RANCANG BANGUN APLIKASI ASISTENSI *ONLINE* LABORATORIUM KOMPUTER DENGAN METODE *CLIENT-SERVER* JAVA RMI API

SKRIPSI

NURFADILLA

H13115006

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2022

RANCANG BANGUN APLIKASI ASISTENSI *ONLINE* LABORATORIUM KOMPUTER DENGAN METODE *CLIENT-SERVER* JAVA RMI API

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains pada program studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin

NURFADILLA

H13115006

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2022

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama : Nurfadilla

NIM : H13115006

Program Studi : Sistem Informasi

Jenjang : S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul:

"Rancang Bangun Aplikasi Asistensi *Online* Laboratorium Komputer dengan Metode *Client-Server* Java RMI API"

Adalah karya tulisan saya sendiri, bukan merupakan pengambil alihan tulisan orang lain dan bahwa skripsi/tesis/disertasi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi/tesis/disertasi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Makassar, 15 November 2022

Nurfadilla

RANCANG BANGUN APLIKASI ASISTENSI *ONLINE* LABORATORIUM KOMPUTER DENGAN METODE *CLIENT-SERVER* JAVA RMI API

Disusun dan diajukan oleh

NURFADILLA
H13115006

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian studi Program Sarjana Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Pada tanggal 15 November 2022
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui

Pembimbing Utama	Pembimbing Pendamping
Dr. Hendra, S.Si, M.Kom.	**Edy Saputra Rusdi, S.Si., M.Si**
NIP.197601022002121001	**NIP.199104102020053001**

Ketua Program Studi

Dr. Hendra, S.Si, M.Kom.
NIP. 197601022002121001

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama	:	Nurfadilla
NIM	:	H13115006
Program Studi	:	Sistem Informasi
Judul Skripsi	:	Rancang Bangun Aplikasi Asistensi *Online* Laboratorium Komputer dengan Metode *Client-Server* Java RMI API

Telah dipertahankan di depan tim penguji dan diterima sebagai bagian persyaratan untuk memperoleh gelar Sarjana Sains pada Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

DEWAN PENGUJI

			Tanda Tangan
1.	Ketua	: Dr. Hendra, S.Si, M.Kom.	(.....)
2.	Sekretaris	: Edy Saputra Rusdi, S.Si., M.Si	(.....)
3.	Anggota	: Drs. Muhammad Hasbi, M.Sc.	(.....)
4.	Anggota	: Rozalina Amran, S.T., M.Eng.	(.....)

Ditetapkan di : Makassar
Tanggal : 15 November 2022

KATA PENGANTAR

Alhamdulillah, Puji dan Syukur kita panjatkan kepada Allah SWT, tak lupa juga shalawat dan salam semoga senantiasa tercurahkan kepada junjungan kita Nabi Muhammad SAW sehingga penulis dapat melaksanakan dan menyusun skripsi yang berjudul "RANCANG BANGUN ASISTENSI *ONLINE* LABORATORIUM KOMPUTER DENGAN METODE *CLIENT-SERVER* JAVA RMI API" dapat diselesaikan guna memenuhi salah satu persyaratan dalam menyelesaikan pendidikan pada program studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Makassar.

Penulis menyadari bahwa dalam proses penulisan skripsi ini banyak mengalami kendala, namun berkat bantuan, bimbingan, kerjasama dari berbagai pihak dan berkah dari Allah SWT sehingga kendala – kendala yang dihadapi tersebut dapat diatasi. Penulis mengucapkan terima kasih kepada kedua orang tua beserta seluruh keluarga yang telah membesarkan dengan penuh cinta dan kasih sayang, memberikan doa, motivasi, semangat, dukungan, dan berjuang hingga penulis mencapai perguruan tinggi.

Disamping itu, penulis juga ingin mengucapkan terima kasih dan penghargaan yang setinggi-tingginya kepada semua pihak yang telah membantu dalam proses penulisan skripsi ini diantaranya adalah:

1. Rektor Universitas Hasanuddin, Bapak Prof. Dr. Ir. Jamaluddin Jompa, M.Sc. beserta jajarannya.
2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Dr. Eng. Amiruddin beserta jajarannya.
3. Ketua Departemen Matematika FMIPA, Prof. Dr. Nurdin, S.Si., M.Si, dan juga Dr. Hendra, S.Si, M.Kom. sebagai ketua Program Studi Sistem Informasi Universitas Hasanuddin.
4. Bapak Dr. Hendra, S.Si, M.Kom. sebagai pembimbing utama yang telah banyak memberikan arahan, ide, motivasi kepada penulis dalam banyak hal.
5. Bapak Edy Saputra R, S.Si., M.Si sebagai pembimbing pertama yang telah memberikan masukan dan arahan kepada penulis.

6. Bapak Drs. Muhammad Hasbi, M.Sc. dan Ibu Rozalina Amran, S.T., M.Eng. sebagai tim penguji atas saran dan masukan pada penelitian yang telah dilakukan oleh penulis.
7. Bapak/Ibu Dosen Departemen Matematika Unhas yang telah memberikan ilmu kepada penulis selama menjadi mahasiswa di Departemen Matematika, dan seluruh Staff Departemen Matematika dan Prodi Sistem Informasi Unhas yang telah membantu penulis dalam urusan berkas administrasi.
8. Teruntuk kedua orang tua saya, terima kasih telah berjuang dan terus mendoakan, dan memberi dukungan sehingga saya dapat menyelesaikan skripsi ini dengan baik.
9. Kepada Koordinator dan Asisten Laboratorium yang telah membantu dalam memberikan data-data yang sangat di perlukan selama penulis melaksanakan penelitian.
10. Teruntuk keluarga, saudara dan ipar – ipar yang selalu mendukung, memberi semangat, dukungan moril, nasehat, doa, dan bisa memberikan kekuatan bagi penulis sehingga berada dititik ini.
11. Teruntuk sahabat dan teman-teman yang selalu mendukung, memberi semangat dan memberi bantuan selama perkuliahan.

Penulis berharap semoga Tuhan Yang Maha Esa mengkaruniakan rahmat dan hidayah-Nya kepada mereka semua. Semoga skripsi ini dapat bermanfaat bagi kita semua, Aamiin.

Makassar,05 September 2022

Nurfadilla

ABSTRAK

Pada masa pandemi COVID-19, kegiatan perkuliahan mengharuskan Mahasiswa belajar secara *daring* sehingga membuat beberapa kegiatan proses belajar mengajar menjadi kurang efektif, Salah satunya ialah matakuliah yang membutuhkan kegiatan laboratorium. Di Program Studi Sistem Informasi misalnya terdapat Mata Kuliah "*Programming*" yang sangat membutuhkan kegiatan laboratorium. Bentuk Kegiatan Lab ini adalah praktikum dan pelatihan yang artinya mengharuskan mahasiswa untuk mempraktikkan langsung agar dapat memahami dan memenuhi tugas praktikum. Pada pelaksanaanya, mahasiswa sebagai "Praktikan" di bimbing langsung oleh seorang "Asisten Praktikum". Untuk memenuhi Tugas Praktikum, Praktikan wajib melakukan Asistensi Minimal dua kali ke Asisten masing-masing sebelum dapat mengumpulkan Tugas Praktikumnya. Melihat sulitnya Kegiatan lab dilakukan secara *offline* di era pandemi saat ini. Oleh Karena itu, dalam penelitian ini penulis akan membuat sebuah "Aplikasi Asistensi *Online*" untuk memudahkan kegiatan asistensi mahasiswa dalam memenuhi tugas praktikum. Aplikasi yang akan dibuat berupa Aplikasi Desktop yang menggunakan metode *client-server* RMI (*Remote Method Invocation*) berbasis Pemrograman Java. RMI adalah salah satu metode dalam sistem terdistribusi yang dapat di implementasikan untuk membuat aplikasi Asistensi *Online*. Java RMI memungkinkan aplikasi untuk memanggil objek yang jauh, dimana objek di komputer yang berbeda dapat berinteraksi dalam jaringan terdistribusi. Hasil yang diharapkan semoga aplikasi ini dapat mempermudah proses kegiatan Laboratorium Komputer Program Studi Sistem Informasi Universitas Hasanuddin khususnya mahasiswa yang sedang sulit-sulitnya melakukan kegiatan asistensi dikampus.

Kata Kunci: Asistensi *Online*, Laboratorium Komputer, RMI, Java, *Client-Server*, Desktop

ABSTRACT

During the COVID-19 pandemic, lecture activities require students to study online, thus making some teaching and learning activities less effective, one of which is courses that require laboratory activities. In the Computer Science Study Program, for example, there is a "Programming" course that really requires laboratory activities. The form of this Lab Activity is practicum and training, which means that it requires students to directly practice in order to understand and fulfill practicum assignments. In its implementation, students as "practitioners" are guided directly by "Practicum Assistants". To fulfill the Practicum Tasks, Practitioners are required to provide assistance at least 2 times to their respective Assistants before being able to collect their Practicum Tasks. Seeing the difficulty of doing offline lab activities in the current pandemic era. Therefore, in this study the author will create an "Online Assistance Application" to facilitate student mentoring activities in fulfilling practical assignments. The application that will be created is a Desktop Application that uses the Java Programming-based client-server RMI (Remote Method Invocation) method. RMI is one of the methods in a distributed system that can be implemented to create Online Assistance applications. Java RMI allows applications to invoke remote objects, where objects on different computers can interact in a distributed network. The expected result is that this application can facilitate the process of Computer Laboratory activities in the Computer Science study program at Hasanuddin University, especially students who have difficulty carrying out mentoring activities on campus.

Keywords: Online Assistant, Computer Laboratory, RMI, Java, Client-Server, Desktop

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Teknologi informasi telah berkembang dengan pesat dan membawa banyak paradigma perubahan dalam aktivitas manusia dalam belajar dan bekerja. Perkembangan teknologi informasi saat ini cenderung mengarah pada sistem terdistribusi (*Client-Server)* dimana fungsi sebuah komputer tidak hanya sebatas pada mesin mandiri, tetapi dapat juga digunakan secara terkoordinasi dengan beberapa komputer lainnya. Khususnya dimasa pandemi *COVID* 19 saat ini, kebutuhan aplikasi semakin dibutuhkan untuk memudahkan kegiatan masyarakat yang bekerja secara *daring* atau *WFH (Work From Home)*. Salah satu sektor yang sangat terhambat karena pandemi ialah sektor pendidikan. Kegiatan perkuliahan mengharuskan mahasiswa belajar secara *daring* sehingga membuat beberapa kegiatan proses belajar mengajar menjadi kurang efektif, Salah satunya ialah mata kuliah yang membutuhkan kegiatan laboratorium.

Di program studi sistem informasi misalnya terdapat mata kuliah "*Programming*" yang sangat membutuhkan kegiatan laboratorium. Bentuk kegiatan lab ini adalah praktikum dan pelatihan yang artinya mengharuskan mahasiswa untuk mempraktikkan langsung agar dapat memahami dan memenuhi tugas praktikum. Pada pelaksanaanya, mahasiswa sebagai "Praktikan" di bimbing langsung oleh seorang "Asisten Praktikum". Untuk memenuhi tugas praktikum, praktikan wajib melakukan asistensi minimal dua kali ke asisten masing-masing sebelum dapat mengumpulkan tugas praktikumnya.

Berikut beberapa penelitian yang dijadikan sebagai referensi atau rujukan. Penelitian pertama dilakukan oleh Muhammad Fauzi Murtadlo pada tahun 2006 berjudul "Pengembangan Aplikasi Sistem Mesin Kasir Terdistribusi Berbasis *Client-Server* dengan Platform Java". Metodologi pengembangan sistem yang digunakan dalam penelitian tersebut adalah metode *waterfall*, yang meliputi pendefinisan masalah, analisis sistem, perancangan sistem, implementasi sistem, uji coba sistem dan perawatan sistem. kesimpulannya aplikasi ini dapat menawarkan berbagai kemudahan dalam pengoperasiannya, kecepatan dan

ketepatan dalam pemrosesanya, keakuratan dalam penghitungan akuntansi transaksi penjualan dan pembelian serta lebih fleksibel tidak tergantung pada platform tertentu (Fauzi, 2006). Selanjutnya penelitian yang dilakukan oleh Adrian Nathaniel Wikana pada tahun 2007 berjudul "Implementasi *Remote Method Invocation (RMI)* untuk Tes *Online* Interaktif *Multiuser* pada *Local Area Network (LAN)*". Aplikasi dibuat dengan bahasa pemrograman java berbasis desktop yang menggunakan jaringan LAN. Hasil yang diperoleh RMI berhasil di implementasikan untuk Tes *Online Multiuser*, beban kerja *Client-Server* menjadi seimbang karena fungsi dari *RMI client* dan *RMI server* (Adrian, 2007). Lalu penelitian selanjutnya dilakukan oleh Sri Lestari pada tahun 2011 berjudul "Implementasi Java RMI pada Rancang Bangun Tes Toefl *Online* Berbasis *Web*". Pada penelitian tersebut di implementasikan RMI untuk membuat aplikasi tes toefl *online* berbasis *web* untuk mengganti sistem yang masih dikelola manual. Hasil penelitian ini membantu lulusan maupun mahasiswa aktif dalam mencari lowongan pekerjaan. (Lestari, 2011). dan terakhir penelitian yang dilakukan oleh Pegel Pangestu pada tahun 2014 berjudul "Implementasi *RMI (Remote Method Invocation)* Pada Aplikasi Kuisioner Fleksibel Berbasis Desktop Dengan Menggunakan Jaringan LAN". Pada penelitian tersebut di implementasikan RMI untuk membuat aplikasi kuisioner fleksibel berbasis java dengan menggunakan jaringan LAN. Penelitian yang menggunakan MySQL sebagai JDBC ini diambil kesimpulan bahwa peneliti berhasil membuat aplikasi yang memudahkan kegiatan kuisioner di kampus universitas muhammadiyah malang (Pangestu, 2014).

Berdasarkan hal diatas dan melihat sulitnya kegiatan lab dilakukan secara *offline* di era pandemi saat ini. Oleh karena itu, dalam penelitian ini penulis akan membuat sebuah "Aplikasi Asistensi *Online*" untuk memudahkan kegiatan asistensi mahasiswa dalam memenuhi tugas praktikum dan menjadi pengganti buku asistensi. Aplikasi yang akan dibuat berupa aplikasi desktop yang menggunakan metode *client-server RMI* berbasis pemrograman java. RMI adalah salah satu metode dalam sistem terdistribusi yang dapat di implementasikan untuk membuat aplikasi asistensi *online*. Java RMI memungkinkan aplikasi untuk memanggil objek yang jauh, dimana objek di komputer yang berbeda dapat berinteraksi dalam jaringan terdistribusi. Hasil yang diharapkan semoga aplikasi

ini dapat mempermudah proses kegiatan laboratorium komputer program studi sistem informasi universitas hasanuddin khususnya mahasiswa yang sedang sulit-sulitnya belajar *programming* di kelas dan sulitnya melakukan kegiatan asistensi dikampus.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang sudah diuraikan diatas, maka perlu melakukan perumusan masalah yaitu sebagai berikut:

1. Bagaimana membangun aplikasi asistensi *online* yang dapat berjalan dengan baik menggunakan metode *client-server* RMI berbasis Java?
2. Bagaimana kinerja aplikasi dalam jaringan terdistribusi?

1.3. Tujuan Penelitian

Adapun tujuan diadakannya penelitian ini yaitu sebagai berikut:

1. Membuat aplikasi asistensi *online* laboratorium komputer dengan menggunakan metode *client-server* RMI berbasis Java.
2. Mengetahui hasil kinerja aplikasi dalam jaringan terdistribusi.

1.4. Batasan Masalah

Batasan masalah dari penelitian ini meliputi:

1. Aplikasi yang dibangun hanya menangani kegiatan asistensi *online* di laboratorium komputer.
2. Aplikasi dibuat berbasis java desktop, sehingga aplikasi perlu disiapkan terlebih dahulu sebelum bisa dijalankan.
3. Aplikasi dapat dikembangkan lagi menjadi aplikasi laboratorium praktikum untuk menangani pengumpulan respon, tugas pendahuluan dan tugas praktikum.

1.5. Manfaat Penelitian

Adapun manfaat yang ingin dicapai dengan melakukan penelitian ini adalah sebagai berikut:

1. Memudahkan asisten dan praktikan dalam menjadwalkan kegiatan asistensi.
2. Memudahkan pengolahan data asistensi.
3. Membantu dan memudahkan kegiatan laboratorium khususnya kegiatan

asistensi menjadi sistematis dan struktural.

1.6. Organisasi Skripsi

Sistematika penulisan skripsi ini adalah sebagai berikut:

BAB I : Pendahuluan

Bab ini membahas mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan dan manfaat penulisan, serta organisasi skripsi.

BAB II : Tinjauan Pustaka

Bab ini membahas mengenai landasan teori, konsep dasar yang mendasari pokok permasalahan dalam tulisan ini.

BAB III : Metodologi Penelitian

Bab ini berisi waktu dan tempat penelitian, tahapan penelitian, rancangan sistem, sumber data, dan instrumen penelitian.

BAB IV : Hasil dan Pembahasan

Bab ini menguraikan tentang perancangan solusi serta implementasi dari masalah-masalah yang telah dianalisis. Pada bagian ini juga akan ditentukan bagaimana sistem dirancang, dibangun, diuji, dan disesuaikan dengan hasil penelitian.

BAB V : Penutup

Bab ini berisi kesimpulan dan saran yang merupakan jawaban yang melatar belakangi masalah pada BAB I serta untuk perbaikan menindak lanjuti hasil penelitian yang nantinya akan berguna bagi pengembangan sistem ini kedepannya.

BAB II
TINJAUAN PUSTAKA

2.1. Landasan Teori

Pada bagian ini akan dijabarkan mengenai konsep dan teori yang digunakan sebagai pondasi dalam penelitian ini.

2.1.1. Asistensi

Arti kata asistensi dalam Kamus Besar Bahasa Indonesia (KBBI) adalah asis.ten.si [n] kegiatan mengasisteni (membantu seseorang dalam tugas profesionalnya) . Sedangkan kegiatan asistensi praktikum merupakan kegiatan yang dilakukan untuk mempersiapkan praktikan sebelum dan sesudah kegiatan praktikum dan pengambilan nilai melalui Aslab (Asisten Laboratorium). Pada laboratorium RPL Komputer Program Studi Sistem Informasi FMIPA Unhas, asisten laboratorium dipilih berdasarkan nilai tertinggi dari tes *coding*. Kebutuhan tenaga asisten laboratorium maksimal sepuluh orang. Setiap satu asisten biasanya menampung maksimal sepuluh praktikan. Hal ini diperlukan karena banyaknya peserta didik yang mengikuti pembelajaran dalam laboratorium dan terbatasnya waktu pembelajaran, sehingga tidak semua peserta didik terakomodasi untuk mendapatkan pemecahan masalah yang mereka hadapi ketika mengikuti pembelajaran dengan seorang pengajar. Oleh karena itu, dibutuhkan kegiatan asistensi diluar dari jadwal pembelajaran dalam laboratorium. Asistensi yang dimaksud dapat juga berupa diskusi dengan asisten lab masing-masing.

Prosedur asistensi yang diutamakan dalam hal ini ialah berupa tugas praktikum yang diberikan setelah kegiatan praktikum laboratorium selesai, asisten laboratorium yang mengajar pada saat itu memberikan tugas praktikum yang harus dikumpulkan sampai jadwal pembelajaran laboratorium berikutnya. Syarat wajib praktikan adalah melakukan asistensi tugas praktikum minimal dua kali sebelum jadwal praktikum laboratorium selanjutnya. Asistensi yang dilakukan berupa diskusi dengan asistensi masing-masing terkait masalah yang dihadapi dalam mengerjakan tugas praktikum. Asisten laboratorium hanya bertugas mengarahkan dan menilai praktikan, bukan membantu mengerjakan

tugas atau memberikan jawaban praktikum sebelum *deadline* kumpul tugas praktikum selesai. Jika praktikan berhasil memenuhi dua kali asistensi barulah asisten laboratorium dapat memberikan nilai praktikum walaupun tugas praktikum belum di selesaikan oleh praktikan. Namun jika kasusnya adalah praktikan berhasil menyelesaikan tugas praktikum tetapi belum pernah melakukan asistensi sama sekali atau hanya melakukan satu kali asistensi maka tetap saja nilai dikali 0 artinya asisten tidak berhak memberikan nilai praktikum bagi praktikan tersebut.

2.1.2. Client-Server

Pada dasarnya *client-server* merupakan konsep arsitektur perangkat lunak atau *software* yang menghubungkan dua objek berupa sistem *client* dan sistem *server* yang saling berkomunikasi melalui jaringan komputer maupun satu komputer yang sama. *Server* akan menyediakan pengelolaan aplikasi, data dan keamanan data *client* (Dicoding Intern, 2020).

Gambar 2.1. *Client-Server*

Pada Gambar 2.1 fungsi pertukaran akses *web*, *client-server* berperan sebagai program *web browser* yang memberikan informasi kepada pengguna atau *user* di seluruh dunia. Hal ini serupa dengan akses email, *database* dan sebagainya yang berkaitan dengan jaringan *browser*. Aplikasi *client-server* membutuhkan laman *web* dan IP *adress* dari *server* khusus. *Client* dapat meminta informasi pada *server* kapanpun ia mau, karena *client* adalah pengguna informasi yang ada di *server*. Proses komunikasi selalu bergerak dua arah, jika *client* ingin menggunakan informasi maka rute yang dituju selalu pada *server*.

Client tidak bisa berkomunikasi kepada sesama *client* (Dicoding Intern, 2020).

Komputer yang meminta layanan (*request*) disebut sebagai *client*, sedangkan yang menyediakan layanan (*response*) disebut sebagai *server*. *Client-Server* memiliki beberapa karakteristik sebagai berikut:

1. *Multitasking* : s*erver* dapat melayani beberapa *client* pada saat yang sama dan mengatur pengaksesan sumberdaya.
2. Adanya pemisahan tier dan fungsi sistem: *client*/*server* memisahkan tier dan layanan menjadi *presentation*, *business logic* dan data *tier*. Dengan pemisahan ini beban komputasi antara *server* dan *client* menjadi seimbang, dan *server* dapat diakses oleh beberapa *client (multiuser).*

Instrumen yang ada pada *client-server* pada dasarnya memiliki fungsi untuk penghematan *bandwidth* serta kinerja yang menggunakan peran *server* sebagai penyimpanan seluruh data yang digunakan oleh *client*. *Software client* hanya akan mendapatkan informasi yang diinginkan begitu *client* mengakses dengan segera. Saat ini perlindungan informasi melalui *server* sudah dapat dienkripsi sehingga memungkinkan *client* mendapatkan data yang aman. Fungsi nyata dari *client-server* adalah seseorang dapat membuat bisnisnya sendiri melalui laman *web* yang dibuat sebelumnya. Sehingga *client* dapat membagikan secara cepat untuk produk atau jasanya kepada *user* di seluruh dunia. *User* akan mendapatkan informasi dengan cepat dari *browser* yang dipakai melalui laman web yang telah dibuat oleh *client* (Dicoding Intern, 2020).

Dalam konteks *database*, *client* membuat sebuah halaman *website* melalui berbagai aplikasi *software* atau *device hardware* dengan memberikan visual halaman yang menarik atau disebut juga *user interface*. *User interface* yang jelas dan menarik akan mempengaruhi jumlah kunjungan para *user* sehingga menjadi elemen penting bagi *client* untuk membuatnya. Proses pengaturan *user interface* tidak lepas terhadap peran *server* khusus yaitu *web server*. *Web server* akan menerima permintaan dan menyimpannya dalam bentuk kode html dengan penyimpanan melalui *workstation*. *Server* tersebut yang nantinya akan memberikan umpan balik secara cepat kepada *client* dalam memberikan

informasi yang di inginkan. Setelah *client* menerima permintaan *user*, selanjutnya *client* akan memeriksa sintaks (bahasa komputer melalui pemrograman) dan menghasilkan *database* yang dibutuhkan dalam bentuk SQL (*Structured Query Language*) atau bahasa lainnya. Proses tersebut akan dilanjutkan ke *server* hingga menunggu *response* yang akan diberikan oleh *server* dalam bentuk sesuai *user* akhir. Setelah *user* merespon, kemudian akan memberikan permintaan *database* kepada *client* untuk ditayangkan (Dicoding Intern, 2020).

2.1.3. Aplikasi

Aplikasi adalah sebuah program siap pakai yang dapat digunakan untuk menjalankan perintah-perintah dari pengguna aplikasi tersebut dengan tujuan mendapatkan hasil yang lebih akurat sesuai dengan tujuan pembuatan aplikasi tersebut, aplikasi salah satu tehnik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputasi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan. Pengertian aplikasi secara umum adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya aplikasi merupakan suatu perangkat komputer yang siap pakai bagi *user*.

Pengertian aplikasi menurut para ahli adalah sebagai berikut (Abdurahman dan Riswaya, 2014):

1. Menurut Jugianto (1999:12) adalah penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses *input* menjadi *output*.
2. Menurut Kamus Besar Bahasa Indonesia (1998:52) adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna.
3. Menurut Rachmad Hakim S, Aplikasi adalah perangkat lunak yang digunakan untuk tujuan tertentu, seperti mengelolah dokumen, mengatur *windows* & permainan (*game*) dan sebagainya.

2.1.4. RMI (*Remote Method Invocation*)

RMI adalah perluasan dari *local method invocation* yang memungkinkan sebuah objek yang hidup dalam satu proses untuk memohon *method* objek yang berada diproses lain. (George Coulouris hal 166). Sistem RMI ini memungkinkan objek yang berjalan di satu JVM (*Java Virtual Machine*) dapat memanggil *method* pada objek yang berjalan di JVM lainnya (Oracle, 2021). RMI didefinisikan sebagai sebuah fasilitas standar java dan merupakan salah satu teknologi sistem terdistribusi yang menangani pemanggilan (*invocation*) suatu metode secara jarak jauh (*remote*) dalam suatu jaringan. Metode yang dipanggil tersebut berada pada *host server* dan dipanggil secara *remote* oleh *host client* pada suatu jaringan. RMI juga disebut sebagai jembatan penghubung antara satu aplikasi dengan aplikasi yang lainnya (Lestari, 2011).

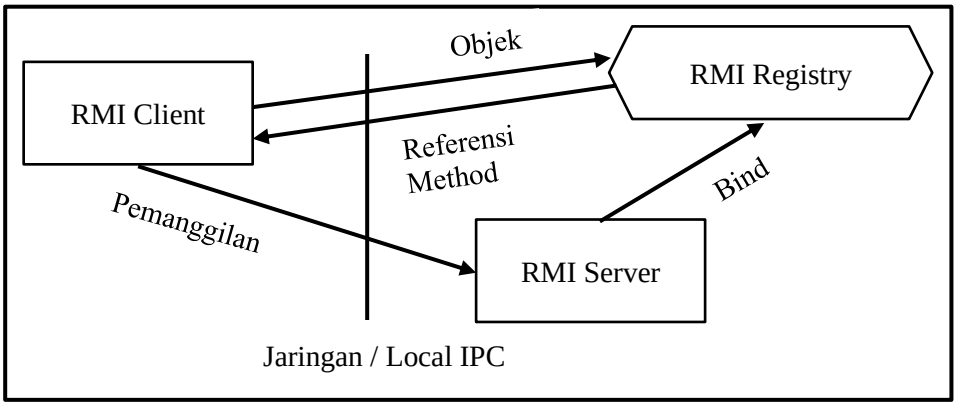
Distribusi Aplikasi RMI terdiri dari dua buah program yang terpisah yaitu program *server* dan program *client*. Pada aplikasi *server* biasanya membuat beberapa objek *remote* yang menyediakan referensi terhadap objek-objek tersebut sehingga dapat diakses, serta menunggu *client* memanggil *method* dari objek-objek *remote* tersebut. Pada aplikasi *client* mendapatkan referensi *remote* ke satu atau lebih objek *remote* di *server* dan menjalankan *method* dari objek tersebut. RMI menyediakan secara timbal balik. Pada aplikasi ini seringkali disebut sebagai aplikasi objek terdistribusi (Wiranti, 2014).

Untuk lebih spesifiknya program *server* bertugas untuk membuat beberapa *remote* objek dan mendaftarkan (*bind*) *remote* objek tersebut ke RMI *Registry* dengan nama yang unik. Program *client* bertugas membuat koneksi ke *server* dan meminta pemanggilan ke *remote* objek berdasar ke referensi yang diterimanya dari RMI *Registry*. RMI *Registry* berfungsi mengkonversi nama objek yang didaftarkan (*bind*) oleh *server* menjadi *remote* objek *reference*. *Remote* objek r*efference* adalah referensi yang berisi *method-method* yang dapat dipanggil secara *remote* oleh *client*. Referensi tersebut dibutuhkan oleh *client* agar dapat mengetahui dan memanggil *remote method* pada *server*. *Method* yang dipanggil secara (*remote method*) punya *remote interface*. *Remote Interface* mendefinisikan *method-method* yang bisa diakses secara *remote*. *Remote*

Interface dapat berupa prosedur atau fungsi (Adrian, 2007).

2.1.4.1. Mekanisme Kerja

Mekanisme kerja yang ada pada sebuah aplikasi RMI dimulai saat RMI *server* akan mendaftarkan *remote* objeknya ke RMI *Registry*. Proses ini disebut dengan *bind*, yaitu mendaftarkan sebuah objek dengan suatu nama yang unik. Dengan terdaftarnya *remote* objek di RMI *Registry* maka RMI *Registry* dapat memberikan referensi ke RMI *Client*, tentang *remote* objek mana yang bisa diakses saat RMI *Client* melakukan pemanggilan *method* (*request*). Setelah referensi diperoleh, RMI *Client* dapat memanggil *method* yang ada di *server*, secara *remote*, RMI *Server* mengirimkan respon setelah dilakukan pemrosesan ke RMI *Client* (Adrian, 2007).



Gambar 2.2. Mekanisme Kerja RMI

Berdasarkan Gambar 2.2, dapat diambil beberapa informasi sebagai berikut:

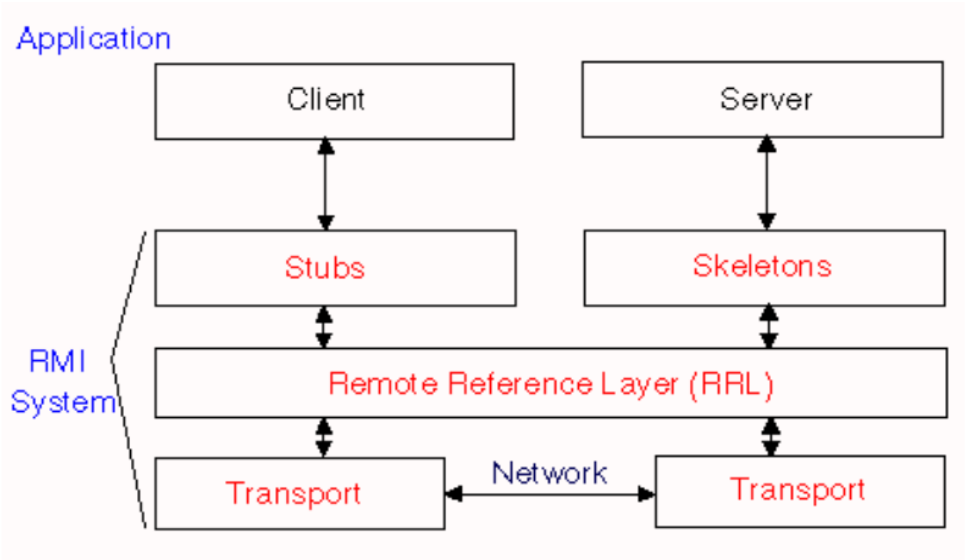
1. Dimulai saat RMI *Server* akan mendaftarkan (*bind*) nama objek ke RMI *Registry* dengan nama yang unik.
2. RMI *Registry* dapat memberikan referensi pemanggilan terpisah objeknya ke RMI *Client* untuk melakukan pemanggilan.
3. RMI *Client* dapat memanggil metode yang berada di *server*.
4. RMI *Server* mengirimkan respon setelah dilakukan pemrosesan ke RMI *Client*.

2.1.4.2. Arsitektur RMI

Sistem RMI dibangun atas tiga lapisan: lapis *stub/skeleton*, lapis *remote reference*, dan lapis *transport*. Tiap lapis dibangun dengan

menggunakan *interface* khusus dan didefinisikan dengan protokol khusus. Hal ini menyebabkan tiap lapis bebas terhadap yang lain dan bisa digantikan dengan implementasi yang lain tanpa mempengaruhi lapis lain dalam sistem.

Berikut ilustrasi Arsitektur Sistem RMI ditunjukkan pada gambar 2.3:

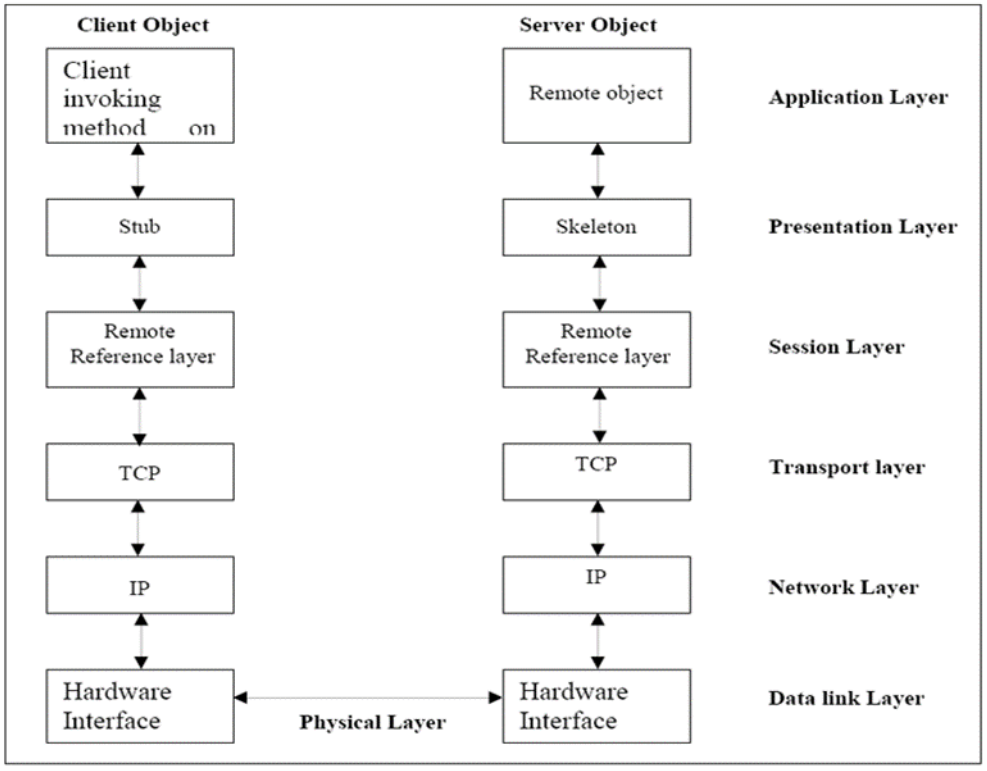


Gambar 2.3. Arsitektur RMI

Pada gambar 2.3. Lapis *stub/skeleton* merupakan *interface* antara lapis aplikasi dan JVM (*Java Virtual Machine*). *Stub* dan *skeleton* ini terbentuk secara otomatis dengan menggunakan *compiler* yang ada di RMI yaitu RMI *compiler* yang didefinisikan oleh *user* dalam *remote interface*. *Client* mereferensi satu *remote* objek melalui *stub* ini. *Stub* akan berlaku sebagai *proxy* yang berlaku sebagai *place holder* bagi *server*. *Client-side stub* untuk suatu *remote* objek bertanggung jawab pada berbagai macam tugas meliputi, inisiasi suatu *remote call*, *marshalling* (mengubah obyek menjadi bentuk yang portabel bagi jaringan) argumen untuk dikirimkan, memberitahukan lapis *remote reference* bahwa suatu *call* harus dipanggil (*invoked*), *unmarshalling return value* (atau *exceptio*n), dan memberitahukan lapis *remote reference* bahwa suatu panggilan telah selesai. *Server-side skeleton*, di lain pihak, bertanggung jawab dalam *unmarshalling* argumen yang masuk dari *client*, memanggil implementasi *remote* objek yang sebenarnya, dan *marshalling return value* (atau *exception*) pada *stream* untuk dikirim kembali ke *client* (Ratnasari, 2003).

Lapis *remote reference* dan lapis *transport* merupakan lapis yang tidak terlihat oleh *user* (*invisible*). Lapis *reference* bertanggung jawab dalam melaksanakan satu protokol *remote reference* khusus. Lapis ini bertanggung jawab untuk mengatur "*liveliness*" dari *remote* objek dan mengatur komunikasi antara *client/server* dengan JVM. Sedangkan lapis *transport* adalah lapis TCP/IP-*based* yang bertanggung jawab dalam mengadakan hubungan antara *server* dan *client* (Ratnasari, 2003).

Pada saat *client* memanggil suatu objek implementasi dari suatu *server*, ketiga lapis dalam RMI akan memerankan perannya masing-masing. Lapis terpenting bagi pemrogram adalah lapis *stub/skeleton*. Pertama-tama suatu pemanggilan akan melewati lapis *stub/skeleton* ini. Fungsi dari lapis ini adalah mengirim data dari lapis *remote reference* melalui *marshal stream.* Di sinilah sebenarnya peranan dari *object serialization*, ini memungkinkan obyek Java untuk dikirimkan ke berbagai ruang alamat yang berbeda. Setelah melewati lapis *stub/skeleton*, data akan melewati lapis *remote reference* yang membawa semantik dari pemanggilan dan mengirimkan data ke lapis *transport* menggunakan *connection-oriented* stream seperti TCP. Ini berarti bahwa lapis *remote reference* adalah menentukan sifat dari objek dan juga menentukan apakah objek berada pada mesin lokal atau mesin *remote* melalui jaringan. Lapis *remote reference* juga akan menentukan apakah bisa di instantiasi dan dimulai secara otomatis, atau harus di deklarasikan dan di inisialisasi terlebih dahulu. Dan terakhir, data akan sampai pada lapis *transport* yang bertanggung jawab dalam menyelenggarakan hubungan dan mengatur hubungan tersebut (Ratnasari, 2003). Arsitektur RMI juga dapat dilihat dalam model OSI, untuk lebih jelasnya perhatikan gambar 2.4:



Gambar 2.4. Arsitektur RMI dalam model OSI

Pada gambar 2.4, RMI digambarkan dengan model OSI (*The Open System Interconnection*), maka gambarnya akan terlihat pada gambar di atas. Model OSI adalah protokol yang dispesifikasi dan diimplementasikan oleh *International Standard Organization* (ISO). Aplikasi pengguna berada pada lapis paling atas. Aplikasi ini menggunakan skema representasi data untuk berkomunikasi secara transparan dengan *remote* objek. Berikut penjelasan dari Gambar 2.4. (Lestari, 2011):

1. Lapis *stub* yang berlaku sebagai berlaku sebagai *proxy* yang memiliki *remote call.*
2. Lapis *skeleton* berada pada sisi *server*, yang bertanggung jawab memanggil implementasi pemanggilan objek yang sesuai bagi jaringan untuk dikirm kembali ke *client*.
3. Lapis *remote refference* meruapakan lapis yang tidak terlihat oleh pengguna yang bertanggung jawab mengatur komunikasi antara *client-server* dengan mesin virtual Java.
4. Lapis *transport* adalah lapis TCP/IP-*based* yang bertanggung

jawab dalam mengadakan hubungan antara *server* dan *client*.

2.1.4.3. Perbandingan RPC dan RMI

Remote Procedure Call (RPC) adalah sebuah metode yang memungkinkan kita untuk mengakses sebuah prosedur yang berada di komputer lain. Untuk dapat melakukan ini sebuah komputer (*server*) harus menyediakan layanan *remote* prosedur. Pendekatan yang dilakukan adalah, sebuah *server* membuka *socket*, menunggu *client* yang meminta prosedur yang disediakan oleh *server*. Bila *client* tidak tahu harus menghubungi *port* yang mana, *client* bisa *merequest* kepada sebuah matchmaker pada sebuah RPC *port* yang tetap. Matchmaker akan memberikan *port* apa yang digunakan oleh prosedur yang diminta *client* (Pyia, 2012).

Sedangkan, RMI adalah sebuah teknik pemanggilan *method remote* yang secara umum lebih baik dari pada RPC. RMI menggunakan paradigma pemrograman berorientasi obyek (*Object Oriented Programming*). RMI memungkinkan kita untuk mengirim obyek sebagai parameter dari *remote method*. Dengan dibolehkannya program java memanggil *method* pada *remote* obyek, RMI membuat pengguna dapat mengembangkan aplikasi java yang terdistribusi pada jaringan (Pyia ,2012).

Berikut beberapa perbandingan RPC dan RMI menurut para pakar *Software Engineer*:

1. Menurut Fortran, RPC adalah Protokol *remote* berbasis C, yang memiliki semantik pemrograman terstruktur, di sisi lain, RMI adalah teknologi berbasis java dan berorientasi objek. Dengan RPC Anda bisa memanggil fungsi jarak jauh yang diekspor ke *server*, di RMI Anda dapat memiliki referensi ke objek jarak jauh dan memanggil metodenya, dan juga meneruskan dan mengembalikan lebih banyak referensi objek jarak jauh yang dapat didistribusikan di antara banyak *instance* JVM, jadi RMI ini jauh lebih baik. RMI sangat bagus digunakan ketika kebutuhan untuk mengembangkan sesuatu yang lebih kompleks daripada arsitektur *client-server* murni. Sangat mudah untuk menyebarkan objek melalui jaringan yang memungkinkan semua *client* untuk berkomunikasi tanpa harus membuat koneksi individu secara

eksplisit (Fortran, Stackoverflow, 2010).

2. Menurut Dhaval Simaria, RMI memiliki pendekatan yang lebih baik dibandingkan dengan RPC, terutama dengan program atau *project* yang lebih besar karena menyediakan kode yang lebih bersih dan lebih mudah untuk di identifikasi jika terjadi kesalahan (Dhaval, Stackoverflow, 2015).
3. Menurut Humprey Bogart, Perbedaan utama antara RPC dan RMI adalah bahwa RMI melibatkan objek. Alih-alih memanggil prosedur dari jarak jauh dengan menggunakan fungsi *proxy*, justru di RMI menggunakan objek *proxy*. Ada transparansi yang lebih besar dengan RMI, yaitu "*exploitation of objects, references, inheritance, polymorphism, and exceptions*" itu semua teknologi yang terintegrasi ke dalam RMI. RMI juga lebih maju dari pada RPC, memungkinkan pemanggilan dinamis, di mana antarmuka dapat berubah saat runtime, dan adaptasi objek, yang menyediakan lapisan abstraksi tambahan (Humprey, Stackoverflow, 2010).

RPC dan RMI keduanya serupa tetapi perbedaan mendasar antara RPC dan RMI adalah RPC mendukung pemrograman prosedural, di sisi lain, RMI mendukung pemrograman berorientasi objek (MKS075, Geekforgeeks, 2022). Untuk lebih jelasnya, perbandinganya dapat dilihat pada tabel 2.1:

Tabel 2.1. Perbandingan RPC dan RMI

NO	RPC	RMI
1	RPC adalah *library* dan *platform* yang bergantung pada OS.	*Platform* Java
2	RPC mendukung Pemrograman Prosedural	RMI mendukung Pemrograman Berorientasi Objek
3	RPC kurang efisien dibandingkan RMI.	Sedangkan RMI lebih efisien dibandingkan RPC.
4	RPC menciptakan lebih banyak *overhead*.	Sementara itu RMI lebih sedikit *overhead* daripada

		RPC.
5	Parameter yang dikirim dalam RPC adalah data biasa atau normal.	Sementara di RMI, objek dikirim sebagai parameter.
6	RPC adalah versi RMI yang lebih lama.	Sementara itu RMI adalah versi penerus dari RPC.
7	Penyediaan kemudahan pemrograman yang tinggi di RPC.	Penyediaan kemudahan pemrograman di RMI rendah.
8	RPC tidak memberikan keamanan apa pun.	Sementara itu RMI memberikan keamanan tingkat *client*.
9	Biaya pengembangannya sangat besar.	Sementara di RMI biaya pengembangannya adil atau masuk akal.
10	Ada masalah besar dalam pembuatan versi di RPC.	Meskipun ada kemungkinan versi menggunakan RDMI.
11	Ada beberapa kode yang diharuskan untuk aplikasi sederhana di RPC.	Sedangkan di RMI ada beberapa kode yang tidak diperlukan untuk aplikasi sederhana di RMI.

2.1.2. Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di *Sun Microsistems* saat ini merupakan bagian dari *Oracle* dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana. *Java* merupakan bahasa pemrograman yang bersifat umum/*non*-spesifik (*general purpose*), dan secara khusus didesain untuk memanfaatkan dependensi implementasi seminimal mungkin. Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi (Udaksana dan Kusaeri, 2018).

Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam *p-code* (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual *Java* (JVM). Java merupakan bahasa pemrograman yang bersifat umum/*non*-spesifik (*general purpose*), dan secara khusus didesain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa *platform* sistem operasi yang berbeda, Kelebihan-kelebihan *java* antara lain (Barri dkk, 2015):

a. *Multiplatform*

Kelebihan utama dari java ialah dapat dijalankan di beberapa *platform*/sistem operasi komputer. Pemrogram cukup menulis sebuah program java dan dikompilasi sekali lalu hasilnya dapat dijalankan di atas beberapa *platform* tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas sistem operasi *Linux* tetapi dijalankan dengan baik di atas *Microsoft Windows*. *Platform* yang didukung sampai saat ini adalah *Microsoft Windows*, *Linux*, *Mac* OS dan *Sun Solaris*.

b. OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek)

OOP artinya semua aspek yang terdapat di java adalah objek. Java merupakan salah satu bahasa pemrograman berbasis objek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut *object*. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis java secara cepat, tepat, mudah dan terorganisir. Kelebihan ini menjadikan java sebagai salah satu 13 bahasa pemograman termudah, bahkan untuk fungsi-fungsi yang *advance* seperti komunikasi antara komputer sekalipun.

c. *Library* Lengkap

Java terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program program yang disertakan dalam pemrograman *java*) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas *java* yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

d. Pengumpulan *spam* otomatis

Java memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung java banyak digunakan sekarang ini karena java merupakan bahasa pemrograman yang baru. Sehingga lebih *update* dibandingkan dengan bahasa pemrograman yang lain.

2.1.3. *MySQL*

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL yang *multi-thread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. *MySQL* adalah implementasi dari manajemen basis data relasional (RDBMS). Pada saat ini *MySQL* merupakan basis data *server* yang sangat terkenal di dunia, semua itu karena bahasa dasar yang digunakan untuk mengakses basis data yaitu SQL (*Structure Query Language*).

Dengan menggunakan SQL, proses pengaksesan basis data lebih *user-friendly* dibandingkan dengan yang lain, misalnya *dBAse* atau *clipper* karena mereka masih menggunakan perintah - perintah pemrograman murni. SQL merupakan bahasa pemrograman yang perlu di pahami karena dapat merelasikan antara beberapa tabel dengan *database* maupun antar *database*. Ada tiga bentuk SQL yang perlu diketahui, yaitu *Data Definition Language* (DDL), *Data Manipulation Language* (DML), dan *Data Control Language* (DCL) (Yasin, 2019).

a. *Data Definition Language* (DDL)

DDL berguna pada saat ingin mendefinisikan data di dalam *database*. Terdapat beberapa *query* yang dikelompokkan ke dalam DDL, yaitu:

CREATE : Digunakan untuk membuat, termasuk di dalamnya membuat *database* baru, tabel baru *view* baru, dan kolom baru.

ALTER : Berfungsi untuk mengubah struktur tabel yang telah dibuat. Mencakup di dalamnya mengubah nama tabel, menambah kolom, mengubah kolom, menghapus kolom, dan memberikan atribut pada kolom.

DROP : Berfungsi untuk menghapus *database* atau tabel.

b. *Data Manipulation Language* (DML)

DML dapat dipakai setelah menjalankan perintah DDL. DML berfungsi untuk memanipulasi, mengubah, atau mengganti isi dari tabel yang sudah ada.

INSERT : Dipakai untuk memasukkan data ke dalam tabel pada *database*.

UPDATE : Dipakai untuk mengubah data yang ada di dalam tabel pada *database.*

DELETE : Dipakai untuk menghapus data di dalam tabel pada *database.*

c. *Data Control Language* (DCL)

Jika sudah mempunyai *user* dan ingin mengatur hak akses masing-masing *user,* pengguna sebaiknya memahami berbagai macam jenis DCL dan cara penggunaannya. DCL berguna untuk memberikan hak akses *database,* mendefinisikan *space,* mengalokasikan *space,* dan melakukan audit penggunaan *database*. Terdapat beberapa perintah DCL yang perlu diketahui, yaitu:

GRANT : Dipakai untuk memberikan izin kepada *user* untuk mengakses *database*.

REVOKE : Dipakai untuk membatalkan izin *user* untuk mengakses *database*.

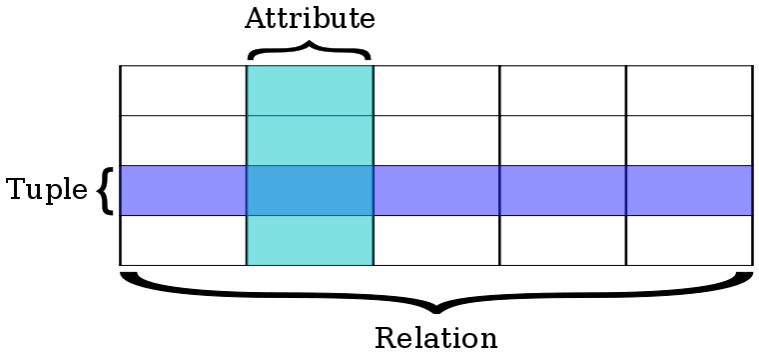
COMMIT : Dipakai untuk menetapkan penyimpanan pada *database*.

ROOLBACK : Dipakai untuk membatalkan penyimpanan pada *database.*

Sebagai suatu pengelola *database* terbesar dan paling banyak digunakan tentunya *MySQL* ini memiliki fitur atau kapabilitas tertentu. Salah satu yang paling dicari oleh para pengguna *MySQL* adalah kemampuannya yang multi-*platform* dan berlisensi GPL, sehingga dapat digunakan oleh komputer hampir di semua OS. Kinerjanya juga dianggap cukup tinggi dalam hal memproses *query – query* yang ada meskipun masih terbatas pada *database* dalam jumlah tertentu. Beberapa fitur lain yang ada pada *MySQL* saat ini yaitu tersedianya tipe data yang sangat beragam seperti *Float, Double, Char, Date* dan lain – lain. *MySQL* juga mendukung penggunaan *field* sebagai *index* serta memiliki tingkat keamanan yang cukup bagus dengan adanya *Subnetmask,* nama *host* serta sandi

yang terenkripsi (Riyadi, 2019).

MySQL menggunakan relational *database* model. Dimana dalam *relational database* model, sebuah *database* adalah kumpulan relasi yang saling terhubung satu sama lainnya. Relasi adalah istilah dalam *relational database*, tapi kita lebih familiar jika menyebutnya sebagai tabel. Selayaknya tabel yang memiliki kolom dan baris, dalam *relational database*, kolom (*column*) disebut *attribute*, sedangkan baris (*row*) disebut *tuple*. Hal ini hanya sekedar penamaan, dan agar lebih gampang, kita hanya akan menggunakan istilah tabel, kolom dan baris dalam tutorial ini, namun jika anda menemui istilah *relation*, *attribut* dan *tuple*, itu hanya penamaan lain dari tabel, kolom, dan baris (Andre, 2017). untuk lebih jelasnya, perhatikan gambar 2.5:



Gambar 2.5. Istilah dalam Relational *Database*

Dari gambar 2.5, dapat diambil informasi berupa peran dari *attribute*, *tuple* dan *relation* dalam sebuah tabel di *database*. *Relation* adalah istilah lain dari tabel pada *database* relasional. lalu *attribute* adalah kolom dari sebuah tabel sedangkan *tuple* adalah baris atau *record* dari sebuah tabel di dalam *database*. *MySQL* dapat digunakan untuk membuat dan mengola *database* beserta isinya. Kita dapat memanfaatkan *MySQL* untuk menambahkan, mengubah dan menghapus data yang berada dalam *database*. *MySQL* merupakan sistem manajemen *database* yang bersifat relasional, artinya data-data yang dikelola dalam *database* akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan menjadi jauh lebih cepat. *MySQL* dapat digunakan untuk mengelola *database* mulai dari yang kecil sampai dengan yang sangat besar. *MySQL* juga dapat menjalankan perintah - perintah *Structured Query Language* (SQL) untuk mengelola *database* yang ada di dalamnya. Hingga kini, *MySQL*

sudah berkembang hingga versi 5. *MySQL* 5 sudah mendukung *trigger* untuk memudahkan pengelolaan tabel dalam *database* (Wiliani dan Zambi, 2017).

2.1.4. Netbeans IDE

Netbeans merupakan sebuah aplikasi *Integrated Development Environment* (IDE) berbasis Java dari *Sun Microsystems* yang berjalan di atas swing dan Javafx. Swing dan Javafx adalah sebuah teknologi Java untuk pengembangan aplikasi dekstop yang dapat berjalan di berbagai macam platform seperti windows, linux, Mac OS X dan juga Solaris. Sebuah *Integrated Development Environment* adalah lingkup pemrograman yang di integrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan *Graphic User Interface* (GUI), yaitu suatu kode editor atau text, suatu *compiler* dan suatu *debugger* (Admin, 2020).

Berdasarkan fungsinya, Netbeans bisa digunakan progammer untuk menulis, *mengcompile*, mencari kesalahan dan menyebarkan program netbeans yang ditulis dalam bahasa pemrograman java namun selain itu netbeans bisa mendukung bahasa pemrograman lainnya dan program ini juga bebas untuk digunakan dan untuk membuat professional dekstop, web, *enterprise*, dan *mobile applications* dengan Java *language*, C/C++, dan bahkan *dynamic languages* seperti PHP, JavaScript, Groovy, dan Ruby.

NetBeans yaitu sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat banyak dan luas, komunitas yang terus bertambah, dan memiliki hampir 100 mitra dan terus bertambah. *Sun Microsystems* mendirikan proyek kode terbuka NetBeans sejak bulan Juni tahun 2000 dan terus menjadi sponsor utama. Dan saat ini netbeans mempunyai dua produk yaitu Platform Netbeans dan Netbeans IDE. Platform Netbeans sendiri adalah *framework* yang bisa digunakan kembali (*reusable*) untuk menyederhanakan pengembangan aplikasi desktop dan Platform NetBeans juga menawarkan layanan-layanan yang umum bagi aplikasi dekstop, mengijinkan pengembang agar fokus ke logika yang spesifik terhadap aplikasi (Admin, 2020).

2.1.5. XAMPP

XAMPP adalah perangkat lunak berbasis web *server* yang bersifat open *source* (bebas), serta mendukung di berbagai sistem operasi, baik Windows, Linux, atau Mac OS. XAMPP digunakan sebagai *standalone server* (berdiri sendiri) atau biasa disebut dengan *localhost* (Adani, 2021).

XAMPP merupakan kumpulan beberapa aplikasi yang digunakan untuk aplikasi *server*, Untuk lebih jelasnya XAMPP tersusun atas kependekan dari beberapa kata berikut ini:

1. X (Cross Platform)

Maksudnya adalah, Xampp dalam dijalankan di berbagai perangkat sistem operasi yang ada, misalnya Windows, Linux, Mac OS, dan Solaris. Dari semua sistem operasi tersebut, software ini bersifat open *source* atau dapat digunakan secara gratis.

2. A (Apache)

Apache merupakan aplikasi web *server* yang bertugas untuk menciptakan halaman *website* yang benar berdasarkan kode program PHP yang ditulis oleh pengembang web (*developer*). Memungkinkan juga untuk mengakses sistem *database* terlebih dahulu untuk mendukung halaman situs yang dihasilkan.

3. M (MySQL / MariaDB)

MySQL merupakan salah satu aplikasi *database server* yang menerapkan bahasa pemrograman SQL. Fungsi dari MySQL sendiri adalah untuk mengelola dan membuat sistem basis data secara terstruktur dan sistematis.

4. P (PHP)

PHP adalah bahasa pemrograman khusus berbasis web untuk kebutuhan pada sisi *server* (*back end*). Sehingga, PHP sangat memungkinkan untuk membuat suatu halaman website menjadi lebih dinamis dengan menerapkan *server-side scripting*. PHP juga mendukung manajemen sistem pada Oracle, Postgresql, Microsoft Access, dan lain sebagainya.

5. P (Perl)

Perl merupakan bahasa pemrograman untuk segala kebutuhan (*cross platform*) yang berfungsi sebagai penunjuk eksistensi dari PHP. Perl biasanya banyak digunakan untuk *website* development pada sistem berbasis CMS

(*Content Management System*) seperti WordPress.

2.1.6. API (*Application Programming Interface*)

Antarmuka pemrograman aplikasi (*Application Programming Interface*/API) adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan oleh *programmer* saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan *programmer* untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi. API dapat menjelaskan cara sebuah tugas (*task*) tertentu dilakukan. Dalam pemrograman *procedural* seperti bahasa C, aksi biasanya dilakukan dengan media pemanggilan fungsi. Karena itu, API biasanya menyertakan penjelasan dari fungsi/rutin yang disediakannya. API menyediakan fungsi dan perintah dengan bahasa yang lebih terstruktur dan lebih mudah untuk dipahami oleh *programmer* bila dibandingkan dengan *Sistem Calls*, hal ini penting untuk aspek *editing* dan pengembangan, sehingga *programer* dapat mengembangkan sistem dengan mudah. API juga dapat digunakan pada Sistem Operasi mana saja asalkan sudah ada paket-paket API nya (Jayanto, 2019).

Menurut M.Ichwan dan Fifin Hakiky *Application Programming Interface* (API) atau Antarmuka Pemrograman Aplikasi adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan oleh *programmer* saat membangun perangkat lunak untuk sistem operasi tertentu. API dikembangkan karena adanya tren industri yang baru, yaitu *distributed system*, untuk menyediakan layanan yang efisien, meningkatkan *reliability* dan *availability*, dan kelebihan lain untuk integrasi sistem (Alvin dan Gusrianty, 2019).

API merupakan *software interface* yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk *library* dan menjelaskan bagaimana agar suatu *software* dapat berinteraksi dengan *software* lain. Penjelasan ini dapat dicontohkan dengan analogi apabila akan dibangun suatu rumah. Dengan menyewa kontraktor yang dapat menangani bagian yang berbeda, pemilik rumah dapat memberikan tugas yang perlu dilakukan oleh kontraktor tanpa harus mengetahui bagaimana cara kontraktor menyelesaikan pekerjaan tersebut. Dari analogi tersebut, rumah merupakan *software* yang akan dibuat, dan kontraktor

merupakan API yang mengerjakan bagian tertentu dari *software* tersebut tanpa harus diketahui bagaimana prosedur dalam melakukan pekerjaan tersebut. (Firdaus dkk, 2019).

2.1.7. Diagram UML (*Unified Modelling Languange*)

UML adalah sekumpulan alat yang digunakan untuk melakukan abstraksi terhadap sebuah sistem atau perangkat lunak berbasis objek. UML merupakan singkatan dari *Unified Modeling Language*. UML juga menjadi salah satu cara untuk mempermudah pengembangan aplikasi yang berkelanjutan. Aplikasi atau sistem yang tidak terdokumentasi biasanya dapat menghambat pengembangan karena *developer* harus melakukan penelusuran dan mempelajari kode program. UML juga dapat menjadi alat bantu untuk *transfer* ilmu tentang sistem atau aplikasi yang akan dikembangkan dari satu *developer* ke *developer* lainya. Tidak hanya antar *developer* terhadap orang bisnis dan siapapun dapat memahami sebuah sistem dengan adanya UML (Fajar, 2016).

UML (*Unified Modeling Language*) merupakan pengganti dari metode analisis berorientasi objek dan desain berorientasi objek (OOAD&D / *object oriented analysis and design*) yang dimunculkan sekitar akhir tahun 80-an dan awal tahun 90-an. UML merupakan gabungan dari metode *Booch, Rumbaugh* (OMT) dan *Jacobson*. Tetapi UML mencakup lebih luas daripada OOAD. Pada pertengahan saat pengembangan UML, dilakukan standarisasi proses dengan OMG (*Object Management Group*) dengan harapan UML bakal menjadi bahasa standar pemodelan pada masa yang akan datang (yang sekarang sudah banyak dipakai oleh berbagai kalangan). Jadi, UML dibuat untuk memudahkan para sistem *developer* untuk berdiskusi dengan bahasa pemodelan yang mudah dipahami.

Jenis – jenis Diagram UML antar lain (Jejaring, 2019):

a. *Use Case* Diagram

   Use case adalah abstraksi dari interaksi antara *sistem* dan aktor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Diagram *use case* berguna dalam tiga hal:

- Menjelaskan fasilitas yang ada (*requirement*).
- Komunikasi dengan klien.
- Membuat test dari kasus-kasus secara umum.

b. *Activity* Diagram

Activity diagram menyediakan analis dengan kemampuan untuk memodelkan proses dalam suatu sistem informasi. *Activity* diagram dapat digunakan untuk alur kerja model, *use case* individual, atau logika keputusan yang terkandung dalam metode individual. *Activity* diagram juga menyediakan pendekatan untuk proses pemodelan paralel.

c. *Package* Diagram

Package diagram utamanya digunakan untuk mengelompokkan elemen diagram UML yang berlainan secara bersama-sama ke dalam tingkat pembangunan yang lebih tinggi yaitu berupa sebuah paket. Diagram paket pada dasarnya adalah diagram kelas yang hanya menampilkan paket, disamping kelas, dan hubungan ketergantungan, disamping hubungan khas yang ditampilkan pada diagram kelas.

d. *Entity Relationship Diagram*

Diagram Hubungan Entitas atau *Entity Relationship* Diagram (ERD), juga dikenal sebagai model hubungan entitas, adalah representasi grafis yang menggambarkan hubungan antara orang, objek, tempat, konsep, atau peristiwa dalam sistem teknologi informasi (TI). ERD menggunakan teknik pemodelan data yang dapat membantu menentukan proses bisnis dan berfungsi sebagai dasar untuk *database* relasional.

e. *Sequence* Diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence* diagram adalah gambaran tahap demi tahap yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case* diagram. Bersifat dinamis. Diagram urutan adalah interaksi yang menekankan pada pengiriman pesan (*message*) dalam suatu waktu tertentu. *Sequence* diagram menekankan penyusunan berbasis waktu untuk kegiatan yang dilakukan dengan satu set dari objek yang berkolaborasi. *Sequence* diagram sangat berguna dalam membantu analis,

memahami spesifikasi *realtime* dan menggunakan kasus yang rumit. Diagram ini dapat digunakan untuk mendeskripsikan baik secara fisik dan logis interaksi antara objek.

f. *Class* Diagram

Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka, kolaborasi - kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Kelas diagram berfungsi untuk menjelaskan tipe dari *object* sistem dan hubungannya dengan *object* yang lain. *Object* adalah nilai tertentu dari setiap *attribute* kelas *entity*. Pada penggambaran kelas diagram ada dikenal dengan kelas analisis yaitu kelas ber-*stereotype*. Tapi yang biasanya dipakai adalah kelas diagram tanpa *stereotype*.

g. *Communication* Diagram

Communication diagram menggambarkan interaksi antar objek seperti *sequence* diagram, tetapi lebih menekankan pada peran masing-masing objek. Setiap *message* memiliki *sequence number*, dimana *message* dari level tertinggi memiliki Nomor 1. Diagram membawa informasi yang sama dengan diagram *Sequence*, tetapi lebih memusatkan atau memfokuskan pada kegiatan obyek dari waktu pesan itu dikirimkan.

h. *Composite Structure* Diagram

Diagram struktur komposit adalah diagram yang menunjukan struktur internal *classifier*, termasuk poin interaksinya ke bagian lain dari sistem. Hal ini menunjukkan konfigurasi dan hubungan bagian, yang bersama-sama melakukan perilaku *classifier*. Diagram struktur komposit merupakan jenis diagram struktur yang statis dalam UML, yang menggambarkan struktur internal kelas dan kolaborasi.

i. *Object* Diagram

Object diagram sangat mirip dengan diagram kelas. Perbedaan utama adalah bahwa diagram objek menggambarkan objek dan hubungan mereka. Tujuan utama dari diagram objek adalah untuk memungkinkan analis untuk mengungkap rincian tambahan kelas. Dalam beberapa kasus, pernyataan variabel dari sebuah *class* diagram dapat membantu pengguna

atau analis dalam menemukan atribut tambahan yang relevan, hubungan, dan atau operasi, atau mungkin menemukan bahwa beberapa atribut, hubungan, atau operasi yang salah tempat. Bersifat statis. Diagram ini mempelihatkan objek - objek serta relasi - relasi antar objek. Diagram objek memperlihatkan instansiasi statis dari segala sesuatu yang dijumpai pada diagram kelas.

j. *Deployment* Diagram

Deployment diagram menggambarkan detail bagaimana komponen di *deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, *server* atau piranti keras), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisikal. Hubungan antar node (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

2.1.8. Metode *Blackbox*

Metode *Blackbox Testing* merupakan sebuah metode yang dipakai untuk menguji sebuah *software* tanpa harus memperhatikan detail *software*. Pada pengujian *blackbox*, estimasi banyaknya data uji dapat dihitung melalui banyaknya *field* data masukan yang akan diuji, aturan masukan yang harus dipenuhi serta batas masukan, baik batas atas maupun batas bawah yang memenuhi spesifikasi. Tidak ada upaya untuk mengetahui kode program apa yang *output* pakai.

Blackbox Testing atau yang lebih sering dikenal dengan sebutan pengujian fungsional merupakan metode pengujian perangkat lunak yang digunakan untuk menguji perangkat lunak tanpa mengetahui struktur internal kode atau program. Dalam pengujian ini, *tester* menyadari apa yang harus dilakukan oleh program tetapi tidak memiliki pengetahuan tentang bagaimana melakukannya. Pada *blackbox testing* ini dilakukan pengujian yang didasarkan pada detail aplikasi seperti tampilan aplikasi, fungsi-fungsi yang ada pada aplikasi, dan kesesuaian alur fungsi dengan bisnis proses yang diinginkan oleh *customer*. *Blackbox testing* ini lebih menguji ke tampilan luar (*Interface*) dari suatu aplikasi agar mudah digunakan oleh pengguna. Pengujian ini tidak melihat dan menguji *source code* program. *Blackbox testing* bekerja dengan mengabaikan struktur

kontrol sehingga perhatiannya hanya terfokus pada informasi domain (Syafnidawaty, 2020).

Keuntungan penggunaan metode *Blackbox Tetsting* adalah (Jaya, 2018):

a. Penguji tidak perlu memiliki pengetahuan tentang bahasa pemrograman tertentu.
b. Pengujian dilakukan dari sudut pandang pengguna, ini membantu untuk mengungkapkan ambiguitas atau inkonsistensi dalam spesifikasi persyaratan.
c. *Programme*r dan *tester* keduanya saling bergantung satu sama lain.

Kekurangan dari metode *Blackbox Testing* adalah:

a. Uji kasus sulit disain tanpa spesifikasi yang jelas.
b. Kemungkinan memiliki pengulangan tes yang sudah dilakukan oleh *programmer*.
c. Beberapa bagian *back end* tidak diuji sama sekali.