# 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].

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 team semakin banyak, arsitektut 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, baru 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/REST dan setiap servis bisa mempunyai database nya sendiri masing-masing. Arsitektur microservis mengubah servis aplikasi menjadi modul yang mandiri, kecil (dibandingkan monolitik), dan setiap servis berjalan secara unik [6]. Dalam artikel yang dijelaskan oleh Chris Richardson, ada banyak pattern model dari arsitektur microservis, tergantung dari aplikasi yang akan di migrasi.

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 menemukan pattern microservis yang paling cocok untuk diterapkan dalam kasus aplikasi rumah sakit beta, menjelaskan bagaimana tahapan migrasi ke arsitektur microservis, menjelaskan bagaimana analisa 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, dan … … .

## 1.2 Rumusan Masalah

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

1. Menentukan pattern microservis yang paling cocok untuk aplikasi rumah sakit beta.
2. Menganalisa desain arsitektur microservis yang benar untuk kasus rumah sakit beta.
3. Melakukan migrasi servis dari arsitektur lama ke model arsitektur yang baru.
4. 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.

## 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 pecahan 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. Penentuan Topik

Penentuan topik dilakukan untuk mengetahui bidang apa yang dapat diselesaikan dengan adanya dukungan sistem informasi.

1. Studi Literatur

Studi literatur dilakukan untuk mempelajari dan menggali berbagai infomrasi yang terkait dengan topik penelitian. Studi dilakukan dengan membaca buku-buku referensi, jurnal-jurnal ilmiah, artikel-artikel di internet, dan sumber informasi lainnya yang dianggap relevan dengan penelitian.

1. Metode Observasi

Metode observasi dilakukan dengan melakukan pengamatan langsung terhadap tempat studi kasus penelitian, yaitu penyebaran infromasi jamu. Metode observasi dilakukan dengan mendatangi tempat studi kasus penelitian untuk memperoleh gambaran yang lebih jelas mengenai kondisi penyediaan sarana informasi untuk jamu.

1. Metode Kuesioner

Metode kuisioner digunakan untuk mengumpulkan informasi yang diperlukan dalam melakukan analisis organisasi, kejelasan masalah, dan tujuan kebutuhan studi kasus.

1. Analisis Solusi Permasalahan

Melakukan analisis terhadap permasalahan yang ada untuk menemukan solusi yang dapat membantu dalam menjawab permasalahan yang ada.

1. Pengumpulan dan Pengolahan Data

Pengumpulan data dilakukan untuk mengetahui fakta-fakta dan menjadikanya sebagai acuan dasar dalam penelitian.

1. Penentuan Metodologi

Penentuan metodologi dilakukan agar dapat membantu usulan solusi.

1. Perancangan Sistem

Melakukan perancangan sistem berdasarkan kebutuhan yang telah diusulkan.

## 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

Memuat secara singkat mengenai beberapa teori yang digunakan sebagai landasan analisis maupun pengembangan sehubungan dengan topik tugas akhir.

1. Bab III Analisis Masalah

Berisi tentang identifikasi masalah, analisis penyebab masalah hingga analisis solusi.

1. Bab IV Pengumpulan dan Pengolahan Data

Memuat informasi terhadap cara pengumpulan dan pengolahan data menjadi sebuah informasi.

1. Bab V Perancangan Sistem

Berisi tentang rancangan desain sistem informasi yang akan dirancang.

1. Bab VI Implementasi dan Pengujian

Berisi tentang hasil implementasi dan pengujian terhadap sistem yang diimplementasi, serta bagaimana sistem yang telah dibuat dapat menjadi solusi terhadap permasalahan pada tempat studi kasus penelitian ini dilakukan.

1. Bab VII Kesimpulan dan Saran

Berisi tentang kesimpulan mengenai hasil penelitian yang telah dilakukan dan saran yang dapat diberikan.

[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.