ANALISIS PADA ARSITEKTUR MICROSERVICE UNTUK LAYANAN BISNIS TOKO ONLINE

Liqoo Mumbahiz Alchuluq[1], Fahrul Nurzaman[2]

E-mail: liqoo.mumbahiz13@gmail.com, fnurzaman@gmail.com

Abstrak

Toko online adalah sebuah website yang digunakan sebagai sarana untuk menawarkan produk – produk yang dijual belikan melalui Internet. Situs jenis ini biasanya menampilkan gambar atau video produk yang dilengkapi dengan deskripsi untuk menjelaskan keunggulan – keunggulan barang yang ditawarkan. Sehingga dapat menarik para pembeli walaupun mereka tidak dapat melihat barang secaralangsung. Dalam toko online memiliki banyak layanan dan toko online harus memilki aplikasi yang cukup stabil dalam menangani banyak layanan dan dalam menerima trafik pengungjung yang besar, sehingga untuk menangani kendala tersebut diperlukan arsitektur aplikasi yang mampu menjadi salah satu solusi dalam penyelesaiannya. Arsitektur yang digunakan untuk layanan bisnis toko online adalah Arsitektur Microservice. Tujuan penelitian adalah menganalisa penggunaan Arsitektur Microservice dalam layanan bisnis toko online jika dibandingkan dengan arsitektur yang banyak digunakan yaitu Arsitektur Monolitik. Hasil yang diperoleh adalah mengetahui apakah sebuah aplikasi perlu menggunakan Arsitektur Microservice dan kapan sebuah aplikasi pindah dari Arsitektur Monolitik ke Arsitektur Microservice. Dari hasil pengujian secara performance test dengan menggunakan 10000 data, Arsitektur Microservice bisa memproses data lebih cepat dengan perbandingan mencapai 888,441 / menit. Simpulan yang diperoleh adalah penggunaan Arsitektur Microservice terhadap layanan bisnis toko online dibuat berdasarkan hasil yang diperoleh dari tahap pengumpulan data, selanjutnya informasi tersebut digunakan untuk membuat desain Arsitektur Microservice dan implementasi Arsitektur Microservice pada layanan bisnis toko online dapat diterapkan pada kondisi ketika suatu aplikasi sudah berkembang menjadi besar seperti memiliki banyak fitur, traik yang besar, dan jika sampai server atau servicenya mati karena tidak kuat menampung beban trafik yang besar. Saran yang diperoleh, Aristektur Microservice ditambahkan konfigurasi keamanan (security) terhadap interaksi antar service.

Kata Kunci: *Arsitektur Microservice, Arsitektur Monolitik, Toko Online.*

1. Pendahuluan

Dalam era teknologi dan informasi sekarang ini, dapat disadari bahwa hampir semua aspek kegiatan disegala bidang ditentukan oleh kualitas dari teknologi dan informasi yang diterima dan dihasilkan. Dalam hal ini komputer mempunyai peranan yang sangat penting sebagai alat untuk mempermudah dalam penyampaian informasi yang ada sehingga dapat membantu memperlancar kinerja perusahaan.

Perkembangan teknologi juga terjadi pada arsitektur aplikasi, yaitu Microservices Architecture. Microservices Architecture semakin populer dan pengadopsi menikmati kesuksesan yang luar biasa. Laporan "Microservices Adoption in 2020", berdasarkan jajak pendapat 1.500 insinyur perangkat lunak, sistem dan arsitek teknis, insinyur dan pembuat keputusan, menyatakan bahwa lebih dari tiga perempat (77 persen) bisnis kini telah mengadopsi layanan mikro.

Dari mereka yang mengadopsi Microservices Architecture, hampir semua (92 persen) melaporkan tingkat keberhasilan yang tinggi. Lebih jauh lagi, sebagian besar bisnis (29 persen) bertaruh besar pada teknologi, tampak memindahkan sebagian besar sistem mereka ke layanan-layanan mikro. Jika dalam Monolith Architecture aplikasi terbungkus dalam satu package besar, dimana perubahan pada salah satu bagian kode program akan besar pengaruhnya

terhadap kode program yang lainnya. Sedangkan Microservices Architecture memiliki konsep aplikasi dibagi menjadi bagian-bagian kecil yang berfungsi spesifik dan tidak bergantung pada komponen program lainnya, konsep tersebut bertujuan agar sistem yang dibangun bisa menangani kegagalan total jika terdapat satu aplikasi yang bermasalah.
Memecah aplikasi menjadi layanan mikro memiliki manfaat rekayasa yang jelas termasuk peningkatan fleksibilitas, penskalaan yang disederhanakan, dan manajemen yang lebih mudah, semuanya menghasilkan pengalaman pelanggan yang lebih baik, terutama bagi toko online yang memiliki trafik pengunjung yang sangat tinggi.

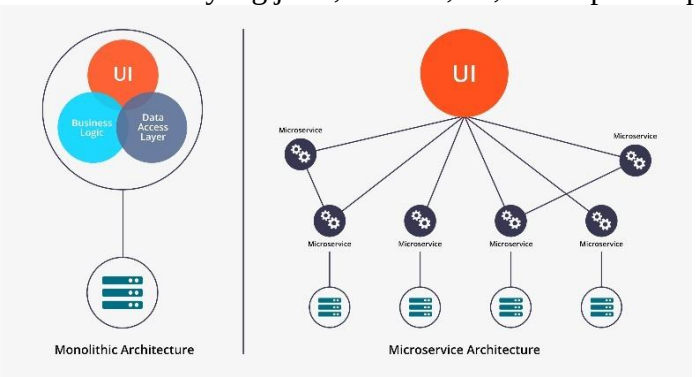
2. Tinjauan Pustaka

2.1 Arsitektur Monolitik

Arsitektur Monolitk merupakan *single deployment unit*, dimana sebuah aplikasi dibuat menjadi satu *package* besar yang berisi seluruh fitur pada aplikasi yang dibuat. Penamaan Arsitektur Monolitik ini populer setelah munculnya Arsitektur Microservice

2.2 Arsitektur Microservice

Microservice yang belum lama ini mulai populer di kalangan praktisi Software Engineering menawarkan pendekatan yang sedikit berbeda. Sistem Informasi enterprise yang pada umumnya dibangun dengan pendekatan monolitik (aplikasi terbungkus dalam satu package besar, dimana perubahan pada salah satu bagian kode program akan besar pengaruhnya terhadap kode program yang lainnya) digeser menjadi pendekatan terdisitribusi. Aplikasi dibagi menjadi bagian-bagian kecil yang berfungsi spesifik (high cohesion) dan tidak bergantung pada komponen program lainnya (loose coupling), dengan antarmuka API (Application Programming Interface), Newman, S. (2015).
Dari sudut pandang paradigma microservices, sebuah konsep kunci dalam resilient engineering adalah penyekatan. Jika salah satu komponen dari sistem gagal, kegagalan tersebut tidak akan cascade / memberikan pengaruh ke kinerja komponen yang lain. Dengan demikian, masalah dapat terisolasi dan sisanya dari komponen-komponen sistem yang lain dapat terus bekerja. Dalam sistem monolitik, jika sebuah layanan gagal, maka semuanya berhenti bekerja. Sistem dapat dijalankan pada beberapa mesin yang redundan untuk mengurangi kesempatan kegagalan/system failure. Sebaliknya, dengan microservices dapat dibangun sistem yang bisa menangani kegagalan total layanan karena layanan fungsionalitas sistem telah tersekat dalam batas yang jelas, Namiot, D., & Sneps- Sneppe, M. (2014).

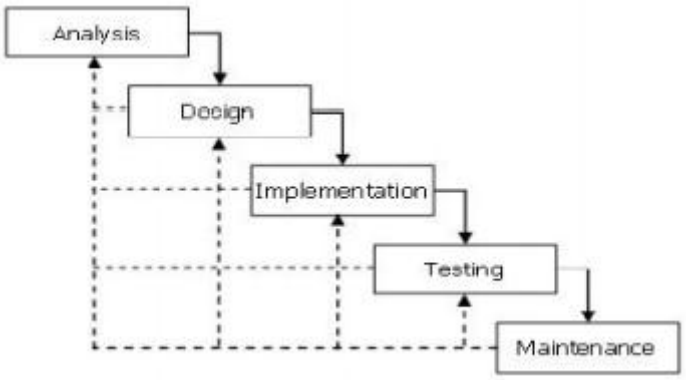


Gambar 1. Perbedaan Arsitektur Monolotik dengan Arsitektur Microservice

3. Metode Penelitian

Dalam membangun Arsitektur Microservice metode penelitian yang digunakan adalah model *SDLC (Software Development Life Cycle)*. *System Development Life Cycle (SDLC)* adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan

untuk mengembangkan sebuah sistem. *SDLC* juga merupakan pola yang diambil untuk mengembangkan sistem perangkat lunak, yang terdiri dari tahap – tahap : rencana (*planning*), analisis (*analysis*), desain (*design*), implementasi (*implementation*), uji coba (*testing*) dan pengelolaan (*maintenance*). Model *SDLC* yang dipakai dalam penelitian ini adalah model *Waterfall*. *Waterfall* Model atau *Classic Life Cicle* merupakan model yang paling banyak dipakai dalam *Software Enginnering (SE)*. Menurut Bassil (2012) disebut *waterfall* karena tahap demi tahap yang harus dilalui menunggu selesainya tahap sebelumnya dan berjalanberurutan.



Gambar 2. Metode Waterfall menurut Bassil (2012)

Pada tahap rencana (*planning*) diawali dengan mencari kebutuhan dari keseluruhan sistem yang akan diaplikasikan ke dalam bentuk software. Hal ini sangat penting, mengingat software harus dapat berinteraksi dengan elemen-elemen yang lain seperti hardware, database, dsb. Tahap ini sering disebut dengan *Project Definition*. Pada tahap ini peneliti melakukan observasi mengenai konsep dari Arsitektur Monolitik dan Arsitektur Microservice, pada observasi ini terfokus pada permasalahan yang terjadi ketika proses transaksi terjadi sangat banyak dalam waktu bersamaan.

Pada tahap analisis (*analysis*) proses pencarian kebutuhan diintensifkan dan difokuskan pada *software*. Untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti tentang domain informasi dari *software*. Berdasarkan hasil observasi, peneliti menganalisis ketika sebuah toko online mendapatkan pesanan dengan jumlah yang sangat besar, beban aplikasi akan semakin berat. Apakah kebutuhan fungsi *software* untuk memenuhi kendala yang dialami oleh *customer* saat melakukan transaksi.

Pada tahap desain (*design*) Proses ini digunakan untuk mengubah kebutuhan-kebutuhan diatas menjadi representasi ke dalam bentuk "*blueprint*" *software* sebelum *coding* dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti 2 aktivitas sebelumnya, maka proses ini juga harus didokumentasikan sebagai konfigurasi dari *software*.

Pada tahap implementasi (*implementation*) untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses coding. Tahap ini merupakan implementasi dari tahap *design* yang secara teknis nantinya dikerjakan oleh programmer. Pada tahap ini, peneliti membangun sebuah aplikasi berdasarkan desain "*blueprint*" yang telah dibuat. Pengembangan aplikasi ini dilakukan dari awal hingga aplikasi siap dijalankan.

Pada tahap uji coba (*testing*) semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari *error*, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya. Setelah proses pembangunan aplikasi selesai, peneliti melakukan

pengujian pada tahap ini. Aplikasi diuji menggunakan aplikasi bernama *jMeter* untuk membandingkan kecepatan transaksi pada Arsitektur Microservice dan Arsitektur Monolitik.

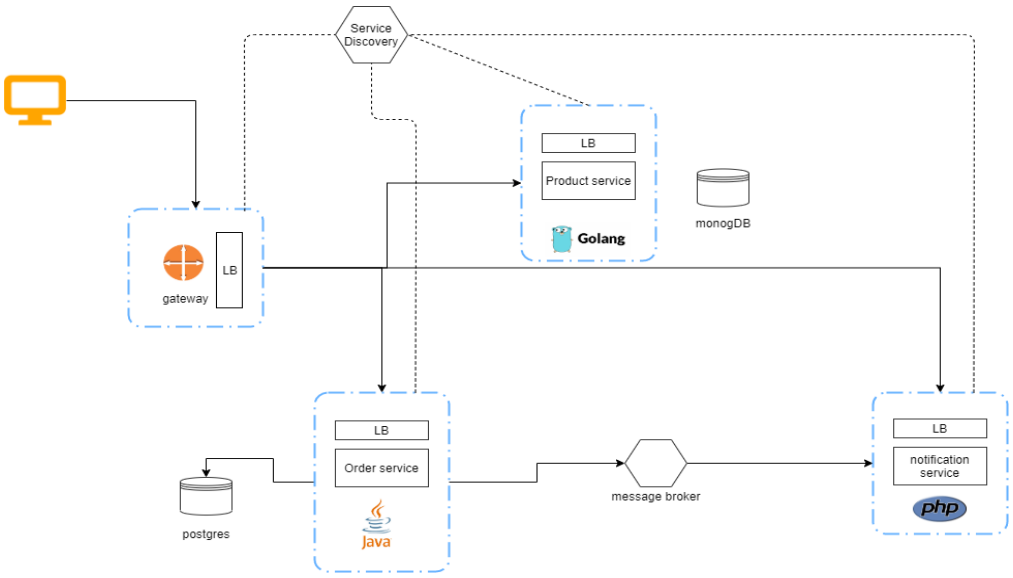
Pada tahap pengelolaan (*maintenance*) pengembangan diperlukan ketika adanya perubahan dari eksternal toko online seperti ketika ada pergantian sistem operasi, atau perangkat lainnya. Peneliti belum sampai pada tahap ini, sehingga tahap ini belum terlaksana. Rencana peneliti akan melakukan beberapa perbaikan tidak pada semua tahapan, namun hanya pada tahapan sebelum terjadi *error*. Sehingga peneliti tidak akan dipusingkan dengan melakukan tahapan dari awal hingga akhir kembali

4. Pembahasan

4.1 Analisis Kebutuhan Fungsional

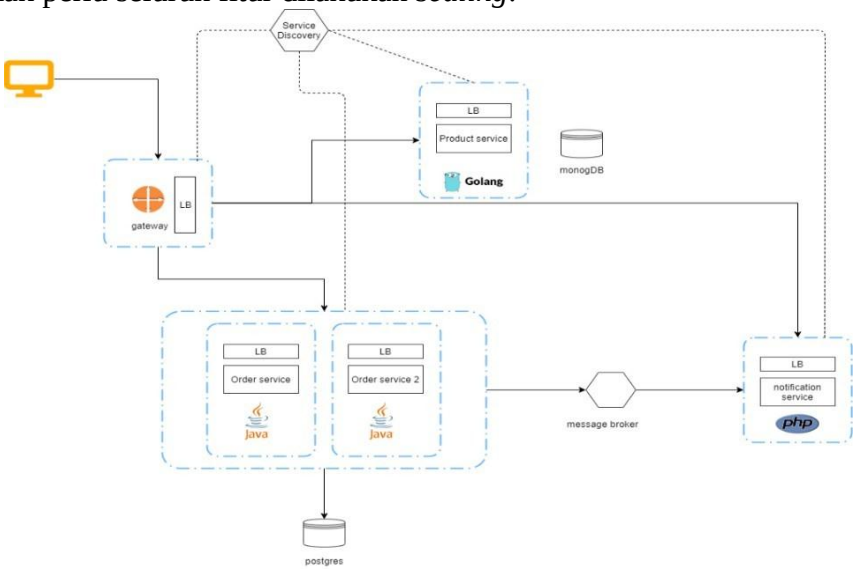
Analisis kebutuhan fungsional dilakukan untuk memberikan gambaran tentang fungsi dari sistem yang dirancang sehingga dapat mengatasi masalah – masalah yang terjadi, sistem tersebut nantinya mampu :

1. Jika diawal pembuatan aplikasi sudah melakukan penelitian, observarsi, dan pembahasan proses bisnis secara mendalam sehingga bisa diprediksi akan menjadi seberapa besar aplikasi yang dibuat, seberapa besar penggunanya serta seberapa besar trafik pengunjung, pemilik aplikasi bisa memulai dengan Arsitektur Microservice.
2. Dengan menggunakan Arsitektur Microservice aplikasi lebih mudah dimengerti karena *service* yang dibuat relatif kecil, berbeda dengan Arsitektur Monolitik yang sudah menjadi satu package aplikasi yang sangat besar dan akan membingungkan bagi *developer* baru untuk membaca flow aplikasi karena sudah terlalu kompleks.
3. Lebih mudah didevelop, dimaintain, dan deploy. Karena aplikasi dibuat relatif kecil itu lebih mudah di develop jadi tidak perlu khawatir ketika merubah satu fitur akan mengganggu fitur yang lain dengan begitu otomatis dimaintainnya lebih mudah dideploynya juga lebih mudah dan ringan tanpa perlu menunggu lama.
4. Lebih mudah untuk mengganti teknologi sesuai kebutuhan, jika pada awal pembuatan menggunakan bahasa pemrograman *PHP* ketika kita ingin melakukan *scaling* dan ingin berganti bahasa pemrograman yang lebih cepat seperti golang atau java maka itu menjadi lebih mudah dan *effort* yang dibutuhkan tidak terlalu besar karena aplikasi yangdibuat relatif kecil, jika menggunakan Arsitektur Monolitik lalu ingin merubah bahasa pemgrograman maka harus seluruh fitur dirubah bahasa pemrogramannya.



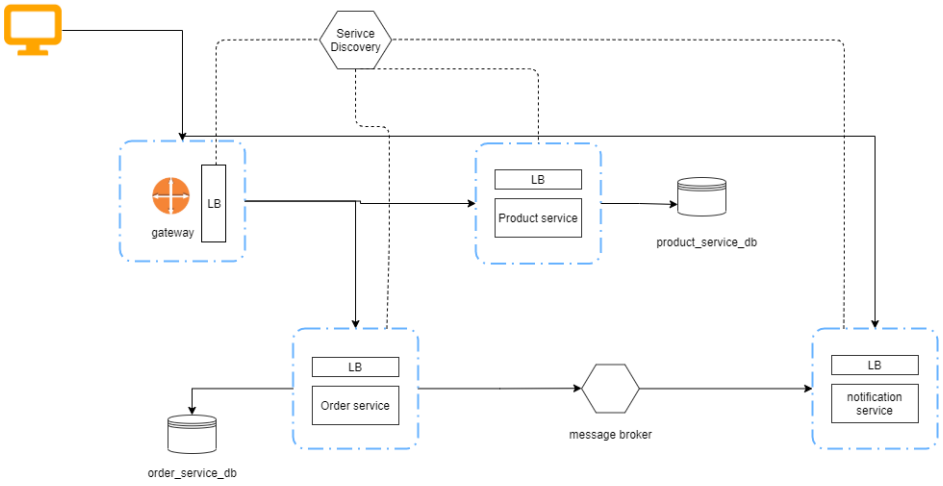
Gambar 3. Penerapan perbedaan teknologi pada Arsitektur Microservice.

5. Mudah discale sesuai kebutuhan, pada Arsitektur Microservice jika kita inginmelakukan scaling pada fitur penjualan maka hanya server fitur penjualan yang dilakukan scaling, jadi tidak perlu seluruh fitur dilakukan *scaling*.



Gambar 4. *Scaling* pada Arsitektur Microservice

4.2 Perancangan Arsitektur



Gambar 5. Arsitektur Microservice

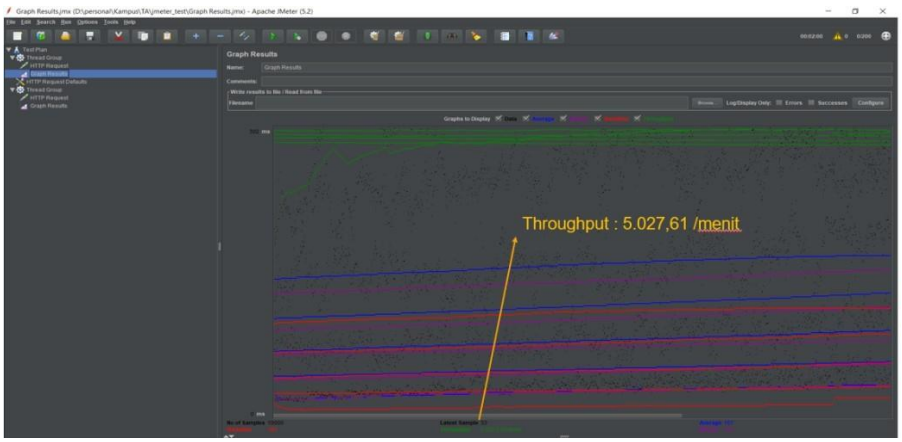
Pada gambar di atas terlihat rancangan Arsitektur Microservice yang akan diimplementasikan, terlihat pada gambar aplikasi dibagai menjadi beberapa aplikasi kecil yaitu, *Product Service*, *Order Service*, *Notification Service*, sedangkan untuk teknologi pendukung pada Arsitektur Microservice terdapat *Gateway*, *Service Discovery*, *dan Message broker*. Berikut adalah penjelasan masing – masing service dan teknologi pendukung yang digunakan :

1. *Product service*
   Berfungsi untuk menghandle fitur produk yaitu berupa, submit data produk, update data produk, list of data produk, hingga menghapus produk.
2. *Order Service*
   Pada service ini berfungsi untuk menghandle fitur order seperti submit order dan menampikan data order berdasarkan customer yang memesan.

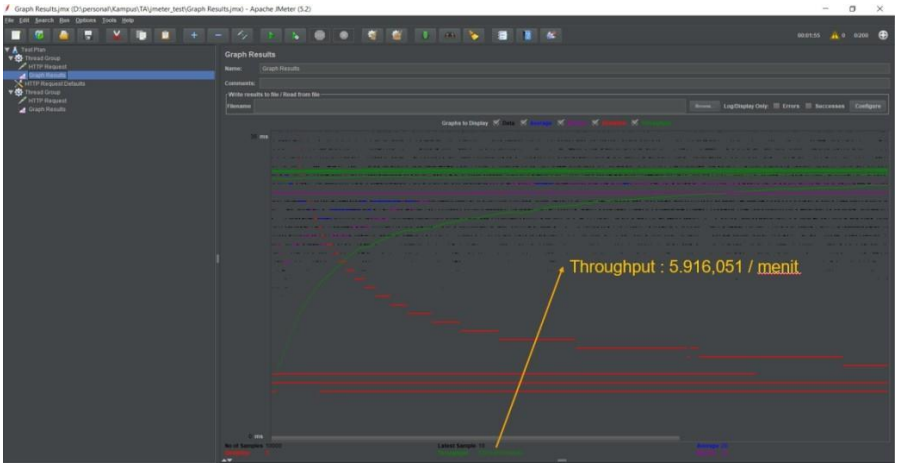
3. *Notification service*
   Setelah customer melakukan pesanan *order service* maka system akan mengarahkan pada *notification service* untuk mengirim notification kepada customer bahwa pesanan yang dilakukan berhasil atau gagal.
4. *Gateway*
   Berfungsi sebagai gerbang antara akses publik ke dalam akses internal aplikasi, pada *gateway* pula masing – masing service dilakukan routing, sebagai contoh kita memiliki 3 *service* yaitu *order service, product service, dan notification service*. Masing – masing service memiliki IP, contoh 172.10.10.1 untuk order service 172.10.10.2 product service dan 172.10.10.3 untuk *notification service,* maka *gateway* akan memberikan alias nama pada masing – masing service "order" untuk *order service*, "product" pada *product service*, "notification" pada *notification service*. Maka ketika *website* membuka halaman produk menggunakan alamat api.toko.com/product/, untuk *order service* menjadi api.toko.com/order, sedangkan *notification service* menjadi api.toko.com/notification.
5. *Service Discovery*
   Cara kerja *service discovery* adalah mencari *service* secara otomatis pada jaringan komputer dengan cara menyimpan semua daftar *list of service* dan juga *end point service*.
6. *Message Broker*
   Message broker digunakan sebagai jembatan komunikasi antara *order service* dengan *notification service, message broker* menggunakan istilah *producer* dan *consumer* dimana *order service* sebagi *producer* dan *notification service* sebagai *consumer*. Pada implementasinya saat *customer* melakukan pemesanan barang maka *order service*akan mengirimkan pesan ke *message broker* dan *message broker* akan meneruskannyake *notification service*. Keuntungan menggunakan *message broker* adalah tidak ketergantungan *service* antara *order service* dengan *notification service*, jadi ketika *notification service* mengalami masalah atau servernya mati maka *order service* bisa tetap melakukan transaksi. Lalu bagaimana dengan notifikasi yang seharusnya dikirm setelah melakukan pemesanan tetapi *notification service* mati, pada saat *notification service* mati maka data – data pesan yang akan dikirimkan ke *notification service* akan di simpan di *message broker*, lalu jika *notification service* sudah diperbaik dan menyala kembali maka pesan – pesan yang sebelumnya berada di *message broker* akan diteruskan ke *notification service*.

4.3 Performance Test

Pada tahap pengujian yang dilakukan yaitu masing – masing arsitektur akan di lakukan *performance test* dengan 10000 data menggunakan aplikasi *JMeter*. Pada pengujian akan dilakukan oleh 100 user, kemudian diberlakukan jeda akses antar user dengan total jeda 100 detik, karena user yang mengakses sebanyak 100 orang dan total jeda 100 detik maka jeda akses antar user adalah 1 detik (100 detik / 100 user), lalu diterapkan sebarapa banyak user akan mengakses yaitu 100 kali. Dapat disimpulkan bahwa pada pengujian ini akan dilakukan oleh 100 user dengan jeda masing – masing user selama 1 detik dan masing – masing user melakukan akses sebanyak 100 kali pada akhirnya akan di lakukan pengujian sebanyak 10000 *http request*. Pada pengujian tersebut dibuat skenario saat user ingin melihat data produk sebanyak 200 data.



Gambar 6. Hasil Performance Test pada Arsitektur Monolitik



Gambar 7. Hasil Performance Test pada Arsitektur Microservice

Pada saat performance test menunjukan API yang menggunakan microservice architecture dapat mengakses data lebih cepat.

Tabel 1. Perbandingan Hasil

Skenario	Arsitektur Monolitik	Arsitektur Microservice	Perbedaan
Get 200 product	5.027,61 / menit	5.916,051 / menit	888,441 / menit

5. Kesimpulan dan Saran

Dari proses analisis, perancangan, implementasi dan pengujian yang dilakukan pada bab-bab sebelumnya, maka dapat diambil kesimpulan sebagai berikut:

1. Telah dihasilkan suatu aplikasi sederhana dengan menggunakan Arsitektur Microservice yang dapat menjadi solusi bagi para pemilik toko online dalam mengembangkan aplikasinya.
2. Telah dihasilakan perbandingan antara penerapan Arsitektur Microservice dengan Arsitektur Monolitik dengan merujuk pada hasil dari pengujian.

3. Pada hasil pengujian didapati Arsitektur Microservice memilki hasil *performance test* yang lebih baik disbanding Arsitektur Monolitik.

Pada implementasinya Arsitektur Microservice digunakan ketika sudah mendapat trafik yang sangat besar dan berimbas pada performa aplikasi yang menjadi lambat, server berulang kali mati karena tidak kuat menampung trafik yang besar, semakin bertambahnya fitur – fitur sehingga struktur aplikasi menjadi sangat besar, dengan semakin bertambah fitur bertambah juga *developer* yang menghandle aplikasi tersebut sehingga sering terjadi konflik dalam pengerjannya.

Untuk pengembangan sistem selanjutnya, dapat diberikan saran-saran sebagai berikut :

1. Bisnis proses yang diterapkan bisa lebih luas lagi tidak hanya melihat produk, memesan barang dan melihat barang yang sudah dipesan.
2. Ditambahkan konfigurasi keamanan (*security*) terhadap interaksi antar *service* pada Arsitektur Microservice.

Daftar Pustaka

[1] Jogianto HM, 2005, "Sistem Teknologi Informasi", Andi, Yogyakarta.

[2] Gerald Jerry Fitz F dan Warren D Steling, 2007, "Analisis dan Perancangan Sistem Informasi untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern", Andi, Yogyakarta.

[3] Teguh Wahyono, 2004 : 14, "Sistem Informasi (Konsep Dasar, Analisis Desain dan Implementasi)".

[4] Gata, Windu dan Gata, Grace, 2013, "Sukses Membangun Aplikasi Penjualan dengan Java", Elex Media Komputindo, Jakarta.

[5] A. S., Rosa dan Shalahuddin, M, 2013, "Rekayasa Perangkat Lunak Terstruktur Dan Berorientasi Objek", Informatika , Bandung.

[6] Taufik, A. E, 2017, "Perancangan Sistem Informasi Pemesanan Pentas Seni Berbasis Web Pada Sanggar Seni Getar Pakuan Bogor". IJSE – Indonesian Journal on Software Engineering, 3(2), 1–7.

[7] Budi Raharjo, 2011, "Belajar Otodidak Membuat Database Menggunakan MySQL", Informatika, Bandung.

[8] Adhi, Prasetio, 2012, "Buku Pintar Pemrograman Web", Mediakita, Jakarta.

[9] Riadi. M, 2013, "*Teori Basis Data (Database)"*, https://www.kajianpustaka.com/2012/10/teori-basis-data-database.html.

[10] Deni sutaji, 2012, "Sistem Inventory Mini Market Dengan PHP dan Jquery-Cet I". Penerbit Lokomedia, Yogyakarta, ISBN : 978-979-1758-82-6.

[11] Newman, S, 2015, "Building Microservices", O'Reilly Media, Inc.

[12] Namiot, D., & Sneps-Sneppe, M, 2014, "On micro-services architecture", International Journal of Open Information Technologies, 2(9).

[13] Khin Me Me Thein, 2014, "Apache Kafka: Next Generation Distributed Messaging System".

[14] F. Gutierrez, 2014, "Spring Boot, Simplifying Everything", Introducing Spring Framework, pp. 263–176.