Bidang Ungulan : Ketahanan Pangan, Informasi dan Teknologi
Kode Topik Penelitian : D.15.1
Kode Rumpun Ilmu : 451

USULAN
PENELITIAN INVENSI UDAYANA

JUDUL PENELITIAN

TRANSFORMASI ARSITEKTUR MONOLITHIC MENUJU MICROSERVICES UNTUK IMPLEMENTASI SISTEM INFORMASI TERINTEGRASI

TIM PENELITI

Dr. Dewa Made Wiharta, ST, M.T. (NIDN: 0022097003)
Dr. Nyoman Putra Sastra, S.T., M.T. (NIDN: 0029087205)
Komang Oka Saputra, S.T., M.T., Ph.D. (NIDN : 0004048106)

PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS UDAYANA
DESEMBER 2019

HALAMAN PENGESAHAN PROPOSAL
PENELITIAN INVENSI UDAYANA

Judul	: Transformasi Arsitektur Monolithic Menuju Microservices untuk Implementasi Sistem Informasi Terintegrasi
Peneliti / Pelaksana	
Nama lengkap	: Dr. Dewa Made Wiharta, ST, MT
NIP/NIDN	: 197009221997021001 / 0022097003
Jabatan Fungsional/Stuktural	: Lektor / Sekretaris Lembaga pada Unit Sumber Daya & Informasi
Program Studi	: Sarjana Teknik Elektro
Nomor HP	: 081703440558
Alamat Surel (e-mail)	: wiharta@unud.ac.id
Anggota 1	
Nama Lengkap	: Dr. Nyoman Putra Sastra, ST., MT.
NIDN	: 0029087205
Perguruan Tinggi	: Sarjana Teknik Elektro
Anggota 2	
Nama Lengkap	: Komang Oka Saputra, S.T., M.T., Ph.D.
NIDN	: 0004048106
Perguruan Tinggi	: Sarjana Teknik Elektro
Institusi Mitra (jika ada)	
Nama Institusi Mitra	:
Alamat	:
Penanggung Jawab	:
Tahun Pelaksanaan	: Tahun ke-2 dari rencana 2 tahun
Biaya Diusulkan	: Rp. 110.000.000

Mengetahui,
Dekan Fakultas Teknik

(Prof. Ir. I Gede Suardana, MT, Ph.D.)
NIP. 196409171989031002

Denpasar, 05 Desember 2019
Ketua Tim Pelaksana

(Dr. Dewa Made Wiharta, ST, MT)
NIP: 197009221997021001

Menyetujui,
Ketua Lembaga Penelitian dan Pengabdian kepada Masyarakat
Universitas Udayana

(Prof. Dr. Ir. I Gede Rai Maya Temaja, MP.)
NIP. 196210091988031002

RINGKASAN

Pada arsitektur monolithic (konvesional), aplikasi dikembangkan dalan satu entitas besar. Walaupun mudah dalam pengembangan awal, model ini menimbulkan masalah ketika codebase berkembang. Akibatnya adalah pengembangan menjadi lebih lambat, resource tidak dapat digunakan secara optimal, terutama karena terbatasnya opsi untuk scaling aplikasi.

Microservices, sebagai teknologi terbaru dalam perancangan dan pengembangan software, melakukan pendekatan berupa arsitektur yang dibangun dalam konsep modularisasi, dengan penekanan pada batasan yang bersifat teknis. Tiap modul, disebut dengan microservice, diimplementasikan sebagai sistem kecil yang independent.

Penelitian ini ditujukan untuk melakukan analisa terhadap tantangan yang diberikan dalam pengembangan microservice, menelusuri fitur utama dalam microservice, dan bagaimana fitur tersebut bisa meningkatkan skalabilitas sistem. Selanjutnya, akan dikembangkan suatu model yang nantinya dapat digunakan sebagai acuan dalam melakukan transformasi dari arsitektur monolithic menuju microservice. Model ini akan diujikan pada sistem informasi yang telah ada di Universitas Udayana, sebagai salah satu bagian dari sistem informasi terintegrasi, yaitu Sistem Informasi Dosen (SIMDOS), dengan alamat https://simdos.unud.ac.id. Secara umum sistem ini memiliki beberapa fungsi antara lain untuk melakukan manajemen data dosen, manajemen data riwayat, pelaporan beban kerja dosen dan remunerasi dosen. Problem yang terjadi saat ini adalah susahnya untuk melakukan optimasi secara horizontal, sebagai contoh, ketika sistem sedang melakukan proses perhitungan yang memerlukan beban komputasi tinggi sering menyebabkan sistem tidak dapat diakses sampai proses komputasi selesai. Selain itu aplikasi SIMDOS juga merupakan salah satu aplikasi utama di Universitas Udayana, yaitu hampir semua aplikasi meminta data dosen ke SIMDOS, sehingga jika aplikasi SIMDOS down maka akan banyak sistem lain tidak dapat berjalan/digunakan

Kata kunci : Arsitektur Monolithic, microservices, sistem informasi

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Arsitektur microservice adalah style arsitektur terbaru yang berkembang pada beberapa tahun terakhir. Tidak ada definisi pasti mengenai arsitektur microservice namun dapat dikatakan microservice adalah koleksi dari beberapa service kecil yang otonom. Tiap service memiliki tugas masing-masing yang memenuhi *single responsibility principle* (Kalske dkk, 2017). Pada arsitektur monolithic, semuanya dikembangkan dalam satu entitas besar. Ini membuat pengembangan awal mudah dan cepat dimengerti. Namum ketika codebase sebuah sistem berkembang, maka mulai muncul masalah pada arsitektur monolithic. Masalah utama adalah semakin besar *codebase* membuat pengembangan menjadi lebih lambat, kesulitan untuk melanjutkan pengembangan dan terbatasnya opsi untuk *scaling* aplikasi.

Microservice memiliki beberapa kelebihan dibandingkan monolithic. Microservice membuat batasan, yaitu kebutuhan bisnisnya dapat terlihat bedasarkan fungsionalitasnya yang ada pada *codebase*. Hal ini sangat penting karena pada codebase monolithic yang besar sulit untuk menentukan lokasi fungsi tertentu berada. Microservice memungkinkan untuk menggunakan teknologi berbeda sebagai *service* yang berbeda yang dapat diimplementasikan dengan teknologi terkini. Ini memungkinkan adopsi teknologi baru bisa lebih cepat dengan *microservice*. *Microservice* juga memungkinkan pengembangan secara kontinu (Cerny dkk., 2018).

Sistem Informasi Dosen (SIMDOS) adalah salah satu aplikasi yang ada pada Universitas Udayana. Aplikasi ini digunakan oleh dosen dan pegawai di Universitas Udayana. Pada sistem SIMDOS ini terdapat beberapa fungsi untuk melakukan manajemen pegawai, manajemen data riwayat, pelaporan beban kerja dosen dan remunerasi dosen. Aplikasi SIMDOS masih menggunakan arsitektur monolithic. Problem yang terjadi saat ini adalah susahnya untuk melakukan optimasi secara horizontal, yaitu ketika sistem sedang melakukan proses perhitungan berat (remunerasi) maka akan membuat sistem tidak bisa diakses sampai proses perhitungan selesai. Selain itu aplikasi SIMDOS juga merupakan aplikasi central di Universitas Udayana, yaitu

hampir semua aplikasi meminta data dosen dan pegawai ke SIMDOS sehingga jika aplikasi SIMDOS down maka akan memengaruhi sistem lain yang bergantung pada SIMDOS.

Permasalahan-permasalahan di atas, yang terkait dengan efisiensi resource dan kecepatan komputasi, merupakan sebuah tantangan dalam pengembangan aplikasi yang mempunyai kompleksitas tinggi. Permasalahan ini memunculkan ide untuk melakukan penelitian dalam pengembangan metode pada arsitekur *microservice*. Dengan aristektur microservice ini maka sistem SIMDOS akan dipecah menjadi beberapa service yang saling terkait. Dengan ini maka untuk melakukan scaling aplikasi akan lebih mudah (Jamshidi dkk., 2018).

1.2 Tujuan Khusus

Secara khusus, penelitian ini bertujuan untuk melakukan trasnformasi pada Sistem Informasi Manajemen Dosen (SIMDOS) Universitas Udayana dari arsitektur monolit menuju arsitektur *microservices*. Manfaat dari transformasi ini adalah untuk mendapatkan suatu sistem yang lebih mudah dalam pengembangan, penanganan masalah, dan distribusi resource yang tepat.

1.3 Urgensi (Keutamaan) Penelitian

Beberapa hal yang mendasari dan menjadi urgensi penelitian ini adalah sebagai berikut,

- Microservices memungkinkan pengembangan aplikasi dengan lebih cepat
- *Scaling* bisa dilakukan dengan lebih efisien
- Kesalahan pada saat pengembangan bisa diisolasi
- Beban pada server akan terkonsentrasi untuk layanan tertentu saja
- Pengembangan untuk model redundansi akan lebih mudah

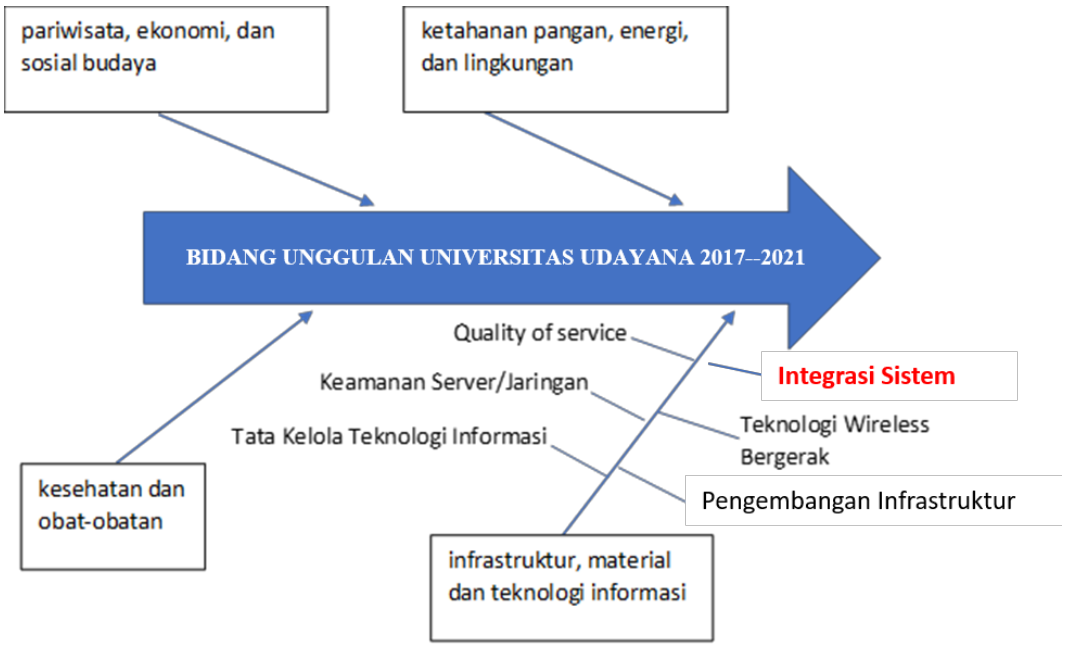
1.4 Rencana Target Capaian Tahunan

No	Jenis Luaran				Indikator Capaian	
	Kategori	Sub-Kategori	Wajib	Tambahan	TS	TS+1
1	Artikel Ilmiah dibuat di Jurnal	Internasional bereputasi	√		√	
2	Paten Model Arsitektur Microservices			√		√
3	Artikel Ilmiah dibuat di Jurnal	Internasional bereputasi	√			√

1.5 Keterkaitan Dengan RIP UNUD

Pengembangan sistem informasi di Universitas Udayana adalah sebagai bentuk berubahnya tata kelola institusi dari manual ke digital menuju era Revolusi Industri 4.0. Unit Sumber Daya Informasi (USDI) sebagai unit yang bertanggung jawab mengembangkan seluruh sistem informasi, di satu sisi mesti juga memikirkan dan bertanggung jawab atas keberadaan *resource* atau infrastruktur Teknologi Informasi. Infrastruktur teknologi informasi dan komunikasi dapat berupa konektivitas, komputasi, storage, dan supply daya. Keempat hal ini adalah pendukung dari sebuah atau lebih sistem informasi sehingga dapat digunakan oleh penggunanya secara handal. Agar semua *resource* tersebut dapat digunakan secara optimal, banyak penelitian telah dilakukan. Salah satu penelitian terkait hal ini adalah penelitian *microservice*, yaitu sebuah sistem informasi dapat bekerja berdasarkan kebutuhan modul-modul kecil yang menggunakan *resource* yang terdistribusi dan seefisien mungkin. Semakin banyak sistem yang dikembangkan maka semakin tinggi kompleksnya penggunaan resources bersama. Dampaknya dapat berupa pada sistem yang menjadi lambat sehingga kenyamanan pengguna terganggu. Kompleksitasi sistem terintegrasi juga menyebabkan semakin sulitnya melakukan pengembangan sistem informasi itu sendiri. Sehingga ke depannya USDI perlu memikirkan untuk mengembangkan suatu model yang dapat membantu mempercepat pengembangan sistem informasi, penggunaan resource yang efisien, dan efektif.

Gambar 1.1 menunjukkan bahwa Universitas Udayana memiliki empat bidang unggulan penelitian, yaitu: 1) pariwisata, ekonomi, dan sosial budaya; 2) ketahanan pangan, energi, dan lingkungan; 3) kesehatan dan obat-obatan; dan 4) infrastruktur, material dan teknologi informasi. Sistem Informasi merupakan bagian dari Teknologi Informasi sehingga merupakan bagian pada kelompok infrastruktur, material, dan teknologi informasi. Dengan mengambil topik ini, maka penelitian ini adalah yang sangat erat kaitannya dengan Rencana Induk Penelitian (RIP) Universitas Udayana, terutama dalam mengoptimalkan dan mengefisienkan penggunaan infrastuktur Teknologi Informasi dan Komunikasi sehingga sistem informasi sebagai wujud tata kelola di era Revolusi Industri 4.0 dapat berjalan dengan efektif, cepat, dan tepat sasaran.



Gambar 1.1. Posisi penelitian di RIP UNUD (yaitu pada bagian Integrasi Sistem)

BAB II
TINJAUAN PUSTAKA

Bagian ini memberikan latar belakang yang jelas tentang arsitektur layanan microser dan membandingkannya dengan pendahulunya. Juga dibahas tentang microservices yang *immutable* dan bagaimana mereka bisa mengubah cara aplikasi dikembangkan.

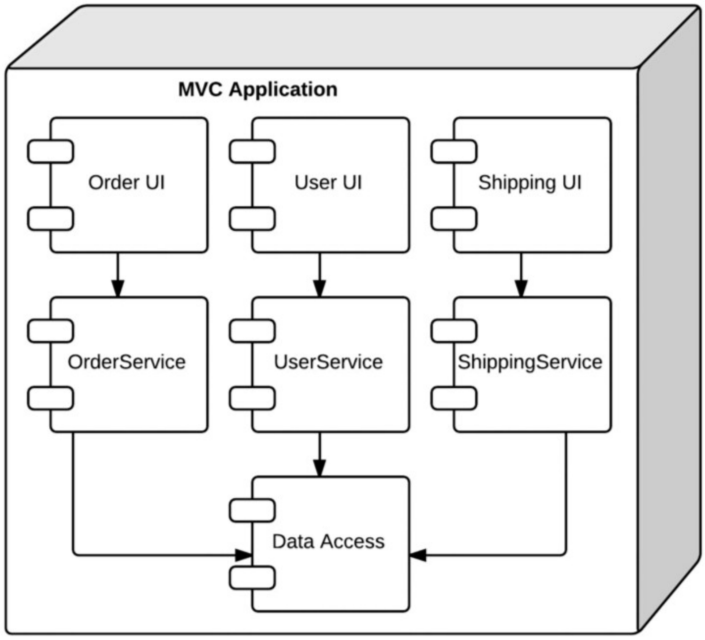
2.1 Arsitektur Sistem

Dalam beberapa tahun terakhir, arsitektur aplikasi web telah berkembang dengan pesat. Dengan munculnya layanan Internet, bisa dilihat bagaimana pergeseran paradigma terjadi, dari arsitektur monolitik berjenjang untuk Arsitektur Berorientasi Layanan (*Service Oriented Architecture*/SOA), dan dari SOA untuk layanan *microservice*. Pemilihan arsitektur perangkat lunak tertentu sepenuhnya tergantung pada persyaratan seperti skalabilitas, kompleksitas, dan kemudahan penempatan. Skalabilitas adalah salah satu ciri terpenting aplikasi web.

2.2 Monolithic Applications

Aplikasi Monolitik cenderung untuk menyatukan semua fungsi yang dibutuhkan dan dikembangkan dan digunakan sebagai satu unit. Ini adalah bentuk arsitektur paling sederhana dan berjalan cukup baik selama dijaga untuk memiliki kompleksitas rendah. Permasalahan muncul ketika arsitektur perlu meningkatkan fiturnya. Seiring berjalannya waktu, peningkatan ukuran aplikasi dan kompleksitas menghasilkan penurunan kecepatan pengembangan, pengujian dan penyebaran. Bahkan jika hanya ingin dikembangkan fitur sederhana yang hanya membutuhkan keadaan yang berbeda pada beberapa baris kode saja, tetapi karena arsitektur yang kompleks, beberapa baris tersebut bisa menjadi ribuan baris, sehingga mengurangi kecepatan pengembangan. Pengujian dan pengembangan monolith dengan jumlah fitur yang meningkat membutuhkan waktu yang relatif lama.

Menskalakan arsitektur monolit seringkali menghasilkan pemanfaatan sumber daya yang tidak seimbang. Masing-masing bagian dari aplikasi tidak bisa diskalakan karena semua kode aplikasi berjalan dalam proses yang sama di server. Sebagai contoh, pada Gambar 2.3, misalkan hanya ingin ditingkatkan layanan pesanan, dua layanan lainnya juga diskalakan dan diduplikasi pada server baru. Jika satu layanan bekerja secara intensif pada pemakaian memori dan layanan lain hanya intensif pada pemakaian CPU, server harus dilengkapi dengan memori dan CPU yang cukup untuk menangani beban dasar untuk setiap layanan. Ini bisa menjadi mahal jika masing-masing server membutuhkan jumlah CPU dan RAM yang tinggi, dan diperburuk lagi bahwa load balancing digunakan untuk mengukur aplikasi secara horizontal. Juga, semua aplikasi harus dipindahkan setiap kali diperbarui.



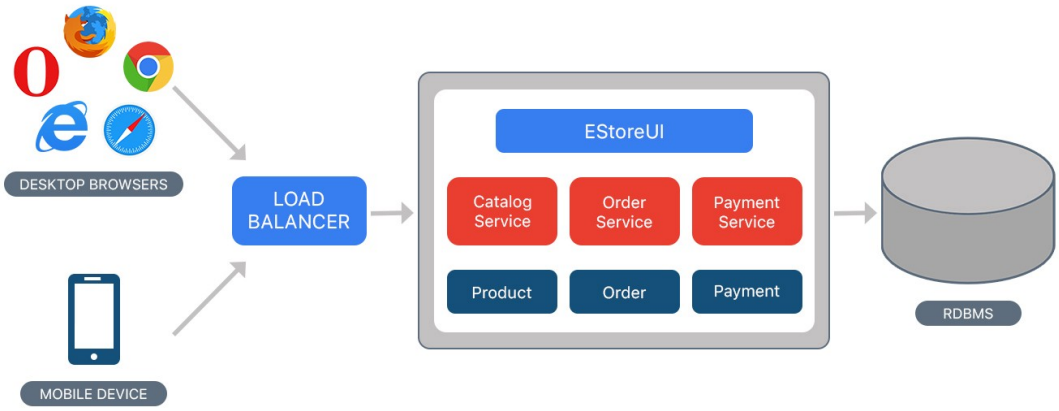
Gambar 2.1. Arsitektur Monolit

Secara harfiah, monolith berarti semua dalam satu bagian. Aplikasi Monolithic menggambarkan aplikasi perangkat lunak satu tingkat/level dimana komponen yang berbeda digabungkan menjadi satu program dalam satu platform tunggal. Komponen dapat berupa:

- Otorisasi - bertanggung jawab untuk mengotorisasi pengguna

- Presentasi - bertanggung jawab untuk menangani permintaan HTTP dan merespons dengan HTML atau JSON / XML (untuk API layanan web).
- Logika bisnis - logika bisnis aplikasi.
- Lapisan basis data - objek akses data yang bertanggung jawab untuk mengakses basis data.
- Integrasi aplikasi - integrasi dengan layanan lain (mis. Via REST API), atau integrasi dengan sumber data lainnya.
- Modul pemberitahuan - bertanggung jawab untuk mengirim pemberitahuan email kapan pun diperlukan.

Sebagai contoh, tinjau suatu aplikasi E-commerce, yang memberi otorisasi kepada pelanggan, melakukan pemesanan, memeriksa inventaris produk, mengesahkan pembayaran, dan mengirimkan produk yang dipesan. Aplikasi ini terdiri dari beberapa komponen termasuk antarmuka pengguna e-store untuk pelanggan (tampilan web Store) bersama dengan beberapa layanan backend untuk memeriksa inventaris produk, mengesahkan dan membebankan pembayaran serta pesanan pengiriman.



Gambar 2.2. Contoh Aplikasi dengan Arsitektur Monolitik

Meskipun memiliki komponen/modul/layanan yang berbeda, aplikasi ini dibangun dan digunakan sebagai satu aplikasi untuk semua platform (mis. desktop, seluler, dan tablet) menggunakan RDBMS sebagai sumber data.

Manfaat dan Kerugian dari Arsitektur Monolitik.

Manfaat:

- Mudah dikembangkan - pada awal pekerjaan, jauh lebih mudah menggunakan Arsitektur Monolitik.
- Mudah diuji. Misalnya, dapat diterapkan pengujian *end to end* hanya dengan meluncurkan aplikasi dan menguji UI dengan *Browser automation*.
- Mudah digunakan, hanya perlu menyalin aplikasi paket ke server.
- Sederhana untuk pengembangan secara horizontal dengan menjalankan banyak salinan di belakang *load balancer*.

Kekurangan:

- Pemeliharaan - Jika aplikasi terlalu besar dan kompleks untuk dipahami sepenuhnya, akan menimbulkan masalah untuk melakukan perubahan dengan cepat dan benar.
- Ukuran aplikasi dapat memperlambat waktu start-up.
- Seluruh aplikasi digunakan kembali pada saat pembaharuan (*update*)
- Aplikasi monolitik juga dapat memberikan masalah dalam penskalaan jika modul yang berbeda memiliki persyaratan sumber daya yang saling bertentangan.
- Keandalan - Bug dalam modul apapun berpotensi dapat menghentikan keseluruhan proses. Terlebih lagi, karena semua *instance* aplikasi identik, bug itu memengaruhi kerja seluruh aplikasi

Terlepas dari betapa mudahnya pada tahap awal, aplikasi monolitik mengalami kesulitan untuk mengadopsi teknologi baru dan maju. Karena perubahan dalam bahasa atau kerangka kerja memengaruhi seluruh aplikasi, itu membutuhkan upaya untuk benar-benar bekerja pada detail aplikasi, oleh karena itu perlu dipertimbangkan waktu dan energi yang akan terbuang.

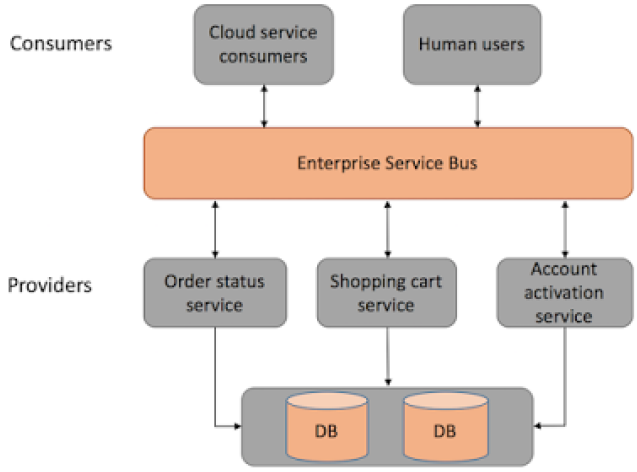
Secara ringkas, Monolith bekerja dengan baik pada tahap awal pekerjaan, tetapi ketika mereka tumbuh, mereka cenderung menjadi lebih kompleks dan sulit untuk

diskalakan. Pengembangan yang terus menerus dan penerapan berkelanjutan sulit dicapai untuk aplikasi ini. Mereka juga memiliki penghalang untuk mengadopsi teknologi baru. Karena perubahan kerangka atau bahasa akan memengaruhi seluruh lapisan aplikasi bisnis dan oleh karenanya menjadi mahal dilihat dari segi waktu dan biaya.

2.3 Service-oriented architecture (SOA)

SOA muncul sebagai cara untuk memecahkan arsitektur ketat yang dibuat oleh aplikasi monolitik. Arsitektur SOA dapat dilihat sebagai arsitektur terdistribusi berbasis komponen. Komponen (ke dalam layanan), ikatan yang longgar, kinerja tinggi, manajemen siklus hidup layanan (mengelola layanan, mengambil tahanan sentral, mempromosikan penggunaan kembali, mengelola acara siklus hidup melalui penciptaan hingga penghentian akhir) adalah fitur penting dari SOA.

SOA terdiri dari dua peran, penyedia layanan dan konsumen layanan. Lapisan Pelanggan adalah titik di mana konsumen (pengguna manusia, layanan lain atau pihak ketiga) berinteraksi dengan SOA dan Penyedia Lapisan terdiri dari semua layanan yang secara longgar digabungkan. Kedua lapisan berkomunikasi melalui ESB (Enterprise Service Bus) seperti yang ditunjukkan pada Gambar 2.3. yang mempublikasikan kemampuan bisnis sebagai layanan untuk konsumen dan mengarahkan pesan ke back-end dengan kemampuan mentransformasikan, mengamankan dan menangani pengecualian pengiriman. ESB ini sendiri sangat kompleks dan sering berkontribusi untuk membuat aplikasi monolit besar dari Arsitektur SOA lainnya. Hal ini mendasari kelahiran layanan *microservices*.



Gambar 2.3 Service Oriented Architecture

Arsitektur berorientasi layanan (SOA) adalah gaya desain perangkat lunak di mana layanan diberikan kepada komponen lain oleh komponen aplikasi, menggunakan protokol komunikasi melalui jaringan. Prinsip dasar arsitektur berorientasi layanan adalah independen terhadap vendor, produk, dan teknologi. Layanan adalah unit fungsionalitas terpisah yang dapat diakses dari jarak jauh dan ditindak-lanjuti serta diperbarui secara independen.

Suatu layanan memiliki empat properti menurut salah satu dari banyak definisi SOA:

1. Secara logis mewakili aktivitas bisnis dengan hasil yang spesifik.
2. Bersifat mandiri.
3. Merupakan *black box* untuk konsumennya.
4. mungkin terdiri dari layanan mendasar lainnya

Layanan yang berbeda dapat digunakan bersamaan untuk menyediakan fungsionalitas aplikasi perangkat lunak besar, suatu prinsip SOA yang menyerupai pemrograman modular. Arsitektur berorientasi layanan mengintegrasikan komponen perangkat lunak terdistribusi, yang dikembangkan secara terpisah. Ini dimungkinkan karena adanya teknologi dan standar yang memfasilitasi komunikasi komponen dan kerjasama melalui jaringan, terutama melalui jaringan IP.

SOA atau arsitektur berorientasi layanan adalah gaya desain arsitektur yang dimaksudkan sebagai cara untuk memecah aplikasi monolitik menjadi modul yang lebih kecil yang berorientasi pada tujuan bisnis. Modul-modul ini dapat berkisar dalam ukuran dari layanan aplikasi kecil hingga layanan perusahaan besar. SOA bergantung pada protokol pengiriman pesan seperti (AMQP atau SOAP) untuk berkomunikasi antar layanan.

2.3.1 Pendekatan implementasi SOA

Arsitektur berorientasi layanan dapat diimplementasikan dengan layanan web. Hal ini dilakukan untuk membuat blok bangunan fungsional dapat diakses melalui protokol Internet standar yang tidak tergantung pada platform dan bahasa pemrograman. Layanan ini dapat berupa aplikasi baru atau sistem lama yang terbungkus, yang membuatnya bisa bekerja dalam jaringan.

Umumnya, pelaksanaan pengembangan SOA dilakukan menggunakan standar layanan web. Salah satu contoh adalah SOAP, yang telah memperoleh penerimaan luas dari industri setelah adanya rekomendasi Versi 1.2 dari W3C (World Wide Web Consortium) pada tahun 2003. Standar-standar ini (juga disebut sebagai spesifikasi layanan web) juga memberikan interoperabilitas yang lebih besar dan beberapa perlindungan dari kuncian ke perangkat lunak vendor secara eksklusif. Seseorang dapat, bagaimanapun, juga mengimplementasikan SOA menggunakan teknologi berbasis layanan lainnya, seperti Jini, CORBA atau REST.

Arsitektur dapat beroperasi secara independen dari teknologi tertentu dan karenanya dapat diimplementasikan menggunakan berbagai teknologi, termasuk:

- Layanan web berdasarkan WSDL dan SOAP
- Perpesanan, mis., Dengan ActiveMQ, JMS, RabbitMQ
- HTTP tenang, dengan Representasi state transfer (REST) merupakan gaya arsitektur berbasis kendala sendiri
- OPC-UA

- WCF (Implementasi Microsoft terhadap layanan Web, membentuk bagian dari WCF)
- Penghematan Apache Thrift
- gRPC
- SORCER

Implementasi dapat menggunakan satu atau lebih dari protokol di atas, dan, misalnya, mungkin menggunakan mekanisme sistem file untuk mengkomunikasikan data setelah spesifikasi antar-muka yang ditentukan antara proses yang sesuai dengan konsep SOA. Kuncinya adalah layanan independen dengan antar-muka yang ditentukan yang dapat dipanggil untuk melakukan tugas-tugas mereka dengan cara standar, tanpa layanan yang memiliki pengetahuan tentang aplikasi panggilan, dan tanpa aplikasi yang memiliki atau membutuhkan pengetahuan tentang bagaimana layanan sebenarnya melakukan tugasnya. SOA memungkinkan pengembangan aplikasi yang dibangun dengan menggabungkan layanan yang digabungkan secara longgar dan interoperable.

Layanan ini beroperasi berdasarkan definisi formal yang independen dari platform dan bahasa pemrograman yang mendasarinya. Definisi antar-muka menyembunyikan implementasi layanan khusus bahasa. Sistem berbasis SOA dapat berfungsi secara independen dari teknologi dan platform pengembangan (seperti Java, .NET, dll.). Layanan yang ditulis dalam C # berjalan pada platform .NET dan layanan yang ditulis dalam Java yang berjalan pada platform Java EE, misalnya, keduanya dapat dikonsumsi oleh aplikasi komposit umum (atau klien). Aplikasi yang berjalan di kedua platform juga dapat menggunakan layanan yang berjalan di sisi lain sebagai layanan web yang memfasilitasi penggunaan kembali. Lingkungan yang dikelola juga dapat membungkus sistem warisan COBOL dan menghadirkannya sebagai layanan perangkat lunak.

Bahasa pemrograman tingkat tinggi seperti BPEL dan spesifikasi seperti WS-CDL memperluas konsep layanan dengan menyediakan metode mendefinisikan dan mendukung orkestrasi layanan fine-grained menjadi layanan bisnis yang coarse-grained, yang dapat diubah dan digabungkan ke dalam alur kerja dan proses bisnis untuk diimplementasikan dalam aplikasi atau portal komposit.

Pemodelan berorientasi layanan adalah kerangka kerja SOA yang mengidentifikasi berbagai disiplin ilmu yang memandu praktisi SOA untuk membuat konsep, menganalisis, merancang, dan merancang aset yang berorientasi layanan. Kerangka kerja pemodelan berorientasi layanan (SOMF) menawarkan bahasa pemodelan dan struktur kerja atau "peta" yang menggambarkan berbagai komponen yang berkontribusi pada pendekatan pemodelan berorientasi layanan yang berhasil. Ini menggambarkan elemen-elemen utama yang mengidentifikasi aspek "apa yang harus dilakukan" dari skema pengembangan layanan. Model ini memungkinkan para praktisi untuk menyusun rencana proyek dan mengidentifikasi tonggak dari inisiatif yang berorientasi layanan. SOMF juga menyediakan notasi pemodelan umum untuk mengatasi keselarasan antara bisnis dan organisasi TI.

SOA telah digabungkan dengan layanan Web, tapi layanan Web hanya satu opsi untuk menerapkan pola-pola yang membentuk gaya SOA. Dengan tidak adanya bentuk asli atau biner panggilan prosedur jarak jauh (RPC), aplikasi dapat berjalan lebih lambat dan membutuhkan lebih banyak daya pemrosesan, meningkatkan biaya. Sebagian besar implementasi memang dikenai biaya overhead ini, tetapi SOA dapat diimplementasikan menggunakan teknologi (misalnya, Java Business Integration (JBI), Windows Communication Foundation (WCF) dan layanan distribusi data (DDS)) yang tidak bergantung pada panggilan prosedur jarak jauh atau terjemahan melalui XML Pada saat yang sama, muncul teknologi parsing XML open-source (seperti VTD-XML) dan berbagai format biner yang kompatibel dengan XML berjanji untuk secara signifikan meningkatkan kinerja SOA. Layanan yang diimplementasikan menggunakan JSON, bukan XML, tidak mengalami masalah kinerja ini.

Layanan stateful mengharuskan konsumen dan penyedia untuk berbagi konteks khusus konsumen yang sama, yang termasuk dalam atau direferensikan oleh pesan yang dipertukarkan antara penyedia dan konsumen. Kendala ini memiliki kelemahan yang dapat mengurangi skalabilitas keseluruhan penyedia layanan jika penyedia layanan perlu mempertahankan konteks bersama untuk setiap konsumen. Ini juga meningkatkan sambungan antara penyedia layanan dan konsumen dan membuat penyedia layanan

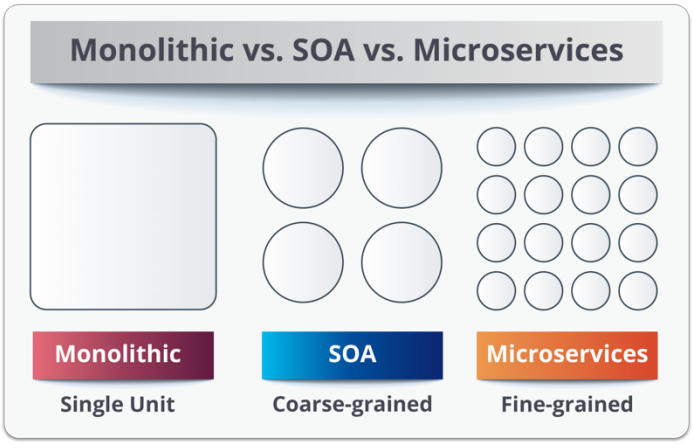
switching lebih sulit. Pada akhirnya, beberapa kritikus merasa bahwa layanan SOA masih terlalu dibatasi oleh aplikasi yang mereka wakili.

Tantangan utama yang dihadapi oleh arsitektur berorientasi layanan adalah pengelolaan metadata. Lingkungan berdasarkan SOA mencakup banyak layanan yang saling berkomunikasi untuk melakukan tugas. Karena kenyataan bahwa desain mungkin melibatkan beberapa layanan yang bekerja bersama, suatu Aplikasi dapat menghasilkan jutaan pesan. Layanan lebih lanjut mungkin milik organisasi yang berbeda atau bahkan perusahaan yang bersaing menciptakan masalah kepercayaan yang sangat besar. Dengan demikian tata kelola SOA masuk ke dalam skema berbagai hal.

Masalah utama lain yang dihadapi oleh SOA adalah kurangnya kerangka pengujian yang seragam. Tidak ada alat yang menyediakan fitur yang diperlukan untuk menguji layanan ini dalam arsitektur berorientasi layanan. Penyebab utama kesulitan adalah:

- Heterogenitas dan kompleksitas solusi.
- Kombinasi pengujian yang sangat besar karena integrasi layanan otonom.
- Termasuk layanan dari vendor yang berbeda dan bersaing.
- Platform terus berubah karena ketersediaan fitur dan layanan baru.

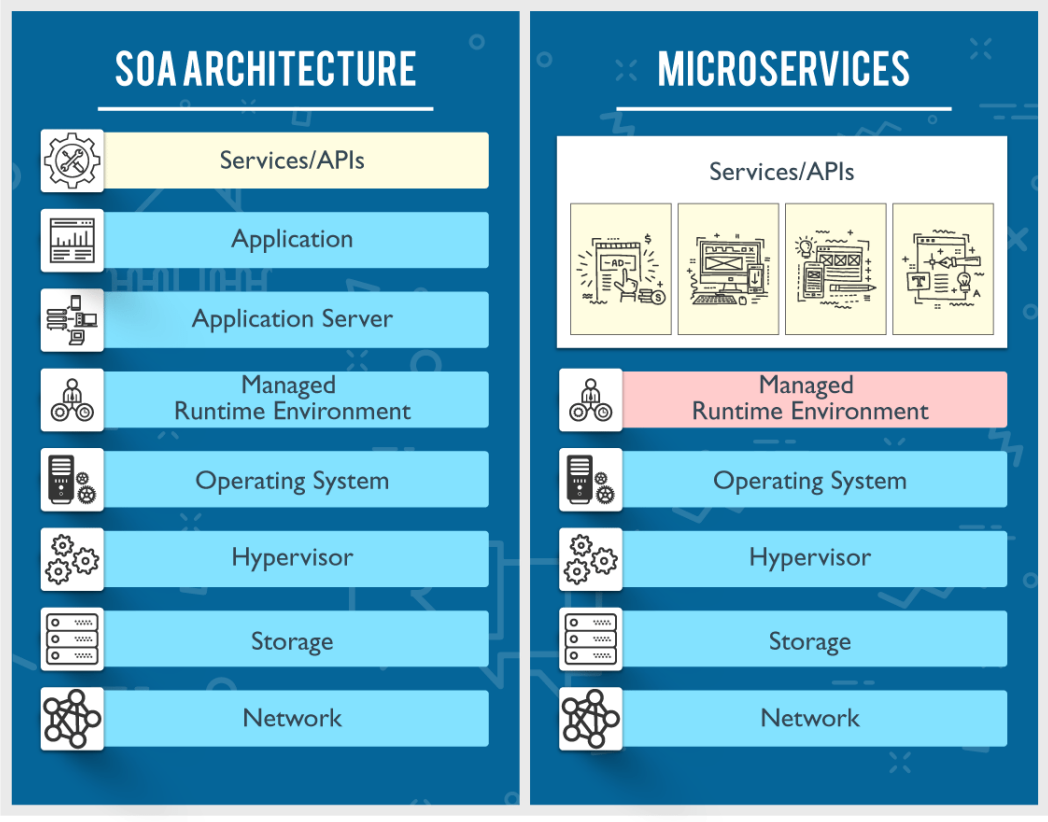
Perbedaan utama antara arsitektur monolitik, SOA, dan layanan *microservice* diilustrasikan dalam Gambar 2.4.



Gambar 2.4 Ilustrasi perbedaan Monolitik, SOA, dan Microservice

Dalam istilah awam, monolith mirip dengan wadah besar, yaitu semua komponen perangkat lunak aplikasi dirakit bersama dan dikemas secara ketat. Arsitektur berorientasi layanan (SOA) pada dasarnya adalah kumpulan layanan. Layanan-layanan ini saling berkomunikasi. Komunikasi dapat melibatkan baik penyampaian data sederhana atau dua atau lebih layanan yang mengoordinasi beberapa kegiatan. Diperlukan beberapa cara untuk menghubungkan layanan satu sama lain. Microservices, alias arsitektur microservice, adalah gaya arsitektur yang menyusun aplikasi sebagai kumpulan layanan otonom kecil yang dimodelkan di sekitar domain bisnis.

Jika dibandingkan antara microservices dengan SOA, keduanya bertumpu pada layanan sebagai komponen utama, tapi terdapat perbedaan yang besar dalam karakteristik layanan, yang ditunjukkan dalam Gambar 2.5.



Gambar 2.5. SOA vs Microservices

Tabel 2.1 memberikan deskripsi perbedaan SOA dan *microservices*.

Tabel 2.1. Karakteristik SOA dan Microservices

SOA	MSA
Follows "**share-as-much-as-possible**" architecture approach	Follows "**share-as-little-as-possible**" architecture approach
Importance is on **business functionality** reuse	Importance is on the concept of "**bounded context**"
They have **common governance** and **standards**	They focus on **people, collaboration** and freedom of other options
Uses **Enterprise Service bus (ESB)** for communication	Simple messaging system
They support **multiple message protocols**	They use **lightweight protocols** such as **HTTP/REST** etc.
Multi-threaded with more overheads to handle I/O	**Single-threaded** usually with the use of Event Loop features for non-locking I/O handling
Maximizes application service reusability	Focuses on **decoupling**
Traditional Relational Databases are more often used	**Modern Relational Databases** are more often used
A systematic change requires modifying the monolith	A systematic change is to create a new service
DevOps / Continuous Delivery is becoming popular, but not yet mainstream	Strong focus on DevOps / Continuous Delivery

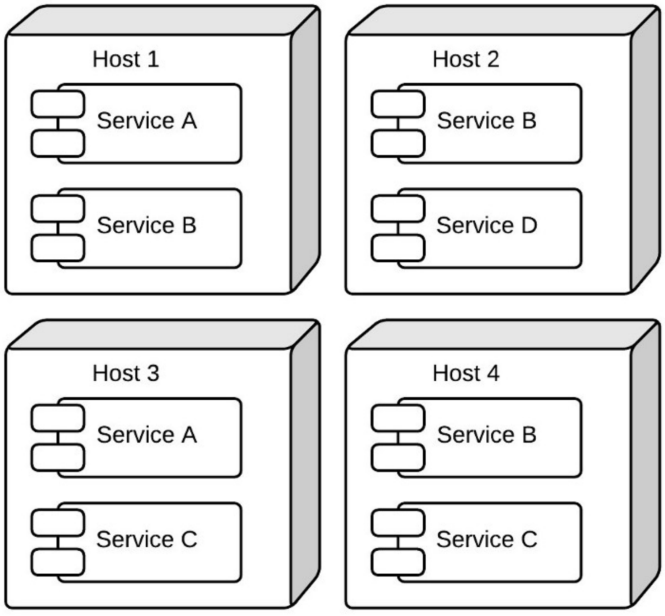
2.4 Microservices

Istilah "Arsitektur Layanan Mikro atau *microservices*" telah muncul selama beberapa tahun terakhir untuk menjelaskan suatu cara khusus dalam merancang aplikasi perangkat lunak sebagai rangkaian layanan yang dapat digunakan sendiri (Gannon dkk., 2015). Meskipun tidak ada definisi yang tepat dari gaya arsitektur ini, terdapat karakteristik umum tertentu dalam kemampuan bisnis, penyebaran otomatis, intelijen di titik akhir, dan kontrol desentralisasi bahasa dan data. Idenya adalah untuk membagi aplikasi menjadi satu set layanan yang lebih kecil, saling berhubungan dan independen daripada membangun aplikasi monolitik tunggal (Alshuqayran dkk, 2016).

Setiap *microservice* adalah aplikasi kecil yang memiliki arsitektur heksagonal sendiri, yang terdiri dari logika bisnis bersama dengan berbagai adapter dan dikembangkan, diuji, dan dikerahkan secara terpisah dari satu sama lain. Layanan

microservice ini akan memaparkan REST atau API berbasis pesan untuk berkomunikasi dengan layanan microser yang berbeda (Gannon, 2015).

Arsitektur *microservice* sering susah dibedakan dengan layanan tipe SOA tradisional. Meskipun ada banyak tumpang tindih, ada banyak perbedaan di antara mereka. Menurut Martin Fowler (, microservices adalah salah satu bentuk SOA, dengan orientasi layanan yang dilakukan dengan benar. Microservices dapat dikategorikan sebagai layanan versi ringan dari SOA. SOA lebih berorientasi pada sisi organisasi TI, sedangkan layanan mikro lebih terkait dengan produk berbasis SAAS (software as a service atau perangkat lunak berbentuk layanan). SOA sebagian besar menggunakan XML / WSDL sedangkan Arsitektur microservice mengadopsi REST API. Perbedaan utama antara SOA dan layanan mikro adalah bahwa yang belakangan lebih mandiri dan digunakan secara mandiri.



Gambar 2.6 Arsitektur Microservices

Ada banyak keuntungan untuk memilih *microservice* dari pada aplikasi monolitik.

1. *Scaling*

   Penskalaan microservices jauh lebih mudah daripada aplikasi monolitik. Penskalaan layanan yang perlu ditingkatkan tidak seperti pada monolit, yang memerlukan

penskalaan keseluruhan aplikasi ke mesin baru. Selain itu, *microservice* memungkinkan untuk mengelompokkan sumber daya yang menuntut layanan tinggi pada perangkat keras yang lebih baik, sehingga bisa membantu dalam mendistribusikan layanan dengan lebih baik pada keseluruhan infrastruktur.

2. Kelincahan dan Inovasi

   Layanan Microservice bersifat independen dan otonom yang menyiratkan suatu derajat kebebasan bagi pengembang dalam mengembangkan dan mempertahankan basis kode dan kemampuan mereka sendiri untuk cepat bereaksi terhadap perubahan skenario dibandingkan dengan monolit yang karena sifatnya lambat dalam perubahan.

3. Minimal dan Ringan

   Karena ukurannya yang kecil dan biasanya satu layanan tidak memberikan fungsi yang besar dan kompleks, mereka lebih mudah untuk dikembangkan dan bahkan IDE bisa bekerja dengan lebih cepat karena mengimpor fungsi yang kecil berarti menguragi beban IDE.

4. *Deployment, Rollback*, dan Isolasi Kesalahan

   Pengembangan parsial lebih cepat bisa dilakukan dengan *microservices* karena ukurannya. Adalah lebih mudah untuk mengembalikan (*rollback*) hanya satu layanan yang mengalami masalah dalam produksi ke versi yang lebih lama. Selama proses *rollback*, kesalahan hanya terisolasi pada satu layanan itu saja. Pengembangan berkelanjutan dapat dengan mudah dicapai dengan *microservices*.

Secara ringkas, migrasi arsitektur monolitik ke *microservice* membawa banyak manfaat. Secara khusus, ini memberikan kemampuan beradaptasi terhadap perubahan teknologi untuk menghindari penguncian teknologi dan, yang lebih penting, mengurangi waktu dalam pengembangan dan menghasilkan penataan yang lebih baik dalam memberikan layanan.

2.5 Kontainer

Penggunaan arsitektur baru berarti membuat banyak perubahan. Untuk mengembangkan aplikasi berbasis microservices, memunculkan beberapa pertanyaan,

seperti lingkungan eksekusi apa yang harus digunakan bagi aplikasi microservices? Atau dengan kata lain, di lingkungan seperti apa microservices harus berjalan?

2.5.1 Opsi Runtime

Beberapa dekade lalu, opsi yang tersedia untuk instalasi dan menjalankan microservices adalah pada server fisik yang bekerja di atas sebuah sistem operasi. Dengan melihat kemampuan server sekarang, model seperti itu akan banyak membuang *resources*. Cara yang lebih efisien adalah dengan menjalankan beberapa layanan (*service*) di atas satu sistem operasi, dengan resiko pada munculnya konflik pada versi library atau komponen aplikasi lain. Menjalankan microservices pada server fisik bukan pilihan yang menarik. Sebagai gantinya, server fisik bisa dibagi menjadi beberapa server virtual (virtual machine/VM), yang mengijinkan beberapa lingkungan eksekusi berada dalam satu server. Pilihan ini memiliki beberapa kekurangan dalam menjalankan microservices.

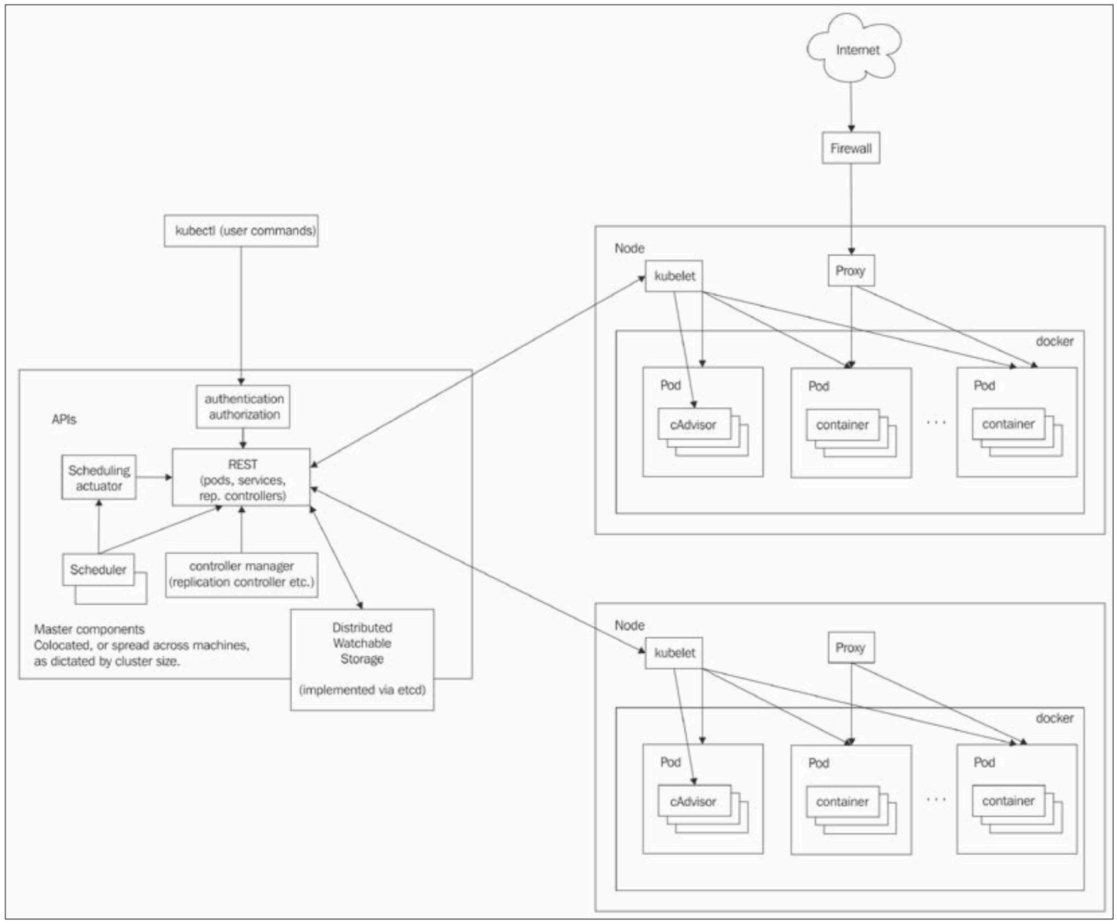
Pilihan terbaik adalah dengan menjalankan aplikasi microservice dalam kontainer. Kontainer membungkus (encapsulasi) suatu lingkungan dengan beban ringan untuk aplikasi microservices, memberikan lingkungan yang konsisten untuk bisa pengembangan, testing dan delivery aplikasi microservice. Kontainer bisa dijalankan dalam server fisik maupun server virtual.

2.6 Kubernetes

Kubernetes mempunyai fungsi utama untuk orkestrasi kontainer. Orkestrasi disini mempunyai pengertian tentang pengaturan, koordinasi, dan manajemen yang otomatis terhadap sistem dan layanan komputer yang kompleks. Ini berarti bahwa Kubernetes menjamin bahwa semua kontainer yang mengeksekusi beragam beban kerja bisa terjadwal untuk berjalan, baik pada mesin fisik maupun virtual. Kubernetes juga harus mengawasi semua kontainer yang bekerja, dan mengganti kontainer yang mati atau tidak memberi respons.

2.6.1 Konsep Kubernetes

Konsep Kubernetes adalah seperti pada gambar berikut:



Gambar 2.7 Konsep Kubernetes

Cluster adalah kumpulan host untuk storage dan networking yang digunakan oleh Kubernetes untuk menjalankan beragam beban kerja dari suatu sistem/aplikasi. Satu node mewakili satu host, yang bisa berupa fisik maupun virtual. Tugas node adalah untuk menjalankan pod. Setiap node pada Kubernetes menjalankan beberapa komponen Kubernetes, seperti kubelet dan proxy kube. Node dikeloka oleh master Kubernetes. Node-node merupakan pekerja dari Kubernetes, dan menanggung semua beban yang berat.

Master adalah pengontrol Kubernetes. Master terdiri dari beberapa komponen, seperti *API server*, *scheduler*, dan *controller manager*. Master bertanggung jawab terhadap penjadwalan pod secara global pada level cluster, dan juga penanganan *events*. Biasanya, semua komponen master di-set pada satu host. Adakalanya sebuah master redundancy dipergunakan dalam sistem dengan cluster yang besar.

Pod adalah unit perkerja dalam Kubernetes. Tiap pod memuat setidaknya satu kontainer. Pod selalu terjadwal bersama (selalu berjalan pada mesin yang sama). Semua kontainer dalam pod yang sama, memiliki alamat IP dan port yang sama, jadi mereka bisa saling berkomunikasi dengan localhost, atau komunikasi inter-process standar. Selain itu, semua kontainer dalam satu pod memiliki akses bersama (*shared*) pada lokal storage pada node yang menaungi (*host*) pod. Shared storage ini akan di-*mounted* pada tiap kontainer. Pod merupakan fitur penting dalam Kubernetes. Model ini memiliki banyak keuntungan antara lain :

- **Transparansi**. Dengan membuat infrastruktur bisa melihat kontainer dalam pod, memungkinkan infrastruktur untuk menyediakan layanan pada kontainer tersebut, seperti manajemen proses dan monitoring resources. Kondisi ini memberi banyak kemudahan bagi user.
- Decoupling **software dependencies**. Salah satu kontainer bisa diperbaharui secara independent. Di masa mendatang, Kubernetes bahkan bisa mendukung live update untuk kontainer individual.
- Kemudahan **penggunaan**. User tidak perlu menjalankan proses manajer-nya, atau memikirkan tentang signal dan exit-code.
- Efisiensi**.** Karena tanggung jawab lebih diberikan pada infrastruktur, kontainer bisa menjadi lebih ringan/kecil.

Pods menyediakan solusi yang sangat bagus untuk mengelola kelompok kontainer yang mempunyai kaitan erat, yang saling berketergantungan, dan perlu melakukan kerjasama dalam host yang sama untuk menyelesaikan pekerjaannya. Penting untuk diingat, bahwa pod dianggap sebagai entiti yang bersifat ephemeral (bertahan singkat) dan bisa dibuang setelah pekerjaan selesai, serta diganti sesuai keinginan. Setiap

penyimpanan pod juga dihilangkan bersamaan dengan pod-nya. Tiap pod memiliki ID unik (UID), sehingga mudah untuk dikenali.

2.6.2 Rancangan Pola Kontainer dalam Kubernetes

Secara umum, pola-pola tertentu telah dirancang dan diimplementasikan untuk menyelesaikan suatu permasalahan. Pola rancangan memberikan efisiensi dalam aplikasi, dan juga untuk developer, mengurangi overhead, dan memberi cara dalam penggunaan ulang kontainer dalam keseluruhan aplikasi. Ada beberapa cara dalam mengelompokkan kontainer dalam Pod Kubernetes.

2.6.2.1 Single node, multiple container patterns

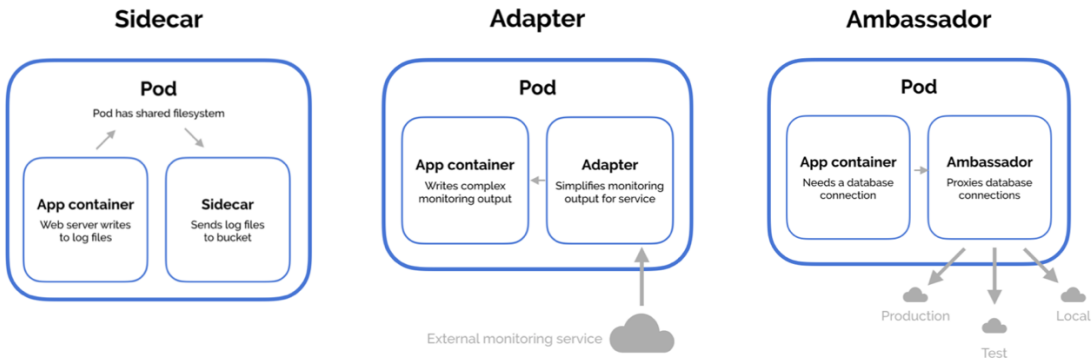
Dalam pola single node, semua kontainer dijadwalkan bersama dalam satu node atau mesin, dengan events terjadi antar kontainer dalam satu pod. Pola single node diatur langsung oleh Kubernetes melalui pod.

2.6.2.2 Sidecar pattern

Kontainer sidecar bekerja dengan kontainer utama (*primary container*). Pola ini baik digunakan jika ada perbedaan yang jelas antara primary container dan pekerjaan-pekerjaan lain yang harus dilakukan. Sebagai contoh, sebuah kontainer web server (sebagai aplikasi utama) yang perlu untuk memecah dan mengirim data log-nya ke penyimpanan log (sebagai tugas *secondary*) bisa menggunakan kontainer sidecar. Kontainer sidecar ini yang bertugas untuk forwarding log. Kontainer sidecar ini juga bisa digunakan di tempat lain dengan tugas yang sama, untuk forwarding log ke web server lain , atau bahkan dalam aplikasi lain. Cara ini memberikan manfaat yang besar, jika dibandingkan dengan penggunaan logging sentral ke kontainer utama. Dengan sidecar, aplikasi tidak lagi terbebani dengan logging sentral.

2.6.3 Ambassador pattern

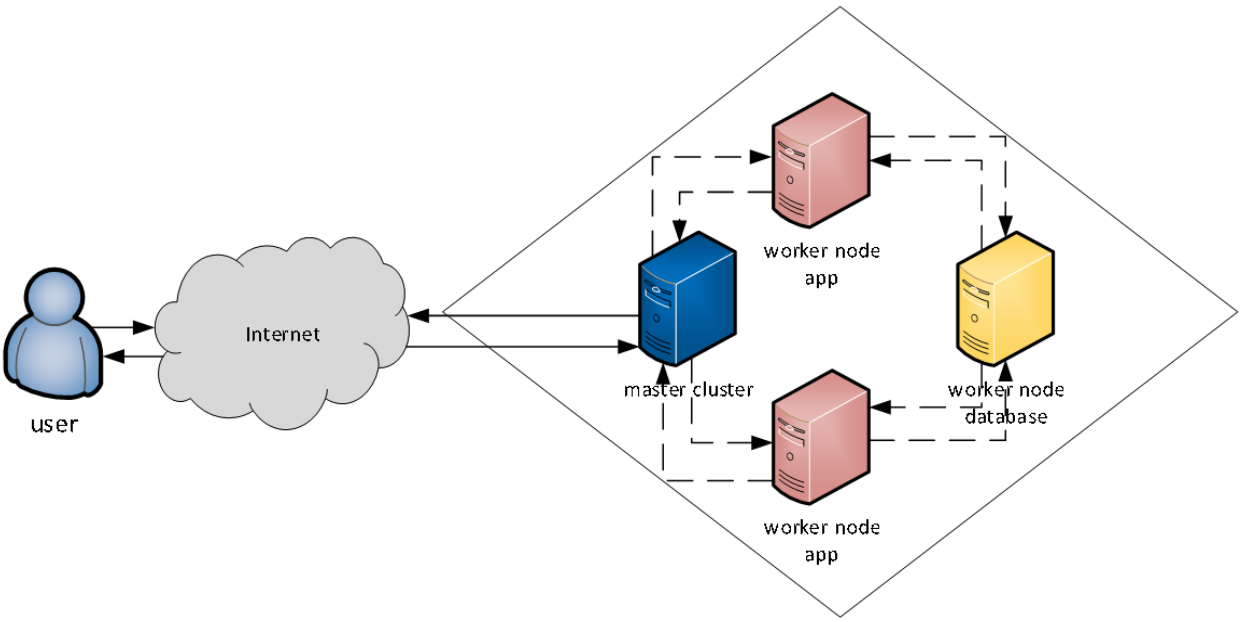
Pola ambassador adalah cara untuk menghubungkan kontainer dengan lingkungan di luarnya. Kontainer ambasador adalah berupa sebuah proxy yang mengijinkan kontainer lain untuk terhubung ke suatu port atau localhost. Salah satu use-case dari pola ambassador adalah penyediaan akses menuju database. Ketika di-develop lokal, bisa digunakan database lokal, sedangkan untuk pengujian dan produksi digunakan database lain.



Gambar 2.8 Pola Kontainer dalam Kubernetes

BAB III
METODE PENELITIAN

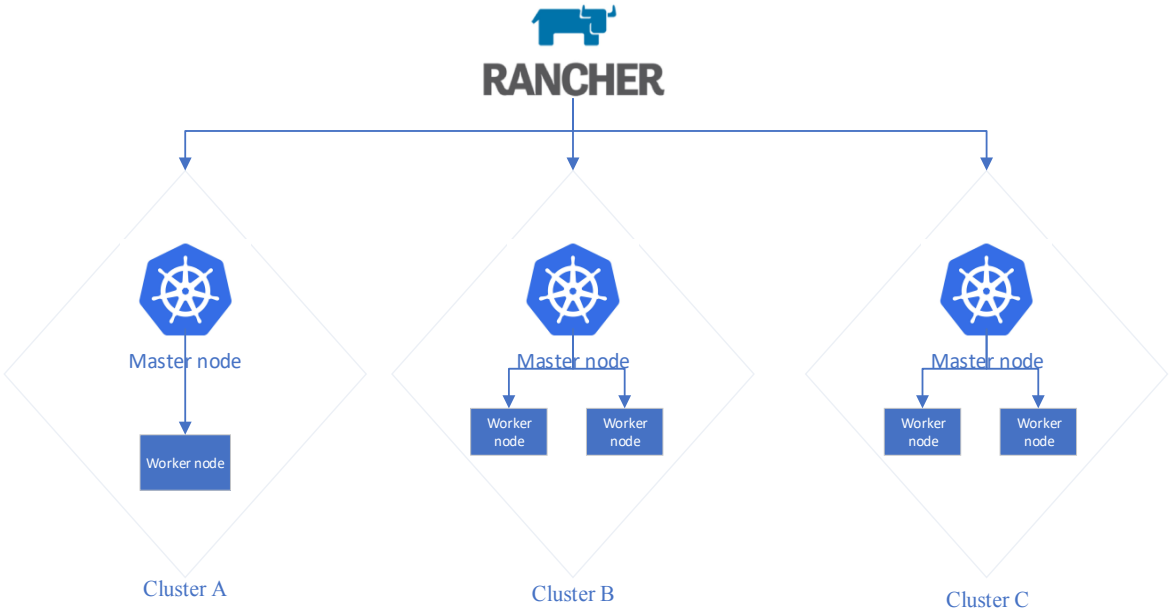
Penelitian ini adalah bagian II dari rencana penelitian dua tahun. Pada tahun I, telah dikembangkan load balancing dan model microservices untuk SIMDOS. *Microservice* adalah arsitektur yang membagi aplikasi menjadi satu set layanan yang lebih kecil, saling berhubungan dan independen. Aplikasi SIMDOS dengan arsitektur *microservice* dirancang menggunakan Docker sebagai platform virtualisasi berbasis *container* yang dikelola oleh Kubernetes sebagai *container orchestration*. Gambar 3.1 menampilkan model topologi SIMDOS pada sebuah Kubernetes *cluster* yang terdiri atas 2 komponen yakni *master* dan *node*. Komponen *master* (*master node*) berfungsi sebagai *control plane* bagi *cluster* yang berperan dalam mekanisme penjadwalan, mengekspos API Kubernetes, menyimpan data *cluster* serta bertanggung jawab pada proses deteksi serta pemberian respon terhadap *events* yang berlangsung di dalam *cluster*. Sedangkan, komponen *node* (*node worker*) berisi *container service* yang berguna menjalankan banyak *pod* yang diatur oleh *master node.*



Gambar 3.1 Topologi SIMDOS pada Kubernetes Cluster

User mengakses aplikasi SIMDOS yang sudah diekspose oleh *master cluster* melalui jaringan internet. *Master cluster* meneruskan *request* tersebut ke *worker node app* dengan *cluster IP* dan mekanisme *load balancing*. *Load balancing* merupakan sebuah teknik untuk mendistribusikan beban kerja dari 2 atau lebih server agar trafik berjalan secara optimal. Kemudian, *worker node app* mengakses data yang tersimpan pada worker node *database*

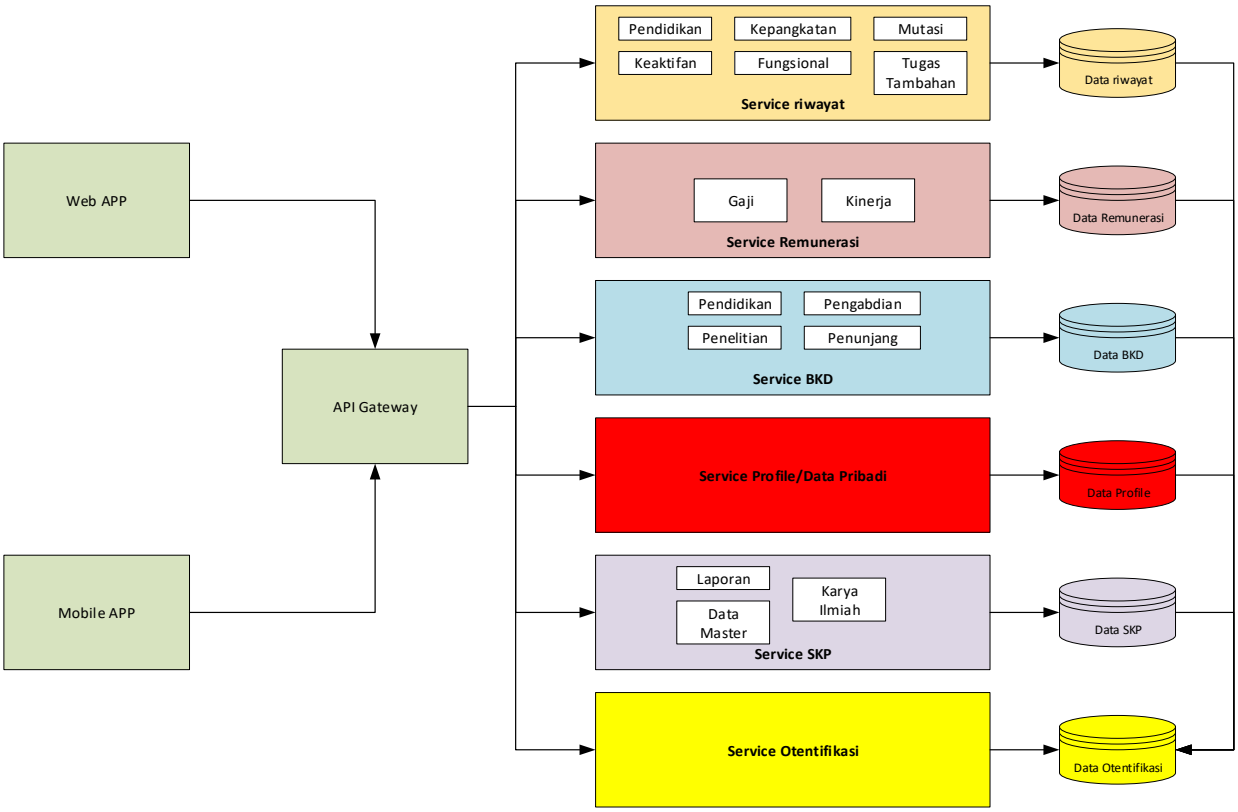
Proses pengembangan aplikasi SIMDOS akan menggunakan *software* Rancher sebagai Kubernetes *management*. Fungsi Rancher berdasarkan kemampuannya, yaitu dapat mengelola banyak *cluster* melalui Kubernetes *master node* yang sudah diintegrasi dengan Rancher seperti ditunjukkan pada Gambar 3.2



Gambar 3.2 Orchestrasi Kubernetes dengan Rancher

Rancher mengimplementasikan lapisan portabel layanan infrastruktur yang dirancang khusus untuk menjalankan aplikasi kontainer. Layanan infrastruktur Rancher yaitu *networking*, storage, *load balancer*, DNS, dan *security* yang dijalankan sebagai container. Rancher mendukung plugin otentikasi pengguna yang fleksibel dan dilengkapi dengan integrasi otentikasi pengguna yang sudah dibuat sebelumnya dengan Active Directory, LDAP, dan GIT.

Arsitektur microservice mampu menangani kasus peningkatan ukuran dan kompleksitas dari codebase program. Pemisahan dan pendefinisian modul yang baik memastikan bahwa kompleksitas terisolasi sehingga membuat pengembangan lebih cepat. Arsitektur microservice mempermudah pengembangan fitur baru ketika kompleksitas sistem tinggi karena setiap service memiliki codebase yang kecil dan modularitas lebih mudah dipertahankan ketika ada batasan modul yang jelas. Model aplikasi SIMDOS dengan arsitektur microservice dibagi menjadi 6 kelompok service yakni riwayat, BKD, SKP, profile, otentifikasi dan remunerasi yang masing-masing memiliki database sesuai Gambar 3.3. Didalam kelompok service tersebut, terdapat lagi set layanan yang lebih kecil dan independent namun tetap dapat saling berhubungan satu sama lainnya.



Gambar 3.3 Model Microservice SIMDOS

Gambar 3.3 di atas menunjukkan bahwa semua set layanan pada aplikasi SIMDOSE dapat diakses melalui website ataupun mobile melalui API Gateway. Microservice yang terdiri dari beberapa service atau kumpulan API berbentuk REST API menggunakan protokol HTTP. Pada monolithic, user hanya mengakses sebuah URL REST API, sedangkan microservice memiliki banyak service REST API sehingga memerlukan API gateway sebagai menejemen API seperti API merger, API authentication dan lainnya.

Penelitian Tahun II

Pada tahun kedua, penelitian ini akan fokus pada beberapa hal yaitu

1. pengembangan mirroring server, yang masing-masing berlaku sebagai *active* dan *back up* server. Kedua kelompok server akan terpisah secara geografis, satu terletak di Kampus Bukit (USDI), dan yang lain diletakkan di Kampus Sudirman (GDLN). Tujuan dari model/topologi ini adalah untuk mengurangi beban trafik pada link Sudirman – Bukit.
2. Pengembangan Kubernetes dan Load balancing untuk menyeimbangkan beban server
3. Integrasi SIMDOS dengan arsitektur microservices dengan IMISSU

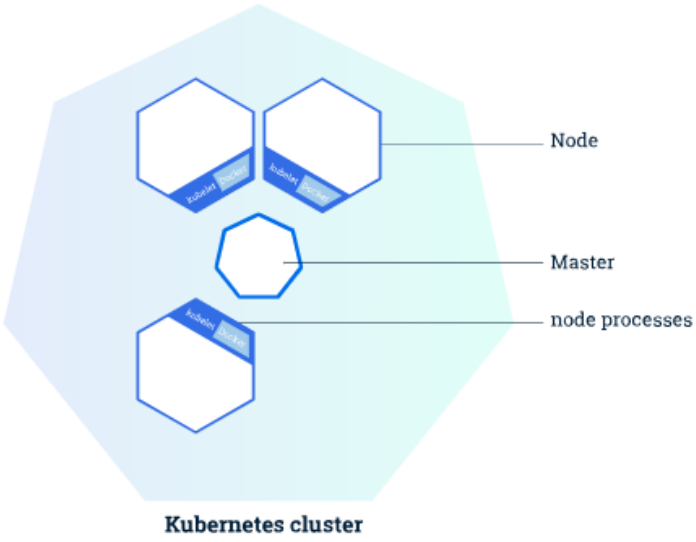
3.1 Pengembangan Kubernetes dan Load Balancing

Pemahaman tentang *container* diperlukan untuk memiliki pemahaman yang lebih baik tentang kubernetes. Sebelumnya aplikasi dideploy pada sistem host, yang mengarah ke keterikatan antara konfigurasi aplikasi, siklus hidup, dan dengan sistem operasi host. Container image adalah perangkat lunak yang berdiri sendiri yang memiliki semua yang diperlukan untuk operasinya. Mereka tidak digabungkan bersama dan memiliki sistem file mereka sendiri. Selain itu *container* tidak bergantung pada sistem operasi host, sehingga mereka dapat dengan mudah digunakan di berbagai server. Praktik umumnya adalah mengemas satu aplikasi ke dalam satu *container image*. Ini memberikan keuntungan besar

dalam kecepatan *deployment*. Dalam *continous delivery* (pengiriman kontinu), *container* membantu membuat build, release, rollbacks yang cepat.

Container adalah cara untuk mengemas aplikasi dan dapat digunakan di lingkungan yang berbeda. Masih terdapat pertanyaan bagaimana mengelola, *scaling*, dan *recovery*. Kubernetes mencoba mengatasi masalah ini. Kubernetes adalah sistem *open source* untuk pengelolaan aplikasi container yang didukung oleh Google.

Kubernetes Cluster adalah sekelompok node komputasi yang bertindak sebagai satu unit. Ini terdiri dari dua jenis node: master dan node. Master adalah unit manajemen, yang bertanggung jawab untuk penjadwalan, penskalaan, penghentian, pembaruan aplikasi. Node bertanggung jawab untuk menjalankan aplikasi. Gambar 3.4 menggambarkan cluster kubernetes.



Gambar 3.4 Kubernetes Cluster

Setiap *worker node* memiliki kubelet. Bertanggung jawab untuk komunikasi dengan master. Selain itu node memerlukan alat untuk membuat, menjalankan, menghapus aplikasi *container*. Bisa jadi dockr, rkt, LXD. Master mengekspos API untuk berkomunikasi dengan *worker node*.

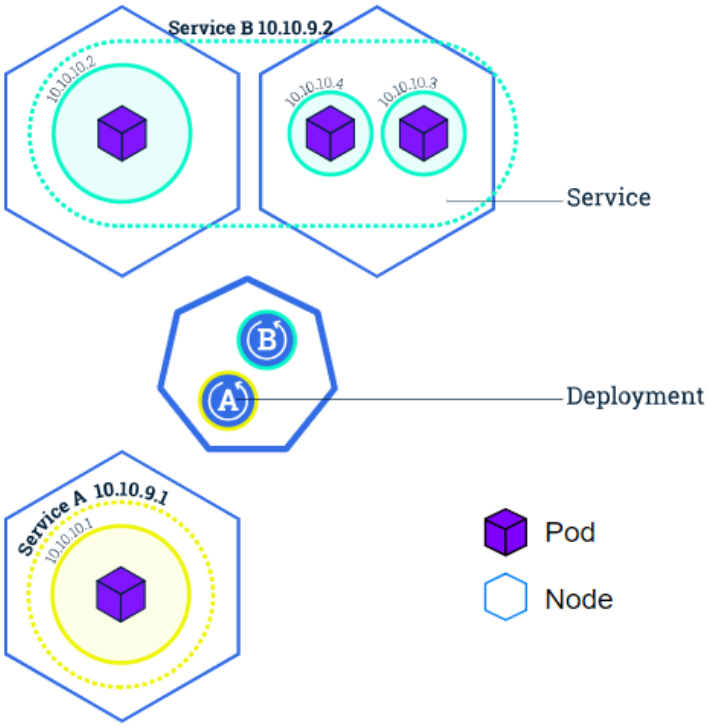
Seseorang dapat menyebarkan aplikasi container dalam cluster kubernetes menggunakan Deployment. Dalam kubernetes, konfigurasi *deployment*

menjelaskan bagaimana untuk memperbarui dan membuat aplikasi. Ketika seseorang membuat deployment Kubernetes menghasilkan Pod. Ini adalah mekanisme abstraksi untuk mewakili satu contoh aplikasi dan beberapa sumber daya yang terkait dengannya. Ketika Pod dibuat, ia berada pada beberapa node sampai diakhiri. Jika node tempat Pod berjalan down, maka pod baru akan dijadwalkan secara otomatis di node lain. Mungkin ada beberapa Pod yang berjalan pada node yang sama tergantung pada keterbatasan sumber daya node. Master bertanggung jawab untuk menjadwalkannya pada semua node yang tersedia. Sebuah *worker* bisa menjadi server fisik atau mesin virtual.

Abstraksi lain yang diperlukan untuk memahami kubernetes adalah *Service*. Ini adalah abstraksi yang mendefinisikan hubungan Pod yang terhubung secara logis dan cara mengaksesnya. *Service* memungkinkan akses ke aplikasi dengan berbagai cara, yang ditentukan oleh jenis *Service* [9]

- ClusterIP (default) – tipe ini membuat aplikasi hanya bisa diakses didalam cluster
- NodePort – Membuat *Service* dapat diakses diluar dari cluster
- LoadBalancer – Membuat load balancer dan memberikan IP tetap pada *Services*

Gambar 3.5 mengilustrasikan contoh cluster kubernetes. Dapat dilihat terdapat tiga node dan dua services. *Service* A hanya memiliki satu pod, sedangkan *Service* B memiliki tiga yang disatukan. Di tengah gambar terdapat master yang dilustrasikan memiliki informasi tentang *Service*.



Gambar 3.5 Contoh Kubernetes Cluster

BAB IV
BIAYA DAN JADWAL KEGIATAN

4.1 Biaya

Anggaran Biaya untuk tahun I dan II ditunjukkan pada Tabel 4.1

Tabel 4.1 Rencana Anggaran Biaya

No	Uraian	Biaya yang diusulkan Tahun II
1	Gaji dan Upah	35,300,000
2	Bahan habis pakai dan Peralatan Lainnya	36,000,000
3	Perjalanan	15,000,000
4	Lain-Jain	23,700,000
		110,000,000

4.2 Jadwal Kegiatan

Tabel 4.2 Jadwal Kegiatan

No	Jenis Kegiatan (Tahun 1)	Tahun 1											
		1	2	3	4	5	6	7	8	9	10	11	12
1	Persiapan, Study Literatur												
2	Penyusunan proposal												
3	Perencanaan Kubernetes												
4	Integrasi Kubernetes dan Load Balancer												
5	Pengembangan Microservice SIMDOSEN pada Kubernetes												
6	Coding												
7	Uji Coba												
8	Penyusunan Laporan dan Karya Ilmiah												

DAFTAR PUSTAKA

Alshuqayran, N., Ali, N., & Evans, R. (2016, November). A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)* (pp. 44-51). IEEE.

Balalaie, Armin., Heydarnoori, Abbas., Jamshidi, Pooyan.: Migrating to Cloud-Native Architectures Using Microservices: An Experience Migrating to Cloud-Native Architectures Using Microservices: An Experience Report. (July 2015).

Cerny, Tomas., Donahoo, Michael J., Trnka, Michal., Contextual understanding of microservice architecture: Current and Future Directions. APPLIED COMPUTING REVIEW (2018).

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. In *Present and Ulterior Software Engineering* (pp. 195-216). Springer, Cham.

Gannon, Dennis., Barga, Roger., Sundaresan, Neel., Cloud-Native Applications, IEEE Cloud Computing, (2017)

Granchelli, G., Cardarelli, M., Di Francesco, P., Malavolta, I., Iovino, L., & Di Salle, A. (2017, April). Towards recovering the software architecture of microservice-based systems. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)* (pp. 46-53). IEEE.

Jamshidi, Pooyan., Pahl, Claus., Mendonca, Nabor C., Lewis, James., Tilkov, Stefan., Microservices The Journey So Far and Challenges Ahead, IEEE Software, (2018).

Kalske, Miika., Transforming monolithic architecture towards microservice architecture. (2017)

Le, V. D., Neff, M. M., Stewart, R. V., Kelley, R., Fritzinger, E., Dascalu, S. M., & Harris, F. C. (2015, July). Microservice-based architecture for the nrdc. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)* (pp. 1659-1664). IEEE.

Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015, September). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *2015 10th Computing Colombian Conference (10CCC)* (pp. 583-590). IEEE.

LAMPIRAN

Lampiran 1. Justifikasi anggaran penelitian

1. Honor

Honor	Honor/ Jam (Rp)	Kuantitas	Biaya Satuan	Tahun II
Biaya pembuatan virtual server		0 paket	900,000	5,400,000
Biaya Instalasi Server Linux		0 paket	900,000	5,400,000
Biaya Konfigurasi Kubernetes		1 paket	900,000	1,500,000
Biaya Konfigursi Network serta firewall pada server linux		0 paket	900,000	5,400,000
Penambahan service web server, database server pada server linux		0 paket	900,000	5,400,000
Pengujian fungsi-fungsi server linux untuk kebutuhan microservice		6 paket	900,000	5,400,000
biaya setup Database Microservice		2 paket	900,000	1,800,000
Pengujian Microservices		6 paket	900,000	5,400,000
biaya Fileserver Microservice setup		2 paket	900,000	1,800,000
Biaya rekonfigurasi coding untuk microservice pada service database		1 paket	2,000,000	2,000,000
Biaya rekonfigurasi coding untuk microservice pada service fileserver		1 paket	2,000,000	2,000,000
Biaya pembuatan Aplikasi frontend microservice		1 paket	2,000,000	2,000,000
Biaya pembuatan Aplikasi backend microservice		1 paket	2,000,000	2,000,000
Biaya integrasi single sign on untuk model microservice		1 paket	2,000,000	2,000,000
Biaya Pengujian Kehandalan dan replikasi database server untuk model redundant		2 server	2,500,000	5,000,000
Biaya Pengujian Kehandalan dan replikasi fileserver untuk model redundant		2 server	2,000,000	5,000,000
Integrasi dan Pengujian kehandalan sistem keseluruhan		1 paket	2,000,000	4,000,000
			Sub Total (Rp)	**35,300,000**

2. Peralatan Penunjang

Material	Justifikasi pemakaian	Kuantitas	Harga Satuan	Tahun I
Pembelian Portable SSD Harddisk 1Tera	Untuk Penyimpanan Data Uji	2 buah	6,500,000	13,000,000

Pembelian Memory Server 32 GB	Untuk Penambahan Kemampuan Server	2 buah	9,000,000	18,000,000
			Sub Total (Rp)	**49,000,000**

3. Bahan Habis Pakai

Material	Justifikasi pemakaian	Kuantitas	Harga Satuan	Tahun I
Dokumentasi	Pembuatan dokumentasi hasil	1 paket	1,000,000	1,000,000
Internet	searching materi	8 bulan	500,000	4,000,000
Sub Total				**5,000,000**

4. Perjalanan

Material	Justifikasi pemakaian	Kuantitas	Biaya Satuan	Tahun I
Trasportasi Seminar Internasional	Transport PP	2 orang	3,000,000	6,000,000
IEEE Member	Biaya berlangganan IEEE untuk artikel pada jurnal dan conference	3 paket	3,000,000	9,000,000
			Sub Total (Rp)	**15,000,000**

Lain -lain

Kegiatan	Justifikasi pemakaian	Kuantitas	Biaya Satuan	Tahun I
Proposal	penggandaan dan Jilid	3 exp	100,000	300,000
Laporan kemajuan	penggadaan dan Jilid	3 exp	150,000	450,000
Laporan akhir	penggadaan dan Jilid	3 exp	150,000	450,000
Biaya Registrasi Seminar Internasional	Biaya Registrasi pengajuan paper di seminar	1 kali	1,000,000	1,000,000
Biaya Proofreading Artikel		1 paket	3,600,000	5,000,000
Biaya submit Artikel di Jurnal Internasional	Biaya submit Artikel di Jurnal Internasional	1 pakert	8,000,000	10,000,000
Biaya Konsumsi Rapat Tim	Selama 10 bulan	36 kali	275,000	9,900,000
			Sub Total (Rp)	**23,700,000**
			TOTAL ANGGARRAN YANG DIPERLUKAN TIAP TAHUN	**110,000,000**

Lampiran 2. Dukungan sarana dan prasarana penelitian

No	Sarana/Prasarana	Fungsi/Kegunaan	Jumlah
1	Laptop	Simulasi/Programming/Penyusunan Laporan	6 unit
2	LCD Projector	Presentasi	1 unit
3	Server	Penyimpanan data	1unit
4	Jaringan	Media komunikasi data	1 set

Lampiran 3. Susunan organisasi tim peneliti dan pembagian tugas

No	Nama/NIDN	Instansi Asal	Bidang Ilmu	Alokasi Waktu (jam/minggu)	Uraian Tugas
1	Dewa Made Wiharta	Unversitas Udayana	T. Elektro	8	Perancangan Model dan integrasi
2	Nyoman Putra Sastra	Unversitas Udayana	T. Elektro/ Jaringan Komputer	8	Mengembangkan Kubernetes dan Load Balancer
3	Komang Oka Saputra	Unversitas Udayana	T. Elektro/ Telekomunikasi	8	Perancangan Model

Lampiran 4. Biodata ketua dan anggota tim peneliti serta mahasiswa yang terlibat

BIODATA PENELITI

A. Biodata

1.	Nama Lengkap(dengan gelar)	Dr. Dewa Made Wiharta, S.T. M.T.
2.	Jabatan Fungsional	Lektor
3.	Jabatan Struktural	-
4.	NIP	197009221997021001
5.	NIDN	0022097003
6.	Tempat dan Tanggal Lahir	Denpasar, 22 September 1970
7.	Alamat Rumah	Jalan Nangka 186 Denpasar 80231 Bali
8.	Nomor Telepon/Faks /HP	081703440558
9.	Alamat Kantor	PS Teknik Elektro, FT UNUD Bukit Jimbaran
10.	Nomor Telepon/Faks	0361703315/0361703315
11.	Alamat e-mail	wiharta@unud.ac.id
12,	Lulusan yang telah dihasilkan	S-1= 35 orang; S-2= 2 Orang; S-3= 0 Orang
13	Mata Kuliah yg Diampu	1. Elektronika Telekomunikasi
		2. Telekomunikasi dan Jaringan Multimedia
		3. Pengolahan Citra Digital

B. Riwayat Pendidikan

	S-1	S-2	S-3
Nama Perguruan Tinggi	Institut Teknologi Sepuluh Nopember	Universitas Gadjah Mada	Institut Teknologi Sepuluh Nopember
Bidang Ilmu	Jurusan Teknik Elektro, Telekomunikasi	Teknik Elektro: Sistem Informasi Telekomunikasi	Teknik Elektro Telekomunikasi Multimedia
Tahun Masuk - Lulus	1993-1997	2000 - 2002	2008- 2016
Judul Tugas Akhir/Tesis	Studi Perbandingan Aplikasi DECT dan CDMA untuk Jaringan Lokal Akses Radio	Pengenalan Citra Wajah dengan Metode Eigenface	Penjejakan Obyek dalam Video dengan Filter Partikel
Nama Pembimbing	Ir. Hang Suharto Ir. Tonda P	Dr. Volker Muller, Dipl. Inf. Ir. F. Soesianto B.Sc.E, PhD	Prof. Ir. Gamantyo H., M.Eng., Ph.D. Dr. Ir. Wirawan, DEA

C. Pengalaman Penelitian dalam 5 Tahun Terakhir

No.	Tahun	Judul Penelitian	Pendanaan	
			Sumber	Jlm (Juta Rp)
1	2017	Pengembangan Metode Deteksi Dan Algoritma Penjejakan Objek Menggunakan Sensor Visual Untuk Humanoid Robot (Tahun 2)	Hibah Invensi	100
2	2017	Penataan dan Pemetaan Cell dalam Jaringan Selular dengan Teknologi 4G LTE di Kabupaten Badung	Penelitian Unggulan Udayana	40
3	2016	Metode Desiminasi Data Dari Jaringan Visual Sensor Nirkabel Dalam Era Internet Of Things	Hibah Bersaing Dikti	50
4	2016	Pengembangan Metode Deteksi Dan Algoritma Penjejakan Obyek Menggunakan Sensor Visual Untuk Humanoid Robot	Hibah Invensi	100
5	2013	Penjejakan Obyek dalam Kerangka Deterministik dan Probabilistik	Hibah Teknik Elektro	7,435
6	2012	Penjejakan Obyek Dengan Interpolasi Histogram Warna Dalam Filter Partikel	PDM	7,5

D. Pengalaman Pengabdian Kepada Masyarakat Dalam 5 Tahun Terakhir

No	Tahun	Judul Pengabdian	Pendanaan	
			Sumber	Jumlah (juta)
1	2017	Pemberdayaan Perempuan Kota Denpasar Melalui Pelatihan E-Commerce	Program Udayana Mengabdi	10
2	2016	Penyusunan Masterplan Pengembangan Teknologi Informasi Dan Komunikasi Di Kabupaten Badung	APBD Kabupaten Badung	228
3	2015	Pengkajian dan Penelitian Bidang Informasi dan Komunikasi berupa Penyusunan Kajian Teknis Terkait Pemanfaatan dan Penyelenggaraan Telekomunikasi di Kota Denpasar	APBD Kota Denpasar	100
4	2013	Kajian Penataan, Regulasi Dan Retribusi Menara Telekomunikasi Di Kota Denpasar	APBD Kota Denpasar	290
5	2012	Kajian Pendahuluan Penataan dan Penerapan Retribusi Menara Telekomunikasi di Kota Denpasar	APBD Kota Denpasar	50

6.	2012	Pembuatan SIM Rumah Tangga Miskin Kota Denpasar	APBD Kota Denpasar	350

E. Pengalaman Penulisan Artikel dalam Jurnal dalam 5 Tahun Terakhir

No.	Judul Artikel Ilmiah	Volume/No/Tahun	Nama Jurnal
1	On The Accuracy of Particle Filter-Based Object Tracking	Vol. 10, No. 11, Nopember 2015, ISSN: 19750080	International Journal of Multimedia and Ubiquitous Engineering
1	On The Accuracy of Particle Filter-Based Object Tracking	Vol. 10, No. 11, Nopember 2015, ISSN: 19750080	International Journal of Multimedia and Ubiquitous Engineering
2	Particle Filter-Based Object Tracking Using Joint Features of Color and Local Binary Pattern Histogram Fourier	Volume 8 No.4 Desember 2015	Jurnal Kursor Universitas Trunojoyo (Akreditasi B DIKTI)

F. Pengalaman Penyampaian Makalah Secara Oral Pada Pertemuan / Seminar Ilmiah Dalam 5 Tahun

No.	Nama Pertemuan Ilmiah	Judul Artikel Ilmiah	Waktu dan Tempat
1	Seminar Nasional Sains dan Teknologi, SENASTEK 2017	Pengembangan Metode Deteksi Dan Algoritma Penjejakan Obyek Menggunakan Sensor Visual Untuk Humanoid Robot (Tahun 2)	Bali, 2017
2	2016 International Conference on Smart-Green Technology in Electrical and Information Systems (ICSGTEIS 2016	Environmental Monitoring as an IoT Application in Building Smart Campus of Udayana University	Bali, 2016
3	Seminar Nasional Sains dan Teknologi, SENASTEK 2016	Pengembangan Metode Deteksi Dan Algoritma Penjejakan Obyek Menggunakan Sensor Visual Untuk Humanoid Robot	Bali, 2016
4.	IEEE – International Conference on Communications and Networking Application (ICCNA)	Tracking Fast Moving Object in Particle Filter Framework,	Bali, April 2011
5.	The 11th Seminar on Intelligent Technology and Its Applications,.	Color-Histogram Based Particle Filter for Tracking Object in Video,	Surabaya, 2010
6.	The 1st International Conference on Sustainable Technology Developmen (ICSTD)	Non-Linear Non-Gaussian State Estimation Using Particle Filter	Bali, Oktober 2010.

G. Karya Buku dalam 5 Tahun Terakhir

No.	Judul Buku	Tahun	Jumlah Halaman	Penerbit
1				

2				

H. Perolehan HKI dalam 10 Tahun Terakhir

No.	Judul/Thema HKI	Tahun	Jenis	No. P/ID
1				
2				

I. Pengalaman Merumuskan Kebijakan Publik/Rekayasa Sosial Lainnya dalam 10 Tahun Terakhir

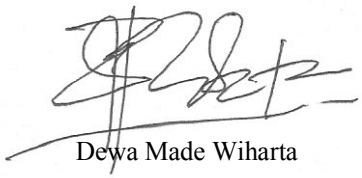
No.	Judul/Tema/Jenis Rekayasa Sosial Lainnya yang Telah Diterapkan	Tahun	Tempat
1	Penyusunan Masterplan Pengembangan Teknologi Informasi Dan Komunikasi Di Kabupaten Badung	2016	Kabupaten Badung
2			

J. Penghargaan dalam 10 tahun Terakhir (dari pemerintah, asosiasi atau institusi lainnya)

No.	Jenis Penghargaan	Institusi Pemberi Penghargaan	Tahun
1			
2			

Semua data yang saya isikan dan tercantum dalam biodata ini adalah benar dan dapat dipertanggungjawabkan secara hukum. Apabila di kemudian hari ternyata dijumpai ketidak-sesuaian dengan kenyataan, saya sanggup menerima risikonya. Demikian biodata ini saya buat dengan sebenarnya untuk memenuhi salah satu persyaratan dalam pengajuan Hibah Invensi Udayana Tahun 2019

Denpasar, 14 Februari 2019



Dewa Made Wiharta

CURRICULUM VITAE

A. Personal Identity

1	Name	Dr. Nyoman Putra Sastra, ST, MT	L/~~P~~
6	Place	Denpasar, 29 Agustus 1972	
7	Alamat Rumah	Jl. PB Sudirman FS 3 Denpasar – Bali	
8	Nomor Telepon/Faks/ HP	+62-361-242233/-/+62-8123836561	
9	Alamat Kantor	Jurusan Teknik Elektro – Universitas Udayana Jl. Kampus Bukit Jimbaran, Badung Bali	
10	Nomor Telepon/Faks	+62-361-703315	
11	Alamat E-mail	putra.sastra@unud.ac.id; putra.sastra@ieee.org	
12	Lulusan yang telah dihasilkan	35	
13	Mata Kuliah yg Diampu	Analisa Sinyal dan Sistem	
		Pengolahan Sinyal Digital	
		Computer Security	
		Sistem Operasi	
		Jaringan Sensor Nirkabel	

B. Riwayat Pendidikan

Program	S-1	S-2	S-3
Nama PT	Institut Teknologi Bandung, Bandung	Institut Teknologi Bandung, Bandung	Institut Teknologi Sepuluh Nopember
Bidang Ilmu	Jurusan Teknik Elektro, Telekomunikasi	Teknik Elektro, Sistem Informasi Telekomunikasi	Teknik Elektro, Telekomunikasi Multimedia
Tahun Masuk-Lulus	1992-1998	1998-2001	2008-2015
Judul Tugas Akhir /Tesis/Disertasi	Perencanaan dan Implementasi Layanan *Ring Back When Free* pada Sentral Gerbang Internasional Menggunakan DSP TMS32032 (Texas Instruments)	Unjuk Kerja Sistem Multi-Carrier CDMA pada *Multipath Fading Channel*	Jaringan Sensor Visual Nirkabel: P
Nama Pembimbing	Prof. Dr. Ir. Nana Rachmana Syambas, M.Eng. Dr. Ir. Ian Yoseph, M.T.	Dr. Ir. Sugihartono	Prof. Ir. Gamantyo Hendrantoro, M.Eng, Ph.D. Dr. Ir. Wirawan, DEA

C. Pengalaman Penelitian Dalam 5 Tahun Terakhir (Bukan Skripsi, Tesis, Maupun Disertasi)

No.	Tahun	Judul Penelitian	Pendanaan	
			Sumber	Jml (Juta Rp)
1.	2018	Single Sign On Instant Messaging sebagai Media Komunikasi di Lingkungan Universitas Udayana	Unggulan Udayana	40
2.	2018	Lampu Penerangan Jalan Umum Pintar (LPJU Smart)	Unggulan Udayana	40
3.	2017	Pengembangan Metode Deteksi Dan Algoritma Penjejakan Objek Menggunakan Sensor Visual Untuk Humanoid Robot: Prabu Udayana I	Invensi Udayana	100
4.	2017	Penataan dan Pemetaan Cell dalam Jaringan Selular dengan Teknologi 4G LTE di Kabupaten Badung	Unggulan Udayana	40
5.	2016	Identifikasi Kualitas Sinyal WLAN untuk Monitoring Pelaksanaan E-Exam pada Sistem E-learing Universitas Udayana	Unggulan Udayana	50
6.	2016	Pengembangan Metode Deteksi Dan Algoritma Penjejakan Objek Menggunakan Sensor Visual Untuk Humanoid Robot: Prabu Udayana I	Invensi Udayana	100
7.	2016	Diseminasi Informasi dari Jaringan Sensor Nirkabel	Hibah Bersaing	50
8.	2015	Pengembangan Sistem Transportasi Cerdas Kota Denpasar Berbasis Webgis	HUPS	25
9.	2015	Protokol Pemilihan Pasangan Lintasan untuk Kehandalan Komunikasi Kooperatif pada Jaringan Ad-Hoc	Hibah Bersaing	75
10.	2014	Protokol Pemilihan Pasangan Lintasan untuk Kehandalan Komunikasi Kooperatif pada Jaringan Ad-Hoc	Hibah Bersaing	75

D. Pengalaman Pengabdian Kepada Masyarakat

No.	Tahun	Judul Pengabdian Kepada Masyarakat	Pendanaan	
			Sumber	Jml (Juta Rp)
1.	2018	Pelatihan Pemilihan Diameter Penghantar untuk Memaksimalkan Penggunaan Energi Listrik dan Mencegah Kebakaran di Banjar Tingkih Kerep, Penebel - Tabanan	DIPA Unud	10
2.	2017	Pelatihan Pemilihan Kabel Listrik sesuai PUIL untuk Menghindari Risiko Kebakaran di Pasar Desa Sinduwati, Kecamatan Sidemen, Karangase	DIPA Unud	10

3	2017	Penerapan interactive e-quiz pada lomba asah terampil gapoktan budhi luhur desa katung	DIPA Unud	10
4	2016	Pengenalan pemrograman android bluetooth low energy (BLE) kepada siswa sma negeri 6 denpasar	DIPA Unud	10

E. Journal Article (5 tahun terakhir)

No.	Judul Artikel Ilmiah	Volume/ Nomor/Tahun	Nama Jurnal
1.	Implementasi E-Cerdas Cermat pada Lomba Asah Terampil Gapoktan Budhi Luhur	18/1/2019	Buletin Udayana Mengabdi
2	Token-based Single Sign-on with JWT as Information System Dashboard for Government	16/42018	Telkomnika
3	Perancangan Hardware Sistem Monitoring Portabel Untuk Monitoring Arus dan Tegangan Listrik Menggunakan Raspberry Pi	7/1/2018	Jurnal Sains dan Teknologi
4	Analisa Konsumsi Daya Sistem Pelacakan Posisi Muatan Roket Berbasis Arduino	5/2/2018	Jurnal Ilmiah Spektrum
5	Modification of ISONER Framework as Enterprise Service Bus to Build Consultation Robot Using External Engine	3/2/2018	International Journal of Engineering and Emerging Technology
6	Analisis Unjuk Kerja Pemantauan Jaringan OpenNMS (Open Network Monitoring System) pada Jaringan TCP/IP	5/2/2018	Jurnal Ilmiah Spektrum
7	Analisa kestabilan gerakan statis pada robot humanoid	5/2/2018	Jurnal Ilmiah Spektrum
8	Analisis jaringan wlan 802.11 g rumah sakit kapal kabupaten badung	5/2/2018	Jurnal Ilmiah Spektrum
9	Pengembangan Komunikasi Multikanal untuk Monitoring Infrastruktur Jaringan Berbasis BOT Telegram	5/2/2018	Jurnal Ilmiah Spektrum

10	Analisis Pemanfaatan Internet di Pusat Pemerintahan Kabupaten Badung	17/2/2018	Majalah Ilmiah Teknologi Elektro
11	Implementasi Algoritma Advanced Encryption Standard (AES) 128 Untuk Enkripsi dan Dekripsi File Dokumen	8/2/2018	Jurnal Eksplora Informatika
12	Efektivitas Pesan Teks dengan Cipher Substitusi, Vigenere Cipher, dan Cipher Transposisi	17/1/2018	Majalah Ilmiah Teknologi Elektro
13	Framework Pengelolaan Infrastruktur TIK di Pemerintah Kabupaten Badung	17/1/2018	Majalah Ilmiah Teknologi Elektro
14	Purwarupa Sistem Smart Traffic Light Pendukung Layanan Darurat Berbasis Teknologi RFID	16/3/2017	Majalah Ilmiah Teknologi Elektro
15	Perbandingan Performansi Pengamanan File Backup LPSE Menggunakan Algoritma DES Dan AES	15/2/2016	Majalah Ilmiah Teknologi Elektro
16	Computer Network Audit using Wireshark and Metasploit Framework (Case Study: STMIK STIKOM Bali Jimbaran Campus II)	1/1/2016	International Journal of Engineering and Emerging Technology
17	Cooperative Diversity Selection Protocol Using Pareto Method with Multi Objective Criterion in Wireless Ad Hoc Networks	11/5/2016	International Journal of Multimedia and Ubiquitous Engineering
18	Internet of Things for Intelligent Traffic Monitoring System: A Case Study in Denpasar	20/12/2015	International Journal of Computer Trends and Technology (IJCTT)
19	Energy Efficiency of Image Compression Implementation in Embedded Linux based Wireless Visual Sensor Network	31/9/2015	Journal of Communication Software and System (JCOMSS)
20	Energy Efficiency of **I**mage Compression for Virtual View Image over Wireless Visual Sensor Network	10/6/2015	Journal of Networks

21	Cooperative diversity paths selection protocol with multi-objective criterion in wireless Ad-Hoc networks	9/24/2014	International Journal of Applied Engineering Research, e-ISSN:1087-1090, ISSN: 0973-4592

F. Pengalaman Penyampaian Makalah Secara Oral Pada Pertemuan/Seminar Ilmiah (5 Tahun Terakhir

No.	Tahun	Judul Artikel Ilmiah	Nama Seminar
1.	2018	Introducing TAMEx Model for Availability of E-Exam in Wireless Environment	**International Conference on Information and Communications Technology (ICOIACT) 2018**
2.	2016	Proposed anonymous authentication scheme for academic service in the university	*ComnetSat 2016*
3.	2016	*Environmental Monitoring as an IoT Application in Buiding Smart Campus Universitas Udayana*	*ICSGTEIS -2016*
4.	2016	*Diseminiasi informasi dari Jaringan Sensor Nirkabel di Era Internet of Things*	*Senastek 2016*

G. Pengalaman Penulisan Buku dalam 5 Tahun Terakhir

No.	Judul Buku	Tahun	Jumlah Halaman	Penerbit
1				

H. Pengalaman Perolehan HKI Dalam 5 – 10 Tahun Terakhir

No.	Judul/Tema HKI	Tahun	Jenis	Nomor P/ID
1	Metode transmisi citra pada Jaringan Sensor Nirkabel	2014	Paten	P00201407239

I. Pengalaman Merumuskan Kebijakan Publik/Rekayasa Sosial/Kerjasama dan Lainnya Dalam 5 Tahun Terakhir

No.	Judul/Tema/Jenis Rekayasa Sosial Lainnya yang Telah Diterapkan	Tahun	Tempat Penerapan	Respons Masyarakat
1.	Penyusunan Rencana Induk Pengembangan Teknologi Informasi dan KOmunikasi Terpadu pemerintah Kota	2015	Pemerintah Kota Denpasar	Baik, karena dipakai acuan untuk sebagai perencanaan TIK di kota Denpasar

	Denpasar berupa Jasa Penyusunan Blue Print			
2	Pengkajian Dan Penelitian Bidang Informasi Dan Komunikasi Berupa Penyusunan Kajian Teknis Terkait Pemanfaatan Dan Penyelenggaraan Telekomunikasi Di Kota Denpasar Berupa Jasa Penyusunan Kajian Cell Plan	2015	Pemerimtah Kota Denpasar	Baik, akan mempermudah penataan dan pengeluaran izin pembangunan menara telekomunikasi di Kota Denpasr

J. Penghargaan yang Pernah Diraih dalam 10 tahun Terakhir (dari pemerintah, asosiasi atau institusi lainnya)

No.	Jenis Penghargaan	Institusi Pemberi Penghargaan	Tahun
1			

Semua data yang saya isikan dan tercantum dalam biodata ini adalah benar dan dapat dipertanggungjawabkan secara hukum. Apabila di kemudian hari ternyata dijumpai ketidak-sesuaian dengan kenyataan, saya sanggup menerima risikonya. Demikian biodata ini saya buat dengan sebenarnya untuk memenuhi salah satu persyaratan dalam pengajuan Hibah Penelitian Strategis nasional.

Denpasar, 7 Februari 2019

NYOMAN PUTRA SASTRA

CURRICULUM VITAE

Identitas Pribadi

1	Nama Lengkap (dengan gelar)	Komang Oka Saputra, ST, MT, PhD L/~~P~~
2	Jabatan Fungsional	Asisten ahli
3	Jabatan Struktural	-
4	NIP/NIK/Identitas lainnya	198104042008011009/5106040404810010
5	NIDN	0004048106
6	Tempat dan Tanggal Lahir	Kintamani, 4 April 1981
7	Alamat Rumah	Desa Katung Kintamani Bangli
8	Nomor Telepon	+628123660060
9	Alamat Kantor	Jurusan Teknik Elektro – Universitas Udayana Jl. Kampus Bukit Jimbaran, Badung Bali
10	Nomor Telepon/Faks	+62-361-703315
11	Alamat E-mail	okasaputra@unud.ac.id;
12	Lulusan yang telah dihasilkan	30
13	Mata Kuliah yg Diampu	Agen cerdas
		Bahasa Inggris
		Dasar Pemrograman Komputer
		Soft Computing
		Decission Suport System
		Telekomunikasi Ramah Lingkungan

B. Riwayat Pendidikan

Program	S-1	S-2	S-3
Nama PT	Universitas Brawijaya	Universitas Indonesia	National Taiwan University of Science and Technology
Bidang Ilmu	Jurusan Teknik Elektro, Telekomunikasi	Jurusan Teknik Elektro, Telekomunikasi	Computer Science and Information Engineering
Tahun Masuk-Lulus	1999-2004	2004-2006	2013-2016
Judul Tugas Akhir /Tesis/Disertasi	Jaringan Hybrid Fiber Coax (HFC) dengan Medium Access Control Protocol	Sequential Rotation Array untuk Meningkatkan Circular Polarization Bandwidth	Hough Transform-Based Clock Skew Measurement over Networks
Nama Pembimbing	Ir. Endah Budi Purnomowati MT	Prof. Eko Tjipto Rahardjo	Prof. Wei-Chung Teng

C. Pengalaman Penelitian Dalam 5 Tahun Terakhir

No.	Tahun	Judul Penelitian	Pendanaan	
			Sumber	Jml (Juta Rp)

1	2017	Analisis penerapan e-quiz dengan soal-soal non-formal pada pengetahuan umum mahasiswa Teknik elektro dan komputer	PNBP	25000000
2	2018	Clock skew measurement method for low time resolutions	PNBP	150000000
3	2018	Analisis penerapan web socket pada quiz berbentuk multiplayer game di system E-Cerdas Cermat	PNBP	40000000
4	2018	Model Pembelajaran Blended dengan Game Match the Box Untuk Mata Kuliah Teknologi Informasi	DIKTI	10000000

D. Pengalaman Pengabdian Kepada Masyarakat

No.	Tahun	Judul Pengabdian Kepada Masyarakat	Pendanaan	
			Sumber	Jml (Juta Rp)
1	2017	Penerapan interactive e-quiz pada lomba asah terampil gapoktan budhi luhur desa katung kintamani bangli	PNBP	10000000
2	2018	Implementasi system kesinoman berbasis android di ulu apad Desa Pakraman Katung	PNBP	10000000
3	2019	Implementasi mobile learning untuk peningkatan pemahaman tri hita karana di Sekaa Teruna Dharma Kanti Desa Katung	PNBP	10000000
4	2019	LENTERA	IEEE HAC	384000000

E. Pengalaman Penulisan Artikel Ilmiah Dalam Jurnal (5 tahun terakhir)

No.	Judul Artikel Ilmiah	Volume/ Nomor/Tahun	Nama Jurnal
1.	Hough Transform-Based Clock Skew Measurement Over Network	64/12/2015	IEEE Transactions on Instrumentation and Measurement
2.	Hough transform-based clock skew measurement by dynamically locating the region of offsets majority	E99-D/8/2016	IEICE Transactions on Information and Systems
3	Proposed Model of Multiplayer Matching Game Plugins Using Websocket in Moodle	14/11/2019	International Journal of Emerging Technologies in Learning (iJET)

F. Pengalaman Penyampaian Makalah Secara Oral Pada Pertemuan/Seminar Ilmiah (5 Tahun Terakhir

No.	Tahun	Judul Artikel Ilmiah	Nama Seminar
1	2014	A Study of Regular Transmission Delay in Bluetooth Communications	The 3rd International Conference on Intelligent Technologies and Engineering Systems (ICITES)
2	2015	A Clock Skew Replication Attack Detection Approach Utilizing the Resolution of System Time	International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)
3	2018	Clock skew measurement method for low time resolutions	SENASTEK
4	2018	Analisis penerapan web socket pada quiz berbentuk multiplayer game di system E-Cerdas Cermat	SENASTEK

Semua data yang saya isikan dan tercantum dalam biodata ini adalah benar dan dapat dipertanggungjawabkan secara hukum. Apabila di kemudian hari ternyata dijumpai ketidak-sesuaian dengan kenyataan, saya sanggup menerima risikonya. Demikian biodata ini saya buat dengan sebenarnya.

Denpasar, 20 Desember 2019

Komang Oka Saputra, ST, MT, PhD

Data Mahasiswa

Mahasiswa 1

1	**Nama Lengkap**	**I Wayan Adi Juliawan Pawana**
2	Tempat dan Tanggal Lahir	Tabanan, 10-07-1993
3	NIM	1881711010
4	Program Studi/Fakultas	Teknik Elektro/Teknik
5	Alamat	Jln. Gelogor Indah 1B GG Bisma No 17
6	No. Telepon	
7	e-mail	adijuliawanpawana@gmail.com

Mahasiswa 2

1	**Nama Lengkap**	**I Gusti Ayu Garnita Darmaputri**
2	Tempat dan Tanggal Lahir	Denpasar, 07-04-1995
3	NIM	1881711011
4	Program Studi/Fakultas	Teknik Elektro/Teknik
5	Alamat	Perumahan Puri Nusa Dua, Jln. Puri Nusa Dua III No. A19
6	No. Telepon	
7	e-mail	garnita.dp@gmail.com

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI UNIVERSITAS UDAYANA
LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT

Kampus Bukit Jimbaran. Telp. (Fax) (0361) 703367: 704622.
E-Mail:info-lppm@unud.ac.id Http://lppm.unud.ac.id

SURAT PERNYATAAN KETUA PENGUSUL

Yang bertanda tangan di bawah ini :

Nama Lengkap : Dr. Dewa Made Wiharta, S.T., M.T.

NIP/NIDN : 197009221997021001 / 0022097003

Pangkat / Golongan : Penata Tk.I / III/d

Jabatan Fungsional : Lektor

Program Studi/Fakultas : Teknik Elektro / Teknik

Dengan ini menyatakan bahwa proposal saya dengan judul:
Transformasi Arsitektur Monolithic Menuju Microservices untuk Implementasi Sistem Informasi Terintegrasiyang diusulkan dalam skema Hibah Penelitian Invensi Universitas Udayana anggaran 2020 dibuat secara bersama-sama oleh tim pengusul dan **bersifat original dan belum pernah dibiayai oleh lembaga/sumber dana lain.**

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia dituntut dan diproses sesuai dengan ketentuan yang berlaku dan mengembalikan seluruh biaya penugasan yang sudah diterima ke BLU.

Demikian Surat Pernyataan ini dibuat dengan sesungguhnya dan dengan sebenar-benarnya.

Denpasar, 5 Desember 2019

Mengetahui,
Ketua LPPM

(Prof. Dr. Ir. I Gede Rai Maya Temaja, MP)
NIP. 19621009 198803 1 002

Yang menyatakan,

METERAI TEMPEL
67F02AHF139098638
6000
ENAM RIBU RUPIAH

(Dr. Dewa Made Wiharta, S.T., M.T..)
NIP. 197009221997021001