Jurnal Algoritma
E-ISSN: 2302-7339 | P-ISSN: 1412-3622
INSTITUT TEKNOLOGI GARUT

Perancangan Arsitektur *Microservice* untuk Portal Berita Daring

Rickard Elsen

Jurnal Algoritma
Institut Teknologi Garut
Jl. Mayor Syamsu No. 1 Jayaraga Garut 44151 Indonesia
Email : jurnal@itg.ac.id

rickardelsen@itg.ac.id

Abstrak **–** Saat ini, portal berita daring sudah marak dan sudah digunakan oleh masyarakat sebagai media untuk mencari berita dan informasi. Dengan banyaknya akses dari masyarakat sebagai pengguna, maka infrastruktur yang diterapkan oleh portal berita daring pun harus mampu melayani kebutuhan masyarakat akan informasi dengan lancar dan minim kendala. Ada banyak arsitektur yang bisa diterapkan terhadap infrastruktur portal berita daring. Salah satunya *microservice*, arsitektur yang menjadikan aplikasi menjadi beberapa komponen yang berdiri sendiri dan memiliki tanggung jawab kerja yang unik. Dengan *microservice*, infrastruktur yang diterapkan bisa dibagi menjadi beberapa komponen yang saling terhubung. *Microservice* mengedepankan fleksibilitas sehingga komponen-komponen yang ada bisa digunakan ulang oleh komponen atau layanan lain. Dalam jurnal ini, penulis akan merancang arsitektur *microservice* untuk portal berita daring berdasarkan hasil observasi, studi literatur, dan studi empiris terhadap beberapa portal berita daring sehingga didapat komponen apa saja yang terdapat pada portal berita daring.

Kata Kunci **–** Arsitektur Perangkat Lunak; *Microservice*; Portal Berita Daring.

I. PENDAHULUAN

Saat ini, internet sudah banyak digunakan untuk berbagai kebutuhan. Contohnya untuk keperluan belanja, belajar, pemasaran, sampai mencari informasi melalui sosial media atau portal berita. Di Indonesia sendiri terdapat banyak portal berita daring. Menurut data dari Kemenkominfo, di tahun 2018 ada 43.000 portal berita daring di Indonesia. Dengan banyaknya portal berita daring mununjukkan bahwa kebutuhan masyarakat akan informasi terus meningkat. Apalagi dengan semakin mudahnya mengakses portal berita tersebut melalui internet. Cukup melalui ponsel pintar yang terhubung ke internet maka masyarakat bisa mendapatkan informasi yang begitu banyak dari berbagai sumber.

Dengan maraknya portal berita daring dan semakin banyaknya masyarakat yang mengakses portal berita tersebut, maka diperlukan infrastruktur yang memadai agar akses masyarakat terhadap portal berita bisa berjalan dengan lancar dan minim gangguan. Sebuah portal berita yang memiliki pengunjung yang banyak dengan traffic yang padat tentunya memerlukan infrastruktur yang memadai agar pengunjung portal berita bisa mendapatkan informasi dengan lancar.

Berbicara mengenai infrastruktur, tentunya tidak bisa lepas dari arsitektur perangkat lunak. Arsitektur perangkat lunak adalah struktur dasar dari sistem perangkat lunak dan disiplin pembuatan struktur dan sistem perangkat lunak. Setiap struktur terdiri dari elemen perangkat lunak, hubungan antar elemen, dan properti dari elemen dan relasi. Arsitektur yang diterapkan pada sebuah sistem perangkat lunak tergantung dari gaya arsitektur yang dipilih. Salah satu gaya arsitektur yang banyak diterapkan adalah arsitektur *microservice*.

Arsitektur *microservice* adalah gaya arsitektur yang menyusun aplikasi sebagai kumpulan layanan yang berfokus pada kemudahan perawatan dan pengujian, mengurangi ketergantungan antar komponen, penerapan komponen secara independent, dikelola berdasarkan kemampuan bisnis, dan dimiliki oleh tim yang kecil. Arsitektur *microservice* memungkinkan pengiriman aplikasi yang besar dan kompleks secara cepat, teratur, dan andal. Arsitektur ini juga memungkinkan organisasi untuk mengembangkan teknologi yang diterapkan.

Saat ini, arsitektur microservice adalah arsitektur yang paling populer. Banyak perusahaan besar seperti Amazon, Netflix, Spotify, eBay, dan Uber menerapkan arsitektur microservice bahkan ikut berkontribusi dalam pengembangan arsitektur microservice. Penelitian terkait arsitektur microservice pun sudah banyak dilakukan sehingga menghasilkan banyak rujukan, model, spesialisasi, dan *best practice* terkait arsitektur microservice. Arsitektur ini juga sudah diterapkan pada banyak *framework* pengembangan perangkat lunak sehingga penerapan microservice pada aplikasi bisa dilakukan dengan mudah.

II. METODOLOGI PENELITIAN

Dalam pembuatan jurnal ini, penulis menggunakan 2 metoda yang dilakukan secara bertahap yaitu studi literatur dan studi empiris. Tahapan yang dilakukan penulis dijelaskan sebagai berikut:

A. Studi Literatur

Pada tahapan ini, penulis melakukan penelusuran dan mempelajari literatur yang berkaitan dengan *microservice* dan portal berita daring. Sumber informasi yang digunakan berasal dari jurnal dan artikel yang berkaitan dengan *microservice*. Dari tahapan ini, didapat informasi mengenai karakteristik *microservice* berikut:

1) Komponen sebagai layanan. Saat ini, banyak pengembang berusaha untuk membangun aplikasi yang dapat digunakan ulang pada aplikasi lain yang akan dikembangkan kemudian. Banyak sekali praktek pengembangan yang melakukan pengembangan ulang terhadap suatu fitur yang sama. Contohnya fitur untuk login. Setiap aplikasi yang membutuhkan fitur login harus mengembangkan ulang fitur ini dalam setiap pengembangannya. Akibatnya, biaya dan waktu pengembangan bisa menjadi lebih tinggi. Pada *microservice*, pengembang bisa mengembangkan fitur login dan digunakan berulang-ulang untuk setiap aplikasi yang membutuhkan fitur ini. Hal ini karena *microservice* memperlakukan fitur login sebagai sebuah layanan mandiri dimana akses terhadap layanan ini tidak dibatasi hanya untuk satu aplikasi saja.
2) Mengacu pada kebutuhan bisnis. Komponen yang digunakan dalam aplikasi yang menerapkan *microservice* dikelola berdasarkan kebutuhan bisnis. Setiap komponen yang digunakan bisa sangat fleksibel dalam hal pengadaan, pembangunan, pengoperasian, hingga pemeliharaan. Contohnya ketika suatu bisnis sangan bergantung pada kecepatan akses pengguna. Maka pada *microservice*, komponen yang berhubungan dengan kecepatan akses seperti *backend* engine dan basis data bisa diperkuat atau diperbanyak (disinkronisasi menggunakan *load balancer*). Komponen yang memiliki impact bisnis yang besar bisa disesuaikan sesuai dengan kebutuhan organisasi.
3) Berfokus pada produk, bukan proyek. Dalam aplikasi yang menerapkan *microservice*, fokus utama pengembangan bukanlah keseluruhan aplikasi yang dinotasikan dalam proyek perangkat lunak. Melainkan berfokus pada masing-masing komponen yang dibangun sehingga membentuk satu kesatuan utuh. Aplikasi tidak perlu berada pada kondisi selesai untuk bisa dijalankan atau diuji langsung oleh pengguna. Salah satu ciri utama *microservice* adalah setiap komponen bisa berdiri sendiri sehingga masing-masing komponen yang diterapkan bisa digunakan dan diuji secara terpisah meskipun proyeknya belum selesai. Karakteristik ini memungkinkan setiap pengembang dari setiap komponen bisa melakukan pengujian terus menerus selama proyek berjalan agar komunikasi antar komponen bisa berjalan lebih baik lagi selama masa pengembangan.
4) Mengutamakan endpoint. *Microservice* menggunakan protokol yang sangat umum digunakan yaitu *Hypertext Transfer Protocol* (HTTP). Artinya, *microservice* tidak memiliki protokol khusus untuk jalur komunikasinya. Namun dengan penggunaan HTTP sebagai jalur komunikasi utama, maka aplikasi yang menerapkan *microservice* harus memiliki endpoint yang handal. Kategori handal dalam endpoint bisa dicapai dengan menerapkan routing yang baik, menjalankan logika secara mandiri, melakukan filter yang

diperlukan melalui *middleware*, menjaga akuntabilitas data, dan menjaga performansi aplikasi.

5) Tata Kelola desentralisasi. Dalam pengelolaan tersentralisasi, satu server akan melayani semua proses yang dibutuhkan agar aplikasi bisa diakses dan digunakan oleh pengguna. Dalam *microservice*, setiap komponen memiliki tanggung jawab masing-masing sehingga proses yang diperlukan agar aplikasi bisa digunakan akan dibagi ke setiap komponen. Instrumentasi antar komponen sangat diperlukan dan menentukan kinerja aplikasi secara keseluruhan.
6) Pengelolaan data desentralisasi. Karakteristik ini berhubungan dengan karakteristik sebelumnya dimana data pada *microservice* diperlakukan sebagai sebuah komponen. Dalam pengelolaan data, diperlukan pengelolaan terhadap sistem pengelolaan basis data. Ada banyak jenis sistem pengelolaan basis data sehingga diperlukan sinkronisasi yang baik mengingat *microservice* sangat bergantung pada instrumentasi antar komponen.
7) Otomatisasi infrastruktur. Dalam pengembangan aplikasi yang menerapkan *microservice*, penerapan infrastruktur aplikasi harus dilakukan secara otomatis. Hal ini karena dalam aplikasi berbasis *microservice*, setiap komponen harus berdiri sendiri dan jumlah komponen aktif bergantung pada kebutuhan bisnis yang disinkronisasi melalui instrumentasi. Jika penerapan infrastruktur tidak dilakukan secara otomatis, maka pekerjaan menjalankan aplikasi keseluruhan akan memakan banyak waktu. Saat ini sudah ada beberapa proyek *opensource* yang membantu dalam otomatisasi infrastruktur contohnya Terraform dan Ansible.
8) Didesain untuk menanggulangi kegagalan. Dalam *microservice*, setiap komponen melakukan pekerjaannya masing-masing. Dalam melakukan tugasnya, komponen bisa saja melakukan kegagalan. Sebagai imbas dari penerapan komponen yang banyak, aplikasi yang menerapkan *microservice* harus siap ketika ada komponen lain yang gagal baik dalam hal pemrosesan informasi, logika, hingga kegagalan dalam komunikasi. Setiap komponen harus didesain agar bisa menanggulangi setiap kemungkinan kegagalan yang terjadi dalam instrumentasi kerja antar komponen.
9) Desain yang evolusioner. Dalam *microservice*, dekomposisi proses antar komponen perlu dipikirkan dan diterapkan secara matang. Kematangan dekomposisi proses ini akan membuat komponen memiliki pekerjaan yang spesifik dan mandiri sehingga di masa depan penguatan komponen akan berfokus pada masing-masing tanggung jawab komponen saja. Hal ini akan mempermudah evolusi perangkat lunak di masa depan, baik dalam hal pemeliharaan atau pembaharuan.

B. Studi Empiris

Pada tahapan ini, penulis melakukan perbandingan beberapa portal berita daring mainstream di Indonesia untuk menemukan karakteristik dari portal berita online. Selain perbandingan, penulis juga mencari sumber pendukung dari jurnal dan artikel daring terkait portal berita online. Tahapan ini bertujuan untuk melihat fitur apa saja yang perlu ada pada portal berita online dari sisi pengguna dan berkaitan dengan microservice. Dari tahapan ini, didapat fitur utama terkait microservice pada portal berita online:

1) Responsif. Portal berita online harus memberikan respon yang cepat setiap kali pengguna mengakses berita dan berganti ke berita lain. Selain itu, akses terhadap fitur tambahan seperti kategori berita, halaman pengelolaan akun, dan berita yang disarankan harus bisa diakses pengguna secara realtime.
2) Selalu tersedia. Portal berita online harus bisa diakses kapanpun dan dimanapun oleh pengguna selama pengguna terhubung ke internet. Selain ketersediaan layanan, pengguna pun harus bisa mengakses semua berita yang terdapat pada portal berita online.
3) Handal. Portal berita online harus mampu melayani pengguna dalam jumlah banyak yang mengakses portal berita online di waktu yang bersamaan. Penurunan performa akibat akses yang banyak dapat menggangu pengalaman pengguna ketika membaca berita pada portal berita online, terutama ketika pengguna membaca lebih dari satu berita.

III. HASIL DAN PEMBAHASAN

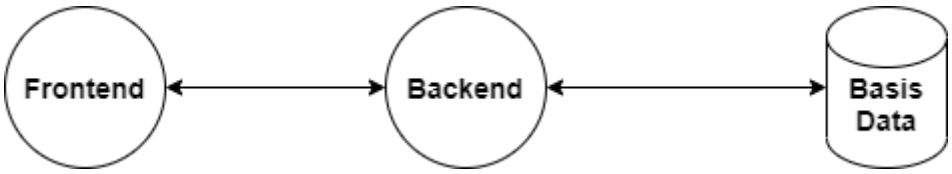
Dalam melakukan perubahan arsitektur dari *monolithic* ke *microservice*, diperlukan analisis terhadap komponen dalam aplikasi. Setiap komponen yang didapat dari hasil analisis akan dibagi sehingga setiap komponen dapat berdiri sendiri dan mampu mengerjakan tugasnya masing-masing. Dalam penerapan *microservice*, beberapa hal yang harus diperhatikan adalah:

1) Aplikasi harus dibagi menjadi komponen-komponen dengan tanggung jawab yang unik.
2) Setiap komponen harus bisa berdiri sendiri.
3) Komunikasi antar komponen harus diinstrumentasi dengan baik
4) Setiap komponen harus siap menerima permintaan dari komponen lain, baik permintaan yang sesuai dengan format yang telah ditentukan atau permintaan yang gagal.
5) Komponen harus bisa diduplikasi dan diinstrumentasi melalui *load balancer*.

Dari hal-hal di atas, maka aplikasi portal berita daring harus memiliki komponen-komponen yang masing-masing bisa berdiri sendiri sesuai dengan tanggung jawabnya. Dalam aplikasi portal berita daring, komponen utama yang diperlukan adalah sebagai berikut:

1) *Frontend*, komponen yang memiliki tanggung jawab melakukan render antarmuka web untuk portal yang akan dilihat oleh pengguna.
2) *Backend*, komponen yang memiliki tanggung jawab dalam pemrosesan data dan logika menjadi informasi yang akan disajikan kepada pengguna.
3) Basis data, komponen yang bertanggung jawab melakukan pengelolaan data.

Berikut desain hubungan antar komponen utama portal berita daring:

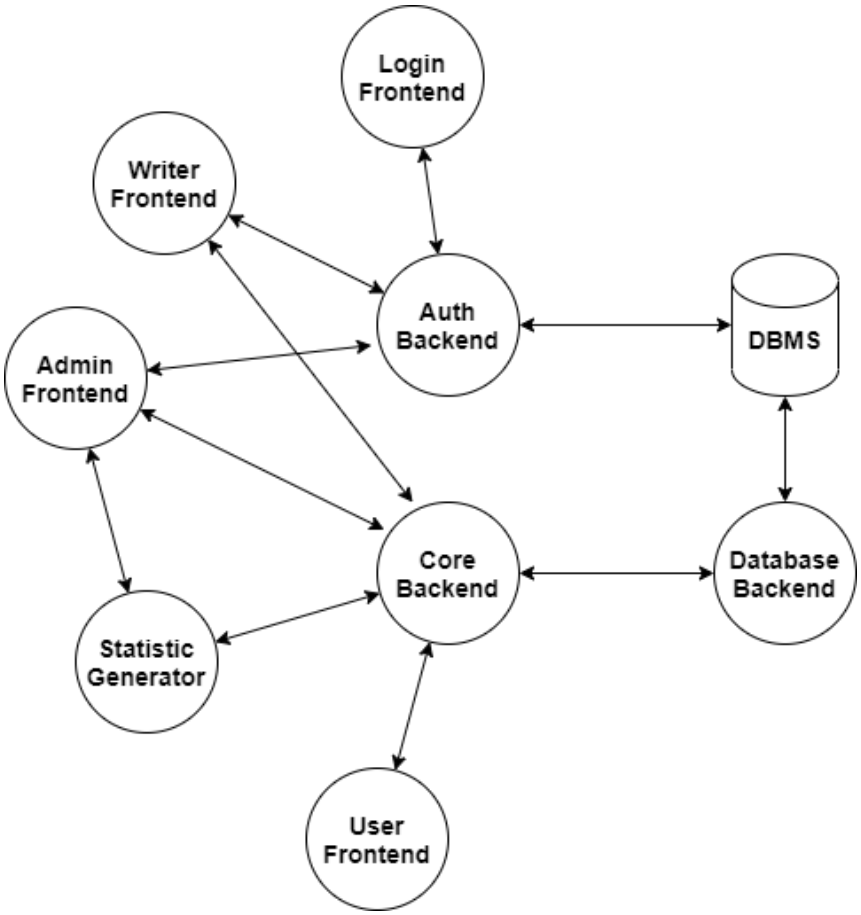


Gambar 1: Hubungan antar komponen utama portal berita daring

Berdasarkan pada komponen utama portal berita daring, didapat komponen-komponen hasil dekomposisi agar pekerjaan dan tanggung jawab masing-masing komponen menjadi unik. Pembagian ini didasari oleh karakteristik dari microservice yang membagi aplikasi menjadi komponen-komponen mandiri sesuai dengan tugasnya yang bersifat unik. Berikut daftar komponen hasil dekomposisi dari komponen utama portal berita daring:

1) *User frontend*, bertugas menampilkan hasil render halaman web kepada pengguna.
2) *Admin frontend*, bertugas menampilkan hasil render halaman web kepada administrator portal berita.
3) *Writer frontend*, bertugas menampilkan hasil render halaman web kepada penulis berita.
4) *Statistic generator*, bertugas untuk menyajikan data-data statistik yang diperlukan.
5) *Login frontend*, bertugas menampilkan hasil *render* halaman web untuk antarmuka autentikasi melalui *login*.
6) *Auth backend*, bertugas untuk melakukan autentikasi pengguna berdasarkan tanggung jawab dari pengguna.
7) *Core backend*, bertugas untuk melakukan pemrosesan data dan logika.
8) *Database backend*, bertugas untuk mengelola *query* terhadap basis data.
9) *Database management system* (DBMS), bertugas menyimpan dan mengelola data.

Dari hasil dekomposisi tersebut, berikut desain hubungan antar komponen portal berita daring:



Gambar 2: Hubungan antar komponen hasil dekomposisi

Desain di atas baru mengambarkan hubungan antar komponen dengan asumsi bahwa kebutuhan bisnis saat ini hanya memerlukan satu *instance* untuk setiap komponen. Dalam penerapannya, satu komponen mungkin akan diaplikasikan menjadi beberapa *instance*. Contoh untuk portal berita yang memiliki kebutuhan akses tinggi, maka komponen *core backend* dan *database backend* bisa dibuat menjadi beberapa *instance*. Tidak hanya itu, DBMS pun dibuat menjadi beberapa *instance* misal 1 DBMS untuk penyimpanan data, 1 DBMS khusus untuk pencarian data, dan 1 DBMS berbasis *memory* untuk akses data yang lebih cepat.

Saat ini, sudah banyak aplikasi *opensource* yang sudah mendukung pengembang dalam pengembangan aplikasi menggunakan arsitektur *microservice*. Dalam kasus portal berita daring, contoh aplikasi *opensource* yang bisa dimanfaatkan adalah sebagai berikut:

1) ReactJS, aplikasi *opensource* buatan Facebook ini sangat cocok digunakan sebagai *framework* untuk *frontend*. Kelebihan *framework* ini adalah sangat fleksibel dalam pengembangan karena bisa digunakan untuk aplikasi web berbasis *Single Page Application* (SPA), *Server Side Rendering* (SSR) hingga aplikasi mobile untuk android dan iOS (menggunakan react native).
2) NodeJS, aplikasi *opensource* ini sangat cocok digunakan sebagai *framework* untuk *backend*. Kelebihan *framework* ini adalah memiliki banyak sekali module untuk membantu pengembang dalam membangun aplikasi. Salah satu *framework* yang banyak digunakan oleh pengembang NodeJS adalah ExpressJS.
3) Hydra, aplikasi *opensource* ini bisa digunakan sebagai OAuth server sehingga pengelolaan autentikasi menjadi lebih mudah.
4) Keto, aplikasi *opensource* ini bisa digunakan untuk model autentikasi berbasis *Role Based Access Control* (RBAC) dimana setiap pengguna bisa dikelola hak aksesnya terhadap sistem.

5) MariaDB, aplikasi *opensource* ini bisa digunakan sebagai DBMS untuk penyimpanan data. Kelebihan MariaDB adalah ringan sehingga tidak perlu infrastruktur yang tinggi untuk bisa menjalankan aplikasi ini.
6) PostgreSQL, sama dengan MariaDB aplikasi ini pun bisa digunakan sebagai DBMS untuk penyimpanan data. Kelebihan yang dimiliki PostgreSQL adalah aplikasi ini memang dirancang untuk *enterprise* sehingga bisa menampung dan mengelola lebih banyak data.
7) MongoDB, aplikasi ini bisa digunakan untuk penyimpanan data berbasis NoSQL. Kelebihan dari aplikasi ini adalah pengembang bisa dengan bebas menentukan struktur data yang akan disimpan sehingga setiap data bisa memiliki keunikan antar satu dan lainnya.
8) Elasticsearch, aplikasi ini bisa menyimpan data berbasis indeks. Kelebihan aplikasi ini adalah mampu melakukan banyak jenis *query* terhadap banyak jenis data sehingga sangat cocok untuk dijadikan basis data untuk data yang memiliki aktifitas *query* yang tinggi.

IV. KESIMPULAN

Penerapan arsitektur *microservice* pada portal berita daring dilakukan dengan melakukan identifikasi terhadap komponen utama yang terdapat pada portal berita daring. Komponen utama tersebut lalu didekomposisi lagi menjadi komponen yang lebih spesifik agar kerja dan tanggung jawab masing-masing komponen menjadi unik dan mandiri. Meskipun sudah didekomposisi, namun komponen turunan ini masih bisa diaplikasikan menjadi beberapa *instance* dengan tujuan untuk menyesuaikan dengan kebutuhan bisnis. Dalam jurnal ini, yang dilakukan hanyalah perancangan namun belum sampai pada pengaplikasian rancangan pada perangkat lunak. Penulis berharap di masa depan rancangan ini bias diterapkan pada perangkat lunak untuk mengetahui pengaruh rancangan arsitektur terhadap portal berita daring.

DAFTAR PUSTAKA

[1] H. W. Hu, Y. K. Tsay, and C. Y. Peng, “Influence of online news features on user behavior for new media,” 2019. doi: 10.1109/BigData.2018.8621963.
[2] W. M. S. Yafooz, S. Z. Z. Abidin, and N. Omar, “Challenges and issues on online news management,” 2011. doi: 10.1109/ICCSCE.2011.6190574.
[3] F. Wan, X. Wu, and Q. Zhang, “Chain-Oriented Load Balancing in Microservice System,” 2020. doi: 10.1109/WCCCT49810.2020.9169996.
[4] R. Petrasch, “Model-based engineering for microservice architectures using Enterprise Integration Patterns for inter-service communication,” 2017. doi: 10.1109/JCSSE.2017.8025912.
[5] “Who is using microservices?” https://microservices.io/articles/whoisusingmicroservices.html (accessed May 16, 2021).
[6] S. Hassan, N. Ali, and R. Bahsoon, “Microservice Ambients: An Architectural Meta-Modelling Approach for Microservice Granularity,” 2017. doi: 10.1109/ICSA.2017.32.
[7] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, *Microservice architecture : Aligning principles, practices, and culture*. 2015.
[8] M. B. Salwen, B. Garrison, and P. D. Driscoll, *Online news and the public*, vol. 9781410611611. 2004. doi: 10.4324/9781410611611.
[9] J. Lewis, “Microservices - A definition of this new architectural term,” *martinfowler.com*, 2014. https://martinfowler.com/articles/microservices.html (accessed May 14, 2021).
[10] C. Richardson, “Microservice Architecture pattern,” *Microservices.io*, 2018. https://microservices.io/index.html (accessed May 14, 2021).
[11] Kementerian Komunikasi dan Informatika, “Menkominfo: Baru 100 Portal Berita Online Terverifikasi,” *kominfo.go.id*, 2018.
[12] F. Bachmann *et al.*, “Documenting Software Architecture: Documenting Interfaces,” *Computer Science Department*, no. June, 2002.