Identifikasi Kandidat *Microservies* Dengan Analisis *Domain Driven Design*

I Wayan Adi Juliawan Pawana[1], Dewa Made Wiharta[2], Nyoman Putra Sastra[3]

[Submission: 04-06-2021, Accepted: 13-08-2021]

Abstract— The ongoing digital transformation requires organization not only to focus on business domains but also on their application landscapes. A more flexible architecture is need such as microservices architecture. Migration Lecturer Information System (SIMDOSEN) at Udayana University which was developed with monolithic architecture to microservices architecture raise a challenge of splitting business domain into distributed services. Domain Driven Design analysis is used to identify candidates for RESTful API-based microservices by splitting business capabilities. The results show a systematic and comprehensible process of developing microservice architecture. As well as having a positive impact on the advancement of IT architecture at Udayana University.

Keywords: Domain Driven Design, Microservices, Monolithic, RESTful API, SIMDOSEN.

***Intisari*— Transformasi digital yang sedang berlangsung menuntut organisasi tidak hanya berfokus pada domain bisnis, tetapi juga pada arsitektur aplikasi. Sehingga dibutuhkan arsitektur yang lebih fleksibel seperti arsitektur *microservices*. Migrasi Sistem Informasi Dosen (SIMDOSEN) pada Universitas Udayana yang dikembangkan dengan arsitektur *monolithic* menuju arsitektur *microservices* memunculkan tantangan untuk memisahkan domain bisnis menjadi layanan yang terdistribusi. Analisis *Domain Driven Design* digunakan untuk mengidentifikasi kandidat *microservices* berbasis RESTful API dengan memecah kemampuan bisnis. Hasil yang diperoleh menunjukkan proses pengembangan *microservices* yang sistematis dan dapat dipahami. Serta memberikan dampak positif untuk kemajuan arsitektur IT pada Universitas Udayana.**

Kata Kunci— Domain Driven Design*, *Microservices, Monolithic, RESTful API, SIMDOSEN.

I. PENDAHULUAN

Sistem informasi pada umumnya dibangun menggunakan arsitektur *monolithic* yang dikembangkan dalam satu entitas besar. Teknik ini membuat pengembangan awal menjadi sederhana dan mudah dimengerti [1].

Permasalahan pada arsitektur *monolithic* muncul ketika codebase mulai berkembang menjadi suatu aplikasi yang besar, baik dari sisi ukuran file maupun jumlah data yang terlibat. Masalah utama tersebut adalah ketika codebase membesar, pengembangan sistem menjadi lebih lambat pada saat terdapat perubahan kebutuhan sistem, serta kesulitan untuk mengelola codebase dan terbatasnya opsi untuk *scaling* aplikasi dan ketegantungan yang tinggi antar modul [1],[2],[3].

Solusi terhadap permasalahan yang muncul pada arsitektur monolithic adalah suatu arsitektur yang dikenal dengan nama microservices. Arsitektur *microservice* adalah *style* arsitektur terbaru yang telah berkembang pada beberapa tahun terakhir. Tidak ada definisi pasti mengenai arsitektur *microservice.* Namun dapat dikatakan bahwa *microservice* adalah koleksi dari beberapa *service* kecil yang otonom [4]. Tiap *service* memiliki tugas masing-masing yang memenuhi *single responsibility principle*. Tiap *service* mampu untuk didistribusikan secara independent pada platfom yang berbeda dan berjalan pada lingkungan tersendiri dengan saling berkomunikasi mengunakan mekanisme komunikasi ringan, contohnya RESTful API [6],[10]. Dengan arsitektur *microservice,* suatu sistem informasi akan memiliki beberapa *service* yang dikelola dan didistribusikan secara independent sehingga kemampuan sistem untuk beradapatasi terhadap perubahan kebutuhan akan semakin mudah [5],[8].

Implementasi arsitektur *microservice* akan menyebabkan ada banyak *services* sehingga membutuhkan suatu standar dan dokumentasi untuk mengelola *services* tersebut [9]. *The OpenAPI Specification* (OAS) mendefinisikan standar, antar-muka dengan bahasa agnostik pada RESTful API.

Pada pemrograman komputer, bahasa agnostik mempunyai pengertian bahwa program dapat menerima data dalam berbagai format, atau dari berbagai sumber, dan masih bisa memproses data tersebut secara efektif. Dengan demikian, manusia dan komputer akan bisa saling mengetahui dan mengerti kemampuan dari *service* tanpa harus mengakses *source code*, dokumentasi atau melakukan inspeksi trafik jaringan [7]. Ketika didefinsikan secara detail, konsumen dapat mengerti dan berinteraksi dengan *service* dengan kebutuhan implementasi yang minim.

SIMDOSEN adalah salah satu aplikasi yang digunakan oleh tenaga pendidik di Universitas Udayana untuk memanajemen data pribadi dosen, kegiatan tridharma perguruan tinggi, riwayat, sasaran kerja pegawai (SKP) dosen, dan remunerasi. SIMDOSEN pada pengembangan awal didesain menggunakan arsitektur *monolithic*, menggunakan framework Codeigniter dan belum menggunakan RESTful service. Dalam perjalanannya, ditemukan beberapa permasalahan pada SIMDOSEN, antara lain:

- Untuk menambah fitur baru pada SIMDOSEN, diperlukan perubahan skema database yang akan berdampak kepada modul lain. Walaupun sudah

[1]*Mahasiswa, Magister Teknik Elektro Universitas Udayana, Jalan Gelogor Indah 1B Gang Bisma No 17, Denpasar, 80221, Indonesia (tlp: 085-857162018; e-mail: adijuliawanpawana@unud.ac.id)*

[2,3] *Dosen, Jurusan Teknik Elektro dan Komputer Fakultas Teknik Universitas Udayana, Jln. Jalan Kampus Bukit Jimbaran 80361 INDONESIA (telp: 0361-703315; fax: 0361-4321; e-mail: wiharta@unud.ac.id, putra.sastra@unud.ac.id)*

didesain dengan baik di awal, tetapi keterikatan antar modul masih tetap tinggi.

- Tidak adanya RESTful API *service*, sehingga sistem lain yang memerlukan data pada SIMDOSEN, akan dilakukan koneksi langsung pada database SIMDOSEN dan menjalankan suatu *query*. Hal ini mengakibatkan programmer sistem lain menjadi kesulitan karena harus mempelajari struktur database SIMDOSEN. Kesulitan pengembangan lebih tinggi lagi dengan tidak adanya dokumentasi sehingga integrasi sistem menjadi proses yang rumit di lingkungan Universitas Udayana.
- Penggunaan teknologi yang tidak bisa bebas. Pada SIMDOSEN terdapat beberapa kebutuhan yang belum bisa diakomodasi secara penuh serta fitur yang belum optimal, seperti pencarian dokumen penelitian dosen yang lebih optimal ketika menggunakan database NoSQL.

Solusi terhadap permasalahan tersebut adalah melakukan migrasi menuju arsitektur microservices. Karena aplikasi SIMDOSEN merupakan suatu aplikasi yang besar, dengan melibatkan banyak fungsi/modul dan melibatkan banyak data, proses migrasi ini akan menjadi suatu pekerjaan yang besar, Pada penelitian ini, dilakukan bagian awal dari proses migrasi dengan melakukan analisis arsitektur *microservices* pada SIMDOSEN dengan menggunakan analisis *Domain Driven Design* (DDD) [19],[22]. Dengan analisis DDD, akan bisa dilakukan identifikasi pada *services* yang dibutuhkan dalam proses migrasi tersebut. Secara keseluruhan, tujuan dari penelitian ini adalah untuk dapat meningkatkan kemampuan adaptasi SIMDOSEN terhadap perkembangan kebutuhan di masa mendatang.

SIMDOSEN mengelola data seluruh dosen Universitas Udayana mulai dari manajemen riwayat, pelaporan beban kerja dosen, perhitungan remunerasi dan manajemen karya ilmiah dosen. Banyaknya tanggung jawab yang menjadi beban dari sistem ini menyebabkan sistem sulit beradaptasi dengan kebutuhan yang dinamis. Model arsitektur *microservices* diharapkan akan mampu memberi efisiensi dan fleksibilitas dalam melakukan adaptasi terhadap perubahan kebutuhan yang cepat.

II. TEORI PENUNJANG

A. Microservices

Arsitektur *Microservices* adalah suatu pendekatan dalam pengembangan sebuah aplikasi sebagai suatu rangkaian *service* kecil, setiap service berjalan dengan proses sendiri dan berkomunikasi dengan suatu mekanisme ringan seperti HTTP API [22],[26]. *Service* tersebut dibuat berdasarkan kemampuan bisnis dan dapat diterapkan secara independen. *Service* dapat dibuat dengan bahasa pemrograman yang berbeda dan dapat menggunakan teknologi penyimpanan data yang berbeda [8].

Tujuan *microservice*s adalah untuk menghasilkan unit-unit otonom yang terisolasi satu sama lain dan mengkoordinasikannya ke dalam suatu infrastruktur terdistribusi dengan teknologi container seperti Docker. Umumnya, dengan mengadopsi model arsitektur ini akan berimplikasi pada menggunakan praktek *agile* seperti DevOps [21],[25] yang mengurangi waktu untuk mengimplementasi perubahan dari lingkungan *development* ke lingkungan *production* berbasis cloud. Dengan menggunakan teknologi *cloud computing* maka penggunaan *resource* CPU dan *memory server* akan lebih optimal [27].

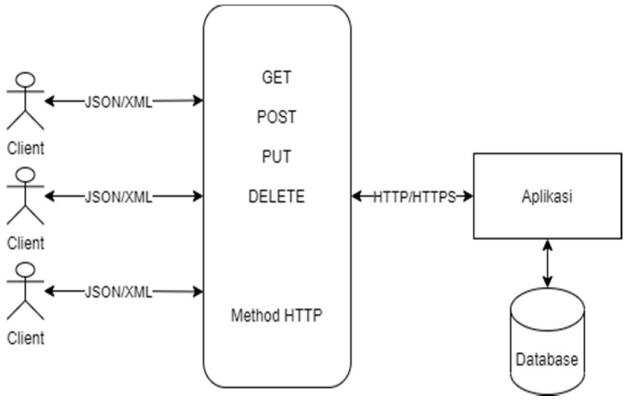
B. Domain Driven Design

Domain Driven Design adalah suatu pendekatan pengembangan software yang berpusat pada pengembangan model domain yang memiliki pemahaman mengenai proses dan aturan dari suatu domain [23]. Sebuah model bertindak sebagai *Ubiquitous Language* untuk membantu komunikasi antara pengembang software dan ahli domain yang melakukan pemetaan domain. Model juga bertindak sebagai landasan konseptual untuk desain *software* itu sendiri, tentang bagaimana *software* dipecah menjadi object atau fungsi. Klasfikasi aktifitas model yang dihasilkan dari *Domain Driven Design* ke dalam proses pengembangan *software* dapat meningkatkan *applicability* [18].

Bounded Context (BC) adalah pola sentral dalam *Domain-Driven Design*. BC adalah fokus dari bagian desain strategis DDD yang berhubungan dengan model dan tim pengembang yang besar. DDD berurusan dengan model-mocel yang besar dengan membaginya ke dalam BC yang berbeda dan menyatakan secara eksplisit tentang keterkaitan mereka [24].

C. RESTful API

RESTful API atau bisa disebut dengan REST API merupakan singkatan dari *Representational State Transfer,* merupakan abstraksi dari suatu arsitektur untuk sebuah *Application Program Interface* (API) yang menggunakan protokol HTTP untuk komunikasi data (antara sistem hypermedia terdistribusi), menggunakan konsep *resource* dengan setiap komponennya dianggap sebagai sebuah *resource* dan setiap *resource* diakses menggunakan HTTP *Method* seperti GET, POST, PUT, DELETE [11],[12].



Gambar 1: Arsitektur RESTful API

RESTful API adalah suatu *style* arsitektur dan bukan suatu bahasa pemrograman atau teknologi. Ini memberikan suatu panduan pada sistem terdistribusi untuk berkomunikasi secara

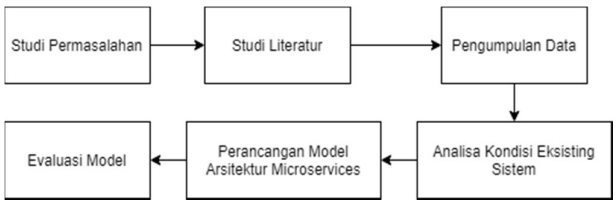
langsung menggunakan protokol web yang sudah ada untuk membuat suatu *web services* dan API, tanpa harus menggunakan protokol yang rumit [12]. Dari perspektif bisnis, RESTful API dapat dilihat sebagai aset berharga [14] yang dapat juga menjadi solusi transformasi digital [15].

Arsitektur RESTful API menyediakan akses ke *resource* sehingga *client* mengakses dan menampilkan hasil *resource* ke sisi *client*. RESTful API mengunakan beberapa format representasi untuk menampilkan hasil seperti XML, JSON, teks, dan gambar. Arsitektur umum RESTful API ditunjukkan pada Gambar 1.

III. METODE PENELITIAN

Susunan tahapan penelitiaan ditunjukkan pada Gambar 2, diawali dengan studi permasalahan, yaitu pada Sistem Informasi Dosen (SIMDOSEN) Universitas Udayana yang saat ini sedang berjalan. Tahapan berikutnya adalah studi literatur yang didapatkan dari berbagai sumber terkait arsitektur *microservices*, proses analisis *microservice*, RESTful API, dan lain-lain.

Tahapan pengumpulan data dilakukan dengan mencari data pada Sistem Informasi Dosen, petunjuk teknis, pentunjuk penggunaan, dan lain-lain. Tahapan berikutnya adalah analisis kondisi eksisting, yaitu analisis model arsitektur *monolithic* sistem informasi dosen yang sedang berjalan saat ini. Hasil analisis ini digunakan sebagai dasar untuk melakukan tahapan berikutnya, yaitu perancangan model arsitektur *microservices.* Tahapan terakhir adalah melakukan evaluasi hasil rancangan model arsitektur *microservices.*



Gambar 2: Metodologi Penelitian

IV. HASIL IDENTIFIKASI KANDIDAT MICROSERVICES DENGAN ANALISIS DDD

A. Proses Decoupling Microservice (DDD)

Proses migrasi sistem dari arsitektur *monolithic* menjadi arsitektur *microservices* bukan suatu pekerjaan biasa. Pada proses migrasi ini dilakukan pendekatan iteratif untuk memastikan kestabilan dari sistem. Hal ini dilakukan karena pada prosesnya dimungkinan muncul masalah seperti identifikasi yang tidak sesuai atau pemisahan tanggung jawab yang kurang tepat. Proses analisis sistem arsitektur *monolithic* dilakukan dengan metode *Domain-Driven Design* (DDD) [22] untuk mengekstrak kandidat *microserices* dari sistem asal. Tahap selanjutnya adalah mendesain API bedasarkan hasil kandidat *microservices* untuk kemudian disusun dokumentasi API menggunakan spesifikasi *OpenAPI Specification 3.*

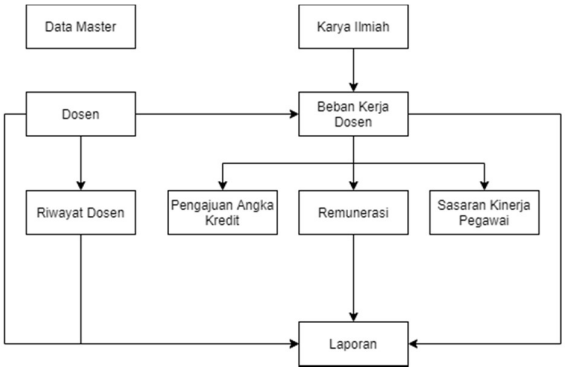
B. Analisis sistem untuk identifikasi kandidat microservices

Metode analisis yang digunakan untuk melakukan migrasi arsitektur *monolithic* ke arsitektur *microservices* adalah *Doman Driven Design*. *Bounded Context* (BC) dari hasil analisis menghasilkan kandidat untuk memecah *services*. Tidak ada acuan bagaimana memecah *service*, bisa berbeda-beda bergantung pada kebutuhan dari masing masing sistem, ukuran tim pengembang dan permintaan dari stakeholder [20]. Karakteristik dari DDD adalah sebagai berikut.

1) setiap project untuk suatu domain tertentu.
2) sistem yang kompleks terdiri dari kumpulan domain module.
3) masalah domain dianalisis oleh pengembang bersama dengan tim ahli.

Setiap *microservices* didesain sebagai komponen yang saling independent, dan hasil analisis dari DDD digunakan sebagai salah satu cara untuk mengekstrak *service* dengan domain spesifik.

Analisis berikutnya adalah analisis struktur database sistem *monolithic*. Ketika mengembangkan sistem *monolithic* maka akan ada dependensi tinggi (*thightly coupling*) antar modul. Pada arsitektur *microservices* tiap *service* memiliki database tersendiri yang terpisah antar *service* lain. Untuk tabel yang digunakan bersama akan dibuatkan *service* master untuk mengurangi dependensi dan keterikatan antar *service*. Hasil analisis domain model DDD ditunjukkan pada Gambar 3.



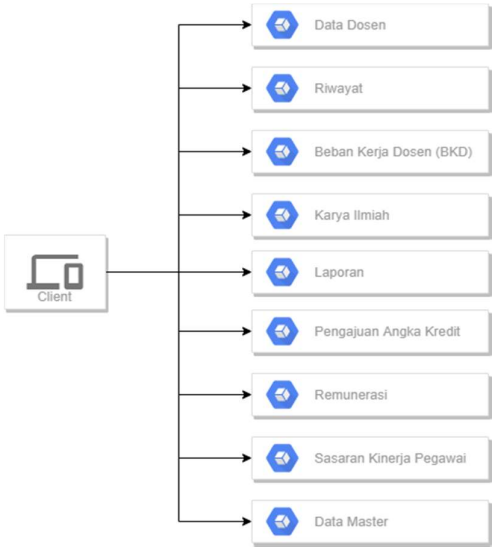
Gambar 3: Domain Model SIMDOSEN

Pada kasus SIMDOSEN, dengan analisis DDD didapatkan kandidat *service* sebagai berikut: *Service* Data Dosen, *Service* Riwayat, *Service* BKD, *Service* Karya Ilmiah, *Service* Laporan, *Service* Pengajuan Angka Kredit (PAK), *Service* remunerasi, *Service* SKP, *Service* Data Master seperti ditunjukkan pada Gambar 4.

Service Data Dosen menyediakan data dosen dari masing masing pribadi dosen. *Service* ini juga memanggil *service* riwayat untuk mendapatkan data riwayat terakhir dosen. *Service* Riwayat menyediakan data riwayat dari dosen di lingkungan Universitas Udayana. Data riwayat yang disediakan adalah riwayat pendidikan, riwayat fungsional, riwayat jabatan, riwayat tugas tambahan, riwayat keaktifan, dan riwayat penugasan. *Service* BKD adalah layanan untuk dosen

menginputkan dan melaporkan kegiatan tridarma perguruan tinggi seperti pendidikan, penelitian, pengabdian dan penunjang. *Service* karya ilmiah adalah layanan untuk dosen menginputkan penelitian yang pernah dikerjakan selama menjadi dosen. *Service* Pengajuan Angka Kredit adalah layanan untuk mengajukan kenaikan pangkat, dalam layanan ini dosen dapat menginputkan data kenaikan pangkat undur pendidikan, penelitian, pengabdian, dan penunjang. *Service* Remunerasi adalah layanan untuk menghitung dan menampilkan kinerja dari dosen. *Service* SKP adalah layanan untuk mengisi target dan realisasi sasaran kinerja pegawai. *Service* laporan adalah layanan untuk melihat laporan keseluruhan di sistem simdosen seperti laporan data dosen, laporan statistik BKD dan laporan statistik remunerasi. *Service* data master adalah layanan data master yang digunakan sebagai data master yang banyak diakses oleh banyak *service* seperti master tahun ajar, master unit, master sub unit dan master kegiatan BKD.

Uraian di atas secara umum digunakan untuk memetakan kandidat *microservice* yang ditunjukkan pada Gambar 4 dan Tabel 1.



Gambar 4: Kandidat Service SIMDOSEN

TABEL I
MODULE DAN SERVICE PADA SIMDOSEN

No	Modul	*Services*
1	Data Dosen	Profil Dosen
2	Riwayat	Riwayat Pendidikan
		Riwayat Fungsional
		Riwayat Jabatan
		Riwayat Tugas Tambahan
		Riwayat Keaktifan
3	BKD	BKD Pendidikan
		BKD Penelitian
		BKD Pengabdan
		BKD Penunjang
		Pengajuan BKD
4	Karya Ilmiah	Karya Ilmiah
5	Laporan	Laporan Data Dosen
		Laporan Statistik BKD
		Laporan Statistik Remunerasi
6	Pengajuan Angka Kredit (PAK)	PAK Pendidikan
		PAK Penelitian
		PAK Pengabdian
		PAK Penunjang
7	Remunerasi	Remunerasi Gaji
		Remunerasi Kinerja
		Perhitungan Remunerasi Gaji
		Perhitungan Remunerasi Kinerja
8	SKP	Target SKP
		Realisasi SKP
9	Data Master	Master Tahun Ajar
		Master Unit
		Master Sub Unit
		Master Kegiatan BKD

Dengan melakukan analisis kebutuhan masing masing *microservices*, membaca *source code* sistem *monolithic* SIMDOSEN dan melakukan pengelompokan kandidat *resource* bedasarkan [16] maka dapat diidentifikasi kebutuhan API yang ditunjukkan pada Tabel II sampai Tabel X.

TABEL II
RANCANGAN MICROSERVICES DATA DOSEN

No	URI	Method HTTP
1	/api/v1/dosen	GET, POST, PUT, DELETE
2	/api/v1/dosen/{dosenId}	GET, PUT, DELETE

TABEL III
RANCANGAN MICROSERVICES RIWAYAT

No	URI	Method HTTP
1	/api/v1/riwayat-pendidikan	GET, POST, PUT, DELETE
2	/api/v1/riwayat-pendidikan/{riwayatPendidikanId}	GET, PUT, DELETE
3	/api/v1/riwayat-pendidikan/dosen{dosenId}	GET
4	/api/v1/riwayat-pendidikan/dosen{dosenId}/latest	GET
5	/api/v1/riwayat-fungsional	GET, POST, PUT, DELETE
6	/api/v1/riwayat-fungsional/{riwayatFungsionalId}	GET, PUT, DELETE
7	/api/v1/riwayat-fungsional/dosen{dosenId}	GET
8	/api/v1/riwayat-fungsional/dosen{dosenId}/latest	GET
9	/api/v1/riwayat-kepangkatan	GET, POST, PUT, DELETE
10	/api/v1/riwayat-kepangkatan/{riwayatKepangkatanId}	GET, PUT, DELETE
11	/api/v1/riwayat-kepangkatan/dosen{dosenId}	GET
12	/api/v1/riwayat-kepangkatan/dosen{dosenId}/latest	GET

13	/api/v1/riwayat-keaktifan	GET, POST, PUT, DELETE
14	/api/v1/riwayat-keaktifan/{riwayatKeaktifanId}	GET, PUT, DELETE
15	/api/v1/riwayat-keaktifan/dosen{dosenId}	GET
16	/api/v1/riwayat-keaktifan/dosen{dosenId}/latest	GET
17	/api/v1/riwayat-tugas-tambahan	GET, POST, PUT, DELETE
18	/api/v1/riwayat-tugas-tambahan/{riwayatTugasTambahanId}	GET, PUT, DELETE
19	/api/v1/riwayat-tugas-tambahan/dosen{dosenId}	GET
20	/api/v1/riwayat-tugas-tambahan/dosen{dosenId}/latest	GET
21	/api/v1/riwayat-penugasan	GET, POST, PUT, DELETE
22	/api/v1/riwayat-penugasan/{riwayatPenugasanId}	GET, PUT, DELETE
23	/api/v1/riwayat-penugasan/dosen{dosenId}	GET
24	/api/v1/riwayat-penugasan/dosen{dosenId}/latest	GET

TABEL IV
RANCANGAN MICROSERVICES BEBAN KERJA DOSEN

No	URI	Method HTTP
1	/api/v1/bkd/identitas/dosen/ {dosenId} / tahun-ajar/ {tahunAjar}	GET, POST, PUT, DELETE
2	/api/v1/bkd/pendidikan/dosen/ {dosenId} / tahun-ajar/ {tahunAjar}	GET, POST, PUT, DELETE
3	/api/v1/bkd/pendidikan/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / id/ {bkdPendidikanId}	GET, PUT, DELETE
4	/api/v1/bkd/pendidikan/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / terpakai	GET
5	/api/v1/bkd/pendidikan/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / tidak-terpakai	GET
6	/api/v1/bkd/penelitian/dosen/ {dosenId} / tahun-ajar/ {tahunAjar}	GET, POST, PUT, DELETE
7	/api/v1/bkd/penelitian/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / id/ {bkdPenelitianId}	GET, PUT, DELETE
8	/api/v1/bkd/penelitian/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / terpakai	GET
9	/api/v1/bkd/penelitian/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / tidak-terpakai	GET
10	/api/v1/bkd/pengabdian/dosen/ {dosenId} / tahun-ajar/ {tahunAjar}	GET, POST, PUT, DELETE
11	/api/v1/bkd/pengabdian/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / id/ {bkdPengabdianId}	GET, PUT, DELETE
12	/api/v1/bkd/pengabdian/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / terpakai	GET
13	/api/v1/bkd/pengabdian/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / tidak-terpakai	GET
14	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar}	GET, POST, PUT, DELETE
15	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / id/ {bkdPendidikanId}	GET, PUT, DELETE
16	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / terpakai	GET
17	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / tidak-terpakai	GET
18	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / kesimpulan	GET, POST, PUT, DELETE
19	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / kesimpulan/pendidikan	GET
20	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / kesimpulan/penelitian	GET
21	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / kesimpulan/pengabdian	GET
22	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / kesimpulan/penunjang	GET
23	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / kesimpulan/lebihan/pendidikan	GET
24	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / kesimpulan/lebihan/penelitian	GET
25	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / kesimpulan/lebihan/pengabdian	GET
26	/api/v1/bkd/penunjang/dosen/ {dosenId} / tahun-ajar/ {tahunAjar} / kesimpulan/lebihan/penunjang	GET

TABEL V
RANCANGAN MICROSERVICES KARYA ILMIAH

No	URI	Method HTTP
1	/api/v1/karya-ilmiah/dosen/ {dosenId} /tahun-ajar/ {tahunAjar}	GET, POST, PUT, DELETE
2	/api/v1/karya-ilmiah/dosen/ {dosenId} /tahun-ajar/ {tahunAjar} / id / {karyaIlmiahId}	GET, PUT, DELETE

TABEL VI
RANCANGAN MICROSERVICES LAPORAN

No	URI	Method HTTP
3	/api/v1/laporan/bkd/ tahun-ajar / {tahunAjar}	GET

4	/api/v1/laporan/dosen/ tahun-ajar / {tahunAjar}	GET
5	/api/v1/laporan/remunerasi-gaji/ tahun-ajar / {tahunAjar}	GET
6	/api/v1/laporan/remunerasi-kinerja/ tahun-ajar / {tahunAjar}	GET

TABEL VII
RANCANGAN MICROSERVICES REMUNERASI

No	URI	Method HTTP
1	/api/v1/remunerasi/remunerasi-gaji/ tahun-ajar / {tahunAjar} / bulan / {bulan}	GET
2	/api/v1/remunerasi/remunerasi-gaji/ tahun-ajar / {tahunAjar} / bulan / {bulan} / hitung	POST
3	/api/v1/remunerasi/remunerasi-gaji/ tahun-ajar / {tahunAjar} / dosen / {dosen} / bulan / {bulan}	GET
4	/api/v1/remunerasi/remunerasi-kinerja/ tahun-ajar / {tahunAjar}	GET
5	/api/v1/remunerasi/remunerasi-kinerja/ tahun-ajar / {tahunAjar} / hitung	POST
6	/api/v1/remunerasi/remunerasi-kinerja/ tahun-ajar / {tahunAjar} / dosen / {dosenId}	GET
7	/api/v1/remunerasi/remunerasi-kinerja/ tahun-ajar / {tahunAjar} / dosen / {dosenId} / kegiatan-remun	GET
8	/api/v1/remunerasi/remunerasi-kinerja/ tahun-ajar / {tahunAjar} / dosen / {dosenId} / kegiatan-nonremun	GET
9	/api/v1/remunerasi/insentif-penelitian-spi/ tahun-ajar / {tahunAjar}	GET
10	/api/v1/remunerasi/insentif-penelitian-spi/ tahun-ajar / {tahunAjar} / hitung	POST
11	/api/v1/remunerasi/insentif-penelitian-spi/ tahun-ajar / {tahunAjar} / dosen / {dosenId}	GET
12	/api/v1/remunerasi/insentif-penelitian-spi/ tahun-ajar / {tahunAjar} / dosen / {dosenId} / kegiatan	GET

TABEL VIII
RANCANGAN MICROSERVICES DATA MASTER

No	URI	Method HTTP
1	/api/v1/master/tahun-ajar	GET, POST, PUT, DELETE
2	/api/v1/master/tahun-ajar/ {tahunAjar}	GET, PUT, DELETE
3	/api/v1/master/kegiatan	GET, POST, PUT, DELETE
4	/api/v1/master/kegiatan/ {kegiatanId}	GET, PUT, DELETE
5	/api/v1/master/unit	GET, POST, PUT, DELETE
6	/api/v1/master/unit/ {unitId}	GET, PUT, DELETE
7	/api/v1/master/sunit	GET, POST, PUT, DELETE
8	/api/v1/master/sunit/ {sunitId}	GET, PUT, DELETE

TABEL IX
RANCANGAN MICROSERVICES PENGAJUAN ANGKA KREDIT (PAK)

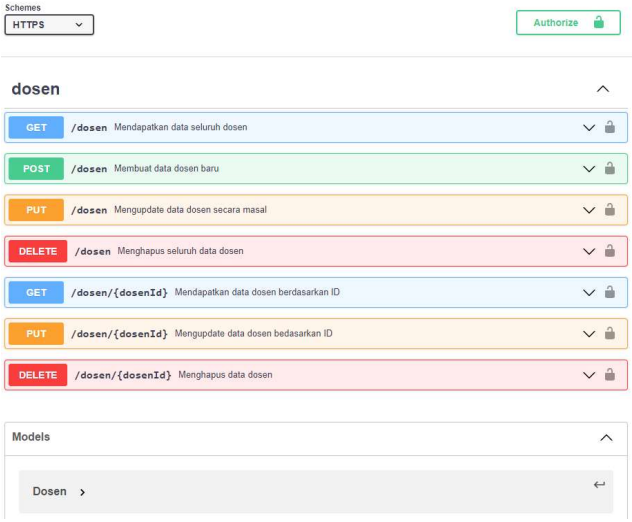
No	URI	Method HTTP
1	/api/v1/pak/pendidikan/dosen/ {dosenId} / tahun-ajar / {tahunAjar}	GET, POST, PUT, DELETE
2	/api/v1/pak/pendidikan/dosen/ {dosenId} / tahun-ajar / {tahunAjar} / id / {pakPendidikanId}	GET, PUT, DELETE
3	/api/v1/pak/penelitian/dosen/ {dosenId} / tahun-ajar / {tahunAjar}	GET, POST, PUT, DELETE
4	/api/v1/pak/penelitian/dosen/ {dosenId} / tahun-ajar / {tahunAjar} / id / {pakPenelitianId}	GET, PUT, DELETE
5	/api/v1/pak/pengabdian/dosen/ {dosenId} / tahun-ajar / {tahunAjar}	GET, POST, PUT, DELETE
6	/api/v1/pak/pengabdian/dosen/ {dosenId} / tahun-ajar / {tahunAjar} / id / {pakPengabdianId}	GET, PUT, DELETE
7	/api/v1/pak/penunjang/dosen/ {dosenId} / tahun-ajar / {tahunAjar}	GET, POST, PUT, DELETE
8	/api/v1/pak/penunjang/dosen/ {dosenId} / tahun-ajar / {tahunAjar} / id / {pakPenunjangId}	GET, PUT, DELETE
9	/api/v1/pak/ dosen / {dosenId} / tahun-ajar / {tahunAjar} / laporan	GET, POST, PUT, DELETE

TABEL X
RANCANGAN MICROSERVICES SKP

No	URI	Method HTTP
1	/api/v1/skp/ dosen / {dosenId} / tahun-ajar/ {tahunAjar} /target	GET, POST, PUT, DELETE
2	/api/v1/skp/ dosen / {dosenId} / tahun-ajar/ {tahunAjar} /realisasi	GET, POST, PUT, DELETE

Struktur *Uniform Resource Identifier* (URI) pada rancangan API SIMDOSEN mengikuti kesepakatan [17] untuk meningkatkan kemampuan *discoverability* dan memastikan konsistensi dalam *microservices*.

Setelah melakukan identifikasi, dibuat dokumentasi API berdasarkan *OpenAPI Specification*. Hal ini bertujuan untuk membuat suatu dokumentasi yang lengkap, terpadu, dan jelas agar tidak terjadi perbedaan interpretasi antara pengembang dalam hal ini desainer, developer, tester dan devops. Selain itu pengembangan akan lebih cepat karena pengembang atau komputer dapat mengerti kemampuan dari suatu *service* tanpa harus mengakses *source code* langsung. Gambar 5 menujukkan hasil dari dokumentasi API mengunakan *OpenAPI Specification* untuk *service* dosen.



Gambar 5: Dokumentasi microservice dosen dengan OpenAPI 3 Spesification

V. KESIMPULAN DAN SARAN

Domain driven design memberikan konsep dan langkah untuk membangun aplikasi berbasis arsitektur *microservices*. *Domain driven design* berfokus pada domain, termasuk konsep, hubungan antar domain dan *business logic*. Arsitektur *microservices* adalah tentang bagaimana mengatur dan membagi perangkat lunak menjadi blok bangunan kecil yang terdistribusi.

Hasil penelitian ini menunjukkan, dengan pendekatan *Domain Driven Design*, proses migrasi arsitektur *monolithic* menjadi arsitektur *microservice* berhasil dilakukan dan menghasilkan 29 *microservice* dan 89 API. Pada arsitektur *microservice* setiap *microservice* memiliki database tersendiri untuk memastikan perubahan suatu *microservice* tidak berdampak pada *microservice* lain.

Keuntungan dari menggunakan DDD dan *microservices* adalah kemampuan untuk menggunakan kembali (*reuse*) fungsi yang sudah ada. Informasi hasil identifikasi model dan fungsionalitas domain dapat digunakan oleh aplikasi lain di lingkungan Universitas Udayana.

Adanya RESTful API dengan standar *OpenAPI Specification 3*, menyebabkan integrasi dengan pengembang lain akan menjadi lebih mudah karena pengembang lain dapat melihat dokumentasi API yang disediakan, dan dapat melihat contoh *response* yang diberikan tanpa perlu mengetahui *source code* dan struktur database dari SIMDOSEN. Hal ini akan mengurangi terjadinya perbedaan pemahaman *service* tim pengembang.

Harapan dari penelitian ini adalah memberikan insight kepada pengembang lain yang melakukan migrasi aplikasi *monolithic* menjadi arsitektur *microservice*. Rencana penelitian ke depan dari hasil ini adalah untuk melakukan pengembangan *cloud-native* di Universitas Udayana.

REFERENSI

[1] J. Kazanavičius and D. Mažeika, "*Migrating Legacy Software to Microservices Architecture*," 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), pp. 1-5, 2019.
[2] S. Baškarada, V. Nguyen, A. Koronios. "*Architecting Microservices: Practical Opportunities and Challenges*." Journal of Computer Information Systems, pp. 1–9, 2018
[3] R. Chen, S. Li and Z. Li, "*From Monolith to Microservices: A Dataflow-Driven Approach*," 24th Asia-Pacific Software Engineering Conference (APSEC), pp. 466-475, 2017.
[4] Martin Fowler (2014) *Microservices*. [Online]. Available: https://martinfowler.com/articles/microservices.html
[5] B. Butzin, F. Golatowski and D. Timmermann, "*Microservices approach for the internet of things*," IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-6, 2016.
[6] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustan, L. Sana, "*Microservices: yesterday, today, and tomorrow*", Present and Ulterior Software Engineering, pp 195-216, 2017.
[7] (2021) *OpenAPI Specification*. [Online]. Available: https://swagger.io/specification/
[8] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. 1st ed. O'Reilly Media, 2015.
[9] S. Karlsson, A. Čaušević and D. Sundmark, "*QuickREST: Property-based Test Generation of OpenAPI-Described RESTful APIs*," IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), pp. 131-141, 2020.
[10] A. Sill, "*The Design and Architecture of Microservices*," IEEE Cloud Computing, vol. 3, no. 5, pp. 76-80, 2016.
[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1. Technical Report RFC 2616*, The Internet Society, http://www.ietf.org/rfc/rfc2616.txt, 1999.
[12] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.d. dissertation, University of California, Irvine, 2007.
[13] C. Pautasso. *Restful web services: principles, patterns, emerging technologies*. In Web Services Foundations, pages 31–51. Springer, 2014.
[14] D. Jacobson, G. Brail, D. Woods, *APIs: A Strategy Guide*. O'Reilly Media, Inc., 2011.
[15] M. Gebhart, P. Giessler, S. Abeck, "*Challenges of the Digital Transformation in Software Engineering*," ICSEA 2016: The Eleventh International Conference on Software Engineering Advances, 2016, pp. 136–141.
[16] D. M. Rathod, S. M. Parikh, B. V. Buddhadev, "*Structural and Behavioral Modeling of RESTful Web Service Interface Using UML*," International Conference on Intelligent Systems and Signal Processing (ISSP), March 2013, pp. 28–33.
[17] P. Giessler, M. Gebhart, D. Sarancin, R. Steinegger, S. Abeck, "*Best Practices for the Design of RESTful Web Services*," International Conferences of Software Advances (ICSEA), 2015.
[18] B. Hippchen, P. Giessler, R. Steinegger, M. Schneider, S. Abeck, "*Designing Microservice-Based Applications by Using a Domain Driven Design Approach*," International Journal on Advances in Software, Vol. 10, No. 3&4, pp. 432–445, 2017.
[19] V. Vaugh, *Implementing Domain-Driven Design*, Addison-Wesley Professional, 2013.
[20] C. Y. Fan, S. P. Ma, "*Migrating Monolithic Mobile Application to Microservice Architecture: An Experiment Report*," IEEE 6th International Conference on AI and Mobile Services, p. 109–112, 2017.
[21] I. W. Len Bass and L. Zhu, *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 2015.
[22] Evans, Eric, *Domain-driven design: tackling complexity in the heart of software*, Addison-Wesley, 2004.
[23] Martin Fowler (2014) *DomainDrivenDesign*. [Online]. Available: https://martinfowler.com/bliki/DomainDrivenDesign.html
[24] Martin Fowler (2014) *BoundedContext*. [Online]. Available: https://martinfowler.com/bliki/BoundedContext.html

[25] P. D. Francesco, I. Malavolta and P. Lago, "*Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption*," IEEE International Conference on Software Architecture (ICSA), pp. 21-30, 2017.

[26] Y. Gan and C. Delimitrou, "*The Architectural Implications of Cloud Microservices*," IEEE Computer Architecture Letters, vol. 17, no. 2, pp. 155-158, 2018.

[27] K. Kurniawan, N. Putra Sastra, and M. Sudarma, "*Analisis Performansi Dan Efisiensi Cloud Computing Pada Sistem Perbankan*", Majalah Ilmiah Teknologi Elektro, vol 19, no 1, pp 11-18, Oct. 2020.