Analisis dan Implementasi REST, GRPC dan Pub/Sub Sebagai Media Pertukaran Data pada Arsitektur Microservice untuk Sistem *Driver Broadcast Message* PT. Blue Bird Tbk

Agung Satria dan Prihandoko

Program Studi Magister Sistem Informasi, Program Pasca Sarjana
Universitas Gunadarma
Jl. Salemba Raya No. 53, Jakarta Pusat, DKI Jakarta
E-mail: id.agungsatria@gmail.com, prihandoko@gmail.com

Abstrak

Berdasarkan pengalaman penulis, kendala yang harus dihadapi pada suatu arsitektur *microservice* salah satunya yaitu proses pengiriman data antar service, kesalahan dalam implementasi metode pengiriman data yang kurang tepat akan mengakibatkan sebuah proses pada *microservice* menjadi lambat, atau bahkan bottleneck pada suatu service. Beberapa metode yang digunakan untuk pengiriman data antar service yang dapat dilakukan yaitu dengan mengimplementasikan REST, gRPC dan Pub/Sub. Pada saat yang bersamaan, PT. Blue Bird Group Tbk. sedang membangun sistem berbasis teknologi untuk mempermudah pelayanan taksi, baik untuk penumpang, mitra pengemudi, dan operator yang berjalan di arsitektur microservice dari yang sebelumnya menggunakan arsitektur monolithic. Pelayanan operator yang salah satunya untuk menginformasikan mitra pengemudi masih menggunakan cara manual dengan menghubunginya langsung. Tujuan penelitian ini untuk meningkatkan pelayanan operator menginformasikan ke satu atau banyak *(broadcast message)* mitra pengemudi dengan membangun sistem driver broadcast message di atas arsitektur *microservice* menggunakan metode Scrum, serta melakukan analisis dan implementasi REST, gRPC dan Pub/Sub. Berdasarkan hasil penelitian, sistem driver broadcast membantu operator menginformasikan ke satu atau banyak mitra pengemudi, serta membantu mitra pengemudi mendapatkan informasi yang cepat lagi tepat, selain itu didapat hasil analisis jika penggunaan REST sangat cocok untuk interaksi luar seperti publikasi API, penggunaan gRPC lebih cocok untuk interaksi antar *private services* dengan proses data yang lebih ringan dan penggunaan Pub/Sub sangat cocok untuk interaksi antar service yang membutuhkan proses *asynchronous* untuk pemrosesan data yang lebih cepat.

Kata kunci :: *Broadcast Message, gRPC, Microservice, Pub/Sub, REST*

Pendahuluan

Arsitektur *monolithic* merupakan arsitektur yang menggambarkan servis yang mengeksekusi seluruh logika pada suatu codebase dalam satu server [1]. *Codebase* dapat terus bertumbuh dan menjadi besar ketika penambahan fitur terus terjadi secara terus-menerus, *Codebase* yang besar dapat terjadi ketika pengembangan perangkat lunak dilakukan dengan menggunakan arsitektur *monolithic* [2].

Pertumbuhan *codebase* yang terjadi secara terus-menerus, dapat membuat pengembang menghadapi kesulitan untuk mengetahui dimanakah letak perubahan akan dilakukan ketika *codebase* sudah bertumbuh besar [3]. Peningkatan pada penggunaan service berbasiskan arsitektur *monolithic* akan sangat mempengaruhi pemeliharaan sebuah server, kompleksnya pembaruan aplikasi dan kinerja service itu sendiri [1].

Jika dibandingkan dengan arsitektur *monolithic*, arsitektur *microservice* merupakan alternatif arsitektur yang lebih terukur dan lebih fleksible [4] untuk sistem besar dan kompleks yang sebelumnya menggunakan arsitektur monolithic [5]. Pada arsitektur *microservice*, sistem informasi dirancang untuk terdistribusi dan menyediakan layanan secara lebih fokus dan spesifik. Sistem yang besar akan dipecah menjadi beberapa sistem-sistem kecil yang disusun dalam satu service, dimana setiap service memiliki tanggung jawabnya sendiri [6]

DOI : http://dx.doi.org/10.32409/jikstik.23.4.3657

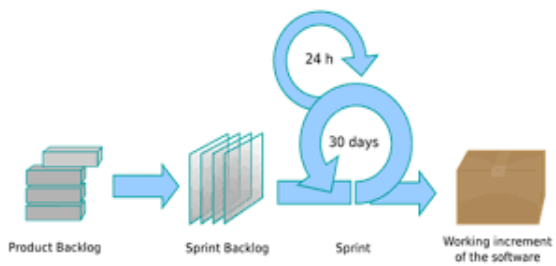
PT. Blue Bird Tbk, tempat penulis bekerja, merupakan perusahaan transportasi asal Indonesia, yang fokusnya untuk menjual jasa kepada masyarakat lewat transportasi. Dengan fokus bisnis tersebut maka PT. Blue Bird Tbk. mempunyai peranan yang cukup penting dalam bidang transportasi di Indonesia dengan menyediakan jasa taksi [7].

Saat ini, PT. Blue Bird Tbk. sedang membangun sistem berbasis teknologi untuk mempermudah pelayanan taksi baik untuk penumpang, mitra pengemudi, dan operator yang berbasiskan arsitektur *microservice.*

Pelayanan operator yang salah satunya untuk meng-informasikan mitra pengemudi masih menggunakan cara manual dengan menghubunginya langsung. Untuk semakin mempermudah pelayanan operator, penulis merencanakan untuk membangun sistem *driver broadcast message*, yang berfungsi untuk operator menginformasikan ke satu atau banyak mitra pengemudi, dan dengan menggunakan layanan Qiscus SDK untuk mempermudah pemeliharaan sistemnya. Qiscus adalah perusahaan teknologi yang menyediakan layanan pesan instan dan komunikasi untuk tempat kerja. Perusahaan ini didirikan di Singapura pada tahun 2013 dan memiliki pusat riset dan pengembangan teknologi di Yogyakarta, Indonesia.

Namun, kendala yang harus dihadapi pada suatu arsitektur *microservice* salah satunya yaitu proses pengiriman data antar *service*, berdasarkan pengalaman penulis, kesalahan dalam implementasi metode pengiriman data yang kurang tepat akan mengakibatkan sebuah proses pada *microservice* menjadi lambat, atau bahkan bottleneck pada suatu service, beberapa metode yang dilakukan untuk pengiriman data antar *microservice* yang akan penulis lakukan analisis yaitu dengan menggunakan REST, gRPC dan Pub/Sub.

Adapun tujuan dilakukan penelitian ini selain untuk meningkatkan pelayanan operator menginformasikan ke satu atau banyak mitra pengemudi dengan membangun sistem d*river broadcast message* di atas arsitektur *microservice* mengunakan metode *scrum*, serta juga melakukan analisis dan implementasi REST, gRPC dan Pub/Sub agar meminimalisir kesalahan dalam pemilihan metode protokol pada proses pertukaran data.



Gambar 1: Metodologi *Scrum Development*

Metode Penelitian

Agar penelitian yang dilakukan dan dapat terukur maka metode pengembangan sistem dapat mengakomodasi penliti dalam melakukan penelitian bertahap. Metode yang gunakan untuk pengembangan sistem *driver broadcast message* ini menggunakan metodologi Scrum, lihat Gambar 1. Scrum adalah salah satu metode pengembangan turunan dari *Agile Development* [8], yaitu:

1. *Product backlog*

   Product backlog merupakan proses pengumpulan kebutuhan yang lakukan melalui pembuatan daftar kebutuhan, yaitu dengan melakukan proses analisis masalah untuk mengidentifikasi *root cause*, beberapa langkah untuk menganalisis *root cause* dengan mencari bukti masalah, yaitu dengan memahami bagaimana masalah tersebut terjadi, membuat pernyataan masalah, menjelaskan penyebab terjadinya masalah yang merupakan penyimpangan dari tujuan yang ingin dicapai, menganalisis dampak masalah, melihat dan memprediksi dampak negatifnya jika masalah ini terus terjadi, mencari penyebab masalah terjadi, mencari apa yang menjadi penyebab masalahnya, dengan menggunakan *Root cause analyze* (RCA) 5-whys [9].

2. *Sprint backlog*

   Sprint backlog adalah proses pemenuhan kebutuhan untuk merencanakan suatu pekerjaan yang nantinya akan dilakukan dalam sebuah sprint [10] sesuai dengan yang diingkan pada proses *product backlog* yang telah ditentukan, bisa berupa hasil desain arsitektur, database dan lain sebagainya.

3. *Sprint*

   Sprint merupakan proses yang dilakukannya pemaparan hasil produk dalam bentuk *prototype* (termasuk pengembangan code dan service, maupun *update flow* berdasarkan hasil diskusi melalui *dialy scrum*) kepada pemilik produk.

4. *Working Increment of the software*

   Dalam proses pengembangan pada fase ini dilakukan penyesuaian kebutuhan dengan cara melakukan pertemuan untuk memberikan penyajian kepada pemilik produk. Setelah masukkan didapat, maka selanjutnya dilakukan perbaikan dan kemudian dilakukan penyajian kembali kepada pihak terkait.

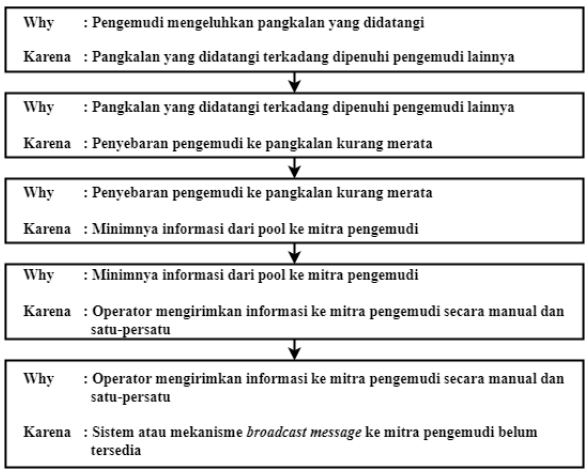
Hasil dan Pembahasan

Berikut merupakan tahapan yang telah dilakukan dalam analisis dan implementasi REST, gRPC, dan

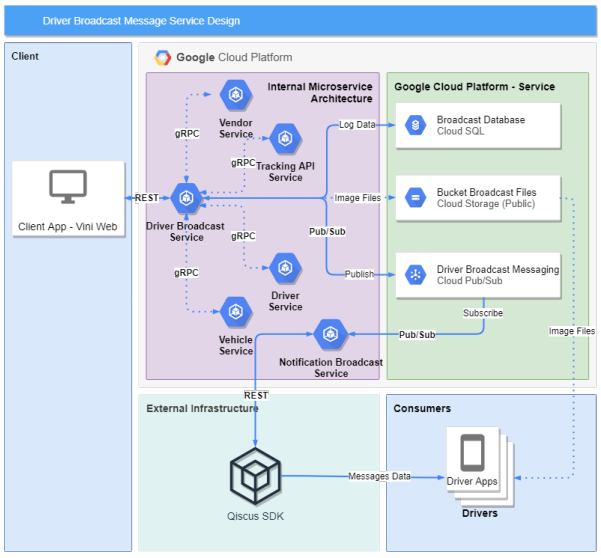
Pub/Sub sebagai media pertukaran data pada arsitektur microservice untuk sistem driver broadcast message PT. Blue Bird Tbk.

Hasil *Product Backlog*

Hasil dari *product backlog* didapat dari penggunaan Gambar 2 *Root Cause Anlyze (RCA) 5-whys,* yaitu didasari oleh pengemudi yang terkadang mengeluhkan pangkalan yang didatangi dikarenakan informasi yang minim, dan didapat *root cause* nya, yaitu belum tersedianya sistem *broadcast message* untuk mengirimkan informasi ke mitra pengemudi.



Gambar 2: Contoh pola *rail fence cipher*



Gambar 3: Hasil *Root Cause Analyze (RCA) 5-Whys*

Hasil *Sprint Backlog*

Pada Gambar 3 yang merupakan arsitektur dari sistem *driver broadcast message* yang dibuat dari diagram *Google Cloud Platform*, dibagi menjadi beberapa zona, yaitu zona *client* untuk *web service*, zona *Google Cloud Platform* yang merupakan kumpulan layanan *Google Cloud Platform*, zona external infrastructure yang merupakan pihak ketiga, zona consumers yaitu *end- user* yang mengkonsumsi informasi dari *client web service.*

Dari desain database sistem *driver broadcast message* pada rincian desain Gambar 4 didapat beberapa tabel, berikut rincian tabel:

1. Tabel *mst_ message_ type*

   Tabel *mst_ message_ type* atau master *message type*, menyimpan master data untuk tipe pesan yang dituju dan memiliki satu *primary key*, nantinya menjadi *foreign key* di tabel ¬*trx_ broadcast* dan wajib memilih salah satu dari tipe pesan yang dipilih, yang bertipekan *one mandatory to many mandatory.*

2. Tabel *trx_ broadcast*

   Tabel t*rx_ broadcast* merupakan tabel transaksi dari data *broadcast message* yang akan dikirim, tabel ini memiliki *primary key*, yang akan dikonsumsi beberapa tabel lainnya.

3. Tabel *trx_ broadcast_ vehtype*

   Tabel *trx_ broadcast_ vehtype* atau *transaction broadcast vehicle type* merupakan bagian dari tabel tranksaksi *broadcast message* yang akan dikirim, memiliki dua *field key* sekaligus, yaitu key pada *trx_ broadcast*, dan key dari *vehicle type* yang didapat hasil dari komunikasi dengan *vehicle service.* Relasi antara *trx_ broadcast* dengan *trx_ broadcast_ vehtype* yaitu bertipekan *one mandatory to many mandatory.*

4. Tabel *trx_ broadcast_ depot*

   Tabel *trx_ broadcast_ depot* atau transaction broadcast depot merupakan bagian dari tabel tranksaksi *broadcast message* yang akan dikirim, memiliki tiga *field key* sekaligus, yaitu key pada *trx_ broadcast,* dan 2 (dua) key dari depot id yang didapat hasil dari komunikasi dengan depot *service.* Relasi antara *trx_ broadcast* dengan *trx_ broadcast_ depot* yaitu bertipekan *one mandatory to many mandatory.*

5. Tabel *trx_ broadcast_ vehstatus*

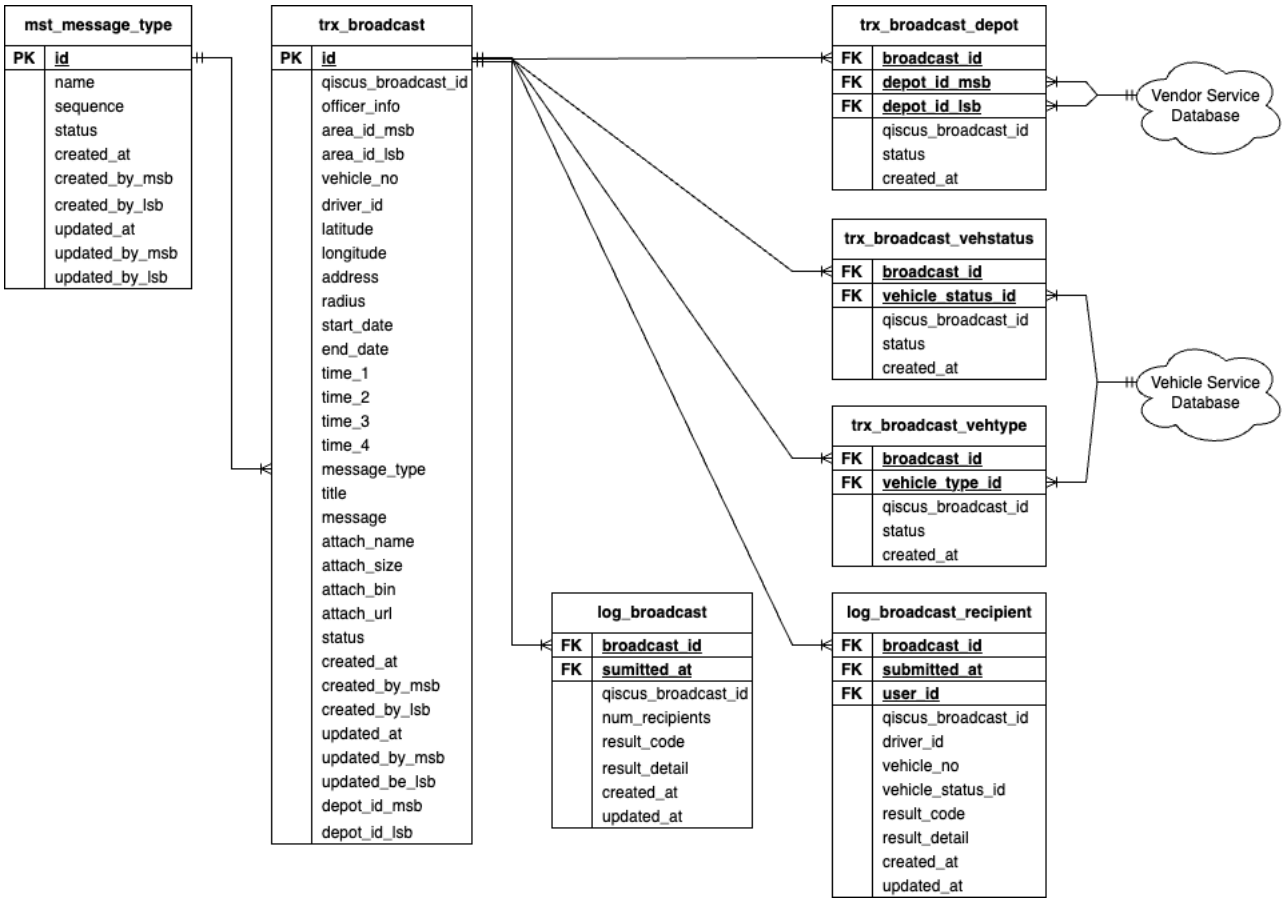
   Tabel *trx_ broadcast_ vehstatus* atau *transaction broadcast vehicle* status merupakan bagian dari tabel tranksaksi *broadcast message* yang akan dikirim, memiliki dua field key sekaligus, yaitu key pada *trx_ broadcast*, dan 2 (dua) key dari *vehicle* status yang didapat hasil dari komunikasi dengan *vehicle service.* Relasi antara *trx_ broadcast* dengan *trx_ broadcast_ vehstatus* yaitu bertipekan *one mandatory to many mandatory.*

6. Tabel *log_ broadcast*

   Tabel *log_ broadcast* merupakan bagian dari tabel pencatatan riwayat tranksaksi *broadcast message* yang diproses, memiliki dua field key sekaligus, yaitu key pada *trx_ broadcast*, dan key yang dihasilkan otomatis dari waktu sistem. Relasi antara *trx_ broadcast* dengan *log_ broadcast* yaitu bertipekan *one mandatory to many mandatory.*

7. Tabel *log_ broadcast_ recipients*

   Tabel *log_ broadcast_ recipients* merupakan bagian dari tabel pencatatan riwayat tranksaksi *broadcast message* yang diproses, memiliki tiga *field key* sekaligus, yaitu key pada t*rx_ broadcast, key* yang dihasilkan otomatis dari waktu sistem, dan key dari pengguna yang melakukan transaksi. Relasi antara *trx_ broadcast* dengan l*og_ broadcast_ recipients* yaitu *one mandatory to many mandatory.*



Gambar 4: Desain Database Sistem *Driver Broadcast Message*

Hasil *Sprint*

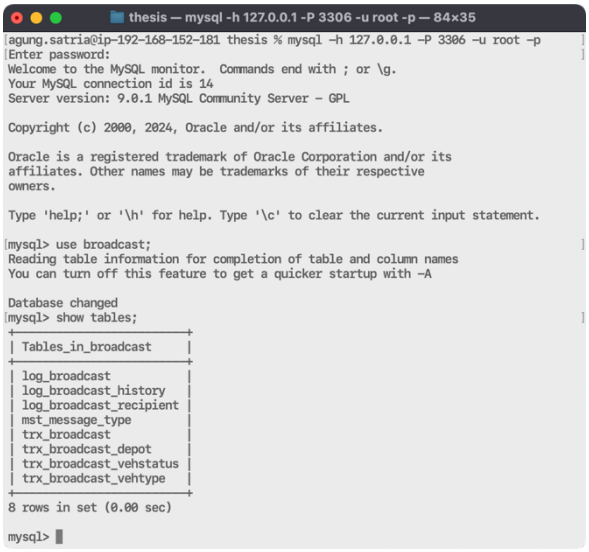
Hasil dari *sprint* akan menghasilkan pengembangan sistem *driver broadcast message* yang berjalan di atas arsitektur *microservice*, yang akan dibagi menjadi beberapa bagian lagi, sebagai berikut:

1. Persiapan Perangkat Pada pengembangan sistem *driver broadcast message* akan dibangun dan dilakukan tes pada perangkat di Tabel 1.

2. Database dan Tabel Dalam pembuatan database dan tabel ini menggunakan MySQL yang mengacu pada Gambar 4 desain database sistem *driver broadcast message.* Setelah itu untuk mengecek database dan tabel telah terbuat dengan sebagaimana mestinya, yaitu dengan melakukan koneksi ke database dengan MySQL *Command Line Interface CLI* yang hasilnya pada Gambar 5.

Tabel 1: Perangkat Pengujian Sistem

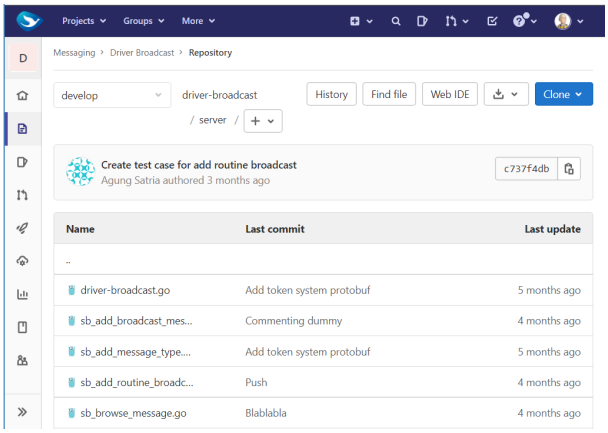
	Laptop	**Telepon Pintar**
Merk	Laptop HP Pavilion	Samsung Galaxy S10 Plus
Memory	8 GB	8 GB
Prosesor	Intel® Core™ i5-1035G1 CPU - 8 Cores	Exynos 9820 @2.73 Ghz - 8 Cores
Grafik	Graphics NV138 / Mesa Intel® UHD Graphics (ICL GT1)	GPU ARM Mali-G76
Penyimpanan	SSD 512 GB M2 NVMe	Internal Storage 128 GB
Sistem Operasi	Ubuntu 20.04.1 LTS x64	Android Version 11 (Red Velvet Cake) x64



Gambar 5: Hasil Pembuatan Database dan Tabel MySQL CLI

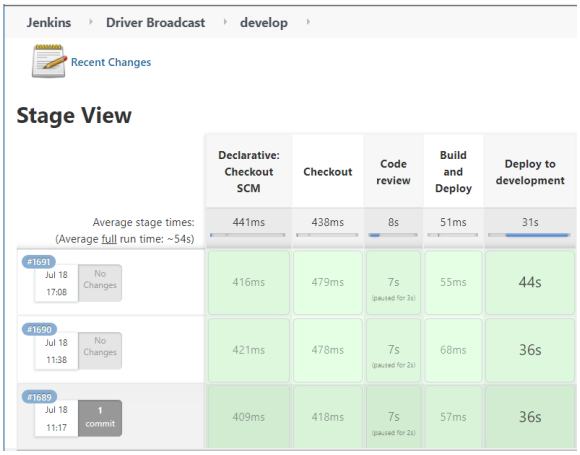
3. Hasil Penulisan Kode Setelah kode ditulis, kode tersebut akan di-push dan disimpan di repository private milik PT. Blue Bird Group Tbk. dengan menggunakan GitLab yang tampilannya pada Gambar 6.

4. Hasil Build Kode Setelah melewati proses development dan penulisan kode, maka tahap berikutnya adalah melakukan deployment aplikasi ke server yang dituju dengan melakukan build kode menggunakan Jenkins pada Gambar 7.

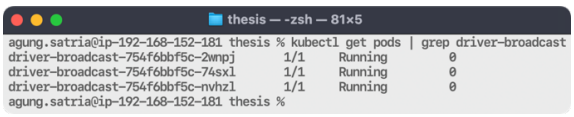


Gambar 6: Hasil Penulisan Kode yang Disimpan pada GitLab

Setelah itu, pada *Google Cloud Platform* menghasilkan replika pada pods seperti Gambar 8.



Gambar 7: Hasil Build Kode Menggunakan Jenkins



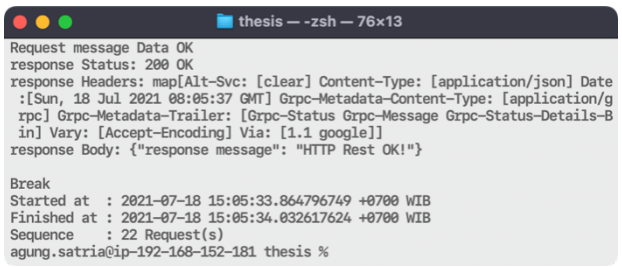
Gambar 8: Replika Pada Pods yang Berjalan

Hasil *Working Increment of The Software*

Pada *Working Increment of the software* yaitu melakukan *review* dan *flow acceptance*. Pada tahap ini, dilakukan *review* oleh pemilik produk (*product owner*) untuk hasil *prototype*, setelah itu melakukan *flow acceptance* pada alur kerja sistem *driver broadcast message*. Setelah itu, dilakukan review pengujuan sistem berupa benchmark test untuk mendapatkan *Request Per Second* (RPS) untuk ketiga metode pengiriman data, yaitu menggunakan REST, gRPC dan Pub/Sub. Setelah itu melakukan review dan *flow acceptance*. Berikut hasil *review benchmark* dan flow *acceptance test:*

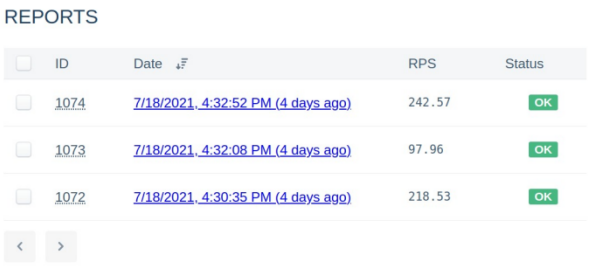
1. Benchmark Test Pada hasil *benchmark test REST*, gRPC dan Pub/Sub sebagai media pertukaran data pada arsitektur microservice untuk sistem *driver broadcast message* PT. Blue Bird Tbk. didapat hasil sebagai berikut:

   (a) Hasil *Benchmark Test REST* Pada Gambar 9, didapat hasil dari benchmark test menggunakan media pertukaran data REST, yaitu service dapat memproses request sebanyak 22 *Requests Per Second* (RPS).



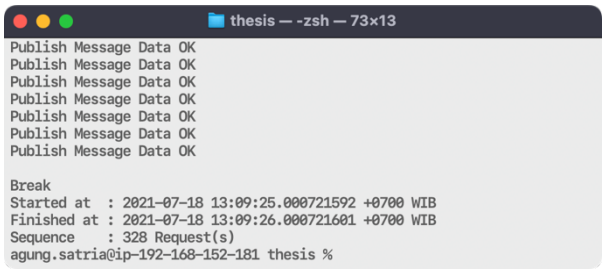
Gambar 9: REST Sanggup Memproses 22 RPS

(b) Hasil *Benchmark Test* gRPC Pada Gambar 10, didapat hasil dari benchmark test menggunakan gRPC, yaitu *service* dapat memproses request hingga rata-rata 186,35 RPS.

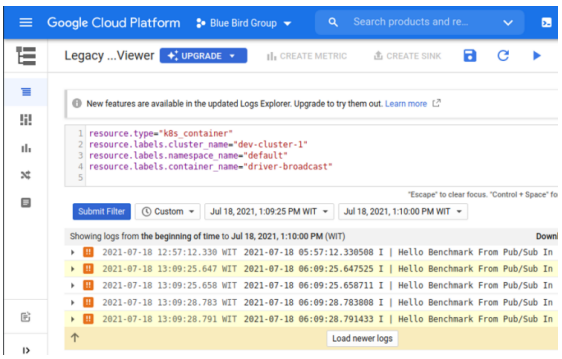


Gambar 10: gRPC Sanggup Memproses 186,35 RPS

(c) Hasil *Benchmark Test Pub/Sub* Pada Gambar 11, didapat hasil dari *benchmark test* menggunakan Pub/Sub, yaitu *service* dapat melakukan *publish* hingga 328 RPS. Namun pada proses *subscribe*, sistem menerima semua pesan tersebut dan pada Gambar 12, pesan pertama datang pada pukul 13:09:25.647 lalu pesan terakhir pada pukul 13:09:28.791, proses tersebut memakan waktu sekitar 3 (tiga) detik. Jadi total *publish message* yang sebanyak 328 proses, memakan waktu hingga 3 (tiga) detik untuk di-*subscribe client*, artinya jika dimasukkan ke dalam satuan RPS, hasilnya yaitu 328/3 RPS atau 109.33 RPS

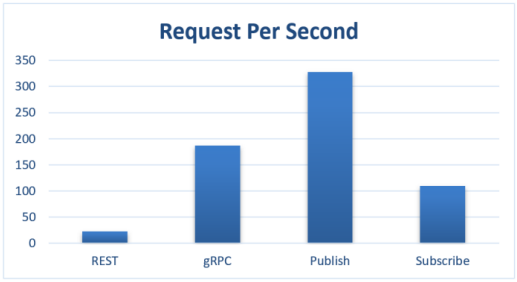


Gambar 11: Pub/Sub Sanggup Publish Hingga 328 RPS



Gambar 12: Log Waktu dari Pesan Pertama dan Terakhir Masuk ke Service

Setelah itu didapat hasil kesimpulan dari benchmark test yang disajikan berbentuk chart pada Gambar 13.

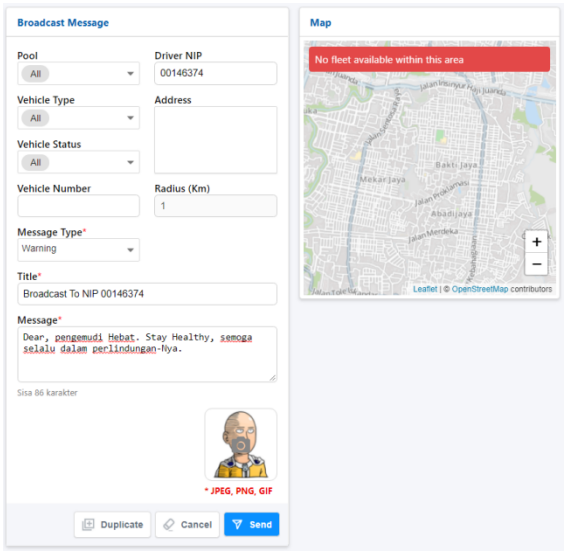


Gambar 13: Chart Perbandingan Metode Dengiriman Data

2. *Flow Acceptance Test* Hasil *flow acceptance test* yaitu hasil dari *User Acceptance Test (UAT)* pada sistem menjadi tolak ukur untuk memastikan kebutuhan user atau operator pengguna web *client* dalam mengirimkan informasi ke satu atau banyak mitra pengemudinya tidak mengalami kendala dalam menyebarkan informasi dengan cara *broadcast message*. Untuk flow utama pengetesan sebagai berikut:

   (a) Membuat *Broadcast Message* Langkah pertama yaitu membuat broadcast message ke NIP pengemudi digunakan untuk pengetesan seperti pada Gambar 14 yang dibuat melalui web *client.*

   (b) Mengecek Pesan di *Driver Apps* Selanjutnya melakukan pemeriksaan pada smartphone yang digunakan untuk pengetesan, yaitu dengan memeriksa notifikasi yang muncul. Pada Gambar 15 dan 16, notifikasi dari *Driver Apps* tentang pesan broadcast yang masuk telah muncul. Lalu setelah membuka notifikasi, maka akan masuk ke aplikasi *Driver Apps* dan memeriksa menu "Pesan Masuk", maka munculah rincian dari pesan yang telah dibuat melalui web client sebelunya.

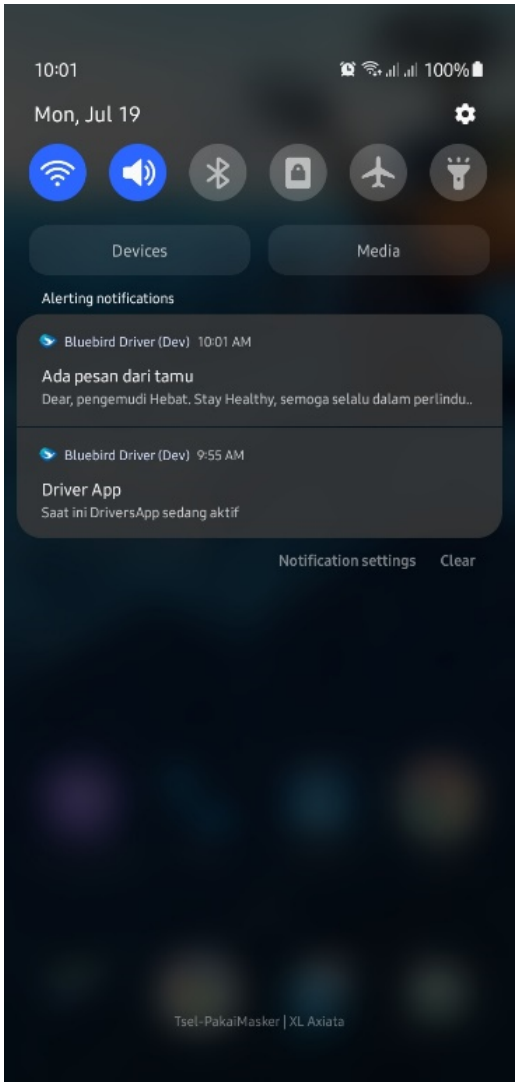


Gambar 14: Pembuatan *Broadcast Message* Melalui Web Client

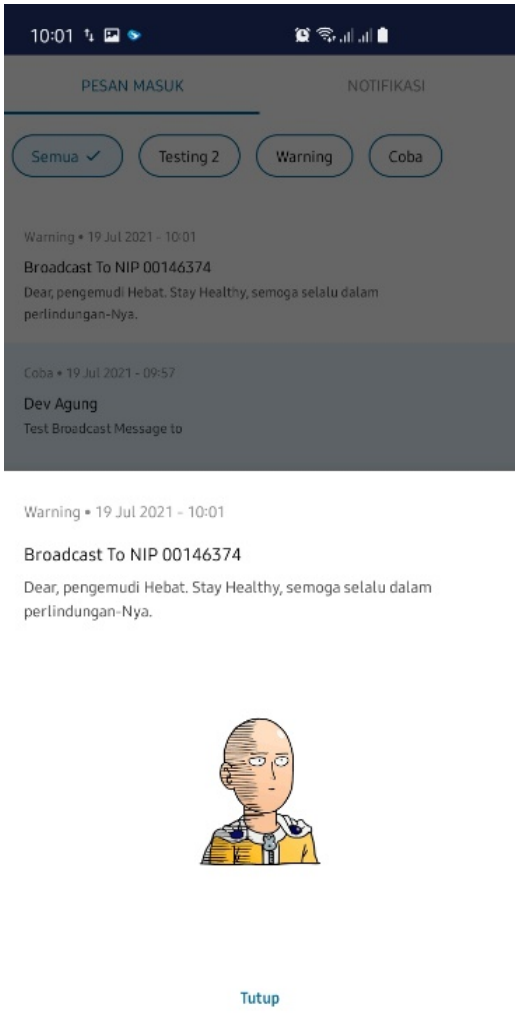
(c) Daftar Riwayat Pengiriman *Broadcast Message* Setelah broadcast message berhasil masuk ke *Driver Apps* pada smartphone, selanjutnya yaitu pada sisi web client bisa melihat daftar riwayat pengiriman *broadcast message* seperti Gambar 17.

(d) Daftar Penerima *Broadcast Message* Pengguna web client juga dapat melihat daftar penerima dan status dari *broadcast message* yang telah dikirimkan sebelumnya pada Gambar 18.

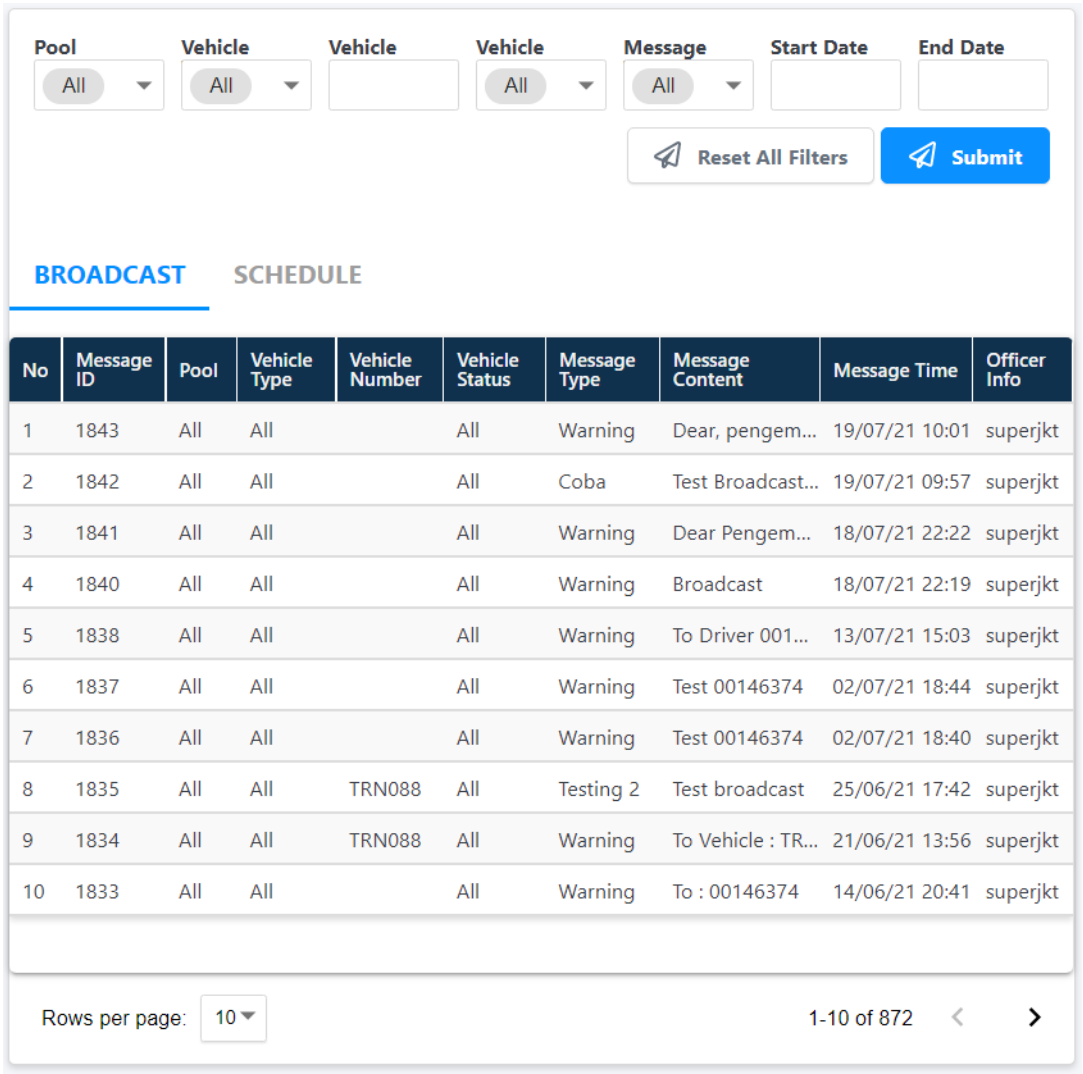
Selanjutnya kriteria yang digunakan untuk pengujian sistem driver *broadcast message* yang berjalan di atas arsitektur *microservice* ini menggunakan metode validasi untuk membuktikan validitas sistem yang akan digunakan oleh operator pool dalam mengelola sistem tersebut. Pada Tabel 2 merupakan hasil dari pengujian sistem.



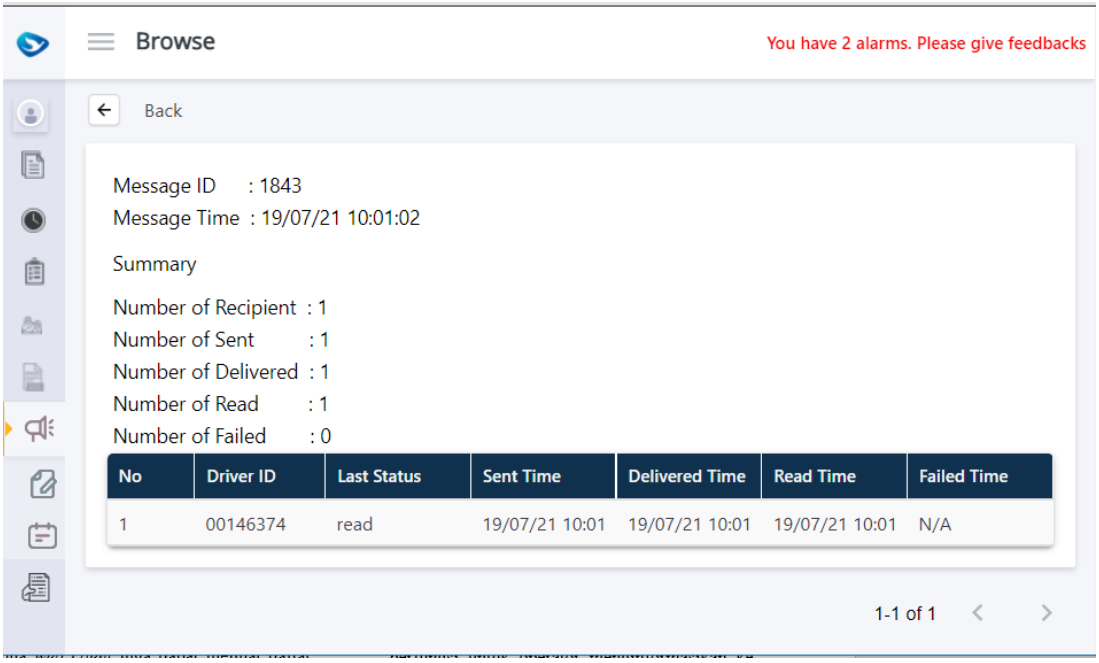
Gambar 15: Notifikasi dan Rincian Pesan *Broadcast*



Gambar 16: Notifikasi dan Rincian Pesan *Broadcast*



Gambar 17: Tampilan Riwayat Pengiriman *Broadcast Message*



Gambar 18: Daftar Penerima dan Status *Broadcast Message*

Tabel 2: Pengujian Sistem *Driver Broadcast Message*

No.	Skenario Pengujian	Validasi Input	Hasil Pengujian	Kesimpulan
1.	Menu *Broadcast Message - Create*	Klik *Mouse*	Muncul tampilan untuk mengisi *form* informasi sebelum mengirimkan *broadcast message* ke satu atau banyak mitra pengemudi	Sesuai harapan
2.	Kirim *Broadcast Message* Berdasarkan *Driver* NIP	Klik Mouse, mengisi data *broadcast* dan memilih NIP yang dituju	Pesan berhasil terkirim, dan diterima pada *Driver Apps* yang telah dilakukan *login* sesuai dengan NIP yang di-*input* kan	Sesuai harapan
3.	Kirim *Broadcast Message* Berdasarkan *Vehicle Number*	Klik Mouse, mengisi data *broadcast* dan mengisi *Vehicle Number* yang dituju	Pesan berhasil terkirim, dan diterima pada *Driver Apps* yang telah dilakukan *login*, dan telah di-*pairing* kan dengan IoT dengan *vehicle number* yang sesuai.	Sesuai harapan
4.	Kirim *Broadcast Message* Berdasarkan *Vehicle Type*	Klik Mouse, mengisi data *broadcast* dan mengisi *Vehicle Type* yang dituju	Pesan berhasil terkirim, dan diterima pada *Driver Apps* yang telah dilakukan *login*, dan telah di-*pairing* kan dengan IoT dengan *vehicle type* yang sesuai.	Sesuai harapan
5.	Kirim *Broadcast Message* Berdasarkan *Pool Area*	Klik Mouse, mengisi data *broadcast* dan mengisi *Pool Area* yang dituju	Pesan berhasil terkirim, dan diterima pada *Driver Apps* yang telah dilakukan *login*, dan *driver* tersebut berasal dari *pool area* yang di-*input* kan.	Sesuai harapan
6.	Kirim *Broadcast Message* Berdasarkan *Radius dan Area*	Klik Mouse, mengisi data *broadcast* dan mengisi *Radius dan Area* yang dituju	Pesan berhasil terkirim, dan diterima pada *Driver Apps* yang telah dilakukan *login*, dan titik *driver* tersebut berada sesuai dengan radius dan area yang di-*input* kan.	Sesuai harapan
7.	Menu *Broadcast Message - Browse*	Klik Mouse	Muncul tampilan dari riwayat *broadcast message* yang dikirim sebelumnya	Sesuai harapan

Penutup

Dengan implementasi REST, gRPC dan Pub/Sub sebagai media pertukaran data pada arsitektur *microservice* untuk sistem driver broadcast message PT. Blue Bird Tbk. berdasarkan analisa, perancangan, implementasi dan pengujian yang telah dilakukan, maka dapat ditarik kesimpulan:

1. Dengan adanya sistem *driver broadcast* yang berfungsi untuk operator menginformasikan ke satu atau banyak mitra pengemudi tentunya membawa manfaat mitra pengemudi mendapatkan informasi yang cepat lagi tepat, juga mempermudah operator dalam melakukan *broadcast message*.

2. Dari pengujian REST, gRPC, dan Pub/Sub yaitu ketiganya dapat digunakan sebaga media pertukaran pertukaran data antar service pada arsitektur *microservice*.

3. Penggunaan REST sangat cocok untuk interaksi luar seperti publikasi API dan penggunaannya lebih mudah dibandingkan menggunakan gRPC dan Pub/Sub yang lebih rumit. Penggunaan gRPC lebih cocok untuk interaksi antar *private services* pada arsitektur microservice, yang membutuhkan kecepatan pengiriman data, dan data yang ringan. Lalu penggunaan Pub/Sub sangat cocok untuk interaksi antar service yang membutuhkan proses *asynchronous* untuk pemrosesan data yang lebih cepat. .

Dari segi pemrosesan pada sistem ini, kecepatan menggunakan metode Pub/Sub untuk publish

message yang menggunakan proses asynchronous mencapai 328 RPS, lebih unggul dari gRPC yang hanya mampu melakukan proses sebanyak 186,35 RPS, lalu disusul oleh REST yang hanya mampu melakukan proses sebanyak 22 RPS, dikarenakan gRPC menggunakan Protobuf yang mana lebih ringan karena dapat mendukung pertukaran data dengan low-latency dibandingkan REST yang menggunakan JSON.

Dalam implementasi REST, gRPC dan Pub/Sub sebagai media pertukaran data pada arsitektur microservice untuk sistem driver broadcast message hanya melakukan pengetesan kecepatan pemrosesan, dan masih perlu pengujian yang lebih merinci seperti penggunaan memory, latency dan load pada processor di sisi server guna menghindari biaya overhead yang berlebih. Juga di sisi penggunaan sistem driver broadcast message nantinya perlu diadakan penambahan fitur untuk scheduled dan routine broadcast agar informasi yang mungkin sebagai informasi rutin tidak perlu buat secara manual lagi oleh operator.

Daftar Pustaka

[1] Shahir Daya, Nguyen Van Duy, Kameswara Eati, Carlos M Ferreira, Dejan Glozic, Vasfi Gucer, Manav Gupta, Sunil Joshi, Valerie Lampkin, Marcelo Martins, Shishir Narain and Ramratan Vennam, "Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach", Reedbook IBM, ISBN-13: 9780738440811, 2016.

[2] Mario Villamizar, Harold Castro and Merino Mauricio Verano Merino, "Evaluating the Monolithic and The Microservice Architecture Pattern to Deploy Web Applications in the Cloud", 10th Computing Colombian Conference (10CCC), pp. 583–590, Oktober 2015.

[3] Yuri Chandra Tri Putra, Thomas Adi Purnomo Sidi dan Joseph Eric Samodra, "Implementasi Arsitektur Microservice pada Aplikasi Web Pengajaran Agama Islam Home Pesantren", Jurnal Informatika Atma Jogja, Vol. 1, No. 1, pp. 88-97, November 2020.

[4] Gafri Putra Aliffansah, Tubagus Mohammad Akhriza dan Dwi Safiroh Utsalina, "Pengembangan Arsitektur Microservice di PT. Hatsonsurya Electric Untuk Peningkatan Skalabilitas dan Kemampuan Beradaptasi Layanan", SeNTIK, Vol. 6, No. 1, pp. 177-183, Agustus 2022.

[5] Suryono Atmojo, Ruli Utamin, Suzana Dewi dan Nurwahyudi Widhiyanta "Implementasi Sistem Informasi Desa Berbasis Arsitektur Microservice", STIKI Informatika Jurnal, Vol. 12, No. 1, pp. 55-66, Juni 2022.

[6] Antonio Messina, Riccardo Rizzo and Pietro Storniolo, "A Simplified Database Pattern for the Microservice Architecture", Conference: DBKDA 2016, ISBN: 978-1-61208-486-2, pp. 35-40, Juni 2016.

[7] Supit Tesalonika, Pelleng Frendy dan Rogahang Joula, "Analisis Sumber Dan Penggunaan Modal Kerja Pada PT. Blue Bird Tbk", Jurnal Administrasi Bisnis, Vol. 9, No. 1, pp. 147-148, Juni 2019..

[8] S. Sauda, N. Oktaviani dan M. Bunyamin, "Implementasi Metode Scrum dalam Pengembangan Test Engine Try Out Sertifikasi", JISKa, Vol. 3, No. 3, pp. 202–210, Agustus. 2019.

[9] Arief Rahmana, Muhammad Fauzy and Annisa Maharani Suyono, "5 Why Analysis Implementation to Detect Root Cause Of Rejected Product (Study At Aerospace Industry)", Turkish Journal of Computer and Mathematics Education, Vol. 12, No. 8, pp .1691-1695, April 2021.

[10] Sandi Pratama, Sulton Ibrahim, dan Muhammad Alfaqih Reybaharsyah, "Penggunaan Metode Scrum Dalam Membentuk Sistem Informasi Penyimpanan Gudang Berbasis Web", Jurnal Intech, Vol. 3, No. 1, pp. 27-35, Mei 2022.