BAB II

TINJAUAN PUSTAKA

Membagi sebuah domain sistem yang besar menjadi sebuah *service-service* kecil yang merupakan keunggulan dari *microservices* yang merupakan pendekatan yang dinilai tepat untuk membangun sistem dengan skala besar, kompleks dan penggunaan nya bersifat jangka panjang telah banyak diterapkan di berbagai bidang sistem informasi salah satu nya adalah pariwisata. Pada tahun 2017 Nuno Silvaa dan Paulo Toméb merancang arsitektur *microservices* untuk sistem pariwisata, tujuan penggunaan *microservices* adalah karena dengan menggunakan arsitektur tersebut *service-service* dapat dibuat dengan menggunakan berbagai bahasa pemerograman seperti C, C++, C#, Go, Java, Node.js, Objective-C, PHP, Python maupun Ruby karena berjalan menggunakan Docker *engine* sehinga mempercepat proses pengembangan sistem (Nuno Silvaa, Paulo Toméb, 2017).

Berdasarkan survey yang dilakukan oleh situs www.leanix.net pada tahun 2017 tentang *microservices* dengan menggunakan kuisioner *online* dan ruang sample 118 orang yang merupakan *CIOs, Heads of IT, Enterprise Architects* dan *Senior IT* di Amerika dan Eropa. Dari data yang didapatkan dapat disimpulkan bahwa

1. Perusahaan yang menggunakan *microservices* dalam proses pengembangan software baru, mampu diselesaikan 5 kali lebih cepat daripada mereka yang tidak menggunakan layanan *microservices*.
2. 71% perusahaan akan mengintensifkan penggunaan layanan *micrososervices* pada tahun 2017.
3. Perusahaan yang sudah menggunakan *microservices* sangat puas dengan hasil yang mereka dapatkan dan berencana untuk lebih intensif mengimplementasikan nya dalam waktu 6 bulan ke depan.
4. Perusahaan yang telah mengimplementasikan *microservices* masih memiliki kendala yang sama dengan perusahaan yang belum

mengimplementasikan. Kendala dan data hilang yang terjadi pada sistem lama tidak mengalami perubahan.

5. Alasan utama perusahaan tidak mengimplementasikan *microservices* adalah karena kurang nya pengetahuan dan tenaga ahli. (Leanix, 2017).

Terdapat juga beberapa penelitian yang dapat digunakan sebagai referensi pembangunan *microservices* ini. Penelitian-penelitian tersebut *microservices* untuk berbagai kebutuhan. Salah satunya adalah (Hatma Suryotrisongko 2017) tentang penerapan *microservices* untuk mengubah sistem manajemen anggota AISINDO (Asosiasi Profesi Sistem Informasi Indonesia) yang beraksitektur monolitik menjadi *microservices* dengan tujuan menguji resiliensi atau kemampuan sistem untuk beradaptasi terhadap perubahan maupun kesalahan/kerusakan infrastruktur. Hasil penelitian yang didapatkan adalah Model/desain arsitektur *software*/sistem informasi yang telah dikembangkan menunjukan aspek kualitas resiliensi. Dengan pendekatan *microservices* dan Docker *container* yang telah dikembangkan dengan sistem *back-end* Spring Boot dan *front-end* Angular2. Pola komunikasi REST *Service* dengan format JSON memberikan kemudahan dalam implementasi dan juga tidak membebani *server*.

Selanjutanya penelitian oleh (Stephen Abrams, John Kunze, David Loy 2017) untuk melakukan migrasi dari arsitektur monolitik ke *microservices* dilakukan University of California pada sistem kurasi digital yang mereka miliki dengan tujuan untuk menciptakan suatu sistem yang independen hal ini ditujukan untuk mengurangi ketergantungan terhadap teknologi tertentu dan lebih mudah melakukan perubahan pada sistem terkait dengan kebijakan lokal.

Penelitian yang dilakukan oleh (Ghifari Munawar 2018) tentang analisis model arsitektur *microservices* pada sistem informasi DPLK. Pada penelitian ini sistem monolitik DLPK yang terdiri dari beberapa modul seperti sistem pendaftaran nasabah, sistem pengelolaan investasi, sistem pengelolaan bisnis dan sistem pelaporan diubah kedalam *microservices* melalui 6 tahapan migrasi yang dimulai dari analisa setiap subsistem monolitik terhadap fungsi bisnis dan tabel basis data yang diharapkan dapat menghasilakan sebuah sistem dapat lebih adaptif terhadap perubahan kebutuhan. Dari penelitian ini dapat disimpulkan bahwa

pendekatan ini cukup efektif sebagai tahap awal dari proses migrasi sistem monolitik ke *microservices*, karena *service* yang memiliki ketergantungan pada lebih dari 1 (satu) subsistem tidak akan menjadi kandidat *microservices*. Teknologi yang digunakan dalam penelitian ini adalah DBMS PostgreSQL, *framework* Springboot, Bahasa Pemrograman Java, serta Apache Tomcat sebagai *web server*.

Sejanlan dengan penelitian yang dilakukan oleh (Arthur de M. Del Esposte, Fabio Kon, Fabio M. Costa dan Nelson Lago, 2017) tentang arsitektur *microservices* untuk membangun *smart city*. Penggunaan arsitektur *microservices* ditujukan karena dalam implementasi *smart city* aplikasi yang dibangun harus bersifat fleksibel dengan kebijakan keamanan, aturan yang berlaku dan kebijakan privasi yang sewaktu-waktu dapat berubah. Pengujian *microservices* dilakukan dengan mengirim data yang berasal dari *device* IoT *smart city* secara berkelanjutan dalam waktu tertentu. Hasil yang didapat adalah sebagai berikut:

1. Dari 350 *device* terdapat 0.01% *request* yang gagal di proses
2. Dari 600 *device* terdapat 0.16% *request* yang gagal di proses
3. Dari 250 *devic*e terdapat 0% *request* yang gagal di proses

Dari penelitian tersebut penulis mencoba mengubah arsitektur monolitik pada sistem KRS Online di Universitas Atma Jaya Yogyakarta menjadi *microservices* untuk menangani beban berat pada saat pengisian KRS yang dilakukan di Universitas Atma Jaya Yogyakarta. Berbeda dengan penelitian sebelum nya *microservices* yang dibangun akan menggunakan .NET Core yang akan berjalan diatas Docker.

Tabel 2. 1. Tabel Perbandingan dengan Penelitian Terdahulu

No	Unsur Pembanding	Silvaa & Paulo (2017)	Suryotrisongko (2017)	Esposte, Kon, Costa & Lago (2017)	Munawar (2018)	Abrams, Kunze, Loy (2010)	Kusuma* (2018)
1	Konten	Microservices Architecture for Tour Activities Application	Arsitektur Microservice untuk Resiliensi Sistem Informasi	InterSCity: A Scalable Microservice-based OpenSource Platform for SmartCities	Analisis Model Arsitektur Microservice Pada Sistem Informasi DPLK	An Emergent Micro-Services Approach to Digital Curation Infrastructure	Pembangunan Backend Sistem Informasi KRS Online Berbasis Microservices
2	Basis data	Mongodb, Postgres SQL	MySQL	PostgreSQL, RabbitMQ, dan Redis	PostgreSQL	File Storage	SQL Server, InfluxDB
3	Bahasa Pemrograman	C, C++, C#, Go, Java, Node.js, Objective-C, PHP, Python, Ruby	Java	C, C++, C#, Go, Java, Node.js, Objective-C, PHP	Java	Java dan Ruby	C#, HTML, Java Script

No	Unsur Pembanding	Silvaa & Paulo Toméb (2017)	Suryotrisongko (2017)	Arthur de M. Del Esposte, Fabio Kon, Fabio M. Costa2 & Nelson Lago (2017)	Munawar (2018)	Abrams, Kunze, Loy (2010)	Kusuma* (2018)
4	Tools	Notepad++, Docker	Notepad++, Docker	VS Code, Docker	Netbeans, Docker	Netbeans, Docker	Notepad++, Docker
5	Orchestrator	Docker Swarm	Docker Swarm	Docker Swarm	Docker Swarm	Docker Swarm	Docker Swarm
6	Cloud Platform	Physical	Amazon ECS	Azure	Digital Ocean	Bare Metal	Azure

*) Sedang dalam proses penelitian