# BAB I PENDAHULUAN

## 1.1 Latar Belakang

Sistem computer adalah interaksi dari perangkat lunak dan perangkat keras yang membentuk sebuah jaringan elektronik. Tugas dari sebuah sistem adalah menerima input, memproses data input, menyimpan data olahan, dan menampilkan output sebagai bentuk informasi. Dalam penerapannya, kita menyebut sistem aplikasi sebagai program komputer yang bertugas untuk menyelesaikan kebutuhan khusus. Terdapat beberapa tahapan umum dalam mengembangkan sistem aplikasi yaitu perencanaan, analisa, desain, pengembangan, testing, implementasi, dan pemeliharaan [1]. Tahap yang cukup penting dan akan menjadi fokus diskusi disini adalah desain dan pengembangan, yang dimana peran arsitektur perangkat lunak sangat berperan penting untuk menetapkan landasan dasar pengembangan aplikasi dari awal sampai selesai. Hasi dari arsitektur perangkat lunak merupakan struktur-struktur yang menjadikan landasan untuk menentukan keberadaan komponen-komponen perangkat lunak, cara komponen-komponen untuk saling berinteraksi dan organisasi komponen-komponen dalam membentuk perangkat lunak [2]. Secara umum perangkat lunak bekerja untuk pengguna pada *desktop* *browser, mobile browser,* dan aplikasi *browser* lainnya. Aplikasi tersebut mungkin akan menggunakan API (*Application Programming Interface*) sebagai pihak ke 3. Aplikasi juga dapat saling berintegrasi dengan aplikasi lain dengan menggunakan *web service*. Aplikasi bekerja dengan menerima *request* (HTTP *request* dan pesan) dengan menjalankan logika perhitungan, mengakses database, bertukar pesan dengan sistem lain, dan mengembalikan HTML/JSON/XML sebagai respon balikan [4].

IEEE 803:1993 mengelompokkan kebutuhan non-fungsional ke dalam sejumlah kategori kualitas dari suatu perangkat lunak, diantaranya yaitu : ketepatan (*correctness)*, *robustness,* performa, ketersediaan dan kualitas antarmuka (*interface)*, keandalan (*reability),* ketersediaan (*availability*). Ketika skala aplikasi masih kecil dan sedikit data yang digunakan, kebutuhan masih mudah untuk dipenuhi, namun ketika aplikasi semakin besar, akan terjadi masalah yang selain disebabkan oleh data yang banyak, namun juga oleh *load* komputasi yang besar yang berasal dari *multiple user* dari berbagai lokasi.

Model arsitektur yang paling sering digunakan saat ini adalah model monolitik. Arsitektur monolitik merupakan arsitektur yang mudah dimengerti dan di modifikasi karena lebih sederhana implementasinya. Arsitektur ini menggunakan kode sumber dan teknologi yang serupa untuk menjalankan semua tugas-tugasnya. Contoh yang dapat diambil adalah aplikasi Wordpress. Wordpress merupakan contoh yang mudah untuk menggambarkan sebuah aplikasi monolitik, dimana semua fitur seperti *security*, performa, manajemen konten, statistik dan semuanya dibangun dengan menggunakan PHP dan database MySQL dalam kode yang sama [5]. Secara garis besar keunggulan dari arsitektur monolitik dapat dirasakan apabila aplikasi ingin mudah untuk dikembangkan, mudah untuk di *deploy,* dan dapat selalu dipantau pertumbuhan perfomanya [5].

Namun apabila aplikasi semakin besar dan anggota tim semakin banyak, arsitektur monolitik akan menghadapi kekurangan yang semakin lama akan semakin signifikan. Pertama, ketika aplikasi semakin besar, barisan code monolitk akan menyulitkan *developer* terutama yang baru bergabung bersama tim, aplikasi akan sulit dimengerti dan di modifikasi. Akibatnya pertumbuhan aplikasi akan melambat dan terlebih karena sulit dimengerti, kualitas kode akan semakin menurun. Kedua, semakin banyak code yang ditulis, maka akan semakin lambat IDE (Integrated Development Environment) yang digunakan, semakin tidak produktif pula proses *development* yang dilakukan. Hal ini juga berpengaruh pada waktu yang dibutuhkan untuk menjalankan aplikasi pertama kali, serta menjadi semakin sulit untuk memodifikasi aplikasi. Seperti untuk mengubah sebuah komponen, *developer* harus *redeploy* keseluruhan aplikasi. Ketiga, akan sulit untuk membagi team secara fungsionalitas, seperti misalnya membagi tim akunting dan tim inventori. Kedua tim tersebut tidak dapat secara mandiri bekerja sendiri, karena hanya ada 1 aplikasi besar yang mengakibatkan adanya saling ketergantungan.

Model arsitektur Microservice adalah pattern alternatif yang dapat mengatasi keterbatasan dari arsitektur monolitik, model ini mulai muncul ke permukaan di tahun 2015 (Google trend). Menurut artikel yang saya kutip, kebanyakan aplikasi mulai dari arsitektur monolitik, sampai hingga aplikasi itu sulit di kembangkan lagi, kemudian aplikasi dipecah menjadi model microservis, hal itu yang terjadi pada perusahaan besar seperti Netflix dan Amazon [8]. Secara garis besar arsitektur microservis mendefinisikan struktur service yang lebih sempit dengan area fungsi yang saling berkaitan. Tiap servis saling berkomunikasi menggunakan protokol seperti HTTP dan setiap servis bisa mempunyai databasenya sendiri masing-masing. Arsitektur microservis mengubah servis aplikasi menjadi modul yang mandiri, kecil (dibandingkan monolitik), dan setiap servis berjalan sesuai dengan perannya masing-masing dan tidak saling ketergantungan [6]. Dalam artikel yang dijelaskan oleh Chris Richardson, ada banyak pattern model dari arsitektur microservis, tergantung dari aplikasi yang akan dimigrasi.

Dalam contoh kasus penelitian ini, peneliti akan menerapkan arsitektur microservis pada aplikasi monolitik rumah sakit Beta, aplikasi rumah sakit Beta telah mengalami banyak kendala dalam hal perawatan dan pengembangan lanjutan terhadap aplikasinya. Hal ini dapat dirasakan dari performa yang semakin menurun secara sekuensial. Penurunan performa tersebut disebabkan selain karena aplikasi yang terus membesar, juga karena data input dan data olahan yang semakin hari semakin banyak.

Tujuan dari penelitian ini adalah menerapkan pattern microservis yang paling cocok untuk kasus aplikasi rumah sakit beta, menjelaskan bagaimana tahapan migrasi ke arsitektur microservis, menjelaskan bagaimana analisis yang baik agar dapat menentukan desain arsitektur microservis yang benar, serta membandingkan performa antara arsitektur microservis dan arsitektur monolitik dari segi kecepatan, beban hardware yang dibutuhkan, kemudahan pengembangan, *high availability*, dll.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang ada, maka permasalahan yang akan dihadapi yaitu :

1. Bagaimana menentukan pattern microservis dan desain arsitektur yang paling tepat/sesuai untuk aplikasi rumah sakit beta.
2. Bagaimana melakukan migrasi dari arsitektur monolitik ke model arsitektur microservice.
3. Bagaimana melakukan perbandingan performa yang dihasilkan dari arsitektur microservis yang baru terhadap arsitektur monolitik.

## 1.3 Batasan Masalah

Berikut adalah batasan masalah yang terdapat pada penelitian ini:

1. Penelitian tidak mencakup semua bagian dari aplikasi rumah sakit beta, namun hanya mengambil bagian (…)
2. Segala data yang digunakan dalam penelitian bersifat tertutup karena menyangkut hal privasi dari rumah sakit.
3. Fokus utama dari penelitian ini adalah berupa analisis perancangan arsitektur dan uji coba perbandingan performa yang dibuat dalam bentuk prototype.

## 1.4 Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah, tujuan dari penelitian ini adalah :

1. Menentukan bagaimana tahap yang benar untuk melakukan migrasi dari arsitektur monolitik ke arsitektur microservis.
2. Membandingkan apa saja kelebihan dan kekurangan dari arsitektur microservis dibandingkan dengan arsitektur monolitik.
3. Mengetahui bagaimana analisa untuk membagi servis monolitik menjadi servis microservis yang lebih fokus dan padat.

## 1.5 Manfaat Penelitian

1. Memberikan panduan untuk melakukan migrasi dari arsitektur konfensional ke arsitektur microservis.
2. Memberikan hasil analisa perbandingan performa dari kedua jenis arsitektur.

## 1.6 Metodologi Penelitian

Metodologi atau langkah-langkah yang dilakukan dalam penelitian ini, yaitu:

1. Analisis

Melakukan studi literatur dan analisa melalui wawancara dengan pihak perancang aplikasi monolitik sebelumnya. Data juga dikumpulkan dari jurnal-jurnal, karya ilmiah, dan situs yang memberikan informasi yang menunjang mengenai konsep arsitektur microservis dan tahap-tahap implementasi pada aplikasi monolitik.

1. Identifikasi

Melakukan identifikasi mengenai target-target yang ingin dicapai setelah menerapkan arsitektur baru. Identifikasi ini berguna untuk menentukan perbandingan apa saja yang akan diperhatikan antara arsitektur microservis dan arsitektur monolitik ketika pengujian dilakukan.

1. Perancangan

Perancangan arsitektur microservis meliputi perancangan model dasar (proses bisnis), perancangan sistem basis data, perancangan kelas s*ervice* yang baru dengan konsep microservis, juga perancangan jalur komunikasi dan pertukaran data antara kelas-kelas *service* yang akan dibuat.

1. Implementasi

Melakukan implementasi hasil perancangan dalam bentuk *web service application (server side).* Selanjutnya dibuat aplikasi *client* sederhana untuk input data dan pengujian performa dari arsitektur server yang telah dibuat.

1. Pengujian

Melakukan pengujian terhadap rancangan aplikasi microservis yang baru dan aplikasi monolitik dengan menggunakan data rumah sakit untuk mengetahui hasil kinerja dan perbandingan performa antara kedua jenis arsitektur.

## 1.7 Sistematika Penulisan

Sistematika penulisan digunakan untuk mendokumentasikan laporan penelitian secara utuh dan lengkap. Sistematika yang dibentuk adalah sebagai berikut:

1. Bab I Pendahuluan

Berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan laporan penelitian.

1. Bab II Landasan Teori

Membahas tentang definisi dari arsitektur microservis, teori-teori pendukung,, dan metode penerapan arsitektur baru yang akan dijadikan landasan dan dipelajari serta dirangkum dari berbagai sumber, seperti buku, karya tulis, jurnal, artikel dari situs ilmiah. Juga pembahasan sekilas mengenai arsitektur konvensional yang digunakan sebelumnya.

1. Bab III Analisa dan Perancangan

Berisi analisa mengenai bagaimana konsep penerapan arsitektur microservice pada aplikasi rumah sakit dan modul-modul apa saja yang akan di migrasi menjadi modul baru untuk diimplementasikan pada tahap selanjutnya. Selanjutnya membahas perancangan aplikasi dari hasil analisa dengan menggunakan arsitektur microservis yang baru.

1. Bab IV Implementasi dan Pengujian

Berisi implementasi dari hasil perancangan aplikasi dalam bentuk perangkat lunak menggunakan teknologi *web service*. Selanjutnya perangkat lunak diuji fungsi dan performanya dari sisi *client* (pengguna).

1. Bab V Kesimpulan dan Saran

Memberikan kesimpulan berdasarkan hasil analisis, perancangan, implementasi dan pengujian, serta evaluasi yang dilakukan, juga saran-saran yang dibutuhkan untuk pengembangan lebih lanjut.

# BAB II LANDASAN TEORI

[1] Hartono, M. (7). Langkah mudah membangun sistem informasi ERP. *Jakarta: PT. Elex Media Komputindo*.

[2] Pangera, A. A., & Ariyus, D. ARSITEKTUR SISTEM OPERASI.

[3] Pradhananga, Y., & Rajarajeswari, P. (2017). Tiarrah Computing: The Next Generation of Computing. *International Journal of Informatics and Communication Technology (IJ-ICT)*, *6*(2), 129-138.

[4] Chris Richardson (2017). Pattern: Monolithic Architecture.

[5] Polychniatis, T., van der Rijnst, S., van Vliet, R., & Wirken, G. (2013). Software architecture WordPress. *Utrecht, NE: Utrecht University. URL: http://www. cs. uu. nl/wiki/pub/Swa/CourseLiterature/arch-E. pdf [June 24, 2015]*.

[6] Chris Richardson (2017). Pattern: Microservice Architecture

[7] Mazzara, M., Mustafin, R., Safina, L., & Lanese, I. (2016). Towards microservices and beyond: An incoming paradigm shift in distributed computing. *arXiv preprint arXiv:1610.01778*.

[8] Thönes, J. (2015). Microservices. *IEEE Software*, *32*(1), 116-116.