BAB I
PENDAHULUAN

1. 1 Latar Belakang

Dewasa ini, manajemen organisasi dihadapkan dengan tantangan globalisasi yang mendorong perkembangan pasar menjadi sangat kompetitif hingga berdampak pada persaingan bisnis yang semakin kompleks (Widajanti, 2008). Globalisasi serta kemajuan ilmu pengetahuan dan teknologi telah mengubah perilaku manusia sebagai upaya memenuhi berbagai kebutuhannya (Poerwanto et al., 2013). Oleh karena itu, keberhasilan suatu organisasi saat ini tergantung pada tingkat fokus manajemen dalam mengkoordinasikan aset infrastruktur yang mendorong organisasi siap untuk berubah menghadapi tantangan yang ada (Hermawan & Suharnomo, 2020). Organisasi perlu melakukan adaptasi dengan dinamika lingkungan melalui perubahan yang sesuai dengan kekuatan dan kebutuhan organisasi, salah satunya dengan menerapkan teknologi informasi dalam fungsi bisnis dan manajerialnya (Poerwanto et al., 2013). Peran teknologi informasi terhadap suatu organisasi cukup besar karena merupakan aset yang mudah ditiru dan diterapkan untuk menciptakan keunggulan kompetitif (Mithas & Krishnan, 2008).

Salah satu organisasi yang berkewajiban menyelenggarakan pendidikan, penelitian dan pengabdian kepada masyarakat serta memiliki otonomi untuk pengelolaan bidang akademik dan non akademik yaitu Perguruan Tinggi (Undang-Undang Nomor 12 Tahun 2012 Tentang Perguruan Tinggi, 2012). Perguruan tinggi termasuk salah satu organisasi akademis yang dapat menggunakan teknologi informasi untuk mendukung berbagai proses bisnis di dalamnya (Joko, 2010). Pada era digital sekarang ini, pemanfaatan teknologi informasi juga merupakan suatu tuntutan dan merupakan strategi yang sangat tepat untuk menciptakan keunggulan bersaing dan meningkatkan kinerja perguruan tinggi (Samsiah et al., 2018). Hal ini juga sejalan dengan salah satu ciri perguruan tinggi modern, yaitu manajemen yang berbasis teknologi informasi dengan mengimplementasikan infrastruktur *ICT* (*Information and Communication Technology*) di dalamnya (Prasojo, 2009). Salah

satu bentuk pemanfaatan *ICT* dalam pendidikan tinggi adalah Sistem Informasi Manajemen yang dapat mendukung proses-proses manajemen, pengambilan keputusan, serta pengolahan data dan informasi (Husaini & IAIN, 2014).

Sebagai salah satu perguruan tinggi negeri di Indonesia, Universitas Pendidikan Indonesia (UPI) telah memanfaatkan teknologi informasi dalam berbagai proses bisnis di dalamnya, seperti penggunaan sistem layanan akademik, sistem layanan surat, *Learning Management System (LMS)*, dan sistem informasi untuk kegiatan *monitoring* dan manajemen pembelajaran. Hal ini juga merupakan bentuk implementasi dari salah satu sasaran strategis UPI tahun 2021-2025, yaitu "(3) modernisasi sistem manajemen bagi penyediaan layanan pendidikan yang handal, efisien, dan mudah ditransformasikan" (Rencana Strategis (RENSTRA) Universitas Pendidikan Indonesia Tahun 2021-2025, 2021). Oleh karena itu, sistem teknologi informasi yang ada di UPI harus dan akan selalu dikembangkan agar menjadi perguruan tinggi yang adaptif serta tidak kehilangan relevansinya dan ditinggalkan oleh mahasiswa (Nizam, 2021). Pemanfaatan dan pengembangan teknologi informasi di UPI juga bertujuan agar institusi mampu beradaptasi terhadap kemajuan yang ada, serta dapat beroperasi secara efektif dan efisien (Supratman et al., 2019).

Salah satu sistem informasi di Universitas Pendidikan Indonesia yang juga dikembangkan untuk mendukung program Kemendikbudristek (Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi) yaitu sistem Tracer Study (UPI, 2021). Tracer Study merupakan suatu sistem *online* yang dapat digunakan oleh Perguruan Tinggi untuk melacak aktivitas para lulusannya setelah masa pendidikan tinggi, baik masa transisi maupun pergerakan lulusan di dunia kerja (Direktorat Jenderal Pendidikan Tinggi, 2011). Sistem Tracer Study ini dapat digunakan untuk mengawasi alumni dan dilakukan secara berkala oleh perguruan tinggi (Akbar & Mukhtar, 2020). Tracer Study mengumpulkan data dari kuesioner PKTS (Pengembangan Karir dan Tracer Study) yang mencakup informasi mengenai waktu tunggu alumni mendapat pekerjaan, persentase alumni yang telah bekerja, dan beberapa pertanyaan terkait informasi dalam peningkatan kurikulum yang ada di perguruan tinggi (Noviyantono & Aidil, 2012). Perguruan tinggi, termasuk UPI, melaksanakan survei Tracer Study secara reguler setiap tahun guna memenuhi

kebutuhan data akreditasi. Data Tracer Study juga disosialisasikan dan digunakan untuk pengembangan kurikulum dan perbaikan pembelajaran di perguruan tinggi (Syafiq, 2017).

Sistem Tracer Study UPI dikembangkan oleh Direktorat Sistem dan Teknologi Informasi (DSTI). Sistem Tracer Study UPI memiliki 4 buah layanan, yaitu layanan *Authentication,* layanan Tracer Study untuk memantau data lulusan/alumni, layanan User Study yang ditujukan untuk pengguna lulusan, dan layanan publikasi berita, pengumuman, dan/atau artikel dari Badan Bimbingan dan Konseling dan Pengembangan Karir (BKPK) UPI. Sistem ini dibangun dengan menerapkan arsitektur *monolithic* dimana seluruh layanan, baik *backend* maupun *frontend* tergabung menjadi satu kesatuan dan berjalan sebagai proses tunggal di lingkungan *server* (Koschel et al., 2017).

Kelebihan dari arsitektur *monolithic* seperti yang digunakan untuk membangun sistem Tracer Study UPI terletak pada kesederhanaannya, karena seluruh bagian tergabung dalam satu sistem yang sama, sehingga membuatnya mudah diimplementasikan dan di-*deploy* (Koschel et al., 2017). Namun, seiring bertambahnya ukuran dan kompleksitas aplikasi, arsitektur ini memiliki banyak kekurangan, khususnya dari sisi *scalability, maintainability, time to market,* dan *performance* (Blinowski et al., 2022; Fritzsch et al., 2019). Selain itu, arsitektur *monolithic* juga memungkinkan kegagalan sebagian sistem dapat mempengaruhi keseluruhan sistem, sehingga proses *maintenance* menjadi lebih sulit (S. Li et al., 2019; Newman, 2020). Dalam satu sistem yang kompleks (Mendonca et al., 2021), arsitektur *monolithic* memusatkan seluruh fungsionalitas dalam satu komponen individu yang besar, yang menimbulkan keterbatasan dalam *scalability, maintenance,* dan performa (Ahmadvand & Ibrahim, 2017; Auer et al., 2021; Balalaie et al., 2018; Clarke et al., 2016; Taibi et al., 2017).

Sebagai solusi atas permasalahan pada arsitektur *monolithic,* hadirlah arsitektur *microservices* (Tapia et al., 2020). Arsitektur ini memecah domain bisnis menjadi konteks kecil yang mengimplementasikan konsep *autonomous, self-contained, loosely coupled,* dan independen yang memungkinkan untuk digunakan sesuai dengan kebutuhan (Lewis & Fowler, 2014; Posta, 2016; Rajesh RV, 2016). *Microservices* pertama kali diperkenalkan dalam Lewis & Fowler (2014) pada *blog*

mereka, di mana *microservices* didefinisikan sebagai "*an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API*". Dalam beberapa penelitian, arsitektur *microservices* telah dibuktikan sangat baik dari sisi *scalability, maintainability, availability, reusability* (Blinowski et al., 2022; Chen et al., 2017; Newman, 2021) dan performa dibandingkan dengan arsitektur *monolithic* (Balalaie et al., 2018; Kalske et al., 2018; Wolfart et al., 2021).

Selain dari segi skalabilitas dan performa, arsitektur *microservices* memiliki banyak kelebihan lain dibandingkan dengan arsitektur *monolithic* (Clarke et al., 2016; Di Francesco et al., 2018; Dragoni et al., 2017). Arsitektur ini sangat mendukung heterogenitas teknologi yang digunakan dalam membangun setiap *service,* dimana dalam satu sistem dapat menggunakan teknologi maupun bahasa yang berbeda-beda sesuai dengan kebutuhan (Auer et al., 2021; Newman, 2021). *Microservices* juga memiliki tingkat resiliensi yang tinggi yang memungkinkan kegagalan pada satu bagian tidak akan mempengaruhi bagian lain (Newman, 2021; Phatak, 2022; Suryotrisongko, 2017). Berikutnya, arsitektur ini juga dapat membantu organisasi menyesuaikan strukturalnya dengan kebutuhan pengembangan, membagi sumber daya pengembang dengan lebih merata dan meminimalisir pengembang bekerja pada satu *codebase* yang sama (Auer et al., 2021; Newman, 2021; Phatak, 2022). Arsitektur *microservices* juga mendukung *composability* dan optimalisasi untuk *replaceability* (Newman, 2021). Pada arsitektur *microservices,* setiap layanan akan saling terhubung dengan menggunakan *API Gateway*, yaitu *API* yang menangani seluruh akses dari luar dengan *microservices* internal (Auer et al., 2021; Newman, 2021).

Untuk membangun arsitektur *microservices,* dalam Newman (2021) disebutkan bahwa terdapat beberapa jenis *API* yang dapat digunakan, salah satunya adalah menggunakan *RPC API*. *RPC* atau *Remote Procedure Call* merupakan teknik menggunakan pemanggilan prosedur (*method*) lokal dan menjalankannya pada *remote service* (layanan jarak jauh) (Newman, 2021). Sebuah penelitian (Sivathanu et al., 2002) menyebutkan bahwa *RPC* sangat mudah untuk digunakan, memberi pembaharuan terhadap paradigma pemanggilan prosedur *client* ke *server*, serta cukup kuat untuk dapat menunjang banyak layanan sistem terdistribusi. Salah

satu *RPC* yang banyak digunakan saat ini adalah *gRPC*, yaitu *framework RPC* yang dikembangkan oleh Google dan dirilis sebagai *open-source* pada tahun 2015 (Google, 2015a). gRPC dibuat untuk menghubungkan ribuan *micro services* yang telah distandarisasi, dapat berjalan di *multi-platform* dan di beberapa *data center* serta dibangun dengan teknologi berbeda (Indrasiri & Kuruppu, 2020). gRPC berkomunikasi menggunakan *protocol buffer* (*protobuf*), yang merupakan mekanisme *open-source* Google untuk serialisasi data terstruktur (Google, 2015b).

Dengan tingkat resiliensi *microservices* yang tinggi (Newman, 2021), arsitektur ini akan cocok apabila diterapkan pada sistem Tracer Study UPI yang memiliki beberapa layanan dalam satu aplikasinya, serta memungkinkan jika diterapkan pada aplikasi lain di UPI yang memiliki banyak aplikasi dan layanan yang redundan. Arsitektur *microservices* dapat mengisolasi kegagalan pada satu layanan agar tidak mempengaruhi keseluruhan sistem sehingga pengembangan pada setiap layanan tidak saling bergantung (Auer et al., 2021; Dragoni et al., 2017). Arsitektur ini juga mendukung skalabilitas yang tinggi, sehingga sangat memudahkan jika terdapat layanan baru yang perlu dibangun (Newman, 2020). Hal ini sangat mendukung Universitas Pendidikan Indonesia sebagai lembaga pendidikan tinggi yang dituntut untuk menjadi adaptif dengan keadaan, dengan melakukan modernisasi pada sistem informasinya (Nizam, 2021). Selain itu, penerapan arsitektur *microservices* juga dapat memudahkan tim pengembang di UPI jika suatu saat perlu memperbarui teknologi yang dipakai dengan teknologi baru yang lebih *powerful* (Dragoni et al., 2017; Taibi et al., 2017).

Untuk menerapkan arsitektur *microservices* pada sistem *monolithic,* perlu dilakukan dekomposisi pada sistem *monolithic* tersebut (Balalaie et al., 2018; Taibi et al., 2017). Salah satu metode dekomposisi sistem *monolithic* menjadi *microservices* yaitu pendekatan *Domain-Driven Design (DDD)* (Newman, 2020; Richardson, 2018). *DDD* pertama kali diperkenalkan dalam Evans (2003) sebagai metodologi pengembangan perangkat lunak yang berfokus pada pemodelan sistem yang kompleks melalui perbaikan berulang dan desain berkelanjutan, dengan mengutamakan pengetahuan domain *(domain knowledge)* (Evans, 2003). Metode ini menguraikan sistem kompleks menjadi bagian-bagian yang lebih kecil dan lebih mudah dikelola *(subdomain),* serta menyelaraskan desain sistem perangkat lunak

dengan kebutuhan bisnis di dunia nyata (Evans, 2003). Sistem *microservices* yang dimodelkan dengan menggunakan metode *DDD* dapat memiliki batasan antar-*service* yang jelas, *low-coupled, cohesive,* serta mudah dipelihara dan dikembangkan (Evans, 2003; Newman, 2020; Vural & Koyuncu, 2021).

Dalam penelitian awal yang dilakukan melalui observasi dan wawancara, Universitas Pendidikan Indonesia memiliki ratusan aplikasi yang seluruhnya masih menerapkan arsitektur *monolithic.* Banyak dari aplikasi-aplikasi tersebut yang memiliki karakteristik, layanan, dan fitur yang serupa namun dibangun menjadi aplikasi yang berbeda. Hal tersebut menyebabkan terjadinya fenomena redundansi aplikasi, sulitnya integrasi antar-aplikasi, ketidakefisienan sumber daya, desentralisasi manajemen, hingga masalah inkonsistensi data. Masalah tersebut juga terjadi pada aplikasi Tracer Study UPI. Keterkaitan yang erat antarkomponen *(high coupled)* pada aplikasi *monolithic* Tracer Study UPI seringkali menyebabkan kegagalan pada seluruh sistem meskipun hanya salah satu layanan yang bermasalah. Ini menunjukkan tingkat resiliensi dan *fault tolerant* yang rendah pada sistem tersebut. Aplikasi *monolithic* Tracer Study UPI juga memiliki kendala terkait *incompatible dependencies* dan *technology flexibility* karena merupakan layanan tunggal yang dikembangkan dalam *environment* bersama (tidak terisolasi dari aplikasi lain). Masalah lain seperti *scalability*, keterbatasan proses *deployment*, serta penggunaan sumber daya yang tinggi juga menjadi beberapa kekurangan yang dapat menurunkan kualitas aplikasi.

Oleh karena itu, berdasarkan studi lapangan dan studi literatur yang telah dilakukan, penelitian ini berfokus pada Rancang Bangun Arsitektur *Microservices* Menggunakan *Google Remote Procedure Call (gRPC) Application Programming Interface (API)* dan Metode *Domain-Driven Design* dengan menggunakan Aplikasi Tracer Study Universitas Pendidikan Indonesia sebagai studi kasus dan percontohan. Penelitian ini mendekomposisi Sistem *Monolithic* pada Tracer Study UPI yang sudah ada saat ini dengan menggunakan metode dekomposisi *Domain-Driven Design* dan memigrasinya menjadi arsitektur *microservices.* Penelitian ini diharapkan dapat memberikan berbagai manfaat dari kelebihan arsitektur *microservices* serta meningkatkan kualitas dan performa sistem teknologi informasi di Universitas Pendidikan Indonesia. Dengan diterapkannya arsitektur

microservices pada sistem Tracer Study UPI, diharapkan berbagai sistem lain di Universitas Pendidikan Indonesia juga dapat mengadopsi arsitektur ini dan menjadi lebih adaptif untuk pengembangan sistem jangka panjang. Selain itu, implementasi microservices dalam bentuk *API* untuk setiap *service*-nya juga dapat membuat proses sinkronisasi data dan integrasi antar-service menjadi lebih mudah, sehingga dapat meningkatkan efisiensi dan konsistensi data. Dengan demikian, aplikasi Tracer Study UPI dapat menjadi lebih tangguh, efisien, serta mudah dipelihara dan dikembangkan di masa mendatang.

1. 2 Rumusan Masalah

Masalah yang dirumuskan dalam penelitian ini berdasarkan latar belakang di atas yaitu sebagai berikut.

1. Bagaimana mendekomposisi dan merancang ulang sistem *monolithic* Tracer Study UPI yang sudah ada saat ini menjadi *microservices*?
2. Bagaimana membangun dan mengimplementasikan Arsitektur *Microservices* untuk Sistem Tracer Study UPI?
3. Bagaimana pengaruh penerapan arsitektur *microservices* dari sisi kinerja pada aplikasi Tracer Study UPI?

1. 3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah didapatkan, tujuan dari dilakukannya penelitian ini diantaranya yaitu sebagai berikut.

1. Mendekomposisi dan merancang ulang Sistem Tracer Study UPI yang sudah ada dengan menggunakan pendekatan *Domain-Driven Design* sebagai metode perancangan *microservices*
2. Membangun dan mengimplementasikan Arsitektur *Microservices* untuk Sistem Tracer Study UPI dengan menggunakan *gRPC API*
3. Mengevaluasi pengaruh dari penerapan arsitektur *microservices* dari sisi kinerja pada aplikasi Tracer Study UPI

1. 4 Manfaat Penelitian

Beberapa manfaat dari penelitian ini adalah sebagai berikut:

1. Manfaat bagi Penulis

Penelitian ini memberikan pengalaman, pengetahuan dan keterampilan bagi penulis dalam melakukan penelitian di bidang perancangan arsitektur *microservices,* dekomposisi sistem *monolithic,* dan pengujian perangkat lunak. Penelitian ini juga dapat menjadi portofolio bagi penulis di masa mendatang.

2. Manfaat bagi Pembaca

Hasil dari penelitian ini dapat memberikan gambaran bagi pembaca mengenai proses migrasi dari sistem yang dibangun dengan arsitektur *monolithic* menjadi sistem yang mengimplementasikan arsitektur *microservices.* Pembaca juga dapat mengetahui cara mendekomposisi, merancang ulang, membangun, dan mengimplementasikan arsitektur *microservices* pada suatu sistem di suatu organisasi. Hasil analisis dari perbandingan pada pengujian kedua arsitektur sistem tersebut juga dapat menjadi acuan bagi pembaca untuk penelitian lebih lanjut.

3. Manfaat bagi Pengembang Perangkat Lunak

Pengembang dapat melihat hasil pengujian dan analisis dari arsitektur *microservices* yang telah dirancang sehingga dapat menilai potensi dan kemungkinan penerapan arsitektur ini pada sistem lain dalam organisasi. Penggunaan metode pendekatan *Domain-Driven Design* dalam perancangan aplikasi menjadi *microservices* juga dapat menyederhanakan proses penetapan kandidat *services*. Oleh karena itu, penelitian ini dapat memberikan panduan praktis bagi pengembang perangkat lunak yang ingin memigrasi arsitektur aplikasi mereka dari *monolithic* ke *microservices*.

Aplikasi Tracer Study yang telah mengimplementasikan arsitektur *microservices* juga dapat memudahkan pengembang dalam organisasi UPI untuk memelihara dan mengembangkan fitur baru di masa yang akan datang. Arsitektur ini juga memberikan kebebasan bagi pengembang di UPI untuk memilih teknologi atau bahasa yang diinginkan untuk membangun suatu layanan mikro baru tanpa mengikuti teknologi yang digunakan pada layanan mikro lain. Dengan menerapkan arsitektur ini, pengembang juga akan dimudahkan dengan kebebasan penggunaan

dependensi pada suatu sistem yang dikembangkan karena tidak akan mengganggu sistem yang lain.

Selain itu, dengan mendekomposisi sistem *monolith* menjadi *microservices,* pengembang dapat menerapkan *services* tersebut pada *client* yang berbeda-beda. *Service* yang sama dapat diterapkan pada beberapa *client* seperti *web, mobile,* hingga *desktop.* Hal ini dapat memudahkan pengembang jika terdapat kebutuhan tertentu yang mengharuskan implementasi sistem pada *client* yang berbeda karena pengembang tidak perlu melakukan perubahan apapun pada *service* yang ingin digunakan serta tidak perlu membangun *service* yang baru.

4. Manfaat bagi Lembaga Universitas Pendidikan Indonesia

Dengan adanya penelitian ini diharapkan dapat meningkatkan kualitas sistem teknologi informasi di Universitas Pendidikan Indonesia. Selain itu, aplikasi Tracer Study UPI merupakan aplikasi penting untuk mengumpulkan data alumni dan mengukur kualitas pendidikan yang diberikan oleh universitas. Dengan memperbaiki arsitektur dan teknologi yang digunakan dalam aplikasi ini, akan meningkatkan kualitas data yang dikumpulkan dan efisiensi pengelolaannya.

Dengan diterapkannya arsitektur *microservices* pada sistem Tracer Study UPI, diharapkan juga sistem-sistem lain di Universitas Pendidikan Indonesia dapat mengadopsi arsitektur ini dan menjadi lebih adaptif untuk pengembangan sistem jangka panjang. Penelitian ini dapat menjadi percontohan bagi sistem dan aplikasi UPI yang lain saat ingin memigrasi dan membangun *microservices.* UPI juga dapat menjadi pelopor bagi universitas lain yang mungkin belum mengimplementasikan arsitektur ini pada sistem-sistem di dalamnya.

5. Manfaat bagi Peneliti Lain

Penelitian ini mengangkat kasus konkret penerapan arsitektur *microservices* menggunakan *gRPC API* pada aplikasi Tracer Study UPI yang dapat dijadikan referensi dan dasar bagi peneliti lain di bidang arsitektur sistem dan pengembangan aplikasi. Peneliti lain dapat memperluas dan memperdalam penelitian ini dengan mengeksplorasi aspek lain dari *microservices, gRPC*, dan pola dekomposisi yang digunakan. Migrasi dari sistem *monolithic* ke *microservices* dalam penelitian ini yang menggunakan metode pendekatan *Domain-Driven Design* juga dapat

memberikan panduan bagi peneliti yang menghadapi tantangan yang sama pada penelitian yang dilakukan.

Penelitian ini dapat menyingkap berbagai masalah lain pada studi kasus yang sama yang layak untuk diteliti secara lebih lanjut dan menjadi topik penelitian baru. Hal ini dapat memunculkan berbagai solusi baru dan menyelesaikan masalah yang ada secara bertahap.

1. 5 Batasan Masalah

Berdasarkan latar belakang penelitian yang telah dipaparkan, terdapat beberapa batasan masalah yang menjaga agar penelitian yang dilakukan tetap fokus pada rumusan masalah. Adapun batasan masalah pada penelitian ini yaitu:

1. Penelitian ini berfokus pada salah satu aplikasi yang ada di Universitas Pendidikan Indonesia, yaitu aplikasi Tracer Study UPI yang dikembangkan oleh Direktorat Sistem dan Teknologi Informasi (DSTI).
2. Ruang lingkup dan data penelitian yang digunakan terbatas pada aplikasi Tracer Study UPI dan DSTI UPI.
3. Arsitektur *microservices* yang dikembangkan pada penelitian ini menggunakan 2 (dua) jenis *API*, yaitu *gRPC API* untuk pengembangan *service* dan *REST API* untuk pengembangan *API Gateway*.
4. Penelitian ini berfokus pada pengembangan arsitektur *microservices* dan tidak membahas optimasi dari sisi *server, database,* maupun *user interface.*
5. Penelitian ini menerapkan keamanan *API* standar, sehingga tidak menganalisis serta merancang keamanan lanjutan pada arsitektur yang dirancang.
6. Metode pengujian aplikasi yang digunakan pada penelitian ini adalah *performance testing*, salah satu jenis dari *black-box* dan *non-functional testing.*
7. Karena salah satu fitur pada sistem *microservices* yang dikembangkan menggunakan layanan *API SIAK* milik UPI, koneksi langsung pada *API SIAK* tersebut digantikan dengan metode *mock API* saat pengujian kinerja berlangsung untuk menghindari terjadinya kegagalan pada layanan *API SIAK* akibat *traffic/load* pengujian yang tinggi.

8. Metode perancangan arsitektur *microservices* yang digunakan pada aplikasi Tracer Study adalah pendekatan *Domain-Driven Design*, sehingga penelitian ini tidak membahas dan menganalisis metode perancangan lain.

1. 6 Sistematika Penulisan

Adapun sistematika penulisan pada penelitian ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan mengenai latar belakang penelitian, rumusan masalah, tujuan penelitian, serta batasan masalah penelitian terkait topik Rancang Bangun Arsitektur *Microservices* Menggunakan *Google Remote Procedure Call (gRPC) Application Programming Interface (API)* dan Metode *Domain-Driven Design*. Dijelaskan pula mengenai pentingnya teknologi informasi dalam sebuah organisasi, penerapan teknologi informasi di Universitas Pendidikan Indonesia, pengertian dari arsitektur *monolithic* dan *microservices,* pengenalan *gRPC API,* serta karakteristik dan kendala pada aplikasi Tracer Study UPI sebagai studi kasus penelitian.

BAB II KAJIAN PUSTAKA

Bab ini berisi penjelasan dan penjabaran mengenai teori-teori, istilah penting, dan metode yang berkaitan dengan topik penelitian, diantaranya yaitu mengenai aplikasi Tracer Study, arsitektur *monolithic, microservices,* metode dekomposisi dan perancangan *Domain-Driven Design, API*, *gRPC API*, *Protocol Buffer*, *API Gateway*, metode dan teknik dekomposisi, serta *integration* dan *performance testing.*

BAB III METODOLOGI PENELITIAN

Bab ini berisikan penjelasan secara rinci mengenai desain penelitian, jenis penelitian, objek penelitian, teknik pengumpulan dan pengambilan data, populasi dan sampel penelitian, sumber data, analisis perhitungan data, dan langkah-langkah yang perlu dilakukan selama penelitian.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menguraikan dan menganalisis hasil dari pengumpulan data yang telah dilakukan. Bab ini juga membahas cara mendekomposisi dan merancang ulang Sistem Tracer Study UPI yang sudah ada saat ini dengan menggunakan salah satu metode dekomposisi sistem *monolithic,* yaitu *Domain-Driven Design.*

Kemudian dirancang bagaimana pengimplementasian arsitektur *microservices* dari hasil dekomposisi sistem *monolithic* tersebut. Setelah itu, dilakukan pengujian terhadap kedua sistem, yaitu sistem *monolithic* sebelumnya dan *microservices* yang baru untuk kemudian dilakukan analisis terhadap perbandingan hasil pengujian *performance testing* antara keduanya.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan dan korelasinya dengan rumusan masalah dan tujuan penelitian pada Bab I Pendahuluan. Pada Bab ini juga diikutsertakan saran yang relevan terkait dengan penelitian yang dilakukan. Saran ini bertujuan untuk meningkatkan kualitas penelitian dan memberikan gambaran bagi penelitian yang akan datang.