BAB I

PENDAHULUAN

1.1 Latar Belakang

Aplikasi Malldesa merupakan hasil dari pengembangan aplikasi yang telah ada sebelumnya pada desa Sidomulyo. Jauh sebelum aplikasi Malldesa tersebut ada, pemerintah desa Sidomulyo telah memanfaatkan media digital untuk mengupayakan masyarakatnya dalam melakukan permohonan surat tanpa harus mendatangi balai desa setempat yang disebut sebagai Sistem Pelayanan *Online*. Namun dalam sistem tersebut hanya berbasiskan form *online* saja, sehingga tercipta lah Aplikasi Malldesa sebagai hasil pengembangan dari sistem tersebut (Nusantara et al., 2022). Aplikasi Malldesa memiliki fitur sangat banyak dan kompleks. Secara garis besar aplikasi ini terdiri dari 3 fitur yang sangat menarik dan menjadi fitur utama dari aplikasi Malldesa. Aplikasi ini dapat digunakan untuk memudahkan masyarakat dalam hal pengajuan surat tanpa harus pergi ke balai desa. Selain fitur tersebut, pada Aplikasi Malldesa juga terdapat fitur jual beli *online* atau E-Commerce, sehingga dapat meningkatkan perekonomian warga. Kemudian fitur yang ketiga adalah aplikasi ini dilengkapi dengan fitur berita dan wisata yang dapat masyarakat ataupun wisatawan baca agar selalu mendapat informasi terbaru terkait berita dan wisata. Berdasarkan hal tersebut secara tidak langsung aplikasi Malldesa diharuskan untuk selalu aktif sedia dalam menangani *request* atau permintaan dari pengguna. Terlebih Aplikasi Malldesa mempunyai tiga fitur yang sangat kompleks, sehingga menjadi tidak dihindari apabila aplikasi akan menjadi lambat atau bahkan mati ketika aplikasi diakses oleh pengguna dengan *traffic* atau jumlah penggunaan yang banyak. Oleh sebab itu diperlukan sebuah cara agar aplikasi tersebut dapat menangani lalu lintas atau *traffic* yang dilakukan oleh pengguna ketika mengakses aplikasi.

Pembangunan aplikasi Malldesa masih mengadopsi arsitektur atau pendekatan yang bersifat *monolith*. Arsitektur *monolith* merupakan sebuah metode dalam pembangunan sistem aplikasi yang mana dalam rancangan

arsitektur ini akan menjalankan semua fungsi atau logika dalam satu kesatuan aplikasi (Daya et al., 2015). Hal tersebut tentu saja akan mengakibatkan kemampuan aplikasi dalam beradaptasi terhadap terjadinya perubahan kebutuhan sistem (*requirement changes*) semakin sulit. Pengelolaan kompleksitas kode dan *maintainability* akan mempengaruhi keseluruhan aplikasi. Begitu pula ketika akan mengubah atau menambahkan teknologi baru, harus melakukan *restarting* ulang aplikasi yang mana hal tersebut akan mengakibatkan aplikasi tidak akan bisa diakses untuk sesaat (Priyadarshini S & Shilpa G, 2017). Dalam hal ini penggunaan arsitektur *monolith* pada suatu saat tentu saja akan menjadi sebuah masalah ketika aplikasi tersebut mempunyai pengguna aktif yang sangat banyak, maka aplikasi tersebut tidak akan mampu dalam menangani *request* atau permintaan dari pengguna secara bersamaan dan akan mengakibatkan aplikasi tersebut mati atau *down* sehingga pengguna tidak dapat mengakses aplikasi Malldesa.

Kekurangan serta kelemahan arsitektur *monolith* yang saat ini digunakan dalam membangun aplikasi Malldesa dapat diatasi dengan menggunakan Arsitektur *microservice*. Arsitektur *microservices* merupakan pola pendekatan untuk mengembangkan aplikasi berbasis web dengan memecah keseluruhan fungsi perangkat lunak menjadi beberapa layanan kecil yang saling terhubung dan dapat dikembangkan dan di-*deploy* secara terpisah (Sendiang et al., 2018). *Microservice* dapat dibangun dengan menggunakan bahasa pemrograman yang berbeda dan berjalan pada mesin yang berbeda, sehingga memudahkan untuk mengembangkan dan memelihara aplikasi. Pendekatan ini memungkinkan tim pengembang untuk bekerja secara paralel pada bagian-bagian tertentu dari aplikasi tanpa mengganggu bagian lainnya. Arsitektur *Microservice* juga memungkinkan untuk peningkatan atau skalabilitas yang lebih baik dan kemampuan untuk meng-*upgrade* aplikasi secara bertahap tanpa *downtime* (Richardson, 2018). Pada penerapan-nya aplikasi yang menggunakan arsitektur *microservice* akan dipecah menjadi bagian-bagian yang terpisah dan lebih kecil, sesuai dengan *logic* atau fungsi-nya masing-masing. Sehingga setiap *service* dapat dikembangkan, diuji, dan dioperasikan secara *independent*, yang mana

hal ini dapat memudahkan dalam pengembangan dan pemeliharaan aplikasi. Berdasarkan hal tersebut maka penggunaan arsitektur *microservices* secara tidak langsung dapat menjawab masalah dari arsitektur *monolith*.

Pemecahan aplikasi menjadi lebih kecil sesuai dengan *service* dan fungsi-nya masing-masing akan menyebabkan komunikasi antar fungsi dalam aplikasi tidak dapat terjadi lagi. Oleh sebab itu, untuk melakukan pertukaran informasi dalam pendekatan *microservice* dibutuhkan suatu metode khusus. Terdapat beberapa cara yang dapat digunakan untuk melakukan interaksi pengambilan data. Pada penelitian ini komunikasi antar *service* menggunakan metode *Choreography internal communication*. Metode ini menekankan pada kesepakatan atau konsensus antar *microservice* terkait cara dan format yang digunakan dalam berkomunikasi, sehingga memudahkan dalam mengintegrasikan *microservice* satu dengan yang lain (Megargel et al., 2021). Metode komunikasi model ini melakukan pertukaran data antar *service* terjadi dengan cara *asynchronous* atau biasa disebut dengan *event-driven*. Sehingga pada penerapan-nya, seluruh data yang akan dikirimkan harus melewati *message broker* terlebih dahulu sebelum di konsumsi oleh masing-masing *service* yang membutuhkan (Petrasch, 2017). Penulis akan menggunakan *RabbitMQ* sebagai *message broker*. *RabbitMQ* menggunakan konsep *Advanced Messaging Queuing Protocol* atau AMQP sebagai konsep transportasi data.

Dalam pengembangan aplikasi menggunakan arsitektur *microservices*, mengharuskan tiap *service* tidak terikat satu sama lain atau *Loose Coupling* (Richardson, 2018). Sehingga pada implementasi *microservice* pada aplikasi Malldesa, penulis akan menggunakan teknologi *virtual machine* khusus, sebagai simulasi untuk memenuhi persyaratan dalam pengembangan aplikasi menggunakan arsitektur *microservice*. Penulis akan menggunakan teknologi *virtual machine* yang dimiliki oleh *Docker*. *Docker* adalah sebuah teknologi yang digunakan sebagai tempat untuk membungkus atau menempatkan aplikasi ke dalam sebuah kontainer (Jaramillo et al., 2016). Sehingga dapat dikatakan bahwa aplikasi akan terkontainerisasi atau terisolasi pada *environment* tersebut. Maka dengan menggunakan teknik ini penerapan arsitektur *microservice* dapat

dilakukan tanpa harus mengeluarkan *cost* atau biaya yang sangat banyak karena harus menggunakan perangkat fisik. Oleh sebab itu penulis berharap implementasi arsitektur *microservice* pada aplikasi Malldesa menggunakan metode *choreography internal communication* dapat berjalan dengan lancar dan mengeluarkan hasil yang sesuai dengan harapan.

1.2 Perumusan Masalah

Berdasarkan pada uraian latar belakang di atas, maka perumusan masalah yang didapat adalah:

1. Bagaimana desain arsitektur *microservice* pada aplikasi Malldesa menggunakan metode *choreography internal communication*.
2. Bagaimana kinerja aplikasi Malldesa dengan Arsitektur *Microservices* terhadap kemampuan dalam menangani beban *traffic* pengguna

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian yang berjudul Implementasi Arsitektur *Microservice* pada aplikasi Malldesa menggunakan metode *Choreography Internal Communication* adalah:

1. Untuk Merancang dan menghasilkan desain sistem arsitektur *microservice* yang terimplementasikan pada sistem aplikasi Malldesa menggunakan metode Choreography Internal Communication.
2. Untuk mengetahui kinerja aplikasi Malldesa terhadap kemampuan dalam menangani beban *traffic* pengguna

1.4 Manfaat Penelitian

Manfaat yang diharapkan pada penelitian ini adalah:

1. Manfaat dari dilakukannya penelitian ini adalah diharapkan dapat membantu mahasiswa, tenaga kerja, dan masyarakat umum dalam menerapkan arsitektur *Microservices*.
2. Sistem pada aplikasi Malldesa tidak lagi terikat satu sama lain sehingga penambahan atau perubahan dapat dilakukan secara leluasa serta memberikan kemudahan dalam pengembangan selanjutnya.
3. Penelitian dapat memberikan kontribusi sebagai artikel mengenai Arsitektur *Microservice*.

1.5 Batasan Masalah

Adapun batasan masalah pada penelitian ini adalah:

1. Implementasi arsitektur *microservice* hanya dilakukan pada sistem *backend* aplikasi Malldesa.
2. Penelitian tidak mengubah struktur sistem pada aplikasi *mobile* melainkan hanya mengubah *endpoint api*.
3. Menggunakan salah satu *Message Broker* yaitu *RabbitMQ*
4. Menerapkan kontainerisasi *virtual* dengan menggunakan *Docker*
5. Bahasa pemrograman yang digunakan tetap menggunakan *PHP* dan *Framework Laravel*
6. Penelitian tidak mengubah atau mendesain ulang *database*