BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Penelitian ini dilakukan tidak terlepas dari hasil penelitian-penelitian terdahulu yang pernah dilakukan sebagai bahan perbandingan dan kajian. Adapun hasil-hasil penelitian yang dijadikan perbandingan tidak terlepas dari topik penelitian yaitu mengenai metode *Remote Method Invocation*.

Berdasarkan penelitian yang dilakukan Binarso, dkk (2012) dimana melakukan penelitian mengenai Pembangunan Sistem Informasi Alumni Berbasis *Web* Pada Program Studi Teknik Informatika Universitas Diponegoro. Hasilnya menunjukan bahwa dengan menggunakan sistem informasi alumni berbaasis *web* ini dapat membantu para alumni untuk dapat berinteraksi dengan sesama alumni ataupun dengan pihak program studi sehingga memudahkan alumni dalam memperoleh biodata alumni lain, informasi beasiswa, lowongan pekerjaan, serta berita terkini mengenai perkembangan program studi.

Berdasarkan penelitian yang dilakukan Arifin dan Sutariyani (2014) dimana melakukan penelitian mengenai Aplikasi Pemesanan Menu Makanan Berbasis *Client Server Smartphone Android* Dan Komputer. Hasilnya menunjukan bahwa dengan menggunakan sistem konvensional yang sudah ada terbukti banyak terjadi kesalahan kesalahan yang dapat merugikan pihak *restaurant* tersebut. Dengan dibuatnya Aplikasi Pemesanan Menu Makanan Berbasis *Client Server* Pada *Smartphone*

Android Dan Komputer Berbasis *Windows* maka dapat mempercepat proses pemesanan dan pembayaran. Pihak pengelolapun dapat mengurangi jumlah pengeluaran karena mengurangi jumlah *waiters* yang tersedia karena member dari *restaurant* dapat memesan menu yang disediakan dengan mandiri menggunakan *smartphone.*

Berdasarkan penelitian yang dilakukan Dharmasurya, dkk (2013) dimana melakukan penelitian mengenai Pengembangan Sistem Terdistribusi Untuk Sistem Informasi Administrasi Kependudukan Dengan Integrasi Teknologi RMI Dan *Web Service*. Hasilnya menunjukan bahwa SIAK yang di bangun sudah memiliki standar operasional dan memiliki *user* antarmuka yang mudah dimengerti. Meski secara fisik tidak terlihat perbedaan dengan *web* biasa.

II.2. Aplikasi

Aplikasi adalah Program yang dibuat oleh manusia yang berfungsi untuk menyelesaikan permasalahan-permasalahan masalah yang akan dihadapi. (Zulfauzi ; 2015 : 57).

Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi oleh komputer atau suatu perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut sebagai suatu paket atau suite aplikasi *(application suite).* Aplikasi-aplikasi dalam suatu paket biasanya memiliki antar muka pengguna yang memiliki kesamaan

sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. (Sitohang ; 2013 : 2).

II.3. *Android*

Android merupakan Sistem Operasi perangkat bergerak yang berbasis Linux yang diperuntukan untuk penggunaan ke perangkat bergerak seperti *mobile* dan komputer tablet layar sentuh (*touchscreen*). (Zulfauzi ; 2015 : 58).

II.4. *Remote Method Invocation* (RMI)

Remote Method Invocation (RMI) merupakan produk dari *Sun Microsystem* (Sekarang *Oracle*) dan salah satu teknologi sistem yang terdistribusi yang digunakan pada bahasa pemrograman *java*. RMI mempermudah *developer* untuk merancang aplikasi terdistribusi dimana *method* dari *remote object* dapat dipanggil melalui JVM (*Java Virtual Machine*) lain yang berada pada mesin atau komputer lain. Salah satu perbedaan RMI dengan *client server* biasa adalah RMI memiliki *live multi thread* tersendiri yang akan otomatis tereksekusi ketika aplikasi *server* berjalan dan memiliki *naming registry* sehingga keamanan terjaga. RMI memiliki dukungan paradigm pemrograman berorientasi objek dan memungkinkan aplikasi mengadaptasi teknologi komputasi terdistribusi berorientasi objek dan arsitektur *n-tier*. Dukungan *Distributed Garbage Colecction* pada RMI memungkinkan untuk mengumpulkan *remote object* yang tidak lagi oleh *client* maupun *server* sehingga tidak memberatkan memori sistem. (Dharmasurya, dkk ; 2012 : 86).

II.5. *Client Server*

Client-server merupakan sebuah paradigm dalam teknologi informasi yang merujuk kepada cara untuk mendistribusikan aplikasi ke dalam dua pihak yaitu pihak *client* dan pihak *server*. *Client* menerima instruksi dari pengguna melalui *interface* yang disediakan, merubah format instruksi ke bentuk yang dapat di mengerti oleh *database server*, dan mengirimkannya melalui jaringan ke *server* yang dituju. Sedangkan komponen *server* digunakan untuk pemrosesan, penyimpanan dan manajemen data, serta bertugas untuk menerima *request* dari *client*, mengolahnya dan mengembalikan hasil pemrosesan tersebut kepada *client*. (Binarso, dkk ; 2012 : 38).

II.6. *Java*

Java merupakan bahasa berorientasi objek untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, aplikasi untuk perangkat cerdas yang dapat berkomunikasi lewat internet/jaringan komunikasi. Melalui teknologi *java*, dimungkinkan perangkat *audio streo* dirumah terhubung jaringan komputer. *Java* tidak lagi hanya untuk membuat *applet* yang memperintah halaman *web* tapi *java* telah menjadi bahasa untuk pengembangan aplikasi skala *interprise* berbasis jaringan besar. Dari pengertian diatas maka dapat disimpulkan bahwa *Java* merupakan bahasa pemrograman berorientasi objek yang dapat digunakan untuk membuat dan menjalankan perangkat lunak pada komputer dan berbagai *platform*. (Binarso, dkk ; 2012 : 75).

II.7. MySQL

MySQL adalah suatu sistem manajemen basis data relasional (RDBMS-*Reational Database Management System*) yang mampu bekerja dengan cepat, kokoh dan mudah digunakan. Contoh RDBMS lain adalah *Oracle*, *Sybase*. Basis data memungkinkan anda untuk menyimpan, menelusuri, menurutkan dan mengambil data secara efesien. *Server* MySQL yang akan membantu melakukan fungsionalitas tersebut. (Urva dan Siregar, 2015 : 51).

II.8. *Hypertext Preprocessor* (PHP)

PHP merupakan bahasa *scripting* yang berjalan di sisi *server* (*server-slide*). Semua perintah yang ditulis akan dieksekusi oleh *server* dan hasil jadinya dapat dilihat melalui *browser*. Saat ini PHP versi 4 sudah di-*release* di pasaran. Mengikuti jejak kesuksesan versi sebelumnya, PHP 3. Selain dapat digunakan untuk berbagai sistem operasi, koneksi *database* yang sangat mudah menyebabkan bahasa *scripting* ini digemari para *programmer web*. Beberapa perintah PHP yang dipelajari sebatas pada perintah untuk menampilkan *tag-tag wml*, akses *database* MySQL dan pengiriman *email*. (Urfa dan Siregar, 2015 : 93).

II.9. *Hypertext Markup Language* (HTML)

Hypertext Markup Language (HTML) sebenarnya bukan sebuah bahasa pemrograman, karena HTML adalah bahasa *mark-up*. HTML digunakan untuk *mark-up* (penanda) terhadap suatu dokumen teks. Simbol *mark-up* yang digunakan oleh HTML ditandai dengan tanda lebih kecil (<) dan tanda lebih besar (>). Kedua tanda

ini disebut *tag*. *Tag* yang digunakan sebagai tanda penutup diberi karakter garis miring (</..>). (Binarso, dkk, 2012 : 76).

II.10. *Extensible Markup Language* (XML)

Extensible Markup Language (XML) adalah sebuah bahasa markah untuk mendeskripsikan data. XML merupakan turunan (*subset*) atau versi ringkas dari SGML (*Standart Generalized Markup Language*), sedangkan SGML merupakan sebuah standar ISO untuk format dokumen. SGML tidak berisi berupa *tag-tag* siap pakai seperti halnya bahasa HTML, melainkan berupa aturan-aturan standar dalam pembuatan *tag-tag* format dokumen. SGML banyak dipakai untuk mengelola dokumen dalam jumlah besar, frekuensi revisi tinggi dan dibutuhkan dalam beragam format tampilan. SGML jarang dipakai karena sangat rumit dan kompleks. XML dibuat dengan konsep yang lebih sederhana dan ringkas, tujuannya agar bisa dipakai sebagai aplikasi *desktop* dan jaringan internet. (Yenni dan Shamir, 2012 : 107).

II.11. *Netbeans*

Netbeans merupakan salah satu IDE yang dikembangkan dengan pemrograman *java*. *Netbeans* mempunyai lingkup pemrograman *java* terintegrasi dalam suatu perangkat lunak yang di dalamnya menyediakan pembangunan pemrograman *GUI*, *text editor*, *compiler* dan *interpreter*. *Netbeans* adalah sebuah perangkat lunak *open source* sehingga dapat digunakan secara gratis untuk keperluan komersial maupun nonkomersial yang didukung oleh *Sun Micro System*. (Riestiana dan Sukadi, 2014 : 34).

II.12. *Unified Modeling Language* (UML)

Menurut Windu Gata (2013) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

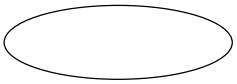
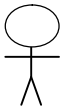
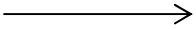
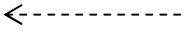
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Urva dan Siregar; 2015 : 93).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

1. *Use case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1 dibawah ini :

Tabel II.1. Simbol *Use Case*

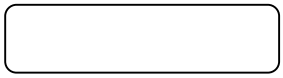
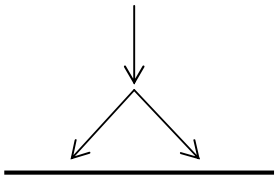
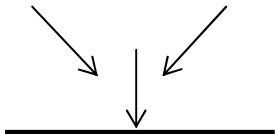
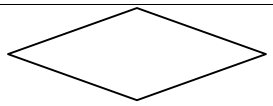
Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, dan dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki *control* terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Urva dan Siregar ; 2015 : 94)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2 dibawah ini :

Tabel II.2. Simbol *Activity Diagram*

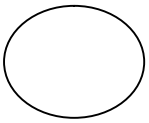
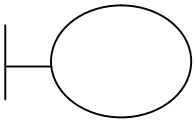
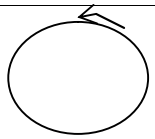
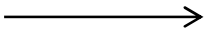
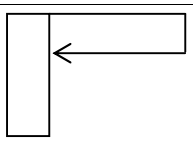
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
	*Swimlane*, untuk menunjukkan siapa melakukan apa.

(Sumber : Urva dan Siregar ; 2015 : 94)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3 dibawah ini :

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.

	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Urva dan Siregar ; 2015 : 95)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti yang dapat dilihat pada tabel II.4 dibawah ini :

Tabel II.4. ***Multiplicity Class Diagram***

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Urva dan Siregar ; 2015 : 95)