Multidisciplinary Indonesian Center Journal (MICJO)
Journal page is available to
https://e-jurnal.jurnalcenter.com/index.php/micjo
Email: admin@jurnalcenter.com

E.ISSN. 3032-2472
Vol. 1 No. 3 Edisi
Juli 2024

PENERAPAN ARSITEKTUR MICROSERVICES PADA MIGRASI SISTEM INFORMASI DWIDAYA TOUR DENGAN PENDEKATAN PENGEMBANGAN APLIKASI CEPAT (RAD)

Iwan Setiawan [1], Rama Adhitya Nugroho [2*], Hadi Supratikta[3]

[1*,2,3] Magister Manajemen, Pascasarjana, Universitas Pamulang
*email Koresponden: aa.iwan.setiawan@gmail.com

DOI: 10.62567/micjo.v1i3.138

Article info:
Submitted: 13/05/24 Accepted: 16/07/24 Published: 30/07/24

Abstract

With the opening of branches around Indonesia and the provision of a range of services including train tickets, airline tickets, and Umrah packages, Dwidaya Tour has seen tremendous commercial development. The absence of communication between branches and services has created difficulties. Dwidaya Tour hence hopes to quickly transition to a centralized system without interfering with business as usual. This study aims to investigate how these problems might be solved through implementation, with a particular focus on the use of microservices architecture to the migration process. By using this method, every system may be linked to the centralized system at the frontend as a backend service. An SDLC strategy based on RAD is used since a speedy migration is required. User assessment is used in this RAD technique as a control mechanism.

Keywords **:** *SDLC, RAD, Microservices Architecture, System Migration.*

Abstrak

Dengan dibukanya cabang-cabang di seluruh Indonesia dan penyediaan berbagai layanan termasuk tiket kereta, tiket pesawat, dan paket Umrah, Dwidaya Tour telah mengalami perkembangan komersial yang sangat pesat. Ketidakmampuan komunikasi antara cabang-cabang dan layanan telah menciptakan kesulitan. Oleh karena itu, Dwidaya Tour berharap dapat segera beralih ke sistem terpusat tanpa mengganggu kegiatan bisnis sehari-hari. Studi ini bertujuan untuk menyelidiki bagaimana masalah-masalah ini dapat diatasi melalui implementasi, dengan fokus khusus pada penggunaan arsitektur microservices dalam proses migrasi. Dengan menggunakan metode ini, setiap sistem dapat terhubung ke sistem terpusat di frontend sebagai layanan backend. Strategi SDLC berbasis RAD digunakan karena migrasi yang cepat diperlukan. Evaluasi pengguna digunakan dalam teknik RAD ini sebagai mekanisme kontrol.

Kata Kunci : SDLC, RAD, Arsitektur Microservices, Migrasi Sistem.

1. PENDAHULUAN

Dwidayatour, juga dikenal sebagai Tour Dwidaya, adalah agen perjalanan terkemuka di Indonesia. Sejak didirikan pada 19 Juli 1967, perusahaan telah membangun reputasi yang kuat. Perusahaan ini memiliki lebih dari sembilan puluh cabang di seluruh negeri dan telah bekerja sama dengan beberapa maskapai penerbangan utama seperti Qatar Airways, Singapore

Multidisciplinary Indonesian Center Journal (MICJO)
Journal page is available to
https://e-jurnal.jurnalcenter.com/index.php/micjo
Email: admin@jurnalcenter.com
E.ISSN. 3032-2472
Vol. 1 No. 3 Edisi
Juli 2024

Airlines, dan Cathay Pacific. Meskipun perusahaan telah mencapai banyak kesuksesan, masih ada masalah dalam mengelola sistem informasi yang terpisah dan tidak terhubung.

Saat ini, setiap lini bisnis dan cabang Dwidayatour menggunakan sistem informasi yang berbeda. Akibatnya, data tidak konsisten dan pelaporan tertunda. Kesalahan dan ketidakakuratan data sering terjadi, tetapi pencatatan dan rekapitulasi data manual memakan waktu yang berharga. Meskipun adopsi arsitektur monolitik menyebabkan masalah ini, Dwidayatour bermaksud untuk mengubah sistem ke arsitektur pemrograman microservices untuk meningkatkan efisiensi dan akurasi pelaporan data serta memenuhi kebutuhan bisnis yang terus berubah. Metode ini akan mempercepat aliran data dan memungkinkan integrasi sistem yang lebih efektif tanpa mengganggu operasi normal.

Siklus hidup pengembangan sistem, juga dikenal sebagai SDLC, dimulai dengan tahapan awal, melewati tahapan berikutnya, dan akhirnya kembali ke tahapan awal, yang menghasilkan daur hidup yang berkelanjutan (Jogiyanto, 2011).

Gambar 1. Diagram Daur Proses Sistem
Sumber : Hadi Supraptika et al., 2021

Menurut Tata Sutabri, S.Kom, MMSI (2016), ada empat metodologi utama untuk pengembangan sistem berbasis SDLC:

1. *Waterfall*
2. *Prototyping*
3. Pengembangan *Software* Cepat (RAD) dan
4. Pengembangan *Software Agile*

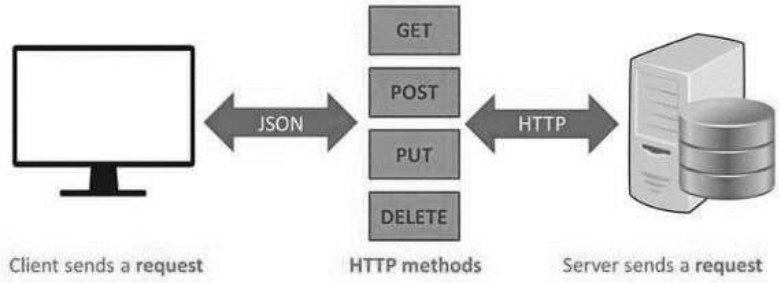
Arsitektur microservice menguraikan aplikasi kompleks menjadi beberapa layanan otonom dengan proses kecil, yang beroperasi secara independen dan berkomunikasi melalui API masing-masing, menggunakan bahasa yang berbeda-beda. Dengan fleksibilitasnya, layanan ini dapat fokus pada tugas-tugas kecil dengan jelas, memungkinkan penggunaan pendekatan modular dalam pembangunan sistem. Kemampuannya untuk disesuaikan dengan kebutuhan bisnis tertentu dan berbagai teknologi membuat penggantian menjadi mudah. Layanan-layanan ini dapat dikembangkan dan digunakan secara terpisah sesuai dengan metodologi pengembangan perangkat lunak yang berkelanjutan. Sebaliknya, pendekatan microservice lebih fleksibel daripada arsitektur monolitik karena mereka bergantung pada satu basis data untuk interaksi dan sulit untuk menentukan lokasi kesalahan yang tepat. Arsitektur layanan microservice memungkinkan manajemen tim layanan untuk mendukung tim produk dan layanan lebih efektif saat mengembangkan platform tunggal. Ketika terjadi masalah dalam proses produksi, mencari dan mengisolasi masalah operasional menjadi lebih mudah dan lebih mudah untuk menentukan tim pengembangan mana yang harus bertanggung jawab untuk menyelesaikan masalah tersebut.

Platform API dalam arsitektur microservice berfungsi sebagai perantara untuk pertukaran data dan layanan transaksi. API, singkatan dari Application Programming Interface, menghubungkan komputer ke bagian perangkat lunak atau sistem lainnya dan mengatur bagaimana bagian atau sistem lain digunakan. Pengembangan API saat ini biasanya bergaya

Multidisciplinary Indonesian Center Journal (MICJO)
Journal page is available to
https://e-jurnal.jurnalcenter.com/index.php/micjo
Email: admin@jurnalcenter.com
E.ISSN. 3032-2472
Vol. 1 No. 3 Edisi
Juli 2024

Restful, dengan serangkaian kata kerja yang terhubung ke metode HTTP, seperti yang ditunjukkan di bawah ini:

1. Mengubah item yang sudah ada dalam koleksi (put).
2. Mengambil satu item atau koleksi (get).
3. Menghapus item dalam koleksi (delete).
4. Menambahkan item ke dalam koleksi (post).



Gambar 2. Alur Data Restful API

Konsistensi dalam menjalankan berbagai operasi dalam berbagai aplikasi memberikan manfaat adopsi standar. Keempat kata kerja HTTP yang berbeda, sesuai dengan fungsi dasar CRUD yang umum digunakan dalam aplikasi modern, memperjelas pemahaman akan konsekuensi dari tindakan yang dilakukan melalui antarmuka yang beragam. Sistem dengan arsitektur microservice dapat dihasilkan menggunakan teknologi Java SpringBoot. Dalam kerangka kerja SpringBoot, terdapat fitur khusus untuk pengembangan layanan web, terutama dalam bentuk Restful API. Untuk membangun sistem berbasis arsitektur microservice dengan menggunakan framework SpringBoot, diperlukan penggunaan anotasi-anotasi yang mengelola permintaan dan respons HTTP sebagai bagian dari proses transaksi layanan, seperti :

1. Anotasi@RequestMapping, yang memetakan permintaan HTTP ke metode pengendali Spring MVC dan REST untuk menangani berbagai jenis permintaan.
2. Anotasi@Controller digunakan untuk menandai kelas sebagai pengendali titik akhir.
3. Anotasi@ResponseStatus menentukan nilai balik dari permintaan yang dikirimkan klien ke server.
4. Anotasi @RestController mempermudah dengan menggabungkan @Controller dan @ResponseBody.
5. Anotasi @GetMapping menangani permintaan GET.
6. Anotasi @PostMapping menangani permintaan POST.
7. Anotasi @PutMapping menangani permintaan PUT.
8. Anotasi @DeleteMapping menangani permintaan DELETE.

Dengan menerapkan arsitektur microservices dalam proses migrasi data menggunakan pendekatan SDLC, khususnya metodologi Rapid Application Development (RAD), penelitian ini diharapkan dapat menyelesaikan masalah ini dengan cepat dan efisien. Melalui inovasi ini, Dwidayatour berharap dapat meningkatkan kualitas layanan, keakuratan data, dan responsibilitas pelanggan dan mitra bisnisnya.

2. METODE PENELITIAN

Dalam pengembangan proyek penelitian untuk migrasi sistem ke arsitektur microservice, pendekatan SDLC menekankan kualitas pengembangan secara menyeluruh dan efisiensi. Salah satu metode pengembangan sistem dalam waktu singkat (RAD) mengadopsi orientasi komponen dalam pengembangan perangkat lunak dengan tahapan yang terstruktur. Proses migrasi sistem Dwidayatour ke arsitektur microservice dilakukan dalam beberapa tahapan:

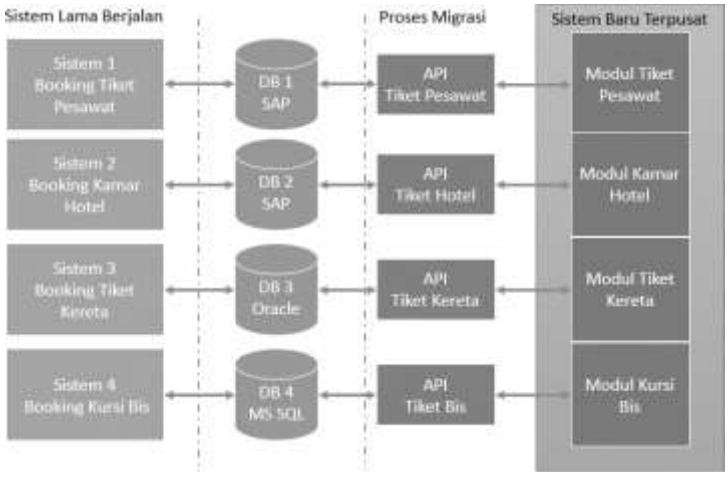
Multidisciplinary Indonesian Center Journal (MICJO)
Journal page is available to
https://e-jurnal.jurnalcenter.com/index.php/micjo
Email: admin@jurnalcenter.com
E.ISSN. 3032-2472
Vol. 1 No. 3 Edisi
Juli 2024

1. Analisis & Desain Cepat (Quick Design): Tim melakukan analisis menyeluruh terhadap parameter dan layanan transaksi yang ada di sistem lama untuk mengalihkannya ke sistem terpusat baru yang menggunakan arsitektur microservice.
2. Siklus Prototipe:
   a. Pengembangan: Desain analisis awal diprogram di frontend dan backend, lalu diuji untuk memastikan konsistensi dengan sistem lama.
   b. Demonstrasi: Pengguna melihat hasil migrasi data dari sistem lama ke sistem baru.
      a. Perbaikan: Berdasarkan feedback pengguna, program diubah dan siklus pengembangan diulang hingga keinginan pengguna terpenuhi.
3. Pengujian. Alat otomatisasi kualitas digunakan untuk mempercepat proses pengujian dengan skrip yang ditanam dalam modul pemrograman.
4. Setelah uji coba, setiap modul dipasang ke server sesuai dengan arsitektur microservice. Proses migrasi dapat dilakukan secara bersamaan dengan sistem lama tanpa masalah, dan pertukaran data dapat dimonitor secara langsung.

Pendekatan metodologi Rapid Application Development (RAD) memungkinkan proses migrasi dilakukan secara bersamaan dengan sistem lama. Ini memungkinkan sistem baru yang dikembangkan berjalan beriringan dengan sistem lama tanpa gangguan operasional. Selain itu, arsitektur microservice memastikan integrasi yang efektif antara sistem yang berbeda dengan fokus pada kualitas dan efisiensi pengembangan.

3. HASIL DAN PEMBAHASAN

Hasil penelitian didokumentasikan melalui penggunaan metodologi Rapid Application Development (RAD) dalam migrasi sistem terpusat yang terstruktur melalui berbagai tahapan yang sesuai dengan standar RAD itu sendiri.



Gambar 3. Alur Layanan API Baru (Terpusat) Sistem Migrasi Data

Tabel 1. API Tiket Bis

Modul	SubModul	@RestController	@RequestMapping	Method
Tiket Bis	Master Bis	mstBisController()	createBis()	Post
Tiket Bis	Master Bis	mstBisController()	updateBis()	Put
Tiket Bis	Master Bis	mstBisController()	DeleteBis()	Delete
Tiket Bis	Master Bis	mstBisController()	listBis()	Get
Tiket Bis	Master Terminal	mstTerminalController()	createTerminal()	Post
Tiket Bis	Master Terminal	mstTerminalController()	updateTerminal ()	Put
Tiket Bis	Master Terminal	mstTerminalController()	DeleteTerminal ()	Delete
Tiket Bis	Master Terminal	mstTerminalController()	listTerminal ()	Get

Multidisciplinary Indonesian Center Journal (MICJO)
Journal page is available to
https://e-jurnal.jurnalcenter.com/index.php/micjo
Email: admin@jurnalcenter.com
E.ISSN. 3032-2472
Vol. 1 No. 3 Edisi
Juli 2024

Tiket Bis	TransaksiTiket Bis	trxTiketBisController()	createTiketBis()	Post
Tiket Bis	TransaksiTiket Bis	trxTiketBisController()	updateTiketBis()	Put
Tiket Bis	TransaksiTiket Bis	trxTiketBisController()	DeleteTiketBis()	Delete
Tiket Bis	TransaksiTiket Bis	trxTiketBisController()	listTiketBis()	Get

Tabel 2. API Tiket Pesawat

Modul	SubModul	@RestController	@RequestMapping	Method
Tiket Pesawat	Master Maskapai	mstMaskapaiController()	createKota()	Post
Tiket Pesawat	Master Maskapai	mstMaskapaiController()	updateKota()	Put
Tiket Pesawat	Master Maskapai	mstMaskapaiController()	DeleteKota()	Delete
Tiket Pesawat	Master Maskapai	mstMaskapaiController()	listKota()	Get
Tiket Pesawat	Master Bandara	mstBandaraController()	createBandara()	Post
Tiket Pesawat	Master Bandara	mstBandaraController()	updateBandara()	Put
Tiket Pesawat	Master Bandara	mstBandaraController()	DeleteBandara()	Delete
Tiket Pesawat	Master Bandara	mstBandaraController()	listBandara()	Get
Tiket Pesawat	TransaksiTiket Pesawat	trxTiketPesawatController()	createTiketPesawat()	Post
Tiket Pesawat	TransaksiTiket Pesawat	trxTiketPesawatController()	updateTiketPesawat()	Put
Tiket Pesawat	TransaksiTiket Pesawat	trxTiketPesawatController()	DeleteTiketPesawat()	Delete
Tiket Pesawat	TransaksiTiket Pesawat	trxTiketPesawatController()	listTiketPesawat()	Get

Tabel 3. API Kamar Hotel

Modul	SubModul	@RestController	@RequestMapping	Method
Kamar Hotel	Master Kota	mstKotaController()	createKota()	Post
Kamar Hotel	Master Kota	mstKotaController()	updateKota()	Put
Kamar Hotel	Master Kota	mstKotaController()	DeleteKota()	Delete
Kamar Hotel	Master Kota	mstKotaController()	listKota()	Get
Kamar Hotel	Master Hotel	mstHotelController()	createHotel()	Post
Kamar Hotel	Master Hotel	mstHotelController()	updateHotel()	Put
Kamar Hotel	Master Hotel	mstHotelController()	DeleteHotel()	Delete
Kamar Hotel	Master Hotel	mstHotelController()	“listHotel()	Get
Kamar Hotel	TransaksiKamar Hotel	trxKamarHotelController()	createKamarHotel ()	Post
Kamar Hotel	TransaksiKamar Hotel	trxKamarHotelController()	updateKamarHotel ()	Put
Kamar Hotel	TransaksiKamar Hotel	trxKamarHotelController()	DeleteKamarHotel ()	Delete
Kamar Hotel	TransaksiKamar Hotel	trxKamarHotelController()	listKamarHotel ()	Get

Tabel 4. API Tiket Kereta

Modul	SubModul	@RestController	@RequestMapping	Method
Tiket Kereta	Master Stasiun	mstStasiunController()	createStasiun()	Post
Tiket Kereta	Master Stasiun	mstStasiunController()	updateStasiun ()	Put
Tiket Kereta	Master Stasiun	mstStasiunController()	DeleteStasiun ()	Delete
Tiket Kereta	Master Stasiun	mstStasiunController()	listStasiun()	Get
Tiket Kereta	Master Kereta	mstKeretaController()	createKereta()	Post
Tiket Kereta	Master Kereta	mstKeretaController()	updateKereta()	Put
Tiket Kereta	Master Kereta	mstKeretaController()	DeleteKereta()	Delete
Tiket Kereta	Master Kereta	mstKeretaController()	listKereta()	Get
Tiket Kereta	TransaksiTiket Kereta	trxTiketKeretaController()	createTiketKereta()	Post
Tiket Kereta	TransaksiTiket Kereta	trxTiketKeretaController()	updateTiketKereta()	Put
Tiket Kereta	TransaksiTiket Kereta	trxTiketKeretaController()	DeleteTiketKereta()	Delete
Tiket Kereta	TransaksiTiket Kereta	trxTiketKeretaController()	listTiketKereta()	Get

Tahap Cicle Prototype:

Pada tahap ini, revisi prototype migrasi data pengguna dijelaskan sebagai beriku

Multidisciplinary Indonesian Center Journal (MICJO)
Journal page is available to
https://e-jurnal.jurnalcenter.com/index.php/micjo
Email: admin@jurnalcenter.com
E.ISSN. 3032-2472
Vol. 1 No. 3 Edisi
Juli 2024

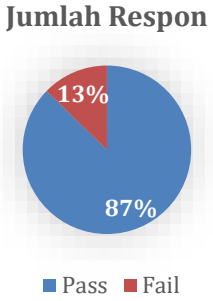
Tabel 5. Prototipe API Sistem Terpusat (Versi Revisi)

Modul	SubModul	@RestController	Cycles
Tiket Bis	Master Bis	mstBisController()	4
Tiket Bis	Master Terminal	mstTerminalController()	4
Tiket Bis	Transaksi Tiket Bis	trxTiketBisController()	4
Tiket Pesawat	Master Maskapai	mstMaskapaiController()	4
Tiket Pesawat	Master Bandara	mstBandaraController()	4
Tiket Pesawat	Transaksi Tiket Pesawat	trxTiketPesawatController()	4
Kamar Hotel	Master Kota	mstKotaController()	4
Kamar Hotel	Master Hotel	mstHotelController()	4
Kamar Hotel	Transaksi Kamar Hotel	trxKamarHotelController()	4
Tiket Kereta	Master Stasiun	mstStasiunController()	4
Tiket Kereta	Master Kereta	mstKeretaController()	4
Tiket Kereta	Transaksi Tiket Kereta	trxTiketKeretaController()	4

Penjelasan:
Jumlah putaran perbaikan dari tiga tahap pengembangan, demonstrasi, dan penyempurnaan digambarkan dalam lingkaran.

Tahap Pengujian

Pada Tahap Pengujian, alat bantu QA Postman digunakan untuk memeriksa URL setiap modul layanan API. Hasil uji coba menunjukkan bahwa rasio keberhasilan skenario pengujian lebih tinggi daripada rasio kegagalan. Gambar 6 menunjukkan perbandingan rasio ini.



Gambar 4. Diagram Perbandingan Hasil Uji API Services
Sumber : Dokumentasi Pribadi

Tabel 6. Hasil Uji API Services

Hasil Test	@ResponseStatus	Jumlah Respon	Jumlah per Hasil Test
Fail	400 Bad Request	1	1
Fail	403 Forbidden	2	3
Fail	405 Method Not Allowed	2	4
Fail	500 Internal Server Error	2	7
Pass	36	36	36
Pass	12	12	48

Selama tahap implementasi, hasil migrasi sistem yang sudah berjalan ke sistem terpusat baru menggunakan arsitektur microservices melalui metodologi Rapid Application Development (RAD) tercatat dalam waktu singkat tiga bulan. Tabel 7 berisi detailnya.

Multidisciplinary Indonesian Center Journal (MICJO)
Journal page is available to
https://e-jurnal.jurnalcenter.com/index.php/micjo
Email: admin@jurnalcenter.com
E.ISSN. 3032-2472
Vol. 1 No. 3 Edisi
Juli 2024

Tabel 7. TimeLine *Deployment* Sistem Baru (Sistem Terpusat)

Modul	Februari 2024	Maret 2024	April 2024
Tiket Kereta			
Tiket Pesawat			
Tiket Bis			
Kamar Hotel			

Keterangan:

- Tiket Pesawat Modul diluncurkan selama dua bulan, dari Februari hingga Maret 2024.
- Tiket Kereta Modul juga diluncurkan selama satu bulan, pada Maret 2024.
- Modul Kamar Hotel diaktifkan selama dua bulan dari Maret hingga April 2024.
- Modul Tiket Bis juga diaktifkan selama satu bulan pada April 2024.

Sebagai informasi, estimasi awal untuk proyek migrasi ini adalah 5 bulan, dengan detail yang tercantum dalam tabel 8.

Tabel 8. Perbandingan Waktu Migrasi

"Modul"	"Estimasi Awal Migrasi tanpaRAD (Satuan: Bulan)"	"Hasil Akhir Migrasi dengan RAD(Satuan: Bulan)"
Kamar Hotel	4	2
Tiket Bis	2	1
Tiket Pesawat	5	2
Tiket Kereta	3	1
Batas Lama Migrasi	5	3

Keterangan:

- Waktu estimasi awal untuk migrasi dihitung dari jumlah bulan yang dibutuhkan untuk sistem migrasi.
- Waktu estimasi untuk migrasi awal tanpa Rapid Application Development (RAD) adalah lima bulan, dengan angka 5 dihitung dari jumlah bulan maksimal di antara semua modul migrasi.
- Waktu estimasi untuk migrasi hasil akhir dengan RAD adalah tiga bulan, dengan angka 3 dilihat dari data dalam tabel 7.

4. KESIMPULAN

Hasil penelitian ini mengatakan bahwa menggunakan arsitektur microservices dalam migrasi sistem memiliki efek yang sangat baik. Dengan arsitektur microservices, migrasi dari sistem yang berjalan secara paralel ke sistem terpusat baru juga dapat dilakukan dengan lancar. Selain itu, arsitektur tersebut efektif dalam menangani tantangan integrasi berbagai sistem dengan berbagai teknologi ke dalam satu sistem terpusat. Metodologi RAD terbukti dapat mempercepat waktu migrasi tanpa mengorbankan kualitas. Peneliti menganjurkan penggunaan arsitektur microservices dan metodologi Rapid Application Development (RAD) saat mengembangkan sistem informasi baru dari awal.

Diharapkan penelitian ini akan memberikan pemahaman yang lebih baik tentang bagaimana kedua metode ini dapat digunakan dalam pengembangan sistem informasi yang baru. Penelitian ini juga akan menjadi rujukan bagi peneliti yang ingin menggunakan atau mengubah metode ini dalam studi kasus lainnya.

E.ISSN. 3032-2472
Vol. 1 No. 3 Edisi
Juli 2024

5. DAFTAR PUSTAKA

Jogyanto. (2011). Sistem Informasi Manajemen (Edisi 3, Cetakan 2). Jakarta: Universitas Terbuka, Kementerian Pendidikan dan Kebudayaan.

Supratikta, H., Taryo, T., & Aziz, F. (2021). *Sistem Informasi Manajemen.*

Sutabri, T. (2016). Sistem Informasi Manajemen (Edisi revisi, Edisi II, Cetakan 1). Yogyakarta: Andi.