

**RANCANG BANGUN ARSITEKTUR *MICROSERVICES*  
MENGUNAKAN *GOOGLE REMOTE PROCEDURE CALL (GRPC)*  
*APPLICATION PROGRAMMING INTERFACE (API)* DAN METODE  
*DOMAIN-DRIVEN DESIGN***

SKRIPSI

Diajukan untuk Memenuhi Sebagian dari  
Syarat Memperoleh Gelar Sarjana Komputer  
pada Program Studi Ilmu Komputer



oleh  
Sekar Madu Kusumawardani  
2007703

**PROGRAM STUDI ILMU KOMPUTER  
FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN  
ALAM  
UNIVERSITAS PENDIDIKAN INDONESIA  
2024**

**RANCANG BANGUN ARSITEKTUR *MICROSERVICES*  
MENGUNAKAN *GOOGLE REMOTE PROCEDURE CALL (GRPC)*  
*APPLICATION PROGRAMMING INTERFACE (API)* DAN METODE  
*DOMAIN-DRIVEN DESIGN***

Oleh  
Sekar Madu Kusumawardani  
2007703

Sebuah Skripsi yang Diajukan untuk Memenuhi Salah Satu Syarat Memperoleh  
Gelar Sarjana Komputer pada Fakultas Pendidikan Matematika dan Ilmu  
Pengetahuan Alam

© Sekar Madu Kusumawardani  
Universitas Pendidikan Indonesia  
Agustus 2024

Hak Cipta Dilindungi Undang-Undang  
Skripsi ini tidak boleh diperbanyak seluruhnya atau sebagian, dengan dicetak  
ulang, difoto kopi, atau cara lainnya tanpa izin dari penulis

**SEKAR MADU KUSUMAWARDANI**

**2007703**

**RANCANG BANGUN ARSITEKTUR *MICROSERVICES*  
MENGUNAKAN *GOOGLE REMOTE PROCEDURE CALL (GRPC)*  
*APPLICATION PROGRAMMING INTERFACE (API)* DAN METODE  
*DOMAIN-DRIVEN DESIGN***

Disetujui dan disahkan oleh:

Pembimbing I,



**Dr. Asep Wahyudin, M.T.**

NIP. 197112232006041001

Pembimbing II,



**Herbert Siregar, M.T.**

NIP. 197005022008121001

Mengetahui,

Kepala Program Studi Ilmu Komputer



**Dr. Muhamad Nursalman, M.T.**

NIP 197909292006041002

**RANCANG BANGUN ARSITEKTUR *MICROSERVICES*  
MENGUNAKAN *GOOGLE REMOTE PROCEDURE CALL (GRPC)*  
*APPLICATION PROGRAMMING INTERFACE (API)* DAN METODE  
*DOMAIN-DRIVEN DESIGN***

Oleh

Sekar Madu Kusumawardani – sekarmadu@upi.edu

2007703

**ABSTRAK**

Aplikasi Tracer Study UPI merupakan suatu sistem *online* yang dapat digunakan oleh Perguruan Tinggi untuk melacak aktivitas para lulusannya setelah masa pendidikan tinggi. Penting bagi UPI sebagai suatu organisasi pendidikan untuk terus mengembangkan sistem tersebut dengan mengikuti perkembangan teknologi agar sistem mampu beradaptasi terhadap kemajuan yang ada serta menjadi perguruan tinggi yang adaptif dan tidak kehilangan relevansinya. Aplikasi Tracer Study UPI saat ini masih menerapkan arsitektur *monolithic* yang memiliki banyak kekurangan yang mengurangi poin adaptif organisasi, yaitu dari sisi *scalability*, *maintainability*, *fault tolerant*, dan *performance*. Penelitian ini bertujuan untuk merancang dan membangun arsitektur *microservices* pada aplikasi Tracer Study UPI menggunakan *gRPC API* untuk mengatasi masalah penerapan arsitektur *monolithic* pada aplikasi Tracer Study UPI. Metode *Domain-Driven Design* pada penelitian ini digunakan untuk mendekomposisi aplikasi Tracer Study yang sudah ada dan merancang ulang sistem tersebut dengan menerapkan arsitektur *microservices*. Penelitian ini menguji dan mengevaluasi hasil pengimplementasian arsitektur *microservices* pada aplikasi Tracer Study UPI dengan berfokus pada pengujian kinerja atau *performance testing*. Hasilnya yaitu implementasi arsitektur *microservices* pada Tracer Study UPI yang berjalan baik sesuai dengan fungsionalitas awal. Dari pengujian kinerja yang dilakukan juga ditemukan bahwa terjadi peningkatan dari sisi kinerja pada Tracer Study UPI setelah diterapkannya arsitektur *microservices*.

Kata Kunci: *microservices*, *tracer study*, *domain-driven design*, *DDD*, *gRPC*, perguruan tinggi adaptif, sistem informasi perguruan tinggi, migrasi *monolithic*

***MICROSERVICES ARCHITECTURE DESIGN AND DEVELOPMENT  
USING GOOGLE REMOTE PROCEDURE CALL (GRPC) APPLICATION  
PROGRAMMING INTERFACE (API) AND DOMAIN-DRIVEN DESIGN  
METHOD***

*Arranged by*

*Sekar Madu Kusumawardani – sekarmadu@upi.edu*

*2007703*

**ABSTRACT**

*The UPI Tracer Study application is an online system that can be used by universities to track the activities of their graduates after higher education. It is important for UPI as an educational organization to continue to develop the system by following technological developments so that the system is able to adapt to existing advances and become an adaptive university and not lose its relevance. The current UPI Tracer Study application still implements a monolithic architecture that has many shortcomings that reduce the adaptive points of the organization, namely in terms of scalability, maintainability, fault tolerant, and performance. This research aims to design and build a microservices architecture on the UPI Tracer Study application using the gRPC API to overcome the problem of implementing a monolithic architecture in the UPI Tracer Study application. The Domain-Driven Design method in this study was used to decompose the existing Tracer Study application and redesign the system by implementing a microservices architecture. This research tests and evaluates the results of implementing microservices architecture on the UPI Tracer Study application by focusing on performance testing. The result is the implementation of microservices architecture on Tracer Study UPI which runs well according to the initial functionality. From the performance testing conducted, it was also found that there was an increase in terms of performance on Tracer Study UPI after the implementation of microservices architecture.*

*Keywords: microservices, tracer study, domain-driven design, DDD, gRPC, adaptive organization, information system, monolithic migration*

## DAFTAR ISI

ABSTRAK .....	i
ABSTRACT .....	ii
KATA PENGANTAR.....	iii
UCAPAN TERIMA KASIH .....	iv
DAFTAR ISI .....	vi
DAFTAR GAMBAR .....	x
DAFTAR TABEL.....	xii
DAFTAR LAMPIRAN .....	xiii
BAB I PENDAHULUAN .....	1
1. 1 Latar Belakang.....	1
1. 2 Rumusan Masalah .....	7
1. 3 Tujuan Penelitian.....	7
1. 4 Manfaat Penelitian.....	8
1. 5 Batasan Masalah.....	10
1. 6 Sistematika Penulisan.....	11
BAB II TINJAUAN PUSTAKA .....	13
2. 1 Peta Literatur .....	13
2. 2 Penelitian Terkait.....	14
2. 3 Tracer Study .....	24
2. 4 Arsitektur <i>Monolithic</i> .....	25
2. 4. 1 <i>The Single-Process Monolith</i> .....	26
2. 4. 2 <i>The Modular Monolith</i> .....	27
2. 4. 3 <i>The Distributed Monolith</i> .....	28
2. 4. 4 Kelebihan Arsitektur <i>Monolithic</i> .....	28
2. 4. 5 Kekurangan Arsitektur <i>Monolithic</i> .....	28

2. 5	Arsitektur <i>Microservices</i> .....	30
2. 5. 1	Kelebihan Arsitektur <i>Microservices</i> .....	32
2. 5. 2	Kekurangan Arsitektur <i>Microservices</i> .....	34
2. 6	Dekomposisi Arsitektur <i>Monolithic</i> .....	35
2. 6. 1	<i>Domain-Driven Design (DDD)</i> .....	35
2. 6. 2	Teknik dan Tahap Dekomposisi <i>Microservices</i> .....	39
2. 7	<i>Application Programming Interface (API)</i> .....	45
2. 7. 1	<i>API Gateway</i> .....	45
2. 7. 2	<i>gRPC API</i> .....	47
2. 8	Pengujian Kinerja ( <i>Performance Testing</i> ) .....	51
2. 8. 1	Klasifikasi dan Indikator Pengujian Kinerja .....	53
2. 8. 2	Alat Pengujian Kinerja ( <i>Performance Testing Tools</i> ) .....	54
2. 8. 3	Metode Pengujian Kinerja.....	54
BAB III METODOLOGI PENELITIAN.....		56
3. 1	Desain Penelitian .....	56
3. 2	Pendekatan Penelitian.....	62
3. 3	Jenis Penelitian .....	63
3. 4	Lokasi dan Waktu Penelitian .....	63
3. 5	Objek Penelitian .....	63
3. 6	Variabel Penelitian.....	63
3. 7	Jenis dan Sumber Data .....	64
3. 8	Teknik Pengumpulan Data .....	64
3. 9	Teknik Analisis Data.....	68
3. 10	Alat dan Bahan Penelitian.....	68
3. 10. 1.	Alat Penelitian .....	69
3. 10. 2.	Bahan Penelitian.....	69

BAB IV HASIL DAN PEMBAHASAN.....	71
4.1 Pengumpulan Data Awal .....	71
4.1.1 Hasil Observasi .....	71
4.1.2 Hasil Wawancara.....	76
4.2 Pengujian Kinerja Sistem <i>Monolithic</i> .....	80
4.3 Analisis Hasil Pengujian Kinerja Sistem <i>Monolithic</i> .....	87
4.4 <i>Initiation: Analyze the Driving Forces</i> .....	88
4.5 <i>Planning: Understand the Legacy System</i> .....	88
4.5.1 Kode Program Aplikasi <i>Monolithic</i> Tracer Study UPI .....	89
4.5.2 Diagram Arsitektur.....	89
4.5.3 Diagram <i>Unified Modelling Language (UML)</i> .....	90
4.5.4 <i>Entity Relationship Diagram (ERD)</i> .....	94
4.5.5 Spesifikasi Sistem <i>Monolithic</i> Tracer Study UPI.....	94
4.6 <i>Planning: Decompose the Legacy System</i> .....	95
4.7 <i>Planning: Define the Microservices Architecture</i> .....	97
4.7.1 Diagram Arsitektur <i>Microservices</i> .....	97
4.7.2 Diagram <i>Unified Modelling Language (UML) Microservices</i> .....	99
4.7.3 <i>Entity Relationship Diagram (ERD) Microservices</i> .....	102
4.7.4 Spesifikasi Sistem <i>Microservices</i> .....	103
4.8 <i>Execution: Execute the Modernization</i> .....	104
4.8.1 <i>SIAC Service</i> .....	106
4.8.2 <i>Auth Service</i> .....	107
4.8.3 <i>Tracer Service</i> .....	108
4.8.4 <i>Post Service</i> .....	111
4.8.5 <i>Data Pipeline Service</i> .....	111
4.9 <i>Execution: Integrate the Microservices and the Legacy</i> .....	111



4. 9. 1	Spesifikasi API Gateway.....	112
4. 9. 2	<i>Containerization dan Deployment</i> .....	115
4. 10	<i>Execution: Verify and Validate the Microservices</i> .....	118
4. 10. 1	Integration Testing.....	118
4. 10. 2	Performance Testing.....	125
4. 11	<i>Monitoring: Monitor the Microservices</i> .....	133
4. 12	Analisis Perbandingan Hasil Pengujian Kinerja Sistem <i>Monolithic</i> dengan <i>Microservices</i> .....	135
BAB V KESIMPULAN DAN SARAN.....		137
5. 1	Kesimpulan.....	137
5. 2	Saran.....	137
DAFTAR PUSTAKA .....		139
LAMPIRAN.....		150

## DAFTAR GAMBAR

Gambar 2. 1	Peta Literatur .....	13
Gambar 2. 2	<i>Three-Tier Application Model</i> (Megargel et al., 2020) .....	26
Gambar 2. 3	Arsitektur <i>Monolithic</i> Sistem FTGO dalam Richardson (2018) .....	26
Gambar 2. 4	<i>The Single Process Monolith</i> (Newman, 2021).....	27
Gambar 2. 5	<i>The Modular Monolith</i> (Newman, 2021) .....	27
Gambar 2. 6	Arsitektur <i>Monolithic</i> pada Sistem Uber dalam (Phatak, 2022).....	29
Gambar 2. 7	Perbandingan <i>Monolithic</i> dan <i>Microservices</i> (Auer et al., 2021).....	31
Gambar 2. 8	Arsitektur <i>Microservices</i> Sistem FTGO dalam Richardson (2018) .	32
Gambar 2. 9	Diagram Konseptual <i>Domain-Driven Design</i> (Evans, 2003).....	37
Gambar 2. 10	<i>Roadmap Modernizing Legacy Systems with Microservices</i> (Wolfart et al., 2021).....	39
Gambar 2. 11	<i>API Gateway</i> dan <i>Integration Glue Code</i> (Hammad et al., 2023) .	43
Gambar 2. 12	<i>Direct Client-to-Microservices</i> (kiri) dan Melalui <i>API Gateway</i> (kanan) (Richardson & Smith, 2016) .....	47
Gambar 2. 13	<i>gRPC</i> pada <i>Product Service</i> (Indrasiri & Kuruppu, 2020).....	49
Gambar 2. 14	Brian Marick’s Testing Quadrant (Crispin & Gregory, 2009).....	52
Gambar 3. 1	Desain Penelitian.....	56
Gambar 4. 1	<i>Monolithic</i> – Fitur Isi Survei PKTS, Identitas.....	73
Gambar 4. 2	<i>Monolithic</i> – Fitur Isi Penilaian Alumni, Responden & Alumni.....	74
Gambar 4. 3	<i>Monolithic</i> – Halaman Daftar <i>Posts Public</i> .....	75
Gambar 4. 4	<i>Deploy</i> Aplikasi <i>Monolithic</i> Tracer Study UPI ke <i>VPS</i> .....	84
Gambar 4. 5	Implementasi Rancangan <i>Performance Testing Monolithic</i> .....	84
Gambar 4. 6	<i>Load Testing</i> Tracer Study <i>Monolithic</i> pada JMeter .....	85
Gambar 4. 7	Repository Source Code Tracer Study <i>Monolithic</i> .....	89
Gambar 4. 8	Diagram Arsitektur Tracer Study <i>Monolithic</i> .....	90
Gambar 4. 9	<i>Use Case Diagram</i> Tracer Study <i>Monolithic</i> .....	91
Gambar 4. 10	<i>Monolithic Sequence Diagram</i> Isi Survei PKTS.....	92
Gambar 4. 11	<i>Monolithic Sequence Diagram</i> Survei Penilaian Alumni.....	93
Gambar 4. 12	<i>Monolithic Sequence Diagram</i> Daftar <i>Posts Public</i> .....	93
Gambar 4. 13	<i>Monolithic Entity Relationship Diagram (ERD)</i> .....	94
Gambar 4. 14	Diagram <i>Domain-Driven Design (DDD)</i> .....	95

Gambar 4. 15 Diagram Arsitektur Tracer Study <i>Microservices</i> .....	98
Gambar 4. 16 <i>Microservices Class Diagram High-Level Overview</i> .....	100
Gambar 4. 17 <i>Microservices Sequence Diagram – Isi Survei PKTS</i> .....	101
Gambar 4. 18 <i>Microservices Sequence Diagram – Login Alumni</i> .....	101
Gambar 4. 19 <i>Microservices Sequence Diagram – Isi User Study</i> .....	102
Gambar 4. 20 Rancangan <i>ERD Tracer Study Microservices</i> .....	103
Gambar 4. 21 <i>Repository API Gateway</i> .....	112
Gambar 4. 22 <i>CORS Rule Configuration</i> .....	113
Gambar 4. 23 Daftar <i>Docker Images Microservices Tracer Study UPI</i> .....	116
Gambar 4. 24 <i>Deployment Diagram Microservices</i> .....	117
Gambar 4. 25 Daftar <i>Containers Hasil Deployment Microservices</i> .....	117
Gambar 4. 26 Penggunaan Resource Server pada Docker Containers .....	118
Gambar 4. 27 <i>Integration Test LoginAlumni()</i> IT.01-001/001 .....	120
Gambar 4. 28 Status <i>Deployed Services</i> pada <i>Server</i> .....	128
Gambar 4. 29 Antarmuka Awal Apache JMeter .....	128
Gambar 4. 30 Implementasi Rancangan <i>Performance Testing Microservices</i> ....	129
Gambar 4. 31 <i>Load Testing Tracer Study Microservices</i> pada JMeter .....	129
Gambar 4. 32 <i>Monitoring Post Service</i> pada <i>Server</i> .....	133
Gambar 4. 33 <i>Monitoring Tracer Service</i> pada <i>Server</i> .....	133
Gambar 4. 34 <i>Monitoring Auth Service</i> pada <i>Server</i> .....	134
Gambar 4. 35 <i>Monitoring API Gateway</i> pada <i>Server</i> .....	134

## DAFTAR TABEL

Tabel 2. 1 Daftar Penelitian Terkait .....	20
Tabel 3. 1 Daftar Pertanyaan Wawancara Pengumpulan Data Awal .....	65
Tabel 4. 1 Hasil <i>Trade-Off</i> pada <i>Driving Forces</i> Pengembangan <i>Microservices</i> .....	79
Tabel 4. 2 Skenario <i>Performance Testing</i> pada Sistem <i>Monolithic</i> .....	81
Tabel 4. 3 Skenario dan <i>Test Case Load Testing</i> .....	82
Tabel 4. 4 Skenario dan <i>Test Case Soak Testing</i> .....	83
Tabel 4. 5 Skenario dan <i>Test Case Stress Testing</i> .....	83
Tabel 4. 6 Hasil <i>Performance Load Testing</i> pada <i>Monolithic</i> .....	85
Tabel 4. 7 Hasil <i>Performance Soak Testing</i> pada <i>Monolithic</i> .....	86
Tabel 4. 8 Hasil <i>Performance Stress Testing</i> pada Sistem <i>Monolithic</i> .....	86
Tabel 4. 9 Daftar <i>gRPC Function Call Microservices</i> .....	104
Tabel 4. 10 Implementasi <i>gRPC Functions Auth Service</i> .....	108
Tabel 4. 11 Implementasi <i>gRPC Functions Tracer Service</i> .....	109
Tabel 4. 12 Daftar <i>Endpoints API Gateway</i> .....	113
Tabel 4. 13 Summary Hasil Integration Testing .....	119
Tabel 4. 14 <i>Integration Testing gRPC Function LoginAlumni</i> .....	120
Tabel 4. 15 <i>Integration Testing gRPC Function GetOrCreateResponden</i> .....	122
Tabel 4. 16 <i>Integration Testing gRPC Function ExportPKTSReport</i> .....	124
Tabel 4. 17 <i>Services IP Address &amp; Proto File</i> .....	125
Tabel 4. 18 Skenario <i>Performance Testing</i> pada Sistem <i>Microservices</i> .....	126
Tabel 4. 19 Hasil <i>Performance Load Testing</i> pada <i>Microservices</i> .....	130
Tabel 4. 20 Hasil <i>Performance Soak Testing</i> pada <i>Microservices</i> .....	131
Tabel 4. 21 Hasil <i>Performance Stress Testing</i> pada <i>Microservices</i> .....	131
Tabel 4. 22 Perbandingan Hasil <i>Load Testing Monolithic</i> dengan <i>Microservices</i> .....	135
Tabel 4. 23 Perbandingan Hasil <i>Soak Testing Monolithic</i> dengan <i>Microservices</i>	136

## DAFTAR LAMPIRAN

Lampiran 1. Surat Izin Penelitian.....	150
Lampiran 2. Pedoman Observasi .....	151
Lampiran 3. Lembar Hasil Observasi .....	152
Lampiran 4. Pedoman Wawancara.....	178
Lampiran 5. Lembar Hasil Wawancara.....	180
Lampiran 6. Instrumen PKTS Tracer Study Nasional .....	184
Lampiran 7. Artefak Perangkat Lunak Sistem Tracer Study Monolithic.....	189
Lampiran 8. Artefak Perancangan Sistem Tracer Study Microservices.....	204
Lampiran 9. Hasil Implementasi Perancangan Sistem <i>Microservices</i> .....	214
Lampiran 10. Hasil <i>Integration Testing</i> Sistem <i>Microservices</i> .....	280
Lampiran 11. Hasil Integrasi Sistem <i>Microservices</i> dengan <i>Client</i> .....	301
Lampiran 12. Surat Pernyataan Kelayakan Produk Penelitian .....	302

## DAFTAR PUSTAKA

- Abbas, R., Sultan, Z., & Bhatti, D. S. N. (2017). Comparative Analysis of Automated Load Testing Tools: Apache JMeter, Microsoft Visual Studio (TFS), LoadRunner, Siege. *2017 International Conference on Communication Technologies (ComTech)*, 39–44.
- Abbot, M. L., & Fisher, M. T. (2015). *The Art of Scalability, Second Edition* (Second). Addison-Wesley Professional.
- Abgaz, Y., Mccarren, A., Elger, P., Solan, D., Lapuz, N., Bivol, M., Jackson, G., Yilmaz, M., Buckley, J., & Clarke, P. (2023). Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review. *IEEE Transactions on Software Engineering*, 49(8), 4213–4242. <https://doi.org/10.1109/TSE.2023.3287297>
- Ahmadvand, M., & Ibrahim, A. (2017). Requirements Reconciliation for Scalable and Secure Microservice (De)composition. *Proceedings - 2016 IEEE 24th International Requirements Engineering Conference Workshops, REW 2016*, 68–73. <https://doi.org/10.1109/REW.2016.14>
- Akbar, R., & Mukhtar. (2020). Perancangan E-Tracer Study Berbasis Sistem Cerdas. *Sistem Informasi Dan Komputer*, 09, 8–12. <https://doi.org/10.32736/sisfokom.v9.i1.631>
- Al Hilmi, M. A., Muhamad, F. P. B., Cahyanto, K. A., & Mutahari, S. R. (2022). Penerapan Microservices pada Pengembangan Aplikasi Manajemen Kegiatan Rumah Tahfiz Qur'an Ulil Albab Kabupaten Indramayu. *IKRA-ITH Informatika: Jurnal Komputer Dan Informatika*, 6(2), 63–72.
- Alankar, B., Sharma, G., Kaur, H., Valverde, R., & Chang, V. (2020). Experimental Setup for Investigating the Efficient Load Balancing Algorithms on Virtual Cloud. *Sensors (Switzerland)*, 20(24), 1–26. <https://doi.org/10.3390/s20247342>
- Al-Debagy, O., & Martinek, P. (2018). A Comparative Review of Microservices and Monolithic Architectures. *18th IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, 149–154. <https://doi.org/10.1109/CINTI.2018.8928192>
- Amiri, M. J. (2018). Object-Aware Identification of Microservices. *Proceedings - 2018 IEEE International Conference on Services Computing, SCC 2018 - Part of the 2018 IEEE World Congress on Services*, 253–256. <https://doi.org/10.1109/SCC.2018.00042>
- Anderson, Ross. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems* (3rd ed.). John Wiley & Sons.
- Apache. (1999). *Apache JMeter*. The Apache Software Foundation. <https://jmeter.apache.org/>

- Arevalo, M., Escobar, C., Monasse, P., Monzon, N., & Colom, M. (2017). The IPOL Demo System: A Scalable Architecture of Microservices for Reproducible Research. In B. Kerautret, M. Colom, & P. Monasse (Eds.), *Kerautret, B., Colom, M., Monasse, P. (eds) Reproducible Research in Pattern Recognition* (Vol. 10214, pp. 3–16). Springer International Publishing. [https://doi.org/10.1007/978-3-319-56414-2\\_1](https://doi.org/10.1007/978-3-319-56414-2_1)
- Ashby, D., & Jensen, C. T. (2018). *APIs for Dummies* (3rd ed.). John Wiley & Sons, Inc. <http://www.wiley.com/go/permissions>.
- Auer, F., Lenarduzzi, V., Felderer, M., & Taibi, D. (2021). From Monolithic Systems to Microservices: An Assessment Framework. *Information and Software Technology*, 137. <https://doi.org/10.1016/j.infsof.2021.106600>
- Bagci, H., & Kara, A. (2016). A Lightweight and High Performance Remote Procedure Call Framework for Cross Platform Communication. *ICSOFT 2016 - Proceedings of the 11th International Joint Conference on Software Technologies, 1*, 117–124. <https://doi.org/10.5220/0005931201170124>
- Bakhtiar, M. I., & Latif, S. (2017). Tracer Study Alumni: Upaya Pengembangan Prodi Bimbingan Konseling Universitas Negeri Makassar. *Jurnal Kajian Bimbingan Dan Konseling*, 2(1), 32–40. <http://journal2.um.ac.id/index.php/jkbb>
- Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software*, 33(3), 42–52. <https://doi.org/10.1109/MS.2016.64>
- Balalaie, A., Heydarnoori, A., Jamshidi, P., Tamburri, D. A., & Lynn, T. (2018). Microservices Migration Patterns. *Software - Practice and Experience*, 48(11), 2019–2042. <https://doi.org/10.1002/spe.2608>
- Banijamali, A., Kuvaja, P., Oivo, M., & Jamshidi, P. (2020). Kuksa \*: Self-Adaptive Microservices in Automotive Systems. *Product-Focused Software Process Improvement, 21st International Conference, PROFES 2020*, 367–384. [https://doi.org/10.1007/978-3-030-64148-1\\_23](https://doi.org/10.1007/978-3-030-64148-1_23)
- Baresi, L., Garriga, M., & De Renzis, A. (2017). Microservices Identification Through Interface Analysis. In F. De Paoli, S. Schulte, & E. Broch Johnsen (Eds.), *European Conference on Service-Oriented and Cloud Computing* (Vol. 10465, pp. 19–33). Springer International Publishing. <https://doi.org/10.1007/978-3-319-67262-5>
- Blinowski, G., Ojdowska, A., & Przybylek, A. (2022). Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. *IEEE Access*, 10, 20357–20374. <https://doi.org/10.1109/ACCESS.2022.3152803>
- Bolanowski, M., Żak, K., Paszkiewicz, A., Ganzha, M., Paprzycki, M., Sowiński, P., Lacalle, I., & Palau, C. E. (2022). Efficiency of REST and gRPC Realizing Communication Tasks in Microservice-Based Ecosystems. *The 21st International Conference on Intelligent Software Methodologies, Tools, and Techniques (SoMeT 2022)*. <https://orcid.org/0000-0002->

Sekar Madu Kusumawardani, 2024

**RANCANG BANGUN ARSITEKTUR MICROSERVICES MENGGUNAKAN GOOGLE REMOTE PROCEDURE CALL (GRPC) APPLICATION PROGRAMMING INTERFACE (API) DAN METODE DOMAIN-DRIVEN DESIGN**

Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu

- Chaieb, M., & Saied, M. A. (2023). Automate Migration to Microservices Architecture using Machine Learning Techniques. *Journal of Systems and Software*, 1–60. <https://doi.org/10.48550/arXiv.2301.06508>
- Chen, R., Li, S., & Li, Z. (2017). From Monolith to Microservices: A Dataflow-Driven Approach. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 2017-December*, 466–475. <https://doi.org/10.1109/APSEC.2017.53>
- Clarke, P., O'Connor, R. V., & Leavy, B. (2016). A Complexity Theory Viewpoint on the Software Development Process and Situational Context. *Proceedings - International Conference on Software and System Process, ICSSP 2016*, 86–90. <https://doi.org/10.1145/2904354.2904369>
- Cojocaru, M., Uta, A., & Oprescu, A. M. (2019). MicroValid: A Validation Framework for Automatically Decomposed Microservices. *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom, 2019-December*, 78–86. <https://doi.org/10.1109/CloudCom.2019.00023>
- Cooksey, B. (2014). *An Introduction to APIs*. Zapier, Inc. <https://zapier.com/learn/apis>
- Cox, R., Griesemer, R., Pike, R., Taylor, I. L., & Thompson, K. (2022). The Go Programming Language and Environment. *Communications of the ACM*, 65(5), 70–78. <https://doi.org/10.1145/3488716>
- Crispin, L., & Gregory, J. (2009). *Agile Testing: A Practical Guide for Testers and Agile Teams*. Addison-Wesley. [www.XProgramming.com](http://www.XProgramming.com)
- D Dikti. (2012). *Buku Panduan Pusat Karir* (Vol. 2). Kementerian Riset Teknologi dan Pendidikan Tinggi Republik Indonesia.
- Dahri, F., Hanafi, A. M. El, Handoko, D., & Wulan, N. (2022). Implementation of Microservices Architecture in Learning Management System E-Course Using Web Service Method. *Sinkron*, 7(1), 76–82. <https://doi.org/10.33395/sinkron.v7i1.11229>
- De Camargo, A., Dos Santos Mello, R., Salvadori, I., & Siqueira, F. (2016). An Architecture to Automate Performance Tests on Microservices. *ACM International Conference Proceeding Series*, 422–429. <https://doi.org/10.1145/3011141.3011179>
- Di Francesco, P., Lago, P., & Malavolta, I. (2018). Migrating Towards Microservice Architectures: An Industrial Survey. *Proceedings - 2018 IEEE 15th International Conference on Software Architecture, ICOSA 2018*, 29–38. <https://doi.org/10.1109/ICOSA.2018.00012>
- Direktorat Jenderal Pendidikan Tinggi. (2011). *Tracer Study Kemendikbud: Tentang Tracer Study*. <https://Tracerstudy.Kemdikbud.Go.Id/>. <https://tracerstudy.kemdikbud.go.id/>



- 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 International Publishing. [https://doi.org/10.1007/978-3-319-67425-4\\_12](https://doi.org/10.1007/978-3-319-67425-4_12)
- Elgheriani, N. S., & Ahmed, N. A. S. (2022). Microservices Vs. Monolithic Architectures [The Differential Structure Between Two Architectures]. *MINAR International Journal of Applied Sciences and Technology*, 4(3), 500–514. <https://doi.org/10.47832/2717-8234.12.47>
- Escobar, D., Cardenas, D., Amarillo, R., Castro, E., Garces, K., Parra, C., & Casallas, R. (2016). Towards the Understanding and Evolution of Monolithic Applications as Microservices. *2016 XLII Latin American Computing Conference (CLEI)*, 1–11. <https://doi.org/10.1109/CLEI.2016.7833410>
- Evans, E. (2003). *Domain-Driven Design Tackling Complexity in the Heart of Software*. [www.domainlanguage.com](http://www.domainlanguage.com)
- Fan, C. Y., & Ma, S. P. (2017). Migrating Monolithic Mobile Application to Microservice Architecture: An Experiment Report. *2017 IEEE 6th International Conference on AI and Mobile Services (AIMS)*, 109–112. <https://doi.org/10.1109/AIMS.2017.23>
- Ferdinand, J., Syahrina, A., & Musnansyah, A. (2021). Perancangan Arsitektur Perangkat Lunak Microservices pada Aplikasi Open Library Telkom University Menggunakan gRPC. *E-Proceeding of Engineering*, 8, 9543–9550.
- Fowler, M. (2004, June). *Strangler Fig Application*. <https://martinfowler.com/Bliki/StranglerFigApplication.html>
- Fritzsich, J., Bogner, J., Zimmermann, A., & Wagner, S. (2019). From Monolith to Microservices: A Classification of Refactoring Approaches. *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, 128–141. <https://doi.org/10.48550/arXiv.1807.10059>
- Google. (2015a). *gRPC*. <https://grpc.io>
- Google. (2015b). *Protocol Buffer Documentation*. <https://protobuf.dev/>
- Guamán, D., Yaguachi, Lady, Samanta, C. C., Danilo, J. H., & Soto, F. (2018). Performance Evaluation in the Migration Process from a Monolithic Application to Microservices. *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–8. <https://doi.org/10.23919/CISTI.2018.8399148>
- Guo, D., Wang, W., Zhang, J., Xiang, Q., Huang, C., Chang, J., & Zhang, L. (2016). Cloudware: An Emerging Software Paradigm for Cloud Computing. *ACM International Conference Proceeding Series, 18-September-2016*, 1–10. <https://doi.org/10.1145/2993717.2993718>

Sekar Madu Kusumawardani, 2024

**RANCANG BANGUN ARSITEKTUR MICROSERVICES MENGGUNAKAN GOOGLE REMOTE PROCEDURE CALL (GRPC) APPLICATION PROGRAMMING INTERFACE (API) DAN METODE DOMAIN-DRIVEN DESIGN**

Universitas Pendidikan Indonesia | [repository.upi.edu](http://repository.upi.edu) | [perpustakaan.upi.edu](http://perpustakaan.upi.edu)

- Gysel, M., Kölbener, L., Giersche, W., & Zimmermann, O. (2016). Service Cutter: A Systematic Approach to Service Decomposition. *Aiello, M., Johnsen, E., Dustdar, S., Georgievski, I. (Eds) Service-Oriented and Cloud Computing. ESOC 2016. Lecture Notes in Computer Science, 9846 LNCS*, 185–200. [https://doi.org/10.1007/978-3-319-44482-6\\_12](https://doi.org/10.1007/978-3-319-44482-6_12)
- Hammad, H., Sahmoud, T., & Ghazala, A. A. R. A. (2023). Convert Monolithic Application to Microservice Application. *ArXiv*. <https://doi.org/10.48550/arXiv.2306.08851>
- Hermawan, I., & Suharnomo, S. (2020). Information Technology as a Strategic Resource in Encouraging Organizational Change Readiness through the Role of the Human Capital Effectiveness. *Jurnal Dinamika Manajemen, 11(2)*, 242–254. <https://doi.org/10.15294/jdm.v11i2.23700>
- Hoffer, J. A., Ramesh, V. (Venkataraman), & Topi, H. (2019). *Modern Database Management* (13th ed.). Pearson Education.
- Hosea, E., Novianus Palit, H., & Dewi, L. P. (2021). Fault Tolerance pada Microservice Architecture dengan Circuit Breaker dan Bulkhead Pattern. *Jurnal Infra, 9*.
- Husaini, M., & IAIN, R. I. (2014). Pemanfaatan Teknologi Informasi dalam Bidang Pendidikan (E-Education). *Jurnal Mikrotik, 2(1)*.
- I. J. Munezero, D. -T. Mukasa, B. Kanagawa, & J. Balikuddembe. (2018). Partitioning microservices: A domain engineering approach. *2018 IEEE/ACM Symposium on Software Engineering in Africa (SEiA)*, 43–49. <https://doi.org/10.1145/3195528.3195535>
- Imran, M., Sasudin, M., Rusli, H. M., & Kama, N. (2022). Monolith Application to Microservices Model Driven Analysis Migration: State-of-The-Art Techniques Article history. *Open International Journal of Informatics (OIJI, 10(2)*). <https://doi.org/10.11113/oiji2022.10n2.231>
- Indrasiri, K., & Kuruppu, D. (2020). *gRPC: Up & Running (Building Cloud Native Applications with Go and Java for Docker and Kubernetes)*. O'Reilly Media, Inc.
- Ismail, A., Ananta, A. Y., Arief, S. N., & Hamdana, E. N. (2023). Performance Testing Sistem Ujian Online Menggunakan JMeter pada Lingkungan Virtual. *JIP (Jurnal Informatika Polinema), 9(2)*, 159–164.
- Janes, A., & Russo, B. (2019). Automatic performance monitoring and regression testing during the transition from monolith to microservices. *2019 IEEE 30th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 163–168. <https://doi.org/10.1109/ISSREW.2019.00067>
- Joko, B. S. (2010). Sistem Informasi Manajemen Perguruan Tinggi Dalam Bidang Pendataan Pendidikan Tinggi. *Jurnal Pendidikan Dan Kebudayaan, 16(2)*, 146–156. <https://doi.org/10.24832/jpnk.v16i2.442>

- Kalske, M., Mäkitalo, N., & Mikkonen, T. (2018). Challenges When Moving from Monolith to Microservice Architecture. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10544 LNCS, 32–47. [https://doi.org/10.1007/978-3-319-74433-9\\_3](https://doi.org/10.1007/978-3-319-74433-9_3)
- Karwowski, W., Rusek, M., Dwornicki, G., & Orłowski, A. (2018). Swarm Based System for Management of Containerized Microservices in a Cloud Consisting of Heterogeneous Servers. *Advances in Intelligent Systems and Computing*, 655, 262–271. [https://doi.org/10.1007/978-3-319-67220-5\\_24](https://doi.org/10.1007/978-3-319-67220-5_24)
- Khoirunnisa, L. (2019). *Rancang Bangun Sistem E-Learning Berbasis Microservices dan Domain Driven Design (Studi Kasus Probistek UIN Maulana Malik Ibrahim Malang)*. UIN Maulana Malik Ibrahim Malang.
- Kim, S.-H., Lee, Y., & Kim, J.-S. (2007). FlexRPC: A Flexible Remote Procedure Call Facility for Modern Cluster File Systems. *2007 IEEE International Conference on Cluster Computing*, 275–284. <https://doi.org/10.1109/CLUSTER.2007.4629241>
- Kore, P. P., Lohar, M. J., Surve, M. T., & Jadhav, S. (2022). API Testing Using Postman Tool. *International Journal for Research in Applied Science and Engineering Technology*, 10(12), 841–843. <https://doi.org/10.22214/ijraset.2022.48030>
- Koschel, A., Astrova, I., & Dotterl, J. (2017). Making the Move to Microservice Architecture. *International Conference on Information Society (i-Society)*, 71–79. <https://doi.org/10.23919/i-Society.2017.8354675>
- Laudon, K. C., & Laudon, J. P. (2012). *Management Information System: Managing Digital Firm* (12th ed.). Pearson Education.
- Levcovitz, A., Terra, R., & Valente, M. T. (2016, May 10). Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems. *3rd Brazilian Workshop on Software Visualization, Evolution and Maintenance*. <https://doi.org/10.48550/arXiv.1605.03175>
- Lewis, J., & Fowler, M. (2014, March 25). *Microservices: A Definition of This New Architectural Term*. [https://Martinfowler.Com/Articles/Microservices.Html](https://martinfowler.com/articles/microservices.html). <https://martinfowler.com/articles/microservices.html>
- Li, C. Y., Ma, S. P., & Lu, T. W. (2020). Microservice Migration Using Strangler Fig Pattern: A Case Study on the Green Button System. *Proceedings - 2020 International Computer Symposium, ICS 2020*, 519–524. <https://doi.org/10.1109/ICS51289.2020.00107>
- Li, S., Zhang, H., Jia, Z., Li, Z., Zhang, C., Li, J., Gao, Q., Ge, J., & Shan, Z. (2019). A Dataflow-Driven Approach to Identifying Microservices from Monolithic Applications. *Journal of Systems and Software*, 157. <https://doi.org/10.1016/j.jss.2019.07.008>
- Ma, S. P., Li, C. Y., Lee, W. T., & Lee, S. J. (2022). Microservice Migration Using Strangler Fig Pattern and Domain-Driven Design. *Journal of Information Sekar Madu Kusumawardani*, 2024
- RANCANG BANGUN ARSITEKTUR MICROSERVICES MENGGUNAKAN GOOGLE REMOTE PROCEDURE CALL (GRPC) APPLICATION PROGRAMMING INTERFACE (API) DAN METODE DOMAIN-DRIVEN DESIGN**
- Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu

- Science and Engineering*, 38(6), 1285–1303.  
[https://doi.org/10.6688/JISE.202211\\_38\(6\).0010](https://doi.org/10.6688/JISE.202211_38(6).0010)
- Martin, R. C. (2014). *Agile Software Development, Principles, Patterns, and Practices* (1st ed., Vol. 1). Pearson Education Limited.
- Maulana, H. (2023). Refactoring Arsitektur Microservice pada Aplikasi Management Information System of LP3I Menggunakan Strangler Pattern. *Jurnal Explora Informatika*, 11, 140–148.  
<https://doi.org/10.30864/eksplora.v11i2.888>
- Mazlami, G., Cito, J., & Leitner, P. (2017). Extraction of Microservices from Monolithic Software Architectures. *IEEE 24th International Conference on Web Services (ICWS 2017)*, 524–531. <https://doi.org/10.1109/ICWS.2017.61>
- Megargel, A., Shankararaman, V., & Walker, D. K. (2020). Migrating from Monoliths to Cloud-Based Microservices: A Banking Industry Example. In *Research Collection School Of Information Systems* (pp. 85–108). Springer.  
[https://doi.org/10.1007/978-3-030-33624-0\\_4](https://doi.org/10.1007/978-3-030-33624-0_4)
- Meier, J. D., Farre, C., Banshode, P., Barber, S., & Rea, D. (2007). *Performance Testing Guidance for Web Applications* (1st ed.). Microsoft Press.
- Mendonca, N. C., Box, C., Manolache, C., & Ryan, L. (2021). The Monolith Strikes Back: Why Istio Migrated from Microservices to a Monolithic Architecture. *IEEE Software*, 38(5), 17–22. <https://doi.org/10.1109/MS.2021.3080335>
- Mili, A., & Tchien, F. (2015). *Software Testing: Concepts and Operations* (Vol. 1). John Wiley & Sons. [www.allitebooks.com](http://www.allitebooks.com)
- Mithas, S., & Krishnan, M. S. (2008). Human Capital and Institutional Effects in the Compensation of Information Technology Professionals in the United States. *Management Science*, 54(3), 415–428.  
<https://doi.org/10.1287/mnsc.1070.0778>
- Molyneaux, I. (2009). *The Art of Application Performance Testing* (1st ed., Vol. 1). O'Reilly Media.
- Mufrizal, R., & Indarti, D. (2019). Refactoring Arsitektur Microservice Pada Aplikasi Absensi PT. Graha Usaha Teknik. *Jurnal Nasional Teknologi Dan Sistem Informasi*, 5(1), 57–68. <https://doi.org/10.25077/teknosi.v5i1.2019.57-68>
- Myers, G. J., Badgett, T., & Sandler, C. (2012). *The Art of Software Testing* (3rd ed.). John Wiley & Sons, Inc.
- Namiot, D., & Sneps-Sneppe, M. (2014). On Micro-services Architecture. *International Journal of Open Information Technologies*, 2(9), 2307–8162.  
<https://www.researchgate.net/publication/265292970>
- Newman, S. (2020). *Monolith to Microservices Evolutionary Patterns to Transform Your Monolith*. O'Reilly Media.

- Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems* (2nd ed.). O'Reilly Media.
- Nizam. (2021). *Membangun Sistem Pendidikan Tinggi Indonesia 4.0*.
- Noviyantono, E., & Aidil. (2012). Integration System of Web Based and SMS Gateway for Information System of Tracer Study. *International Conference on Engineering and Technology Development (ICETD)*, 12(1).
- Park, J., Moon, M., & Keunhyuk, K. (2019). Approach to Identify Microservices Based on Analysis Class Model. *International Journal of Advanced Science and Technology*, 28(4), 8–14. <http://sersec.org/journals/index.php/IJAST/article/view/289>
- Phatak, J. J. (2022). An Overview of Microservice Architecture Impact in Terms of Scalability and Reliability in E-Commerce: A Case Study on Uber and Otto.De. *International Journal of Advanced Research in Science, Communication and Technology (IJARST)*, 2(1). <https://doi.org/10.48175/568>
- Poerwanto, Sisbintari, I., & Suhartono. (2013). Transformasi Organisasi Basis Peningkatan Sumber Daya Manusia dalam Memperkuat Daya Saing. *Jurnal Al-Azhar Indonesia Seri Pranata Sosial*, 2(2), 119–132.
- Posta, C. (2016). *Microservices for Java Developers: A Hands-On Introduction to Frameworks and Containers* (1st ed.). O'Reilly Media.
- Pradeep, S., & Sharma, Y. K. (2019). A Pragmatic Evaluation of Stress and Performance Testing Technologies for Web Based Applications. *Proceedings - 2019 Amity International Conference on Artificial Intelligence, AICAI 2019*, 399–403. <https://doi.org/10.1109/AICAI.2019.8701327>
- Prasojo, L. D. (2009). Sistem Manajemen Perguruan Tinggi Modern. *Dinamika Pendidikan*, 16(1), 98–109. <https://journal.uny.ac.id/index.php/dinamika-pendidikan/article/view/4121>
- Pressman, R. F. (2010). *Software Engineering: A Practitioner's Approach* (7th ed.). McGraw-Hill. [www.mhhe.com/pressman](http://www.mhhe.com/pressman).
- Rademacher, F., Sorgalla, J., & Sachweh, S. (2018). Challenges of Domain-Driven Microservice Design A Model-Driven Perspective. *IEEE Software*, 35(3), 36–43. <https://doi.org/10.1109/MS.2018.2141028>
- Radhiyan, M. F. (2020). *Analisis dan Desain Arsitektur Microservices dengan GraphQL Sebagai API Gateway untuk Sistem Informasi Akademik AIS UIN Jakarta (Studi Kasus : AIS untuk Mahasiswa)*. UIN Syarif Hidayatullah.
- Rajesh RV. (2016). *Spring Microservices: Build Scalable Microservices with Spring, Docker, and Mesos*. Packt Publishing.
- Ramachandran, A. T., R. A., G.S., M., R., R., K., B., & Parmar, M. (2021). Understanding Migration from Monolithic to Microservice Architecture and its Challenges. *International Journal of Scientific Research and Engineering Development*, 4, 1742–1752. [www.ijrsred.com](http://www.ijrsred.com)

Sekar Madu Kusumawardani, 2024

**RANCANG BANGUN ARSITEKTUR MICROSERVICES MENGGUNAKAN GOOGLE REMOTE PROCEDURE CALL (GRPC) APPLICATION PROGRAMMING INTERFACE (API) DAN METODE DOMAIN-DRIVEN DESIGN**

Universitas Pendidikan Indonesia | [repository.upi.edu](http://repository.upi.edu) | [perpustakaan.upi.edu](http://perpustakaan.upi.edu)

- Reiser, R. A., & Dempsey, J. V. (2018). *Trends and Issues in Instructional Design and Technology* (4th ed.). Pearson Education.
- Rencana Strategis (RENSTRA) Universitas Pendidikan Indonesia Tahun 2021-2025, Peraturan Majelis Wali Amanat Universitas Pendidikan Indonesia Nomor 04 Tahun 2020 Tentang Rencana Strategis Universitas Pendidikan Indonesia Tahun 2021-2025 (2021).
- Rezaldy, M., Asror, I., & Sardi, I. L. (2017). Desain dan Analisis Arsitektur Microservices Pada Sistem Informasi Akademik Perguruan Tinggi Dengan Pendekatan Architecture Tradeoff Analysis Method (ATAM) (Studi Kasus: iGracias Universitas Telkom). *E-Proceeding of Engineering*, 4(2).
- Richardson, C. (2018). *Microservices Patterns: With Examples in Java* (1st ed.). Manning Publications Co.
- Richardson, C. (2020). *Monolithic Architecture Pattern*. <https://microservices.io/patterns/monolithic.html>
- Richardson, C., & Smith, F. (2016). *Microservices: From Design to Deployment*. Nginx. <https://www.nginx.com/blog/microservices-from-design-to-deployment-ebook-nginx/>
- Riyanto, Hermadi, I., & Nurhadryani, Y. (2023). Analisis Uji Performa Aplikasi Dari Hasil Implementasi Refactoring Arsitektur Monolitik Ke Mikroservis dengan Decomposition dan Strangler Pattern. *Jurnal Sistem Cerdas*, 6(3), 189–203.
- Rizki, M., Fajar, A. N., & Retnowardhani, A. (2020). Microservices Architecture Design: Proposed for Online HealthCare. *International Journal of Emerging Trends in Engineering Research*, 8(4), 1040–1046. <https://doi.org/10.30534/ijeter/2020/14842020>
- Samsiah, S., Marlina, E., & Ardi, H. A. (2018). Pengaruh Knowledge Management Dan Teknologi Informasi Terhadap Keunggulan Bersaing Dan Kinerja Universitas. *Jurnal Manajemen*, 22(2), 154–167. <https://doi.org/10.24912/jm.v22i2.356>
- Schmidt, R. A., & Thiry, M. (2020). Microservices Identification Strategies: A Review Focused on Model-Driven Engineering and Domain Driven Design Approaches. *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–6. <https://doi.org/10.23919/CISTI49556.2020.9141150>
- Schomburg, H. (2003). *Handbook for Graduate Tracer Studies* (2nd ed.). InWent. [www.uni-kassel.de/incher](http://www.uni-kassel.de/incher)
- Singh, V., & Peddoju, S. K. (2017). Container-based Microservice Architecture for Cloud Applications. *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 847–852. <https://doi.org/10.1109/CCAA.2017.8229914>

- Siti Rochimah, & Bintang Nuralamsyah. (2023). Decomposing Monolithic to Microservices: Keyword Extraction and BFS Combination Method to Cluster Monolithic's Classes. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 7(2), 263–270. <https://doi.org/10.29207/resti.v7i2.4866>
- Sivathanu, M., Arpaci-Dusseau, A. C., & Arpaci-Dusseau, R. H. (2002). Evolving RPC for Active Storage. *ACM SIGOPS Operating Systems Review*, 36, 264–276. <https://doi.org/10.1145/635508.605425>
- Sommerville, I. (2011). *Software Engineering* (9th ed.). Pearson Education.
- Steinegger, R. H., Giessler, P., Hippchen, B., & Abeck, S. (2017). Overview of a Domain-Driven Design Approach to Build Microservice-Based Applications. *The Third International Conference on Advances and Trends in Software Engineering (SOFTENG 2017)*.
- Supratman, Defit, S., & Vitriani. (2019). Indeks Kesiapan Perguruan Tinggi dalam Mengimplementasikan Smart Campus. *Jurnal Teknologi Informasi Dan Ilmu Komputer (JTIK)*, 6(3), 267–276. <https://doi.org/10.25126/jtiik.20196986>
- Suryotrisongko, H. (2017). Arsitektur Microservice untuk Resiliensi Sistem Informasi. *Jurnal SISFO*, 6(2), 235–250.
- Swarnalatha, K. S., Mallya, A., Mukund, G., & Ujwal Bharadwaj, R. (2022). Solving Problems of Large Codebases Uber's Approach Using Microservice Architecture. *Emerging Research in Computing, Information, Communication and Applications*, 653–662. [https://doi.org/10.1007/978-981-19-5482-5\\_57](https://doi.org/10.1007/978-981-19-5482-5_57)
- Syafiq, A. (2017). *Konsep dan Implementasi Tracer Study*. Lembaga Layanan Pendidikan Tinggi Wilayah IV. [https://www.ildikti4.or.id/wp-content/uploads/2017/06/Ahmad-Syafiq\\_KONSEP-DAN-IMPLEMENTASI-TS-2017.pdf](https://www.ildikti4.or.id/wp-content/uploads/2017/06/Ahmad-Syafiq_KONSEP-DAN-IMPLEMENTASI-TS-2017.pdf)
- Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. *IEEE Cloud Computing*, 22–32. <https://doi.org/10.1109/MCC.2017.4250931>
- Taibi, D., & Systä, K. (2019). From Monolithic Systems to Microservices: A Decomposition Framework Based on Process Mining. *9th International Conference on Cloud Computing and Services Science*, 153–164. <https://doi.org/10.5220/0007755901530164>
- Tanuska, P., Vlkovic, O., & Spendla, L. (2012). The Usage of Performance Testing for Information Systems. *International Journal of Computer Theory and Engineering*, 144–147. <https://doi.org/10.7763/ijcte.2012.v4.439>
- Tapia, F., Mora, M. ángel, Fuertes, W., Aules, H., Flores, E., & Toulkeridis, T. (2020). From Monolithic Systems to Microservices: A Comparative Study of Performance. *Applied Sciences (Switzerland)*, 10(17), 1–35. <https://doi.org/10.3390/app10175797>
- Undang-Undang Nomor 12 Tahun 2012 Tentang Perguruan Tinggi, Pub. L. No. 12, Sekretariat Negara (2012).

Sekar Madu Kusumawardani, 2024

**RANCANG BANGUN ARSITEKTUR MICROSERVICES MENGGUNAKAN GOOGLE REMOTE PROCEDURE CALL (GRPC) APPLICATION PROGRAMMING INTERFACE (API) DAN METODE DOMAIN-DRIVEN DESIGN**

Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu

- UPI. (2021). *Tracer Study UPI*. <https://tracerstudy.upi.edu>
- Vale, G., Correia, F. F., Guerra, E. M., Rosa, T. de O., Fritzsche, J., & Bogner, J. (2022). Designing Microservice Systems Using Patterns: An Empirical Study on Quality Trade-Offs. *IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, 67–79. <https://doi.org/10.1109/ICSA53651.2022.00015>
- Villamizar, M., Garces, O., Castro, H., Verano, M., Salamanca, L., & Casallas, R. (2015). Evaluating the Monolithic And the Microservice Architecture Pattern to Deploy Web Applications in the Cloud. *10th Computing Colombian Conference (10CCC)*, 583–590. <https://doi.org/10.1109/ColumbianCC.2015.7333476>
- Vural, H., & Koyuncu, M. (2021). Does Domain-Driven Design Lead to Finding the Optimal Modularity of a Microservice? *IEEE Access*, 9, 32721–32733. <https://doi.org/10.1109/ACCESS.2021.3060895>
- Wang, J., & Wu, J. (2019). Research on Performance Automation Testing Technology Based on JMeter. *2019 International Conference on Robots and Intelligent System (ICRIS)*, 55–58. <https://doi.org/10.1109/ICRIS.2019.00023>
- Widajanti, E. (2008). Peran Teknologi Informasi untuk Mencapai Keunggulan Kompetitif. *Jurnal Akuntansi Dan Sistem Teknologi Informasi*, 6(1), 60–71.
- Wolfart, D., Assunção, W. K. G., Da Silva, I. F., Domingos, D. C. P., Schmeing, E., Villaca, G. L. D., & Paza, D. D. N. (2021). Modernizing Legacy Systems with Microservices: A Roadmap. *ACM International Conference Proceeding Series*, 149–159. <https://doi.org/10.1145/3463274.3463334>