BAB I. PENDAHULUAN

1.1. Latar Belakang

Dengan perkembangan teknologi internet yang menjadi bagian tidak terpisahkan pada bermacam bidang, sistem pengelolaan konten memberikan dukungan yang kuat untuk informasi dan menjadi bagian penting untuk informasi perusahaan dan pemerintahan. Pada masa lalu, sistem aplikasi tradisional mengadopsi arsitektur monolitik karena ukurannya yang kecil, struktur yang sederhana dan populasi pengguna yang kecil, serta keperluan komunikasi waktu nyata yang sangat rendah. Pada arsitektur monolitik ini modul fungsional dan operasi data dari seluruh sistem operasi diperlakukan secara keseluruhan dan diatur secara seragam, dikembangkan dan di-deploy [1].

Bersamaan dengan kenaikan konsep pemisahan depan dan belakang dan kedewasaan teknologi yang bersangkutan, arsitektur yang memisahkan depan dan belakang membuat pengembangan sistem dapat menerapkan *decoupling*. Hasil implementasi ini menghasilkan *single-page application* (SPA). SPA itu sendiri secara teori, merupakan aplikasi dimana peramban tidak memerlukan pemuatan halaman saat digunakan, dimana pengguna tidak berpindah halaman, saat terjadi *request* kepada *server* setiap kali terjadi suatu peristiwa pada aplikasi. Berdasarkan SPA, tim *frontend* membuat dan memelihara aplikasi web yang menggunakan REST API untuk mengambil data dari layanan *backend*. Pendekatan ini memberikan pengalaman yang baik untuk pengguna, namun membuat SPA tidak terskala dengan baik. Seiring berjalannya waktu dan mengikuti perkembangan bisnis, kondisi ini membuat proyek *frontend* menjadi sangat sulit untuk dipelihara, terlebih pada aplikasi yang kaya fitur dan berdiri diatas *backend* dengan arsitektur *microservices* [2]. Dalam beberapa tahun terakhir, arsitektur *microservices* mendapatkan perhatian dan menjadi salah satu objek penelitian kunci pada bidang ilmu informasi [2].

Pada perkembangan arsitektur *microservices* pada *backend*, memberikan

implementasi solusi yang baik. Dari sini kemudian muncul ide untuk mengkombinasikan pemisahan pada bagian *frontend* dan *backend*, dan membawa implementasi *microservices* pada bagian *frontend*, mengubah bentuk aplikasi web dari SPA menjadi kombinasi banyak aplikasi *frontend* kecil. Ide dibelakang *micro frontend* adalah untuk memperlakukan aplikasi web sebagai kombinasi fitur yang dimiliki oleh tim yang berbeda, dimana setiap tim mempunyai fungsi bisnis independen yang mereka fokuskan. Sebuah tim adalah lintas-fungsional dan mengembangkan fitur-fiturnya dari ujung ke ujung, dari *backend* ke *frontend* [3].

Aplikasi Data Diri Pelamar milik PT. XYZ adalah aplikasi web yang menggunakan arsitektur monolitik dan digunakan oleh para pelamar PT. XYZ untuk mengisi data diri yang diperlukan. Karena aplikasi sering mendapatkan perubahan dan penyesuaian seringkali terjadi malfungsi yang menyebabkan pengalaman pengguna dalam menggunakan aplikasi menurun, dan efektifitas pengembang dalam pemeliharaan menurun pula, selain itu, seiring dengan pertumbuhan dari aplikasi – aplikasi yang ada pada PT. XYZ melahirkan ketergantungan yang cukup kuat antar modul sehingga pada tahap pengembangan kedepannya direncanakan perombakan arsitektur yang monolitik tadi menjadi *microservices.* Beberapa faktor lain yang juga menjadi pertimbangan untuk menggunakan *microservices* antara lain, adalah *scope* proyek yang semakin besar sehingga proses pemeliharaan akan semakin susah untuk dilakukan dengan arsitektur monolitik yang digunakan. Selain itu, untuk memberikan kemudahan kepada pengembang, proses bisnis yang ada dibuat menjadi *single responbility* dimana proses bisnis yang ada dapat berjalan secara independen tanpa memikirkan layanan yang ada harus dikembangkan dengan kerangka kerja atau jenis basis data, memberikan kebebasan dan kedinamisan pengembangan per proses bisnis. Lalu, ketika ada proses bisnis baru yang harus dibuat, dengan *microservices*, maka tidak diperlukan *refractor* kode secara besar – besaran untuk pengaturan proyek yang sudah ada, cukup memanggil *api* dan layanan terkait yang diperlukan. Terakhir, jika tim pengembang atau tim lain memerlukan suatu data dari suatu proses bisnis, maka cukup dibuka layanan

untuk data terkait, tanpa memberikan semua layanan yang lain, sehingga pengelolaan data lebih aman.

Dari rencana ini kemudian muncul ide untuk mengimplementasikan *micro frontend* kebagian *frontend* Data Diri Pelamar (DPP). Penerapan ini diharapkan dapat memaksimalkan kinerja aplikasi dan kinerja *microservices* setelah diterapkan karena memiliki prinsip yang mirip. Penerapan juga diharapkan dapat membantu baik pelamar dalam menggunakan aplikasi agar dapat fokus menggunakan aplikasi tanpa gangguan saat adanya perubahan yang dilakukan oleh pengembang pada suatu bagian dan dapat membantu pengembang menjamin ketersedian aplikasi dan membuat pemeliharaan per tim menjadi lebih baik karena masing – masing tim mempunyai fokusnya tersendiri dimana jika malfungsi terjadi pada suatu bagian pada aplikasi tidak menyebabkan pengaruh pada bagian lain, sehingga bagian tersebut dapat tetap digunakan oleh pengguna, selagi pengembang fokus melakukan pemeliharaan pada bagian yang malfungsi tanpa mempengaruhi bagian lain yang masih berjalan.

1.2. Rumusan Masalah

Berdasarkan latar belakang, maka dapat disimpulkan rumusan masalah sebagai berikut:

1.2.1. Bagaimana mengubah arsitektur *frontend* pada aplikasi DPP menjadi *micro frontend*?

1.2.2. Bagaimana menjaga komunikasi setiap bagian – bagian kecil pada *frontend* sebagai suatu kesatuan?

1.3. Batasan Masalah

Agar penelitian ini dapat dikerjakan dengan maksimal, maka diperlukan batasan-batasan sebagai berikut:

1.3.1. Implementasi arsitektur micro *frontend* hanya diterapkan untuk aplikasi Data Diri Pelamar (DPP).

1.3.2. Implementasi memanfaatkan kerangka kerja vue.js dan single-spa.

1.3.3. Basis data, dan REST API, dan *backend* secara keseluruhan sudah

tersedia, perubahan yang dilakukan hanya pada bagian *frontend*.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah :

1.4.1. Mengimplementasikan arsitektur *micro frontend* pada aplikasi DPP.

1.4.2. Membuat aplikasi DPP menjadi lebih mudah untuk digunakan dan dipelihara.

1.5. Metode Penelitian

Metode penelitian yang digunakan dalam eksperimen ini adalah sebagai berikut :

1.5.1. Analisis Kebutuhan Sistem

1.5.2. Perancangan Sistem

1.5.3. Pembangunan Sistem

1.5.4. Pengujian Sistem

1.5.5. Penulisan Laporan

1.6. Sistematika Penulisan

Untuk lebih memahami laporan ini, maka materi-materi yang tertera pada laporan ini dikelompokkan menjadi beberapa sub bab dengan sistematika sebagai berikut:

1.6.1. Bab I Pendahuluan

Berisi latar belakang, rumusan masalah, batasan masalah, tujuan masalah, metode penelitian dan sistematika penulisan.

1.6.2. Bab II Tinjauan Pustaka

Berisi penelitian-penelitian terdahulu yang berkaitan dengan penelitian ini.

1.6.3. BAB III Landasan Teori

Berisi teori-teori yang berkaitan dengan penelitian ini.

1.6.4. BAB IV Analisis dan Perancangan Sistem

Berisi deskripsi masalah, analisis kebutuhan sistem, dan perancangan.

1.6.5. BAB V Implementasi dan Pengujian Sistem

Berisi deskripsi sistem, hasil sistem dan pembahasan sistem.

1.6.6. BAB VI Penutup

Berisi kesimpulan dan saran.