Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id

E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May

OPEN ACCESS

https://doi.org/10.35870/jimik.v5i2.688

Implementasi Google *Cloud Pub/Sub* menggunakan Metode *Subscription Pull* dalam Pengiriman Data Promosi Toko di PT XYZ

Daniel Richardo [1], Tatit Kurniasih [2*]

[1,2*] Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Kota Salatiga, Provinsi Jawa Tengah, Indonesia.

Email: danielrichardo103@gmail.com [1], tatit.kurniasih@uksw.edu [2*]

Histori Artikel:
Dikirim 21 Februari 2024; *Diterima dalam bentuk revisi* 21 Maret 2024; *Diterima* 30 Maret 2024; *Diterbitkan* 10 Mei 2024.

Abstrak

Seiring dengan berkembangnya perusahaan yang menggunakan model bisnis terdistribusi, muncul masalah skalabilitas dalam proses penyebaran data dari server ke klien. PT XYZ, merupakan salah satu perusahaan yang menghadapi tantangan ini, sering mengalami keterlambatan dan request blocking ketika menggunakan model komunikasi sinkronus seperti RESTful API. Selain itu, terjadi pemborosan sumber daya dan waktu di sisi server karena harus melayani permintaan toko satu per satu, yang berdampak pada kerugian bagi toko karena tidak mendapatkan data yang diperlukan segera untuk menjalankan proses bisnis. Penelitian ini mengajukan sebuah solusi untuk mengatasi masalah ini dengan menerapkan model komunikasi publish/subscribe. Melalui implementasi arsitektur ini, klien dan server dipisahkan oleh broker, yaitu Google Cloud. Teknologi pub/sub diimplementasikan dengan menggunakan bahasa pemrograman Python dengan metode pengembangan Waterfall. Hasil dari penelitian ini adalah rancangan aplikasi dari sisi klien yang mampu menerima data dari broker yang telah dideploy sebelumnya. Hasil uji coba menunjukkan bahwa klien dapat menerima data kapan pun dengan lebih cepat tanpa mengganggu proses pengambilan data oleh klien lain.

Kata Kunci: Sistem Terdistribusi; Google Cloud Pub/Sub; Microservices; Python.

Abstract

As a company adopts a distributed business model, scalability issues often arise in the distribution of data from server to client. PT XYZ, as one of the companies facing this challenge, frequently encounters delays and request blocking when using synchronous communication models such as RESTful API. Additionally, there is resource and time wastage on the server side as it has to handle store requests one by one, resulting in losses for the store due to delayed access to required data for business processes. This research proposes a solution to address these issues by implementing a publish/subscribe communication model. Through the implementation of this architecture, clients and servers are separated by a broker, namely Google Cloud. Pub/sub technology is implemented using the Python programming language with the Waterfall development method. The results of this research include the design of a client-side application capable of receiving data from a previously deployed broker. Test results demonstrate that the client can receive data at any time without disrupting the data fetching process for other clients.

Keyword: Distributed System; Google Cloud Pub/Sub; Microservices; Python.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.688

1. Pendahuluan

Perkembangan informasi teknologi yang pesat mendorong perusahaan-perusahaan untuk terus mengembangkan teknologi untuk memenuhi kebutuhan bisnis. Pengiriman data menjadi aspek yang krusial bagi PT XYZ, sehingga migrasi ke teknologi *cloud computing* menjadi salah satu opsi yang menjajikan. Cloud computing adalah model yang memungkinkan akses jaringan yang mudah dan dapat diakses di mana saja, dan *on-demand* terhadap kumpulan sumber daya komputasi yang dapat dikonfigurasi, termasuk jaringan, server, penyimpanan, aplikasi, dan layanan (Mell & Grance, 2011). Sumber daya yang disediakan oleh layanan *cloud* dapat dengan cepat tersedia dan dirilis dengan keterlibatan sumber daya manusia yang minim. Teknologi *cloud* dirancang untuk menyesuaikan akses sumber daya yang cepat untuk memenuhi kebutuhan bisnis (Khriji *et al*., 2022), memungkinkan perusahaan untuk dapat dengan mudah mengimplementasikan suatu aplikasi dan langsung dijalankan dengan cepat. Dengan semakin besarnya bisnis perusahaan, menimbulkan beberapa masalah skalabiltas bagi perusahaan seperti PT XYZ. Model pengiriman data dengan model *client/server* dan replikasi data digunakan untuk melakukan *copy* dan pendistribusian data dan objek database untuk melaksanakan sinkronisasi sehingga konsistensi data dapat terjamin (Maulana, n.d.) . Namun Metode ini menimbulkan masalah ketika *client* mengakses *resource* yang sama secara bersamaan. Situasi ini berpengaruh pada *workflow* yang tidak efisien karena akses *server* yang terbatas harus melayani banyak client, sehingga menghasilkan antrian untuk pengiriman data (Fauzi & Bhawiyuga, 2019).

Arsitektur microservices adalah sebuah arsitektur dimana pengembangan aplikasi dimana layanan-layanan yang beroperasi dilakukan pemisahan (*decoupling*) sesuai dengan fungsinya, sehingga masing-masing layanan dapat beroperasi secara independen (Dharma Handayani, 2020). Dengan dilakukannya pemisahan sistem terdistribusi memiliki skalabilitas, fleksibilitas dan ketersediaan layanan yang lebih baik dari sistem monolitik (Nam et al., 2022). Sedangkan *service-to-service* merujuk pada model komunikasi antar layanan yang ada didalam microservies untuk berbagi data, memanggil fungsi dan berkomunikasi. REST (*representational state transfer*) merupakan sebuah bagian besar dari model *service-to-service*. Restful API terdiri dari *endpoints* yang masing-masing merupakan implementasi fungsional dari proses bisnis yang berjalan (Ehsan *et al*., 2022). REST bukanlah sebuah protokol, melainkan seperangkat pedoman untuk mendesain API untuk mengakses dan memanipulasi sumber daya menggunakan HTTP (Golmohammadi *et al*., 2024) . Dalam implementasinya di PT XYZ RESTful API menggunakan model sinkronus yang kurang cocok diterapkan karena efek antrian dalam pengiriman data ke toko. Oleh karena itu saat ini PT XYZ perlu melakukan migrasi dari REST ke model komunikasi asinkronus, dimana perlu pihak ketiga diantara *service* dengan *client*.

Google Cloud Pub/Sub merupakan salah satu layanan *cloud computing* yang disediakan oleh platform Google. Sistem Pub/Sub pada umunya terdapat 3 pihak : (1) *publisher* yang menerbitkan pesan ke sistem (2) *Subscriber* yang menarik pesan dari topik tertentu dan (3) broker yang berperan sebagai *middleware* dan bertanggungjawab sebagai *storage* bagi *subscription*, *event matching* dan *routing*. Model arsitektur Pub/Sub yang asinkron membuat *server* tidak perlu menunggu *client* untuk merespon permintaan aliran data, kedua pihak tidak terhubung secara langsung, tetapi dijembatani oleh *middleware*(Ramperez *et al*., 2022). Teknologi pub/sub sangat cocok digunakan ketika terdapat banyak node yang perlu menerima data dari satu entitas. Dalam sistem ini, server hanya perlu mengirimkan pesan sekali ke broker, karena sifat *broadcast* dari pub/sub. Setiap client dan server tidak perlu mengetahui eksistensi satu sama lain secara langsung. Oleh karena itu, toko memiliki kemampuan untuk menarik data dari broker kapan saja tanpa mengganggu aktivitas toko lainnya. Metode ini lebih efisien dari model sinkronus dimana server harus melayani tiap *request* yang masuk. Dalam praktiknya GC Pub/Sub menggunakan metode subscription *pull* dan *push*. Metode *push* dapat mengirimkan pesan secara otomatis kepada *subscriber* secara langsung ketika pesan sudah tersedia. Metode ini menciptakan pengiriman dengan latensi rendah dan rasio pengiriman yang tinggi Namun dengan *traffic* jaringan yang relatif tinggi (Wang et al., 2011).

Selain itu penggunaan metode *push* dapat menyebabkan server menjadi *overload* pada beberapa kasus apabila terjadi antrian *message* dalam jumlah yang besar. Sedangkan metode *pull* memberikan akses kontrol baik *subscriber* untuk menentukan kapan pesan dapat diambil, dan berapa jumlah pesan yang dapat ditarik sekaligus dengan menggunakan *batching*. Metode *subscription* ini cocok digunakan untuk aplikasi yang membutuhkan pengambilan dan penjadwalan pesan atau pemrosesan data yang dilakukan secara *batch* (Gutiérrez & Vera, 2022). Dalam hal ini metode *pull* lebih cocok diaplikasikan pada proses pengiriman

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.688

data toko di PT XYZ untuk menghindari masalah-masalah diatas. Berdasarkan latar belakang yang telah dipaparkan, penelitian ini bertujuan untuk membangun sebuah sistem penerimaan dan pengolahan data menggunakan pola arsitektur *publish/subscribe* dengan arus pesan yang terkendali oleh penerima data, untuk memenuhi kebutuhan PT XYZ dalam mempercepat proses pengiriman data ke toko. Penelitian ini akan menghasilkan sebuah rancangan sistem penerimaan data menggunakan layanan *platform* Google Pub/Sub menggunakan metode *subscription pull*.

2. Metode Penelitian

SDLC (*System Developpent lLfe Cycle*) adalah metodologi umum yang digunakan untuk mengembangkan perangkat lunak atau sistem informasi. Metode Waterfall adalah contoh salah satu metode pengembangan dalam SDLC. Metode pengembangan Waterfall dibagi menjadi 5 tahap yaitu analisis kebutuhan, desain, implementasi, pengujian dan *maintenance*. Metode ini sangat mudah dipahami dan digunakkan. Setiap tahap pengembangan harus diselesaikan sebelum lanjut ke fase berikutnya sehingga tidak ada *overlapping* dalam semua fase (Senarath, 2021). Peneliti memutuskan untuk menggunakan metode ini untuk mengembangkan sistem aplikasi karena *workflow* yang jelas dan prosesnya dapat didokumentasikan dengan baik.

2.1 Analisis Kebutuhan

Dalam analisis kebutuhan, peneliti melakukan diskusi dengan *stakeholder* untuk mengetahui apa saja kebutuhan dan ekspektasi kegunaan dari sistem aplikasi yang hendak dibuat. selain itu ditentukkan juga batasan sistem aplikasi. Implementasi Pub/Sub akan dilakukkan pada sistem sinkronisasi database toko dengan server pusat. Banyaknya toko yang meminta data ke server pusat setiap harinya pada suatu waktu, membuat arsitektur pub/sub dinilai cocok untuk diimplementasikan

2.2 *Desain*

Pada tahap *desain*, perlu ditentukkan struktur sistem aplikasi secara keseluruhan. Sistem aplikasi akan menggunakan bahasa pemrograman python, Google Cloud Pub/Sub untuk *communication service,* dan database oracleSQL untuk menampung data .dari sisi python peneliti memutuskan untuk melakukan pendekatan OOP (Object-Oriented Programming). Konsep utama dari OOP adalah objek sebagai pusat desain dan implementasi perangkat lunak. Objek dalam OOP adalah instansiasi dari suatu *class* yang merupakan cetak biru untuk objek tersebut. Pendekatan ini dilakukan karena peneliti memandang tabel dari database dapat diobjektifikasi dan dibuat menjadi sebuah modul yang bisa digunakkan dengan fleksibel.

2.3 Implementasi

Implementasi dilakukkan dengan perkiraan durasi pengerjaan sekitar 2 minggu. Tahap awal dilakukkan dengan mempersiapkan database, layanan Google Cloud dan *virtual environtment* dari apliaksi python. Setelah persiapan selesai, maka aplikasi akan dikembangkan hingga aplikasi dapat berjalan sesuai yang diinginkan.

2.4 Pengujian

Pada tahap pengujian dilakukkan beberapa pengujian untuk menguji seberapa aplikasi dapat memenuhi kebutuhan yang telah di tentukkan pada tahap pertama. Pengujian dilakukkan dari sisi fungsionalitas dan performa untuk memastikan aplikasi dapat digunakkan dengan baik.

2.5 Maintenance

Aplikasi akan dilakukkan *deployment* ke tahap *production,* sehingga aplikasi dapat digunakkan sepenuhnya untuk melakukkan sinkronisasi database toko dengan *server*. *Maintenance* berkala akan dilakukkan untuk memantau kinerja sistem aplikasi yang telah dibuat. apabila terdapat bug atau

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.688

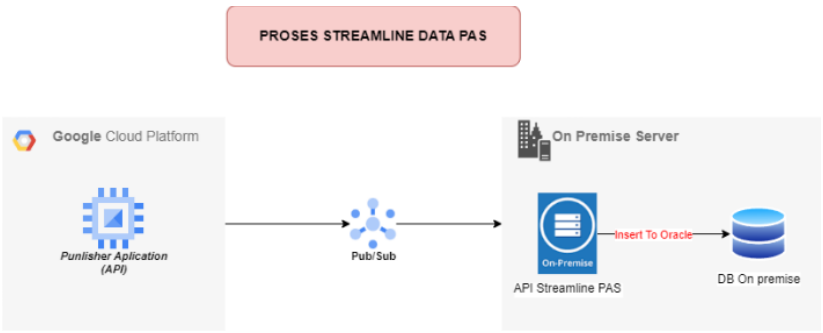
kesempatan untuk meningkatkan sistem aplikasi, maka revisi akan dilakukkan, untuk memastikkan aplikasi dapat bekerja lebih baik lagi.

3. Hasil dan Pembahasan

3.1 Perancangan Sistem

Penelitian ini menghasilkan aplikasi sistem sinkronisasi data menggunakan pola aristektur *publish/subscribe* dan bahasa pemrograman python dengan layanan Google Cloud. Dalam sinkronisasi data penulis menyiapkan database oracleSql untuk menampung data yang masuk ke sistem. Sebelumnya perlu disiapkan juga *service* Google Cloud Pub/Sub dari pembuatan *topic* dan *subscription*. *topic* adalah saluran komunikasi agar pengirim pesan dapat mengirimkan pesannya, lalu *topic* terhubung dengan *subscription* dengan metode pull agar para penerima pesan dapat menarik pesan yang diinginkan.

Proses *datastream* akan dimulai dari aplikasi pengirim data atau *publisher* dari server pusat yang melakukan *publish* message berupa JSON ke *topic* yang sudah dibuat sebelumnnya, dimana semua *message* ditampung kemudian aplikasi sinkronisasi data akan secara aktif meminta data yang ada didalam *topic* Pub/Sub melalui *subscription pull* . gambaran besar proses dapat dilihat pada gambar 1. Setelah *service* google cloud pub/sub berhasil dibuat, maka perancangan sistem dapat dilanjut ke pembuatan aplikasi Python.



Gambar 1. Proses Streamline Data Sistem

Dari sisi database, penulis menggunakan oracleSql yang digunakkan sebagai database offline tiap-tiap toko dalam PT XYZ. Database ini nantinya digunakan untuk menampung data yang berasal dari Pub/Sub untuk aplikasi Python dapat berinteraksi dengan database oracleSql, diperlukkan library cx_oracle, sehingga query dapat dijalankan di dalam script. Sebanyak 7 tabel diperlukan dalam aplikasi ini, yang terdiri dari 6 tabel *details* dan 1 tabel *master* sebagai tabel utama. Oleh karena itu dalam eksekusi query, penulis memutuskan untuk membuat *function* query builder, hal ini diperlukan untuk mensiasati banyaknya tabel yang diperlukan dalam sekali query. Selain itu script yang dibuat dapat lebih rapi dan mudah melakukan *maintain*.

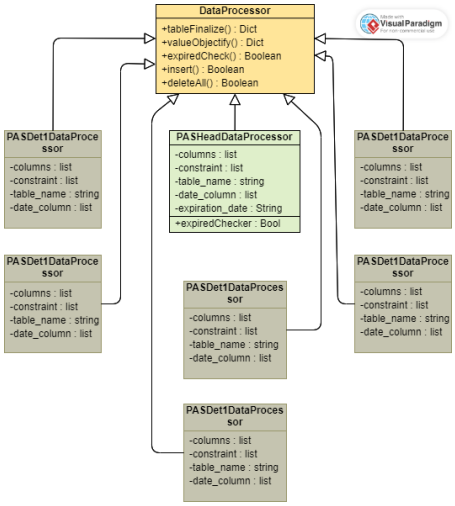
Tabel 1. Tabel Database

Tabel	Tipe Data
TPROMO_HEAD	Master
TPROMO_DET1	Detail
TPROMO_DET2	Detail
TPROMO_DET3	Detail
TPROMO_DET4	Detail
TPROMO_DET5	Detail
TPROMO_DET6	Detail

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.688

3.2 Aplikasi Python

Dalam perancangan aplikasi Python ini, penulis memilih untuk mengadopsi pendekatan *Object-Oriented Programming* (OOP). Pendekatan ini memungkinkan penulis untuk memodelkan dan mengorganisir kode secara terstruktur dengan menggunakan konsep class. Untuk meningkatkan fleksibilitas dan memastikan hubungan yang konsisten antara kelas-kelas, penulis menerapkan struktur *class* dengan hubungan *parent-child* menggunakan teknik *dependency injection*.



Gambar 2. Class Diagram

Penerapan *dependency injection* dilakukan pada tujuh tabel sebagai *children class* dari Data Processor. Keputusan ini diambil karena terdapat banyak tabel dengan atribut yang serupa di dalamnya. Oleh karena itu, ke-7 kelas ini mengandung metode-metode yang serupa untuk memfasilitasi pemrosesan data yang efisien. Tiap-tiap *children class* berisikan data tentang kolom, nama tabel, constraint dan kolom tanggal. Data-data tersebut akan digunakan untuk membangun *query command*. Berikut adalah *method* yang ada didalam *parent class* Data Processor:

Tabel 2. Tabel Method Data Processor

Method	*Return Type*	Keterangan Fungsi
tableFinalize()	dictionary	Melengkapi data kolom yang tidak ada
valueObjectify()	dictionary	Mengubah type dari data kolom menjadi object (contoh : 5.5 = Decimal(5.5))
expiredCheck()	boolean	Melakukan cek apabila suatu data kurang dari tanggal *expired*
insert()	Boolean	*Eksekusi query insert* ke database
deleteAll()	Boolean	*Eksekusi query delete* ke database

Script dibawah merupakan potongan code dari aplikasi Python yang telah selesai dibuat. *on_pull* merupakan *callback function* yang akan dieksekusi setiap kali aplikasi menerima data dari *Pub/Sub*. Pada tahap awal data ditampung kedalam variable Bernama "data_dict". Kemudian dilakukan *instantiate* terhadap class DET dan class HEAD, yang digunakkan untuk mendapatkan data-data dari kolom, *constraint* & nama tabel, dan beberapa method untuk pemrosesan data. Kemudian program untuk menentukkan bahwa suatu data sudah expired atau tidak, dijalankan. Apabila aplikasi menemukkan bahwa data tersebut sudah *expired* maka aplikasi hanya akan menjalankan *query delete* tanpa menjalankan *insert*, apabila data belum *expired* maka program akan melakukan *query delete* kemudian *insert* ke database. Hal ini merupakkan kebutuhan bisnis dari PT XYZ.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.688

Kode Program

```
def on_pull(message):
    data_dict = get_msg(message)
    message.ack()
    isExpired = False
    data = data_dict['data']
    imp_det_repo = PASDetailsImplementations()
    imp_head = PASHeadImplementation().implementations()

    for head_props in data:

        head_exp_date = head_props.get("tgl_akhir", head_props.get("TGL_AKHIR", None))
        isExpired = imp_head.expired_check(head_exp_date)
        head_param = {}

        faktur_delete = head_props['faktur']
        print(f"faktur to be deleted = {faktur_delete}")
        for details_tab in imp_det_repo.implementations:
            det_tab = imp_det_repo.get_table(details_tab)()
            det_tab.delete_all(faktur_delete, det_tab.table_name)
        imp_head.delete_all(faktur_delete, imp_head.table_name)

        if(not isExpired):
            for column in imp_head.columns:
                head_param[column.upper()] = head_props.get(column.lower(),
head_props.get(column.upper(), None))
            head_param_list = [head_param]
            imp_head.insert(head_param_list, imp_head.date_column)

            for table in imp_det_repo.implementations:
                det_param = head_props.get(table.lower(), head_props.get(table.upper()))
                if (det_param):
                    imp_det = imp_det_repo.get_table(table)()
                    imp_det.insert(det_param, imp_det.date_column)

def sub_pull():
    subscriber = pubsub.SubscriberClient()
    subscription_path = subscriber.subscription_path(
        proj_name, sub_name)
    subscriber.subscribe(subscription_path, callback=on_pull)
    print('Listening for messages on: {}'.format(subscription_path))
    while True:
        time.sleep(30)
```

3.3 Pengujian Sistem

Selanjutnya, kami melakukan pengujian sistem untuk menilai sejauh mana kesesuaian antara sistem yang telah dikembangkan dengan spesifikasi dan kebutuhan yang telah ditetapkan pada tahap awal perancangan aplikasi. Pengujian ini dilakukan melalui penerapan metode *blackbox*. Metode *blackbox* fokus pada evaluasi output yang dihasilkan oleh sistem tanpa memerinci proses internal yang terjadi. Implementasi dari teknologi Pub/Sub diharapkan dapat membantu proses sinkronisasi data menjadi lebih cepat dan efisien. Berikut adalah tabel pengujian *blackbox* yang telah dilakukkan, pengujian dilakukkan dengan mengirimkan *message* melalui service pub/sub untuk diterima ke sistem untuk diproses.

Tabel 3. Pengujian Metode *Blackbox*

No	Pengujian	Jumlah message	Hasil	Kesimpulan
1	Sistem dapat memproses data yang masuk dari *service* Pub/Sub dan memasukkan data ke database oracle dengan lengkap	1	Data dapat diproses dengan format yang sesuai dan melakukan eksekusi insert ke database dengan lengkap	Valid
2	Sistem dapat memproses	20	20 message yang dikirimkan	Valid

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.688

	banyak data sekaligus yang telah masuk ke Pub/Sub tanpa terjadi error		sekaligus masuk ke database dengan kondisi lengkap	
3	Sistem dapat mengenali *message* yang sudah *expired* kemudian menghapus data yang sebelumnya ada di database sesuai dengan *primary key*	1	Row dengan kondisi *expired* terhapus dari database ketika *message* masuk dengan tanggal *expired*	Valid
4	Sistem dapat memproses data yang memiliki data yang kosong	1	Value kolom yang kosong dapat masuk ke database dengan value *null* atau *none*	Valid
5	Sistem dapat dijalankan dan memproses data yang masuk di beberapa perangkat sekaligus tanpa mengganggu proses satu sama lain, dengan server mengirimkan sebanyak satu (1) kali	10	Saat sistem dijalankan di 2 perangkat, kedua perangkat dapat menerima dan memproses data yang masuk secara serentak saat server mengirimkan data sebanyak satu (1) kali	Valid

Dalam rangkaian pengujian yang telah dilakukan, hasil menunjukkan bahwa seluruh kriteria dan kebutuhan aplikasi berhasil memenuhi standar yang ditetapkan. Sistem sinkronisasi data ke database toko menggunakan teknologi pub/sub terbukti mampu menerima data secara lengkap dari server pusat tanpa kehilangan paket data. Selain itu, sistem mampu menangani kondisi khusus, seperti eksekusi penghapusan data promosi yang telah kedaluwarsa, sesuai dengan kebutuhan yang diinginkan. Hasil pengujian nomor 5 menunjukkan bahwa sistem dapat secara serentak menerima data melalui beberapa perangkat sekaligus, menandakan keberhasilan teknologi pub/sub dalam memfasilitasi pemisahan antara *client* dan server. Hal ini memungkinkan setiap proses pengambilan data di toko tidak mengganggu operasi toko lainnya. Pengujian juga menunjukkan bahwa server pusat hanya perlu mengirimkan pesan sekali untuk menyampaikan data kepada seluruh toko yang membutuhkan paket data.

4. Kesimpulan

Ketika melakukan proses bisnis di PT XYZ, dalam hal ini distribusi data dari *server* pusat ke banyaknya *client*, seringkali terjadi masalah seperti terciptanya antrian dan *blocking* ketika banyak *client* yang melakukan *request* ke *server*. Hal ini menyebabkan penundaan yang berujung pada pemborosan *resource* pada sisi server, dan *client* yang dirugikkan karena keterlambatan data yang dibutuhkan. Model komunikasi sinkronus seperti RESTful API menjadi kurang cocok apabila bisnis memiliki skala besar, terdistribusi dan membutuhkan kecepatan *real-time* dalam proses pengiriman data. Oleh karena itu dalam artikel ini penulis mengajukan Pub/Sub dengan metode *pull* sebagai solusi dalam permasalahan ini. Dalam arsitektur pub/sub *server* dan *client* dipisahkan oleh *broker* seperti Google Cloud, dengan dilakukkannya pemisahan ini atau *decoupling* maka *server* pusat hanya perlu mengirim data yang diperlukkan *client* satu kali saja. Dan client atau toko dapat menarik data yang dibutuhkan kapan saja tanpa menganggu proses toko yang lain. Pengembangan dalam artikel ini berhasil mengembangkan aplikasi dari sisi *client* untuk menerima data dari *service* Google Cloud Pub/Sub, sehingga toko-toko dapat diuntungkan dari sisi fleksibilitas dan kecepatan penerimaan data.

E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.688

5. Ucapan Terima Kasih

Artikel ini disusun untuk memenuhi persyaratan tugas akhir strata 1 Program Studi Teknik Informatika Universitas Kristen Satya Wacana Salatiga. Dalam penyusunan artikel ini penulis mendapat berbagai tantangan, tetapi dengan bantuan banyak orang, puji Tuhan penulis dapat menyelesaikan penulisan artikel ini dengan baik. Oleh karena itu penulis dalam kesempatan ini ingin mengucapkan rasa terimakasih sebesar-besarnya kepada:

1) Prof. Ir. Daniel H.F. Manongga, M.Sc., Ph.D. selaku Dekan Fakultas Teknologi Informasi Universitas Kristen Satya Wacana.
2) Budhi Kristianto, S.Kom., M.Sc., Ph.D. selaku Ketua Program Studi Teknik Informatika, Universitas Kristen Satya Wacana.
3) Tatit Kurniasih, S.E, M.MSc selaku dosen pembimbing yang telah menuntun dalam pembuatan artikel.
4) Ibu terkasih yang telah membantu dan mendukung penulis selama proses penulisan artikel.
5) Teman-teman, mentor dan senior yang sudah membantu dan menemani penulis dalam penulisan artikel.
6) Semua pihak yang terlibat dalam pembuatan artikel ini.

6. Daftar Pustaka

Ehsan, A., Abuhaliqa, M. A. M., Catal, C., & Mishra, D. (2022). RESTful API testing methodologies: Rationale, challenges, and solution directions. *Applied Sciences*, *12*(9), 4369.

Fauzi, M., & Bhawiyuga, A. (2019). Implementasi Arsitektur Publish Subscribe Pada Constrained Application Protocol (COAP) di Lingkungan Internet of Things (IoT). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *3*(7), 7060-7067.

Golmohammadi, A., Zhang, M., & Arcuri, A. (2023). Testing restful apis: A survey. *ACM Transactions on Software Engineering and Methodology*, *33*(1), 1-41. DOI: https://doi.org/10.1145/3617175.

GUTIÉRREZ, S. L., & VERA, Y. P. (2022). A Cloud Pub/Sub Architecture to Integrate Google Big Query with Elasticsearch using Cloud Functions.

Khriji, S., Benbelgacem, Y., Chéour, R., Houssaini, D. E., & Kanoun, O. (2022). Design and implementation of a cloud-based event-driven architecture for real-time data processing in wireless sensor networks. *The Journal of Supercomputing*, *78*(3), 3374-3401.

Maulana, H. (2016). Analisis Dan Perancangan Sistem Replikasi Database Mysql Dengan Menggunakan Vmware Pada Sistem Operasi Open Source. *InfoTekJar: Jurnal Nasional Informatika dan Teknologi Jaringan*, *1*(1), 32-37. DOI: https://doi.org/10.30743/infotekjar.v1i1.37.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.

Nam, J., Jun, Y., & Choi, M. (2022). High Performance IoT Cloud Computing Framework Using Pub/Sub Techniques. *Applied Sciences*, *12*(21), 11009. DOI: https://doi.org/10.3390/app122111009.

E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.688

Rampérez, V., Soriano, J., Lizcano, D., & Miguel, C. (2022). Automatic evaluation and comparison of pub/sub systems performance improvements. *Journal of Web Engineering*, *21*(4), 1055-1080. DOI: https://doi.org/10.13052/jwe1540-9589.2144.

Senarath, U. S. (2021). Waterfall methodology, prototyping and agile development. *Tech. Rep.*, 1-16. DOI: https://doi.org/10.13140/RG.2.2.17918.72001.

Uminingsih, U., & Handayani, S. D. (2020). Pengorganisasian Kerja Sistem Parkir Menggunakan Arsitektur Microservice. *Jurnal Teknologi*, *13*(1), 27-35.

Wahid, A. A. (2020). Analisis metode waterfall untuk pengembangan sistem informasi. *J. Ilmu-ilmu Inform. dan Manaj. STMIK, no. November*, 1-5.

Wang, J., Li, Z., Zhang, Y., & Jiang, S. (2011). Space-Time based on content-based pub/sub in Delay Tolerance Networks. *Procedia Engineering*, *15*, 2955-2960. DOI: https://doi.org/10.1016/j.proeng.2011.08.556.