PRA SKRIPSI

PEMBUATAN *FRAMEWORK REMOTE PROCEDURE CALL* BERBASIS NODEJS UNTUK KOMUNIKASI DATA *CLIENT-SERVER* PADA APLIKASI MANAJEMEN KARYAWAN

HADI HIDAYAT HAMMURABI

NIM: 155410097

PROGRAM STUDI INFORMATIKA

PROGRAM SARJANA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

YOGYAKARTA

2023

PRA SKRIPSI

PEMBUATAN *FRAMEWORK REMOTE PROCEDURE CALL* BERBASIS NODEJS UNTUK KOMUNIKASI DATA *CLIENT-SERVER* PADA APLIKASI MANAJEMEN KARYAWAN

Diajukan sebagai salah satu syarat untuk menyelesaikan studi

Program Sarjana

Program Studi Informatika

Fakultas Teknologi Informasi

Universitas Teknologi Digital Indonesia

Yogyakarta

Disusun Oleh

HADI HIDAYAT HAMMURABI

NIM: 155410097

PROGRAM STUDI INFORMATIKA

PROGRAM SARJANA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

YOGYAKARTA

2023

HALAMAN PERSETUJUAN

Judul	:	PEMBUATAN *FRAMEWORK REMOTE PROCEDURE CALL* BERBASIS NODEJS UNTUK KOMUNIKASI DATA *CLIENT-SERVER* PADA APLIKASI MANAJEMEN KARYAWAN
Nama	:	Hadi Hidayat Hammurabi
NIM	:	155410097
Program Studi	:	Informatika
Program	:	Sarjana
Semester	:	Gasal
Tahun Akademik	:	2022/2023

Telah diperiksa dan disetujui untuk diseminarkan di hadapan dosen penguji seminar PRA SKRIPSI.

Yogyakarta, ……

Dosen Pembimbing

Thomas Edyson Tarigan, S. Kom., M. Cs.

NIDN: 0023107402

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT yang telah memberi rahmat dan hidayahnya-Nya sehingga penyusunan laporan Pra Skripsi ini dapat diselesaikan.

Tentunya, penulisan laporan Pra Skripsi ini tidak lepas dari dukungan dan peran serta berbagai pihak. Atas adanya bantuan tersebut, penulis hendak menyampaikan terima kasih kepada beberapa pihak, di antaranya sebagai berikut:

1. Bapak Ir. Totok Suprawoto, M.M., M.T. sebagai Rektor Universitas Teknologi Digital Indonesia.
2. Bapak Thomas Edyson Tarigan, S. Kom., M. Cs. selaku dosen pembimbing Pra Skripsi.

Laporan ini penulis susun untuk sebagai salah satu syarat untuk menyelesaikan studi jenjang Strata 1 program studi Informatika serta untuk memperoleh gelar Sarjana Komputer di Universitas Teknologi Digital Indonesia.

Penulis menyadari bahwa dalam penulisan laporan Pra Skripsi ini tentu terdapat banyak kekurangan dan jauh dari kata sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat konstruktif dengan harapan agar dapat menjadi lebih baik lagi. Semoga laporan Pra Skripsi ini memberikan manfaat bagi penulis dan tentu pula bagi pembaca.

Yogyakarta, ……...

(Penulis)

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. Latar Belakang

Komunikasi data *client-server* merupakan aktivitas berkirim data antara aplikasi di sisi *client* (pengguna) dengan aplikasi yang berjalan di sisi *server*. Hal ini dilakukan agar memungkinkan dua atau lebih *client* memiliki suatu pusat data agar seluruh kegiatan dalam suatu aplikasi dapat terintegrasi.

Di masa sekarang, kemajuan teknologi sangat pesat seiring dengan perkembangan informasi yang menjadi penyebab munculnya berbagai metode untuk mengimplementasi komunikasi data *client-server* ini. Tidak hanya itu, teknologi pemrograman juga berkembang dengan pesat sehingga muncul banyak sekali bermunculan bahasa pemrograman dan teknologi baru.

Kebutuhan terhadap aplikasi juga semakin banyak sehingga dibutuhkan sebuah *framework* untuk mempermudah dan mempercepat pengembangan aplikasi. Dengan adanya *framework*, para pengembang aplikasi dapat dipermudah karena adanya struktur yang rapi sehingga perawatan aplikasi menjadi mudah, modul-modul yang disediakan sangat membantu dalam menyelesaikan suatu masalah, dan banyak lagi keuntungan lainnya.

Salah satu metode yang dapat diterapkan untuk pengembangan aplikasi berbasis *client-server* adalah RPC yang merupakan singkatan dari *Remote Procedure Call*. Metode ini digunakan untuk memanggil suatu prosedur yang ada di komputer lain, dalam hal ini komputer *client* memanggil prosedur yang ada di komputer *server*. Implementasi metode ini dapat dilakukan di berbagai teknologi

pemrograman, seperti Javascript untuk *client*, dengan bantuan NodeJS untuk bagian *server*. Dengan demikian, pemenuhan kebutuhan *framework* untuk mempermudah implementasi RPC ini lah yang mendorong penulis untuk meneliti **"PEMBUATAN *FRAMEWORK REMOTE PROCEDURE CALL* BERBASIS NODEJS UNTUK KOMUNIKASI DATA *CLIENT-SERVER* PADA APLIKASI MANAJEMEN KARYAWAN"**. Nantinya, *framework* ini diharapkan dapat memudahkan para pengembang aplikasi dalam mengembangkan aplikasi berbasis *client-server* dengan metode RPC.

1.2. Rumusan Masalah

Berdasarkan uraian latar belakang tersebut, maka yang menjadi rumusan masalah yaitu bagaimana mengimplementasi *framework* RPC menggunakan NodeJS untuk studi kasus aplikasi manajemen karyawan.

1.3. Ruang Lingkup

Adapun ruang lingkup permasalahan dalam penelitian ini, yaitu:

1. Membuat sebuah *framework* berbasis NodeJS.
2. *Framework* ini dibuat untuk implementasi metode RPC.
3. Protokol yang digunakan adalah HTTP.
4. *Framework* diterapkan pada aplikasi manajemen karyawan.
5. Manajemen yang dilakukan meliputi operasi pengambilan, penambahan, pengubahan, dan penghapusan data karyawan.
6. *Framework* akan diuji berdasarkan pengujian fungsionalitas.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membangun sebuah *framework* berbasis NodeJS dengan metode RPC sebagai media komunikasi antara *client* dengan *server*.

1.5. Manfaat Penelitian

Manfaat dari penelitian ini, yaitu:

1. Mengetahui perancangan sebuah *framework* RPC untuk NodeJS.
2. Menghasilkan *framework* yang dapat membantu para pengembang dalam mengimplementasi metode RPC.

BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Beberapa penelitian mengenai NodeJS dan arsitektur *client-server* yang pernah dilakukan, sekaligus menjadi acuan dalam pengembangan *framework* ini ditampilkan pada tabel.

Tabel 2.1. Tinjauan Pustaka

Penulis	**Teknologi/ Metode**	**Objek**	**Masalah**	**Hasil**
Setiawan Wawan (2017)	NodeJS	Absensi Siswa	Absensi dilakukan manual oleh guru piket	Aplikasi web presensi online
Iqbal Sulistyo (2017)	*Model-View-Controller*	Penanganan Tilang	Pelaksanaan tilang yang masih konvensional	Sistem informasi tilang
Lalu Himawan Satraji (2017)	*Web Service*	Delivery Makanan Khas Yogyakarta	Pemesanan delivery untuk makanan khas Yogyakarta masih belum ada	Aplikasi delivery makanan khas Yogyakarta

Yosafat Aria Negara (2018)	*Client-Server*	Layanan Kamar Hotel	Pelayanan hotel yang masih manual	Aplikasi layanan kamar hotel
Riko (2019)	REST API	Sistem Penjadwalan Pendadaran dan Seminar Proposal	Penjadwalan ujian pendadaran dan seminar proposal yang masih dilakukan manual oleh akademik	Sistem informasi penjadwalan pendadaran dan seminar proposal
Hadi Hidayat Hammurabi (2021)	NodeJS, RPC	Aplikasi Berbasis RPC	Pengembangan aplikasi RPC yang belum banyak didukung *framework*	*Framework* RPC NodeJS

Berkembangnya teknologi informasi, terutama aplikasi web, menunjukkan kebutuhan teknologi semakin meluas. Hal ini tampak pada penelitian Wawan Setiawan tahun 2017, kebutuhan untuk absensi siswa dapat dilakukan dengan menggunakan teknologi berupa aplikasi web. Pada penelitian ini, penulis menggunakan teknologi NodeJS untuk membuat seluruh bagian dari aplikasi tersebut.

Berbagai usaha dilakukan agar dapat memenuhi kebutuhan lainnya, termasuk peningkatan arsitektur aplikasi dari segi struktur kode. Menurut Iqbal Sulistyo pada penelitiannya di tahun 2017, bahwa struktur kode *Model-View-Controller* (MVC) dapat membantu pengembang dalam membuat aplikasi dengan cepat dan mudah. Begitu pula pada penelitian Yosafat Aria Negara tahun 2018,

arsitektur aplikasi dibuat dengan menerapkan konsep *client-server* untuk memisahkan kode pada sisi client dengan sisi server yang praktiknya digunakan untuk membangun aplikasi layanan kamar hotel.

Implementasi konsep *client-server* ini dapat dilakukan dengan beberapa cara, salah satunya adalah dengan menggunakan *web service*. Dengan demikian, layanan-layanan menjadi terbuka dan mudah diakses untuk kepentingan integrasi data dan kolaborasi informasi yang bisa diakses melalui internet, hal ini termuat dalam penelitian yang dilakukan oleh Lalu Himawan Satraji pada tahun 2017.

Kemudian, *web service* juga dapat dibuat dalam berbagai bentuk dan jenisnya, salah satunya adalah REST API. Seperti hasil penelitian Riko tahun 2019 yang mengimplementasi REST API untuk membuat aplikasi penjadwalan ujian pendadaran dan seminar proposal, dikatakan bahwa REST API digunakan untuk memfasilitasi pertukaran informasi atau data antara dua atau lebih aplikasi.

Pada penelitian ini, membuat *framework client-server* yang menerapkan metode *Remote Procedure Call* (RPC) dengan menggunakan bahasa pemrograman Javascript dan dijalankan dengan NodeJS sebagai servernya. Implementasi *client-server* di sini menerapkan protokol yang sama dengan *web service* dan salah satu metode pengiriman data yang terdapat pada REST API. Pembuatan *framework* ini bertujuan untuk mempermudah dalam melakukan pengembangan aplikasi berbasis *client-server*, terutama metode RPC.

2.2. Dasar Teori

2.2.1. Protokol HTTP

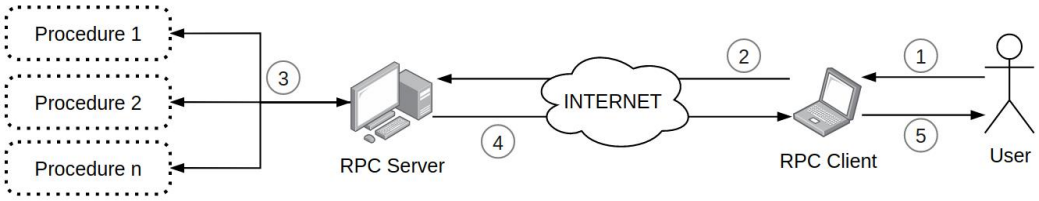
HTTP atau *Hyper Text Transfer Protocol* merupakan protokol yang sangat sederhana, berdasarkan skema yang sudah dikenal dari serangkaian kecil perintah yang dikeluarkan oleh *client* (*web browser*) dan informasi terkait dengan bentuk kode balasan yang dikeluarkan oleh *server* (situs *web*). (Goralski. W, 2017)

2.2.2. Komunikasi Data Client-Server

Client digambarkan sebagai program yang meminta untuk administrasi atau data, dan *server* digambarkan sebagai prosedur atau aplikasi yang memberikan administrasi atau data. Administrasi atau data yang diminta dan diberikan antara *server* dan *client* dapat menjadi aset, misalnya informasi, dokumen, objek, perangkat tampilan, atau kontrol. Contoh kerangka kerja *client-server* adalah program web, pengelolaan akun, dan kerangka email. (Shobhika Sejwal, 2019)

2.2.3. Remote Procedure Call

RPC (*Remote Procedure Call*) memungkinkan pengguna untuk melakukan panggilan ke prosedur jarak jauh yang berada di ruang proses lain. Proses ini dapat berjalan di mesin yang sama atau mesin lain di jaringan. Mekanisme RPC banyak digunakan dalam membangun sistem terdistribusi karena mereka mengurangi kompleksitas sistem dan biaya pengembangan. Tujuan utama RPC adalah membuat panggilan prosedur jarak jauh menjadi transparan bagi pengguna. Dengan kata lain, ini memungkinkan pengguna untuk melakukan panggilan prosedur jarak jauh seperti panggilan prosedur lokal. (Hakan Bagci, 2016)



Gambar 2.1. Mekanisme RPC (*Remote Procedure Call*)

Proses di awali pada tahap 1, yaitu pengguna (*user*) mengakses aplikasi *client*. Kemudian dilanjutkan ke tahap 2, aplikasi *client* menghubungi RPC *server* melalui internet. Di tahap 3, RPC *server* menjalankan prosedur yang dibutuhkan oleh *client*. Kemudian hasil dari proses tersebut masuk ke tahap 4, yaitu mengirim hasil kepada aplikasi *client* melalui internet. Selanjutnya pada tahap 5, aplikasi *client* mengolah data yang diterima dan ditampilkan ke pengguna.

2.2.4. Framework Aplikasi

Framework pada dasarnya adalah alat bantu untuk membangun aplikasi sehingga terhindar dari *bug* dan menghemat waktu. *Framework* memiliki aturan dan arsitektur sehingga memungkinkan untuk membuat berbagai jenis aplikasi. (Dasari Hermitha Curie, 2019)

Aturan-aturan yang ada pada sebuah *framework* bergantung pada arsitektur *framework* itu sendiri. Misalkan, salah satu arsitektur adalah MVC (*Model-View-Controller*) yang memisahkan bagian pemodelan data, alur bisnis, dan tampilan. Pada *framework* dengan arsitektur ini, terdapat aturan yang mengharuskan pengembang memisahkan ketiga bagian tersebut sehingga tidak dapat melakukan hal-hal di luar aturan tersebut.

2.2.5. Javascript

Javascript adalah bahasa pemrograman dinamis yang dapat menambahkan interaktivitas ke situs web. (Mozilla, 2020) Dengan Javascript, halaman web dapat memiliki animasi, tombol yang dapat diklik, hingga menampilkan *popup*. Beberapa web *browser* terkenal seperti Mozilla Firefox dan Google Chrome menjadikan Javascript sebagai bahasa pemrograman standar bagi setiap halaman web yang dimuat.

Melalui Javascript, pengembang dapat mengelola halaman web melalui pemrograman melalui *Document Object Model* (DOM). DOM merupakan cara agar Javascript dapat memanipulasi elemen-elemen yang terdapat pada halaman web, seperti memunculkan teks, menghilangkan elemen tertentu, hingga melakukan hal tertentu saat sebuah tombol diklik oleh pengguna.

2.2.6. NodeJS

NodeJS merupakan perangkat lunak yang digunakan untuk menjalankan program berbasis Javascript. Dengan menggunakan metode asingkron dalam manajemen prosesnya, NodeJS memungkinkan untuk membangun aplikasi jaringan yang dapat diskalakan. (NodeJS, 2020)

NodeJS biasa digunakan untuk membuat aplikasi berbasis Web. Dengan demikian memungkinkan pengembang untuk menggunakan Javascript di sisi *client* dan juga di sisi *server* sehingga dapat mengurangi beban dalam mempelajari maupun menerapkan bahasa pemrograman lainnya.

Javascript yang sebelumnya hanya dapat berjalan di lingkup client, dengan menggunakan NodeJS, Javascript dapat memperluas cakupannya ke ranah server.

Dengan demikian, pengembang mendapat keuntungan berupa kode dari Javascript dapat mengakses hal-hal yang berkaitan dengan sistem operasi, seperti mengelola *file*, mengakses DBMS (*Database Management System*), mengelola protokol jaringan.

BAB III

METODE PENELITIAN

3.1. Peralatan

Dalam melaksanakan penelitian digunakan peralatan untuk menunjang berjalannya pembuatan *framework*.

3.1.1. Perangkat Keras

Adapun perangkat keras yang digunakan dalam membangun *framework* ini, yaitu komputer dengan:

a. *processor* Intel Core i5

b. *memory* 8GB

c. Solid State Disk (SSD) 120GB

3.1.2. Perangkat Lunak

Adapun perangkat lunak yang digunakan dalam membangun *framework* ini, yaitu:

a. Arch Linux sebagai sistem operasi

b. Chromium sebagai *web browser*

c. VSCodium sebagai penyunting kode

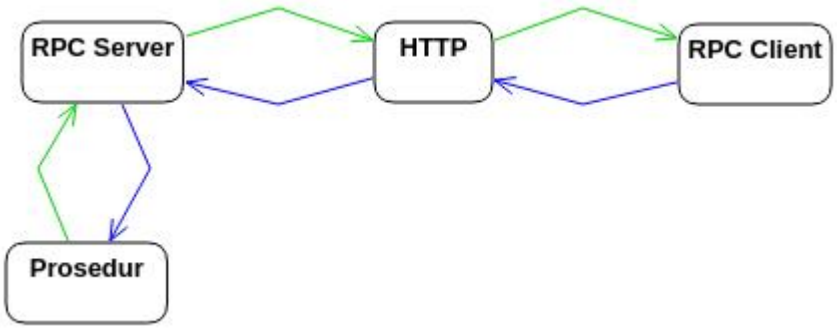
d. NodeJS untuk menjalankan *framework*

e. Git untuk memanajemen versi

3.2. Perancangan Framework

3.2.1. Komponen Utama Framework

Framework yang dibuat dalam penelitian ini terdiri dari beberapa bagian (komponen) yang tergambarkan pada ilustrasi berikut.



Keterangan warna:

_____ : garis request

_____ : garis response

Gambar 3.1. Bagian utama dari *framework*

Gambar di atas merupakan gambaran bagian utama yang membentuk *framework* dalam penelitian ini. Setiap bagian memiliki peran masing-masing dan saling terintegrasi.

RPC *server* berperan untuk melayani segala permintaan RPC *client* guna memenuhi kebutuhan pengguna. Di dalamnya terdapat berbagai prosedur yang dapat dijalankan saat dibutuhkan, terutama oleh RPC *client*.

HTTP (*HyperText Transfer Protocol*) merupakan protokol yang menjadi sarana RPC *client* berkomunikasi dengan RPC *server*. Protokol ini berdiri di atas jaringan internet sehingga komunikasi dapat dilakukan dari manapun selama terkoneksi dengan internet.

RPC *client* berperan sebagai media berinteraksinya pengguna dengan aplikasi. Bila terjadi aksi yang memerlukan RPC *server* maka RPC *client* akan menghubungi RPC *server* untuk melanjutkan pemrosesan.

Prosedur merupakan aktivitas-aktivitas yang dapat dilakukan oleh RPC *server*. Tiap prosedur berisi kumpulan baris kode yang bertujuan untuk memenuhi kebutuhan tertentu secara spesifik. Prosedur juga dapat memiliki nilai balikan (*return value*) maupun tidak.

3.2.2. RPC Server

Dalam rancangan *framework* ini, RPC Server menyediakan layanan yang di dalamnya terdapat kumpulan prosedur. Tiap prosedur dibuat sesuai dengan kebutuhan masing-masing sistem yang dikembangkan. Kemudian prosedur-prosedur tersebut, digunakan oleh sisi *client* dengan cara *Remote Procedure Call*.

Saat sebuah prosedur digunakan oleh *client,* maka *server* akan menjalankannya dan mengirimkan kembali data yang dihasilkan. Jika prosedur tersebut tidak menghasilkan data, maka tidak ada data yang dikirimkan kembali ke *client*.

3.2.3. RPC Client

Tugas utama dari RPC *client* adalah menyambungkan aplikasi dengan RPC *server*. Kemudian, aplikasi dapat memerintahkan server untuk menjalankan prosedur melalui RPC *client*.

Dalam perancangan ini, RPC *client* dapat mengelola data yang dihasilkan dari pemanggilan prosedur. Pengelolaan data dilakukan saat prosedur

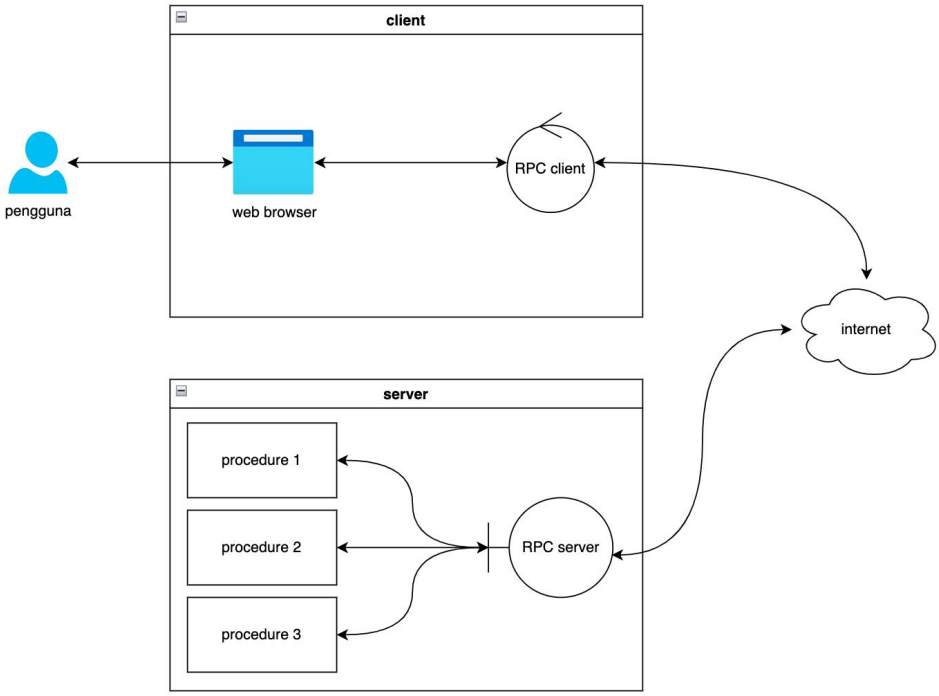
menghasilkan data maupun tidak. Jika prosedur tidak menghasilkan data, maka RPC *client* akan memberi informasi bahwa tidak ada data yang didapatkan.

3.3. Perancangan Aplikasi

Perancangan aplikasi merupakan tahapan untuk menggambarkan bagaimana sebuah aplikasi dibagun dengan mengacu pada kebutuhan-kebutuhan fungsionalitas. Dalam perancangan ini terdiri atas diagram *usecase*, diagram *sequence*, diagram *activity*, rancangan tabel, dan rancangan antarmuka.

3.3.1. Arsitektur Sistem

Aplikasi Manajemen Karyawan dibangun dengan beberapa lapisan teknologi yang saling terintegrasi. Adapun bagian utamanya dapat dilihat pada diagram berikut.



Gambar 3.2. Diagram arsitektur sistem

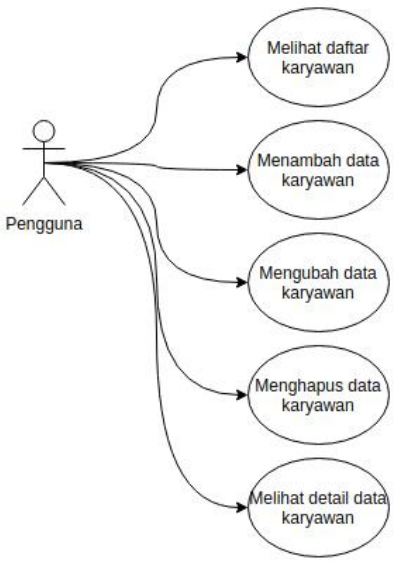
Secara arsitektur, Aplikasi Manajemen Karyawan terbagi 2 bagian, yaitu *client* dan *server*. *Client* berguna sebagai tempat interaksi antara aplikasi dengan pengguna. Pengguna dapat mengakses *client* dengan menggunakan *web browser*.

Pada bagian *client*, terdapat RPC *client* yang terkoneksi dengan *server* melalui Internet. Kemudian *client* dapat mengirim dan menerima data melalui koneksi yang aktif tersebut.

Di sisi *server*, pengiriman dan penerimaan data, serta koneksi *client* diterima dan dilayani oleh RPC *server*. Bagian inilah yang akan mengarahkan prosedur mana yang harus dijalankan saat interaksi dengan *client* terjadi.

3.3.2. Diagram Usecase

Diagram *usecase* adalah gambaran dari beberapa atau semua aktor, *usecase*, dan interaksi di antara keduanya. Berikut ini adalah diagram *usecase* aplikasi manajemen data karyawan.

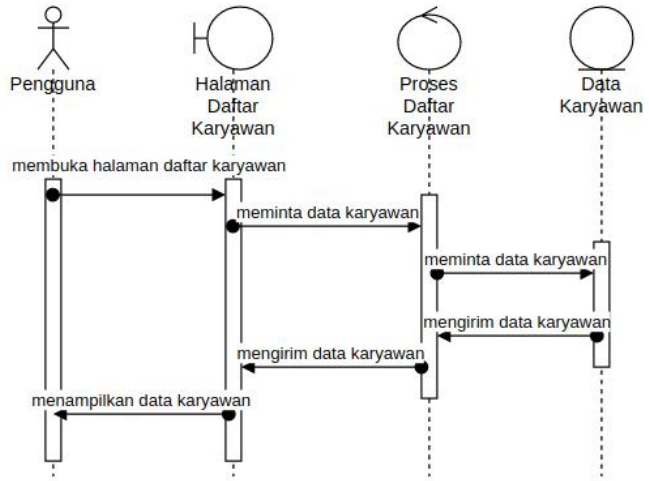


Gambar 3.3. Diagram *usecase*

Pada gambar di atas, terdapat gambaran aktivitas pengguna terhadap aplikasi. Pengguna dapat memanajemen data karyawan, seperti melihat daftar karyawan, menambah data karyawan, mengubah data karyawan, menghapus data karyawan, dan melihat data karyawan secara detail.

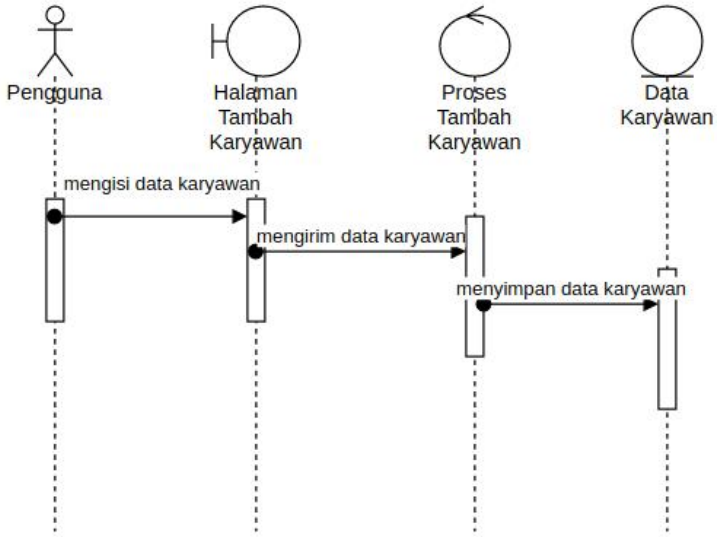
3.3.3. Diagram Sequence

Diagram *sequence* digunakan untuk menunjukkan aliran fungsionalitas dalam setiap aktivitas yang terdapat pada diagram *usecase* sebelumnya. Berikut ini diagram *sequence* aplikasi manajemen karyawan.



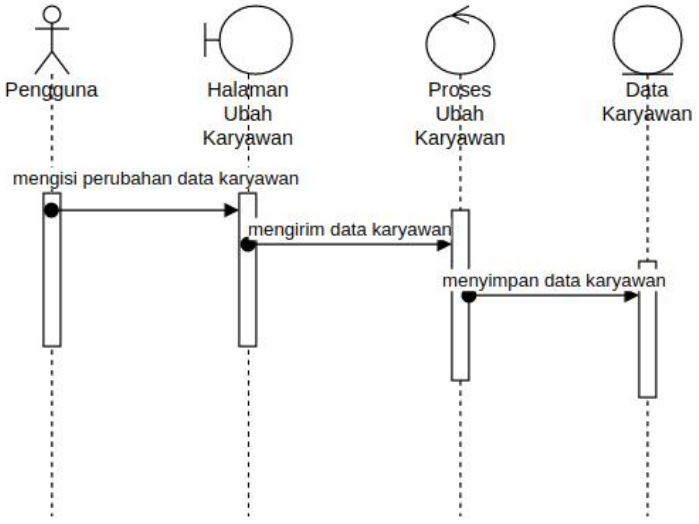
Gambar 3.3. Diagram *sequence* menampilkan daftar karyawan

Pada gambar di atas, menggambarkan aliran saat pengguna membuka halaman daftar karyawan, kemudian diproses dengan mengambil data karyawan dan ditampilkan kembali pada pengguna.



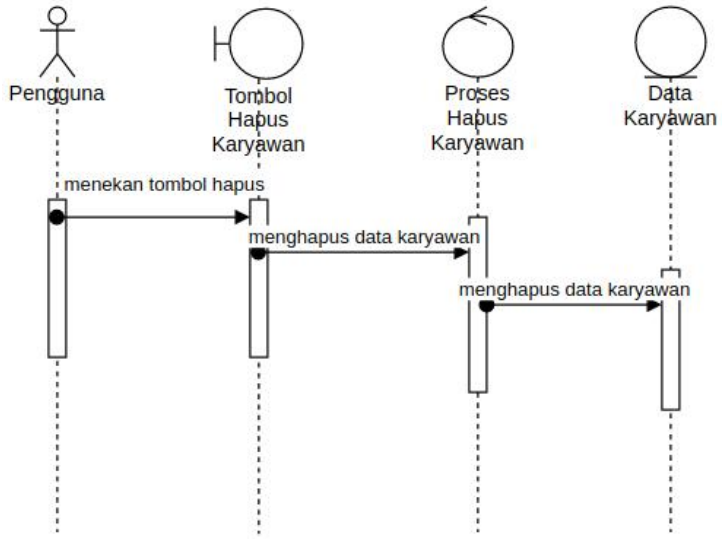
Gambar 3.4. Diagram *sequence* menambah data karyawan

Pada gambar di atas, pengguna akan melakukan penambahan data dengan mengisi data karyawan baru melalui halaman tambah karyawan. Kemudian data tersebut di kirim ke proses penambahan data untuk selanjutnya di simpan ke data karyawan.



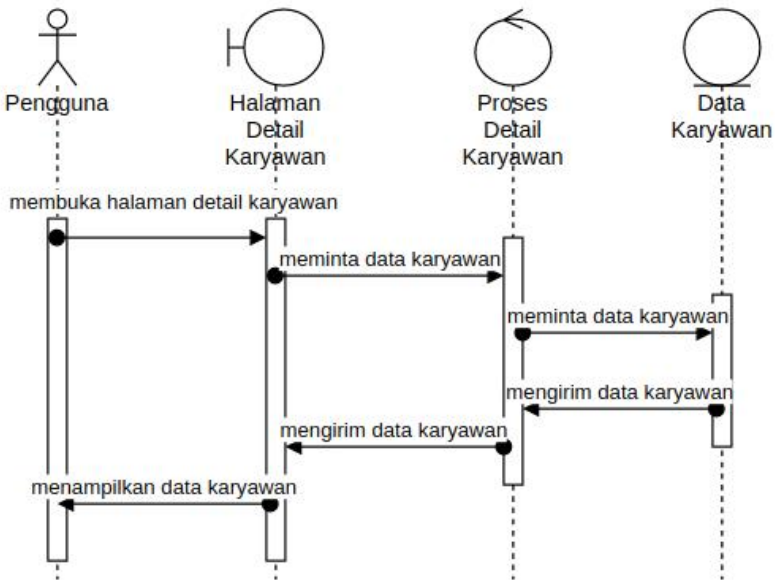
Gambar 3.5. Diagram *sequence* mengubah data karyawan

Pada gambar di atas, pengguna melakukan perubahan data terhadap data karyawan yang sudah tersedia. Pengguna mengisi perubahan data karyawan melalui halaman ubah karyawan. Kemudian data tersebut di kirim ke proses perngubahan data untuk selanjutnya di simpan ke data karyawan.



Gambar 3.6. Diagram *sequence* menghapus data karyawan

Pada gambar di atas, pengguna melakukan penghapusan data dengan menekan tombol hapus data karyawan. Kemudian data tersebut di kirim ke proses penghapusan data untuk selanjutnya di simpan ke data karyawan.

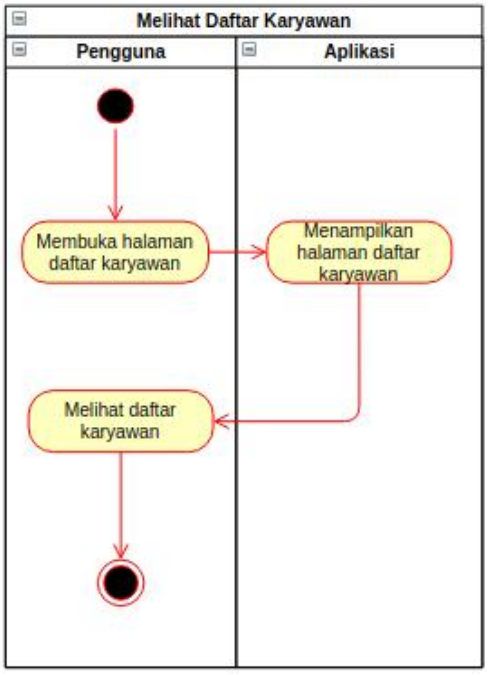


Gambar 3.7. Diagram *sequence* melihat detail data karyawan

Pada gambar di atas, pengguna meminta untuk melihat data karwayan secara detail. Kemudian permintaan tersebut diproses dengan mengambil data karyawan dan ditampilkan kembali pada halaman detail karyawan.

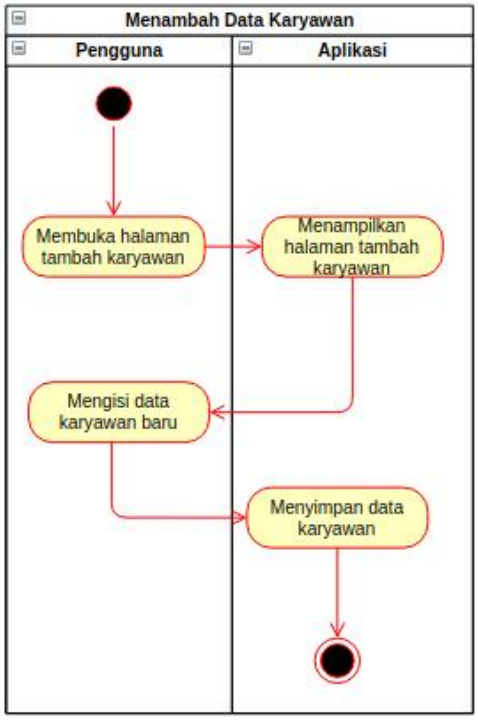
3.3.4. Diagram Activity

Diagram *activity* menggambarkan proses-proses yang terjadi di setiap aktivitas yang ada pada aplikasi.



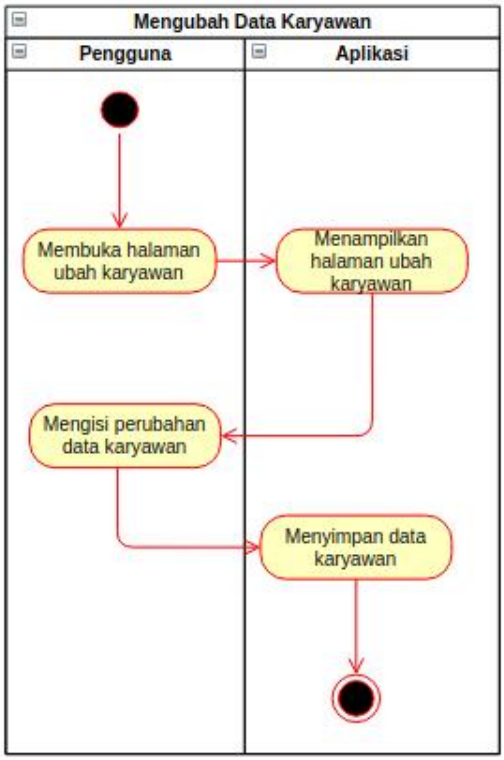
Gambar 3.8. Diagram *activity* melihat daftar karyawan

Pada gambar di atas, terdapat rangkaian aktivitas saat pengguna melihat daftar karyawan. Awalnya, pengguna membuka halaman daftar karyawan. Kemudian, aplikasi menampilkan halaman daftar karyawan sehingga pengguna dapat melihat daftar karyawan.



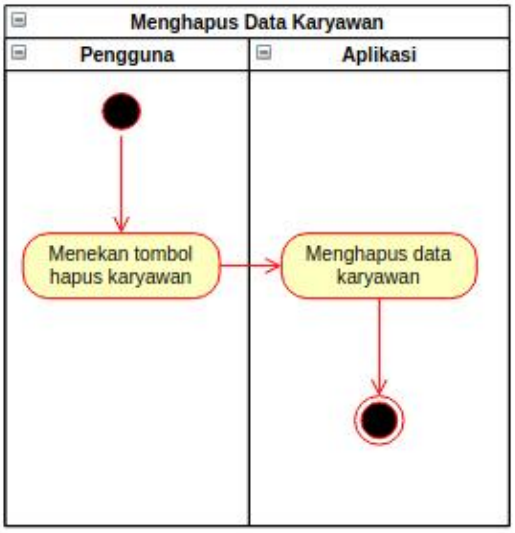
Gambar 3.9. Diagram *activity* menambah data karyawan

Pada gambar di atas, terdapat rangkaian aktivitas saat pengguna menambah data karyawan. Awalnya, pengguna membuka halaman tambah karyawan. Kemudian, aplikasi menampilkan halaman tambah karyawan sehingga pengguna dapat mengisi data karyawan baru. Saat data tersebut dikirimkan kembali ke aplikasi, data disimpan pada pusat data karyawan.



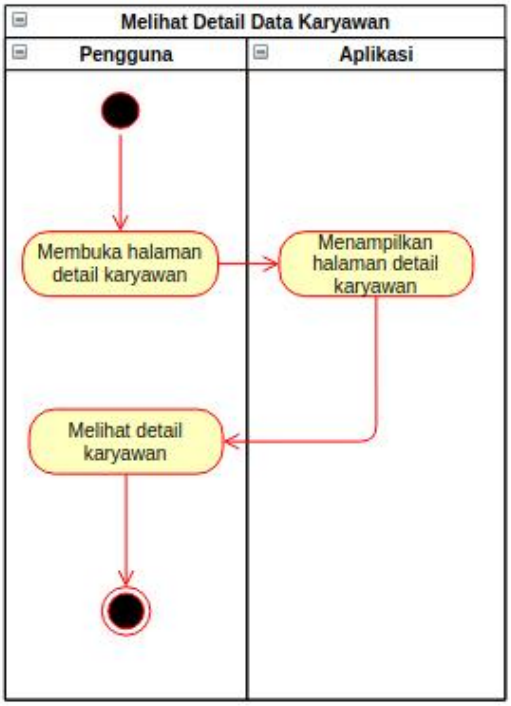
Gambar 3.10. Diagram *activity* mengubah data karyawan

Pada gambar di atas, terdapat rangkaian aktivitas saat pengguna mengubah data karyawan. Awalnya, pengguna membuka halaman ubah karyawan. Kemudian, aplikasi menampilkan halaman ubah karyawan sehingga pengguna dapat mengisi perubahan data karyawan. Saat data tersebut dikirimkan kembali ke aplikasi, data disimpan pada pusat data karyawan.



Gambar 3.11. Diagram *activity* menghapus data karyawan

Pada gambar di atas, terdapat rangkaian aktivitas saat pengguna menghapus data karyawan. Awalnya, pengguna menekan tombol hapus karyawan. Kemudian, aplikasi menghapus data karyawan tersebut dari pusat data karyawan.



Gambar 3.12. Diagram *activity* melihat detail data karyawan

Pada gambar di atas, terdapat rangkaian aktivitas saat pengguna melihat detail data karyawan. Awalnya, membuka halaman detail karyawan. Kemudian, aplikasi menampilkan data karyawan tersebut dari pusat data karyawan sehingga pengguna dapat melihat detail karyawan.

3.3.5. Rancangan Tabel

Rancangan tabel menunjukkan bagaimana tabel digunakan untuk menyimpan data dalam pusat data. Pada aplikasi ini, terdapat 1 tabel yang menyimpan seluruh data. Berikut ini tabel tersebut.

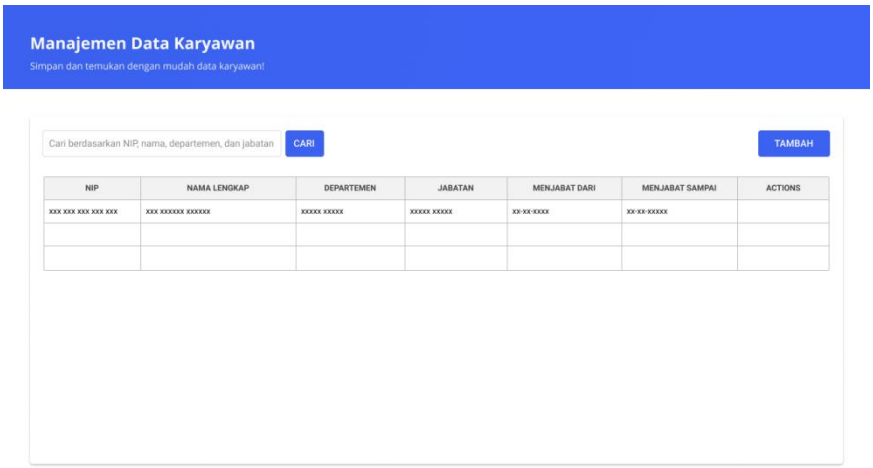
Tabel 3.13. Struktur tabel karyawan

Nomor	Nama Kolom	Tipe Data	Keterangan
1	id	Integer	identitas unik setiap data
2	nip	Varchar	kode unik kepegawaian
3	nama	Varchar	nama lengkap pegawai
4	departemen	Varchar	nama bagian tempat pegawai ditugaskan
5	jabatan	Varchar	nama tingkatan pegawai dalam instansi
6	menjabat_dari	Date	tanggal mulai menjabat
7	menjabat_sampai	Date	tanggal berakhirnya jabatan

Pada tabel di atas, seluruh data karyawan di simpan pada penyimpanan karyawan yang digunakan untuk mengelola data-data tiap karyawan.

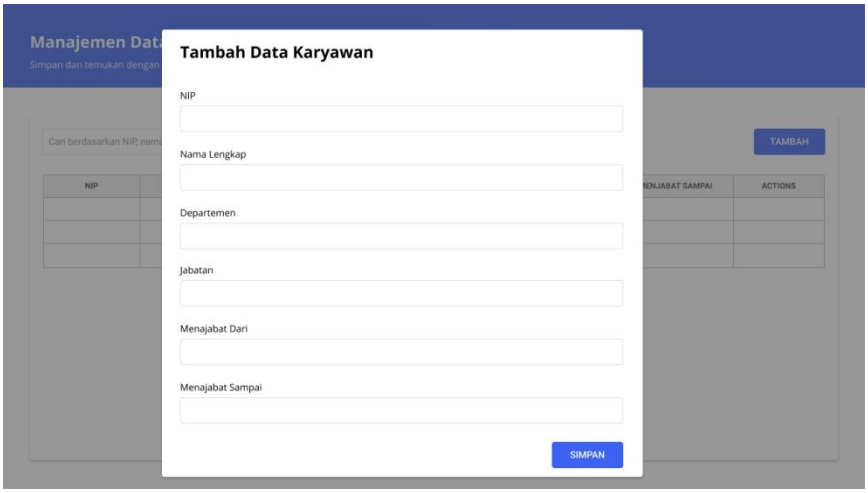
3.3.6. Rancangan Antarmuka

Rancangan antarmuka ini berupa gambaran umum desain tampilan yang akan digunakan saat proses pengembangan sistem. Setiap rancangan antarmuka halaman sistem dapat dilihat pada rincian gambar berikut.



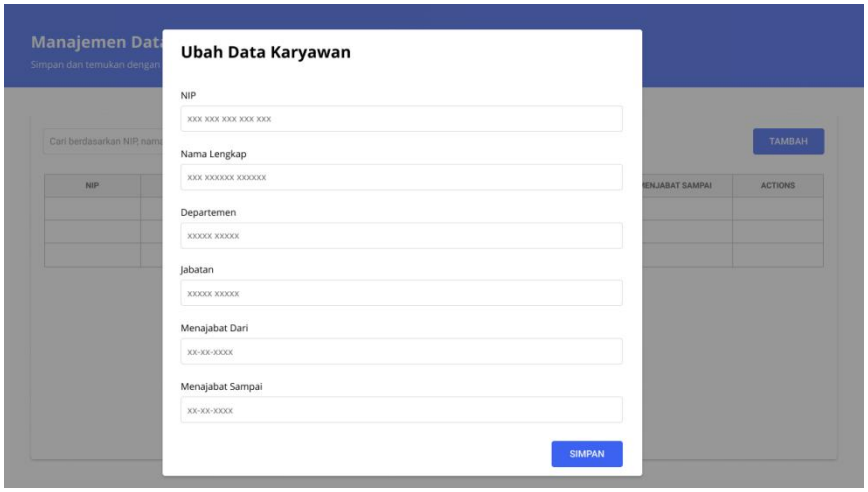
Gambar 3.14. Rancangan antarmuka daftar karyawan

Gambar di atas merupakan halaman depan aplikasi saat pertama kali dibuka. Pada gambar tersebut, terdapat beberapa bagian utama, yaitu judul aplikasi, input pencarian, tombol untuk menambah data, dan tabel untuk menampilkan seluruh data karyawan yang tersedia.



Gambar 3.15. Rancangan antarmuka menambah data karyawan

Gambar di atas merupakan dialog yang berisi input untuk menambah data karyawan. Setelah input diisi, lalu tombol simpan ditekan, maka aplikasi akan menyimpan data tersebut.

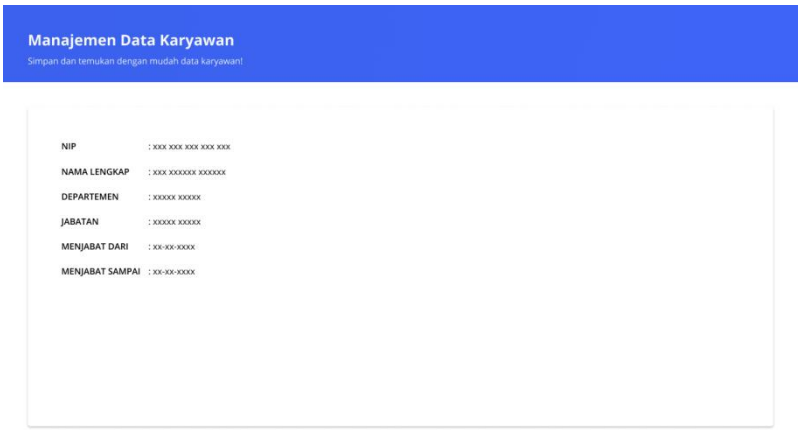


Gambar 3.16. Rancangan antarmuka mengubah data karyawan

Gambar di atas merupakan halaman untuk mengubah data karyawan yang berisi data sebelumnya untuk diubah pada formulir input. Setelah input diisi, lalu tombol simpan ditekan, maka data karyawan akan diperbarui sesuai dengan yang perubahan yang dilakukan.

Gambar 3.17. Rancangan antarmuka konfirmasi hapus data karyawan

Gambar di atas merupakan tampilan dialog konfirmasi penghapusan data. Munculnya dialog ini, berguna untuk memastikan kembali dan mengurangi kesalahan saat pengguna akan menghapus data karyawan. Bila tombol lanjutkan ditekan, data karyawan terkait akan terhapus.



Gambar 3.18. Rancangan antarmuka melihat detail data karyawan

Gambar di atas merupakan tampilan detail data karyawan. Setiap data karyawan dapat dilihat secara detail melalui halaman di atas.

DAFTAR PUSTAKA

Bagci, Hakan. 2016. *A Lightweight and High Performance Remote Procedure Call Framework for Cross Platform Communication*. Turkey: SCITEPRESS.

Curie, Dasari Hermitha. 2019. *Analysis on Web Frameworks*. India: Karunya Institute of Technology and Sciences.

Fredrich, Todd. 2019. *Learn REST: A RESTful Tutorial*. https://restapitutorial.com. Diakses 10 Maret 2020.

Goralski, Walter. 2017. *The Illustrated Network*. United States: Elsevier Inc.

Mozilla. 2020. *Resources for developers, by developers*. https://developer.mozilla.org. Diakses 20 Agustus 2020.

Negara, Yosafat Aria. 2018. *Aplikasi Komunikasi Socket Client-Server Layanan Kamar Hotel Berbasis Android Pada Jaringan Lokal Menggunakan TCP IP*. Diploma thesis, STMIK Akakom Yogyakarta.

NodeJS. 2020. *NodeJS*. https://nodejs.org. Diakses 20 Agustus 2020.

Putranto, Bambang Purnomosidi Dwi. 2013. *Pengembangan Aplikasi Cloud Menggunakan Node.js*. https://github.com/bpdp/buku-cloud-nodejs. Diakses 10 Maret 2020.

Riko. 2019. *Implementasi REST API untuk Sistem Penjadwalan Pendadaran dan Seminar Proposal Skripsi Mahasiswa (Studi Kasus Program Studi Sistem Informasi STMIK Akakom Yogyakarta)*. Sistem Informasi STMIK Akakom Yogyakarta

Satraji, Lalu Himawan. 2017. *Implementasi Web Service untuk Delivery Makanan Khas Yogyakarta Berbasis Mobile*. Teknik Informatika STMIK Akakom Yogyakarta.

Sejwal, Shobhika. 2019. *An Approach to Resolve Heterogeneity Using RPC in Client Server Systems*. India: Amity University.

Srinivasan, R. 1995. *RPC: Remote Procedure Call Protocol Specification Version 2.* Sun Microsystems.

Sulistyo, Iqbal. 2017. *Implementasi Teknologi Responsive Web pada Sistem Informasi Tilang Berbasis Codeigniter*. Teknik Informatika STMIK Akakom Yogyakarta.

Wawan, Setiawan. 2017. *Absensi Siswa dengan Teknologi NodeJS Studi Kasus SMKN 1 Sawit*. Teknik Informatika Sekolah Tinggi Manajemen Informatika dan Komputer Akakom.