BAB I

PENDAHULUAN

1.1 Latar Belakang

Sistem informasi adalah suatu kombinasi terartur apapun dari pengguna, perangkat keras, perangkat lunak, komputer, jaringan, komunikasi data dan basis data yang mengumpulkan, mengubah dan menyebarkan informasi di dalam suatu bentuk organisasi. (O'Brien, 2005).

Universitas Atma Jaya Yogyakarta telah menerapkan penggunaan sistem informasi pada berbagai proses bisnis nya. Termasuk dalam proses pengambilan rencana studi yang dilaksanakan pada awal semester dengan cara melakukan input langsung kedalam sistem KRS *online*. Pengisian sistem KRS secara *online* saat ini belum dapat dilakukan oleh seluruh mahasiswa Atma Jaya Yogyakarta dan hanya dilakukan pada fakultas teknik yang dikarenakan arsitektur sistem yang digunakan belum mampu menangani permintaan pengguna dalam jumlah yang banyak.

Pengambilan KRS yang dilakukan secara serentak oleh setiap mahasiswa di Universitas Atma Jaya Yogyakarta tentu mengakibatkan beban yang berat pada *server* yang digunakan. Hal tersebut dapat menyebabkan *server* yang digunakan untuk pengambilan KRS mengalami *down* sehingga mengakibatkan seluruh kegiatan pengambilan KRS terhenti. Jika pada arsitektur *monolitik* yang digunakan ketika muncul permasalahan seperti beban berat dan *server* yang digunakan mengalami peningkatan penggunaan CPU dan RAM yang dapat mengakibatkan *server* yang digunakan mengalami *down*, masalah tersebut biasanya diselesaikan dengan menambah jumlah *core* dan RAM pada *server* namun dirasa masih belum efektif, maka dibutuhkan sebuah arsitektur baru yang dapat selalu tersedia pada saat proses pengambilan KRS.

Microservices adalah sebuah pendekatan untuk mengembangkan aplikasi dengan rangkaian *service* yang kecil, yang mana setiap *service* berjalan pada proses nya sendiri-sendiri. Setiap *service* dapat berkomunikasi dengan

mekanisme yang ringan. Keuntungan yang didapatkan dalam penerapan arsitektur *microservices* pada KRS *online* adalah sistem yang akan dibuat menjadi *scaleable* yang dimana kita ketahui sumber data dari sistem KRS berasal dari beberapa sumber yaitu penjadwalan, penawaran kelas dan mahasiswa dimana dengan arsitektur *microservices* kita dapat membuat *service-service* kecil sesuai dengan domain nya masing-masing untuk membentuk sistem KRS.

Setiap *service* akan berkomunikasi menggunakan API sehingga dapat memudahkan kita dalam mengelola data secara *real time* karena data yang dihasilkan akan berbentuk JSON yang dapat diolah dengan javascript framework di *browser* pengguna sehingga meringankan beban *server*. Ditambah lagi telah banyak koorporasi di dunia yang telah menerapkan arsitektur ini, menurut survei dari 118 responden 70% sudah bermigrasi dari sistem yang bersifat monolitik ke *microservices*. (leanix.com, 2017).

Dengan penggunaan arsitektur *microservices* kita membatasi *resource* yang dapat digunakan seperti besar penggunaan RAM dan CPU pada tiap *service* sehingga kita dapat memberikan *resource* yang lebih besar kepada *service* yang lebih membutukan *resource* yang besar, selain itu arsitektur *microservices* juga memungkinkan replikasi dan distribusi *service* yang kita miliki pada *server* sehingga jika suatu *service* yang kita miliki mati maka akan dialihkan ke replika-replika nya dan jika kita memiliki beberapa *server* maka *service* yang mati tersebut akan dipindahkan ke *server* yang beban nya lebih sedikit sehingga tiap-tiap *server* memiliki beban yang merata.

Pada pembuatan *microservices* untuk KRS *online* ini akan menggunakan bahasa pemrograman C# yang akan diimplementasikan dalam bentuk SDK .NET Core 2.1, penggunaan .Net Core 2.1 dipilih karena telah di dukung di berbagai sistem operasi serti Windows, Mac Os dan varian Linux. .Net Core 2.1 memiliki kebutuhan sistem yang ringan dan kompatibel dengan Docker *container* yang akan kita gunakan.

SignalR digunakan sebagai *service* untuk menampilkan data secara *real time* karena selain karena dapat berjalan pada .Net Core 2.1, SiganalR merupakan solusi yang tepat dalam komunikasi yang bersifat *broadcast*.

Pemilihan Docker sebagai *container* dari *microservices* yang akan dibuat karena dalam Docker kita tidak membangun mesin *virtual* sendiri sehingga lebih hemat RAM, CPU dan *storage*. Waktu yang diperlukan dalam *startup* Docker juga sangat cepat bahkan jauh lebih cepat daripada *server* riil. Ini terjadi karena Docker berbagi pakai kernel Linux dari *server* riil, tidak memerlukan instalasi sistem operasi di dalam *container*. *Container* hanya berisi *project* yang akan kita bangun berserta semua *library* yang dibutuhkan. Sedangkan sebagai media penyimpanan data akan memanfaatkan *database server* yang sudah dimiliki Universitas Atma Jaya Yogyakarta dengan DBMS SQL Server 2008 R2.

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan, maka rumusan permasalahan pada tugas akhir adalah sebagai berikut:

1. Arsitektur monolitik tidak dapat menangani beban berat pada saat proses pengisian KRS pada Universitas Atma Jaya Yogyakarta
2. Pada arsitektur monolitik tidak dapat dilakukan *scaling* pada tiap *service domain* sehingga tidak bisa diketahui lama waktu eksekusi dan jumlah *resource* yang digunakan.

1.3 Tujuan

Pada penelitian ini akan dilakukan dua hal untuk menyelesaikan masalah tersebut yaitu:

1. Pemanfaatan fitur *load balancing* yang dimiliki oleh Docker yang membuat *backend* pengisian KRS dapat menangani beban berat saat pengisian KRS di Universitas Atma Jaya Yogyakarta.
2. Membangun *backend* berbasis *microservices* dengan memecah aplikasi monolitik yang sudah ada menjadi beberapa *service domain* sehingga dapat dilakukan *scaling*.

1.4 Batasan Masalah

Dari perumusan masalah, backend berbasis *microservices* ini dapat dibangun dengan batasan-batasan sebagai berikut:

1. *Microservices* yang akan dibuat hanya akan digunakan pada saat proses pengisian KRS di Universitas Atma Jaya Yogyakarta.
2. *Load balancing* akan dilakukan dengan menggunakan *swarm mode* yang merupakan fitur dari Docker.

1.5 Metodologi Penelitian

Metodologi yang dilakukan dalam pembangunan backend berbasis *microservices* ini yakni:

1. Studi Pustaka

Pada bagian langkah studi pustaka ini dilakukan untuk mencari referensi atau sumber pustaka yang serupa dan berkaitan dengan aplikasi yang dibuat. Secara khusus pada langkah ini dilakukan pencarian referensi dalam pembangunan *microservices* seperti bagai mana cara melakukan *scaling* terhadap *service domain* yang ada. Pada tahapan ini dapat membantu memberikan gambaran untuk langkah selanjutnya dan juga memberikan teori-teori yang ada. Selain itu, melalui studi pustaka ini didapatkan data-data yang merupakan hasil dari referensi atau penelitian yang sudah ada dan tertulis.

2. Observasi

Tahap observasi ini digunakan untuk mengetahui langkah-langkah dalam pembuatan pembangunan *microservices*. Pada bagian ini secara khusus akan dilakukan observasi dalam melakukan pendekatan terhadap suatu lokasi dengan metode yang sudah didapatkan pada saat studi pustaka. Selain itu, pada tahap ini juga akan dilihat bagaimana sebelumnya pengambilan KRS di Universitas Atma Jaya Yogyakarta dilakukan.

3. Analisis Kebutuhan dan Perancangan Aplikasi

Tahapan analisis kebutuhan aplikasi dilakukan untuk memotret kebutuhan sistem baik fungsional maupun non fungsional. Analisis dilakukan dengan melakukan studi literatur dan wawancara narasumber yaitu kepala Kantor Sistem Informasi Universitas Atma Jaya Yogyakarta, dari data-data yang sudah dikumpulkan melalui studi literatur dan wawancara maka *microservices* akan dirancang sesuai dengan analisis-analisis kebutuhan yang telah kita buat.

4. Pembangunan Microservices

Tahap pembangunan *microservices* dilakukan untuk mengembangkan rancangan yang telah dihasilkan pada tahap sebelumnya. Hasilnya adalah sebuah *backend* berbasis *microservices* yang mampu menangani pelaksanaan KRS secara online di Universitas Atma Jaya Yogyakarta.

5. Pengujian Microservices

Tahap pengujian *microservices* dilakukan untuk mengetahui apakah *microservices* yang telah dibangun mampu menangani menangani pelaksanaan KRS secara online dengan dan mencegah terjadinya *down* di Universitas Atma Jaya Yogyakarta.

1.6 Sistematika Penulisan Tugas Akhir

BAB 1: Pendahuluan

Bab ini berisi pembahasan latar belakang, masalah, tujuan pembuatan sistem menggunakan arsitektur *microservices*, batasan-batasan dan metodologi yang digunakan untuk mengerjakan tugas akhir ini, dan sistematika penulisan tugas akhir.

BAB 2: Tinjauan Pustaka

Bab ini berisi tentang uraian singkat hasil-hasil penelitian terdahulu yang ada hubungannya dengan permasalahan yang akan ditinjau penulis yang berhubungan dengan topik penelitian dalam Tugas Akhir ini.

BAB 3: Landasan Teori

Bab ini berisi penjelasan tentang dasar teori yang digunakan penulis dalam melakukan perancangan dan pembuatan *microservices* yang dapat digunakan sebagai pembanding atau acuan di dalam pembahasan masalah.

BAB 4: ANALISIS DAN PERANCANGAN SISTEM

Bab ini akan membahas analisis dan perancangan *microservices* seperti perancangan arsitektur dan load balancing.

BAB 5: IMPLEMENTASI DAN PENGUJIAN MICROSERVICES

Bab ini akan membahas pengujian ketahanan dan pengujian performa yang dimiliki oleh *microservices*.

BAB 6: KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan mengenai penelitian yang dilakukan serta saran-saran yang dapat digunakan untuk pengembangan dimasa yang akan datang.