda parameter. Masing-masing *web service* yaitu SOAP dan REST API + Jersey API menerapkan XML. Percobaan dilakukan berbeda pada keduanya dan diuji pada server Apache Tomcat, Android Emulator baik sebagai server *Cloud Google App Engine*. Hasil percobaan menunjukkan bahwa layanan *web* REST lebih baik dalam kinerja daripada layanan *web* SOAP (Dudhe, 2014).

Penelitian yang akan dilakukan berjudul "*Studi Perbandingan SOAP Web Service dengan RESTFUL Web Service dalam Pertukaran Data pada Platform Android dan Windows Phone*". Penelitian ini bertujuan untuk menganalisis perbandingan performa pengambilan data suatu aplikasi atau perangkat lunak. Studi kasus yang digunakan adalah aplikasi *mobile* monitoring klimatologi Karangploso Malang. Dalam penelitian ini metode pertukaran data yang digunakan adalah REST *web service* dan SOAP *web service*. Analisa REST dan SOAP *web service* yang dilakukan mengarah kepada kinerja *web service* berdasarkan *respon time*. Perangkat lunak yang akan digunakan dalam analisa ditujukan untuk perangkat *mobile* android dan windows phone. Penelitian ini dilakukan dengan pendekatan pengembangan *native mobile application development*. Penelitian ini diharapkan dapat memberikan solusi baru bahwa performa REST *web service* lebih efektif pada aplikasi *mobile platform* Windows Phone dibandingkan dengan SOAP dan REST *web service* pada aplikasi *mobile platform* android.

2.2. *Web Service*



Gambar 2.1 Arsitektur Web Service
Sumber: (Utama, 2012)

Web service menyediakan standar operasi antara perangkat lunak yang berbeda dan bejalan pada *platform* atau *framework* yang berbeda (Lee, 2014). Teknologi *web service* berorientasi pada *service* dan umumnya terintegrasi dengan aplikasi *web* untuk menyalurkan proses bisnis (Al-Fedaghi, 2011). *Web service* biasa digunakan untuk menyalurkan proses binis antara aplikasi dengan berbeda *vendor*, bahasa pemrograman, *platform* dan aplikasi *client* (Al-Fedaghi, 2011). Arsitektur *web service* menyediakan model konseptual dan konteks untuk

memahami *web service* dan berbagai komponen yang berhubungan (Lee, 2014). Arsitektur dari *web service* dan komponen–komponen yang berkaitan ditunjukkan dalam **Gambar 2.1** (Utama, 2012).

- *Service Provider*: berfungsi menyediakan layanan/*service* dan mengolah sebuah *registry* supaya layanan-layanan tersebut dapat tersedia.
- *Service Registry*: berfungsi sebagai lokasi *central* yang mendriskripsikan semua layanan/*service* yang telah di-*register*.
- *Sevice Consumer*: berfungsi sebagai peminta layanan yang mencari dan menemukan layanan yang dibuthkan serta menggunakan layanan tersebut.
- *Publish/Unpublish Service*: menerbitkan/menghapus layanan ke dalam atau dari *registry*.
- *Find Service*: *Service Consumer* mencari dan menentukan layanan yang dibutuhkan.
- *Bind to service*: *Service Consumer* yang telah menenukan layanan yang dicari, kemudian melakukan ikatan/*binding* ke *Service Provider* untuk melakukan interaksi dan mengakses layanan/*service* yang disediakan oleh *Service Provider*.

Web service memungkinkan dalam pengaksesan data dengan mudah serta dapat mengurangi penggunaan waktu. Pertukaran pesan yang dilakukan oleh *web service* pada awalnya menggunakan format data XML. Penggunaan format data JSON untuk pertukaran pesan sekarang ini lebih disukai daripada XML pada implementasi *web service*. Serialisasi data yang lebih baik dan juga proses pemetaan yang mudah merupakan kelebihan JSON (Priya, 2015). Perbandingan JSON dan XML seperti yang ditunjukkan pada **Tabel 2.2** (book, 2015).

Tabel 2.2 Perbandingan JSON dan XML

JSON	XML
Singkatan dari *JavaScript Object Notation*.	Singkatan dari *Extensible Markup Language*.
Javascript merupakan perluasan JSON.	Standard Generalized Markup Language (SGML) merupakan perluasan XML.
JSON dikembangkan oleh *Douglas Crockford*. JSON merupakan salah satu format jenis berbasis text atau standar untuk pertukaran data yang dapat dibaca oleh manusia.	XML dikembangkan oleh World Wide Web Consortium (W3C). XML merupakan *Markup Language* yang memiliki format berisikan seperangkat aturan dalam pengkodean dokumen yang dapat dibaca oleh manusia dan mesin.
JSON lebih ringan daripada XML, JSON memiliki format data yang berturut – turut dan dengan redudansi yang	XML tidak ringan layaknya JSON, XML memiliki tag awal dan tag akhir, dan membutuhkan lebih banyak karakter

sedikit. JSON tidak mempunyai tag awal dan tag akhir seperti XML.	daripada JSON untuk mewakili data yang sama.
JSON mendukung tipe data integer dan string, dan JSON juga mendukung array.	XML tidak menyediakan tipe data sehingga perlu dipecah menjadi tipe data tertentu. XML tidak mendukung array secara langsung.
JSON tidak mendukung komentar.	XML mendukung komentar.
JSON tidak mendukung untuk Namespaces.	XML mendukung Namespaces.
JSON berorientasi pada data dan dapat dipetakan dengan mudah.	XML berorientasi pada dokumen dan dibutuhkan usaha lebih untuk pemetaan.
JSON lebih baik digunakan untuk *web service*.	XML lebih baik digunakan untuk konfigurasi SOAP *web service*.

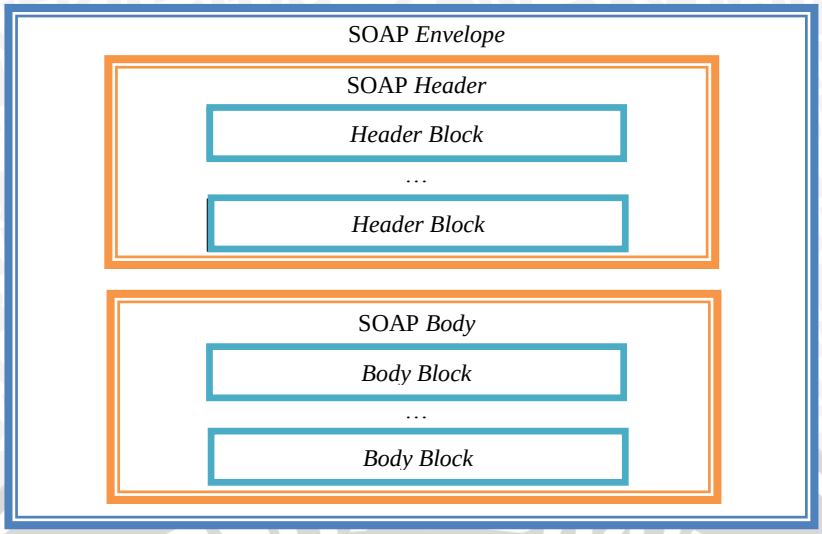
Sumber: (book, 2015)

Implementasi *Web service* pada awalnya digunakan untuk Simple Object Access Protocol (SOAP) dan Web Service Description Language (WSDL). WSDL merupakan standar *interface* yang digunakan untuk mengakses aplikasi. SOAP merupakan standar protokol *inter operable* yang digunakan untuk menghubungkan aplikasi. Beberapa tahun terakhir pendekatan *alternative* menggunakan Representational State Transfer (REST) telah banyak digunakan untuk aplikasi skala internet (Ramanathan, 2014).

2.2.1. *Simple Object Access Protocol* (SOAP)

SOAP adalah protokol berbasis XML yang digunakan untuk pertukaran informasi antara aplikasi. SOAP dikembangkan awalnya oleh Microsoft untuk menggantikan menggantikan metode tradisional seperti Distributed Component Object Model (DCOM) dan Common Object Request Broker Architecture (CORBA). Keuntungan dalam penggunaan SOAP yang utama, adalah memungkinkan untuk pertukaran informasi antara aplikasi yang berjalan pada *platform* yang berbeda serta dikembangkan dengan bahasa pemrograman yang berbeda. Faktor interoperabilitas tersebut memberikan tingkat fleksibilitas yang lebih besar dalam mengembangkan aplikasi yang menggunakan *web service* (Ramanathan, 2014).

Pesan SOAP digambarkan sebagai *envelope* yang membungkus data yang harus dikirim. Elemen *envelope* berisi elemen *header* yang opsional dan elemen *body* yang wajib. Elemen *envelope* dan *header* mengandung isi berupa spesifikasi aplikasi dan bukan merupakan bagian dari spesifikasi SOAP. *Header* berisikan informasi seperti bagaimana pesah harus diproses dan juga informasi yang berkaitan dengan SOAP *engine*. Elemen *body* berisikan bagian utama untuk mengakhiri informasi dari pesan SOAP dalam bentuk dokumen XML (Ramanathan, 2014). Struktur pesan pada SOAP *web service* ditunjukkan dalam **Gambar 2.2**.



Gambar 2.2 Struktur Pesan SOAP
Sumber: (Lee, 2014)

SOAP pada umumnya diterapkan menggunakan protokol HTTP. SOAP memiliki dua jenis gaya pesan yang digunakan untuk pertukaran informasi yaitu Remote Procedure Call (RPC) dan gaya pesan dokumen. Gaya pesan RPC *web service* ini digabungkan dengan erat dan adanya perubahan yang dibuat pada *interface* RPC akan merusak kesepakatan antara *client* dengan *service*. Client melakukan *request* yang dinyatakan ke dalam suatu *method* dengan serangkaian *argument* yang akan mengembalikan tanggapan mengandung nilai. *Client* mengirimkan *request* dan menunggu tanggapan. Gaya pesan dokumen pada SOAP *web service* antara *request* dan *response* dibuat dalam format file XML. Gaya pesan dokumen pada SOAP *web service* lebih handal, terukur dan dapat meningkatkan kinerja. Gaya pesan SOAP RPC tidak disarankan pada implementasi Service Oriented Architecture (SOA) (Ramanathan, 2014). *Request* dan *response* sederhana dari SOAP diperlihatkan dalam **Gambar 2.3**.

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelo
pe/">
  <soap:Body>
    <getProductDetails
          xmlns="http://gadget.com/ws">
      <productID>15282054/A</productID>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```

SOAP *Request*

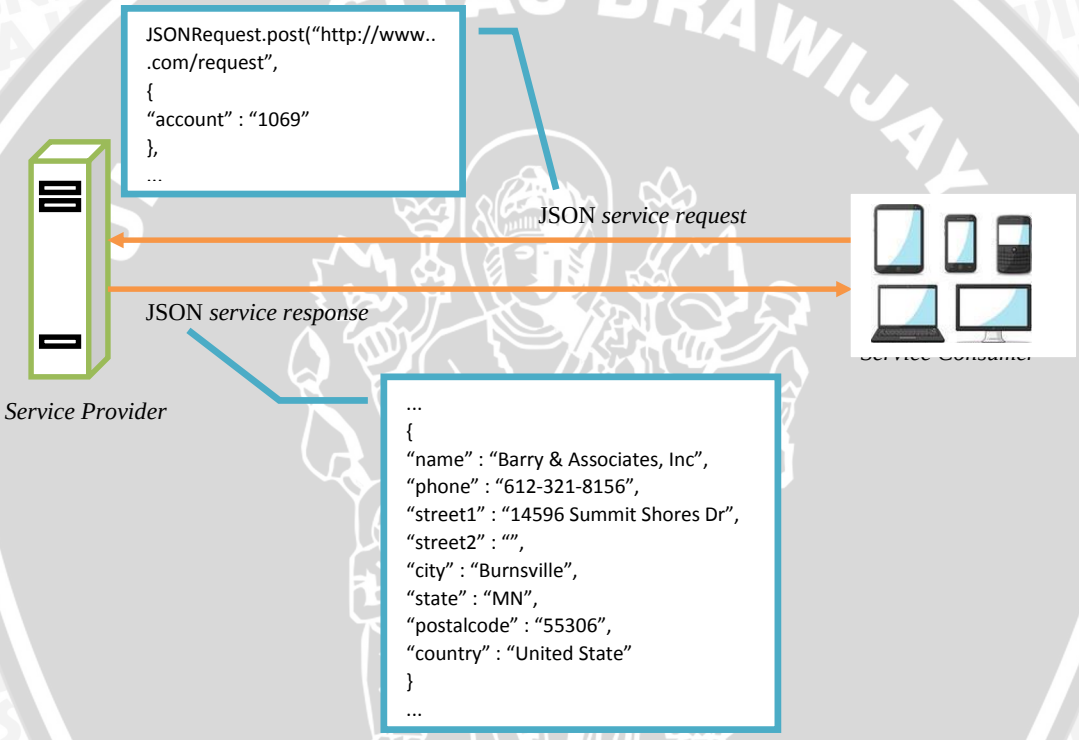
SOAP *Response*

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/e
nvelope/">
  <soap:Body>
    <getProductDetailResponse
        xmlns="http://gadget.com/ws">
      <getProductDetailResult>
          <productName>ipod</productName>
          <productID>15282054/A</productID>
          <description>iPod, 40GB –
              Silver</description>
          <price currency="EUR">34.00</price>
          <inStock>false</instock>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
```

Gambar 2.3 SOAP *Request* dan SOAP *Response*
Sumber: (Ramanathan, 2014)

2.2.2. *Representational State Transfer* (REST)

REST adalah salah satu jenis *web service* yang menerapkan konsep perpindahan antar *state*. *State* REST yang dimaksud adalah arsitektur yang berupa *client server* dimana *client* mengirimkan *requet* ke *server* kemudian *server* memproses *request* tersebut dan mengembalikan respon. *Request* dan respon terjadi pada proses representasi *resource* (Padiya, 2013). Setiap *resource* pada REST diidentifikasi oleh Uniform Resource Identifier (URI). Penyedia *service* pada REST adalah *resource* dan pengguna yang mungkin atau tidak juga bisa menjadi *resource*. Dasar ID dari REST adalah dengan menggunakan mekanisme HTTP, untuk menghubungkan aplikasi daripada menggunakan mekanisme yang kompleks seperti CORB, RPC dan SOAP (Ramanathan, 2014). Proses sederhana dari *request* REST dan *response* yang berupa JSON ditunjukkan dalam **Gambar 2.4**.



Gambar 2.4 REST *Request* dan REST *Response*
Sumber: (Ramanathan, 2014)

REST berfokus mengenai interaksi antara *resource* dan pengubahan *state*, REST menawarkan implementasi yang sederhana dari *web service* dibandingkan dengan pengiriman dan penerimaan pesan seperti pada SOAP *web service*. Prinsip dari REST *web service* diantaranya:

1. Identifikasi *resource* melalui URI: REST *web service* menyediakan sejumlah *resource* yang mengidentifikasi tujuan dari interaksi dengan *client*. *Resource* diidentifikasi dengan URI yang mengakomodasi ruang pengalamatan *resource* dan penemuan *service*.
2. Keseragaman dalam *resource*: setiap *resource* pada REST dimanipulasi melalui empat operasi GET, PUT, POST, dan DELETE untuk membaca, membuat,

memperbarui dan menghapus masing–masing *resource*. Operasi PUT bertanggung jawab menciptakan *resource* yang dapat dihapus dengan menggunakan perintah DELETE. Operasi POST transfer *resource* dari satu *state* ke *state* yang lain.

3. Pesan dengan *self-description*: *resource* pada RESTful dikategorikan dengan cara menyajikan *resource*, sehingga dapat diakses dalam berbagai format data seperti HTML, PDF, XML, JPEG, teks biasa dan lain-lain.
4. Penggunaan *hyperlink* untuk interaksi *stateful*: semua interaksi dengan *resource* yang *stateless* merupakan pesan *request self-comprehended*. Interaksi *stateful* merupakan salah satu interaksi *state* yang eksplisit. Ada cara untuk bertukar *state* seperti menulis ulang URI, *cookies* yang tersembunyi. *State* juga dapat ditanam pada pesan respon sehingga dapat digunakan sebagai referensi pada waktu yang akan datang.

Alasan utama yang membuat REST lebih banyak digunakan adalah proses kerja yang sederhana dimana *request* merupakan operasi HTTP dan *response* berupa *file* XML polos, JSON atau Resource Description Framework (RDF). Kelemahan REST adalah kurangnya standar dan sering dipandang sebagai kelemahan dalam implementasi SOA. REST tetap lebih disukai untuk implementasi SOA karena proses kerja yang sederhana.

2.2.3. Perbandingan SOAP dan REST

SOAP adalah sebuah protokol dan REST adalah paradigma arsitektur yang keduanya paling banyak digunakan dalam membangun *web service*. SOAP dan REST memiliki perbedaan dalam perspektif dan penggunaan. SOAP dan REST masing–masing memiliki kelebihan dan kekurangan, keduanya dapat digunakan untuk membuat aplikasi yang memanfaatkan *web service*. Keputusan dalam menggunakan SOAP atau REST tergantung pada desain aplikasi, implementasi dan penggunaan. SOAP akan menjadi sebuah pilihan ketika aplikasi yang dibangun memiliki desain yang kompleks dan harus diamankan misalnya aplikasi pemerintahan atau perbankan. REST akan menjadi pilihan dalam membangun suatu aplikasi yang berbasis pada *web* dan *mobile* dikarenakan memiliki sifat yang ringan, *request* dan *response* tidak terikat dengan format data XML (Ramanathan, 2014). Berikut beberapa perbedaan antara SOAP dan REST seperti yang disajikan pada **Tabel 2.3**.

Tabel 2. 3 Perbandingan SOAP dan REST

SOAP	REST
Teknologi yang sudah berkembang sejak akhir tahun 90an.	Teknologi baru sebagai bandingan dari SOAP. Pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000.
Dalam perusahaan dan skenario Business 2 Business (B2B), SOAP masih sangat menarik untuk diterapkan.	Ini tidak mengatakan kalau REST tidak siap untuk perusahaan. Pada kenyataannya seperti yang diketahui

	keberhasilan penerapan RESTful pada aplikasi perbankan.
Interaksi *Client-Server* adalah gabungan erat.	Interaksi *Client-Server* adalah gabungan longgar.
SOAP menyusul REST dalam hal implementasi karena memiliki standar yang ditetapkan dalam pembangunan SOAP.	Para pengembang REST berpendapat REST memiliki fleksibilitas *interface*.
Perubahan pada sisi penyedia *service* SOAP berarti pada sisi *client* juga harus dilakukan perubahan kode.	Perubahan pada sisi penyedia *service* REST tidak perlu dilakukan perubahan kode pada sisi *client*.
SOAP memiliki *payload* yang berat dibandingkan dengan REST.	REST ringan untuk melakukan transfer data ringan melalui *interface* yang umum dikenal dengan URI.
SOAP membutuhkan parsing biner.	REST mendukung semua jenis data secara langsung.
SOAP kurang sesuai untuk infrastruktur jaringan nirkabel.	REST sesuai untuk infrastruktur jaraingan nirkabel.
SOAP *web service* selalu mengembalikan data XML.	REST *web service* memberikan fleksibilitas dalam hal tipe data yang dikembalikan.
SOAP mengonsumsi lebih banyak bandwidth karena respon yang diberikan bisa 10 kali lebih banyak byte dibandingkan dengan REST.	REST mengonsumsi lebih sedikit bandwidth karena respon yang diberikan ringan.
SOAP menggunakan *request* POST dan memerlukan *request* XML yang kompleks sehingga membuat respon berat atau sulit.	RESTful API dapat menggunakan *request* GET sederhana, *server proxy* menengah sehingga respon ringan atau mudah.
SOAP menggunakan HTTP berbasis API yang tertuju pada satu atau lebih HTTP URI dan respon yang dihasilkan dalam bentuk XML/JSON. Skema respon ditentukan setiap objek.	REST menambahkan unsur yang menggunakan standar URI dan juga operasi penting HTTP (yaitu GET / POST / PUT dll.).
Bahasa , *platform*, dan transportasi *agnostic*.	Bahasa dan *platform agnostic.*
Dirancang untuk menangani lingkungan komputasi terdistribusi.	Mengansumsikan model komunikasi *point-to-poin* tidak untuk lingkungan komputasi terdistribusi yang mana pesan dapat tersalur melalui satu atau lebih perantara.
Sulit dalam hal pengembangan, membutuhkan *tools*.	Jauh lebih mudah untuk pengembangan *web service* daripada SOAP.

Asumsi yang salah kalau SOAP lebih aman. SOAP menggunakan *WS-security. WS-security* dibuat karena spesifikasi SOAP adalah *transport-independent* dan tidak ada asumsi keamanan dapat dibuat yang tersedia pada *transport layer.*	REST mengansumsikan transportasi pada HTTP (atau HTTPS) dan mekanisme keamanan yang dibangun untuk protokol akan tersedia.
Standar yang berlaku untuk *web service* dan memiliki dukungan yang lebih baik dari standar yang lainnya dan *tool* dari *vendor*.	Kurangnya dukungan keamanan yang standar, kebijakan, pesan yang handal, dll., sehingga untuk aplikasi yang memiliki fungsional yang lebih canggih lebih sulit untuk dikembangkan.

Sumber: (Wagh, 2012)

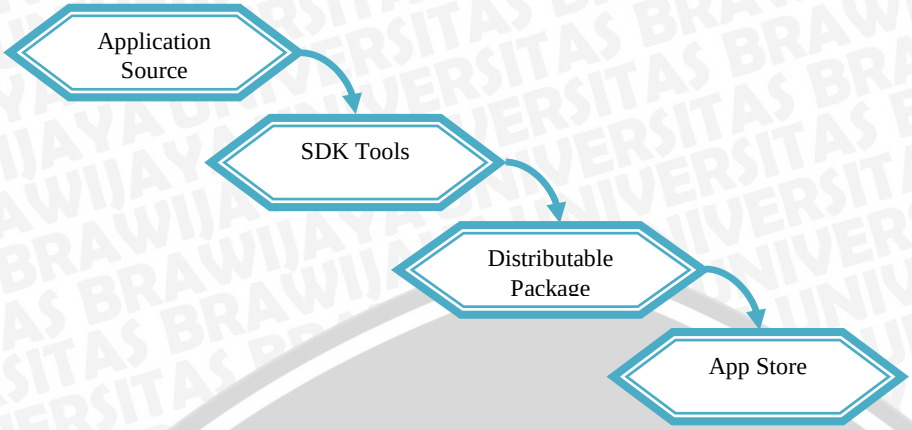
2.3. Pendekatan Pengembangan Aplikasi *Mobile*

Perkembangan aplikasi perangkat *mobile* semakin berkembang dengan pesat. Pengembangan aplikasi yang dilakukan dapat menggunakan beberapa pendekatan yang sesuai dengan kebutuhan dan tujuan yang diharapkan. Pendekatan pengembangan untuk aplikasi perangkat *mobile* diantaranya yaitu *Native Mobile Application Development*, *Hybrid Mobile Application Development*, dan *Web Mobile Application Development*. Penelitian ini menggunakan pendekatan pengembangan untuk aplikasi perangkat *mobile* yaitu *native mobile application development* untuk Windows Phone dan Android. Pendekatan *native mobile application development* dilakukan dengan tujuan untuk menyelaraskan pertukaran data *web service* sehingga akan memudahkan saat pengambilan data pada proses pengujian.

2.3.1. *Native Mobile Application Development*

Native mobile application merupakan file biner *executable* yang dirancang untuk sistem operasi dan komponen perangkat *mobile*. Aplikasi *native* diinstal pada sistem operasi perangkat *mobile* dan pengguna dapat langsung menjalankannya. Tahap–tahap dalam pengembangan aplikasi dengan pendekatan *native mobile* seperti yang diperlihatkan dalam **Gambar 2.5**.

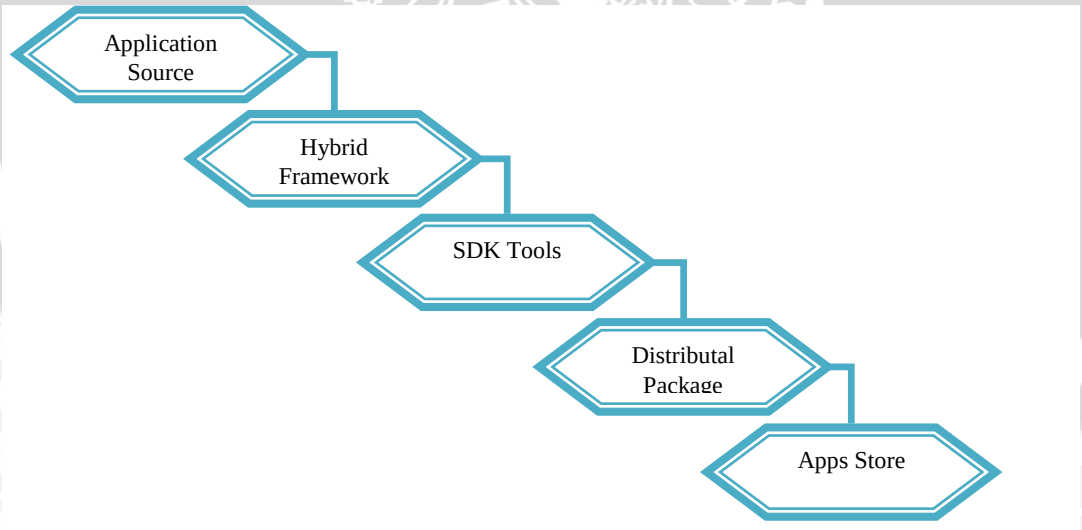
Aplikasi *native* dapat dengan bebas mengakses fungsional perangkat *mobile* seperti *dial number*, kamera, SMS, dan lokasi atau GPS. Pengembangan aplikasi *native* memerlukan pengetahuan yang mendalam mengenai *platform* tertentu daripada pengembangan aplikasi *mobile* yang lain. Keuntungan dari aplikasi *native* kinerjanya mudah dan cepat, memiliki kemampuan untuk memanfaatkan perangkat lunak dan perangkat keras khusus dari perangkat *mobile*. Kelemahan yang dimiliki dalam pengembangan aplikasi *native* yaitu kode yang dibuat untuk *platform* tertentu tidak dapat digunakan untuk *platform* yang lain (Tun, 2014).



Gambar 2.5 *Native Mobile Application*
Sumber: (Tun, 2014)

2.3.2. *Hybrid Mobile Application Development*

Pengembangan aplikasi *mobile hybrid* dilakukan dengan menggabungkan pengembangan *native* dengan teknologi *web*, namun aplikasi *hybrid* berperilaku lebih seperti aplikasi *web* daripada aplikasi *mobile*. Tahap pengembangan aplikasi dengan pendekatan *Hybrid* seperti yang digambarkan dalam **Gambar 2.7**.

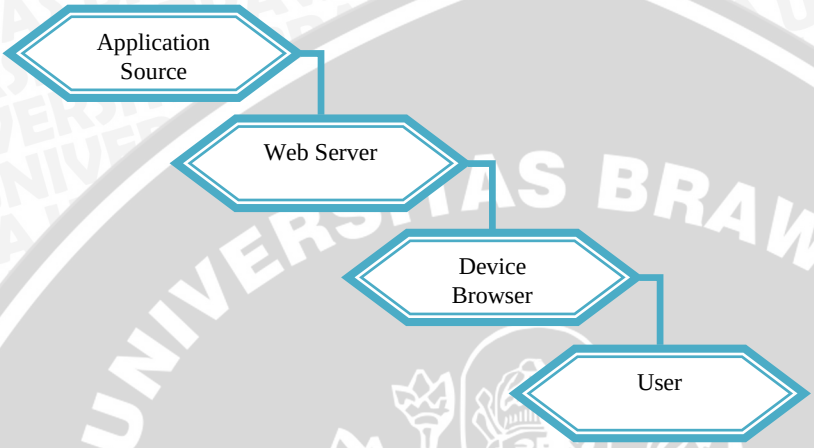


Gambar 2.6 *Hybrid Mobile Application*
Sumber: (Tun, 2014)

Keuntungan yang didapat dalam mengadopsi pendekatan *hybrid* adalah menghemat biaya dan waktu dalam pengembangan dan pemeliharaan aplikasi. Pendekatan *hybrid* menghasilkan kode yang dapat berjalan dalam berbagai *platform* dan perangkat keras. Kekurangan yang terdapat pada pendekatan *hybrid* adalah kurang dukungan secara penuh untuk mengakses fitur asli bawaan dari *device* (Tun, 2014).

2.3.3. *Web Mobile Application Development*

Aplikasi *web mobile* merupakan halaman *web* yang dapat berada pada server atau satu set HTML, CSS, JavaScript, dan file terkait lainnya yang disimpan pada perangkat tertentu. Halaman – halaman *web* diformat untuk *smartphone* dan tablet, dan diakses melalui *mobile* (*web browser*). Alur untuk pengembangan aplikasi dengan pendekatan *web mobile* seperti yang diperlihatkan dalam **Gambar 2.8**.



Gambar 2.7 *Web Mobile Application*
Sumber: (Tun, 2014)

Kelebihan yang dimiliki oleh aplikasi *web mobile* antara lain dalam pembuatan tidak terlalu kompleks, lebih murah, lebih cepat dalam pembuatan dan mudah dalam pemeliharaan. Aplikasi *web* tidak tergantung pada salah satu jenis *platform*. Aplikasi *web* dapat berjalan pada *web browser* dengan *device* apa saja yang menginstalnya. Kekurangan yang dimiliki oleh pendekatan aplikasi *web mobile* diantaranya aplikasi tidak dapat dijalankan apabila tidak terhubung dengan internet. Sepenuhnya aplikasi berjalan pada *web browser* dan tidak bisa mengakses fitur bawaan dari *device* (Tun, 2014).

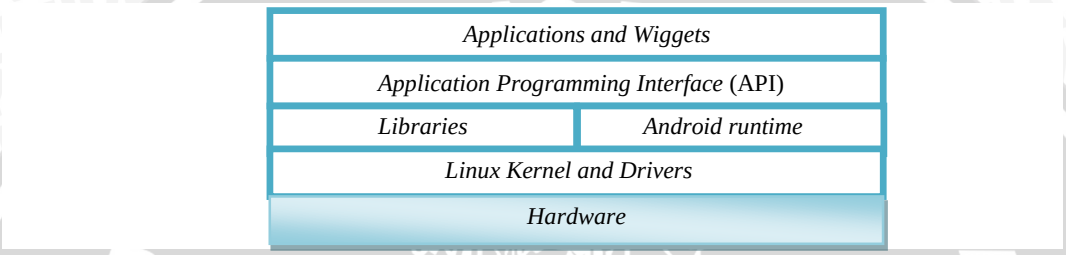
2.4. Perbedaan Android dan Windows Phone

Arsitektur Android dan Windows Phone merupakan penjelasan dari penguraian suatu sistem yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan, kesempatan, dan hambatan yang terjadi. Android merupakan Tumpukan perangkat lunak *open source* yang mencakup sistem operasi, *middleware*, dan aplikasi utama bersama dengan satu set API untuk menulis aplikasi *mobile* dikenal dengan Android (Meier, 2009). Android merupakan proyek yang di prakarsai oleh Google melalui *Open Handset Alliance*, yang mencakup 30 perusahaan di Information and Communications Technology (ICT) (Sharma, 2014). Sedangkan Windows Phone adalah keluarga sistem operasi perangkat bergerak yang dikembangkan oleh Microsoft dan merupakan pengganti *platform* Windows *Mobile*. Windows

Phone 8 adalah generasi kedua dari sistem operasi Windows Phone untuk *smartphone* dari Microsoft.

2.4.1. Android

Android merupakan sistem operasi Android berbasis pada Linux kernel 2.6.x, yaitu kernel monolitik. Kernel tersebut tersusun atas *driver* untuk *hardware* perangkat *mobile*: layar, kamera, *keyboard*, Universal Serial Bus (USB), ahasath, dll. Kernel menyediakan antarmuka *hardware* dan manajemen memori, proses dan *resource* yang lain. Android aplikasi dikembangkan dengan menggunakan bahasa pemrograman java (Sharma, 2014). Arsitektur Android ditunjukkan dalam **Gambar 2.6**.



Gambar 2.8 Arsitektur Android
Sumber: (Sharma, 2014)

Android dibangun dengan menggunakan asas *object oriented*, dimana elemen-elemen penyusun sistem operasinya berupa objek yang dapat kita gunakan kembali atau lebih dikenal dengan *reusable*. Arsitektur Android memiliki 4 layer kompenen. Analisis arsitektur Android dibagi dalam beberapa komponen elemen dimulai dari paling bawah hingga paling atas, diantaranya:

1. Linux *Kernel*

   Merupakan tumpukan paling bawah pada arsitektur Android. Kernel mempunyai peran sebagai *abstraction layer* antara *hardware* dan keseluruhan *software*. Android dibangun diatas kernel Linux 2.6. linux merupakan sistem operasi terbuka yang bagus dalam manajemen memori dan proses. Kernel linux menyediakan driver layar, kamera, *keypad*, WiFi, *flash memory*, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan keamanan.

2. Android *Runtime* dan Layer Libraries

   Android *Runtime* dan layer *libraries* terletak pada satu lapisan setelah kernel Linux. Proses aplikasi monitoring klimatologi BMKG Karangploso dijalankan pada layer ini, karena Android *Runtime* mempunyai dua bagian diantaranya adalah *Core Libraries* dan *Dalvik Virtual Machine*. Core *Libraries* berfungsi untuk menerjemahkan Bahasa *Java*. *Dalvik Virtual Machine* merupakan sebuah mesin virtual berbasis *register* untuk menjalankan fungsi-fungsi seacra efisien.

   Layer *libraries* merupakan layer dimana fitur-fitur pada Android ditempatkan. Layer ini diakses ketika menjalankan aplikasi. Beberapa *library* yang terdapat pada Android diantaranya adalah *Library* Media untuk memutar media audio video,

Library Graphic untuk tampilan, *Library* SQLite untuk mendukung *database*, dan lain-lain. *Library-library* tersebut bukan aplikasi yang dapat berjalan sendiri, namun hanya dapat digunakan oleh program yang berada pada level diatasnya.

3. Layer *Applications Framework*

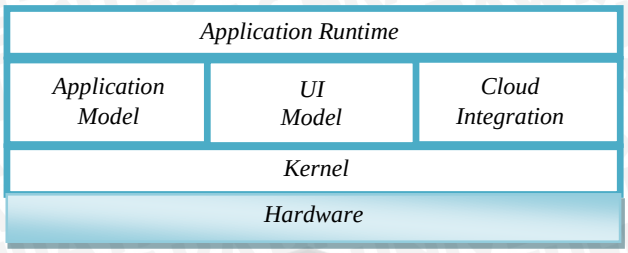
Layer *apllications framework* merupakan layer dimana para *developer* dapat melakukan pengembangan atau pembuatan aplikasi yang akan dijalankan pada sistem operasi Android. *Developer* dalam pembuatan aplikasi monitoring klimatologi BMKG Karangploso mendapat akses penuh dari layer ini untuk memanfaatkan API (*Android Protocol Interface*) *framework* yang juga digunakan dalam *core libraries* pada layer sebelumnya. Komponen-komponen yang termasuk didalam *Applications Framework* adalah *views* (digunakan untuk membangun aplikasi meliputi *lists*, *grids*, *text boxes*, *buttons*, dan *embeddable*), *content provider* (memungkinkan aplikasi untuk dapat mengakses data dari aplikasi lain atau untuk membagi data yang dimilikinya), *resource manager* (menyediakan akses ke *non-code resource* misalnya *localized strings*, *graphic*, dan *layout files*), *notification manager* (memungkinkan untuk semua aplikasi menampilkan *custom alerts* pada *status bar*), dan *activity manager* (memanage *life cycle of* dari aplikasi dan menyediakan *common navigation backstack*).

4. Layer *Applications* dan *Widget*

Layer teratas dari arsitektur Android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling nampak pada pengguna ketika menjalankan program. *User* hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan pada Android *Runtime* dengan menggunakan kelas dan *service* yang tersedia pada *framework* aplikasi. Aplikasi Android berbeda dengan sistem operasi lainnya, karena baik aplikasi *native* maupun aplikasi lain berjalan diatas lapisan aplikasi dengan menggunakan pustaka API yang sama.

2.4.2. Windows Phone

Windows Phone 8 dirilis pada 29 Oktober 2012 di San Francisco. Industri *smartphone* tidak banyak berkembang karena grid ikon yang diciptakan Apple telah ditiru oleh Android. Pada Windows Phone diputuskan untuk tidak menggunakan hal yang sama. *Interface* Windows Phone disebut *Live Tile*, berupa kotak-kotak yang menampilkan berbagai informasi dengan penampilan yang sangat berbeda dari iPhone atau Android. Windows Phone 8 memiliki *interface* yang dikenal sebagai *user interface modern* seperti pendahulunya. Aplikasi Windows Phone dikembangkan dengan menggunakan Microsoft Visual Studio atau .NET *framework* dan bahasa pemrograman C# (Morena, 2013). Arsitektur Windows Phone ditunjukkan dalam **Gambar 2.7**.



Gambar 2.9 Arsitektur Windows Phone
Sumber: (Jeffery, 2010)

Arsitektur Windows Phone lebih *simple* dan mempunyai 3 layer dibandingkan dengan arsitektur Android yang mempunyai 4 layer. Hal ini bisa jadi petimbangan bagi para *developer* dalam membangun atau membuat sebuah aplikasi didasarkan pada arsitektur sistem operasinya. Adapun susunan layer dari bawah hingga atas dalam arsitektur Windows Phone diantaranya yaitu:

1. *Kernel*

Sama seperti Android, *kernel* merupakan tumpukan paling bawah pada arsitektur Windows Phone. Sistem operasi Windows Phone sebelumnya, yaitu versi 7, menggunakan inti program (*kernel*) Windows CE. *Kernel* mempunyai tugas untuk mengorganisir jalannya beragam aplikasi agar dapat diakses oleh *hardware*. *Kernel* Windows CE juga digunakan untuk mengembangkan Windows *Mobile*, sistem operasi ponsel Microsoft sebelum Windows Phone 7.

Generasi dari *kernel* Windows CE selanjutnya adalah *kernel* Windows NT yang digunakan pada Windows Phone 8. *Kernel* Wndows NT merupakan sebuah *kernel* yang digunakan di sistem operasi Windows 8 untuk komputer dan tablet. Dengan adanya perubahan ini, *developer* aplikasi Windows dapat membuat aplikasi untuk Windows Phone 8. Pengembangan aplikasi untuk Windows Phone 8 bisa dilakukan dengan beragam bahasa pemrograman.

2. *Applications Model*, UI *Model*, dan *Cloud Integration*

Layer kedua setelah *kernel* pada Windows Phone berbeda dengan layer pada Android. Layer kedua Windows Phone terdiri dari 3 bagian diantaranya yaitu:

- *Application Model*

Application model pada Windows Phone menangani manajemen, perizinan, isolasi, dan pembaharuan *software*.

- UI *Model*

Pada bagian UI *model* Windows Phone berkaitan dengan manajemen sesi dan model orientasi navigasi halaman.

- *Cloud Integration*

Bagian dari *Cloud Integration* adalah lapisan yang menangani ikatan ke Xbox *Live*, *Bing*, dan integrasi *LiveID*, juga termasuk layanan lokasi serta *push notification framework*.

3. *Application Runtime*

Layer paling atas dari Windows Phone adalah *Application Runtime*. *Application Runtime* pada Windows Phone dapat berupa *Silverlight* atau XNA. Perbedaan yang

mendasar dari *Silverlight* dan XNA yaitu *Silverlight* merupakan API yang umumnya digunakan *developer* dalam pembuatan aplikasi, sedangkan XNA biasanya digunakan *developer* dalam pembuatan Game. Pengembangan aplikasi monitoring klimatologi BMKG Karangploso dalam penelitian ini dilakukan dengan pengembangan *native* yang dibangun menggunakan *tools Silverlight*.

2.5. Pengujian Perangkat Lunak

Pengujian perangkat lunak membutuhkan perancangan kasus uji dengan tujuan untuk mendapatkan kesalahan dengan waktu yang relative singkat dan usaha yang minimum. Semua elemen sistem penting untuk menguji sebuah *object oriented sistem* pada berbagai macam level yang berbeda dalam sebuah usaha untuk menemukan kesalahan–kesalahan yang mungkin terjadi (Pressman, 2001).

Ada banyak metode–metode perancangan kasus uji yang telah berkembang guna menyediakan pendekatan pengujian secara sistematis bagi para pengembang. Metode perancangan kasus uji ini juga menyediakan mekanisme yang dapat membantu kelengkapan dari pengujian dan memberikan kemungkinan tertinggi untuk dapat menemukan berbagai kesalahan dalam sebuah sistem atau perangkat lunak (Pressman, 2001). Pada penelitian ini pengujian dilakukan dengan menggunakan pengujian kinerja. Efisien adalah perilaku waktu perangkat lunak yang berkaitan dengan respon, waktu pemrosesan, dan pemanfaatan sumber daya, yang mengacu kepada sumber daya material baik memori, CPU, koneksi jaringan yang digunakan oleh perangkat lunak (Ebert, 2006). Pengujian kinerja dilakukan untuk memastikan bahwa perangkat lunak yang dibuat memberikan fungsionalitas sesuai dengan kebutuhan pengguna yang telah didefinisikan (Pressman, 2001). Pengujian ini dilakukan unutuk mengukur kecepatan pengambilan data yang akan diakses dari *client* ke *server*. Analisis yang akan dilakukan mencakup analisis kecepatan data, dan analisis *line of code*.

Analisis Line of code Menurut Jones LOC merupakan ukuran yang kurang akurat dan merupakan sebuah topik yang menimbulkan perdebatan selama bertahun-tahun, dipandang sebagai sebuah ukuran untuk mengestimasi biaya dan waktu, tidak dapat dipastikan bahwa dua program yang mempunyai LOC sama akan membutuhkan waktu implementasi yang sama walaupun keduanya diimplementasikan dengan kondisi pemrograman yang standard (Pressman, 2001). Menurut (Widhyaestoeti, 2012) dalam mengukur *line of code* dapat menggunakan *time complexity*, yaitu sebuah fungsi C(N) yang diberikan untuk waktu tempuh dan atau kebutuhan *storage* dengan ukuran N input data. Pada penelitian ini menggunakan rumus $Time\ Complexity = Cn^2$ dimana *line of code* bertindak sebagai (C) Konstanta dan (n^2) adalah *Looping*/baris *coding*. Rumus *time complexity* menjelaskan bahwa *line of code* tidak mempengaruhi baris *coding* sehingga *line of code* bernilai Konstan. Meskipun metode ini kurang akurat dan merupakan metodologi yang belum diterima secara luas, tetapi metrik dengan orientasi ukuran yang merupakan kunci pengukuran dan banyak estimasi *software* yang menggunakan model ini.

BAB 3 METODOLOGI

Di dalam bab ini menerangkan tahap–tahap yang akan dilakukan dalam perancangan, implementasi, dan pengujian dari perangkat lunak yang akan dianalisa. Penyertaan kesimpulan dan saran sebagai catatan mengenai perangkat lunak yang telah dianalisis dan kemungkinan tahap untuk pengembangan selanjutnya. Diagram alir dalam pengerjaan tugas akhir ini ditunjukkan dalam **Gambar 3.1**.

Gambar 3.1 Diagram Alir Pengerjaan Tugas Akhir

3.1. Studi Literatur

Metode ini dipergunakan untuk memperoleh dasar teori sebagai referensi untuk penyusunan tugas akhir dan pengembangan aplikasi. Dasar teori dan pustaka yang berhubungan dalam penyusunan tugas akhir antara lain:

1. *Web Service*
   a. Simple Object Access Protocol (SOAP)

b. Representational State Transfer (REST)

c. Perbandingan SOAP dan REST

2. Pendekatan Pengembangan Aplikasi *Mobile*

a. *Native mobile application development*

b. *Hybrid mobile application development*

c. *Web mobile application development*

3. Perbedaan Android dan Windows Phone

a. Android

b. Windows Phone

4. Pengujian Perangkat Lunak

3.2. Perancangan Skenario Penggunaan dan Pengujian *Web Service*

Tahap perancangan skenario penggunaan dan pengujian *web services* merupakan langkah pertama dalam memulai merancang sebuah sistem yang memenuhi kebutuhan fungsional dari *web services* yang akan dianalisis pada tugas akhir ini. Skenario dimulai dengan adanya *request* dari Android dan Windows Phone *client*, *request* diterima oleh server *web service*, kemudian server mengecek apakah data yang diminta oleh *client* terdapat pada database. Server akan mengirimkan kembalian data apabila data yang diminta tersebut terdapat pada database. Perancangan skenario penggunaan dan pengujian *web services* dengan memadukan berbagai ilmu yang telah diperoleh dari dasar–dasar teori yang dijadikan referensi. Dasar–dasar teori tersebut dipadukan antara satu dengan yang lain guna merancang analisis perbandingan SOAP *web service* dengan REST *web service* dalam pengambilan data pada *platform* Android dan Windows Phone.

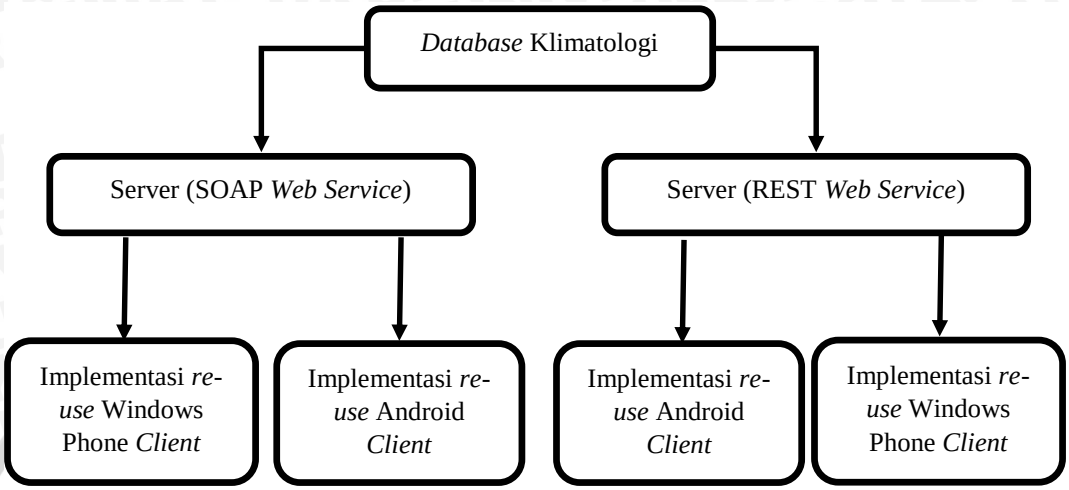
3.3. Implementasi *Re-use Client*

Berdasarkan pada skenario penggunaan dan pengujian *web service* maka implementasi dilakukan dengan me-*reuse* aplikasi *mobile* Monitoring Klimatologi BMKG Karangploso yang berjalan pada *platform* Android dan Windows Phone. Implementasi *re-use client* skenario penggunaan dan pengujian kinerja ditunjukkan dalam **Gambar 3.2**.

Pada implementasi *re-use* ini memerlukan beberapa komponen aplikasi pendukung diantaranya:

1. MySQL sebagai *database* server.
2. Pemrograman PHP untuk SOAP dan REST *web service*.
3. Format data yang digunakan menggunakan JSON.
4. Aplikasi *mobile* klimatologi BMKG menggunakan Eclipse sebagai IDE dari pemrograman Java untuk *platform* Android.

5. Aplikasi *mobile* klimatologi BMKG Microsoft visual studio 2013 sebagai IDE dari pemrograman C# untuk *platform* Windows Phone.
6. Android SDK sebagai emulator aplikasi Android dan Windows Phone SDK 8 sebagai emulator aplikasi windows phone.



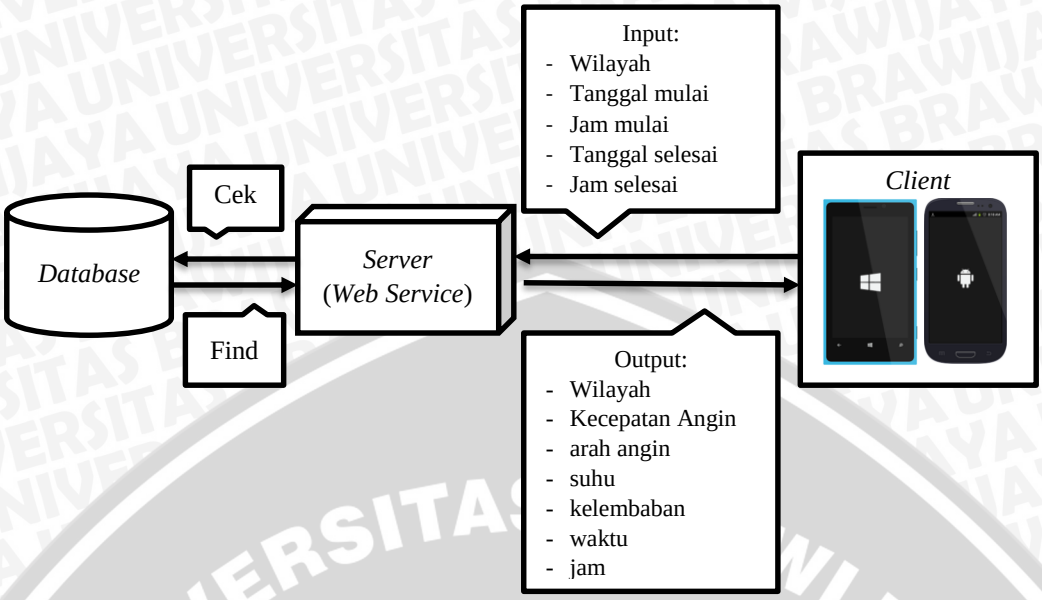
Gambar 3.2 Implementasi *Re-use Client*

Implementasi aplikasi memiliki dua bagian yaitu bagian *server* dan bagian *client*. Bagian *server* aplikasi telah dibangun dengan menggunakan bahasa pemrograman PHP untuk *web service* dan MySQL sebagai database *server*. Bagian client di-*reuse* dengan menggunakan bahasa pemrograman Java dan android SDK untuk *platform* android, dan bahasa pemrograman C# dan windows phone SDK 8 untuk *platform* Windows Phone.

3.4. Pengujian Kinerja

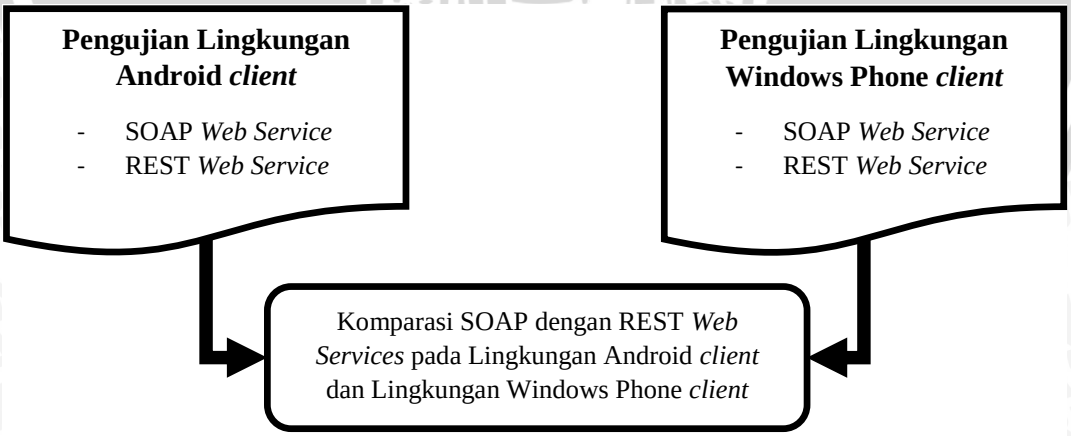
Proses pengujian dilakukan melalui kinerja pengambilan data dan komparasi kinerja pada aplikasi *mobile* monitoring klimatologi BMKG Karangploso Malang. Skema pengujian kinerja pengambilan data ditunjukkan dalam **Gambar 3.3**.

Pengujian kinerja pengambilan data dilakukan untuk mengukur kecepatan akses pengambilan data antara masukan dengan keluaran yang diharapkan. Pengujian dilakukan dengan cara menginputkan data masukan. Input data yang dimasukkan berupa memilih wilayah, tanggal mulai, jam mulai, tanggal selesai, dan jam selesai. Data yang telah diinputkan kemudian diproses oleh sisi *server* SOAP dan *server* REST (*web service*), *server* melakukan pengecekan terhadap database guna memastikan bahwa data yang diminta terdapat pada database. Data yang terdapat pada database kemudian ditransfer ke sisi *client*. Output data yang diterima dan ditampilkan berupa hasil kecepatan waktu yang diperoleh dengan menampilkan id wilayah, wilayah, kecepatan angin, arah angin, suhu, kelembaban, waktu, dan jam.



Gambar 3.3 Skema Pengujian Kinerja Pengambilan Data

Pengujian kinerja pengambilan data dilakukan pada lingkungan Android *client* dan lingkungan Windows Phone *client*. Pengujian yang dilakukan pada lingkungan Android *client* antara lain dengan menggunakan SOAP *web service*, dan dengan menggunakan REST *web service*. Pengujian yang sama pada lingkungan Android *client* juga dilakukan pada lingkungan Windows Phone *client*. Hasil dari pengujian antara SOAP dan REST *web services* pada lingkungan Android *client* kemudian akan dikomparasi dengan hasil dari pengujian antara SOAP dan REST *web services* pada lingkungan Windows Phone *client* seperti yang ditunjukkan dalam **Gambar 3.4**.



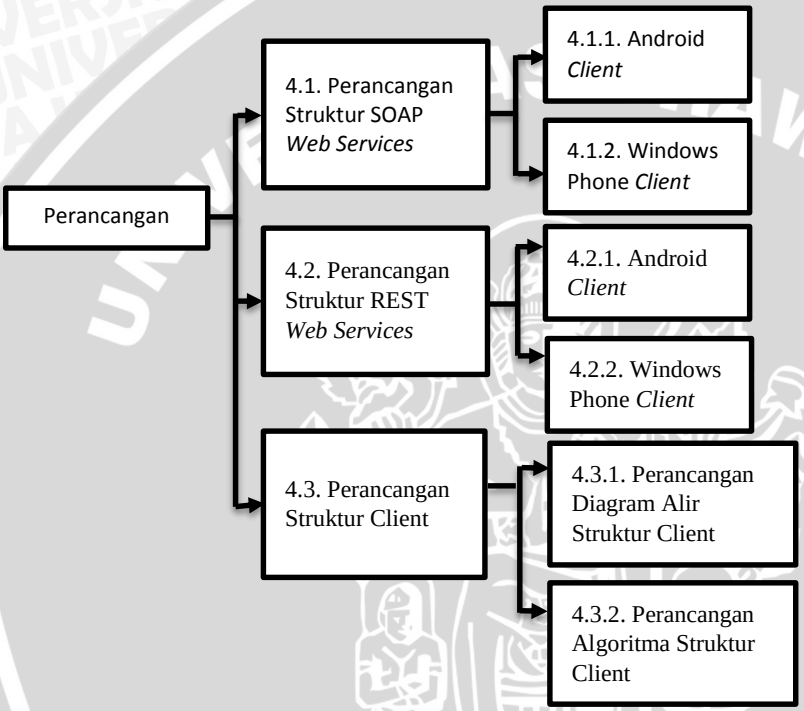
Gambar 3.4 Komparasi *Web Service* Lingkungan Android *Client* dan Windows Phone *Client*

3.5. Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian *web services* atau aplikasi yang dikembangkan selesai dilaksanakan dan didasarkan pada kesesuaian antara teori dengan penerapan. Kesimpulan dibuat untuk memberikan jawaban terhadap rumusan masalah tentang bagaimana hasil rancangan dan hasil implementasi dan hasil kinerja SOAP dan REST *web service* pada Android dan Windows Phone yang telah ditetapkan. Pembuatan saran merupakan tahapan akhir dalam penyusunan tugas akhir. Saran ditujukan untuk memperbaiki berbagai kesalahan yang terjadi selama penulisan dan memberikan pertimbangan atau arahan untuk pengembangan *web services* pada aplikasi *mobile* khusunya pada Android dan Windows Phone selanjutnya.

BAB 4 PERANCANGAN

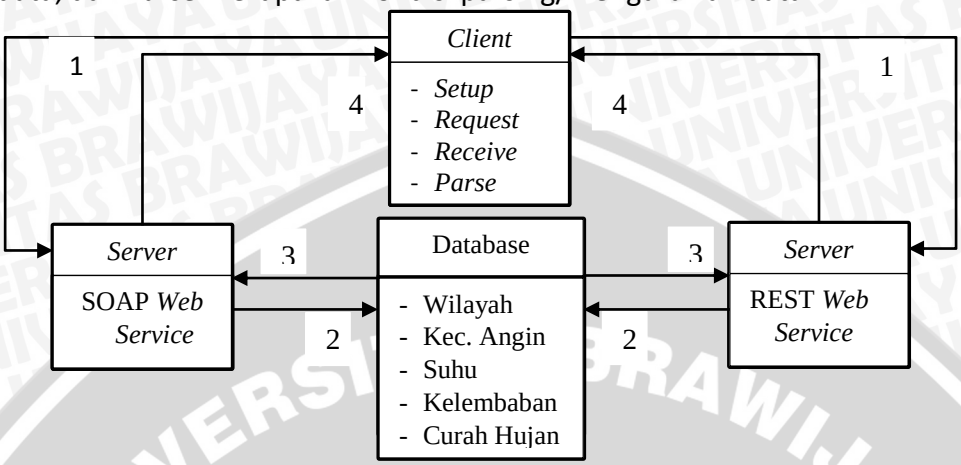
Pada bab ini akan menguraikan tentang perancangan skenario penggunaan dan pengujian *web services*. Perancangan terdiri 3 tahap. Tahap pertama merupakan perancangan struktur SOAP *web service*. Tahap kedua perancangan struktur REST *web service*. Tahap akhir yaitu tahap keempat merupakan perancangan diagram alir dan perancangan algoritme struktur *client*. Pohon perancangan dalam pengerjaan penelitian ini akan ditunjukkan dalam **Gambar 4.1**.



Gambar 4.1 Pohon Perancangan

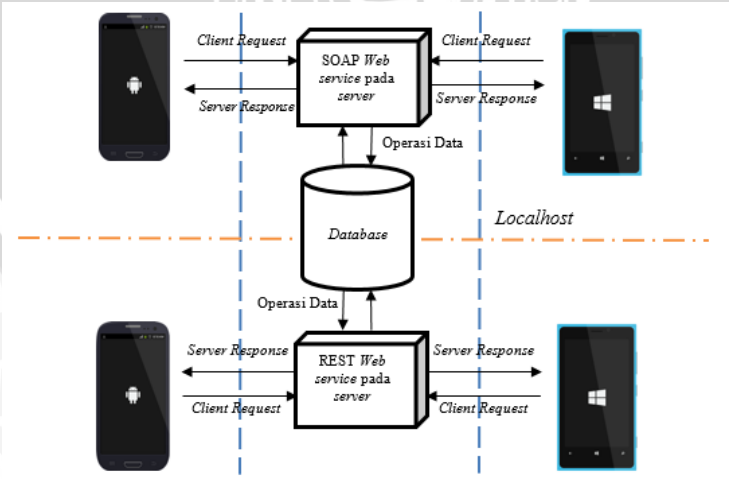
Perancangan pada bab ini merupakan *re-use* aplikasi monitoring Klimatologi BMKG Karangploso dengan menggunakan *platform* Android dan Windows Phone. Aplikasi terdahulu dibangun dengan menggunakan REST *web service* yang berjalan pada *platform* Android dan Windows Phone belum optimal. Aplikasi pada Android terdahulu menggunakan pengembangan *web mobile*, dan untuk Windows Phone menggunakan pengembangan *native*. Perancangan ini dibuat untuk mendapatkan optimalisasi kinerja pengambilan data antara SOAP dan REST *web service* yang terdapat pada Android dan Windows Phone dengan menggunakan pendekatan pengembangan *native*. Aplikasi monitoring klimatologi BMKG Karangploso dengan *platform* Android dibuat ulang dengan pengembangan *native* yang akan digunakan sebagai pembanding dalam proses pengujian. Proses perancangan dimulai dari *client*, *client* dalam perancangan ini adalah Android dan Windows Phone. Skenario untuk *client* dengan menggunakan pertukaran data SOAP *web service* dengan *client* REST *web service* sama, seperti dalam **Gambar 4.2**, yaitu:

1. *Client* melakukan *request* ke server. Struktur *request client* terhadap *web service* terdiri dari *Setup* merupakan kondisi pengesetan, *Request* merupakan kondisi untuk meminta data, *Receive* merupakan kondisi untuk menerima data, dan *Parse* merupakan kondisi parsing/menguraikan data.



Gambar 4.2 Diagram Skenario Struktur Program *Client*

2. *Request* yang dilakukan *client* kemudian diterima oleh server *web service*, pada tahap ini, server mengecek apakah data yang diminta oleh *client* terdapat pada database atau tidak. Server berasal dari *localhost*, hal ini bertujuan untuk meminimalkan paket data/data dari luar skenario dalam pengambilan data di jaringan.
3. Database melakukan pencarian terhadap data yang diminta oleh server. Data yang telah ditemukan dalam database kemudian dikirimkan ke server.
4. Server memastikan bahwa data tersedia di database. Jika data ditemukan maka data akan dikirimkan ke *client*. Jika data tidak ditemukan maka server memberitahu ke *client* bahwa data yang diminta tidak ditemukan dalam database. Hasil perancangan skenario penggunaan dan pengujian *web service* ditunjukkan dalam **Gambar 4.3**.



Gambar 4.3 Hasil Perancangan Skenario Penggunaan dan Pengujian *Web Services*

4.1. Perancangan Struktur SOAP *Web Service*

SOAP merupakan Protokol berbasis XML yang digunakan untuk pertukaran informasi antar aplikasi. SOAP dikembangkan awalnya oleh Microsoft untuk menggantikan menggantikan metode tradisional seperti Distributed Component Object Model (DCOM) dan Common Object Request Broker Architecture (CORBA) [RAM-14:3]. SOAP *web service* menggunakan format pertukaran data berupa XML. Struktur *client* web service SOAP Android dan Windows Phone yang dirancang merupakan Aplikasi Monitoring Klimatologi Karangploso.

4.1.1. Android *Client*

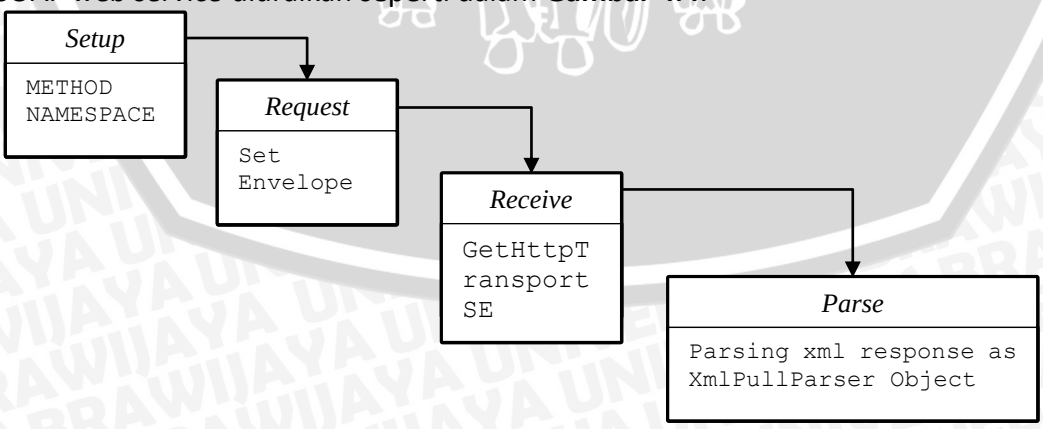
Perancangan struktur SOAP *web service* pada Android *client* dimulai dengan *Setup* yang terdiri dari METHOD dan NAMESPACE. METHOD merupakan metode *request* HTTP yang akan digunakan. NAMESPACE merupakan kumpulan *entities* (*class*, *object*, *function*) yang dikelompokkan dalam satu nama. Ada tujuh buah *method* utama yang dispesifikasikan oleh HTTP yang ditunjukkan pada **Tabel 4.1**.

Tabel 4.1 Spesifikasi *Method* utama HTTP

Method	Deskripsi
GET	Mengambil dokumen dari server
HEAD	Mengambil header dokumen dari server
POST	Mengirimkan data ke server untuk diproses
PUT	Menyimpan data yang ada di bagian Body ke server
TRACE	Mengikuti jejak pesan dari proxy server sampai ke server
DELETE	Menghapus data dari server

Sumber: (Wagh, 2012)

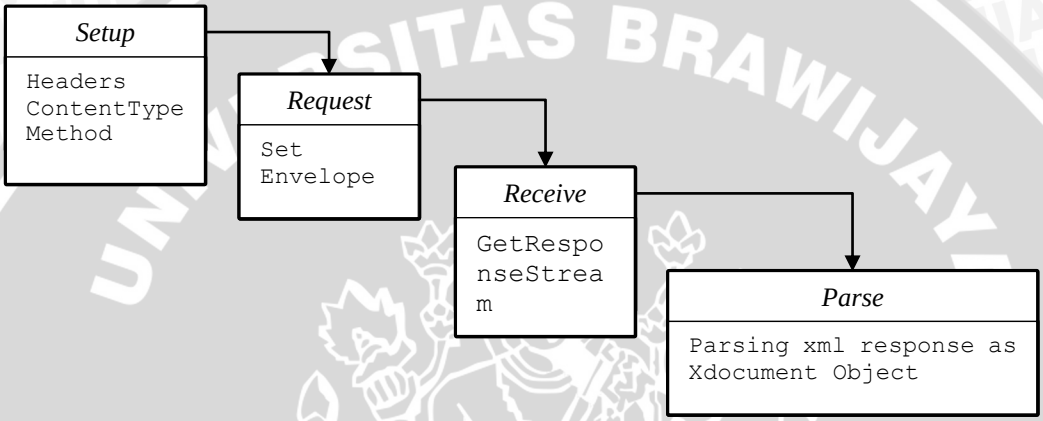
Request untuk melakukan permintaan data dengan mengeset Envelope. Receive untuk menerima data dengan menggunakan fungsi GetHttpTransportSE. Parsing data berfungsi untuk mengurai format data XML dengan menggunakan XmlPullParser Object. Diagram perancangan struktur pada Android *client* dengan SOAP *web service* diuraikan seperti dalam **Gambar 4.4**.



Gambar 4.4 Diagram Struktur Android *client web services* SOAP

4.1.2. Windows Phone *Client*

Perancangan struktur SOAP *web service* pada Windows Phone *client* dimulai dengan *Setup* yang terdiri dari Headers, ContentType dan Method. Headers mendefinisikan fungsi dimana *web service* berada. ContentType digunakan untuk menentukan jenis Multipurpose Internet Mail Extension (MIME) *response* yang dikirim ke *client*. Method mendifinisikan fungsi-fungsi yang dapat dikerjakan oleh suatu object. *Request* untuk meminta data dengan mengeset Envelope. *Receive* untuk menerima data dengan menggunakan fungsi GetResponseStream. Parsing data berfungsi untuk mengurai format data XML dengan menggunakan Xdocumet Object. Diagram perancangan struktur pada Windows Phone *client* dengan SOAP *web service* diuraikan seperti dalam **Gambar 4.5**.



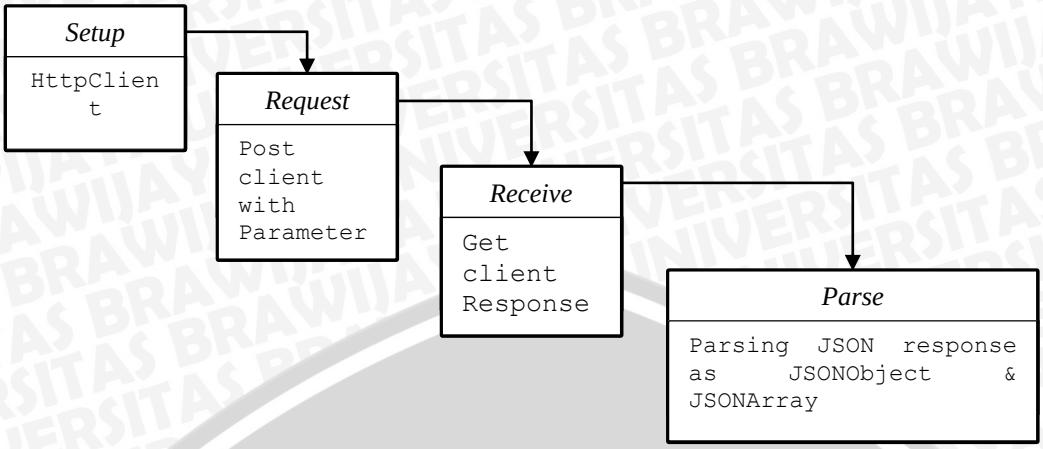
Gambar 4.5 Diagram Struktur Windows Phone client web services SOAP

4.2. Perancangan Struktur REST *Web Service*

Perancangan struktur REST *client* tidak memerlukan format pesan seperti *envelope* yang di dalamnya terdapat *header* dan *body* seperti yang dibutuhkan pada *request* perancangan SOAP *client*. Aplikasi *web* yang menerapkan prinsip REST dinamakan RESTful *web service* [MUM-13:2]. REST *web service* menggunakan format pertukaran data berupa JSON. Struktur *client web service* REST Android dan Windows Phone yang dirancang merupakan aplikasi Monitoring Klimatologi Karangploso.

4.2.1. Android *Client*

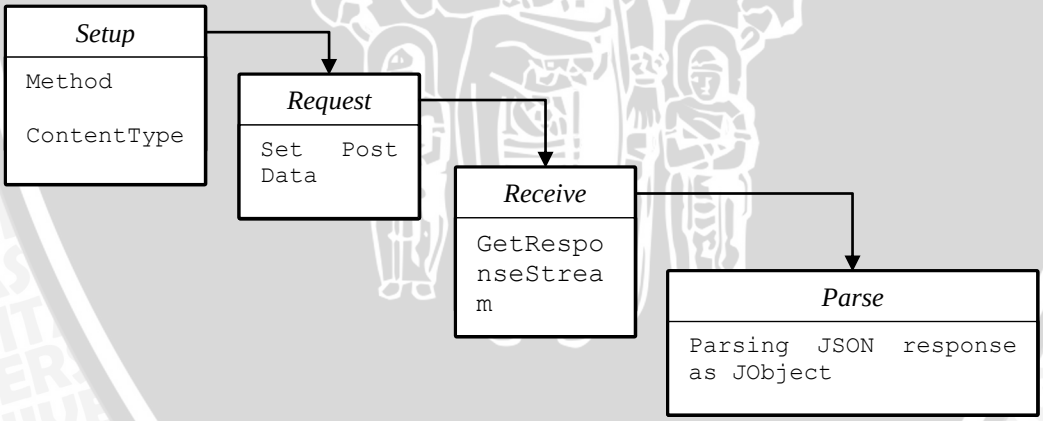
Perancangan struktur REST *web service* pada Android *client* dimulai dengan *Setup* yang terdiri dari HttpClient, HttpClient berfungsi untuk pengoperasian pengambilan data. *Request client* untuk meminta data dengan menggunakan parameter. *Receive* menerima data dengan menggunakan *response client*. Parsing data berfungsi untuk mengurai format data JSON dengan menggunakan JSONObject dan JSONArray. Diagram perancangan struktur pada Android *client* dengan REST *web service* diuraikan seperti dalam **Gambar 4.6**.



Gambar 4.6 Diagram Struktur Android *client web services* REST

4.2.2. Windows Phone *Client*

Perancangan struktur REST *web service* pada Windows Phone *client* dimulai dengan *Setup* yang terdiri dari Method dan ContentType. Method mendifinisikan fungsi-fungsi yang dapat dikerjakan oleh suatu object. ContentType digunakan untuk menentukan jenis MIME *response* yang dikirim ke *client*. *Request* untuk meminta data dengan Post Data. *Receive* untuk menerima data dengan menggunakan fungsi GetResponseStream. Parsing data berfungsi untuk mengurai format data JSON dengan menggunakan Jobject. Diagram perancangan struktur pada Windows Phone *client* dengan REST *web service* diuraikan seperti dalam **Gambar 4.7**.



Gambar 4.7 Diagram Struktur Windows Phone client web services REST

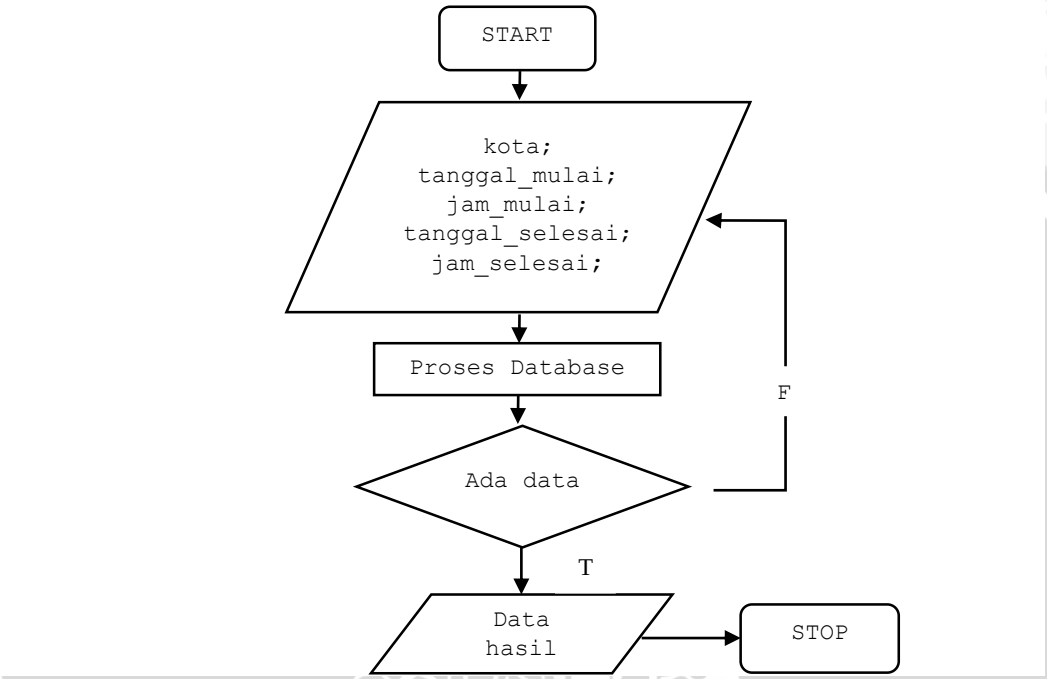
4.3. Perancangan Struktur *Client*

Perancangan struktur *client* penelitian ini difokuskan dalam fitur monitoring pada aplikasi monitoring klimatologi BMKG Karangploso. Perancangan struktur *client* diperoleh dari perancangan struktur *client web services* SOAP dan REST untuk Android dan Windows Phone yang akan diimplementasikan pada Bab 5. Perancangan struktur *client* dimulai dari proses monitoring yang dilakukan oleh

user dengan cara menentukan kategori monitoring yang akan dilakukan. Selanjutnya sistem akan menampilkan halaman monitoring sesuai dengan kategori monitoring yang dipilih. Perancangan struktur *client* dibagi menjadi dua bagian, yaitu perancangan diagram alir dan perancangan algoritme struktur *client*.

4.3.1. Perancangan Diagram Alir Struktur *Client*

Diagram alir merupakan gambaran secara visual dari algoritme–agoritme proses yang akan dirancang. Diagram alir struktur *client* diperlihatkan dalam **Gambar 4.8**.



Gambar 4.8 Diagram Alir Proses Monitoring Struktur *Client*

4.3.2. Perancangan Algoritme Struktur *Client*

Perancangan algoritme struktur *client* didapatkan dari perancangan diagram alir pada sub-bab sebelumnya. Berikut adalah algoritme struktur *client* fitur monitoring dalam **Gambar 4.9**.

```
Nama Algoritme : monitoring

Deklarasi :
• kota  -> wilayah
• date1 -> tanggal mulai
• date2 -> tanggal selesai
• time1 -> jam mulai
• time2 -> jam selesai

Deskripsi :

• Input  : kota, date1, time1, date2, time2
```

- Proses :
  a. Menampilkan halaman monitoring
  b. Mengambil data input berupa kota, tanggal mulai, jam mulai, tanggal selesai, dan jam selesai dari form.
  c. Melakukan pengecekan pada database sesuai dengan data yang telah diinputkan.
  d. Jika data terdapat pada database, maka data ditampilkan pada halaman monitoring.
  e. Jika data tidak terdapat pada database, maka 34cenario pemberitahuan pada halaman monitoring bahwa data tidak ditemukan/tidak ada data untuk ditampilkan.
- Output : data sesuai input yang tersedia di database dan ditampilkan pada halaman monitoring.

Gambar 4.9 Algoritme Proses Monitoring Struktur *Client*

BAB 5 IMPLEMENTASI

Bab implementasi ini membahas mengenai analisis proses perancangan struktur *client* yang telah dibuat pada Bab 4. Pembahasan terdiri dari penjelasan tentang spesifikasi aplikasi, dan implementasi algoritme yang ditunjukkan oleh Pohon implementasi dalam **Gambar 5.1**.

Gambar 5.1 Pohon Implementasi

5.1 Spesifikasi Aplikasi

Hasil dari perancangan struktur *client web services* yang telah diuraikan pada Bab 4 menjadi acuan dalam melakukan implementasi dan berfungsi sesuai dengan kebutuhan analisis. Spesifikasi aplikasi ditunjukkan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1. Spesifikasi Perangkat Keras

Penelitian dalam analisisnya menggunakan sebuah komputer/Laptop dengan spesifikasi perangkat keras ditunjukkan pada **Tabel 5.1**. Spesifikasi perangkat keras aplikasinya menggunakan *device* Android dan Windows Phone seperti yang terlihat pada **Tabel 5.2**.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer/Laptop

Nama Komponen	Spesifikasi
Prosesor PC	Intel (R) Core (TM) i3-2350M CPU @ 2.30Hz 2.29 GHz
Memori (RAM) PC	6.00 GB
Harddisk PC	500 GB

Tabel 5.2 Spesifikasi Perangkat Keras *Device*

Nama Komponen	Spesifikasi Perangkat Keras
Prosesor Android	Quadcore Qualcomm Snapdragon 1.2 Ghz
Memori (RAM) Android	1 GB
Haddisk Android	16 GB
Prosesor Windows Phone	Quadcore 1.2GHz
Memori (RAM) Windows Phone	512 MB
Harddisk Windows Phone	4 GB

5.1.2. Spesifikasi Perangkat Lunak

Studi perbandingan kinerja SOAP *web service* dengan RESTful *web service* dalam pertukaran data pada *platform* Android dan Windows Phone ini menggunakan perangkat lunak dengan spesifikasi seperti yang terdapat pada **Tabel 5.3**.

Tabel 5.3 Spesifikasi Perangkat Lunak Komputer/Laptop

Nama	Spesifikasi
Sistem Operasi	Microsoft Windows 8.1
Bahasa Pemrograman	Java, C#, dan PHP 5.3.8
Tools pemrograman	Eclipse, Microsoft Visual Studio, dan Sublime Text 2
Server localhost	XAMPP Server Version 1.7.7
DBMS	MySQL
Tools DBMS	MySQL Version 5.5.16

5.2. Implementasi

Studi perbandingan kinerja SOAP *web service* dengan RESTful *web service* dalam pertukaran data pada *platform* Android dan Windows Phone ini akan mengimplementasikan perancangan pada Bab 4. Perancangan tersebut diantaranya adalah perancangan struktur *client web service* SOAP dan REST pada Android dan Windows Phone.

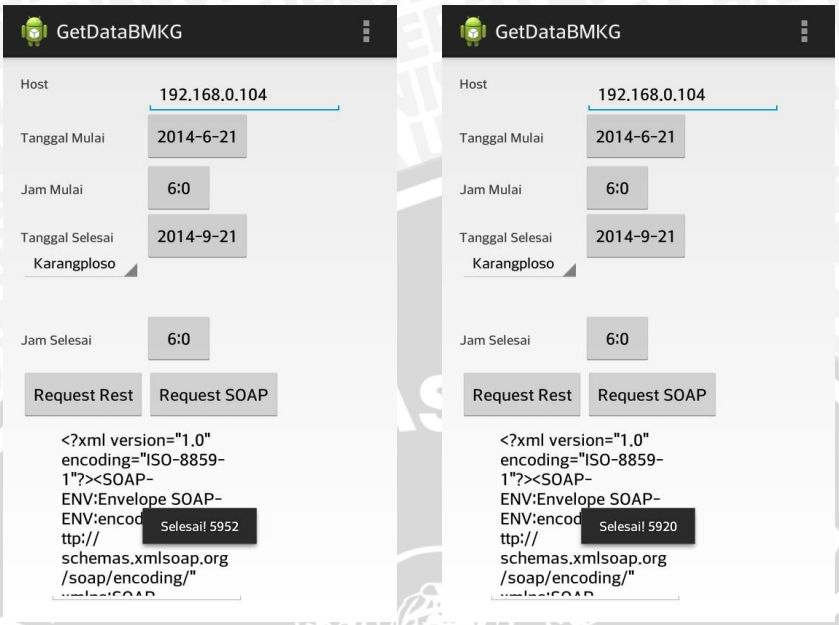
5.2.1. Implementasi Struktur *Client Web Service* SOAP

Implementasi struktur client *web service* SOAP merupakan hasil dari perancangan pada Sub Bab 4.1. Implementasi ini mencakup pada Android *client* dan pada Windows Phone *client*.

5.2.1.1. Android *Client*

Android *client* pada saat *request* proses monitoring dilakukan dengan memilih wilayah dan memasukkan data tanggal dan jam untuk melakukan monitoring. *Request* tersebut diproses oleh *server*, *server* mengecek apabila data tersedia di *database* maka data ditampilkan dan sebaliknya maka akan muncul

pemberitahuan jika data tidak tersedia. Implementasi SOAP *web service* pada Android *client* dapat dilihat dalam **Gambar 5.2**.



Gambar 5.2 Implementasi SOAP *web service* pada Android *client*

Implementasi struktur algoritme proses monitoring *web service* SOAP pada Android *client* pada Sub Bab 4.1.1 terdiri dari *Setup* ditunjukkan dalam **Kode 5.1**.

```
1.  String resultdata = "[]";
2.  String SOAP_ACTION =
"http:///"+texthost.getText()+"/nusoap/SOAP.php/searc
h";
3.      String METHOD_NAME = "search";
4.      String NAMESPACE =
"http:///"+texthost.getText()+"/nusoap/SOAP.php";
5.              String URL =
"http:///"+texthost.getText()+"/nusoap/SOAP.php";
6.              SoapObject request = new
SoapObject(NAMESPACE, METHOD_NAME);
7.             request.addProperty("kota",
selectedwilayah);
8.             request.addProperty("date1",
buttonTanggalMulai.getText());
9.             request.addProperty("time1",
buttonJamMulai.getText());
10.                request.addProperty("date2",
buttonTanggalSelesai.getText());
11.                request.addProperty("time2",
buttonJamSelesai.getText());
12.                SoapSerializationEnvelope
envelope =
13.                    new
SoapSerializationEnvelope(SoapEnvelope.VER11);
14.     envelope.dotNet = true;
```

```
15.                 envelope.setOutputSoapObject(request);
```

Kode 5.1 Implementasi Struktur Algoritme Setup dengan SOAP *web service* pada Android *client*

Struktur algoritme dari implementasi Android *client* dalam **Kode 5.1** merupakan penjelasan kondisi Setup SOAP yang terdiri dari:

Baris no. 1 : Deklarasi varibel untuk menyimpan data hasil nantinya
Baris no. 2 : Deklarasi dan inisialisasi alamat url dari method SOAP yang akan diakses
Baris no. 3 : Deklarasi dan inisialisasi nama method SOAP yang akan diakses
Baris no. 4 : Deklarasi dan inisialisasi namespace yang nantinya akan ditambahkan pada header request SOAP
Baris no. 5 : Deklarasi dan inisialisasi variabel untuk menyimpan alamat server SOAP
Baris no. 6 : Membuat object SOAP dari library yang dipakai untuk membuat struktur dalam dari SOAP Envelope
Baris no. 7-11 : Memasukkan parameter kota, date1, time1, date2, time2 yang dibutuhkan pada method SOAP yang dipanggil nantinya
Baris no. 12-15 : Membuat SOAP envelope

Implementasi struktur algoritme *request* ditunjukkan dalam **Kode 5.2**.

```
1.    HttpTransportSE androidHttpTransport = new
   HttpTransportSE(URL);
2. try {
3.      androidHttpTransport.debug = true;
4.      androidHttpTransport.call(SOAP_ACTION,
   envelope);
```

Kode 5.2 Implementasi Struktur Algoritme *Request* dengan SOAP *web service* pada Android *client*

Struktur algoritme dari implementasi Android *client* dalam **Kode 5.2** merupakan kondisi *Request* SOAP yang terdiri dari:

Baris no. 1 : Deklarasi dan inisialisasi HttpTransportSE sebagai sarana untuk mengirim request SOAP pada server
Baris no. 3-4 : Mengatur kemudian melakukan request pada alamat yang dituju dengan menambahkan isi dari SOAP_ACTION dan envelope yang telah kita buat pada header request.

Implementasi struktur algoritme *Receive* ditunjukkan dalam **Kode 5.3**.

```
1. resultdata = androidHttpTransport.responseDump;
2.                 } catch (HttpResponseException e) {
3.                     // TODO Auto-generated catch
   block
4.                     Log.d("dwihardy",
   "HttpResponseException");
5.                     e.printStackTrace();
```

```
6.                } catch (IOException e) {
7.                    // TODO Auto-generated catch
   block
8.                    Log.d("dwihardy",
   "IOException");
9.                    e.printStackTrace();
10.                   } catch (XmlPullParserException
   e) {
11.                       // TODO Auto-generated
   catch block
12.                       Log.d("dwihardy",
   "XmlPullParserException");
13.                       e.printStackTrace();
14.                   } catch (Exception e)
15.                {
16.                       Log.d("dwihardy",
   "Exception : "+e.toString());
17.                   }
18.                   return resultdata;
19.           }
```

Kode 5.3 Implementasi Struktur Algoritme *Receive* dengan SOAP *web service* pada Android *client*

Struktur algoritme dari implementasi Android *client* dalam **Kode 5.3** merupakan kondisi *Receive* SOAP yang terdiri dari:

Baris no. 1 : Menyimpan hasil kembalian dari *request* yang kita lakukan
Baris no. 2-17 : Menampilkan pada *log* jika terjadi kesalahan
Baris no. 18 : Mengembalikan data hasil yang telah disimpan

Implementasi struktur algoritme *Parse* ditunjukkan dalam **Kode 5.4**.

```
1. protected void onPostExecute(String result) {
2.    try {
3.                   XmlPullParserFactory xmlfactory
   = XmlPullParserFactory.newInstance();
4.                   XmlPullParser xmlparser =
   xmlfactory.newPullParser();
5.       xmlparser.setFeature(XmlPullParser.FEATURE_PROC
   ESS_NAMESPACES, false);
6.                   InputStream is = new
   ByteArrayInputStream(result.getBytes());
7.                   xmlparser.setInput(is, null);
8.                   String text = null;
9.                   int event =
   xmlparser.getEventType();
10.                  while (event !=
   XmlPullParser.END_DOCUMENT)
11.                  {
12.                        String name =
   xmlparser.getName();
13.                        switch(event)
14.                        {
```

```
15.                                         case
   XmlPullParser.START_TAG:
16.                                             break;
17.                                         case
   XmlPullParser.TEXT:
18.                                             text =
   xmlparser.getText();
19.                                             break;
20.                                         case
   XmlPullParser.END_TAG:
21.
        if(name.equals("ID"))
22.                                             {
23.             Log.d("Text ID", text);
24.                                             }
25.                 Else
26.      if(name.equals("KECEPATAN_ANGIN"))
27.                                             {
28.
       Log.d("Text Kecepatan Angin", text);
29.                                             }
30.                                             break;
31.                                     }
32.                                     event =
   xmlparser.next();
33.                                 }
34.                             } catch (XmlPullParserException
   e) {
35.                                 // TODO Auto-generated
   catch block
36.                                 e.printStackTrace();
37.                             } catch (IOException e) {
38.                                 // TODO Auto-generated
   catch block
39.                                 e.printStackTrace();
40.                             }
```

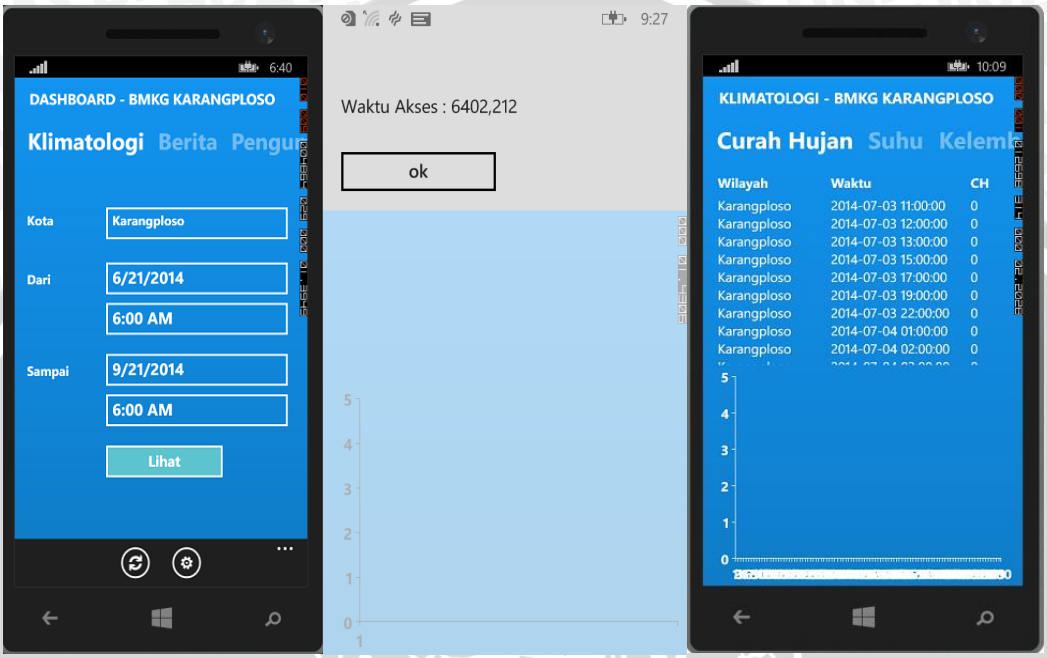
Kode 5.4 Implementasi Struktur Algoritme *Parse* dengan SOAP *web service* pada Android *client*

Struktur algoritme dari implementasi Android *client* dalam **Kode 5.4** merupakan kondisi *Parse* SOAP yang terdiri dari:

Baris no. 3 : Membuat factory yang merupakan implementasi dari KXmlParser dan KXmlSerializer
Baris no. 4-5 : Membuat implementasi dari XML Pull Parser berdasarkan konfigurasi yang telah dibuat pada factory
Baris no. 6 : Mengkonversi hasil kembalian dari request menjadi InputStream
Baris no. 7-33 : Mengeset input yang akan diolah pada xmlparser, lalu mulai mengolah data tersebut
Baris no. 34-40 : Melakukan log jika terjadi kesalahan

5.2.1.2. Windows Phone *Client*

Windows Phone *client* pada *request* proses monitoring dilakukan dengan memilih wilayah dan memasukkan data tanggal dan jam untuk melakukan monitoring. *Request* tersebut diproses oleh *server*, *server* mengecek apabila data tersedia di *database* maka data ditampilkan dan sebaliknya maka akan muncul pemberitahuan jika data tidak tersedia. Implementasi struktur SOAP *web service* pada Windows Phone *client* keseluruhan dapat dilihat dalam **Gambar 5.3**.



Gambar 5.3 Implementasi Struktur SOAP *web service* pada Windows Phone *client*

Implementasi struktur algoritme proses monitoring *web service* SOAP pada Windows Phone *client* pada Sub Bab 4.1.2 terdiri dari *Setup* ditunjukkan dalam **Kode 5.5**.

```
1.   HttpWebRequest spAuthReq =
HttpWebRequest.Create("http://localhost/nusoap/SOAP.
php") as HttpWebRequest;
2.              spAuthReq.Headers["SOAPAction"] =
"http://localhost/nusoap/SOAP.php/search";
3.              spAuthReq.ContentType = "text/xml;
charset=utf-8";
4.              spAuthReq.Method = "POST";
5.              waktuakses = DateTime.Now;
6.              spAuthReq.BeginGetRequestStream(new
AsyncCallback(DoRequestData), spAuthReq);
7.          }
```

Kode 5.5 Implementasi Struktur Algoritme Setup dengan SOAP web service pada Windows Phone *client*

Struktur algoritme dari implementasi Windows Phone *client* dalam **Kode 5.5** merupakan penjelasan kondisi *Setup* SOAP yang terdiri dari:

Baris no. 1 : Membuat object HttpWebRequest sebagai media untuk melakukan transport request
Baris no. 2 : Menambahkan header pada request
Baris no. 3 : Mengeset tipe konten yang akan dikirim untuk request
Baris no. 4 : Mengeset metode yang dipakai untuk request
Baris no. 5 : Mengeset variabel sebagai penanda yang nantinya digunakan untuk mendapatkan jumlah waktu yang dibutuhkan untuk melakukan sebuah request
Baris no. 6 : Mulai melakukan request

Implementasi struktur algoritme *Request* ditunjukkan dalam **Kode 5.6**.

```
1. private void DoRequestData(IasyncResult asyncResult)
2.    {
3.     string AuthEnvelope =
4.      @"<?xml version=""1.0"" encoding=""utf-8""?>
5.            <soap:Envelope
   xmlns:xsi=""http://www.w3.org/2001/XMLSchema-
   instance""
   xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
   xmlns:soap=""http://schemas.xmlsoap.org/soap/envelop
   e/"">
6.                 <soap:Body>
7.                    <search
   soapenv:encodingStyle=""http://schemas.xmlsoap.org/s
   oap/encoding/"">
8.                       <kota
   xsi:type=""xsd:string"">"+this._kota+ @"</kota>
9.                       <date1
   xsi:type=""xsd:string"">" + this._tanggalDari +
   @"</date1>
10.                         <time1
   xsi:type=""xsd:string"">" + this._timeDari +
   @"</time1>
11.                         <date2
   xsi:type=""xsd:string"">" + this._tanggalSampai +
   @"</date2>
12.                         <time2
   xsi:type=""xsd:string"">" + this._timeSampai +
   @"</time2>
13.                    </search>
14.                </soap:Body>
15.             </soap:Envelope>";
16.             UTF8Encoding encoding = new
   UTF8Encoding();
17.             HttpWebRequest request =
   (HttpWebRequest)asyncResult.AsyncState;
18.       System.Diagnostics.Debug.WriteLine("Request
   is :" + request.Headers);
```

```
19.         Stream _body =
    request.EndGetRequestStream(asyncResult);
20.         string envelope =
    string.Format(AuthEnvelope);
21.         System.Diagnostics.Debug.WriteLine("Envelope
    is :" + envelope);
22.                 byte[] formBytes =
    encoding.GetBytes(envelope);
23.                 _body.Write(formBytes, 0,
    formBytes.Length);
24.                 _body.Close();
25.                 request.BeginGetResponse(new
    AsyncCallback(DoRetrieveData), request);
26.             }
```

Kode 5.6 Implementasi Struktur Algoritme *Request* dengan SOAP *web service* pada Windows Phone *client*

Struktur algoritme dari implementasi Windows Phone *client* dalam **Kode 5.6** merupakan kondisi *Request* SOAP yang terdiri dari:

Baris no. 3-15 : Mendefinisikan SOAP Envelope dengan mencantumkan komponen-komponen dan parameter yang dibutuhkan untuk melakukan sebuah SOAP request
Baris no. 16 : Mendefinisikan encoding yang nantinya akan digunakan
Baris no. 17-25 : Menambahkan SOAP Envelope yang telah dibuat kedalam stream request yang sedang berjalan

Implementasi struktur algoritme *Receive* ditunjukkan dalam **Kode 5.7**.

```
1. HttpWebRequest request =
   (HttpWebRequest)asyncResult.AsyncState;
2.                   HttpWebResponse response =
   (HttpWebResponse)request.EndGetResponse(asyncResult)
   ;
3.   if (request != null && response != null)
4.       {
5.       if (response.StatusCode == HttpStatusCode.OK)
6.           {
7. StreamReader reader = new
   StreamReader(response.GetResponseStream());
```

Kode 5.7 Implementasi Struktur Algoritme *Receive* dengan SOAP *web service* pada Windows Phone *client*

Struktur algoritme dari implementasi Windows Phone *client* dalam **Kode 5.7** merupakan kondisi *Receive* SOAP yang terdiri dari:

Baris no. 1 : Mendapatkan status dari HttpWebRequest yang sedang dijalankan
Baris no. 2 : Mengambil hasil dari HttpWebRequest
Baris no. 3-6 : Pengecekan kevalidan isi respon yang diperoleh

Baris no. 7 : Mengambil hasil request sebagai object StreamReader

Implementasi struktur algoritme *Parse* ditunjukkan dalam **Kode 5.8**.

```
1. StreamReader reader = new
   StreamReader(response.GetResponseStream());
2.                              string Notificationdata
   = RemoveAllNamespaces(reader.ReadToEnd());
3.                              Xdocument doc =
   Xdocument.Parse(Notificationdata);
```

Kode 5.8 Implementasi Struktur Algoritme *Parse* dengan SOAP *web service* pada Windows Phone *client*

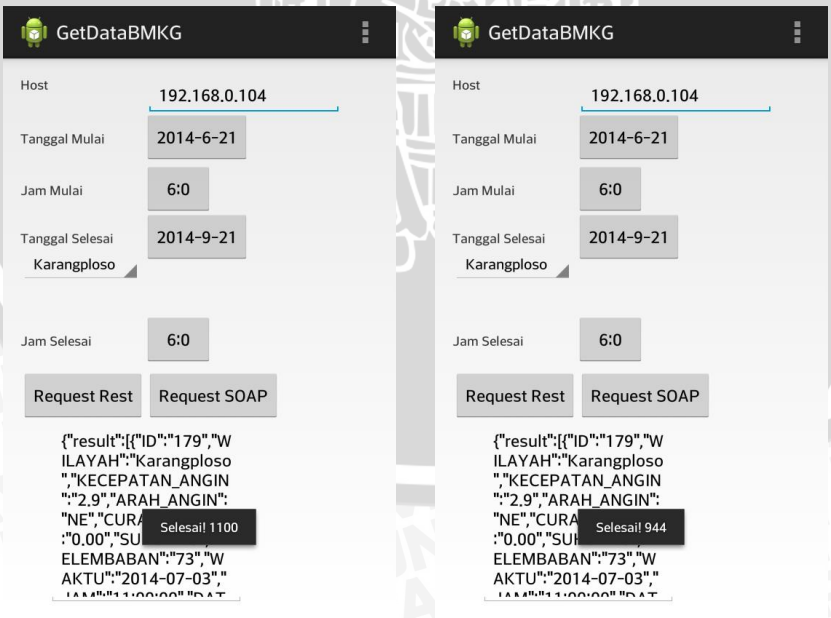
Struktur algoritme dari implementasi Windows Phone *client* dalam **Kode 5.8** merupakan kondisi *Parse* SOAP yang terdiri dari:

Baris no. 1 : Mengambil hasil request sebagai object StreamReader
Baris no.2 : Menghapus semua komponen selain body dari hasil kembalian xml SOAP
Baris no. 3 : Melakukan parsing data menjadi object XDoument untuk pembacaan XML nantinya

5.2.2. Implementasi Struktur *Client Web Service* REST

Implementasi struktur *client web service* REST merupakan hasil dari perancangan pada Sub Bab 4.2. Implementasi ini mencakup pada Android *client* dan pada Windows Phone *client*.

5.2.2.1. Android *Client*



Gambar 5.4 Implementasi REST *web service* pada Android *client*

Android *client* pada saat *request* proses monitoring dilakukan dengan memilih wilayah dan memasukkan data tanggal dan jam untuk melakukan monitoring. *Request* tersebut diproses oleh *server*, *server* mengecek apabila data tersedia di *database* maka data ditampilkan dan sebaliknya maka akan muncul pemberitahuan jika data tidak tersedia. Implementasi struktur REST *web service* pada Android *client* keseluruhan dapat dilihat dalam **Gambar 5.12**.

Implementasi struktur algoritme proses monitoring *web service* REST pada Android *client* pada Sub Bab 4.2.1 terdiri dari *Setup* ditunjukkan dalam **Kode 5.13**.

```
1. AsyncHttpClient client = new AsyncHttpClient();
2. RequestParams params = new RequestParams();
3.            params.put("date1",
   buttonTanggalMulai.getText());
4.                        params.put("date2",
   buttonTanggalSelesai.getText());
5.                        params.put("time1",
   buttonJamMulai.getText());
6.                        params.put("time2",
   buttonJamSelesai.getText());
7.                        params.put("kota",
   selectedwilayah);
8.      startingtime = System.currentTimeMillis();
9. Toast.makeText(getApplicationContext(), "Request
   mulai, silahkan tunggu! ",
   Toast.LENGTH_LONG).show();
```

Kode 5.9 Implementasi Struktur Algoritme *Setup* dengan REST *web service* pada Android *client*

Struktur algoritme dari implementasi Android *client* dalam **Kode 5.9** merupakan penjelasan kondisi *Setup* REST yang terdiri dari:

Baris no. 1 : Membuat object AsyncHttpClient sebagai media untuk melakukan *request*.
Baris no. 2-7 : Membuat object parameter untuk menambahkan parameter date1, time1, date2, time2, dan kota untuk ditambahkan saat melakukan *request*.

Implementasi struktur algoritme *request* ditunjukkan dalam **Kode 5.10**.

```
1. client.post("http://"+texthost.getText()+"/servicejs
   on/n-klimatologi.php", params, new
   AsyncHttpResponseHandler(){
```

Kode 5.10 Implementasi Struktur Algoritme *Request* dengan REST *web service* pada Android *client*

Struktur algoritme dari implementasi Android *client* dalam **Kode 5.10** merupakan kondisi *Request* REST yang terdiri dari:

Baris no. 1 : Mulai melakukan request dengan menggunakan metode POST ke server dengan parameter yang telah dibuat

Implementasi struktur algoritme *Receive* ditunjukkan dalam **Kode 5.11**.

```
1. public void onSuccess(String response)
2.       {
3.       Toast.makeText(getApplicationContext(),
   "Selesai! "+(System.currentTimeMillis()-
   startingtime), Toast.LENGTH_LONG).show();
4.       String result = "{\"result\":"+response+"}";

5. textHasil.setText(result);
```

Kode 5.11 Implementasi Struktur Algoritme *Receive* dengan REST *web service* pada Android *client*

Struktur algoritme dari implementasi Android *client* dalam **Kode 5.11** merupakan kondisi *Receive* REST yang terdiri dari:

Baris no. 3 : Menampilkan waktu tempuh yang dibutuhkan
Baris no. 4 : Menyimpan hasil request
Baris no. 5 : Menampilkan hasil request ke text area

Implementasi struktur algoritme *Parse* ditunjukkan dalam **Kode 5.12**.

```
1. //parse rest
2. JSONObject jsonobj = new JSONObject(result);
3. JSONArray resultjson =
   jsonobj.getJSONArray("result");
4. for(int i=0; i<resultjson.length(); i++)
5.   {
6.     JSONObject klimatologi =
   resultjson.getJSONObject(i);
7.     Log.d("Hasil ID", klimatologi.getString("ID"));
8.                 }
9.             } catch (JSONException e) {
10.          Log.d("gagal parsing json", "gagal karena
   "+e.getMessage());
11.      // TODO Auto-generated catch block
12.          e.printStackTrace();
13.      }
```

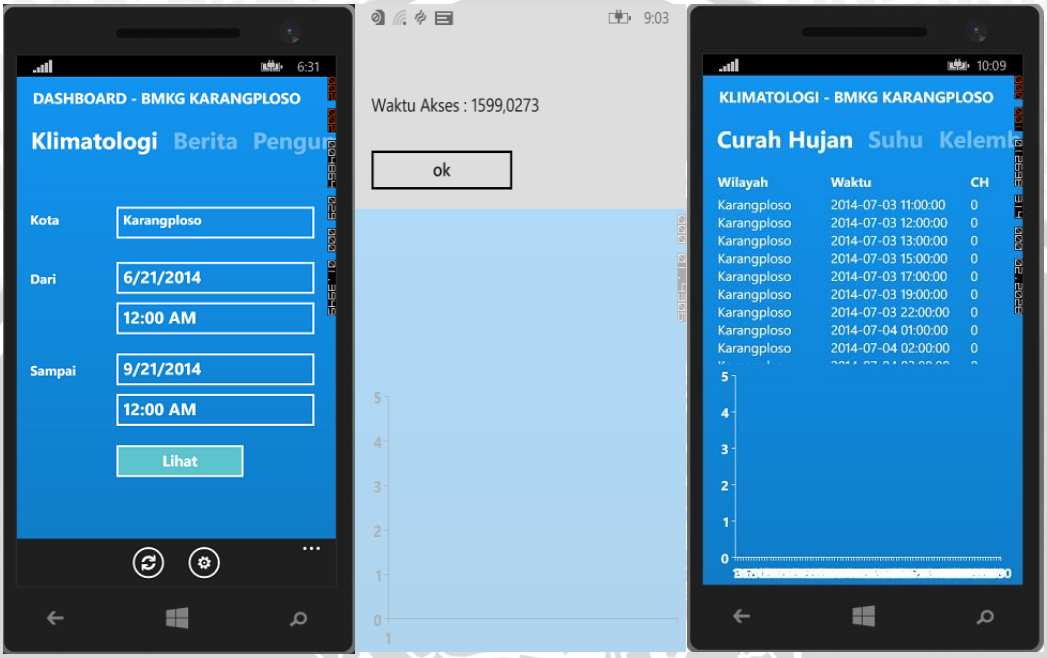
Kode 5.12 Implementasi Struktur Algoritme *Parse* dengan REST *web service* pada Android *client*

Struktur algoritme dari implementasi Android *client* dalam **Kode 5.12** merupakan kondisi *Parse* REST yang terdiri dari:

Baris no. 2 : Membuat object JSON dari hasil yang didapatkan
Baris no. 3 : Mengambil data berupa array dari tag 'result' pada object JSON
Baris no. 4-13 : Membaca data-data yang diperlukan, dan menampilkan log jika terjadi kesalahan

5.2.2.2. Windows Phone *Client*

Windows Phone *client* pada *request* proses monitoring dilakukan dengan memilih wilayah dan memasukkan data tanggal dan jam untuk melakukan monitoring. *Request* tersebut diproses oleh *server*, *server* mengecek apabila data tersedia di *database* maka data ditampilkan dan sebaliknya maka akan muncul pemberitahuan jika data tidak tersedia. Implementasi struktur REST *web service* pada Windows Phone *client* keseluruhan dapat dilihat dalam **Gambar 5.5**.



Gambar 5.5 Implementasi Struktur REST *web service* pada Windows Phone *client*

Implementasi struktur algoritme proses monitoring *web service* REST pada Windows Phone *client* pada Sub Bab 4.2.2 terdiri dari *Setup* ditunjukkan dalam **Kode 5.13**.

```
1. Uri uri = new Uri(ViewModel.Instance.base_url + "n-
   klimatologi.php");
2. HttpWebRequest request = HttpWebRequest.Create(uri)
   as HttpWebRequest;
3. request.Method = "POST";
4. request.ContentType = "application/x-www-form-
   urlencoded";
        waktuakses = DateTime.Now;
```

Kode 5.13 Implementasi Struktur Algoritme *Setup* dengan REST *web service* pada Windows Phone *client*

Struktur algoritme dari implementasi Windows Phone *client* dalam **Kode 5.13** merupakan penjelasan kondisi *Setup* REST yang terdiri dari:

Baris no. 1 : Membuat URI dari URL yang akan dituju

Baris no. 2 : Membuat object HttpWebRequest sebagai media untuk melakukan request
Baris no. 3 : Mengeset metode request ke POST
Baris no. 4 : Mengeset tipe konten yang akan digunakan

Implementasi struktur algoritme *Request* ditunjukkan dalam **Kode 5.14**.

```
1. request.BeginGetRequestStream(new
   AsyncCallback(GetRequestStreamKlimatologiCallback),
   request);
```

Kode 5.14 Implementasi Struktur Algoritme *Request* dengan REST *web service* pada Windows Phone *client*

Struktur algoritme dari implementasi Windows Phone *client* dalam **Kode 5.14** merupakan kondisi *Request* REST yang terdiri dari:

Baris no. 1 : Mulai melakukan request

Implementasi struktur algoritme *Receive* ditunjukkan dalam **Kode 5.15**.

```
1. void
   GetResponseStreamKlimatologiCallback(IasyncResult
   callbackResult)
2.         {
3.             Dispatcher.BeginInvoke(() => {
4.                 MessageBox.Show("Waktu Akses : " +
   (DateTime.Now.Subtract(waktuakses).TotalMilliseconds
   ));
5.             });
6.             RemoveDataKlimatologi();
7.             try
8.             {
9.                 HttpWebRequest request =
   callbackResult.AsyncState as HttpWebRequest;
10.                HttpWebResponse response =
   request.EndGetResponse(callbackResult) as
   HttpWebResponse;
11.                using (StreamReader
   httpWebStreamReader = new
   StreamReader(response.GetResponseStream()))
12.                {
```

Kode 5.15 Implementasi Struktur Algoritme *Receive* dengan REST *web service* pada Windows Phone *client*

Struktur algoritme dari implementasi Windows Phone *client* dalam **Kode 5.15** merupakan kondisi *Receive* REST yang terdiri dari:

Baris no. 3-5 : Menampilkan waktu akses yang dilakukan
Baris no. 6 : Menghapus semua data yang telah tersimpan dalam device sekarang
Baris no. 9 : Mendapatkan status dari request yang dilakukan

Baris no. 10 : Mengambil hasil respon dari request yang dilakukan
Baris no. 11 : Mengkonversi respon yang didapatkan menjadi bentuk StreamReader

Implementasi struktur algoritme *Parse* ditunjukkan dalam **Kode 5.16**.

```
1. string result = httpWebStreamReader.ReadToEnd();
2.                     string responseString =
   "{\"data\":" + result + "}";
3.                     Jobject jsonData =
   Jobject.Parse(responseString);
4.                     List<Jtoken> klimatologis =
   jsonData.SelectTokens("data[*]").ToList();
5.                     if (klimatologis.Count == 0)
6.                     {
7.                         DataKlimatologi.Add(new
   Mcuaca() { Wilayah = "Data not found..", ArahAngin =
   " -", KecepatanAngin = 0, CurahHujan = 0, Suhu = 0,
   Kelembaban = 0 });
8.                     }
9. else
10.                        {
11.
   ObservableCollection<float> curahhujan = new
   ObservableCollection<float>();
12.
   ObservableCollection<float> suhu = new
   ObservableCollection<float>();
13.
   ObservableCollection<float> kelembaban = new
   ObservableCollection<float>();
14.
   ObservableCollection<float> kecepatanAngin = new
   ObservableCollection<float>();
15.                            int i = 0;
16.                            foreach (var data in
   klimatologis)
17.                            {
18.
   Dispatcher.BeginInvoke(new Action(delegate
19.                                {
20.
   curahhujan.Add(data.Value<float>("CURAH_HUJAN"));
21.
   suhu.Add(data.Value<float>("SUHU"));
22.
   kelembaban.Add(data.Value<float>("KELEMBABAN"));
23.
   kecepatanAngin.Add(data.Value<float>("KECEPATAN_ANGI
   N"));
24.
   DataKlimatologi.Add(new MCuaca()
25.                                {
```

```
26.                                                 ID =
   data.Value<int>("ID"),
27.                                                 Wilayah =
   data.Value<string>("WILAYAH"),
28.
   KecepatanAngin =
   data.Value<float>("KECEPATAN_ANGIN"),
29.                                                 ArahAngin =
   ": " + data.Value<string>("ARAH_ANGIN"),
30.                                                 CurahHujan
   = data.Value<float>("CURAH_HUJAN"),
31.                                                 Suhu =
   data.Value<float>("SUHU"),
32.                                                 Kelembaban
   = data.Value<float>("KELEMBABAN"),
33.                                                 Waktu =
   data.Value<string>("WAKTU"),
34.                                                 Jam =
   data.Value<string>("JAM"),
35.                                                 DateTime =
   data.Value<string>("DATE_TIME")
36.                                             });
```

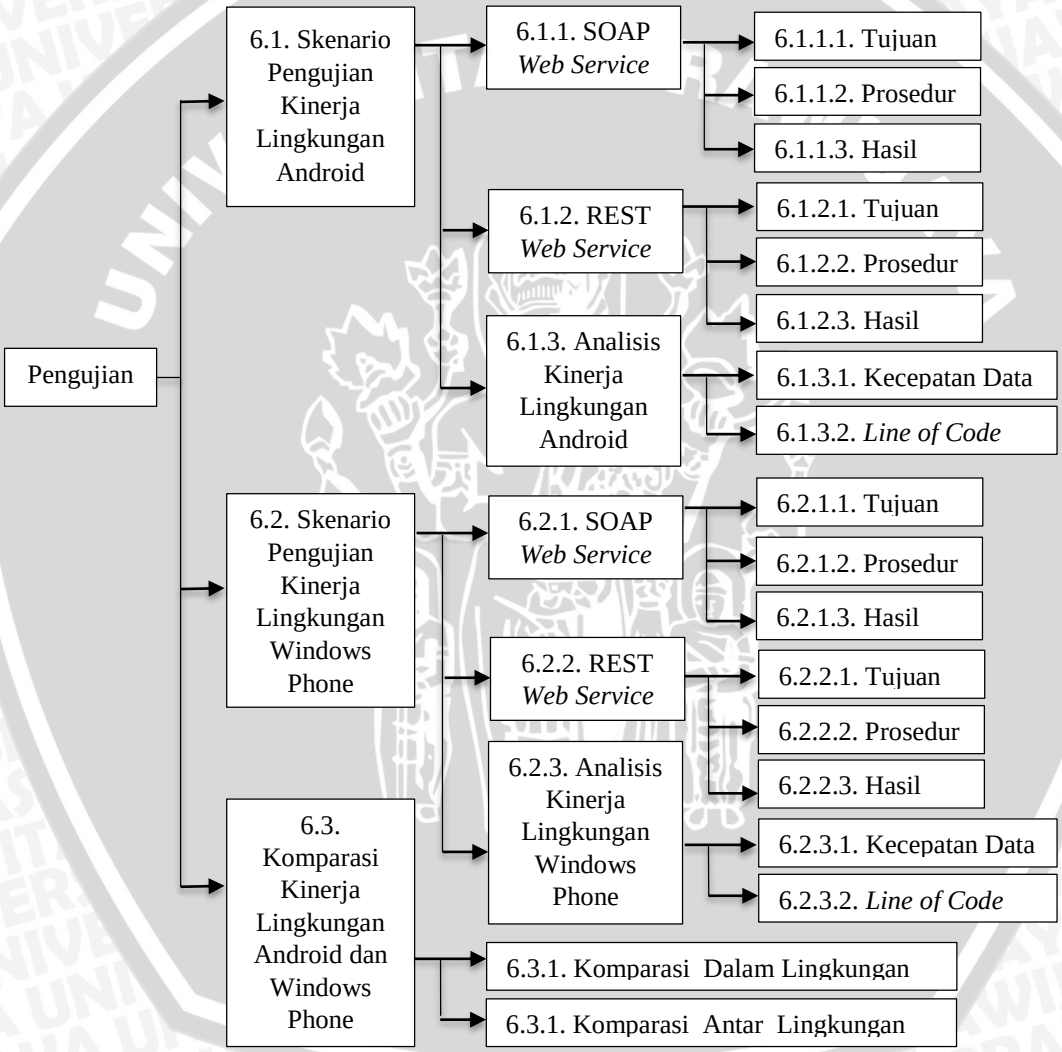
Kode 5.16 Implementasi Struktur Algoritme *Parse* dengan REST *web service* pada Windows Phone *client*

Struktur algoritme dari implementasi Windows Phone *client* dalam **Kode 5.16** merupakan kondisi *Parse* REST yang terdiri dari:

Baris no. 1 : Mengambil semua karakter yang ada pada StreamReader yang telah dibuat sebagai sebuah data string
Baris no. 2 : Menambahkan beberapa atribut tambahan agar dapat dibaca oleh library JSON
Baris no. 3 : Melakukan parsing data hasil sebagai JObject
Baris no. 4 : Mengambil data berupa array dari tag json 'data'

BAB 6 PENGUJIAN & ANALISIS

Bab pengujian dan analisis akan membahas mengenai prosedur pengujian *web services*. Pengujian dilakukan melalui pengujian kinerja pengambilan data. Pengujian kinerja pertukaran data digunakan untuk menganalisis kecepatan akses pengambilan data dari SOAP *web service* dan REST *web service* pada Android *client* dan Windows Phone *client* sesuai dengan tujuan penelitian. Diagram alir skenario pengujian dan analisis dapat dilihat dalam **Gambar 6.1**.



Gambar 6.1 Diagram Alir Skenario Pengujian dan Analisis

6.1. Pengujian Kinerja Lingkungan Android

Pada sub bab ini mengacu dalam **Gambar 4.3** yang menjelaskan mengenai skenario pengujian kinerja pengambilan data dalam lingkungan Android. Pengujian kinerja merupakan pengujian yang dilakukan untuk mengukur kecepatan pengambilan data yang diakses dari *client* Android ke *server*. *Server* dalam penelitian ini yaitu *server localhost* bertujuan untuk meminimalkan paket data/data dari luar skenario dalam pengambilan data di jaringan. Pendekatan pengembangan menggunakan *native development* juga digunakan untuk mendapatkan kinerja aplikasi yang optimal. Pengujian kinerja tidak menekankan pada jalannya algoritme sistem. Namun lebih kepada mengamati kecepatan pertukaran data dalam pertukaran data dari *client* ke *server*.

6.1.1. Skenario SOAP *web service*

Pada pengujian kinerja lingkungan Android ini akan dilakukan pengamatan cara kerja dari kecepatan pertukaran data terhadap pengambilan data pada aplikasi monitoring klimatologi BMKG Karangploso dengan menggunakan SOAP *web service*. Pengujian dilakukan di device oleh karena itu dibutuhkan kondisi normal dan peralatan yang baik untuk jalannya pengujian. Berikut akan dijelaskan mengenai tujuan, prosedur, dan hasil analisis dari pengujian lingkungan Android dengan SOAP *web service*.

6.1.1.1. Tujuan

Tujuan pengujian kinerja pada lingkungan Android dengan SOAP *web service* adalah untuk mengetahui apakah sistem dapat berjalan dan memberikan hasil optimal pada saat melakukan pertukaran data *client-server*. Kesuksesan pengujian ditunjukkan dengan diperolehnya nilai kecepatan akses pengambilan data yang optimal.

6.1.1.2. Prosedur

Prosedur pada pengujian lingkungan Android dengan SOAP *web service* dilakukan dengan menjalankan sistem pada fitur monitoring klimatologi BMKG Karangploso. Pengujian dilakukan dengan percobaaan pengambilan data dalam kurun waktu 1 bulan, 2 bulan, dan 3 bulan sebanyak 20 kali dengan meng-*input*kan data wilayah, tanggal mulai, jam mulai, tanggal selesai, dan jam selesai. Percobaan dilakukan sebanyak 20 kali dikarenakan untuk mendapatkan hasil kecepatan akses yang bervariasi. Kasus uji percobaan kecepatan akses pengambilan data pada pengujian lingkungan Android dengan SOAP *web service* sebagai berikut:

1. **Kasus uji kurun waktu 1 bulan**

Kasus uji kurun waktu 1 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 Juli 2014 di lingkungan Android

dengan SOAP *web service* yang ditunjukkan pada **Tabel 6.1**. Detil kasus uji kurun waktu 1 bulan SOAP *web service* lingkungan Android terdapat pada **Lampiran 1**.

Tabel 6.1 Kecepatan Akses Pengambilan Data kurun waktu 1 bulan Lingkungan Android dengan SOAP *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2316
2.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	4429
3.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2009
4.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1899
5.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1687
6.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1928
7.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1736
8.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	4632
9.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1859
10.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1976
11.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2008
12.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1700
13.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1714
14.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1788
15.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2092
16.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2058
17.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	4398
18.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2032
19.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	3874
20.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	3994
Rata-rata			50129/20			**2506.45** ms

2. Kasus uji kurun waktu 2 bulan

Kasus uji kurun waktu 2 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 Agustus 2014 di lingkungan Android dengan SOAP *web service* yang ditunjukkan pada **Tabel 6.2**. Detil kasus uji kurun waktu 2 bulan SOAP *web service* lingkungan Android pada **Lampiran 2**.

Tabel 6.2 Kecepatan Akses Pengambilan Data kurun waktu 2 bulan Lingkungan Android dengan SOAP *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3891
2.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3544
3.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3361
4.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3429
5.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3320

6.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3498
7.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3343
8.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3424
9.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4172
10.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4075
11.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4437
12.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	5176
13.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4735
14.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4786
15.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4672
16.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4715
17.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4968
18.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3695
19.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3667
20.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	3715
Rata-rata			76956/20			**3847.8** ms

3. Kasus uji kurun waktu 3 bulan

Kasus uji kurun waktu 3 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 September 2014 di lingkungan Android dengan SOAP *web service* yang ditunjukkan pada **Tabel 6.3**. Detil kasus uji kurun waktu 3 bulan SOAP *web service* lingkungan Android terdapat pada **Lampiran 3**.

Tabel 6.3 Kecepatan Akses Pengambilan Data kurun waktu 3 bulan Lingkungan Android dengan SOAP *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	8063
2.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5952
3.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5920
4.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5398
5.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	4964
6.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5404
7.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5611
8.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5458
9.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5401
10.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5503
11.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5638
12.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5692
13.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	4930
14.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5094
15.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5249
16.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5534

17.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5872
18.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5443
19.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5188
20.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	4566
Jumlah			110880/20			**5544** ms

6.1.1.3. Hasil

Hasil pengujian kecepatan akses pada skenario lingkungan Android dengan SOAP *web service* ditunjukkan berdasarkan pada kurun waktu 1 bulan pada **Tabel 6.1**. Diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 Juli 2014 pukul 6 pagi. Dari masing-masing pengambilan data tersebut kemudian output mendapatkan sejumlah 401 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Pengambilan data input mendapatkan jumlah kecepatan akses sebesar 50129 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **2506.45** ms.

Diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 Agustus 2014 pukul 6 pagi untuk kurun waktu 2 bulan pada **Tabel 6.2**. Output dari masing-masing pengambilan data tersebut mendapatka 648 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Jumlah kecepatan akses dalam pengambilan data input sebesar 76956 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **3847.8** ms.

Dalam kurun waktu 3 bulan pada **Tabel 6.3** diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 September 2014 pukul 6 pagi. Dari masing-masing pengambilan data tersebut kemudian output mendapatkan sejumlah 1565 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Pengambilan data input mendapatkan jumlah kecepatan akses sebesar 110880 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **5544** ms.

Kecepatan akses pengambilan data pada penelitian ini menggunakan satuan ms (*millisecond*), untuk memudahkan dalam perhitungan maka satuan akan menggunakan s (*second*). Perhitungan dari 1 *millisecond* = 0.001 *second*, maka dari itu untuk rata-rata kecepatan akses lingkungan Android dengan menggunakan SOAP *web service* pada kurun waktu 1 bulan adalah sebesar 2506.45*0.001 = **2.5064** *second*, kurun waktu 2 bulan adalah sebesar 3847.8*0.001 = **3.8478** *second*, dan pada kurun waktu 3 bulan sebesar 5544*0.001 = **5.544** *second*.

6.1.2. Skenario REST *web service*

Pada pengujian kinerja lingkungan Android ini akan dilakukan pengamatan cara kerja dari kecepatan pertukaran data terhadap pengambilan data pada aplikasi monitoring klimatologi BMKG Karangploso dengan menggunakan REST *web service*. Pengujian dilakukan di device oleh karena itu dibutuhkan kondisi normal dan peralatan yang baik untuk jalannya pengujian. Berikut ini akan dijelaskan mengenai tujuan, prosedur, dan hasil analisis dari pengujian lingkungan Android dengan REST *web service*.

6.1.2.1. Tujuan

Tujuan pengujian kinerja pada lingkungan Android dengan REST *web service* adalah untuk mengetahui apakah sistem dapat berjalan dan memberikan hasil optimal pada saat melakukan pertukaran data *client-server*. Kesuksesan pengujian ditunjukkan dengan diperolehnya nilai kecepatan akses pengambilan data yang optimal.

6.1.2.2. Prosedur

Prosedur pada pengujian lingkungan Android dengan REST *web service* ini dilakukan dengan cara menjalankan sistem khususnya pada fitur monitoring klimatologi BMKG Karangploso. Pengujian dilakukan dengan cara mengambil data dalam kurun waktu 1 bulan, 2 bulan, dan 3 bulan sebanyak 20 kali dengan meng-*input*kan data wilayah, tanggal mulai, jam mulai, tanggal selesai, dan jam selesai. Percobaan dilakukan sebanyak 20 kali dikarenakan untuk mendapatkan hasil kecepatan akses yang bervariasi. Kasus uji coba kecepatan akses pengambilan data pada pengujian lingkungan Android dengan REST *web service* sebagai berikut:

1. Kasus uji kurun waktu 1 bulan

Kasus uji kurun waktu 1 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 Juli 2014 di lingkungan Android dengan REST *web service* yang ditunjukkan pada **Tabel 6.4**. Detil kasus uji kurun waktu 1 bulan REST *web service* lingkungan Android terdapat pada **Lampiran 4**.

Tabel 6.4 Kecepatan Akses Pengambilan Data kurun waktu 1 bulan Lingkungan Android dengan REST *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	324
2.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	166
3.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	197
4.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	293
5.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	178
6.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	155
7.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	158
8.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	161

9.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	195
10.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	190
11.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	160
12.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	112
13.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	163
14.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	130
15.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	184
16.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	139
17.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	174
18.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	135
19.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	174
20.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	117
Rata-rata			3505/20			**175.25** Ms

2. Kasus uji kurun waktu 2 bulan

Kasus uji kurun waktu 2 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 Agustus 2014 di lingkungan Android dengan REST *web service* yang ditunjukkan pada **Tabel 6.5**. Detil kasus uji kurun waktu 2 bulan REST *web service* lingkungan Android terdapat pada **Lampiran 5**.

Tabel 6.5 Kecepatan Akses Pengambilan Data kurun waktu 2 bulan Lingkungan Android dengan REST *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	401
2.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	324
3.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	291
4.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	334
5.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	317
6.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	303
7.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	312
8.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	287
9.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	276
10.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	269
11.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	419
12.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	909
13.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	329
14.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	172
15.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	308
16.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	326
17.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	300
18.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	378
19.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	319
20.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	419

Rata-rata	6993/20	**349.65** Ms

3. Kasus uji kurun waktu 3 bulan

Kasus uji kurun waktu 3 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 September 2014 di lingkungan Android dengan REST *web service* yang ditunjukkan pada **Tabel 6.6**. Detil kasus uji kurun waktu 3 bulan REST *web service* lingkungan Android terdapat pada **Lampiran 6**.

Tabel 6.6 Kecepatan Akses Pengambilan Data kurun waktu 3 bulan Lingkungan Android dengan REST *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1100
2.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	944
3.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	982
4.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	975
5.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	992
6.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1071
7.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1093
8.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	891
9.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	730
10.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1065
11.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	843
12.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1171
13.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	931
14.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	959
15.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1004
16.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1029
17.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	854
18.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	793
19.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	743
20.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	885
Rata-rata			19055/20			**952.75** ms

6.1.2.3. Hasil

Hasil pengujian kecepatan akses pada skenario lingkungan Android dengan REST *web service* ditunjukkan berdasarkan pada **Tabel 6.4**. Diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 Juli 2014 pukul 6 pagi. Dari masing-masing pengambilan data tersebut kemudian output mendapatkan sejumlah 401 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Pengambilan data

input mendapatkan jumlah kecepatan akses sebesar 3505 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **175.25** ms.

Diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 Agustus 2014 pukul 6 pagi untuk kurun waktu 2 bulan pada **Tabel 6.5**. Output dari masing-masing pengambilan data tersebut mendapatka 648 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Jumlah kecepatan akses dalam pengambilan data input sebesar 6993 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **349.65** ms.

Dalam kurun waktu 3 bulan pada **Tabel 6.6** diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 September 2014 pukul 6 pagi. Dari masing-masing pengambilan data tersebut kemudian output mendapatkan sejumlah 1565 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Pengambilan data input mendapatkan jumlah kecepatan akses sebesar 19055 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **952.75** ms.

Kecepatan akses pengambilan data pada penelitian ini menggunakan satuan millisecond (ms), untuk memudahkan dalam perhitungan maka satuan akan menggunakan second (s). Perhitungan dari 1 *millisecond* = 0.001 *second*, maka dari itu untuk rata-rata kecepatan akses lingkungan Android dengan menggunakan REST *web service* pada kurun waktu 1 bulan adalah sebesar 175.25*0.001 = **0.17525** *second*, kurun waktu 2 bulan adalah sebesar 349.65*0.001 = **0.3496** *second*, dan pada kurun waktu 3 bulan sebesar 952.75*0.001 = **0.95275** *second*.

6.1.3. Analisis Kinerja Lingkungan Android

Berdasarkan pengujian kecepatan akses lingkungan Android dengan REST *web service*, analisis dapat dibagi menjadi beberapa bagian, diantaranya:

6.1.3.1. Kecepatan Data

Berdasarkan hasil pengujian lingkungan Android yang diperoleh pada sub-bab 6.1.1 tentang skenario SOAP *web service* dan sub-bab 6.1.2 tentang *skenario* REST *web service* diperoleh hasil kecepatan akses rata-rata dalam pengambilan data yang diambil sebanyak 20 kali ditunjukkan pada **Tabel 6.7**.

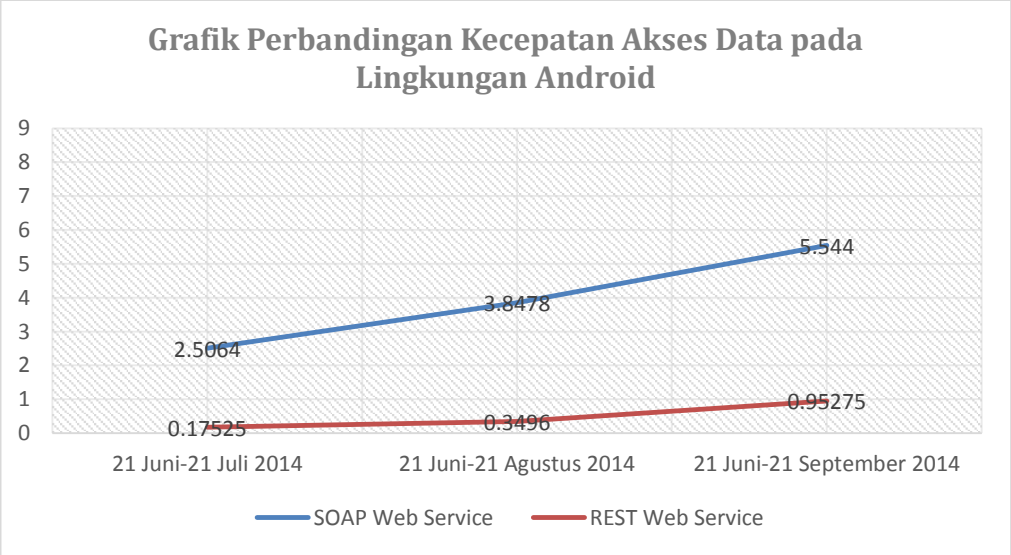
Tabel 6.7 Hasil Kecepatan Akses Rata-Rata dalam Pengambilan Data pada Lingkungan Android

No.	Kurun Waktu	Banyak Data	Android	
			SOAP *Web Service*	REST *Web Service*

1.	1 bulan (21 Juni 2014–21 Juli 2014)	401 Data	**2.5064** *second*	**0.17525** *second*
2.	2 bulan (21 Juni 2014–21 Agustus 2014)	1049 Data	**3.8478** *second*	**0.3496** *second*
3.	3 bulan (21 Juni 2014–21 September 2014)	1565 Data	**5.544** *second*	**0.95275** *second*

Hasil pengujian dalam pengambilan data pada lingkungan Android dapat diketahui bahwa untuk kurun waktu 1 bulan (21 Juni 2014-21 Juli 2014) dengan skenario SOAP *web service* menghasilkan kecepatan akses data rata-rata sebesar 2.5064 detik, sedangkan pada skenario REST *web service* menghasilkan kecepatan akses data lebih cepat dibandingkan pada skenario SOAP *web service* yaitu sebesar 0.17525 detik. Kurun waktu 2 bulan (21 Juni 2014-21 Agustus 2014) dalam skenario SOAP *web service* menghasilkan kecepatan akses data rata-rata sebesar 3.8478 detik. Dan untuk skenario REST *web service* menghasilkan kecepatan akses rata-rata data sebesar 0.3496 detik. Pada kurun waktu 3 bulan (21 Juni 2014-21 September 2014) dalam skenario SOAP *web service* kecepatan akses rata-rata data diperoleh sebesar 5.544 detik, dan pada skenario REST *web service* diperoleh hasil kecepatan akses rata-rata data sebesar 0.95275 detik.

Hal ini menunjukkan bahwa pada kurun waktu 1 bulan sampai 3 bulan kecepatan akses rata-rata pengambilan data sebanyak 20 kali dengan menggunakan SOAP dan REST *web service* dipengaruhi oleh banyaknya data. Hasil kecepatan akses pengambilan data dapat disimpulkan bahwa REST *web service* lebih cepat daripada SOAP *web service*. Grafik perbandingan kecepatan akses yang dipengaruhi oleh banyaknya data pada lingkungan Android ditunjukkan dalam **Gambar 6.2**.



Gambar 6.2 Grafik Perbandingan Kecepatan Akses Data pada Lingkungan Android

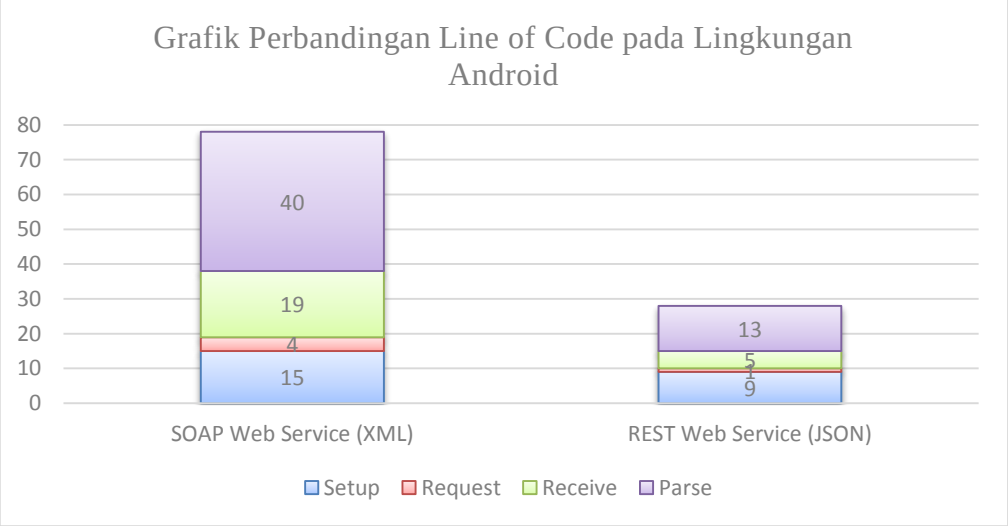
6.1.3.2. *Line of Code*

Berdasarkan pada bab 5 yaitu implementasi, analisis *line of code* dilakukan untuk mengetahui banyaknya baris *coding* guna menunjang keefektifan suatu program pada *client web service* SOAP sub-bab 5.2.1.1 dan *client web service* REST sub-bab 5.2.2.1 dalam lingkungan Android. Banyaknya jumlah *line of code* SOAP dan REST *web service* pada lingkungan Android terbagi menjadi beberapa bagian seperti yang ditunjukkan pada **Tabel 6.8**.

Tabel 6.8 Jumlah *line of code* SOAP dan REST *web service* pada Lingkungan Android

No.	*Line of Code*	Android	
		SOAP *Web Service*	REST *Web Service*
1.	*Setup*	15 *line*	9 *line*
2.	*Request*	4 *line*	1 *line*
3.	*Receive*	19 *line*	5 *line*
4.	*Parse*	40 *line*	13 line
Jumlah		**78** *line*	**28** *line*

Struktur *client web service* pada Lingkungan Android terdiri dari beberapa bagian diantaranya *Setup*, *Request*, *Receive*, dan *Parse*. SOAP *web service* menggunakan format data XML, dan REST *web service* menggunakan format data berupa JSON. Tiap-tiap bagian struktur memiliki jumlah *line of code* (baris kode) sesuai dengan kebutuhan. Detil *screenshot* perbandingan *Line of code* dari lingkungan Android terdapat pada **Lampiran 7**. Pada lingkungan Android menggunakan SOAP *web service* memiliki jumlah baris sebanyak 78 *line*, dan pada lingkungan Android menggunakan REST *web service* memiliki jumlah baris sebesar 28 *line*. Grafik perbandingan *line of code* untuk SOAP dan REST *web service* pada lingkungan Android ditunjukkan dalam **Gambar 6.3**.



Gambar 6.3 Grafik Perbandingan *Line of Code* pada Lingkungan Android

6.2. Pengujian Kinerja Lingkungan Windows Phone

Pada sub bab mengacu dalam **Gambar 4.3** yang menjelaskan skenario pengujian kinerja pengambilan data dalam lingkungan Windows Phone. Pengujian kinerja merupakan pengujian yang dilakukan untuk mengukur kecepatan pengambilan data yang akan diakses dari *client* Windows Phone ke *server*. *Server* yang digunakan dalam penelitian ini merupakan *server localhost* yang bertujuan untuk meminimalkan paket data/data dari luar skenario dalam pengambilan data di jaringan. Pendekatan pengembangan menggunakan *native development* juga digunakan untuk mendapatkan kinerja aplikasi yang optimal. Pengujian kinerja tidak menekankan pada jalannya algoritme sistem. Namun lebih kepada mengamati kecepatan pertukaran data dalam pengambilan data dari *client* ke *server*.

6.2.1. Skenario SOAP *web service*

Pada pengujian kinerja lingkungan Windows Phone ini akan dilakukan pengamatan cara kerja dari kecepatan pertukaran data terhadap pengambilan data pada aplikasi monitoring klimatologi BMKG Karangploso dengan menggunakan SOAP *web service*. Pengujian dilakukan di device oleh karena itu dibutuhkan kondisi normal dan peralatan yang baik untuk jalannya pengujian. Berikut ini akan dijelaskan mengenai tujuan, prosedur, dan hasil analisis dari pengujian lingkungan Windows Phone dengan SOAP *web service*.

6.2.1.1. Tujuan

Tujuan pengujian kinerja pada lingkungan Windows Phone dengan SOAP *web service* adalah untuk mengetahui apakah sistem dapat berjalan dan memberikan hasil optimal pada saat melakukan pengambilan data *client-server*. Kesuksesan pengujian ditunjukkan dengan diperolehnya nilai kecepatan akses pengambilan data yang optimal.

6.2.1.2. Prosedur

Prosedur pada pengujian lingkungan Windows Phone dengan SOAP *web service* ini dilakukan pada fitur monitoring klimatologi BMKG Karangploso. Pengujian dilakukan dengan cara melakukan percobaaan pengambilan data dalam kurun waktu 1 bulan, 2 bulan, dan 3 bulan sebanyak 20 kali dengan meng-*input*kan data wilayah, tanggal mulai, jam mulai, tanggal selesai, dan jam selesai. Percobaan dilakukan sebanyak 20 kali untuk mendapatkan hasil kecepatan akses yang bervariasi. Kasus uji percobaan kecepatan akses pengambilan data pada pengujian lingkungan Windows Phone dengan SOAP *web service* sebagai berikut:

1. **Kasus uji kurun waktu 1 bulan**

   Kasus uji kurun waktu 1 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 Juli 2014 di lingkungan Windows

Phone dengan SOAP *web service* yang ditunjukkan pada **Tabel 6.9**. Detil kasus uji kurun waktu 1 bulan SOAP *web service* lingkungan Windows Phone terdapat pada **Lampiran 8**.

Tabel 6.9 Kecepatan Akses Pengambilan Data kurun waktu 1 bulan Lingkungan Windows Phone dengan SOAP *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	4185.3901
2.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2591.5004
3.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	3698.0291
4.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2721.6668
5.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2393.7487
6.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	3409.8582
7.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2693.7148
8.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2607.1837
9.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	3839.6798
10.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	6737.864
11.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	3180.7271
12.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2925.9403
13.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	5483.0382
14.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2055.0276
15.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	3621.751
16.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2994.7041
17.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	4816.2255
18.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2337.5296
19.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	3172.7409
20.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2809.2261
Rata-rata				68275.546/20		**3413.77** ms

2. Kasus uji kurun waktu 2 bulan

Kasus uji kurun waktu 2 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 Agustus 2014 di lingkungan Windows Phone dengan SOAP *web service* yang ditunjukkan pada **Tabel 6.10**. Detil kasus uji kurun waktu 2 bulan SOAP *web service* lingkungan Windows Phone terdapat pada **Lampiran 9**.

Tabel 6.10 Kecepatan Akses Pengambilan Data kurun waktu 2 bulan Lingkungan Windows Phone dengan SOAP *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	7646.3446
2.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6347.3946
3.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4906.7714

4.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	5420.6606
5.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6754.3654
6.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	5936.3908
7.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6322.3436
8.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6167.1887
9.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6184.2505
10.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6670.8833
11.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4028.5931
12.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6430.3701
13.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4256.9025
14.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6146.8199
15.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4692.1314
16.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6696.0501
17.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4408.3111
18.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	4687.3255
19.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6372.7151
20.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	6584.9002
Rata-rata			116660.7125/20			**5833.03** ms

3. Kasus uji kurun waktu 3 bulan

Kasus uji kurun waktu 3 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 September 2014 di lingkungan Windows Phone dengan SOAP *web service* yang ditunjukkan pada **Tabel 6.11**. Detil kasus uji kurun waktu 3 bulan SOAP *web service* lingkungan Windows Phone terdapat pada **Lampiran 10**.

Tabel 6.11 Kecepatan Akses Pengambilan Data kurun waktu 3 bulan Lingkungan Android dengan SOAP *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	7267.556
2.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	9205.3897
3.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	9089.2073
4.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5545.5723
5.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	6366.8557
6.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	8249.8311
7.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	8708.1381
8.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	9243.4047
9.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	14522.0609
10.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	7984.7449
11.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	11660.3715
12.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	7128.1841
13.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	6402.212
14.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	7743.1754

15.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	6512.2403
16.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	6911.7299
17.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	7591.8596
18.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	8795.1669
19.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	6013.7795
20.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	5950.3008
Rata-rata			160891.78/20			**8044.58** ms

6.2.1.3. Hasil

Hasil pengujian kecepatan akses pada skenario lingkungan Windows Phone dengan SOAP *web service* ditunjukkan berdasarkan pada kurun waktu 1 bulan pada **Tabel 6.9**. Diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 Juli 2014 pukul 6 pagi. Dari masing-masing pengambilan data tersebut kemudian output mendapatkan sejumlah 401 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Pengambilan data input mendapatkan jumlah kecepatan akses sebesar 68275.546 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **3413.77** ms.

Diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 Agustus 2014 pukul 6 pagi untuk kurun waktu 2 bulan pada **Tabel 6.10**. Output dari masing-masing pengambilan data tersebut mendapatka 648 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Jumlah kecepatan akses dalam pengambilan data input sebesar 116660.7125 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **5833.03** ms.

Dalam kurun waktu 3 bulan pada **Tabel 6.11** diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 September 2014 pukul 6 pagi. Dari masing-masing pengambilan data tersebut kemudian output mendapatkan sejumlah 1565 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Pengambilan data input mendapatkan jumlah kecepatan akses sebesar 160891.78 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **8044.58** ms.

Kecepatan akses pengambilan data pada penelitian ini menggunakan satuan millisecond (ms), untuk memudahkan dalam perhitungan maka satuan akan menggunakan second (s). Perhitungan dari 1 *millisecond* = 0.001 *second*, maka dari itu untuk rata-rata kecepatan akses lingkungan Windows Phone dengan menggunakan SOAP *web service* pada kurun waktu 1 bulan adalah sebesar 3413.77*0.001 = **3.41377** *second*, kurun waktu 2 bulan adalah sebesar 5833.03*0.001 = **5.83303** *second*, dan pada kurun waktu 3 bulan sebesar 8044.58*0.001 = **8.04458** *second*.

6.2.2. Skenario REST *web service*

Pengujian kinerja lingkungan Windows Phone ini akan dilakukan pengamatan cara kerja dari kecepatan pertukaran data terhadap pengambilan data aplikasi monitoring klimatologi BMKG Karangploso dengan menggunakan REST *web service*. Pengujian dilakukan di device oleh karena itu dibutuhkan kondisi normal dan peralatan yang baik untuk jalannya pengujian. Berikut akan dijelaskan mengenai tujuan, prosedur, dan hasil analisis dari pengujian lingkungan Windows Phone dengan REST *web service*.

6.2.2.1. Tujuan

Tujuan pengujian kinerja pada lingkungan Windows Phone dengan REST *web service* adalah untuk mengetahui apakah sistem dapat berjalan dan memberikan hasil optimal pada saat melakukan pengambilan data *client-server*. Kesuksesan pengujian ditunjukkan dengan diperolehnya nilai kecepatan akses pengambilan data yang optimal.

6.2.2.2. Prosedur

Prosedur pengujian lingkungan Windows Phone dengan REST *web service* dilakukan pada fitur monitoring klimatologi BMKG Karangploso. Pengujian dengan mengambil data kurun waktu 1 bulan, 2 bulan, dan 3 bulan sebanyak 20 kali dengan meng-*input*kan data wilayah, tanggal mulai, jam mulai, tanggal selesai, dan jam selesai. Percobaan dilakukan sebanyak 20 kali untuk mendapat hasil kecepatan akses yang bervariasi. Kasus uji percobaan kecepatan akses pengambilan data pada pengujian lingkungan Windows Phone dengan REST *web service* sebagai berikut:

1. Kasus uji kurun waktu 1 bulan

Kasus uji kurun waktu 1 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 Juli 2014 di lingkungan Windows Phone dengan REST *web service* yang ditunjukkan pada **Tabel 6.12**. Detil kasus uji kurun waktu 1 bulan REST *web service* lingkungan Windows Phone terdapat pada **Lampiran 11**.

Tabel 6.12 Kecepatan Akses Pengambilan Data kurun waktu 1 bulan Lingkungan Windows Phone dengan REST *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1342.8606
2.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	718.1203
3.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2546.305
4.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2035.5278
5.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	822.971
6.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	2395.0606

7.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1262.9524
8.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	761.0679
9.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	956.3028
10.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1686.5486
11.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1442.5701
12.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	721.2954
13.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	844.6605
14.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	793.9329
15.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	1446.2394
16.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	977.0685
17.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	988.8209
18.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	974.1659
19.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	930.7979
20.	Karangploso	21-06-2014	06:00	21-07-2014	06:00	668.2489
Rata-rata			24315.5174/20			**1215.77** Ms

2. Kasus uji kurun waktu 2 bulan

Kasus uji kurun waktu 2 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 Agustus 2014 di lingkungan Windows Phone dengan REST *web service* yang ditunjukkan pada **Tabel 6.13**. Detil kasus uji kurun waktu 2 bulan REST *web service* lingkungan Windows Phone terdapat pada **Lampiran 12**.

Tabel 6.13 Kecepatan Akses Pengambilan Data kurun waktu 2 bulan Lingkungan Windows Phone dengan REST *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1754.7604
2.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	719.3148
3.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	2520.3817
4.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	2721.8736
5.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	2793.5972
6.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	772.0225
7.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1120.2724
8.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1518.1967
9.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1048.1448
10.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1304.4075
11.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	768.8202
12.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1406.7364
13.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	2543.196
14.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1437.9305
15.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1657.4034
16.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1765.5923

17.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	819.111
18.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1700.2215
19.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1845.8747
20.	Karangploso	21-06-2014	06:00	21-08-2014	06:00	1608.6588
Rata-rata			31826.5164/20			**1591.32** Ms

3. Kasus uji kurun waktu 3 bulan

Kasus uji kurun waktu 3 bulan menjelaskan pengujian kecepatan akses data pada tanggal 21 Juni 2014 sampai tanggal 21 September 2014 di lingkungan Windows Phone dengan REST *web service* yang ditunjukkan pada **Tabel 6.14**. Detil kasus uji kurun waktu 3 bulan REST *web service* lingkungan Windows Phone terdapat pada **Lampiran 13**.

Tabel 6.14 Kecepatan Akses Pengambilan Data kurun waktu 3 bulan Lingkungan Windows Phone dengan REST *web service*

No.	Wilayah	Tanggal Mulai	Jam Mulai	Tanggal Selesai	Jam Selesai	Waktu Akses (ms)
1.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	3949.5962
2.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	3600.4123
3.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1599.0273
4.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1973.7548
5.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1616.6881
6.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	2145.7387
7.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1878.0429
8.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1165.6514
9.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1091.463
10.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1178.5298
11.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1169.3782
12.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1157.4659
13.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1566.2693
14.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1142.0309
15.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1360.6166
16.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1133.6872
17.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1169.1624
18.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1138.6466
19.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	1143.5607
20.	Karangploso	21-06-2014	06:00	21-09-2014	06:00	2099.6532
Rata-rata			33279.37/20			**1663.96** ms

6.2.2.3. Hasil

Hasil pengujian kecepatan akses pada skenario lingkungan Windows Phone dengan REST *web service* ditunjukkan berdasarkan pada **Tabel 6.12**. Diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal

21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 Juli 2014 pukul 6 pagi. Dari masing-masing pengambilan data tersebut kemudian output mendapatkan sejumlah 401 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Pengambilan data input mendapatkan jumlah kecepatan akses sebesar 24315.5174 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **1215.77** ms.

Diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 Agustus 2014 pukul 6 pagi untuk kurun waktu 2 bulan pada **Tabel 6.13**. Output dari masing-masing pengambilan data tersebut mendapatka 648 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Jumlah kecepatan akses dalam pengambilan data input sebesar 31826.5164 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **1591.32** ms.

Dalam kurun waktu 3 bulan pada **Tabel 6.14** diketahui dengan 20 kali pengambilan data input pada wilayah Karangploso, pada tanggal 21 Juni 2014 pukul 6 pagi sampai dengan tanggal 21 September 2014 pukul 6 pagi. Dari masing-masing pengambilan data tersebut kemudian output mendapatkan sejumlah 1565 data berupa wilayah karangploso, curah hujan, suhu, arah angin, kecepatan angin, kelembaban, waktu, dan jam yang sesuai dengan data input. Pengambilan data input mendapatkan jumlah kecepatan akses sebesar 33279.37 ms, kemudian dibagi 20 sehingga mendapatkan rata-rata kecepatan akses sebesar **1663.96** ms.

Kecepatan akses pengambilan data pada penelitian ini menggunakan satuan millisecond (ms), untuk memudahkan dalam perhitungan maka satuan akan menggunakan second (s). Perhitungan dari 1 *millisecond* = 0.001 *second*, maka dari itu untuk rata-rata kecepatan akses lingkungan Windows Phone dengan menggunakan REST *web service* pada kurun waktu 1 bulan adalah sebesar 1215.77*0.001 = **1.21577** *second*, kurun waktu 2 bulan adalah sebesar 1591.32*0.001 = **1.59132** *second*, dan pada kurun waktu 3 bulan sebesar 1663.96*0.001 = **1.66396** *second*.

6.2.3. Analisis Kinerja Lingkungan Windows Phone

Berdasarkan pengujian kecepatan akses lingkungan Android dengan REST *web service*, analisis dapat dibagi menjadi beberapa bagian, diantaranya:

6.2.3.1. Kecepatan Data

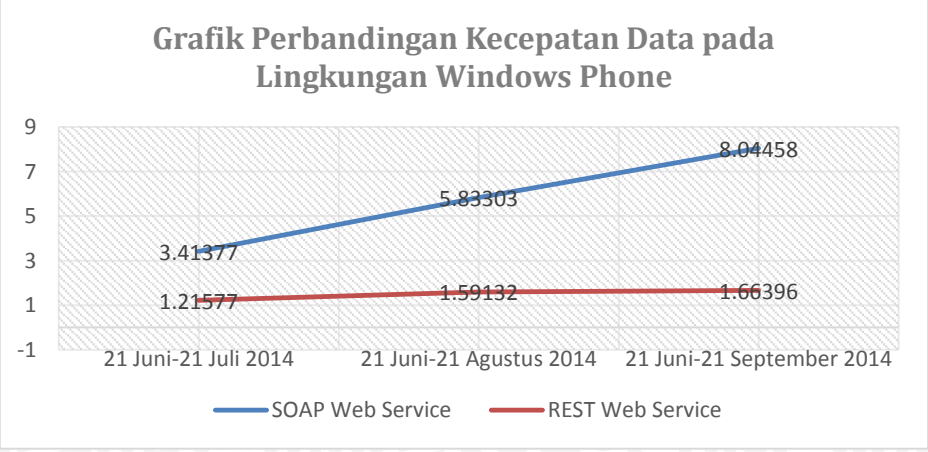
Berdasarkan hasil pengujian lingkungan Windows Phone yang diperoleh pada sub-bab 6.2.1 tentang skenario SOAP *web service* dan sub-bab 6.2.2 tentang *skenario* REST *web service* diperoleh hasil kecepatan akses rata-rata dalam pengambilan data yang diambil sebanyak 20 kali ditunjukkan pada **Tabel 6.15**.

Tabel 6.15 Hasil Kecepatan Akses Rata-Rata dalam Pengambilan Data pada Lingkungan Windows Phone

No.	Kurun Waktu	Banyak Data	Windows Phone	
			SOAP *Web Service*	REST *Web Service*
1.	1 bulan (21 Juni 2014–21 Juli 2014)	401 Data	**3.41377** *second*	**1.21577** *second*
2.	2 bulan (21 Juni 2014–21 Agustus 2014)	1049 Data	**5.83303** *second*	**1.59132** *second*
3.	3 bulan (21 Juni 2014–21 September 2014)	1565 Data	**8.04458** *second*	**1.66396** *second*

Hasil pengujian dalam pengambilan data pada lingkungan Windows Phone dapat diketahui bahwa untuk kurun waktu 1 bulan (21 Juni 2014-21 Juli 2014) dengan skenario SOAP *web service* menghasilkan kecepatan akses data rata-rata sebesar 3.41377 detik, sedangkan pada skenario REST *web service* menghasilkan kecepatan akses data lebih cepat dibandingkan pada skenario SOAP *web service* yaitu sebesar 1.21577 detik. Kurun waktu 2 bulan (21 Juni 2014-21 Agustus 2014) dalam skenario SOAP *web service* menghasilkan kecepatan akses data rata-rata sebesar 5.83303 detik. Dan untuk skenario REST *web service* menghasilkan kecepatan akses rata-rata data sebesar 1.59132 detik.

Pada kurun waktu 3 bulan (21 Juni 2014-21 September 2014) dalam skenario SOAP *web service* kecepatan akses rata-rata data diperoleh sebesar 8.04458 detik, dan pada skenario REST *web service* diperoleh hasil kecepatan akses rata-rata data sebesar 1.66396 detik. Hal ini menunjukkan bahwa pada kurun waktu 1 bulan sampai 3 bulan kecepatan akses rata-rata pengambilan data sebanyak 20 kali dengan menggunakan SOAP dan REST *web service* dipengaruhi oleh banyaknya data. Grafik perbandingan kecepatan akses yang dipengaruhi oleh banyaknya data pada lingkungan Windows Phone ditunjukkan dalam **Gambar 6.4**.



Gambar 6.4 Grafik Perbandingan Kecepatan Akses Data pada Lingkungan Windows Phone

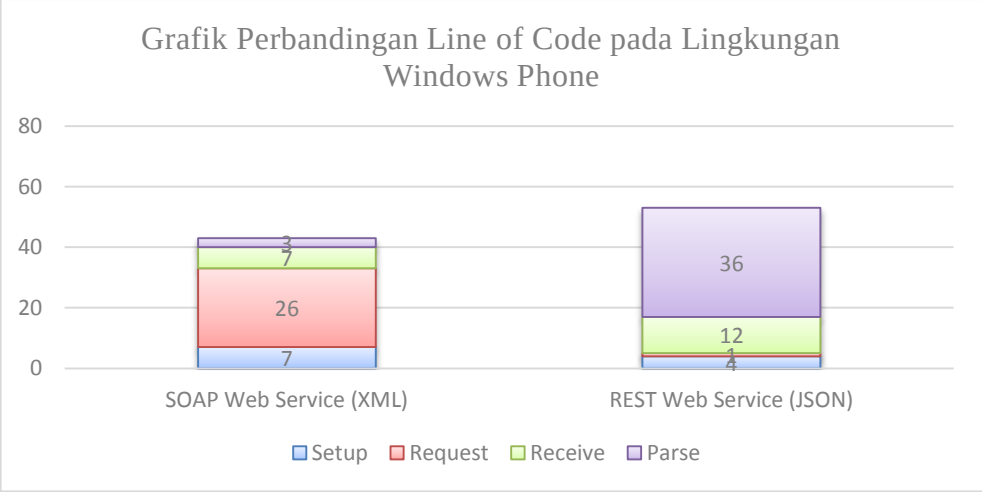
6.2.3.2. *Line of Code*

Berdasarkan pada bab 5 yaitu implementasi, analisis *line of code* dilakukan untuk mengetahui banyaknya baris *coding* guna menunjang keefektifan suatu program *client web service* SOAP sub-bab 5.2.1.2 dan *client web service* REST sub-bab 5.2.2.2 dalam lingkungan Windows Phone. Banyaknya jumlah *line of code* SOAP dan REST *web service* pada lingkungan Windows Phone terbagi beberapa bagian seperti yang ditunjukkan pada **Tabel 6.16**.

Tabel 6.16 Jumlah line of code SOAP dan REST *web service* pada Lingkungan Windows Phone

No.	*Line of Code*	Windows Phone	
		SOAP *Web Service*	REST *Web Service*
1.	*Setup*	7 *line*	4 *line*
2.	*Request*	26 *line*	1 *line*
3.	*Receive*	7 *line*	12 *line*
4.	*Parse*	3 *line*	36 line
Jumlah		**43** *line*	**53** *line*

Struktur *client web service* pada Lingkungan Windows Phone terdiri dari beberapa bagian diantaranya *Setup*, *Request*, *Receive*, dan *Parse*. SOAP *web service* menggunakan format data XML, dan REST *web service* menggunakan format data berupa JSON. Tiap-tiap bagian struktur memiliki jumlah *line of code* sesuai kebutuhan. Detil *screenshot* perbandingan *Line of code* dari lingkungan Windows Phone terdapat pada **Lampiran 14**. Lingkungan Windows Phone menggunakan SOAP *web service* memiliki jumlah baris 43 *line*, dan lingkungan Windows Phone menggunakan REST *web service* memiliki jumlah baris sebesar 53 *line*. Grafik perbandingan *line of code* SOAP dan REST *web service* lingkungan Windows Phone ditunjukkan dalam **Gambar 6.5**.



Gambar 6.5 Grafik Perbandingan *Line of Code* Lingkungan Windows Phone

6.3. Komparasi Kinerja Lingkungan Android dan Windows Phone

Komparasi kinerja lingkungan Android dan Windows Phone merupakan hasil dari analisis pengujian kinerja kecepatan akses data SOAP *web service* dan REST *web service*. Mekanisme komparasi kinerja dimulai dari komparasi didalam lingkungan, yaitu komparasi lingkungan Android, dan komparasi lingkungan Windows Phone. Hasil komparasi dalam lingkungan yang telah didapat kemudian akan dikomparasi kembali dengan antar lingkungan, yaitu komparasi Android dengan Windows Phone.

6.3.1. Komparasi Dalam Lingkungan

Komparasi dalam lingkungan dibagi menjadi 2 bagian, yaitu Komparasi dalam lingkungan Android dan Komparasi dalam Lingkungan Windows Phone.

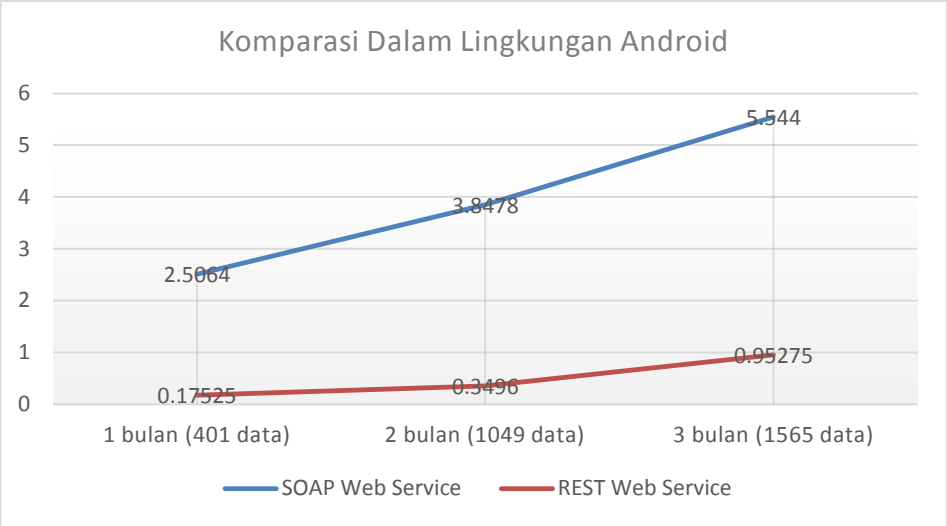
a. Komparasi dalam lingkungan Android

Komparasi dalam lingkungan berdasarkan analisis pengujian kinerja yang terdapat pada sub-bab 6.1.3.1 yang didapatkan pada lingkungan Android. Komparasi kinerja pengambilan data dalam lingkungan Android menggunakan dua metode yaitu SOAP *web service* dan REST *web service*. Diketahui pada kinerja kecepatan akses pengambilan data pada lingkungan Android dipengaruhi oleh banyaknya data dari tiap bulan seperti yang ditunjukkan **Tabel 6.17**.

Tabel 6.17 Komparasi dalam Lingkungan Android

Kurun Waktu (Banyak data)	Android	
	SOAP *Web Service*	REST *Web Service*
1 bulan (401 data)	**2.5064** *second*	**0.17525** *second*
2 bulan (1049 data)	**3.8478** *second*	**0.3496** *second*
3 bulan (1565 data)	**5.544** *second*	**0.95275** *second*
Rata-rata	**3.9660** *second*	**0.4191** *second*

Komparasi dalam lingkungan Android **Tabel 6.17** menunjukkan bahwa pada kurun waktu 1 bulan sampai 3 bulan mengalami perubahan kecepatan akses yang semakin menurun. Kecepatan rata-rata akses pada SOAP *web service* dalam lingkungan Android sebesar **3.9660 *second***. Kecepatan rata-rata akses pada REST *web service* dalam lingkungan Android sebesar **0.4191 *second***. Komparasi kecepatan akses dalam lingkungan Android dapat ditunjukkan dalam bentuk grafik dalam **Gambar 6.6**. Hal ini menunjukkan bahwa kecepatan akses pengambilan data menggunakan REST *web service* dalam lingkungan Android lebih unggul dibandingkan dengan SOAP *web service*.



Gambar 6.6 Komparasi Dalam Lingkungan Android

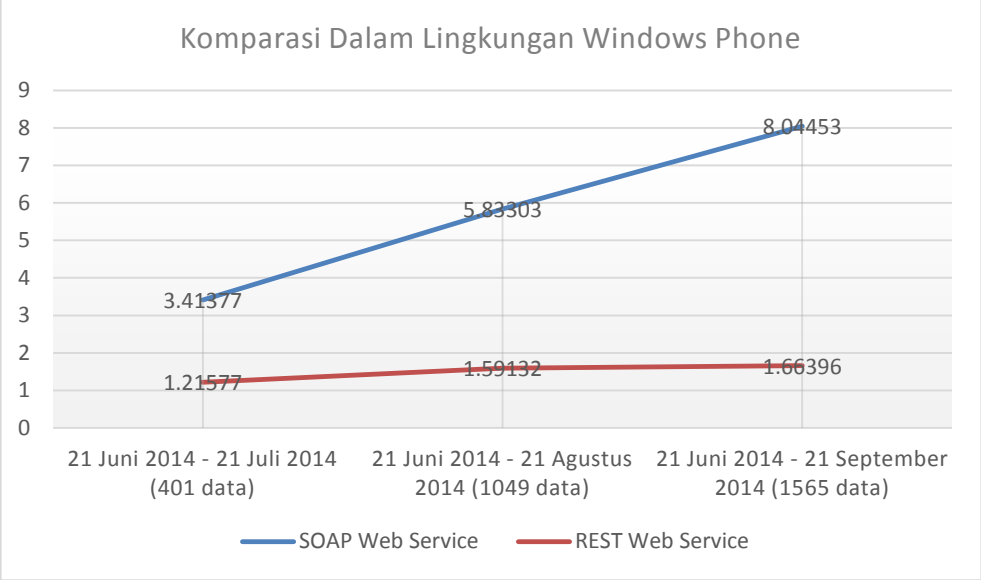
b. Komparasi dalam lingkungan Windows Phone

Komparasi dalam lingkungan berdasarkan analisis pengujian kinerja yang terdapat pada sub-bab 6.2.3.1 yang didapatkan pada lingkungan Windows Phone. Komparasi kinerja pengambilan data dalam lingkungan Windows Phone menggunakan dua metode yaitu SOAP *web service* dan REST *web service*. Diketahui pada kinerja kecepatan akses pengambilan data pada lingkungan Windows Phone dipengaruhi oleh banyaknya data dari tiap bulan seperti yang ditunjukkan **Tabel 6.18**.

Tabel 6.18 Komparasi dalam Lingkungan Windows Phone

Kurun Waktu (Banyak data)	Windows Phone	
	SOAP *Web Service*	REST *Web Service*
1 bulan (401 data)	**3.41377** *second*	**1.21577** *second*
2 bulan (1049 data)	**5.83303** *second*	**1.59132** *second*
3 bulan (1565 data)	**8.04458** *second*	**1.66396** *second*
Rata-rata	**5.7637** *second*	**1.49035** *second*

Komparasi dalam lingkungan Windows Phone **Tabel 6.18** menunjukkan bahwa pada kurun waktu 1 bulan sampai 3 bulan mengalami perubahan kecepatan akses yang semakin menurun. Kecepatan rata-rata akses pada SOAP *web service* dalam lingkungan Windows Phone sebesar **5.7637** ***second***. Kecepatan rata-rata akses pada REST *web service* dalam lingkungan Windows Phone sebesar **1.49035** ***second***. Komparasi kecepatan akses dalam lingkungan Windows Phone dapat ditunjukkan dalam bentuk grafik dalam **Gambar 6.7**. Hal ini menunjukkan bahwa kecepatan akses pengambilan data menggunakan REST *web service* dalam lingkungan Windows Phone lebih unggul dibandingkan dengan SOAP *web service*.



Gambar 6.7 Komparasi Dalam Lingkungan Windows Phone

6.3.2. Komparasi Antar Lingkungan

Komparasi antar lingkungan merupakan hasil dari komparasi dalam lingkungan antara lingkungan Android dan lingkungan Windows Phone. Komparasi dalam lingkungan Android mendapatkan hasil bahwa kinerja kecepatan akses pengambilan data menggunakan REST *web service* lebih cepat dibanding dengan menggunakan SOAP web service. Komparasi dalam lingkungan Windows Phone juga menunjukkan bahwa kinerja kecepatan akses pengambilan data menggunakan REST *web service* lebih cepat dibandingkan menggunakan SOAP *web service*. Komparasi antar lingkungan Android dan Windows Phone bertujuan untuk mendapatkan hasil kinerja kecepatan akses data paling cepat dan efektif untuk aplikasi monitoring klimatologi BMKG Karangploso Malang.

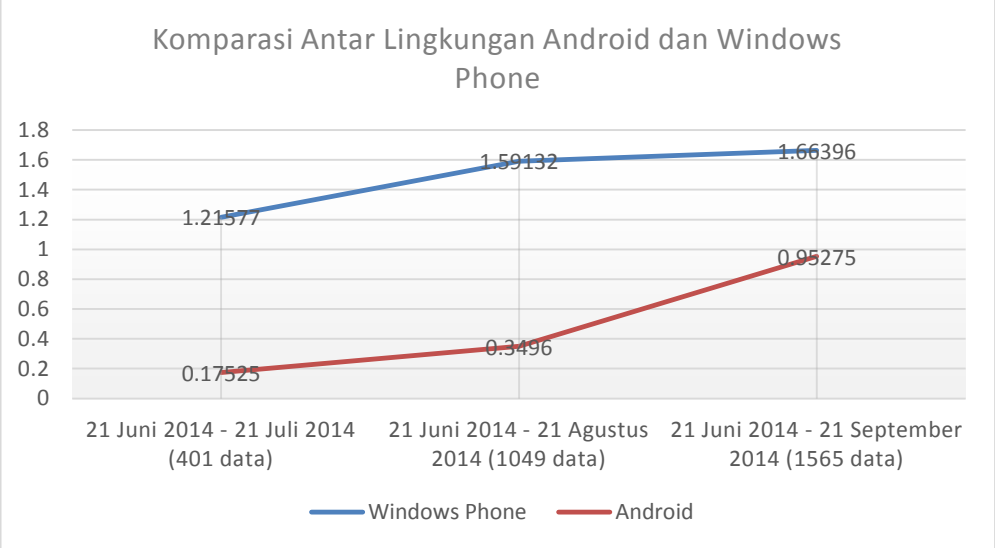
Tabel 6.19 Komparasi Antar Lingkungan Android dan Windows Phone

Kurun Waktu (Banyak data)	Komparasi Antar Lingkungan	
	REST Android	REST Windows Phone
1 bulan (401 data)	**0.17525** *second*	**1.21577** *second*
2 bulan (1049 data)	**0.3496** *second*	**1.59132** *second*
3 bulan (1565 data)	**0.95275** *second*	**1.66396** *second*
Rata-rata	**0.4191** *second*	**1.49035** *second*

Komparasi antar lingkungan pada lingkungan Android dan lingkungan Windows Phone dengan menggunakan REST *web service* berdasarkan **Tabel 6.19** dapat diketahui untuk kurun waktu 1 bulan dengan banyak 401 data, kecepatan akses pengambilan data pada lingkungan Android sebesar 0.17525 *second* dan pada lingkungan Windows Phone sebesar 1.21577 *second*. Kurun waktu 2 bulan dengan banyak 1049 data, untuk kecepatan akses pengambilan data pada

lingkungan Android didapatkan sebesar 0.3496 *second* dan pada lingkungan Windows Phone sebesar 1.59132 *second*. Komparasi antar lingkungan dalam kurun waktu 3 bulan dengan banyak 1565 data, pada lingkungan Android didapatkan 0.95275 *second* dan pada lingkungan Windows Phone didapatkan 1.66396 *second* kecepatan akses pengambilan data.

Uraian dari komparasi antar lingkungan yang telah dilakukan membuktikan bahwa kinerja kecepatan akses pengambilan data dipengaruhi oleh banyaknya data. Semakin banyak data yang diambil, maka semakin banyak waktu yang butuhkan. Hasil akhir dari komparasi kinerja rata-rata kecepatan akses pengambilan data dengan menggunakan REST *web service* pada **Tabel 6.19** untuk lingkungan Android sebesar **0.4191** *second* dan untuk lingkungan Windows Phone sebesar **1.49035** *second*. Perhitungan rata-rata kecepatan akses pengambilan data antar lingkungan Android dan Windows Phone diketahui dengan cara membagi rata-rata perolehan kecepatan akses pengambilan data pada lingkungan Windows Phone terhadap lingkungan Android seperti berikut $\frac{1.49035\ second}{0.4191\ second}$ = 3.55 kali. Hal ini menunjukkan bahwa pada lingkungan Android 3.55x lebih cepat dibandingkan dengan lingkungan Windows Phone. Komparasi kecepatan akses antar lingkungan Android dan Windows Phone dapat ditunjukkan dalam bentuk grafik dalam **Gambar 6.8**.



Gambar 6.8 Komparasi Antar Lingkungan Android dan Windows Phone

6.3.3. Analisis Kesimpulan

Analisis kesimpulan merupakan sebuah analisis akhir dari penelitian studi perbandingan kinerja SOAP *web service* dengan REST *web service* dalam pertukaran data pada *platform* Android dan Windows Phone. Analisis kesimpulan bertujuan untuk menganalisis keseluruhan penelitian yang telah dilakukan. Analisis kesimpulan meliputi analisis pengujian kinerja kecepatan akses pengambilan data, analisis *line of code*, dan hasil komparasi antar lingkungan yang

telah dilakukan sebelumnya. Analisis kesimpulan kinerja kecepatan akses pertukaran data pada penelitian ini diketahui sebagai berikut:

1. REST *web service* lebih baik dalam segi kecepatan akses pengambilan data dibandingkan dengan SOAP *web service* dalam kasus uji antara lingkungan Android maupun lingkungan Windows Phone.
2. REST *web service* dalam lingkungan Android mempunyai hasil kecepatan akses pengambilan data lebih cepat dibandingkan dengan lingkungan Windows Phone.
3. Baris *line of code* dari lingkungan Android dengan menggunakan REST *web service* lebih sedikit dibandingkan dengan menggunakan SOAP *web service*.
4. Baris *line of code* dari lingkungan Windows Phone dengan menggunakan SOAP *web service* lebih sedikit dibandingkan dengan menggunakan REST *web service*.
5. Hasil komparasi studi perbandingan pada lingkungan Android dan lingkungan Windows Phone diketahui bahwa REST *web service* dalam lingkungan Android mempunyai rata-rata kecepatan akses pengambilan data sebesar **0.4191** *second* dan 3.55x lebih cepat dibandingkan dengan REST *web service* dengan rata-rata kecepatan akses pengambilan data sebesar **1.49035** *second* dalam lingkungan Windows Phone.

BAB 7 PENUTUP

7.1. Kesimpulan

Berdasarkan perancangan, implementasi dan hasil pengujian dari studi perbandingan kinerja SOAP *web service* dengan REST *web service* dalam pertukaran data pada *platform* Android dan Windows Phone, maka didapatkan kesimpulan sebagai berikut:

1. Studi perbandingan kinerja dengan pendekatan *native mobile application development* lingkungan Android menggunakan REST *web service* lebih baik dalam segi kecepatan akses pengambilan data dibandingkan dengan SOAP *web service*.
2. Studi perbandingan kinerja dengan pendekatan *native mobile application development* lingkungan Windows Phone dengan menggunakan REST *web service* lebih baik dalam segi kecepatan akses pengambilan data dibandingkan dengan SOAP *web service*.
3. Baris *line of code* secara keseluruhan dengan menggunakan REST *web service* dan SOAP *web service* pada lingkungan Android dan lingkungan Windows Phone tidak mempengaruhi kinerja dalam pengambilan data *client-server*.
4. Berdasarkan pada komparasi studi perbandingan pada lingkungan Android dan lingkungan Windows Phone diketahui bahwa REST *web service* dalam lingkungan Android mempunyai rata-rata kecepatan akses pengambilan data sebesar **0.4191** *second* dan 3.55x lebih cepat dibandingkan dengan REST *web service* dengan rata-rata kecepatan akses pengambilan data sebesar **1.49035** *second* dalam lingkungan Windows Phone.

7.2. Saran

Studi perbandingan kinerja SOAP *web service* dengan REST *web service* dalam pengambilan data pada *platform* Android dan Windows Phone dapat dikembangkan lebih baik lagi dengan beberapa saran diantaranya sebagai berikut:

1. Data pada penelitian studi perbandingan dibatasi 3 bulan, untuk penelitian selanjutnya diharapkan dapat menggunakan data *real-time*.
2. Aplikasi dapat dikembangkan lebih lanjut dengan menggunakan server publik/hosting sehingga dapat mengetahui bagaimana kecepatan akses pengambilan data jika dipengaruhi oleh jaringan.

DAFTAR PUSTAKA

Al-Fedaghi, S. (2011). Developing web application. *International journal of software engineering and its applications*, 12.

book, w. (2015). *http://www.withoutbook.com/DifferenceBetweenSubjects.php?subId1=73&subId2=37&d=Difference%20between%20JSON%20and%20XML*. Retrieved from http://www.withoutbook.com/DifferenceBetweenSubjects.php?subId1=73&subId2=37&d=Difference%20between%20JSON%20and%20XML: http://www.withoutbook.com/

Cholli, B. V. (2012). application development on windows phone 7. *IJCST*, 3.

Dudhe, A. (2014). performance analysis of soap and restful mobile web services in cloud environment. *IJCA*, 4.

Ebert, C. (2006). SOAP and Web services. In *Open Source* (p. 6). IEEE.

Geofisika, B. M. (2013). Retrieved from http://karangploso.jatim.bmkg.go.id/index.php/tentang-kami#axzz3rRpen4bg.

Ghifary, M. (2011). Pemodelan dan Implementasi Antarmuka Web Services Sistem Informasi UNPAR.

Gronli, T.-M. (2014). mobile application platform heterogeneity: android vs windows phone vs iOS vs firefox os. 7.

Hidayat, N. F. (2012). the development of mobile client application in yogyakarta tourism and culinary information system based on social media integration. *IJACSA*, 5.

Jeffery, J. (2010, october). *windows 7 phone, silverlight, and sharepoint*. Retrieved from Joel's sharepoint architect blog: http://joelblogs.co.uk/2010/10/04/windows-phone-7-in-7-introducing-windows-phone-7/

Kasaedja, B. A. (2014). rancang bangun web service perpustakaan universitas sam ratulangi. *e-journal teknik elektro dan komputer*, 13.

Ladan, M. I. (2011). Web Services Metrics: A survey and A Classification. *International Conference on Network and Electronics Engineering*, 6.

Ledlie, J. (2004). open problems in data collection network. 6.

Lee, S.-H. (2014). a study on web service analysis and bio-information based web service security mechanism. *International journal of security and its applications*, 10.

Luthfillah, I. (2014). rancang bangun restful web service untuk optimalisasi kecepatan akses studi kasus aplikasi sistem pakar berbasis web. 7.

Meier, R. (2009). *android application development.* wrox.

Morena, T. T. (2013). short message service based applications in the gsm network. *IEEE*, 4.

Padiya, S. M. (2013). Web Services based on SOAP and REST Principles. *International Journal of Scientific and Research Publications*, 4.

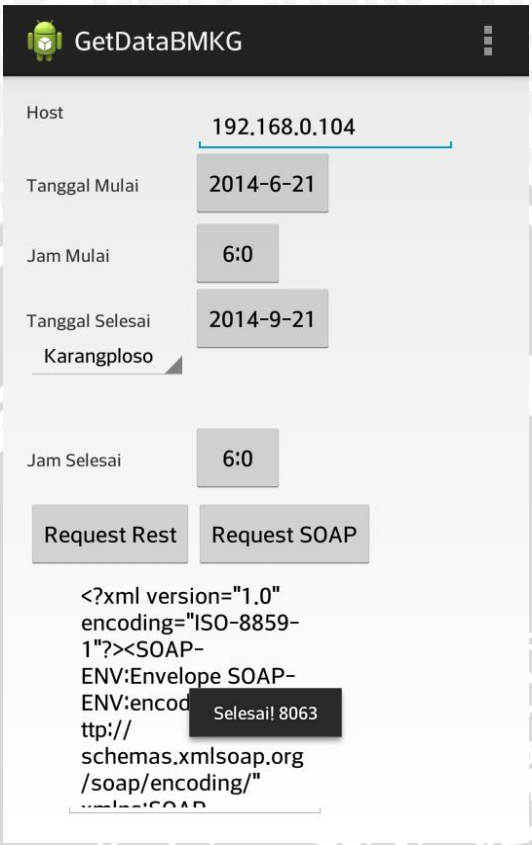
Pressman, R. S. (2001). *Software Engineering.*
Priya, S. (2015). human resource management system using spring restful web service . *international journal of engineering research online*, 5.
Puspita, E. (2012). rancang bangun sistem pencarian hp menggunakan teknologi web service dengan library Nusoap.
Ramanathan, R. (2014). software service architecture to access weather data using restful web services. *IEEE*, 8.
Reddy, V. G. (2013). cloud application programming interface based on rest framework. *international journal of engineering research and technology*, 5.
Sharma, P. G. (2014). architecture for mobile quiz application using android application framework. *IIJCS*, 5.
Tun, P. M. (2014). chossing a mobile application development approach. *ASEAN journal of management and inovation*, 6.
Utama, Y. (2012). Pengenalan web service. 4.
Wagh, K. (2012). a comparative study of soap vs rest web service provising techniques for mobile host. *Journal of information engineering and applications*, 6.
Widhyaestoeti, D. (2012). analysis-algorithm.
Zhang, Y. (2010). heterogeneous networking: a new survivability paradigm. 6.

LAMPIRAN

Lampiran 1 Contoh Tampilan Pengujian Kinerja kurun 1 Bulan SOAP *Web Service* pada Lingkungan Android.

Lampiran 2 Contoh Tampilan Pengujian Kinerja kurun 2 Bulan SOAP *Web Service* pada Lingkungan Android.

Lampiran 3 Contoh Tampilan Pengujian Kinerja kurun 3 Bulan SOAP *Web Service* pada Lingkungan Android.



Lampiran 4 Contoh Tampilan Pengujian Kinerja kurun 1 Bulan REST *Web Service* pada Lingkungan Android.

Lampiran 5 Contoh Tampilan Pengujian Kinerja kurun 2 Bulan REST *Web Service* pada Lingkungan Android.

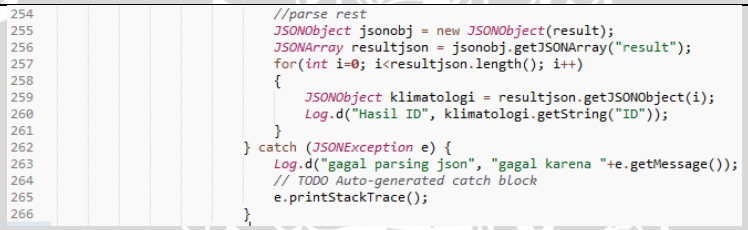
Lampiran 6 Contoh Tampilan Pengujian Kinerja kurun 3 Bulan REST *Web Service* pada Lingkungan Android.

Lampiran 7 Perbandingan *Line of Code* SOAP dan REST Web Service pada Lingkungan Android.

Jumlah Line	Detil perbandingan *line of code Setup* SOAP dan REST *web service* pada Lingkungan Android
15 line	74 //setup soap 75 String resultdata = "[]"; 76 String SOAP_ACTION = "http://"+texthost.getText()+"/nusoap/SOAP.php/search"; 77 String METHOD_NAME = "search"; 78 String NAMESPACE = "http://"+texthost.getText()+"/nusoap/SOAP.php"; 79 String URL = "http://"+texthost.getText()+"/nusoap/SOAP.php"; 80 81 SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME); 82 request.addProperty("kota", selectedwilayah); 83 request.addProperty("date1", buttonTanggalMulai.getText()); 84 request.addProperty("time1", buttonJamMulai.getText()); 85 request.addProperty("date2", buttonTanggalSelesai.getText()); 86 request.addProperty("time2", buttonJamSelesai.getText()); 87 88 SoapSerializationEnvelope envelope = 89 new SoapSerializationEnvelope(SoapEnvelope.VER11); 90 envelope.dotNet = true; 91 envelope.setOutputSoapObject(request);
9 line	234 //setup rest 235 AsyncHttpClient client = new AsyncHttpClient(); 236 RequestParams params = new RequestParams(); 237 params.put("date1", buttonTanggalMulai.getText()); 238 params.put("date2", buttonTanggalSelesai.getText()); 239 params.put("time1", buttonJamMulai.getText()); 240 params.put("time2", buttonJamSelesai.getText()); 241 params.put("kota", selectedwilayah); 242 startingtime = System.currentTimeMillis(); 243 Toast.makeText(getApplicationContext(), "Request mulai, silahkan tunggu! ", Toast.LENGTH_LONG).show();
Jumlah Line	Detil perbandingan *line of code Request* SOAP dan REST *web service* pada Lingkungan Android
4 line	92 //request soap 93 HttpTransportSE androidHttpTransport = new HttpTransportSE(URL); 94 try { 95 androidHttpTransport.debug = true; 96 androidHttpTransport.call(SOAP_ACTION, envelope);
1 line	244 //request rest 245 client.post("http://"+texthost.getText()+"/servicejson/n-klimatologi.php", params, new AsyncH
Jumlah Line	Detil perbandingan *line of code Receive* SOAP dan REST *web service* pada Lingkungan Android

19 line	97 //receive soap
	98 resultdata = androidHttpTransport.responseDump;
	99 } catch (HttpResponseException e) {
	100 // TODO Auto-generated catch block
	101 Log.d("dwihardy", "HttpResponseException");
	102 e.printStackTrace();
	103 } catch (IOException e) {
	104 // TODO Auto-generated catch block
	105 Log.d("dwihardy", "IOException");
	106 e.printStackTrace();
	107 } catch (XmlPullParserException e) {
	108 // TODO Auto-generated catch block
	109 Log.d("dwihardy", "XmlPullParserException");
	110 e.printStackTrace();
	111 } catch (Exception e)
	112 {
	113 Log.d("dwihardy", "Exception : "+e.toString());
	114 }
	115 return resultdata;
	116 }

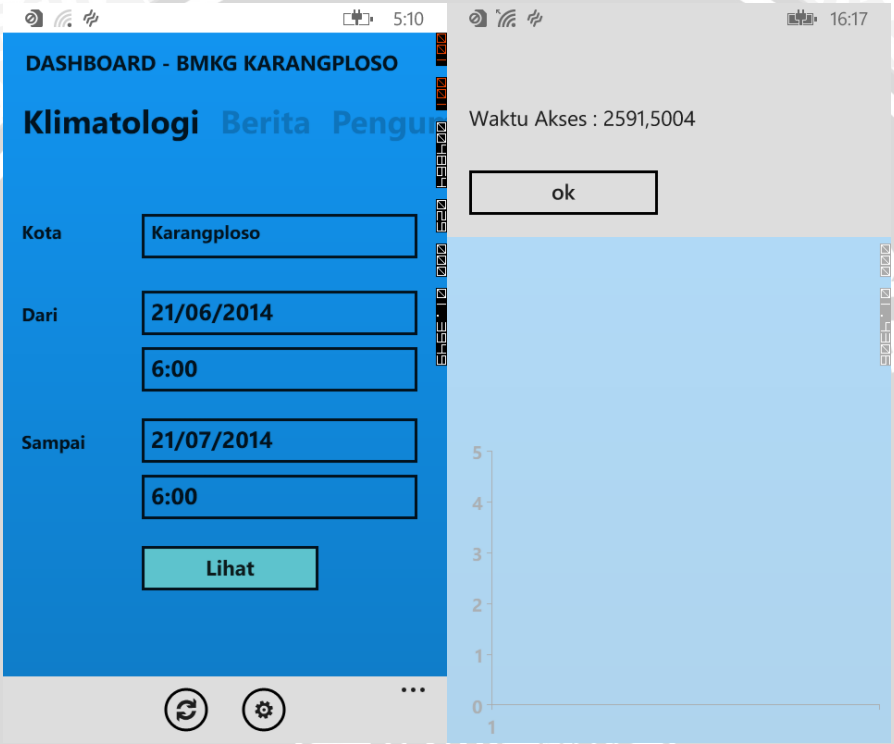
5 Line	249 Toast.makeText(getApplicationContext(), "Selesai! "+(System.currentTimeMillis()-startingtime), Toast.LENGTH_LONG).show(); 250 String result = "{\"result\":"+response+"}"; 251 try { 252 //receive rest 253 textHasil.setText(result);

Jumlah Line	Detil perbandingan *line of code Parse* SOAP dan REST *web service* pada Lingkungan Android
40 line	
13 line	

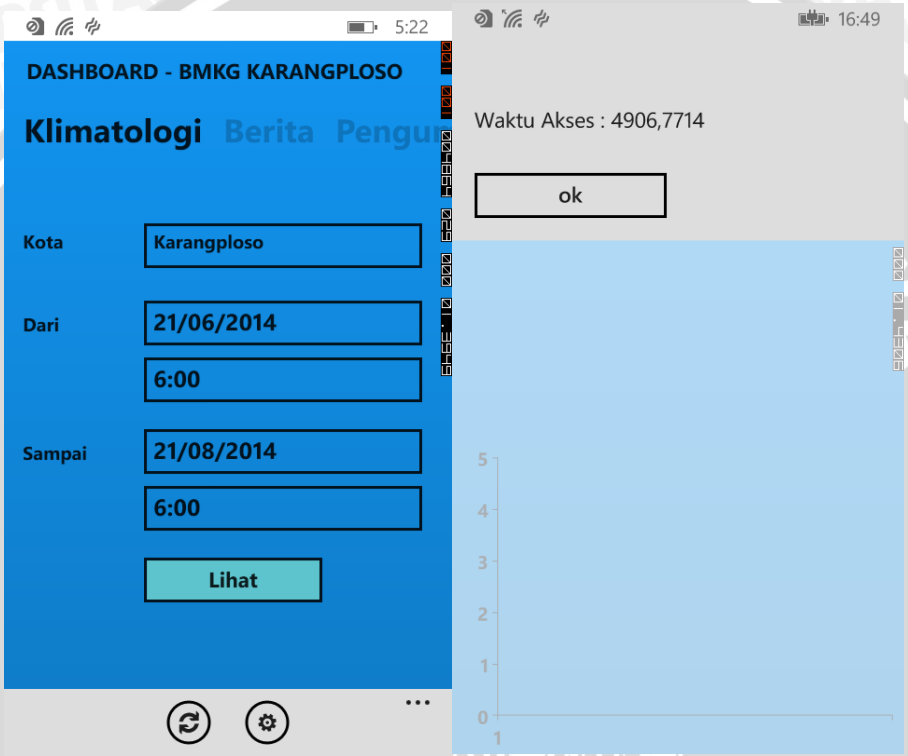
```
protected void onPostExecute(String result) {

    try {
        XmlPullParserFactory xmlfactory = XmlPullParserFactory.newInstance();
        XmlPullParser xmlparser = xmlfactory.newPullParser();
        xmlparser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false);
        InputStream is = new ByteArrayInputStream(result.getBytes());
        xmlparser.setInput(is, null);
        String text = null;
        int event = xmlparser.getEventType();
        while (event != XmlPullParser.END_DOCUMENT)
        {
            String name = xmlparser.getName();
            switch(event)
            {
                case XmlPullParser.START_TAG:
                    break;
                case XmlPullParser.TEXT:
                    text = xmlparser.getText();
                    break;
                case XmlPullParser.END_TAG:
                    if(name.equals("ID"))
                    {
                        Log.d("Text ID", text);
                    }
                    else if(name.equals("KECEPATAN_ANGIN"))
                    {
                        Log.d("Text Kecepatan Angin", text);
                    }
                    break;
            }
            event = xmlparser.next();
        }
    } catch (XmlPullParserException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

```
                    //parse rest
                    JSONObject jsonobj = new JSONObject(result);
                    JSONArray resultjson = jsonobj.getJSONArray("result");
                    for(int i=0; i<resultjson.length(); i++)
                    {
                        JSONObject klimatologi = resultjson.getJSONObject(i);
                        Log.d("Hasil ID", klimatologi.getString("ID"));
                    }
                } catch (JSONException e) {
                    Log.d("gagal parsing json", "gagal karena "+e.getMessage());
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
```

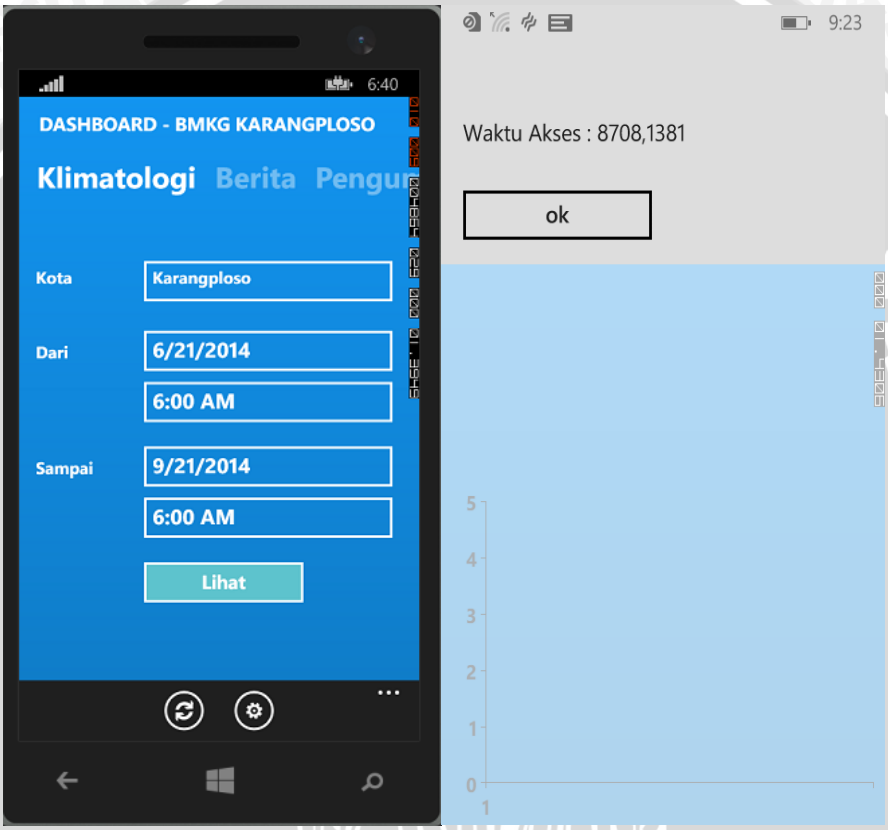
Lampiran 8 Contoh Tampilan Pengujian Kinerja kurun 1 Bulan SOAP *Web Service* pada Lingkungan Windows Phone.



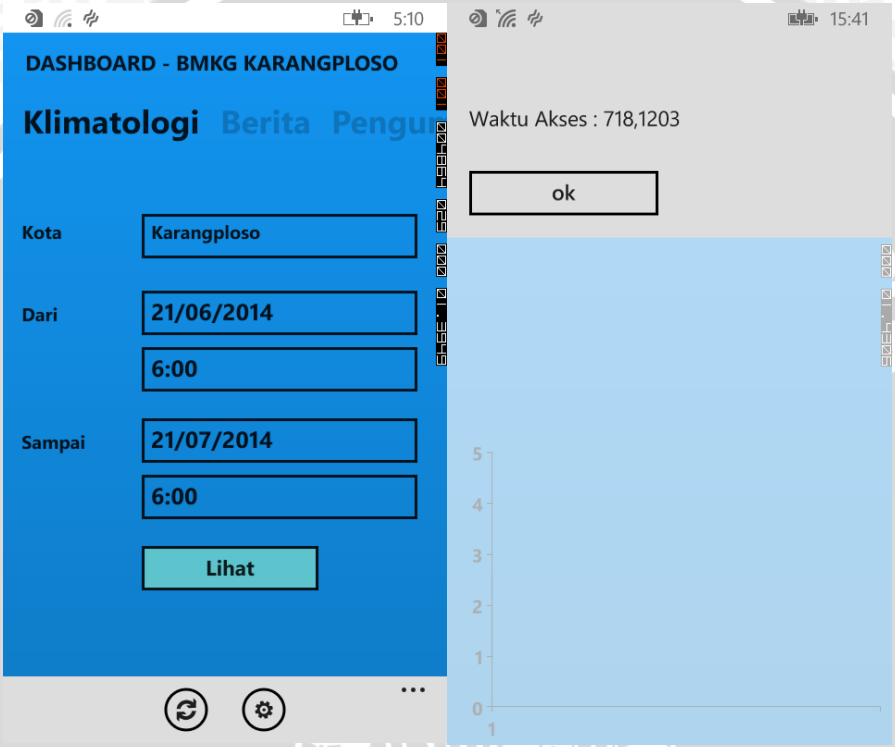
Lampiran 9 Contoh Tampilan Pengujian Kinerja kurun 2 Bulan SOAP *Web Service* pada Lingkungan Windows Phone.



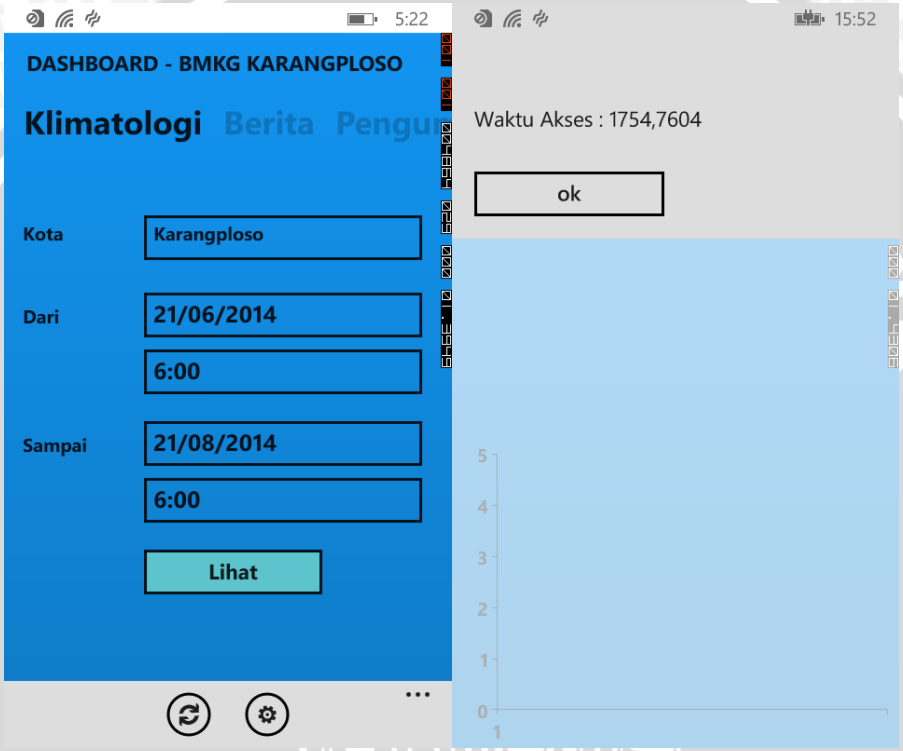
Lampiran 10 Contoh Tampilan Pengujian Kinerja kurun 3 Bulan SOAP *Web Service* pada Lingkungan Windows Phone.



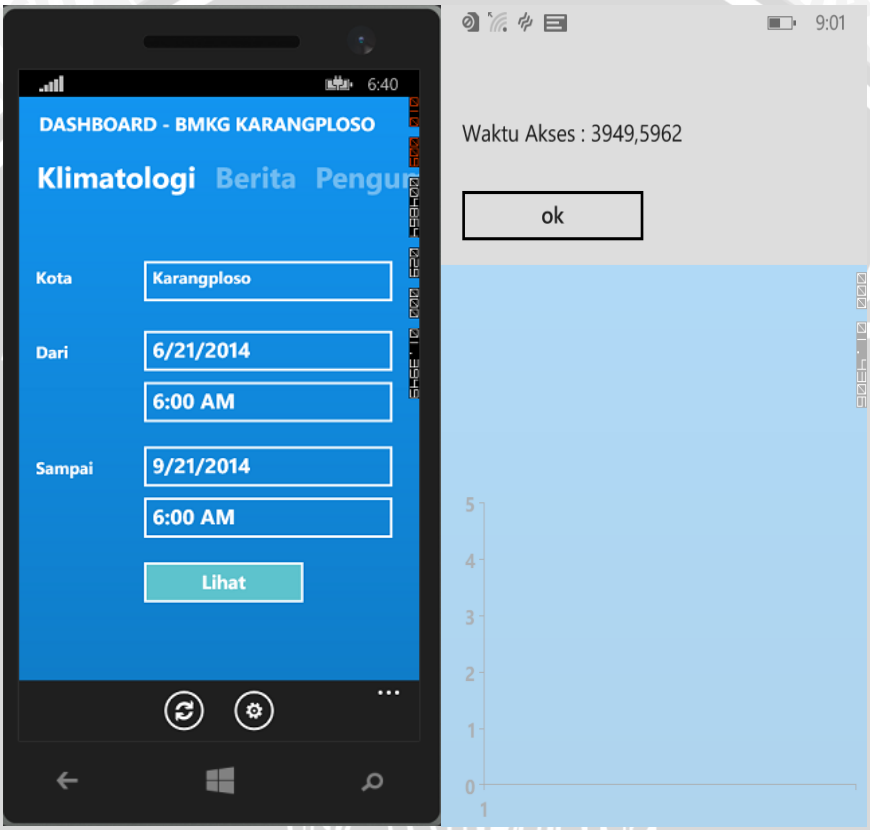
Lampiran 11 Contoh Tampilan Pengujian Kinerja kurun 1 Bulan REST *Web Service* pada Lingkungan Windows Phone.



Lampiran 12 Contoh Tampilan Pengujian Kinerja kurun 2 Bulan REST *Web Service* pada Lingkungan Windows Phone.



Lampiran 13 Contoh Tampilan Pengujian Kinerja kurun 3 Bulan REST *Web Service* pada Lingkungan Windows Phone.



Lampiran 14 Perbandingan *Line of Code* SOAP dan REST *Web Service* pada Lingkungan Windows Phone.

Jumlah Line	Detil perbandingan *line of code Setup* SOAP dan REST *web service* pada Lingkungan Windows Phone

7 line	`//setup`
	`HttpWebRequest spAuthReq = HttpWebRequest.Create("http://localhost/nusoap/SOAP.php"`
	`spAuthReq.Headers["SOAPAction"] = "http://localhost/nusoap/SOAP.php/search";`
	`spAuthReq.ContentType = "text/xml; charset=utf-8";`
	`spAuthReq.Method = "POST";`
	`waktuakses = DateTime.Now;`
	`spAuthReq.BeginGetRequestStream(new AsyncCallback(DoRequestData), spAuthReq);`
	`}`

4 line	`Uri uri = new Uri(ViewModel.Instance.base_url + "n-klimatologi.php");`
	`HttpWebRequest request = HttpWebRequest.Create(uri) as HttpWebRequest`
	`request.Method = "POST";`
	`request.ContentType = "application/x-www-form-urlencoded";`
	`waktuakses = DateTime.Now;`

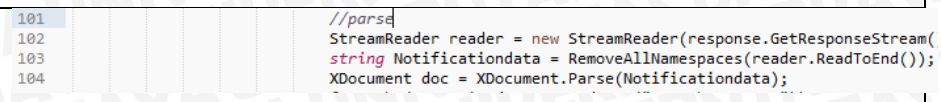
Jumlah Line	Detil perbandingan *line of code Request* SOAP dan REST *web service* pada Lingkungan Windows Phone

26 line	`//request`
	`private void DoRequestData(IAsyncResult asyncResult)`
	`{`
	`string AuthEnvelope =`
	`@"<?xml version=""1.0"" encoding=""utf-8""?>`
	<soap:Envelope xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance"" xmlns:xsd=""http
	<soap:Body>
	<search soapenv:encodingStyle=""http://schemas.xmlsoap.org/soap/encoding/"">
	<kota xsi:type=""xsd:string"">"+this._kota+ @"</kota>
	<date1 xsi:type=""xsd:string"">" + this._tanggalDari + @"</date1>
	<time1 xsi:type=""xsd:string"">" + this._timeDari + @"</time1>
	<date2 xsi:type=""xsd:string"">" + this._tanggalSampai + @"</date2>
	<time2 xsi:type=""xsd:string"">" + this._timeSampai + @"</time2>
	</search>
	</soap:Body>
	</soap:Envelope>";
	`UTF8Encoding encoding = new UTF8Encoding();`
	`HttpWebRequest request = (HttpWebRequest)asyncResult.AsyncState;`
	`System.Diagnostics.Debug.WriteLine("REquest is :" + request.Headers);`
	`Stream _body = request.EndGetRequestStream(asyncResult);`
	`string envelope = string.Format(AuthEnvelope);`
	`System.Diagnostics.Debug.WriteLine("Envelope is :" + envelope);`
	`byte[] formBytes = encoding.GetBytes(envelope);`
	`_body.Write(formBytes, 0, formBytes.Length);`
	`_body.Close();`
	`request.BeginGetResponse(new AsyncCallback(DoRetrieveData), request);`
	`}`

1 line	`//request` `request.BeginGetRequestStream(new AsyncCallback(GetRequestStreamKlimatologiCallback), request);`
Jumlah Line	Detil perbandingan *line of code Receive* SOAP dan REST *web service* pada Lingkungan Windows Phone

7 line	`HttpWebRequest request = (HttpWebRequest)asyncResult.AsyncState;`
	`HttpWebResponse response = (HttpWebResponse)request.EndGetResponse(asyncResult)`
	`if (request != null && response != null)`
	`{`
	`if (response.StatusCode == HttpStatusCode.OK)`
	`{`
	`StreamReader reader = new StreamReader(response.GetResponseStream());`

12 line	`//response`
	`void GetResponseStreamKlimatologiCallback(IAsyncResult callbackResult)`
	`{`
	`Dispatcher.BeginInvoke(() => {`
	`MessageBox.Show("Waktu Akses : " + (DateTime.Now.Subtract(waktuakses).TotalMilliseconds))`
	`});`
	`RemoveDataKlimatologi();`
	`try`
	`{`
	`HttpWebRequest request = callbackResult.AsyncState as HttpWebRequest;`
	`HttpWebResponse response = request.EndGetResponse(callbackResult) as HttpWebResponse;`
	`using (StreamReader httpWebStreamReader = new StreamReader(response.GetResponseStream()))`
	`{`

Jumlah Line	Detil perbandingan *line of code Parse* SOAP dan REST *web service* pada Lingkungan Windows Phone
3 line	
36 line	

```
//parse
StreamReader reader = new StreamReader(response.GetResponseStream(
string Notificationdata = RemoveAllNamespaces(reader.ReadToEnd());
XDocument doc = XDocument.Parse(Notificationdata);
```

```
//parse
string result = httpWebStreamReader.ReadToEnd();

string responseString = "{\"data\":" + result + "}";
JObject jsonData = JObject.Parse(responseString);
List<JToken> klimatologis = jsonData.SelectTokens("data[*]").ToList();
if (klimatologis.Count == 0)
{
    DataKlimatologi.Add(new MCuaca() { Wilayah = "Data not found..", ArahAngin = " -", KecepatanAng
}
else
{
    ObservableCollection<float> curahhujan = new ObservableCollection<float>();
    ObservableCollection<float> suhu = new ObservableCollection<float>();
    ObservableCollection<float> kelembaban = new ObservableCollection<float>();
    ObservableCollection<float> kecepatanAngin = new ObservableCollection<float>();
    int i = 0;
    foreach (var data in klimatologis)
    {
        Dispatcher.BeginInvoke(new Action(delegate
        {
            curahhujan.Add(data.Value<float>("CURAH_HUJAN"));
            suhu.Add(data.Value<float>("SUHU"));
            kelembaban.Add(data.Value<float>("KELEMBABAN"));
            kecepatanAngin.Add(data.Value<float>("KECEPATAN_ANGIN"));
            DataKlimatologi.Add(new MCuaca()
            {
                ID = data.Value<int>("ID"),
                Wilayah = data.Value<string>("WILAYAH"),
                KecepatanAngin = data.Value<float>("KECEPATAN_ANGIN"),
                ArahAngin = ": " + data.Value<string>("ARAH_ANGIN"),
                CurahHujan = data.Value<float>("CURAH_HUJAN"),
                Suhu = data.Value<float>("SUHU"),
                Kelembaban = data.Value<float>("KELEMBABAN"),
                Waktu = data.Value<string>("WAKTU"),
                Jam = data.Value<string>("JAM"),
                DateTime = data.Value<string>("DATE_TIME")
            });
```