Perancangan Dan Pengujian Perangkat Lunak Integrasi Data Jemaat Menggunakan SOA

Rico[*1], Rian Oktavianus[2]
[1,2]Jurusan Teknik Informatika; STMIK Pontianak. Jl. Merdeka No.372 Pontianak, 0561-735555
e-mail: [*1]riicoo.ch@gmail.com, [2]rianoktavianus@stmikpontianak.ac.id

Abstrak

GKKB jeruju Pontianak merupakan organisasi keagamaan yang memiliki bagian yang khusus untuk menangani data jemaat, disana terdapat bagian seketaris dan perlengkapan. Seketaris memiliki aplikasi seketaris yang berbasis web dan perlengkapan memiliki aplikasi berbasis desktop. Aplikasi seketaris menyimpan data keluar masuknya data jemaat termasuk stok sedangkan aplikasi perlengkapan menyimpan data barang yang bermasalah. Ketika melakukan perbaikan perlengkapan tidak bisa melakukan pergantian karena perlengkapan tidak mengetahui stok masih tersisa atau tidak sebab kedua aplikasi tersebut tidak terintegrasi. Salah satu solusi untuk mengatasi masalah ini adalah dengan menggunakan Service Oriented Architecture (SOA) untuk membangun sebuah perangkat lunak yang dapat menjembatani pertukaran data antara kedua aplikasi tersebut. Bentuk penelitian ini menggunakan studi kasus dan metode yang digunakan adalah Action research. Adapun teknik pengumpulan data yang digunakan adalah wawancara, observasi, dan studi dokumentasi. Metode perancangan aplikasi menggunakan Extreme Programming dan alat pemodelan aplikasi yang digunakan adalah UML. Pengujian menggunakan metode white-box. Dengan terintegrasinya kedua aplikasi tersebut, proses kerja pada bagian tata usaha akan semakin cepat dan efektif.

Kata kunci*— Extreme Programmming, Perangkat lunak integrasi, Service Oriented Architecture.*

Abstract

GKKB jeruju Pontianak is a religious organization that has a special section for handling church data, there is a secretariat and equipment. The Secretary has a web based application and equipment has a desktop based application. The Secretary's application has role to record in out of the church data including stock, in other hand the application equitment has role to record all data that have problems which need maintenance. When making repairs, the equipment cannot make changes because equipment do not know whether the stock is ready or not since those applications not integrated. The alternative solution to overcome this problem is using Service Oriented Architecture (SOA) to develop a software to be a middleware that has ability to handle data exchange among those applications. The form of this research is case study and the method is Action research. Techique for collecting data are interview, observation, and study documentation. The method used for developing software is Extreme Programming and the tools for modelling is UML. Software testing using White-box. With integration of those applications it will make the work process more efficient.

Keywords— *Extreme Programmming, integration software, Service Oriented Architecture.*

1. PENDAHULUAN

GKKB jeruju Pontianak terdapat aplikasi seketaris yang berbasis web dan aplikasi perlengkapan yang berbasis desktop. Seksi perlengkapan dalam melakukan perbaikan dan menyusun tempat ibadah akan mengakses aplikasi seketaris untuk melihat jumlah data jemaat dan alat yang perlu pergantian. Sebelum menganti komponen, seksi perlengkapan harus memastikan

komponen tersebut masih tersedia, untuk itu seksi perlengkapan harus bertanya pada seketaris karena fungsi cek ketersedian stock hanya terdapat pada aplikasi seketaris. Jarak antar dua ruangan ini sangat jauh ditambah lagi kedua aplikasi tersebut tidak saling bertukar data menyebabkan penyampain informasi tidak dapat diterima pada saat di butuhkan. Untuk mempercepat akses informasi, pihak GKKB jeruju Pontianak menginginkan kedua aplikasi ini bisa saling terhubung.

Dalam ilmu rekayasa perangkat lunak, menghubungkan dua aplikasi yang berbeda platform agar dapat saling bertukar informasi tidaklah semudah yang dipikirkan oleh kebanyakan orang karena setiap aplikasi adalah produk unik yang dibangun dengan bahasa pemograman, algoritma, dan *programmer* yang berbeda-beda. Menghubungkan dua aplikasi memang dapat dilakukan tetapi tidak melalui meng-*upload* kedua aplikasi tersebut ke suatu jaringan internet seperti kebanyakan orang awam pikirkan melainkan menggunakan sebuah perangkat lunak penengah yang dapat menjembatani pertukaran data antar aplikasi tanpa perlu mengkhawatirkan perbedaan platform, perangkat lunak yang dimaksud adalah *Webservice* .

Webservice merupakan sebuah metode pertukaran data tanpa memperhatikan dimana sebuah *database* ditanamkan, bahasa pemograman yang digunakan, serta dapat menyesuaikan platform yang mendasari sehingga pengkonsumsian data dapat berjalan lancar[1]. *Webservice* mampu menjadi sebuah jembatan penghubung antara berbagai perangkat lunak yang ada karena menggunakan format XML yang telah menjadi standar pertukaran data. *Webservice* yang menggunakan format XML dalam pertukaran data pada dasarnya merupakan representasi dari *Service Oriented Architecture*[2].

Service Oriented Architecture merupakan arsitektur pengembangan perangkat lunak yang mengedepankan pengunaan kembali (*reuse*) komponen-komponen yang sudah ada. Komponen dalam hal ini berupa fungsi dari perangkat lunak yang dibungkus dengan *Webservice* atau disebut juga dengan *service*[3]. *Service Oriented Architecture* memiliki peran sebagai penengah yang memisahkan satu *service* dengan *service* lainnya sehingga tidak ada *dependecy* atau ketergantungan, mengatur komunikasi antar *service* agar dapat digunakan kembali dan mengatur bagaimana untuk memangil sebuah *service* sehingga perangkat lunak dapat terintegrasi[4].

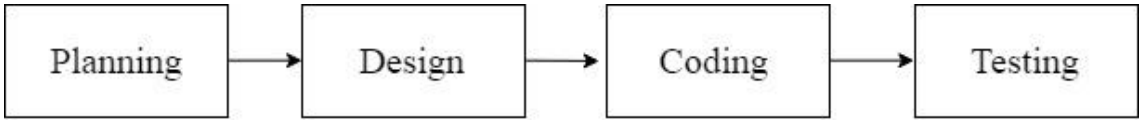
Penelitian tentang integrasi aplikasi telah dilakukan sebelumnya oleh beberapa orang ahli yang telah mendeskripsikan apa saja yang harus diperhatikan dalam merancang perangkat lunak integrasi, yaitu: identifikasi masalah, perencanaan, dan tindakan. Dengan mengikuti langkah diatas maka dihasilkan sebuah perangkat lunak integrasi yang diselesaikan tepat waktu serta dapat digunakan oleh pihak sekolah.

2. METODE PENELITIAN

Bentuk penelitian yang digunakan adalah studi kasus, yaitu salah satu metode penelitian yang bertujuan untuk mencari secara intensif dari latar belakang dan keadaan atau kejadian sekarang terhadap suatu objek yang diteliti yang dijadikan untuk sebuah kasus dengan menggunakan cara-cara yang sistematis yang dilakukan dengan melihat, mengamati apa yang akan dibutuhkan[5]. Metode yang digunakan dalam penelitian ini adalah *Action Research*. Metode dimulai dari melakukan identifikasi masalah, perencanaan dan tindakan. Kegiatan dalam identifikasi masalah berupa mengumpulkan data-data yang bersumber dari seketaris dan perlengkapan melalui observasi kegiatan sehari-hari dua bagian tersebut untuk mengetahui kendala yang sebenarnya terjadi di lapangan, melakukan wawancara kepada seketaris untuk mengkonfirmasi kebenaran dari observasi dan yang terakhir adalah melakukan studi litelatur guna mendapatkan solusi permasalahan ataupun data tambahan.

Kegiatan perencanaan mencakup pemodelan, perancangan dan pembangunan perangkat lunak. Pemodelan perangkat lunak menggunakan *tools* dari UML, yaitu : activity diagram, use case diagram, sequence diagram, dan class diagram. Perancangan dan pembangunan perangkat

lunak menggunakan metode Extreme Programming yang mencakup langkah-langkah : *Planning, Design, Coding,* dan *Testing*[6].



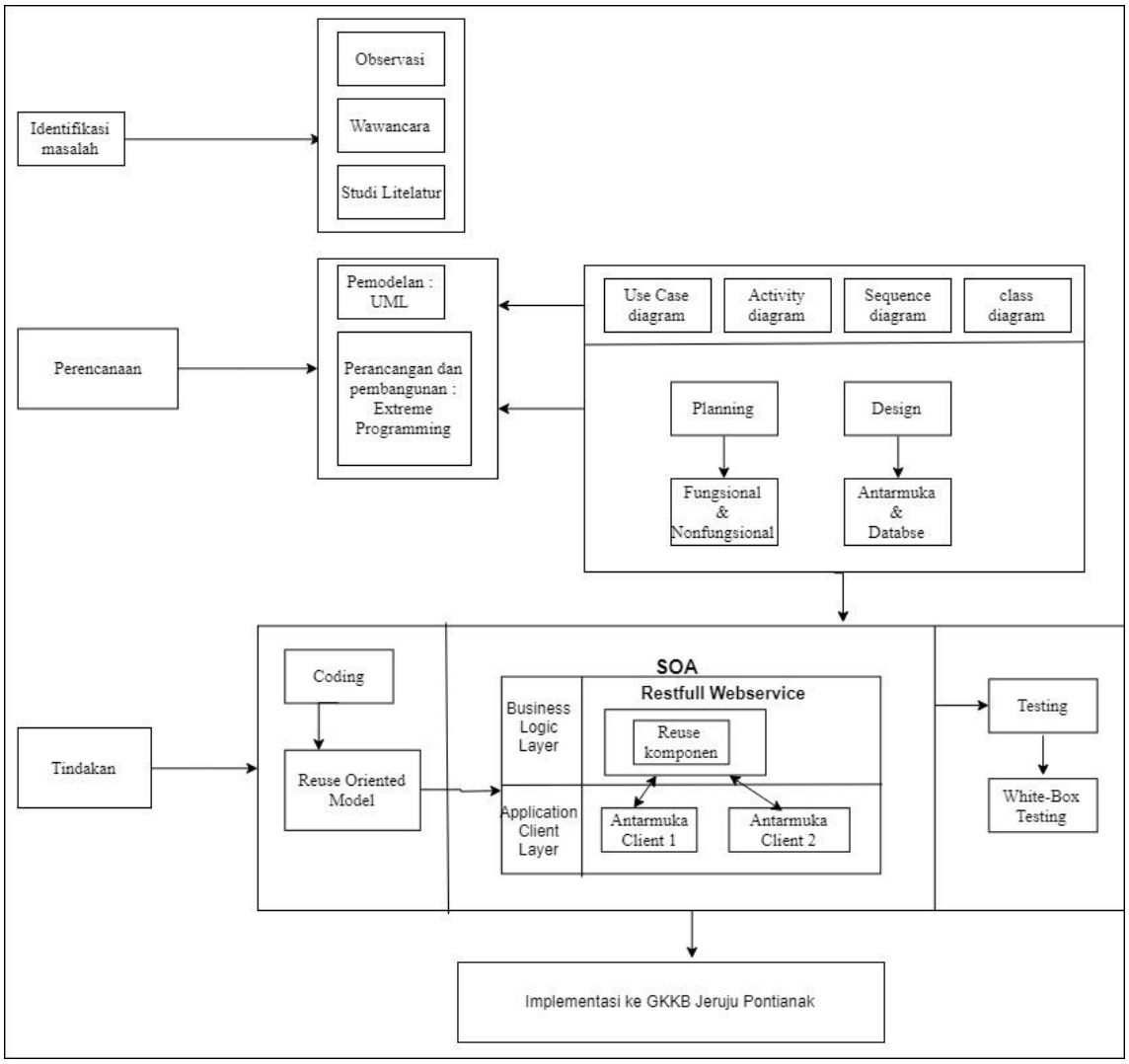
Gambar 1 Model *Extreme Programming*

Pada tahap *Planning*, peneliti menentukan kebutuhan fungsional dan nonfungsional dari perangkat lunak. *Design* merupakan tahapan dimana peneliti menentukan desain awal dari *Database* dan rancangan awal antar muka perangkat lunak yang nantinya akan digunakan dalam tahapan selanjutnya.

Tahapan *Coding* dan *Testing* merupakan bentuk dari tindakan berdasar rencana yang sudah disusun sebelumnya, tahapan ini peneliti akan menggunakan teknik *Reuse Object Model* atau menggunakan komponen(baris kode) yang sudah ada untuk membangun perangkat lunak baru. Salah satu Metode yang dapat digunakan dalam White-Box testing adalah Flow graph, Graph metrix. dan Cyclomatic complexity metric yang dikembangkan oleh McCabe. Cyclomatic complexity metric (V(G)) dapat diekspresikan dengan tiga cara berdasarkan dari program flow graph, yaitu :

$$V(G) = R \text{ dan } V(G) = E - N + 2 \; 3. \; V(G) = P+1 \quad (1)$$

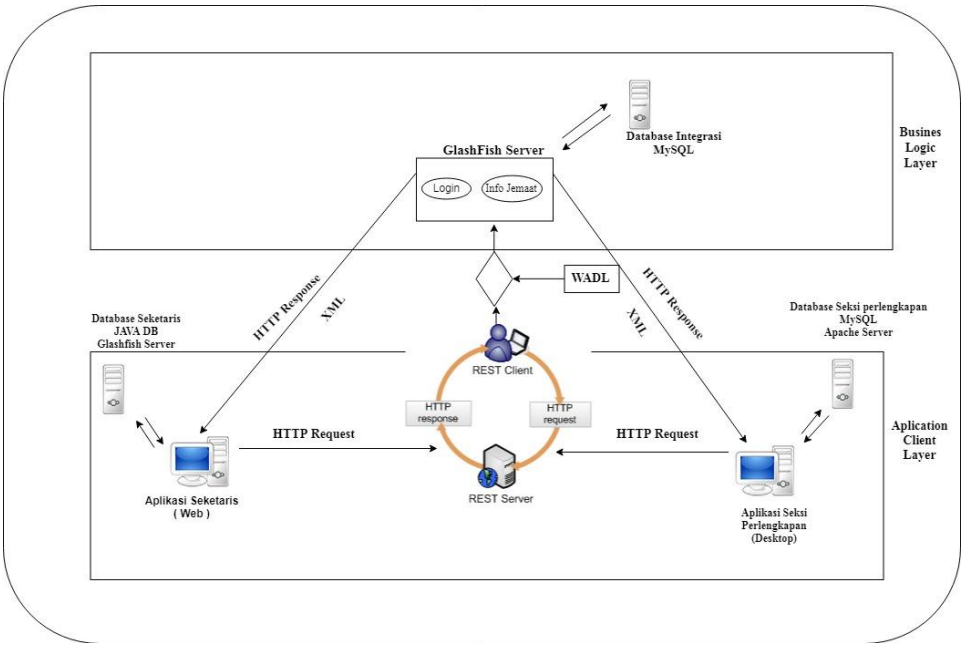
Untuk mendekati model *Service Oriented Architecture* komponen yang sudah dibangun tadi akan dipisah menjadi dua *layer* yaitu : *Business application logic* atau fungsi yang akan dibungkus dengan *Webservice* dan *layer* satunya lagi adalah *Application Client layer* atau antarmuka yang digunakan oleh penguna untuk mengakses *Webservice*. Tahapan terakhir adalah melakukan pengujian terhadap perangkat lunak yang sudah jadi, peneliti menggunakan teknik *White-Box Testing* untuk memastikan *output* logika dari tiap komponen sesuai dengan yang sudah direncanakan. Hal tersebut dapat dilakukan melalui pemeriksaan tiap node dari baris kode dan melakukan perhitungan dengan rumus dari *Cyclomatic complexity*. Alur dari penelitian dapat dilihat pada gambar 2.



Gambar 2 Alur penelitian

3. HASIL DAN PEMBAHASAN

Perangkat lunak integrasi data jemaat yang dihasilkan bermodelkan *client-server*. Pada sisi *Server* terdapat *database* seketaris yang diakses juga oleh aplikasi seketaris dan menggunakan GlassFish Server sebagai *repository Webservice*. Pada sisi *client*, aplikasi perlengakapan memiliki *database* yang sama dengan *server* untuk menyimpan hasil operasi lokal dari aplikasi ini yang berupa tambah, ubah, dan hapus data jemaat. Sedangkan pada aplikasi pengolahan perlengkapan terdapat *database* lokal yang menyimpan hasil operasi lokal dari aplikasi ini yang berupa tambah. Fungsi login dan cek data jemaat ditempatkan pada sisi *Server* sehingga kedua *client*(aplikasi seketaris dan seksi perlengkapan) dapat mengaksesnya secara bersamaan. Proses pengunaan *Webservice* melalui pemangilan *service* melalui HTTP *request* yang berapa url *service*. *Request* HTTP diproses oleh REST yang menghasilkan sebuah file WADL yang merupakan deskripsi dari *service* sebelum benar-benar dikirim ke *Server*. File WADL ini akan diproses oleh *Server* dan hasilnya adalah HTTP *response* berupa format data XML yang dapat dibaca oleh aplikasi pada *client*.



Gambar 3 Model perangkat lunak integrasi

Hal pertama yang dilakukan untuk mengembangkan perangkat lunak integrasi data jemaat sesuai dengan gambaran diatas adalah melakukan analisis *database*. Dikarenakan setiap aplikasi memiliki databasenya masing-masing, yaitu aplikasi dengan *database* Mysql dan seksi perlengkapan menggunakan Derby maka akan dibuat satu buah *database* baru dengan platform Mysql yang akan disimpan di *server* agar dapat diakses bersama. Berikut adalah rancangan awal dari *database* tersebut.

Tabel 1 rancangan awal tabel *database*

Column	Type	Null	Auto Increment
Id	Varchar(10)	Tidak	Ya
User	Varchar(50)	Tidak	Tidak
hak_akses	Varchar(20)	Tidak	Tidak
nama_barang	Varchar(50)	Tidak	Tidak
satuan	Int(3)	Tidak	Tidak
harga_beli	Int(7)	Tidak	Tidak
merek_barang	Varchar(50)	Tidak	Tidak
Stok	Int(3)	Tidak	Tidak
lokasi	Varchar(50)	Tidak	Tidak
barang_lab	Varchar(50)	Tidak	Tidak
keterangan_rusak	Text	Tidak	Tidak
tgl_rusak	Date	Tidak	Tidak
tgl_diperbaiki	Date	Tidak	Tidak

Tabel 2 Hasil normalisasi tabel seketaris

Column	Type	Null
Id	Int(11)	Tidak
nama_jemaat	Varchar(50)	Tidak
Jenis_kelamin	Varchar (50)	Tidak

Alamat	Varchar (100)	Tidak
Komisi	Varchar(100)	Tidak

Tabel 3 hasil normalisasi tabel perlengkapan

Column	Type	Null
id_barang	Varchar(10)	Tidak
Nama_jenis	Varchar(10)	Tidak
Dipakai	Varchar(50)	tidak

Setelah selesai dengan pembuatan *database*, langkah selanjutnya adalah membuat *Class-Responbility-Collabolator Card* guna menunjukan interaksi antar class. Biasanya digambarkan dengan tabel yang mempunyai dua kolom yang berisi tanggung jawab dan kelas yang melakukan tanggung jawab tersebut[7]. Berikut adalah hasil dari rancangan *CRC Card* perangkat lunak integrasi tata usaha sekolah.

Tabel 4 daftar CRC card

Class : Database	
Responsibility	Collabolator
Melakukan koneksi ke *Database Server*	Info Jemaat
Melakukan eksekusi Query	Penguna
	Login

Class : Stok	
Responsibility	Collabolator
Mendapatkan variabel info jemaat	
Memberikan nilai pada variabel info jemaat	

Class : Penguna	
Mendapatkan variabel penguna	
Memberikan nilai pada variabel penguna	

Class : Login	
Mendapatkan variabel penguna	
Melakukan verifikasi penguna	

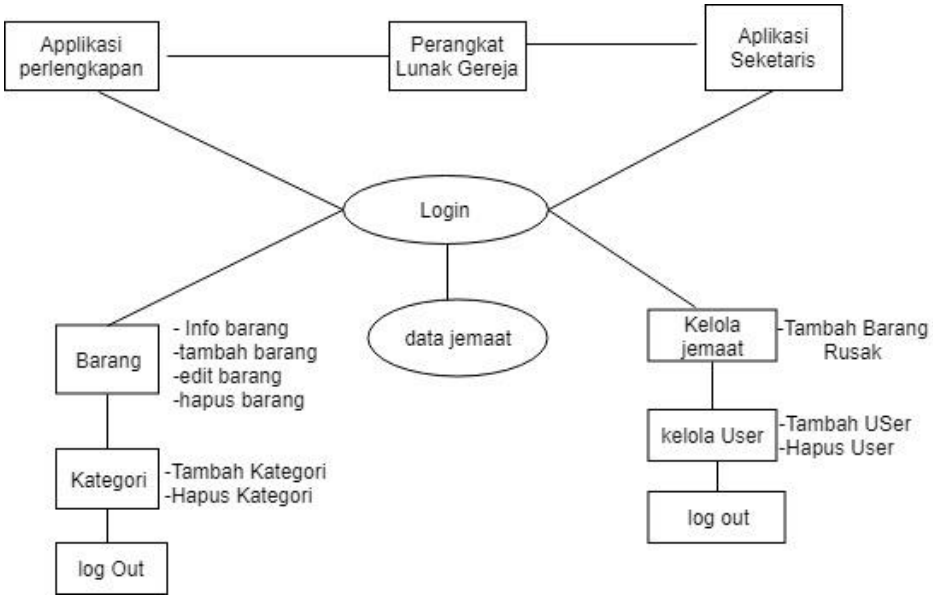
Class : FunctionList	
Responsibility	Collabolator
Menampilkan data jemaat	Info jemaat
Menghapus barang	
Meng-*update* detil barang	

Selain *CRC Card*, penggunaan salah satu tool UML *use case* dapat membantu menunjukan fungsionalitas suatu class yang tidak dijelaskan dalam CRC, selain itu penggunaan *use case* juga menunjukan interaksi antar class dengan aktor[8].

Tabel 5 *Use case* diagram perangkat lunak integrasi data jemaat

Aktor	Seketaris		
	1.	Use Case	Login Seketaris
Deskripsi			1. Seketaris harus melakukan login untuk mengelola data jemaat.
			2.*Username* dan *password* harus dimasukan
			3.Apabila *username* dan *password* sesuai, maka akan masuk ke menu pengelolaan data jemaat.
	2.	Use Case	Kelola data jemaat
Deskripsi			seketaris dapat melakukan tambah, edit, dan hapus data jemaat.
	3.	Use Case	Info data jemaat
Deskripsi			seketaris dapat melihat data jemaat yang statusnya masih aktif.
	4.	Use Case	Kelola user
Deskripsi			Seketaris dapat menambah, edit, dan hapus *user*
Aktor	Perlengkapan		
	1.	Use Case	Login perlengkapan
Deskripsi			1.Perlengkapan harus melakukan login untuk menginput data kerusakan ataupun melakukan *update* status barang
			2.*Username* dan *password* harus dimasukan
			3.Apabila *username* dan *password* sesuai, maka akan masuk ke menu pengolahan data jemaat.
	2.	Use Case	Info stok
			Perlengkapan dapat melihat data barang yang statusnya masih tersedia
	3.	Use Case	Laporan perlengkapan
			perlengkapan dapat menginput data barang yang rusak dan meng-*update* statusnya jika sudah diperbaiki

Dari diagram *use case* dibuatlah gambaran arsitektur perangkat lunak integrasi yang memberikan gambaran umum tentang fungsi dan menu-menu yang ada dalam perangkat lunak.



Gambar 3 arsitektur perangkat lunak

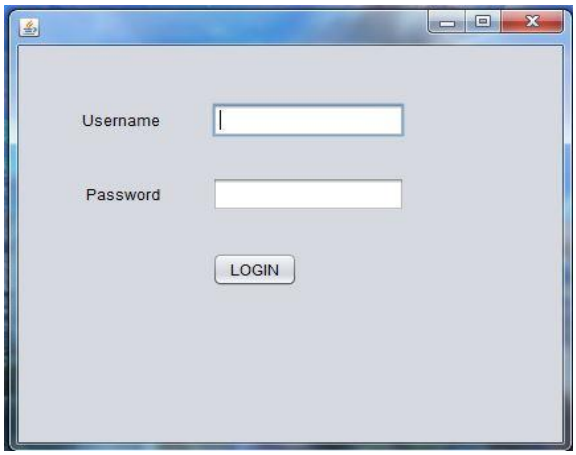
Gambar di atas merupakan arsitektur perangkat lunak integrasi data yang telah dibuat. Masing-masing modul memiliki sub-menu yang memiliki peran sesuai dengan namanya. Pada halaman depan aplikasi seketaris, terdapat 3 modul, yaitu : kelolajemaat, kelola user dan keluar. Pada halaman depan aplikasi seksi perlengkapan terdapat 3 modul, yaitu : barang, kategori dan keluar.

Sesuai penjelasan di atas, setiap aplikasi memiliki menu atau fungsinya masin-masing yang tidak terdapat pada aplikasi yang satu dengan lainnya. Fungsi yang dapakai bersama adalah fungsi login dan cek data jemaat tetapi untuk mengakses fungsi ini client atau aplikasi harus melakukan *request* ke *server* jadi perlu diingat fungsi yang dimaksud tidak dapat diakses jika *server* mati.

Berikut adalah urian cara kerja perangkat lunak integrasi data jemaat :

1. User melakukan login pada aplikasi seketaris atau aplikasi seksi perlengkapan.
2. Aplikasi meminta akses *Webservice.*
3. *Webservice* memproses data dari *client*, jika data cocok maka *user* akan diarahkan ke halaman utama aplikasi.

Client web



Client Desktop(Java)

Gambar 4 *client* yang akan mengakses *Webservice* login

Dibawah ini adalah kode dari *Webservice* login yang digunakan bersama oleh kedua aplikasi

```
    @GET
    @Path("login/{username}/{userPassword}")
    @Produces({"application/xml", "application/json"})
    public Users login(@PathParam("username") String u,
    @PathParam("userPassword") String p) {
        return super.login(u, p);
    }

}
```

Gambar 5 *Webservice* login

Setelah itu terdapat fungsi cek data jemaat yang cara kerjanya adalah sebagai berikut :

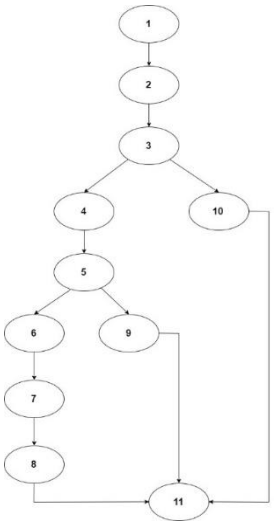
1. Setelah login *user* diarahkan ke halaman utama aplikasi yang memiliki menu cek data jemaat
2. *User* memilih kategori yang inggin dicek datanya, lalu menekan tombol *submit*
3. *Client* melakukan *request Webservice* cek data jemaat
4. *Webservice* memproses data yang yang masuk lalu memberikan nilai balik berupa jumlah data jemaat.

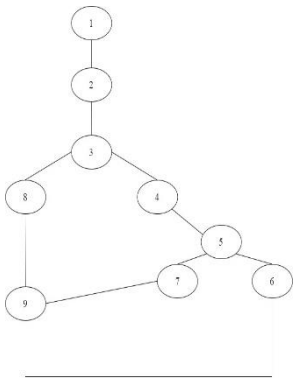
Client web *Client Desktop*(Java)

Gambar 6 *client* yang akan mengakses *Webservice* login

Pengujian perangkat lunak dilakukan menggunakan pengujian *white-box*. Pengujian *white-box* merupakan pengujian terhadap alur logika yang ada di dalam aplikasi tersebut[9]. Kebenaran perangkat lunak yang diuji dilihat berdasarkan hasil perhitungan *Cyclomatic Complexity* dengan jumlah node logika atau if dari aplikasi tersebut. Berikut adalah hasil pengujian dari *Webservice* login dan cek data jemaat.

Tabel x hasil pengujian *white-box* pada *Webservice*

Web service login	
Cyclomatic complexity V(G) = E – N + 2 12 – 11 + 2 = 3 Path 1 = 1,2,3,10,11 Path 2 = 1,2,3,4,5,9,11 Path 3 = 1,2,3,4,6,7,8,11	

Webservice cek data jemaat	
Cyclomatic complexity V(G) = E – N + 2 10 – 9 + 2 = 3 Path 1 = 1,2,3,8,9 Path 2 = 1,2,3,4,5,6,9 Path 3 = 1,2,3,4,5,7,9	

4. KESIMPULAN

Kesimpulan harus mengindikasi secara jelas hasil-hasil yang diperoleh, kelebihan dan kekurangannya, serta kemungkinan pengembangan selanjutnya.

Kesimpulan dapat berupa paragraf, namun sebaiknya berbentuk point-point dengan menggunakan numbering atau bullet.

Berikut adalah beberapa kesimpulan dari hasil perancangan dan pengujian perangkat lunak integrasi data jemaat GKKB jeruju , diantaranya :

a. Perangkat lunak integrasi data jemaat merupakan sebuah perangkat lunak yang menghubungkan aplikasi seketaris dengan aplikasi seksi perlengkapan.
b. Aplikasi yang terhubung mengakses fungsi yang sama, yaitu login dan cek data jemaat di mana fungsi ini tersimpan di *server*.
c. Perangkat lunak ini dapat melakukan operasi *read* pada *database* lalu menampilkan hasil operasi tersebut pada dua aplikasi yang berbeda platform.
d. Kelebihan perangkat lunak ini terletak pada kemampuan *reuseable* untuk fungsi login dan cek data jemaat yaitu ketika terdapat aplikasi baru, *programmer* tidak perlu melakukan koding dari awal untuk membangun fungsi tersebut karena sudah ada *Webservice* yang sudah membungkusnya.
e. Kekurangan perangkat lunak ini adalah tidak ada *Webservice* yang membungkus fungsi *create,update* dan *delete* sehingga operasi tersebut harus di tulis kembali (*non-reuseable*).

5. SARAN

Dari kesimpulan yang telah dikemukakan, beberapa saran yang dapat diterapkan untuk penelitian selanjutnya adalah :

a. Penambahan fitur keamanan pada sistem login, akses database, serta *Webservice* .
b. Penambahan fitur AJAX pada platform website untuk mengambil data dai *database* tanpa reload.

c. Menjadikan fitur lain dalam aplikasi yang dapat digunakan bersama selain login dan cek data jemaat menjadi web services, yaitu fitur create, read, update, dan delete.

DAFTAR PUSTAKA

[1] Lumintang, Yoshiko, 2015, Rancang Bangun Web Service Sistem Informasi Terintegrasi Gereja Masehi Injili Di Minahasa (Studi Kasus : Gereja Gmim Getsemani Lansot Tomohon), E journal Teknik Informatika, Volume 5, No. 1 (2015), ISSN : 2301-8364

[2] Satoto, Iman, 2014, Integrasi Data Kepegawaian Aplikasi Sub Sistem di Universitas Diponegoro Melalui Web *Service*, *Jurnal Teknologi Informasi-Aiti*, Vol. 11. No.1, Februari 2014 : 1 – 109

[3] Rafika, 2010, Desain Interoperabilitas Cross-Aplikasi dengan Service Oriented Architecture: Studi Kasus Lembaga Ilmu Pengetahuan Indonesia(LIPI). Jurnal ITB, Bandung, Nopember 2.

[4] Slameto, Andika, 2015, Penerapan *Service Oriented Architecture* dalam Proses Integrasi Sistem Informasi Inventaris Laboratorium dan Sistem Informasi Laporan Kerusakan Komputer pada Laboratorium STMIK Amikom, *Jurnal Teknologi Informasi*, Yogyakarta, Nopember 30.

[5] Sugiyono, 2008, Metode Penelitian Kuantitatif, Kualitatif dan R&D. Alfabeta, Bandung.

[6] Pressman, R.S., 2012., *Software Engineering : a practitioner's approach 7th edition*, McGraw-Hill, New York.

[7] Pungus, Stenley, 2008, Penerapan Service Oriented Architecture untuk Pengintegrasian Sistem Informasi Perguruan Tinggi. *Jurnal Teknologi* Volume 15 Nomor 5. Desember 2008.

[8] Maulana, Kamal. 2012., Rancang Bangun E-Commerce Template Untuk Aplikasi Content And Management Electronic Mall (Came-Mall), *Journal of Informatics and Technology*, Vol 1, No 2, Tahun 2012

[9] Pare, Selfina. 2013., Desain Dan Implementasi E-Commerce Pada Toko As 88 Celluler Merauke, *Jurnal Ilmiah Mustek Anim Ha* Vol.2 No. 3, Desember 2013 ISSN 2089-6697