Al Qalam: Jurnal Ilmiah Keagamaan dan Kemasyarakatan
https://jurnal.stiq-amuntai.ac.id/index.php/al-qalam
P-ISSN: 1907-4174; E-ISSN: 2621-0681
DOI : 10.35931/aq.v16i1. 862

PERANCANGAN ARSITEKTUR BACKEND MICROSERVICE PADA STARTUP CAMPAIGN.COM

Zaky Riko Virgiawan[#1], Harwikarya[*2]
[#]***Teknik Informatika, Fakultas Ilmu Komputer, Universitas Mercubuana***
[1]**41517120048@student.mercubuana.ac.id**
[2]harwikarya@mercubuana.ac.id

Abstrak

Dalam memulai pengembangan perangkat lunak, salah satu poin terpenting adalah menentukan arsitektur teknologi sesuai dengan analisa kebutuhan produk. Sehingga nantinya dapat berjalan selaras bersamaan dengan visi produk tersebut. Namun, seiring berjalannya waktu dengan user dan data yang semakin bertambah perlu ada perombakan arsitektur kembali, dengan membangun pondasi yang lebih kokoh. Hal ini dialami oleh perusahaan *startup* sosial Campaign.com. Campaign.com merupakan sebuah aplikasi untuk mengambil aksi sosial yang dapat menyalurkan donasi dari sponsor. Desain sistem yang dibangun di campaign.com masih menggunakan arsitektur monolitik. Antarmuka pengguna, pemrosesan logika, dan akses data digabungkan menjadi satu program dan ditempatkan dalam satu basis data. Namun hal ini membuat aplikasi sering mengalami *bugs* dan *server downtime*. Dengan masalah yang sedang dihadapi ini penulis mencoba melakukan eksperimen dengan merancang dan menganalisa sistem arsitektur *backend*. Tujuan yang ingin dicapai dari perancangan arsitektur ini adalah mendeskripsikan proses penyederhanaan dari sebuah sistem arsitektur *backend microservice* agar mudah dalam pemeliharaan sistem dan penggunaan algoritma *Docker* yang akan membuat sistem menjadi efisien terhadap penulisan kode.
Kata kunci— Arsitektur, microservice, docker, integrasi

PENDAHULUAN

Campaign.com merupakan sebuah aplikasi untuk mengambil aksi sosial yang mana dapat menyalurkan donasi dari sponsor. Di campaign.com desain sistem yang dibangun masih menggunakan arsitektur monolitik. Antarmuka pengguna, pemrosesan logika, dan akses data digabungkan menjadi satu program dan ditempatkan dalam satu basis data, namun hal ini membuat aplikasi sering mengalami bugs dan server downtime, dengan masalah yang sedang dihadapi ini penulis mencoba melakukan perbandingan dan menganalisa sistem arsitektur backend yang digunakan oleh Gojek.

Gojek sukses menjadi startup pertama asal Indonesia yang mendapat gelar unicorn. Tidak butuh waktu lama bagi perusahaan aplikasi PT Go-Jek Indonesia untuk mendapatkan tempat dihati masyarakat, aplikasi karya anak bangsa ini memberikan dampak yang luar biasa untuk

memajukan ekonomi kreatif, bagaimana tidak setelah namanya mencuat di Indonesia banyak sekali perusahaan startup bermunculan, mulai dari ide bisnis yang mirip-mirip sampai yang tidak kalah inovatifnya dengan Gojek. Salah satu faktor kesuksesan Gojek adalah teknologi yang digunakan untuk model bisnis yang mereka bangun. Di Dalam aplikasi Gojek kita dapat memesan makanan, jasa antar jemput, mengirim barang dan masih banyak lagi fitur lainya, hal inilah yang membuat penulis tertarik untuk melakukan penelitian teknologi informasi yang gojek terapkan.[1]

Wildan G. Budhi seorang Tech and Macro Economics Enthusiasm melalui websitenya di https://wildangbudhi.medium.com, telah menulis artikel tentang arsitektur microservice yang digunakan oleh gojek untuk membangun sistem backend mereka. Dalam tulisanya ada satu kalimat menarik "1 App, 1 Server, Many Product" ini yang menjadi identitas utama microservice, satu aplikasi, satu server namun ada beberapa produk di dalamnya.[2]

Yang membedakan dengan penelitian sebelumnya atau referensi yang penulis cantumkan adalah adanya integrasi data dari sistem lama menuju sistem baru. Jadi bukan sekedar implementasi tetapi juga melakukan dekomposisi arsitektur basis data. Penelitian akan dilaksanakan secara bertahap sesuai dengan rancangan yang telah disusun. Mulai dari studi literatur sampai akhirnya implementasi pada subjek studi kasus.[3] [4]

LANDASAN TEORI

A. Microservice

Microservice adalah teknik pengembangan perangkat lunak yang mengatur aplikasi sebagai kumpulan layanan yang digabungkan secara longgar. Kopling adalah tingkat saling ketergantungan antara modul perangkat lunak. Jadi kita dapat mengatakan Microservice adalah tempat aplikasi atau program dibagi menjadi banyak modul kecil yang saling ketergantungan. Penggunaan satu aplikasi dan satu server dengan teknik Microservices dapat membagi semua program menjadi program atau aplikasi independen, baik itu di server terpisah atau di satu server tetapi menggunakan Docker Container.

B. Docker

Docker adalah teknologi virtualisasi sistem operasi berbasis Container untuk membangun, menguji, dan menyebarkan aplikasi terdistribusi dalam lingkungan yang terisolasi. Docker tidak

[1] Mehdi Allahyari dkk., "A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques," 2017.

[2] Calvin Seviro Bima Sakti and Indra Hermawan, "Implementasi Arsitektur Microservice Pada Back End Sistem Informasi Atlantas Berbasis Website," *Jurnal Teknologi Terpadu* 6, no. 2 (2020).

[3] Rakhmi Khalida, Adi Muhajirin, and Siti Setiawati, "Teknis Kerja Docker Container untuk Optimalisasi Penyebaran Aplikasi," *PIKSEL : Penelitian Ilmu Komputer Sistem Embedded and Logic* 7, no. 2 (September 23, 2019): 167–76.

[4] Astie Darmayantie, "Desain Sistem Terfederasi Dengan Pendekatan Microservice Architecture Pada Kasus Studi Sistem Pelaporan Pajak," *Jurnal Ilmiah Informatika Komputer* 25, no. 1 (2020): 50–63.

membangun mesin virtual sendiri, sehingga lebih hemat memori, processor dan storage. Docker menyediakan virtualisasi ringan dengan overhead hampir nol.[5]

C. Swagger

Swagger merupakan tools yang digunakan untuk mendeskripsikan struktur API sehingga mudah untuk dibaca. Dengan membaca struktur API, secara otomatis membuat pendokumentasian menjadi interaktif karena dapat diuji secara langsung dan memiliki user interface. Keuntungan lainnya dari menggunakan swagger dapat digunakan secara bersamaan.

Gambar 1. User Management

Dokumentasi sangat penting sebagai alat komunikasi antara Back-end Developer dan Front-end Developer. Dengan dokumentasi yang readable, client akan lebih produktif.

D. Monolith

Aplikasi Monolitik adalah perangkat lunak di mana komponen yang berbeda (seperti otorisasi, logika bisnis, modul notifikasi, dll.) digabungkan menjadi satu program dari satu platform.[6]

METODE PENELITIAN

Dalam membangun arsitektur backend microsrvice untuk artikel jurnal ilmiah, dilakukan beberapa tahapan analisis. Berikut tahapan-tahapan yang telah dilakukan.

[5] Sakti dan Hermawan, "Implementasi Arsitektur Microservice Pada Back End Sistem Informasi Atlantas Berbasis Website."

[6] Konrad Gos and Wojciech Zabierowski, "The Comparison of Microservice and Monolithic Architecture," in *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)* (2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Lviv, Ukraine: IEEE, 2020), 150–53.

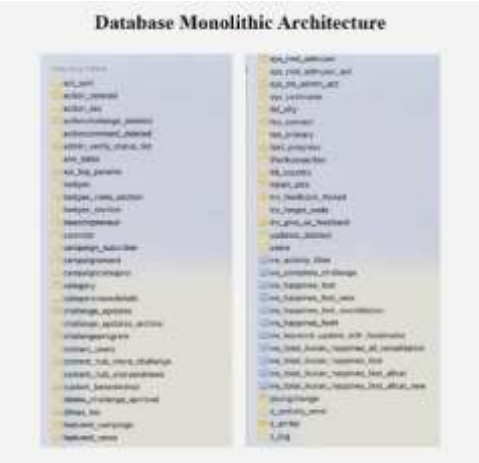
Gambar 2. Diagram Alir Analisis Model

Adapun penjelasannya, pada tahapan pertama yaitu dekomposisi modul sistem basis data yang sedang berjalan. Kemudian membagi menjadi beberapa modul yang lebih kecil termasuk menghubungkan dari modul satu ke modul yang lain. Memisahkan *environment* menjadi tiga tahap yaitu; development, staging, production. Tahapan berikutnya, *bugfixing* memperbaiki masalah-masalah yang ditemukan pada saat testing.

A. Dekomposisi Database

Karya ilmiah ini berawal dari mengintegrasikan sistem monolitik menjadi sistem microservice dengan merombak arsitektur backend. Pada tahapan dekomposisi database, kami memisahkan tabel-tabel menjadi sebuah modul seperti yang telah dibahas Francisco Ponce dkk.[7] dalam arsitektur monolitik, semua fungsionalitas dienkapsulasi menjadi satu webservice tunggal, sehingga modulnya tidak dapat dieksekusi secara independen dan semua logika berjalan menangani permintaan dalam satu proses.

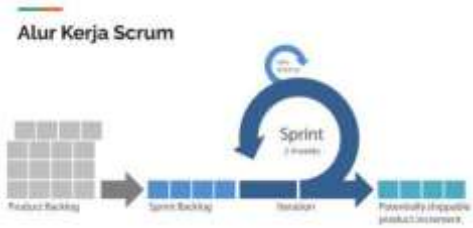
[7] Francisco Ponce, Gaston Marquez, and Hernan Astudillo, "Migrating from Monolithic Architecture to Microservices: A Rapid Review," in *2019 38th International Conference of the Chilean Computer Science Society (SCCC)* (2019 38th International Conference of the Chilean Computer Science Society (SCCC), Concepcion, Chile: IEEE, 2019), 1–7.



Gambar 3. Database Monolithic Architecture

B. Scrum Agile Software Development

Scrum adalah salah satu bentuk metode agile development sistem yang digunakan oleh penulis dalam mengerjakan proyek karya ilmiah ini. Pada sebelumnya di campaign.com menggunakan metode waterfall dalam pengembangan software dan memutuskan berganti ke Metode Scrum karena alasan "bagaimana memberikan value secepatnya kepada user & stakeholder" sesuai dengan namanya agile yaitu lincah yang terus menerus merespon perubahan.



Gambar 4. Alur Kerja Scrum

C. Rebuild Function API

Setelah menentukan metode untuk software development life cycle langkah berikutnya adalah rebuild function APi yang sebelumnya menuju sistem microservice. Ini adalah salah satu poin terpenting pada eksperimen yang dilakukan penulis karena disinilah tahapan menulis ulang kode lama menjadi baru dengan merubah Bahasa pemrograman, cara penulisan fungsi-fungsi dan tentunya backend arsitektur.

D. Message Broker

Setelah proses rebuild selesai tahapan berikutnya adalah membuat message broker penjelasan singkatnya adalah message broker inilah yang akan menghubungkan antar modul web-service.

Gambar 5. EventPattern

Sebagaimana yang telah penulis jelaskan pada bab source code bahwa dengan cara @EventPattern ini, pengirim pesan dan konsumen dapat mengkoordinasikan permintaan dan akan dilayani oleh penanganan yang telah disediakan. Ini disebut sistem Subscribe dan Unsubscribe.

E. Migrasi Data

Tahap migrasi data yang krusial dan menghabiskan waktu pengerjaan yang cukup panjang (pada proyek ini kurang lebih dua minggu) karena beberapa faktor salah satunya adalah data live production harus tetap berjalan dan penempatan data pada kolom tabel baru. Penulis membagi menjadi dua cara untuk melakukan migrasi data:

1. Membuat function baru untuk get data dan insert menuju table database baru secara bertahap.
2. Cara kedua yaitu migrasi secara manual dengan import export table sql. Dengan metode ini migrasi menjadi lambat karena harus disamakan terlebih dahulu setiap kali insert data, penamaan kolom dan data mana saja yang diperlukan.

F. Deployment

Tahapan terakhir dalam eksperimen ini ditandai proses deployment atau penyebaran web-service agar api dapat diakses oleh berbagai platform. Penulis menggunakan Bitbucket untuk third party pipeline source code nya. Dalam penulisan kode dilakukan branching atau percabangan tujuannya untuk memisahkan mana kode yang sedang live production dan mana kode yang sedang dalam proses pengujian atau pengembangan.[8] Sehingga bila terjadi kesalahan tidak langsung berdampak pada kode production. Ketika pull request terdapat tiga environment yaitu. Release/development, Release/staging, Release/production masing masing env memiliki ip database yang berbeda guna memisahkan data per release an web-service.

Setelah membuat pull request langkah berikutnya adalah approve and merge. Ini adalah akhir dari proses deployment, pada tahap ini juga kode penulis di review dan dicek ulang oleh sistem bitbucket apakah lolos atau ada error. Bila sukses kode sudah dapat berjalan dan api dapat menangani request client / user.

[8] Yujing Wang and Darrel Ma, "Developing a Process in Architecting Microservice Infrastructure with Docker, Kubernetes, and Istio," *ArXiv:1911.02275 [Cs]*, November 6, 2019.

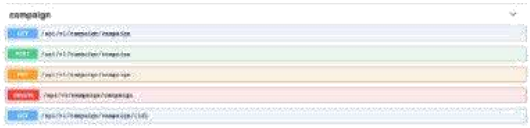
HASIL DAN PEMBAHASAN

A. Pengujian

Hasil Eksperimen sudah tampak jelas pada saat tahap pengujian. Setiap pengembang perangkat lunak pasti akan melewati fase pengujian atau testing sistem software yang telah dibuat guna mengetahui kelayakan dan performa yang dihasilkan. Penulis pada kesempatan kali ini melakukan proses pengujian dengan membagi menjadi dua tahap yakni pertama diuji secara manual dan kedua diuji secara produk penjelasanya sebagai berikut.

1. Testing secara manual, penulis setelah membuat API akan langsung menguji melalui swagger dengan menyimpan data bila API metode POST, mengubah data bila API menggunakan metode PUT, menarik data bila API menggunakan metode GET dan menghapus data bila API menggunakan metode DELETE.

Gambar 6. API microservice melalui Swagger



Gambar 7. API microservice Testing Online Tool.

Pengujian dilakukan juga melalui https://reqbin.com/ untuk mengecek kecepatan dan ukuran data balikan dari beberapa api, dibandingkan dengan api arsitektur monolitik, ini lebih cepat dengan rata-rata 800-900 ms per request.

2. Testing secara produk, penulis langsung berkoordinasi dengan team front-end apps (ios & android), front-end web, QA, product. Testing pada tahap ini dapat diartikan bahwa API sudah diintegrasikan dengan produk-produk yang ada pada campaign.com.[9] [10]

[9] Vinod Keshaorao Pachghare, "Microservices Architecture for Cloud Computing" 2, no. 1: 14.

[10] R Elsen dkk., "Microservice Architecture Design for Autograder Using Distributed Architecture," *IOP Conference Series: Materials Science and Engineering* 1098, no. 3 (March 1, 2021): 032083.

Gambar 8. Aplikasi Campaign.com

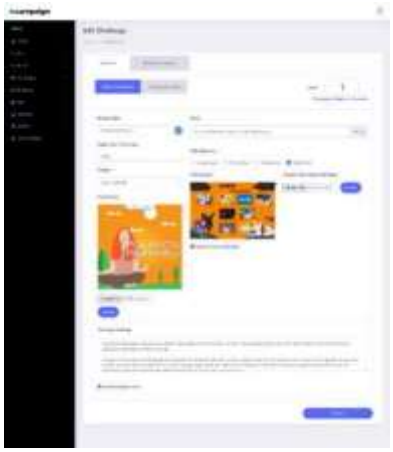
Ini adalah hasil user interface dari beberapa halaman (home, user, campaign, challenge) aplikasi campaign.com yang telah meggunakan API system microservice.

Gambar 9. Action Page

Gambar 10. Explore Page

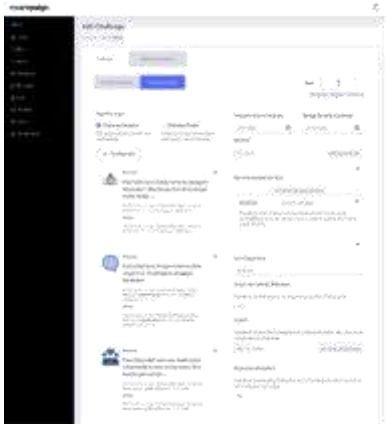
Gambar 11. Challenge Page

Dan ini adalah tampilan beberapa halaman website utama campaign.com tentunya dengan data yang disajikan telah menggunakan api sistem microservice.



Gambar 12. Edit Challenge Page

Gambar 13. Edit Detail Challenge Page



Gambar 14. Report Challenge Page

Ini adalah tampilan beberapa halaman admin dashboard campaign, website yang digunakan untuk mengatur dan mengolah data aplikasi campaign.com dan telah terintegrasi dengan api sistem microservice.

Namun dengan keterbatasan biaya dan skill penulis, sangat disayangkan belum dapat menerapkan automation testing atau bahkan unit testing sehingga arsitektur backend microservice yang telah selesai pada eksperimen ini belum teruji secara keseluruhan dengan kondisi yang mungkin belum sempat teruji pada dua poin yang penulis sampaikan diatas.

B. Bug Fixing

Setelah masa pengujian hasil eksperimen perlu adanya perbaikan atau yang sering disebut bug fixing. Dalam hal ini ketika pengujian berlangsung tidak jarang user mendapatkan berbagai macam error dari API sistem microservice, mulai dari response 500 Internal Server Error, 502 Bad Gateway dan 403 forbidden. Setiap pesan memiliki cara penanganannya tersendiri. untuk cara perbaikan, penulis menggunakan metode debugging agar dapat mengidentifikasi error lalu menemukan lokasi yang menyebabkan error dan kemudian melakukan analisa terhadap error yang telah ditemukan.

Gambar 15. Bug Fix Error API Status 403

Error response 403 biasanya terjadi karena tidak mendapatkan permissions terkait file atau data pada function sebuah API. Dan dapat dilihat bahwa letak kesalahan function tersebut telah terdeteksi oleh swagger, ini juga suatu keuntungan tersendiri untuk penulis yang telah menggunakan swagger dengan memudahkan proses debugging sehingga tidak perlu mencari file secara keseluruhan project.

Gambar 16. Bug Fix Error API Status 500

Lalu response error status 500 ini yang paling sering ditemukan pada saat pengujian dan API diimplementasi oleh front-end, sebabnya bermacam-macam mulai dari salah mengirim parameter, pencegahan atau pembatasan nilai null, dan yang cukup sering terjadi kesalahan penulisan query pada controller. Perbaikannya pun simple dan tidak terlalu rumit karena sama dengan error sebelumnya yaitu sudah dijelaskan dan ditunjukan oleh pesan swagger.

Gambar 17. Bug Fix Error API Status 200

Hasil apabila API sudah dilakukan bug fixing akan merespon status 200 seperti gambar diatas. Pada response body juga sudah mengembalikan data yang sesuai dengan parameter yang dikirim oleh user.

Gambar 18. Bug Fix Error API Status 502

Sempat juga mengalami error response status 502 yang menjadikan API terasa lambat dan bila mendapat request yang cukup tinggi sampai downtime dan solusi yang diterapkan untuk masalah ini adalah melakukan indexing pada tabel database agar error 502 dapat teratasi.

Meskipun beberapa contoh error yang penulis sampaikan diatas hal yang perlu diperhatikan adalah bahwa setiap kali mengalami kerusakan pada sebuah function API tidak mengakibatkan satu aplikasi tidak dapat berjalan dan berdampak ke seluruh fitur yang ada, hanya pada function module yang sedang mengalami perbaikan saja yang akan terdampak, disini sangat terasa sekali manfaat dari arsitektur microservice karena jika masih menggunakan monolitik, satu API yang mengalami masalah dapat berdampak satu aplikasi force close tidak dapat diakses secara keseluruhan.

KESIMPULAN DAN SARAN

Pada akhirnya hasil dari karya ilmiah ini telah menjawab dua pertanyaan dari masalah yang ada yaitu bagaimana proses perancangan arsitektur backend microservice dan bagaimana cara mengintegrasikan sistem monolitik menjadi microservice. Tentunya dengan kekurangan dan keterbatasan ilmu, penulis sudah memberikan penjelasan dan menyusun karya ilmiah ini dengan maksimal. Yang intinya dapat diambil pelajaran bahwa pemilihan sistem arsitektur backend dalam membangun sebuah software dan proses pengembanganya harus dianalisis sesuai dengan kebutuhan. Dua arsitektur yang penulis kaji yaitu monolitik dan microservice memiliki kelebihan dan kekurangannya masing-masing. Berikut perbandingan dalam bentuk tabel.

Monolith		**Microservices**	
Kelebihan	Kekurangan	Kelebihan	Kekurangan
Cepat untuk diterapkan (fast deployment)	Tidak dapat dipercaya, terdapat error kecil di satu tempat akan	Skalabilitas dapat menangani jutaan request per detik	Kompleksitas teknologi pembelajaran untuk

	merusak keseluruhan		mendukung layanan mikro
Mudah Untuk dipahami	Perbarui satu baris kode harus memperbaharui seluruh aplikasi	Sambungan antar komponen, satu system mati tidak mempengaruhi keseluruhan sistem	Memantau setiap komponen belajar membaca grafik untuk memeriksa situasi atau error
	Tekhnologi tidak dapat terdiri dari beberapa Bahasa pemrograman		Kompleksitas dalam pengujian

Berdasarkan eksperimen karya ilmiah ini, penulis mengetahui perbedaan antara arsitektur monolitik dan microservice dengan kekurangan dan kelebihannya. Yang mana sebenarnya dapat mengguanakan kedua arsitektur tersebut, namun lebih efisien jika menggunakan arsitektur microservice. Hal ini terbukti dari hasil eksperimen yang telah dilakukan. Dan dengan menggunakan arsitektur microservice secara devops tagihan untuk server menjadi terjangkau dikarenakan database sudah dipisah-pisah sesuai dengan modul dan maintenance untuk load balancing pun jauh lebih mudah karena apabila traffic sedang naik tidak harus semua dinaikan cukup modul database yang terkait saja.

DAFTAR PUSTAKA

Allahyari, Mehdi, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. “A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques,” 2017.

Darmayantie, Astie. “Desain Sistem Terfederasi Dengan Pendekatan Microservice Architecture Pada Kasus Studi Sistem Pelaporan Pajak.” *Jurnal Ilmiah Informatika Komputer* 25, no. 1 (2020): 50–63. https://doi.org/10.35760/ik.2020.v25i1.2523.

Elsen, R, M R Nashrulloh, R Cahyana, A Mulyani, and A Latifah. “Microservice Architecture Design for Autograder Using Distributed Architecture.” *IOP Conference Series: Materials Science and Engineering* 1098, no. 3 (March 1, 2021): 032083. https://doi.org/10.1088/1757-899X/1098/3/032083.

Gos, Konrad, and Wojciech Zabierowski. “The Comparison of Microservice and Monolithic Architecture.” In *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, 150–53. Lviv, Ukraine: IEEE, 2020. https://doi.org/10.1109/MEMSTECH49584.2020.9109514.

Khalida, Rakhmi, Adi Muhajirin, and Siti Setiawati. “Teknis Kerja Docker Container untuk Optimalisasi Penyebaran Aplikasi.” *PIKSEL : Penelitian Ilmu Komputer Sistem Embedded and Logic* 7, no. 2 (September 23, 2019): 167–76. https://doi.org/10.33558/piksel.v7i2.1819.

Pachghare, Vinod Keshaorao. “Microservices Architecture for Cloud Computing” 2, no. 1 (n.d.): 14.

Ponce, Francisco, Gaston Marquez, and Hernan Astudillo. “Migrating from Monolithic Architecture to Microservices: A Rapid Review.” In *2019 38th International Conference*

of the Chilean Computer Science Society (SCCC), 1–7. Concepcion, Chile: IEEE, 2019. https://doi.org/10.1109/SCCC49216.2019.8966423.

Sakti, Calvin Seviro Bima, and Indra Hermawan. "Implementasi Arsitektur Microservice Pada Back End Sistem Informasi Atlantas Berbasis Website." *Jurnal Teknologi Terpadu* 6, no. 2 (2020).

Wang, Yujing, and Darrel Ma. "Developing a Process in Architecting Microservice Infrastructure with Docker, Kubernetes, and Istio." *ArXiv:1911.02275 [Cs]*, November 6, 2019. http://arxiv.org/abs/1911.02275.