SOPINGI SOPINGI

Sopingi-JSINBIS.docx

Forum Perpustakaan Perguruan Tinggi Indonesia Jawa Tengah

Document Details

Submission ID
trn:oid:::3618:93972794

Submission Date
May 2, 2025, 1:23 PM GMT+7

Download Date
May 2, 2025, 1:26 PM GMT+7

File Name
Sopingi-JSINBIS.docx

File Size
1.7 MB

6 Pages

2,512 Words

17,181 Characters

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text

Exclusions

- 7 Excluded Matches

Top Sources

7% Internet sources
3% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Top Sources

7% Internet sources
3% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
docs.confluent.io		1%
2	Internet	
dev3lop.com		<1%
3	Internet	
eprints.uniska-bjm.ac.id		<1%
4	Internet	
id.scribd.com		<1%
5	Internet	
jurnal.politeknik-kebumen.ac.id		<1%
6	Internet	
rizkafithriani95.wordpress.com		<1%
7	Publication	
Herliyani Hasanah. "Pemanfaatan Digital Marketing Menggunakan Website dan ...		<1%
8	Publication	
Muhammad Irsyad Yanuardi, Aminudin Aminudin, Mahar Faiqurahman. "Analisis ...		<1%
9	Internet	
www.aspecto.io		<1%
10	Internet	
e-jurnal.dharmawacana.ac.id		<1%
11	Internet	
journal.uin-alauddin.ac.id		<1%

12 Internet

pt.scribd.com <1%

13 Internet

ijsrcseit.com <1%

14 Publication

Κακομήτας, Δημήτριος | Kakomitas, Dimitrios-Stylianos. "Cloud Gateways for Het... <1%

Jurnal Sistem Informasi Bisnis 01(2024)

On-line: http://ejournal.undip.ac.id/index.php/jsinbis/article/view/id_paper

Pendekatan *Event Driven Architecture* untuk Sinkronisasi *Real Time* NeoFeeder PDDIKTI

Sopingi[a,*], Sri Sumarlinda[b]

[a] Universitas Duta Bangsa Surakarta
[b] Universitas Duta Bangsa Surakarta

Naskah Diterima : 5 November 2019; Diterima Publikasi : 25 Agustus 2021
DOI : 10.21456/vol14iss1pp1-9

Abstract

The Higher Education Database has an important role as an information center for higher education in Indonesia. NeoFeeder is present as middleware to bridge the differences between academic information systems between universities and the PDDIKTI centralized database. Current NeoFeeder implementations still use batch systems or manual triggers, which can cause delays in data updates, especially if the data is large or there are sudden changes. To overcome this, a system architecture is needed that is more adaptive, scalable and responsive to changes in data. This research aims to produce an Event-Driven Architecture based integration model that can increase the efficiency of data synchronization between the internal academic system and PDDIKTI NeoFeeder. This research uses a Rapid Application Development approach. The research results show that the system is capable of sending more than 1,000 data per second with an average latency of 4.05 ms and a response time of under 40 ms. In conclusion, the Event Driven Architecture approach is effective in helping synchronize academic data to NeoFeeder in real time

Keywords: Event Driven, NeoFeeder, Kafka Apache, Synchronization, Real Time

Abstrak

Pangkalan Data Pendidikan Tinggi memiliki peran penting sebagai pusat informasi pendidikan tinggi di Indonesia. NeoFeeder hadir sebagai middleware untuk menjembatani perbedaan sistem informasi akademik antar perguruan tinggi dengan database terpusat PDDIKTI. Implementasi NeoFeeder saat ini masih menggunakan sistem *batch* atau pemicu manual, yang dapat menyebabkan keterlambatan pembaruan data, terutama jika data besar atau ada perubahan mendadak. Untuk mengatasi hal ini, diperlukan arsitektur sistem yang lebih adaptif, scalable, dan responsif terhadap perubahan data. Penelitian ini bertujuan untuk menghasilkan model integrasi berbasis Event-Driven Architecture yang dapat meningkatkan efisiensi sinkronisasi data antara sistem akademik internal dan NeoFeeder PDDIKTI. Penelitian ini menggunakan pendekatan Rapid Application Development. Hasil penelitian menunjukkan bahwa sistem mampu mengirimkan lebih dari 1.000 data per detik dengan latensi rata-rata 4,05 ms dan response time di bawah 40 ms. Kesimpulannya, pendekatan Event Driven Architecture efektif dalam membantu sinkronisasi data akademik ke NeoFeeder secara real time.

Kata kunci: Event Driven, NeoFeeder, Kafka Apache, Sinkronisasi, Waktu Nyata.

1. Pendahuluan

Pangkalan Data Pendidikan Tinggi memegang peranan krusial dalam ekosistem pendidikan tinggi di Indonesia, berfungsi sebagai pusat penyimpanan dan diseminasi informasi yang komprehensif mengenai berbagai aspek penyelenggaraan pendidikan tinggi Dalam konteks ini, NeoFeeder muncul sebagai solusi *middleware* yang dirancang untuk menjembatani kesenjangan antara sistem informasi akademik yang beragam di setiap perguruan tinggi dengan *database* terpusat PDDIKTI (Auliana & Nurasiah, 2018).

NeoFeeder diharapkan dapat mempermudah dan mempercepat proses pelaporan data, sekaligus meningkatkan akurasi dan konsistensi data yang dilaporkan. Sistem informasi yang mumpuni akan membuat kinerja suatu instansi terlaksana dengan baik, bisa menangani berbagai pengolahan data dengan memakai teknologi informasi, dan memenuhi kebutuhan pemakai sistem mengenai gambaran yang jelas tentang rancangan sistem yang akan dibuat serta diimplementasikan (Rafiezah Rizcha & Yaakub, 2023). Berdasarkan manual penggunaan *webservice* Neofeeder setiap sinkronisasi data dari sistem informasi akademik ke Neofeeder dilakukan setiap *record* belum mampu singkronisasi *record* secara masal, sehingga proses sinkronisasi data antara sistem internal kampus dengan NeoFeeder menghadapi tantangan, terutama dalam hal keterlambatan, inkonsistensi data, dan integrasi *real-time*.

Implementasi integrasi NeoFeeder saat ini masih bersifat *batch-oriented* atau bergantung pada pemicu manual, yang dapat menyebabkan keterlambatan pembaruan data, terutama ketika jumlah data tinggi

*) Corresponding author: sopingi@udb.ac.id

Jurnal Sistem Informasi Bisnis 01(2024)

On-line: http://ejournal.undip.ac.id/index.php/jsinbis/article/view/id_paper

atau terjadi perubahan mendadak pada data akademik. Sistem integrasi yang tidak responsif dapat menimbulkan risiko kesalahan pelaporan dan keterlambatan pelaporan di PDDIKTI.

Untuk mengatasi tantangan tersebut, diperlukan pendekatan arsitektur sistem yang lebih adaptif, skalabel, dan mampu merespons perubahan data secara otomatis. *Event Driven Architecture* (EDA) muncul sebagai solusi yang relevan. Dengan EDA, setiap perubahan data diproses sebagai suatu peristiwa (*event*) yang memicu layanan tertentu untuk menangani sinkronisasi secara otomatis dan *real-time*, tanpa perlu intervensi manual atau jadwal sinkronisasi terjadwal. Arsitektur *microservice* berbasis *event* menggabungkan *scalability*, *maintainability*, kemudahan penerapan, *resilience* dan *reusability (Ubur, 2023)*

Melalui penelitian dan pengembangan sistem ini, diharapkan dapat dihasilkan model atau prototipe integrasi berbasis *Event-Driven Architecture* yang mampu meningkatkan efisiensi dan efektivitas dalam proses sinkronisasi data antara sistem akademik internal dan NeoFeeder PDDIKTI. Dengan demikian, sistem informasi perguruan tinggi dapat bergerak menuju transformasi digital yang lebih modern, *real-time* dan responsif terhadap dinamika manajemen pendidikan tinggi.

2. Kerangka Teori

2.1. Event-Driven Architecture

Event-Driven Architecture adalah pola arsitektur perangkat lunak di mana sistem dibangun berdasarkan produksi, deteksi, konsumsi, dan reaksi terhadap peristiwa (*event*). Setiap event mencerminkan perubahan keadaan yang penting, seperti input dari pengguna atau perubahan pada basis data (Amazon Web Services, n.d.)

Menurut survei global oleh Solace pada tahun 2021, 72% organisasi di seluruh dunia telah menggunakan EDA dalam berbagai tingkat kematangan. Namun, hanya 13% yang mencapai tingkat kematangan penuh di mana EDA diterapkan secara luas di seluruh organisasi. Manfaat utama yang dirasakan termasuk peningkatan responsivitas aplikasi (46%), peningkatan pengalaman pelanggan (44%), dan kemampuan untuk merespons perubahan secara real-time (43%) (Solace Corporation, n.d.).

2.2. Web Service

Web service adalah komponen perangkat lunak yang memfasilitasi komunikasi dan interaksi antar aplikasi melalui jaringan, umumnya menggunakan protokol HTTP. Dirancang untuk mendukung interoperabilitas sistem yang heterogen dengan memanfaatkan standar terbuka seperti XML, JSON, SOAP, dan REST (Sopingi et al., 2020)

2.3. NeoFeeder

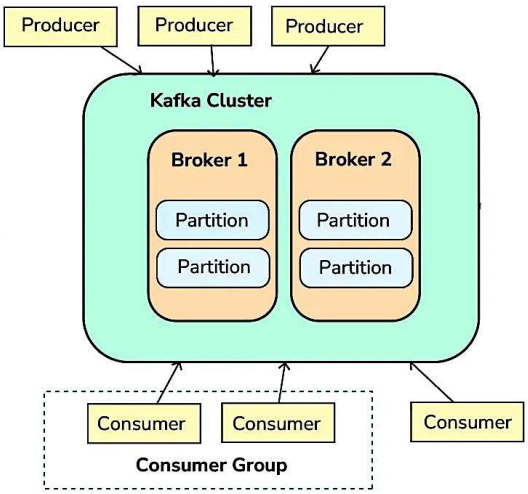
Neo Feeder merupakan aplikasi yang diluncurkan pada 25 Februari 2022 untuk membantu perguruan tinggi mengelola dan menyinkronkan data ke PDDikti (2022). Aplikasi ini diperlukan karena pelaporan data harus sesuai dengan Permenristekdikti Nomor 61 Tahun 2016 (Brawijaya - & Widodo -, 2023).

2.4. Event Streaming

Event streaming adalah praktik menangkap data secara real-time dari berbagai sumber. Konsep ini memungkinkan organisasi untuk mengakses dan memproses data saat data tersebut dihasilkan, bukan setelah disimpan di *database* (Apache, n.d.)

Salah satu *platform* aplikasi yang dapat melakukan *event streaming* adalah Apache Kafka yang bersifat *open source* sehingga dapat dikembangkan untuk aplikasi pendistribusian data secara *real time* di dalam lingkungan dengan ketersediaan tinggi dan toleransi kesalahan rendah (Bucur et al., 2020).

Pada Gambar 1 disajikan arsitektur Kafka dimana terdiri dari kafka cluster yaitu sekelompok broker yang bekerja bersama untuk menyediakan layanan terdistribusi (Hesse et al., 2020). Producers sebagai pengirim pesan dan Consumers sebagai penerima pesan.



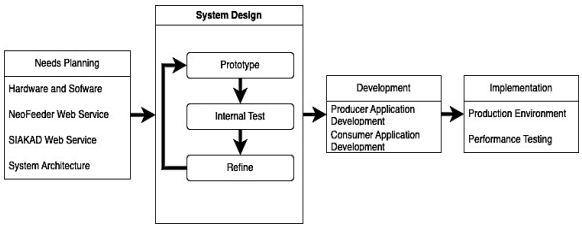
Gambar 1. Arsitektur Kafka

3. Metode

Penelitian ini fokus pada sinkronisasi antara sistem informasi akademik yang sudah ada dengan aplikasi NeoFeeder PDDIKTI dengan menggunakan *event driven architecture*. Dalam penelitian ini menggunakan pendekatan *Rapid Aplication Development* (RAD). Pendekatan RAD memfasilitasi percepatan transformasi kebutuhan pengguna menjadi solusi perangkat lunak dengan menggabungkan desain, pengembangan, dan pengujian menjadi tahap yang lebih singkat (Živanović et al., 2020). Tahapan penelitian disajikan pada Gambar 2.

Jurnal Sistem Informasi Bisnis 01(2024)

On-line: http://ejournal.undip.ac.id/index.php/jsinbis/article/view/id_paper



Gambar 3. Tahap Penelitian dengan Pendekatan RAD

Terdapat 4 (empat) tahap utama dalam penelitian yang dilakukan yaitu:

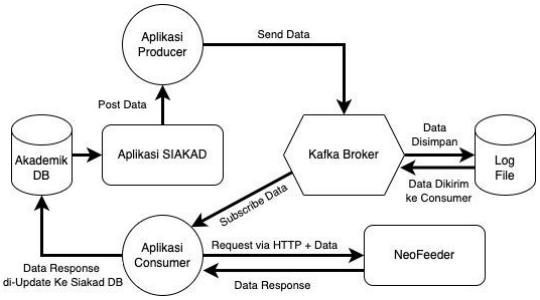
a. Perencanaan Kebutuhan
   Penelitian ini menggunakan 3 server yaitu server SIAKAD, server NeoFeeder dan server untuk Apache Kafka sebagai *event streaming*. Masing-masing server terhubung melalui jaringan internet.
b. Desain Sistem
   Prototipe membantu dalam pengujian awal terhadap fungsi utama sistem *event streaming* dalam melakukan sinkronisasi secara *real time* sehingga potensi kesalahan dapat diperbaiki. Pada tahap ini prototipe dijalankan dalam bentuk perintah-perintah *console* dari Apache Kafka baik sebagai *producer* dan *consumer*.
c. Pengembangan
   Pembuatan aplikasi untuk *producer* dan *consumer* dikembangkan dengan bahasa pemrograman NodeJS yang terintegrasi dengan SIAKAD menggunakan koneksi *database* dan integrasi NeoFeeder menggunakan *web service.*
d. Implementasi
   Aplikasi diimplementasikan di lingkungan sebenarnya sehingga dapat melakukan sinkronisasi dari SIAKAD ke NeoFeeder secara otomatis dan *real time*, pada tahap ini juga dilakukan pengujian terhadap *performance*. Untuk pengujian performance menggunakan fungsi *org.apache.kafka.tools.ProducerPerformance* dan *org.apache.kafka.tools.ConsumerPerformance*

4. Hasil dan Pembahasan

4.1. Hasil

Adapun hasil penelitian yang sudah dilakukan adalah sebagai berikut:

a. Arsitektur Sistem
   Sebagai gambaran sinkronisasi NeoFeeder dengan pendekatan *event driven architecture*, penulis membuat arsitektur cara kerja sistem sebagai berikut:



Gambar 3. Arsitektur Sinkronisasi

Setiap ada perubahan pada database akademik, SIAKAD akan mengirimkan ke Kafka Broker melalui aplikasi *producer* tanpa harus terhubung langsung ke NeoFeeder. Kafka broker akan menyimpan data ke dalam *log file* dan antri untuk dikirim ke aplikasi *consumer*. Setiap ada data yang dikirim ke aplikasi *consumer* maka secara otomatis aplikasi *consumer* akan melakukan pengiriman data ke NeoFeeder, setelah mendapatkan respon dari NeoFeeder aplikasi consumer akan melakukan update secara langsung ke database akademik.

Mekanisme ini akan mempercepat proses sinkronisasi di SIAKAD, karena SIAKAD tidak perlu menunggu respon dari NeoFeeder, semua proses sinkronisasi data akan menjadi tanggung jawab aplikasi *consumer*. SIAKAD juga dapat melakukan sinkronisasi dengan banyak data sekaligus meskipun di aplikasi *consumer* tetap mengirim satu data ke NeoFeeder untuk setiap *request*.

b. Pengembangan Sistem
   Aplikasi *producer* dan *consumer* dikembangkan dengan menggunakan *Framework ExpressJS*. ExpressJS memiliki kemampuan yang lebih efisien untuk *routing*, menangani *request* HTTP, dan keamanan *middleware* (Grudniak & Dzieńkowski, 2021). Sedangkan untuk terhubung sebagai client dari Kafka penulis menggunakan paket *KafkaJS* dengan kelebihan ringan, fleksibel dan tanpa dependensi (Ornelas, n.d.)

Berikut adalah potongan *script* pada inisialisasi *producer client* dari Kafka:

```
const { Kafka } = require('kafkajs');

const kafka = new Kafka({
  clientId: 'sopingi-kafka-producer',
  brokers: ['ip_server_kafka:9094']
})
```

Script untuk proses mengirim data ke Kafka broker dari aplikasi *producer:*

```
const producer_siakad = kafka.producer()
const topic_siakad = 'sync-siakad'
```

Jurnal Sistem Informasi Bisnis 01(2024)

On-line: http://ejournal.undip.ac.id/index.php/jsinbis/article/view/id_paper

```
const connectKafka = async () => {
  await producer_siakad.connect()
};
connectKafka().catch(console.error);

router.post('/', async function(rq, rs, n)
{

 try {
  var data = JSON.stringify(rq.body)
  await producer_siakad.send({
   topic: 'sync-siakad',
   compression: CompressionTypes.GZIP,
   messages: [{ value: data }],
  });
      rs.send({ pesan:'Berhasil'})
 } catch (e) {
  rs.status(500).send({pesan:'gagal' })
 }
}
```

Sedangkan potongan *script* pada inisialisasi *consumer client* dari Kafka:

```
const { Kafka } = require('kafkajs');

const kafka = new Kafka({
  clientId: 'sopingi-kafka-consumer',
  brokers: ['ip_server_kafka:9094']
})
```

Script untuk proses *subscribe* data dari Kafka broker ke aplikasi *consumer*

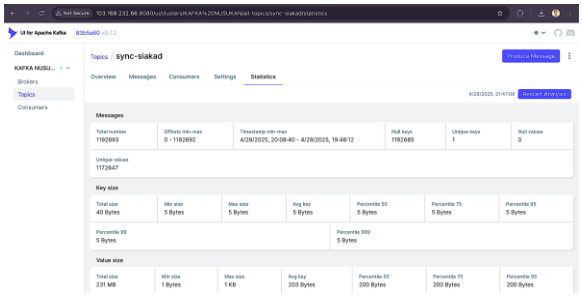
```
const consumer_siakad = kafka.consumer({ groupId: 'siakad' });
const topic_siakad = 'sync-siakad'

const connectKafka = async () => {
 await consumer_siakad.subscribe({
        topic:topic_siakad,
        fromBeginning: true
 });

 await consumer_siakad.run({
  eachMessage: async ({ data }) => {
   var json = JSON.parse(data.value);
   var neoFeeder = await runWs(json);
})
 };

connectKafka().catch(console.error);
```

Monitoring hasil sinkronisasi dapat dilihat melalui aplikasi Kafka UI seperti pada Gambar 4. Melalui aplikasi Kafka UI dapat dilihat history data akademik yang disinkronkan ke NeoFeeder.



Gambar 4. Aplikasi Kafka UI

c. Pengujian *Performance*

Pengujian *performance* terdiri dari pengujian latensi dalam pengiriman data dari *producer*, jumlah data yang dapat dikirim ke *consumer* dan *response time* ExpressJS dalam mengeksekusi pengiriman data dari SIAKAD.

Pengujian latensi *producer* menggunakan org.apache.kafka.tools.ProducerPerformance yaitu *class* bawaan di Apache Kafka. Pengujian *producer* dilakukan dengan 10.000 data dengan *throughput* 200 data / per detik. Hasil pengujian *producer* dapat dilihat di Gambar 5 berikut:

```
Number of messages read: 1
1002 records sent, 200.2 records/sec (0.22 MB/sec), 6.3 ms avg latency, 390.0 ms max latency.
1002 records sent, 200.0 records/sec (0.22 MB/sec), 4.0 ms avg latency, 19.0 ms max latency.
1001 records sent, 200.0 records/sec (0.22 MB/sec), 3.8 ms avg latency, 7.0 ms max latency.
1001 records sent, 200.2 records/sec (0.22 MB/sec), 4.0 ms avg latency, 7.0 ms max latency.
1001 records sent, 199.9 records/sec (0.22 MB/sec), 4.0 ms avg latency, 15.0 ms max latency.
1001 records sent, 200.0 records/sec (0.22 MB/sec), 3.8 ms avg latency, 7.0 ms max latency.
1003 records sent, 200.2 records/sec (0.22 MB/sec), 3.6 ms avg latency, 7.0 ms max latency.
1000 records sent, 199.9 records/sec (0.22 MB/sec), 3.5 ms avg latency, 13.0 ms max latency.
1001 records sent, 200.1 records/sec (0.22 MB/sec), 3.5 ms avg latency, 6.0 ms max latency.
10000 records sent, 199.9 records/sec (0.22 MB/sec), 3.98 ms avg latency, 390.00 ms max latency,
 6 ms 50th, 7 ms 95th, 8 ms 99th, 25 ms 99.9th.
$
```

Gambar 5. Hasil Pengujian Latensi *Producer*

Berdasarkan hasil tersebut saat *producer* mengirim 10.000 data dengan *throughput* 200 didapatkan rata-rata latensi 4,05 ms.

Pengujian *consumer* menggunakan org.apache.kafka.tools.ConsumerPerformance pengujian *consumer* dilakukan untuk mengukur jumlah data yang diterima selama 5 detik dengan hasil yang disajikan pada Tabel 1 .

Tabel 1. Hasil Pengujian *Customer Performance*

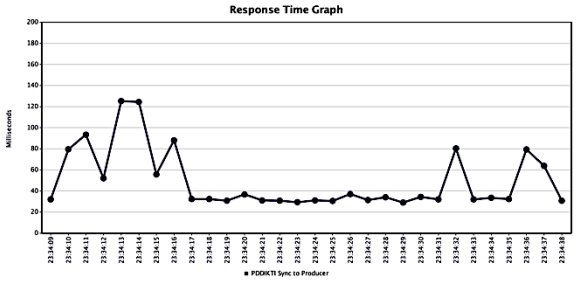
Header	Value
data.consumed.in.MB	1,9913
MB.sec	0,5654
data.consumed.in.nMsg	10372
nMsg.sec	2944,9177
rebalance.time.ms	3418
fetch.time.ms	104
fetch.MB.sec	19,1475
fetch.nMsg.sec	99730,7692

Berdasarkan tabel 1 terlihat bahwa *consumer* mampu menerima rata-rata 2.944,9 pesan per detik dan telah diterima 10.372 pesan selama 5 detik.

Pengujian ExpressJS dalam mengeksekusi pengiriman data dari SIAKAD menggunakan aplikasi JMeter yang mampu menguji performa *response time* pada aplikasi *web service* (Sopingi & Wulandari, 2023). Pengujian *response time* dilakukan dengan 100 *request* dalam dalam waktu 30 detik dihasilan grafik yang disajikan pada Gambar 6.

Jurnal Sistem Informasi Bisnis 01(2024)

On-line: http://ejournal.undip.ac.id/index.php/jsinbis/article/view/id_paper



Gambar 6. Grafik Pengujian *Response Time*

Berdasarkan hasil grafik pada Gambar 6 dapat diketahui bahwa *response time* paling lama 125 ms dan rata-rata dibawah 40 ms.

4.1. Pembahasan

Berdasarkan hasil penelitian dan pengujian bahwa pendekatan *event driven architecture* mampu memberikan performa yang sangat baik dan membantu sinkronisasi data akademik secara *real time* dan otomatis tanpa manual *trigger*. Pengiriman data memiliki performa yang sangat baik mampu mengirim data di atas 1.000 data per detik dengan rata-rata latensi 4,05 ms dan *response time* rata-rata di bawah 40 ms. Hasil ini didukung dari penelitian terdahulu bahwa *platform event streaming* seperti Apache Kafka memiliki *throughput* yang tinggi (Köstler et al., 2021), begitu juga dengan penelitian Luan Lazzari, dkk menyatakan bahwa Aplikasi berbasis *event driven* memberikan respons yang lebih cepat terhadap perubahan data (Lazzari & Farias, 2023).

Hasil pengujian *response time* menunjukan *Framework ExpressJS* memiliki performa yang baik untuk implementasi sinkronisasi seperti penelitian Prayogi, dkk yang menyatakan bahwa ExpressJS mampu mencapai 100% *throughput* (Prayogi et al., 2020).

5. Kesimpulan

Pendekatan *Event Driven Architecture* mampu membantu sinkronisasi data akademik ke NeoFeeder secara *real time* dengan latensi rata-rata 4,05 ms dan *response time* rata-rata di bawah 40 ms. Penulis menyadari bahwa dalam penelitian ini belum membahas terkait konsistensi data antara data akademik dengan data di NeoFeeder, maka saran untuk penelitian berikutnya adalah melakuan pengembangan sinkronisasi yang mampu menjaga konsistensi antara data akademik di database perguruan tinggi dengan data di NeoFeeder.

Ucapan Terima Kasih

Penulis sampaikan apresiasi dan terima kasih kepada Universitas Duta Bangsa Surakarta yang telah memberikan fasilitas berupa penyediaan *server*, jaringan internet dan penyediaan data untuk pengujian.

Daftar Pustaka

Amazon Web Services, I. (n.d.). *What is EDA (Event-Driven Architecture)?* Retrieved March 15, 2025, from https://aws.amazon.com/what-is/eda/

Apache, K. (n.d.). *What is event streaming*. Retrieved March 15, 2025, from https://kafka.apache.org/intro

Auliana, S., & Nurasiah, I. (2018). Penerapan Knowledge Management pada Proses Pelaporan Data Perguruan Tinggi (Studi Kasus di STIE Bina Bangsa). *Indonesian Journal of Strategic Management*, *1*(1). https://doi.org/10.25134/ijsm.v1i1.838

Brawijaya -, H., & Widodo -, S. (2023). Implementation of PDDIKTI Neo Feeder Web Service in Recording of Independent Campus Activities. *Jurnal Riset Informatika*, *5*(2). https://doi.org/10.34288/jri.v5i2.210

Bucur, V., Stan, O., & Miclea, L. (2020). An Analysis of the Implementation of Kafka in High-Frequency Electronic Trading Environments. *International Journal of Modeling and Optimization*, 52–56. https://doi.org/10.7763/IJMO.2020.V10.746

Grudniak, M., & Dzieńkowski, M. (2021). REST API performance comparison of web applications based on JavaScript programming frameworks. *Journal of Computer Sciences Institute*, *19*, 121–125. https://doi.org/10.35784/jcsi.2620

Hesse, G., Matthies, C., & Uflacker, M. (2020). How Fast Can We Insert? An Empirical Performance Evaluation of Apache Kafka. *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, 641–648. https://doi.org/10.1109/ICPADS51040.2020.00089

Köstler, J., Reiser, H. P., Habiger, G., & Hauck, F. J. (2021). SmartStream: towards byzantine resilient data streaming. *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 213–222. https://doi.org/10.1145/3412841.3441904

Lazzari, L., & Farias, K. (2023). *Uncovering the Hidden Potential of Event-Driven Architecture: A Research Agenda*.

Ornelas, T. (n.d.). *Kafkajs, a Modern Apache Kafka Client for Node.Js*. Retrieved March 28, 2025, from https://kafka.js.org

Prayogi, A. A., Niswar, M., Indrabayu, & Rijal, M. (2020). Design and Implementation of REST API for Academic Information System. *IOP Conference Series: Materials Science and Engineering*, *875*(1), 012047. https://doi.org/10.1088/1757-899X/875/1/012047

Jurnal Sistem Informasi Bisnis 01(2024)

On-line: http://ejournal.undip.ac.id/index.php/jsinbis/article/view/id_paper

Rafiezah Rizcha, Y., & Yaakub, S. (2023). Sistem Informasi Manajemen Sumber Daya Manusia Pada Universitas Muhammadiyah Jambi. *Jurnal Manajemen Sistem Informasi*, *8*(1), 78–93. https://doi.org/10.33998/jurnalmsi.2023.8.1.765

Solace Corporation. (n.d.). *Results from the Industry's First Event-Driven Architecture Survey*. Retrieved March 15, 2025, from https://solace.com/event-driven-architecture-statistics

Sopingi, S., & Wulandari, S. (2023). *Integrasi Sistem Pembelajaran dengan Google Classroom melalui Google Apps Script*. *6*(2), 195–206. https://doi.org/https://doi.org/10.31764/justek.v6i2.15061

Sopingi, Setyowati, R., & Purnomo, S. (2020). Pengembangan Web Service Digital Assessment Test of English for International Communication (TOEIC). *Jurnal E-Komtek (Elektro-Komputer-Teknik)*, *4*(1), 75–90. https://doi.org/10.37339/e-komtek.v4i1.232

Ubur, S. D. (2023). *Reviewing the Scope and Impact of Implementing a Modernised IT Event-Driven Architecture from Traditional Architecture using Agile Frameworks: A Case study of Bimodal operational strategy*.

Živanović, S., Popović, M., Vorkapić, N., Pjević, M., & Slavković, N. (2020). An overview of rapid prototyping technologies using subtractive, additive and formative processes. *FME Transactions*, *48*(2), 246–253. https://doi.org/10.5937/fmet2001246Z