BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Pada penelitian ini menggunakan lima buah tinjauan pustaka, Tinjauan Pustaka pertama diulis oleh Manfred Sneps-Sneppe dan Dmtry Namiot, untuk membangun aplikasi dalam menyebarkan aplikasi lebih mudah dari Arstitektur Monolitik ke Arsitektur Microservices. (Manfred Sneps-Sneppe dan Dmtry Namiot tahun (2014)). Tinjauan Pustaka kedua ditulis oleh Indra Yatini B. S.Kom., M.Kom, bagaimana membuat aplikasi web yang menyerupai aplikasi desktop dengan J-Query. (Indra Yantini B. S.Kom., M.Kom tahun (2014)).

Tinjauan Pustaka yang ketiga ditulis oleh Jose Ignacio Fernandes-Villamor, Carlos A. Iglesias serta Mercendes Garijo, klasifikasi *framework* yang ringan untuk REST. (Jose Ignacio Fernandes-Villamor ., dkk. (2010)). Tinjauan pustaka yang keempat ditulis oleh Zuniar Rizqi Prasetyo, Untuk Mencari Lokasi SPBU Terdekat. (Zuniar Rizqi Prasetyo tahun (2015)) . Dan tinjauan pustaka yang terakhir ditulis oleh Kartika Puji Pangesti, sistem informasi kearsipan sekolah. Detail dari Tinjauan Pustaka disajikan dalam bentuk tabel yang terlihat pada Tabel 2.1 Tabel Perbandingan :

Tabel 2.1 Tabel Perbandingan

Parameter / Penulis	Objek	Bahasa Serveside, Clientside	DataBases	Platform	Private Cloud
Manfred Sneps-Sneppe dan Dmtry Namiot tahun (2014)	Transisi dari TDM ke IP jaringan.	Tidak Terindentifikasi	Tidak Terindentifikasi	Tidak Terindentifikasi	Tidak Menggunakan
Indra Yatini B. S.Kom., M.Kom tahun (2014)	Aplikasi web seperti Aplikasi Desktop.	Bahasa - PHP, JavaScript dan JQuery.	Tidak Terindentifikasi	Tidak Terindentifikasi	Tidak Menggunakan
Jose Ignacio Fernandes-Villamor ., dkk. (2010)	Klasifikasi *framework* untuk Arsitektur REST.	Tidak Terindentifikasi	Tidak Terindentifikasi	Tidak Terindentifikasi	Tidak Menggunakan
Zuniar Rizqi Prasetyo tahun (2015)	Lokasi SPBU terdekat dikota Jepara dan Kudus.	Bahasa - JavaScript.	NoSQL-MonggoDB	Node.js	Tidak Menggunakan
Kartika Putri Pangesti tahun (2015)	Sistem informasi kearsipan sekolah.	Bahasa - PHP	NoSQL-MonggoDB	Tidak Terindentifikasi	Tidak Menggunakan
Usulan (2016)	Pengelolaan Skripsi.	Bahasa - JavaScript	NoSQL-MonggoDB	Node.js	Docker

2.2. Dasar Teori

2.2.1 Arsitektur Microservice

Arsitektur Microservice, atau hanya *Microservice*, adalah metode khas mengembangkan sistem perangkat lunak yang digunakan berkat skalabilitas, metode arsitektur ini dianggap sangat ideal ketika harus mengaktifkan dukungan untuk berbagai perangkat lunak dan perangkat-mencakup *web, mobile, Internet Of things, cloud service.* Karakteristik tertentu dari *Arsitektur Microservice* pada dasarnya adalah metode pengembangan aplikasi perangkat lunak sebagai *suite independen deployable*, *small,* layanan modular di mana setiap layanan menjalankan proses yang unik dan berkomunikasi melalui didefinisikan dengan baik, mekanisme ringan untuk melayani tujuan bisnis.

Layanan berkomunikasi satu sama lain tergantung pada kebutuhan aplikasi yang digunakan, pengembang menggunakan *HTTP/REST* dengan *JSON* atau *protubuf*. *Profesional DevOps*, tentu saja pengembang bebas untuk memilih protokol komunikasi yang mereka anggap cocok, tetapi dalam banyak situasi, *REST* (*Representational State Transfer*) adalah metode integrasi yang berguna karena kompleksitasnya relatif lebih rendah lebih protokol lainnya.

Martin Fowler menunjukkan contoh pengunaan *Microservice* ialah Netflix, eBay, Amazon, Layanan Digital Pemerintahan Inggris, realestate.com.au, Forward, Twitter, PayPal, Gilt, Bluemix, SoundCloud, The Guardian, dan banyak situs skala besar lainnya dan aplikasi ini semua berevolusi dari *arsitektur monolitik* ke *arsitektur microservice*. (Huston Tom, 2015)

2.2.2 Node.js

Node.js® merupakan salah satu peranti pengembang yang bisa digunakan untuk membuat aplikasi berbasis *Cloud*. Node.js dikembangkan dari *engine JavaScript* yang dibuat oleh Google untuk *browser Chrome / Chromium (V8)* ditambah dengan *libUV* serta beberapa pustaka internal lainnya. Dengan menggunakan Node.js, semua pengembangan akan dilakukan menggunakan *JavaScript*, baik pada sisi klien maupun server. (Bambang Purnomosidi D.P, 2013)

2.2.3 Database NoSQL

Database NoSQL adalah *database* yang tidak menggunakan relasi antar tabel dan tidak menyimpan data dalam format tabel kaku (kolom yang fix) seperti layaknya *Relasional Database.* (Ferdhika Yudira , 2015)

2.2.4 MongoDB

MongoDB adalah salah satu produk *databases NoSQL OPEN SOURCE* yang menggunakan struktur data *JSON* untuk menyimpan datanya. MongoDB adalah salah satu *database NoSQL* yang paling populer di internet. MongoDB sering dipakai untuk aplikasi berbasis *Cloud, Grid Computing*, atau *Big Data.* (Putra Adi Candra, 2015)

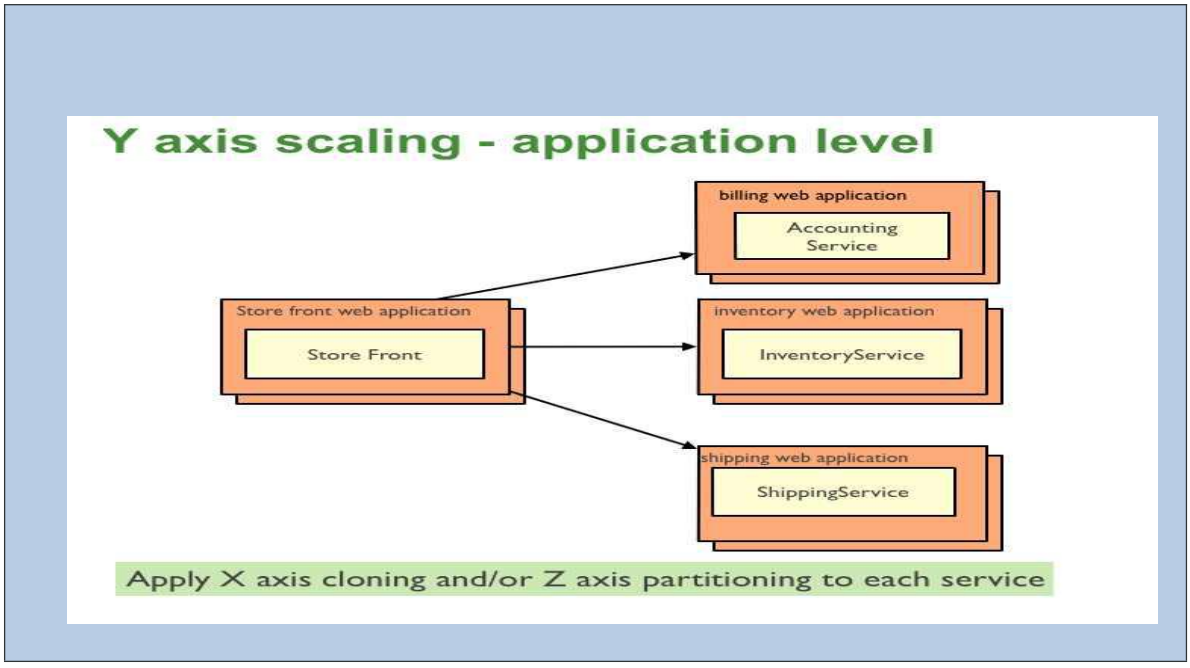
2.2.5 Senecajs

Seneca adalah sebah *toolkit microservices* untuk *Node.js* yang bermotto "***Build it Now, Scale it Later!***" yang mana disini seneca sudah menyediakan *plugin* untuk fondasi pada aplikasi yang akan dibangun. Hal ini membuat programer berfokus pada pembuatan *code* atau bisa disebut *coding*. Setiap kali memanggil salah satu kode

yang tidak diketahui atau yang mana perintah apa yang ingin dijalankan maka satu objek *javascript* akan menjalankanya dan yang lain akan berhenti. (Richard Rodger, 2015)

2.2.6 Skema Arsitektur Microservices Pada Aplikasi Web

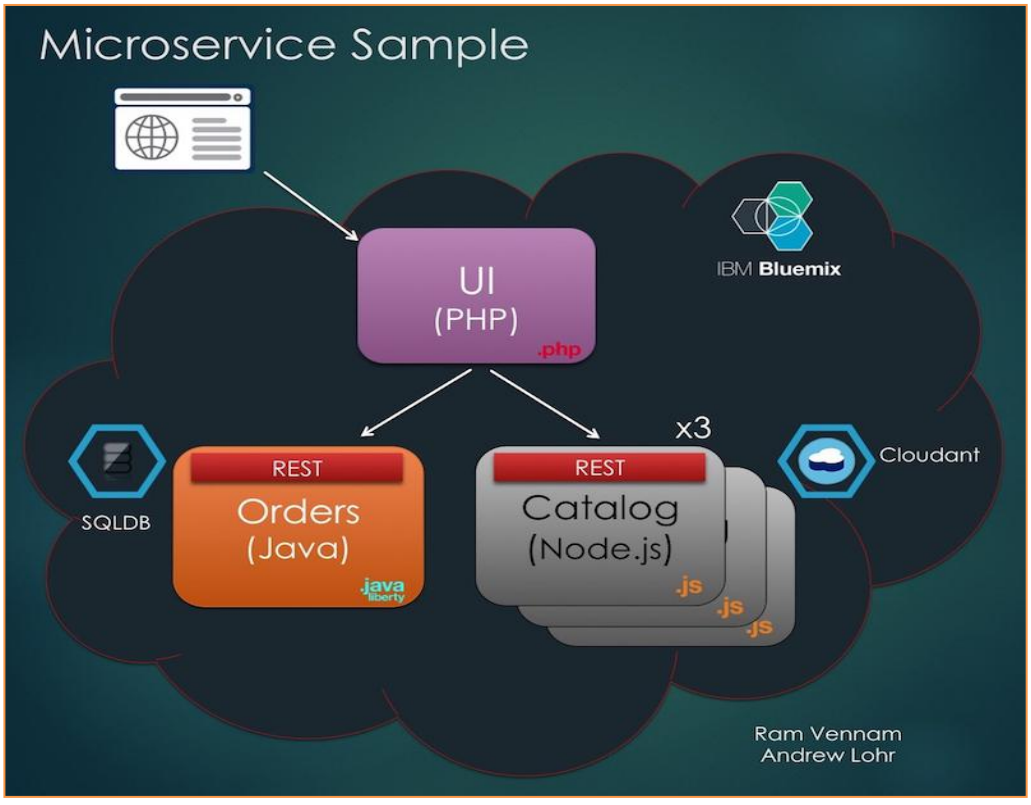
Skema Arsitektur Microservice menggambarkan hubungan antara satu layanan dengan layanan lain, yang digambarkan dalam implementasi suatu aplikasi web. Dapat digambarkan sebagai berikut.



Gambar 2.1 Skema Arsitektur Microservice (a)

Diatas adalah skema *Microservices* dalam aplikasi web yang mencontohkan Aplikasi e-commerce yang melaksanakan layanan mulai dari mengambilkan pesanan pelangan, memverifikasi persedian, ketersediaan kredit dan pengiriman pesanan-pesanan. Dalam aplikasi ini terdiri dari beberapa komponen termasuk *StoreFrontUI*, yang mengimplementasikan antarmuka pengguna, bersama layanan *backend* untuk memeriksa kredit ,menjaga persediaan dan pengiriman pesanan. Dalam

aplikasi ini setiap layanan digunakan secara independent dari layanan-layanan lain sehingga lebih mudah untuk membuat versi baru dari layanan. Aplikasi ini digunakan sebagai satu set layanan.



Gambar 2.2 Skema Arsitektur Microservice (b)

Skema Arsitektur Microservice juga pernah dibuat oleh Ram Vennam dalam blognya yang menggambarkan penerapan Arsitektur Microservice dalam aplikasi e-commerce yang didefinisikan oleh tiga aplikasi terpisah. Salah satu kekuatan utama dari Arsitektur Microservice adalah kemampuan untuk memilih bahasa yang ingin digunakan untuk setiap aplikasi. Pada skema aplikasi diatas pada bagian backend aplikasi menggunakan Node.js untuk bagian layanan katalog karena kemampuan *non-blocking* untuk kemampuan melayani banyak permintaan, dan untuk penggunaan *databases* dengan menggunakan *databases non-relasional [databases Cloudant]* pada katalog item. Untuk pesanan menggunakan Java Jax-RS dikarenakan

kemampuanya menangani dan memproses banyak pesanan ditoko dengan menggunakan *databases SQL Relasional.* Sedangkan PHP dipilih untuk membuat *UI* dikarenakan untuk mempermudah dalam Membuat UI (*User Interface*). (Richardson Chris, 2014 & Lohr Andrew, 2015)

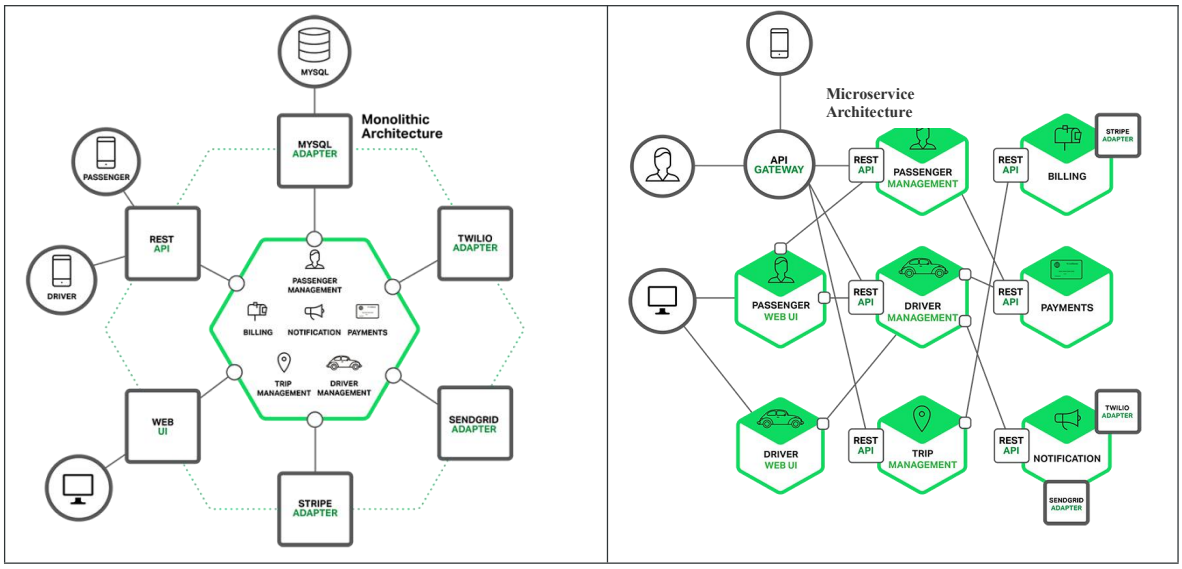
2.2.7 Docker

Docker adalah sebuah platform terbuka untuk *developer*, *sysadmin* atau siapapun yang bertujuan menggunakan sebuah platform untuk membangun, mendistribusikan dan menjalankan aplikasi dimanapun mulai dari laptop, data center, virtual mesin ataupun cloud. Docker membuat proses pemaketan aplikasi bersama komponennya (*dependencies*) secara cepat dalam sebuah *container* yang terisolasi, sehingga dapat dijalankan dalam infrastruktur local (*local data center*) ataupun cloud tanpa melakukan perubahan/konfigurasi lagi pada *container*, selama host menjalankan Docker Engine.

Docker merupakan *software open source* di bawah Lisensi Apache Versi 2.0 yang bisa dipergunakan secara gratis. Saat ini Docker hanya bisa berjalan pada Linux, namun kita bisa menggunakan mesin virtual pada mesin windows, atau menggunakan Boo2Docker. (Fahmi Persada, 2014)

2.2.8 Keungulan Arsitektur Microservice

Microservices memliliki sejumlah keungulan dan untuk mengilustrasikan keungulan dari arsitektur microservice ini dengan membadingkanya dengan Arsitektur Monolitik. Digambarkan sebagai berikut.



Gambar 2.3 Perbandingan Arsitektur Monolitik dan Arsitektur Microservice

Pada Pengilustrasian mencontohkan tetang pembagunan aplikasi baru yakni pemesanan taxi. Pada penggambaran Arsitektur Monolitik pengembang membuat satu layanan lalu hanya perlu menyalin satu aplikasi yang telah dibuat lalu dikemas/packets ke server. Dalam waktu-kewaktu aplikasi ini menjadi besar sehingga menjadi suatu aplikasi yang kompleks dan sangat kompleks untuk pengembang tunggal untuk memahami aplikasi tersebut. Akibatnya untuk memperbaiki bug dan menerapkan fitur baru menjadi benar-benar sulit dan memakan waktu jika baris kode sulit dipahami. Sehingga untuk menagulangi hal tersebut dibuatlah Arsitektu Microservice yang mana setiap area fungsional dijalankan oleh MICROSERVICE sendiri. Selain itu aplikasi web dibagi menjadi seperangkat aplikasi web sederhana (seperti satu untuk penumpang dan satu untuk driver contoh memanggil taxi). Hal ini membuat lebih mudah menerapkan layanan yang berbeda bagi pengguna tertentu, perangkat, atau penggunaan khusus.

Dari Penjabaran penggambaran Aplikasi Microservice diatas dapat diterjemahkan keungulan Microservice.

1. Menangani masalah kompleksitas.

2. Arsitektur ini memungkinkan setiap layanan untuk dikembangkan *independent* oleh tim yang difokuskan pada layanan tersebut.

3. Memungkinkan layanan untuk digunakan secara mandiri.

4. Baris kode ditulis lebih sedikit dan lebih banyak membuat flexibilitas untuk membuat perubahan pada aplikasi yang dikelola.

5. Setiap layanan dapat dipantau, diperbaruhi secara *independent* untuk mencocokan tautan.

6. Setiap layanan dapat dikembangkan dan diperbarui secara mandiri.

7. Menghilangkan komitmen jangka panjang untuk setumpuk teknologi.

(Richardson Chris, 2015).