Pengembangan *Backend* Dalam Migrasi Aplikasi Monolitik Ke *Microservice* Menggunakan Metode *Iterative Incremental Development* Pada Modul Pendaftaran dan Penjadwalan SOFI

1st Muhammad Nurul Afif Maliki
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
muhammadafifjpr@student.telkomuniversity.ac.id

2nd Ekky Novriza Alam
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
ekkynovrizalam@telkomuniversity.ac.id

3rd Tien Fabrianti Kusumasari
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
tienkusumasari@telkomuniversity.ac.id

***Abstrak*— Transformasi digital membawa perubahan signifikan di berbagai bidang, termasuk pendidikan. Universitas Telkom, sebagai salah satu perguruan tinggi swasta, mengembangkan aplikasi "SOFI" untuk mendukung kegiatan akademik di Fakultas Rekayasa Industri (FRI). Meskipun bermanfaat, aplikasi ini menghadapi masalah skalabilitas. Penelitian ini bertujuan mengimplementasikan Domain Driven Design dalam tahap perancangan sistem sehingga dapat mengetahui seberapa kecil modul pendaftaran dan penjadwalan yang sudah ada, hal ini dapat menjaga keutuhan serta konsistensi domain bisnis serta penggunaan metode Iterative Incremental Development dalam proses pengembangan backend pada modul pendaftaran dan penjadwalan yang dipecah menjadi aplikasi microservice untuk memastikan kesesuaian pengembangan. Hasilnya penerapan Domain Driven Design pada perancangan sistem terbukti efektif dalam proses migrasi pada modul pendaftaran dan penjadwalan. Aplikasi Sidang Fakultas SOFI berhasil dimigrasikan dari arsitektur monolitik ke microservices menggunakan Iterative Incremental Development. Enam firur berhasil dikembangkan. Pengujian terhadap 35 API menunjukkan hasil load testing yang sangat baik, dengan tingkat keberhasilan method utama mencapai sekitar 100% untuk 50 hingga 300 pengguna.**

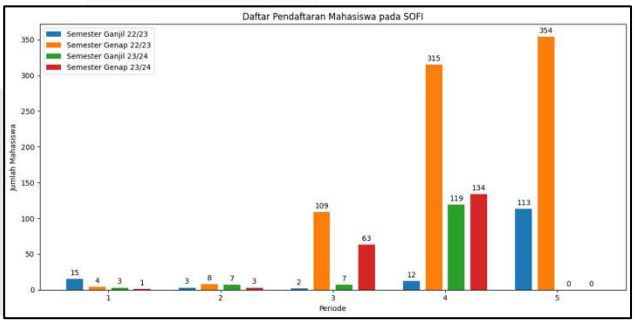
***Kata kunci*— Domain Driven Design, Iterative Incremental, Microservices**

I. PENDAHULUAN

Transformasi digital yang pesat tidak hanya mempengaruhi industri bisnis, tetapi juga pendidikan, yang kini berkembang menuju pendidikan 4.0. Tujuannya adalah membekali siswa dengan berbagai kemampuan agar siap menghadapi tantangan revolusi industri 4.0 dan tuntutan global [1]. Dalam konteks ini, Universitas Telkom telah mengembangkan aplikasi "SOFI" (Sidang Online Fakultas Rekayasa Industri) untuk mendukung kegiatan akademik di Fakultas Rekayasa Industri. SOFI adalah aplikasi berbasis web yang memfasilitasi pendaftaran, penjadwalan, pelaksanaan, dan revisi sidang Tugas Akhir (TA).

Pengembangan aplikasi SOFI menggunakan arsitektur monolitik yang memiliki beberapa keunggulan, seperti kemudahan pengembangan dan pengujian untuk aplikasi berskala kecil [2]. Namun, arsitektur ini juga memiliki kelemahan signifikan, termasuk kesulitan dalam modifikasi kode, layanan yang tidak independen, waktu mulai aplikasi yang lama, dan tantangan skalabilitas [3]. Hal ini menyebabkan masalah skalabilitas yang terbatas, yang menjadi kendala utama ketika beban pengguna meningkat [4].

Aplikasi sidang fakultas SOFI mengalami masalah skalabilitas yang sangat terbatas. Hal ini terbukti dalam sesi wawancara dengan pengembang aplikasi SOFI, yang dimana pengembang kesulitan dalam melakukan skalabilitas aplikasi seperti horizontal *scaling* pada layanan tertentu yang memiliki *load* pengguna yang sangat tinggi dan tidak menentu.



GAMBAR 1
Daftar Pendaftaran Sidang Sofi

Terlebih lagi, berdasarkan data dari Layanan Administrasi Akademik Fakultas Rekayasa Industri (LAA FRI) memperlihatkan fluktuasi jumlah pengajuan mahasiswa pada SOFI yang tidak menentu pada berbagai periode dan semester yang digambarkan pada Gambar 1 dengan puncaknya mencapai 315 pengajuan pada periode ke-4 dan 354 pengajuan pada periode ke-5 di Semester Genap 22/23. Keadaan ini menunjukkan bahwa layanan tertentu mengalami beban yang sangat tinggi secara tiba tiba, sehingga menyulitkan pengembang untuk meningkatkan layanan.

Untuk mengatasi masalah ini, migrasi dari arsitektur monolitik ke arsitektur *microservice* diusulkan. Pendekatan berbasis *microservice* telah terbukti meningkatkan skalabilitas, ketersediaan, performa, dan mengurangi biaya pemeliharaan [5][6][7][8]. Meskipun demikian, proses migrasi ini kompleks dan menghadapi berbagai tantangan, seperti identifikasi kebutuhan bisnis dan teknis yang kompleks, serta pengembangan *backend* yang efektif [3].

Dalam penelitian ini, pendekatan *Iterative Incremental Development* dipilih untuk mengembangkan *backend microservice*, karena pendekatan ini memungkinkan pengembangan bertahap dan adaptif terhadap perubahan kebutuhan [9]. Selain itu, prinsip *Domain Driven Design* (DDD) digunakan dalam analisis sistem untuk memastikan pemecahan sistem yang optimal sesuai dengan kebutuhan bisnis.

Tujuan dari penelitian ini adalah mengimplementasikan *Domain Driven Design* dalam tahap perancangan sistem sehingga dapat mengetahui seberapa kecil modul pendaftaran dan penjadwalan yang sudah ada, hal ini dapat menjaga keutuhan serta konsistensi domain bisnis. Mengimplementasikan metode *Iterative Incremental Developmen*t dalam proses pengembangan *backend* pada modul pendaftaran dan penjadwalan yang dipecah menjadi aplikasi *microservice* untuk memastikan kesesuaian pengembangan.

II. KAJIAN TEORI

A. Monolitik

Arsitektur monolitik adalah jenis arsitektur layanan yang dibangun dengan semua komponennya berjalan dalam satu blok tunggal. Contoh umumnya adalah aplikasi web yang terdiri dari lapisan UI, lapisan bisnis, dan lapisan akses data, yang beroperasi sebagai satu entitas tunggal [10]. Aplikasi monolitik biasanya memiliki satu kode sumber tunggal, dan seiring bertumbuhnya ukuran kode, beberapa masalah mulai muncul seperti kesulitan dalam modifikasi kode, layanan yang tidak independen, waktu mulai aplikasi yang lama, dan sulitnya pengembangan [11].

B. *Microservice*

Arsitektur *microservice* ditemukan oleh Lewis dan Fowler, merupakan arsitektur terdistribusi yang mendekomposisi sistem menjadi beberapa komponen kecil dan independen yang dapat dipanggil sesuai kebutuhan [12]. *Microservice* semakin diminati dengan munculnya inovasi infrastruktur berbasis cloud seperti *software-as-a-service* dan *function-as-a-service* [13]. Perusahaan ternama seperti Netflix, Amazon, dan Ebay telah beralih ke arsitektur *microservice*, di mana layanan berkomunikasi satu sama lain menggunakan protokol HTTP seperti REST API [3].

C. *Iterative Incremental Development*

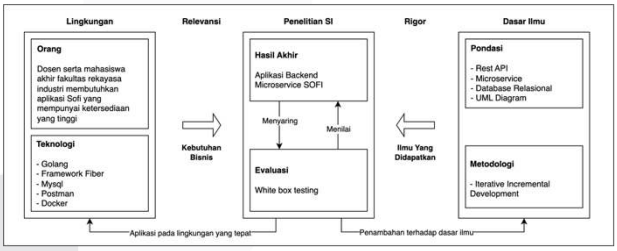
Metode *Iterative dan Incremental* adalah bagian dari *Feature-Driven Development* (FDD), di mana perangkat lunak dibagi menjadi berbagai fitur yang berbeda, dan setiap fitur dibangun secara terpisah [14]. Metode ini direkomendasikan dalam praktik pengembangan perangkat lunak karena dapat meningkatkan tingkat keberhasilan proyek dan menghemat anggaran [15].

Metode *Iterative Incremental* merupakan penyempurnaan dari metode *Waterfall* yang dinilai kaku, dengan sifat *Iterative* yang memungkinkan pengembangan bertahap dan penambahan fitur secara perlahan [16]. Meskipun *Iterative* dan *Incremental* tampak serupa, keduanya memiliki perbedaan mendasar; *Iterative* mengacu pada tindakan yang diulang-ulang, sedangkan *Incremental* mengacu pada tindakan penambahan yang baru [17]. Metode ini melibatkan identifikasi persyaratan, analisis, spesifikasi desain, pemrograman, dan pengumpulan umpan balik dari pemangku kepentingan sebelum melanjutkan ke fase berikutnya [18]. Setiap iterasi dalam pengembangan menggunakan metode ini melibatkan perencanaan, analisis dan desain, implementasi, pengujian, evaluasi, serta *deployment* [19].

III. METODE

A. Kerangka Berpikir

Kerangka berpikir, atau model konseptual, adalah kegiatan memetakan faktor-faktor untuk memberikan solusi dari masalah dan dampaknya terhadap target yang dituju. Kerangka berpikir menjelaskan konsep-konsep atau proposisi terkait pemecahan masalah yang telah diidentifikasi [20]. Dalam penelitian ini, model kerangka berpikir digunakan untuk mengilustrasikan fenomena yang diinvestigasi dan menyediakan solusi. Kerangka berpikir dalam penelitian ini mengacu pada *design science research* (Hevner, 2007). Kerangka ini menyediakan definisi, metodologi, batasan, panduan terstruktur, dan hasil konsisten untuk perancangan dan implementasi proyek penelitian. Hal ini juga meningkatkan kredibilitas penelitian desain ilmiah di kalangan komunitas yang lebih luas, termasuk teknik, arsitektur, seni, dan lainnya [21]. Model konseptual migrasi aplikasi monolitik ke *microservice* digambarkan pada Gambar 2.

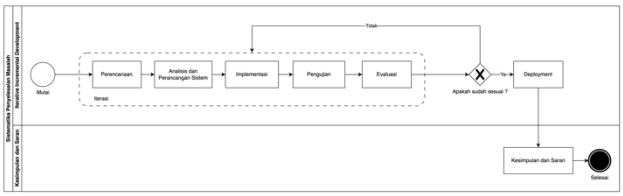


GAMBAR 2
Model Konseptual

B. Sistematika Penyelesaian Masalah

Penelitian ini bertujuan untuk melakukan migrasi aplikasi SOFI yang saat ini menggunakan arsitektur monolitik ke arsitektur *microservices*. SOFI berfungsi sebagai perantara bagi mahasiswa tingkat akhir yang sedang mengerjakan tugas akhir dengan dosen hal ini digunakan untuk memfasilitasi proses penyusunan tugas akhir.

Dalam melakukan pengembangan, penyelesaian masalah yang ada peneliti menggunakan metode *Iterative Incremental Development*, yang dimana terdapat beberapa tahap yang akan dilakukan dalam pelaksanaan pengembangan aplikasi tersebut yaitu tahap perencanaan, analisis dan perancangan sistem, implementasi migrasi aplikasi ke *microservice*, pengujian sampai dengan tahap *deployment*. Untuk gambaran lebih jelas mari liat pada Gambar 3.



GAMBAR 3
Sistematika Penyelesaian Masalah

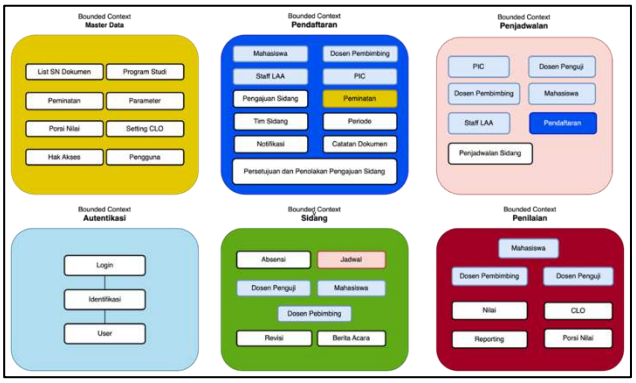
C. Pengumpulan Data

Dalam penelitian ini penulis melakukan proses pengumpulan data, dengan fokus utama pada wawancara mendetail bersama pengembang aplikasi tersebut, yang dalam konteks ini adalah Bapak Ekky Novriza Alam, S. Kom., M.T. Tujuan dari pengumpulan data ini adalah untuk memperoleh wawasan yang komprehensif mengenai serangkaian fitur yang diintegrasikan dalam aplikasi, mengidentifikasi berbagai kelompok pengguna yang memiliki hak akses, serta menganalisis domain-domain bisnis dari aplikasi. Metode kualitatif dipilih sebagai pendekatan utama dalam penelitian ini, dengan wawancara langsung sebagai teknik pengumpulan data utama, sehingga memungkinkan pengambilan informasi yang mendalam dan kontekstual langsung dari sumbernya. Setelah wawancara yang informatif dengan pengembang, langkah selanjutnya adalah proses penyusunan dan analisis data yang telah diperoleh.

IV. HASIL DAN PEMBAHASAN

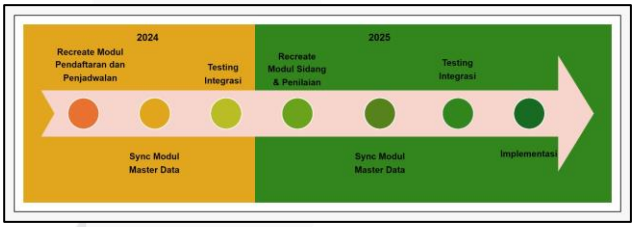
A. Identifikasi Layanan

Pada identifikasi layanan menggunakan analisis *Domain-Driven Design* (*DDD*), yang dilaksanakan melalui diskusi intensif dengan domain *expert* untuk mengaitkan konsep bisnis secara mendalam dengan implementasi teknis, ditujukan untuk mempercepat pengembangan perangkat lunak yang secara langsung terkait dengan domain bisnis yang kompleks, contohnya aplikasi untuk sidang fakultas SOFI. Untuk menangani masalah pengguna dan mengatasi perbedaan dalam konteks domain saat mengembangkan perangkat lunak skala besar, pendekatan *bounded context* digunakan untuk secara eksplisit membagi model besar menjadi konteks-konteks kecil yang lebih terkelola dan mendefinisikan hubungan antar mereka. Ini memudahkan proses pengembangan perangkat lunak dengan mengeliminasi kebingungan yang mungkin timbul dari perbedaan kosakata atau konsep ilmiah. Gambar 4 menampilkan hasil dari analisis *bounded context* yang telah dilakukan.



GAMBAR 4
Analisis Bounded Context

Gambar 4 menunjukkan analisis *bounded context* yang lengkap untuk aplikasi sidang fakultas SOFI. Namun, terdapat pembatasan dalam pemilihan *bounded context* karena keterbatasan waktu pengembangan yang telah disepakati sebelumnya. *Bounded context* pendaftaran dan penjadwalan dipilih sebagai fokus utama karena peran strategis mereka dalam arsitektur sistem secara keseluruhan. Adapun *bounded context* lainnya seperti, master data, autentikasi, sidang dan penilaian dijadikan sebagai fokus pada penelitian lebih lanjut. Hal ini dikarenakan proses pendaftaran dan penjadwalan tidak hanya krusial untuk operasi bisnis utama, tetapi juga berperan penting sebagai penghubung data esensial untuk *bounded context* lainnya, maka dari itu proses migrasi aristektur monolitik ke *microservice* memiliki rencana pemecahan layanan yang ditunjukan pada Gambar 5.



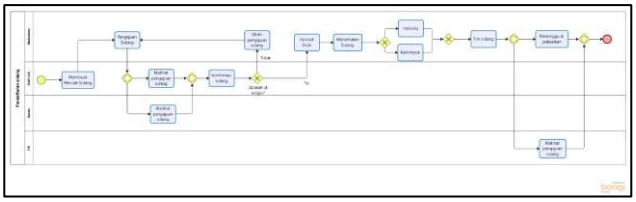
GAMBAR 5
Rencana Migrasi Aplikasi

Kedua konteks ini dirancang untuk mengelola dan menyediakan informasi yang diperlukan untuk menjalankan berbagai aktivitas bisnis lain. Integrasi yang cermat antara pendaftaran dan penjadwalan memungkinkan aplikasi menyediakan alur kerja yang efisien, memastikan semua permintaan pengguna diproses dengan akurat. Oleh karena itu, pengembangan awal pada kedua *bounded context* ini dianggap kritikal; tanpanya, aplikasi mengalami kesulitan beroperasi secara efektif, menghasilkan pengalaman pengguna yang tidak optimal dan potensi kegagalan dalam mencapai tujuan bisnis

B. *Initial Planning*

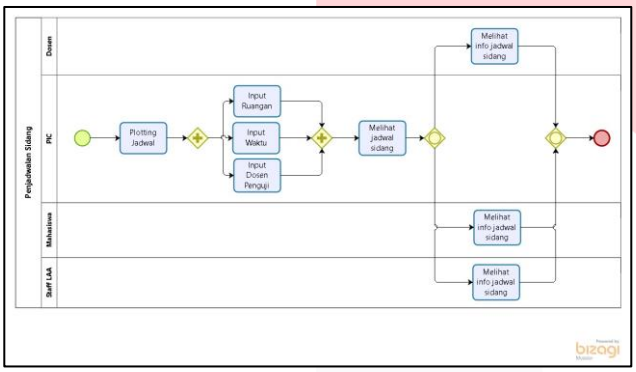
Pada tahap awal *Iterative Incremental Development*, yang dikenal sebagai *Initial Planning*, dilakukan identifikasi fungsionalitas aplikasi *existing* dengan menetapkan fitur atau kebutuhan yang akan dilakukan migrasi. Selain itu, tahap ini juga melibatkan analisis proses bisnis dan pendefinisian aktor yang terlibat.

Proses bisnis pendaftaran sidang digambarkan pada Gambar 6 yang dimana berfungsi untuk menggambarkan proses melakukan pendaftaran sidang akhir pada aplikasi yang akan di migrasi.



GAMBAR 6
Proses Bisnis Pendaftaran Sidang

Proses bisnis penjadwalan digambarkan pada Gambar 7 yang dimana berfungsi untuk menggambarkan cara kerja dari penjadwalan sidang mahasiswa yang nantinya akan dijadwalkan oleh PIC.



GAMBAR 7
Proses Bisnis Penjadwalan Sidang

C. Tahap Perencanaan

Perencanaan yang dilakukan pada tahap awal dalam fase pertama bertujuan untuk merancang rencana migrasi aplikasi fakultas SOFI dari arsitektur monolitik ke *microservice*. Fungsionalitas pada sistem *existing* dijabarkan pada Tabel 1. Hal ini didapatkan berdasarkan percobaan langsung pada aplikasi *existing* yang dimana masih menggunakan arsitektur monolitik.

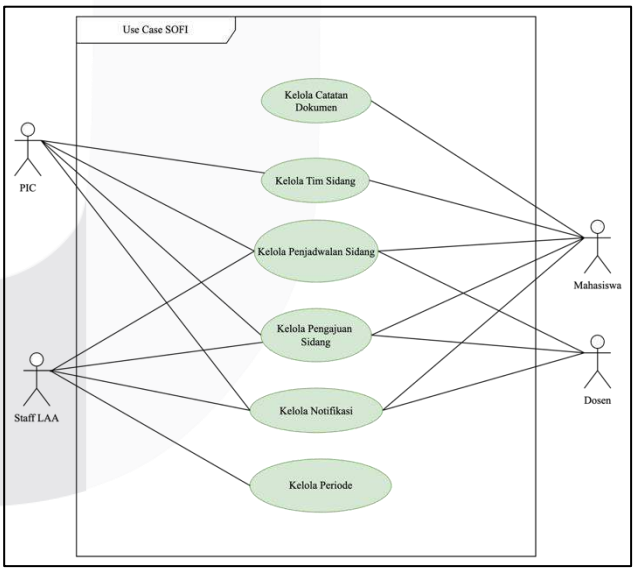
TABEL 1
Fungsionalitas Sistem Sofi Modul Pendaftaran Dan Penjadwalan

Fitur	***Method***
Kelola Periode	Mendapatkan Seluruh Periode Sidang
	Mendapatkan Detail Periode Sidang
	Membuat Periode Sidang
	Mengubah Periode Sidang
	Menghapus Periode Sidang
Kelola Pengajuan Sidang	Mendapatkan Seluruh Pengajuan Sidang
	Cek Pengajuan Sidang Pengguna
	Membuat Pengajuan Sidang
	Mengubah Pengajuan Sidang
	Menyetujui Pengajuan Sidang
	Menolak Pengajuan Sidang
Kelola Tim Sidang	Mendapatkan Detail Tim Sidang
	Mendapatkan Detail Tim Sidang Pengguna
	Membuat Tim Sidang
	Membuat Sidang Individu
	Menambahkan Anggota Tim Sidang
	Keluar Tim Sidang
	Ubah Nama Tim Sidang
Kelola Catatan Dokumen	Mendapatkan Detail Catatan Dokumen
	Upload Slide
Kelola Notifikasi	Mendapatkan Notifikasi Pengguna
	Ubah Notifikasi
Kelola Penjadwalan Sidang	Mendapatkan Seluruh Jadwal Sidang
	Mendapatkan Detail Jadwal Sidang
	Membuat Jadwal Sidang
	Mengubah Jadwal Sidang
	Menandai Jadwal Sidang
	Menghapus Jadwal Sidang

D. Tahap Analisis dan Perancangan

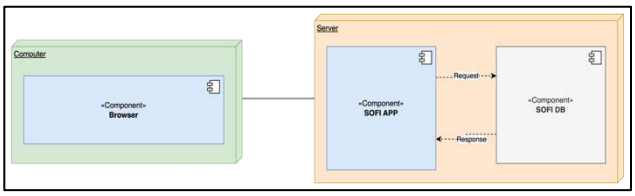
Tahap kedua, yaitu analisis dan perancagan, dilakukan untuk menganalisis sistem *existing* yang dimana akan dilakukan migrasi ke arsitektur *microservice* serta merancang sistem yang akan dilakukan *refactoring* ke arsitektur *microservice.* Pada perancangan akan terdapat perubahan pada hasil analisis sistem *existing* guna memastikan bahwa migrasi ke arsitektur *microservice* berjalan dengan baik kebutuhan yang telah ditetapkan pada fase sebelumnya. Pada tahap analisis, menghasilkan *use case diagram, entity relationship diagram*, *deployment diagram existing*. Namun pada percangan sistem terjadi perubahan pada *deployment diagram* yang dimana akan disesuaikan untuk aplikasi *microservice* yang akan di implementasikan.

Use case diagram digunakan untuk menggambarkan fungsi-fungsi yang tersedia bagi pengguna dalam aplikasi sidang fakultas SOFI. Perancangan *use case diagram* ini didasarkan pada fitur aplikasi *existing*. Diagram ini dibuat untuk mengilustrasikan hubungan antara aktor pengguna dengan sistem. *Use case diagram* yang telah dibuat digambarkan pada Gambar 8.



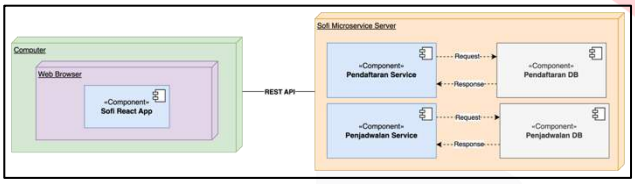
GAMBAR 8
Use Case Diagram Modul Pendaftaran Dan Penjadwalan

Deployment diagram existing memberikan gambaran mengenai struktur infrastruktur SOFI yang saat ini. Diagram ini menggambarkan bahwa aplikasi SOFI saat ini yang berjalan pada arsitektur monolitik. *Deployment diagram existing* digambarkan pada Gambar 9.



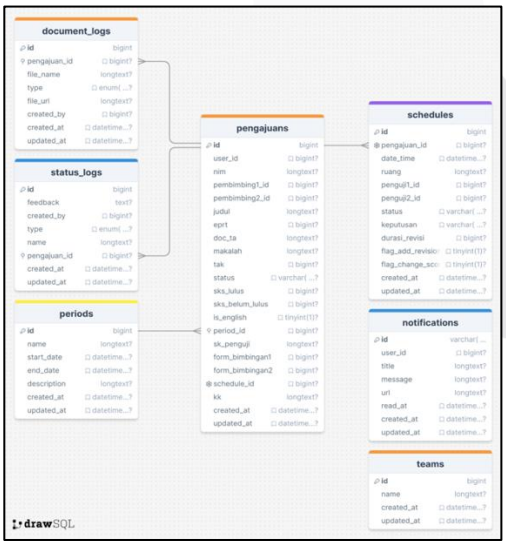
GAMBAR 9
Deployment Diagram Monolitik

Ketika aplikasi monolitik akan dilakukan migrasi ke arsitektur *microservice* maka terjadi perubahan pula terhadap *Deployment diagram* yang sudah ada, yang dimana layanan-layanan tertentu dilakukan *deployment* secara terpisah. Hal ini berdampak pada fleksibilitas dalam peningkatan skalabilitas sistem karena kedua layanan ini telah terpisah dari layanan lainnya, yang dimana ketika layanan "Pendaftaran" dan "Penjadwalan" menginginkan peningkatan performa pada server yang sedang berjalan, maka hanya kedua layanan tersebut yang mengalami peningkatan performa tanpa melibatkan layanan lain. *Deployment diagram targeting* digambarkan pada Gambar 10.



GAMBAR 10
Deployment Diagram Microservice

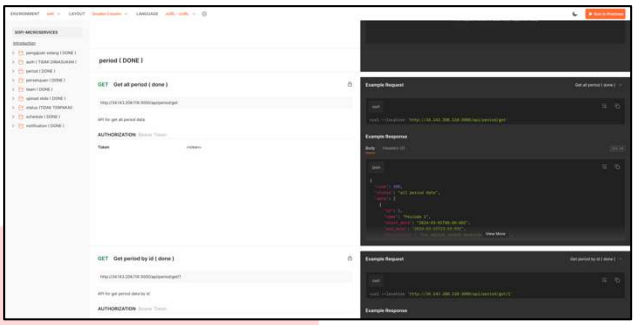
Entity Relationship Diagram (*ERD*) adalah suatu metode perancangan yang digunakan untuk menggambarkan struktur basis data yang terlibat dalam proses migrasi aplikasi sidang fakultas SOFI. ERD aplikasi yang akan di lakukan migrasi digambarkan pada Gambar 11.



GAMBAR 11
Entity Relationship Diagram

E. Tahap Pengembangan

Tahap ketiga dalam *Iterative Incremental Development* adalah tahap pengembangan. Pada tahap ini, *backend* dikembangkan berdasarkan fungsionalitas sistem *existing* yang telah dianalisis, rancangan UML, dan ERD yang telah disiapkan sebelumnya dari tahap perencanaan, analisis, dan perancangan. Pengembangan dilakukan melalui kolaborasi dengan tim *frontend* menggunakan *repository* di GitHub. Bahasa pemrograman Golang digunakan untuk mengembangkan *backend* dengan menerapkan konsep *Domain Driven Design,* yang bertujuan untuk memisahkan logika bisnis dari logika aplikasi itu sendiri. Pada Gambar 12 merupakan API dari aplikasi yang telah dilakukan migrasi.



GAMBAR 12
Dokumentasi Api

F. Tahap Pengujian

Setelah tahap pengembangan selesai untuk dilaksanakan, maka tahap selanjutnya adalah pengujian terhadap sistem yang sudah di lakukan migrasi, yang dimana hal ini bertujuan agar mengetahui apakah *method* yang telah di migrasi berjalan sesuai harapan atau tidak. Pengujian ini dilakukan menggunakan fitur yang ada di aplikasi postman yaitu fitur "*integration testing*". Pada Tabel 2 menjabarkan mengenai *integration testing* dari migrasi layanan yang telah dilakukan. *Integration testing* dilakukan menggunakan aplikasi postman.

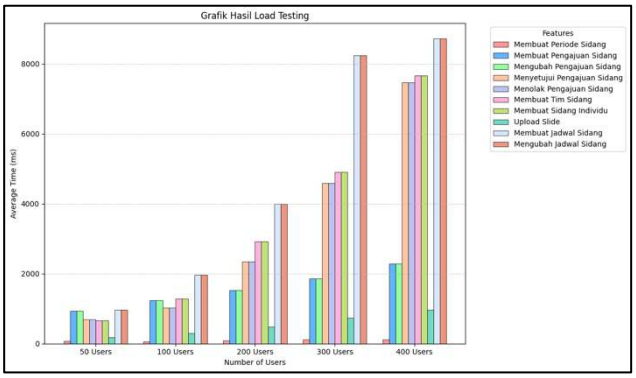
TABEL 2
Hasil *Integration Testing Microservice*

No	*Method*	*Status Code*	*Result*
1.	Mendapatkan Seluruh Periode Sidang	200	*Pass*
2.	Mendapatkan Detail Periode Sidang	200	*pass*
3.	Membuat Periode Sidang	201	*pass*
4.	Mengubah Periode Sidang	200	*pass*
5.	Menghapus Periode Sidang	200	*pass*
6.	Mendapatkan Seluruh Pengajuan Sidang (Staff LAA)	200	*pass*
7.	Mendapatkan Seluruh Pengajuan Sidang (Dosen)	200	*pass*
8.	Mendapatkan Seluruh Pengajuan Sidang (PIC)	200	*pass*
9.	Cek Pengajuan Sidang Pengguna	200	*pass*
10.	Membuat Pengajuan Sidang	201	*Pass*
11.	Mengubah Pengajuan Sidang	200	*pass*
12.	Menyetujui Pengajuan Sidang	200	*Pass*
13.	Menolak Pengajuan Sidang	200	*pass*
14.	Mendapatkan Detail Tim Sidang	200	*pass*
15.	Mendapatkan Detail Tim Sidang Pengguna	200	*pass*
16.	Membuat Tim Sidang	201	*Pass*
17.	Membuat Sidang Individu	201	*pass*
18.	Menambahkan Anggota Tim Sidang	200	*pass*
19.	Keluar Tim Sidang	200	*pass*
20.	Ubah Nama Tim Sidang	200	*pass*
21.	Mendapatkan Detail Catatan Dokumen	200	*pass*
22.	*Upload* Slide	201	*Pass*
23.	Mendapatkan Notifikasi Pengguna	200	*pass*
24.	Ubah Notifikasi	200	*pass*
25.	Mendapatkan Seluruh Jadwal Sidang (Staff LAA)	200	*pass*
26.	Mendapatkan Seluruh Jadwal Sidang (PIC)	200	*Pass*
27.	Mendapatkan Seluruh Jadwal Sidang (Dosen)	200	*pass*
28.	Mendapatkan Detail Jadwal Sidang (Staff LAA, PIC dan Dosen)	200	*Pass*
29.	Mendapatkan Detail Jadwal Sidang (Mahasiswa)	200	*pass*
30.	Membuat Jadwal Sidang	200	*Pass*
31.	Mengubah Jadwal Sidang	200	*pass*
32.	Menandai Jadwal Sidang	200	*pass*
33.	Menghapus Jadwal Sidang	200	*pass*

Setelah tahap pengujian "*Integration Testing*" selesai maka tahap pengujian selanjutnya adalah *Load Testing* Pengujian ini bertujuan untuk mengukur seberapa baik sistem dapat menangani beban tertentu dengan berbagai tingkat penggunaan yang diharapkan. Untuk mencapai ini, simulasi dilakukan dengan jumlah pengguna yang berbeda secara bersamaan, yaitu 50, 100 dan 200, 300 dan 400 pengguna.

Penggunaan 400 pengguna dalam simulasi ini didasarkan pada data yang telah digambarkan pada Gambar 1. Grafik tersebut memperlihatkan jumlah pengajuan mahasiswa pada SOFI yang dapat mencapai puncaknya hingga 354 pengajuan pada periode ke-5 Semester Genap 22/23. Angka ini memberikan dasar yang kuat untuk mensimulasikan beban 400 pengguna secara bersamaan, guna memastikan bahwa sistem dapat menangani puncak beban yang tinggi.

Hasil pengujian *load testing* terlihat pada Gambar 13 yang dimana digunakan untuk menganalisis respons dan kinerja sistem. Dalam rangka membuktikan fleksibilitas dan independensi aplikasi, pengujian dilakukan pada *virtual machine.*



GAMBAR 13
Hasil *Load Testing*

Grafik pada Gambar 13 menggambarkan hasil dari *load testing* aplikasi ketika diakses oleh 50, 100, 200, 300, dan 400 pengguna simultan. Sumbu horizontal menampilkan jumlah pengguna, sementara sumbu vertikal menunjukkan waktu rata-rata (dalam milidetik) yang diperlukan untuk menyelesaikan setiap *method* aplikasi, seperti Membuat Periode Sidang, Membuat Pengajuan Sidang, Mengubah Pengajuan Sidang, Menyetujui Pengajuan Sidang, Menolak Pengajuan Sidang, Membuat Tim Sidang, Membuat Sidang Individu, *Upload* Slide, Membuat Jadwal Sidang, dan Mengubah Jadwal Sidang. Analisis grafik menunjukkan bahwa waktu eksekusi meningkat seiring bertambahnya jumlah pengguna, dengan *method* Membuat Jadwal Sidang dan Mengubah Jadwal Sidang memerlukan waktu terlama, terutama pada 300 dan 400 pengguna. *Method* Membuat Periode Sidang dan *Upload* Slide memiliki waktu eksekusi tercepat.

Peningkatan waktu eksekusi yang konsisten pada semua *method* mencerminkan pola kinerja yang dapat diprediksi saat beban meningkat. Hasil ini menekankan pentingnya optimisasi aplikasi atau peningkatan infrastruktur untuk memastikan kinerja yang baik saat jumlah pengguna bertambah. Namun, pada pengujian *load testing* dengan 400 pengguna, terjadi *error* pada dua *method*, yaitu Membuat Jadwal Sidang dan Mengubah Jadwal Sidang. *Error* ini terjadi pada 176 pengguna atau sekitar 22% dari total 800 pengguna. Penting untuk dicatat bahwa kedua *methpd* ini hanya dapat diakses oleh pengguna PIC. Dalam kasus sebenarnya, PIC di aplikasi hanya dimiliki sekitar 10 pengguna. Oleh karena itu, hal ini tidak berdampak signifikan pada aplikasi saat diakses oleh 400 pengguna secara bersamaan. Berdasarkan hasil tersebut secara keseluruhan, aplikasi berjalan dengan baik dan stabil untuk jumlah pengguna antara 50 hingga 300 pengguna.

G. Tahap Evaluasi

Berdasarkan hasil pengujian yang dilakukan pada tahap-tahap sebelumnya, dapat disimpulkan bahwa pada siklus migrasi semua fitur telah berjalan dengan baik. Pengguna dapat menjalankan semua fitur yang dilakukan migrasi. Setiap komponen dari aplikasi atau sistem telah memenuhi kriteria yang ditentukan dan tidak ada masalah yang signifikan ditemukan selama pengujian. Oleh karena itu, diputuskan bahwa fase iterasi dapat dihentikan.

H. Tahap *Deployment*

Tahap *deployment* dilaksanakan setelah menyelesaikan proses *Iterative Incremental Development*. Langkah ini dilakukan setelah semua iterasi dalam *Iterative Incremental Development* selesai. *Deployment* dilakukan menggunakan platform idCloudHost, yang menyediakan layanan *virtual private server* untuk memastikan aplikasi dapat diakses oleh semua pengguna. Dengan menggunakan layanan *compute engin*e untuk menjalankan aplikasi sidang fakultas SOFI dan database MySQL secara global, dengan spesifikasi 8 vCPU (4 core) dan 32 gigabyte memori. Aplikasi server ditempatkan di Jakarta. idCloudHost tidak menyediakan layanan secara gratis. Namun, idCloudHost menggunakan konsep "*Pay as you go*" yang memungkinkan pengguna membayar *resource* hanya saat digunakan.

V. KESIMPULAN

Penelitian ini menunjukkan bahwa penerapan *Domain Driven Design* pada perancangan sistem terbukti efektif dalam proses migrasi pada modul pendaftaran dan penjadwalan. Hal ini dapat membantu dalam pemecahan sistem menjadi konteks yang lebih kecil dan terkelola. Pendekatan ini memungkinkan pengembangan yang lebih terstruktur dan menjaga konsistensi domain bisnis yang ada. Selain itu, pada modul pendaftaran dan penjadwalan sidang aplikasi SOFI yang awalnya memiliki arsitektur monolitik telah berhasil dimigrasikan ke arsitektur *microservice* dengan mengimplementasikan metode *Iterative Incremental Development*. Terdapat enam fitur yang berhasil dikembangkan. Melalui pengujian fungsionalitas terhadap 35 API yang telah dibuat, hasil *load testing* menunjukkan tingkat keberhasilan yang sangat baik, dengan fitur utama mencapai tingkat keberhasilan sekitar 100% untuk 50 hingga 300 pengguna. Namun, ketika jumlah pengguna mencapai 400, tingkat keberhasilan sistem menurun menjadi 97,8%. Hal ini terjadi karena dua *method*, yaitu "Membuat Jadwal Sidang" dan "Mengubah Jadwal Sidang" masing-masing memiliki tingkat *error* sebesar 11%. Namun, hal ini tidak akan menjadi masalah karena *method* tersebut hanya bisa diakses oleh PIC, yang pada kenyataannya hanya mencapai 10 pengguna. Oleh karena itu, seluruh hasil pengujian fungsionalitas memenuhi standar yang diharapkan, menegaskan bahwa sistem yang telah dimigrasikan mampu memberikan layanan yang handal dan sesuai dengan kebutuhan pengguna.

REFERENSI

[1] K. K. Katyeudo and R. A. C. de Souza, "Digital Transformation towards Education 4.0," *Informatics in Education*, vol. 21, no. 2, pp. 283–309, 2022, doi: 10.15388/infedu.2022.13.

[2] L. De Lauretis, "From monolithic architecture to microservices architecture," in *Proceedings - 2019 IEEE 30th International Symposium on Software Reliability Engineering Workshops, ISSREW 2019*, Institute of Electrical and Electronics Engineers Inc., Oct. 2019, pp. 93–96. doi: 10.1109/ISSREW.2019.00050.

[3] A. Trichur Ramachandran, Abhishek, Mamatha, Rashmi, Badrinath, and M. Parmar, "Understanding Migration from Monolithic to Microservice Architecture and its Challenges," *International Journal of Scientific Research and Engineering Development*, vol. 4, no. 3, 2021, [Online]. Available: www.ijsred.com

[4] O. Al-Debagy and P. Martinek, "A Comparative Review of Microservices and Monolithic Architectures," 2018.

[5] G. Munawar and A. Hodijah, "Analisis Model Arsitektur Microservice Pada Sistem Informasi DPLK," *Publikasi Jurnal & Penelitian Teknik Informatika*, vol. 3, no. 1, 2018.

[6] T. Prasandy, Titan, F. D. Mirad, and T. Darwis, "Migrating Application from Monolith to Microservices," *Migrating application from monolith to microservices*, 2020.

[7] D. Kuryazov, D. Jabborov, and B. Khujamuratov, "Towards Decomposing Monolithic Applications into Microservices," in *14th IEEE International Conference on Application of Information and Communication Technologies, AICT 2020 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020. doi: 10.1109/AICT50176.2020.9368571.

[8] S. Salii, J. Ajdari, and X. Zenuni, "Migrating to a microservice architecture: benefits and challenges," 2023.

[9] A. Rahmatulloh, D. W. Sari, R. N. Shofa, and I. Darmawan, "Microservices-based IoT Monitoring Application with a Domain-driven Design Approach," in *2021 International Conference Advancement in Data Science, E-Learning and Information Systems, ICADEIS 2021*, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ICADEIS52521.2021.9701966.

[10] V. Velepucha and P. Flores, "Monoliths to microservices-Migration Problems and Challenges: A SMS," in *Proceedings - 2021 2nd International Conference on Information Systems and Software Technologies, ICI2ST 2021*, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 135–142. doi: 10.1109/ICI2ST51859.2021.00027.

[11] G. Blinowski, A. Ojdowska, and A. Przybylek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access*, vol. 10, pp. 20357–20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[12] A. Suljkanović, B. Milosavljević, V. Indić, and I. Dejanović, "Developing Microservice-Based Applications Using the Silvera Domain-Specific Language," *Applied Sciences (Switzerland)*, vol. 12, no. 13, Jul. 2022, doi: 10.3390/app12136679.

[13] Y. Abgaz *et al.*, "Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review," *IEEE Transactions on Software Engineering*, vol. 49, no. 8, pp. 4213–4242, Aug. 2023, doi: 10.1109/TSE.2023.3287297.

[14] M. Al-Zewairi, M. Biltawi, W. Etaiwi, and A. Shaout, "Agile Software Development Methodologies: Survey of Surveys," *Journal of*

Computer and Communications, vol. 05, no. 05, pp. 74–97, 2017, doi: 10.4236/jcc.2017.55007.

[15] C. Larman and V. R. Basili, "Iterative and Incremental Development: A Brief History," 2003.

[16] A. Alshamrani and A. Bahattab, "A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model," *A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model* , vol. 12, no. 1, 2015, [Online]. Available: www.IJCSI.org

[17] E. S. Prasatya, C. M. Saputra, and P. Djoko, "Pengembangan Sistem Informasi Data Pasien Seksi Rehabilitasi BNN Kota Malang Menggunakan Metode Iterative Incremental," *Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 2, no. 12, pp. 6587–6596, 2018, [Online]. Available: http://j-ptiik.ub.ac.id

[18] K. Petersen and C. Wohlin, "A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case," 2009, [Online]. Available: www.ericsson.com

[19] A. B. M. Moniruzzaman and A. S. D. Hossain, "Comparative Study on Agile software development methodologies," 2013.

[20] E. Lisna Rahmadani, H. Sulistiani, and F. Hamidy, "RANCANG BANGUN SISTEM INFORMASI AKUNTANSI JASA CUCI MOBIL (STUDI KASUS : CUCIAN GADING PUTIH)," *Jurnal Teknologi dan Sistem Informasi (JTSI)*, vol. 1, no. 1, pp. 22–30, 2020, [Online]. Available: http://jim.teknokrat.ac.id/index.php/sisteminformasi

[21] A. R. Hevner, "A Three Cycle View of Design Science Research," 2007.