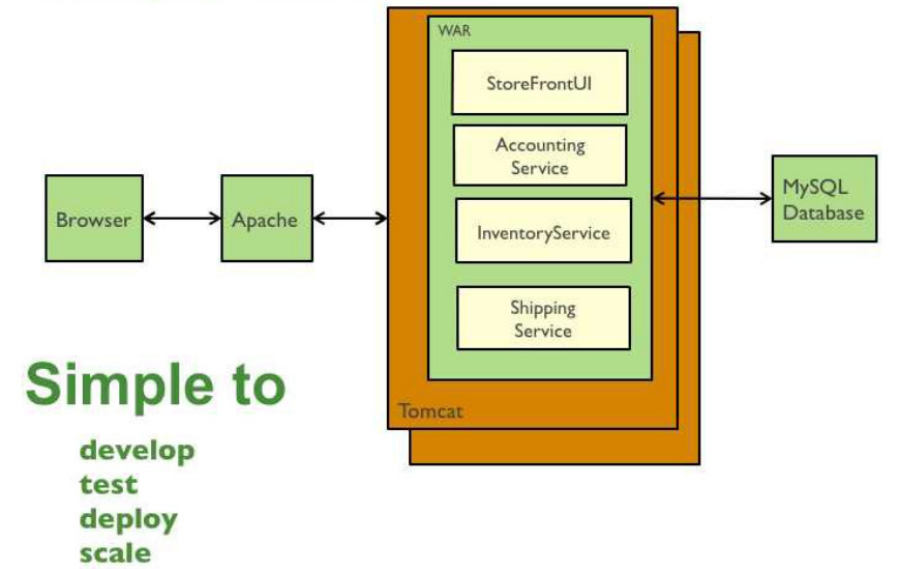
TEKNOLOGI INFORMASI

Arsitektur Monolitik Vs *Microservice* untuk Pengembangan Aplikasi pada Era Digital

Oleh
Elyyani | Pussainsa OR-PA BRIN

Perkembangan aplikasi pada era digital saat ini mendorong para pengembang aplikasi untuk menyediakan alternatif arsitektur lain agar pengelolaan sistem pada masa depan menjadi lebih fleksibel terhadap berbagai perubahan (*requirement change*). Ketika aplikasi yang ada makin besar dan jumlah pengakses makin banyak, tentu akan memengaruhi proses kinerja aplikasi yang tersedia sehingga performanya akan menurun. Saat ini, pendekatan monolitik telah bergeser menjadi pendekatan terdistribusi, yaitu aplikasi dibagi menjadi bagian-bagian kecil yang berfungsi spesifik (*high cohesion*) dan tidak bergantung kepada komponen program lainnya (*loose coupling*), dengan dikembangkannya layanan melalui antarmuka API (*Application Programming Interface*) (Newman, 2015).

Fitur-fitur pada arsitektur monolitik dibangun dalam aplikasi tunggal dan basis data tunggal sebagai satu sistem besar dan berdasarkan satu baris kode yang memiliki arsitektur terpusat dengan teknologi yang seragam. Dalam pembuatan aplikasi monolitik, semua komponen menjadi satu kesatuan antara *front end* dan *back end* serta diletakkan dalam satu server yang besar. Permasalahan pada arsitektur ini adalah jika salah satu komponen melakukan pembaruan pada *back end/front end* maka seluruh server akan terdampak.

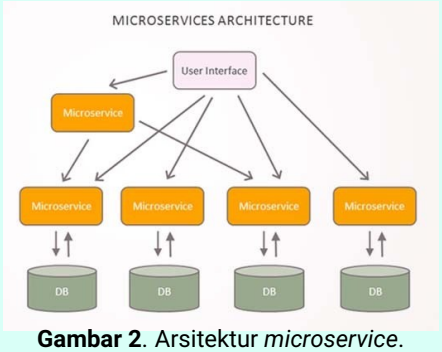


Gambar 1. Arsitektur monolitik.
(Sumber: `https://microservices.io/i/DecomposingApplications.011.jpg`)

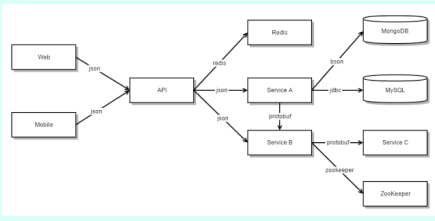
Gambar 1 menunjukkan arsitektur monolitik aplikasi *enterprise* yang dibangun atas tiga bagian, yaitu basis data, sisi-klien, dan sisi-server. Sisi-server akan menangani permintaan HTTP, menjalankan beberapa logika sesuai dengan domain, kemudian mengambil dan memperbarui data dari basis data, dan mengirimkan data tersebut ke sisi-klien.

Pola arsitektur *microservice* ditunjukkan oleh Gambar 2, aplikasi dibuat menjadi layanan-layanan kecil yang saling berkaitan agar setiap layanan dapat bekerja optimal sesuai dengan kebutuhannya. Arsitektur *microservice* akan membagi layanan menjadi bagian-bagian yang lebih kecil, misalnya ada server khusus untuk memproses layanan antrean, layanan *hosting* khusus untuk antarmuka pengguna, server khusus untuk manajemen basis data, dan sebagainya (`datacommcloud.co.id`).

Arsitektur *microservice* memungkinkan tiap layanan pada aplikasi mengalami pengembangan tersendiri yang secara signifikan memengaruhi hubungan antara aplikasi dan basis data dan masing-masing layanan memiliki skema basis data sendiri. Dalam hal ini



Gambar 2. Arsitektur *microservice*.
(Sumber: `datacommcloud.co.id`)



Gambar 3. Implementasi arsitektur *microservice* berbasis web.
(Sumber: `medium.com`)

Tabel 1. Kelebihan dan kekurangan arsitektur monolitik dan *microservice*. (Sumber: https://medium.com dan https://id.cloud-ace.com)

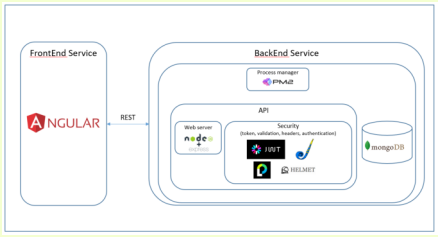
Monolitik		*Microservice*	
Kelebihan	**Kekurangan**	**Kelebihan**	**Kekurangan**
• Mudah dibangun • Mudah diuji coba • Mudah di-*deploy* ke server	• Performa menurun ketika aplikasi makin besar • Perubahan teknologi pada aplikasi akan mengubah aplikasi secara keseluruhan • Kesalahan pada salah satu fungsi akan memengaruhi keseluruhan aplikasi • Sulit untuk dirawat ketika jumlah data bertambah besar • Bergantung kepada satu teknologi	• Aplikasi *scalable*, aman, dan dapat diandalkan • Setiap layanan berdiri sendiri • Mudah dalam fase perbaikan • Tidak ada hambatan dalam menggunakan teknologi baru • Setiap tim pengembang dapat mengembangkan setiap layanan tanpa mengganggu layanan yang lain	• Ketika satu entitas pada basis data berubah maka setiap entitas yang sama di setiap basis data layanan harus diubah • *Deployment* yang kompleks, perlu konfigurasi untuk menjalankan setiap layanan karena memiliki *runtime* yang berbeda • Perlu automasi yang tinggi dalam melakukan *deployment* • Ada kemungkinan komunikasi antarlayanan mengalami kegagalan sehingga harus mempersiapkan cara penanganan yang tepat

layanan dapat menggunakan jenis basis data dan bahasa pemrograman yang paling sesuai dengan keperluannya. Hal tersebut dikarenakan pola *microservice* memiliki metode dengan membagi layanan ke bagian yang lebih kecil, tetapi tetap berkaitan dan setiap layanan yang dibuat dapat menggunakan teknologi yang berbeda pula. Walaupun aplikasi menjadi lebih padat dan kompleks, pengaksesannya tetap ringan sehingga setiap layanan dapat bekerja lebih optimal.

Gambar 3 menunjukkan bahwa penerapan teknologi *microservice* ini memungkinkan setiap layanan menggunakan teknologi yang berbeda sehingga dapat diintegrasikan ke aplikasi web, Android, ataupun yang lainnya, dapat menggunakan API *Gateway* yang bertugas sebagai *load balancing, caching, access control*, API *metering*, dan *monitoring*. Manfaat lain yang diperoleh dari pendekatan *microservice* adalah dapat meningkatkan fleksibilitas dan independensi dalam memperbarui suatu layanan tanpa memengaruhi layanan lainnya. Tabel 1 menunjukkan kelebihan dan kekurangan arsitektur monolitik dan *microservice*.

Setelah mengetahui kelebihan dan kekurangan kedua arsitektur di atas, pemilihan arsitektur untuk pengembangan aplikasi dapat disesuaikan dengan kebutuhan masing-masing, seperti seberapa besar aplikasi yang akan dibangun. Arsitektur *microservice* dapat digunakan untuk skala proyek yang sangat besar (*enterprise*) dan memerlukan dana yang cukup besar karena setiap layanannya membutuhkan satu server dan satu basis data serta melibatkan berbagai macam *stack programming* (*polyglot programming*). Sementara itu, penggunaan arsitektur monolitik dapat digunakan untuk skala proyek kecil dengan dana yang terbatas dan tidak melibatkan banyak pengembang sehingga satu server dan satu basis data dapat digunakan bersama. Dalam hal penerapannya, arsitektur *microservice* sudah digunakan pada aplikasi Gojek, Grab, hingga Netflix.

Saat ini, sistem informasi DSS (*Decision Support System*) sedang dikembangkan di Pusat Sains Antariksa. Untuk mengantisipasi perubahan teknologi pada masa depan serta untuk menjaga performa aplikasinya, arsitektur *microservice* ini sangat tepat digunakan untuk mengembangkan aplikasi pada era digital. Gambar 4 menunjukkan rancangan arsitektur basis data menggunakan pola arsitektur *microservice* untuk bagian layanan *front end* dan layanan



Gambar 4. Rancangan basis data menggunakan arsitektur *microservice*

back end pada sistem informasi DSS Pusat Sains Antariksa. Antara layanan *front end* dan *back end* dihubungkan oleh REST sebagai arsitektur yang digunakan dalam API, dengan data yang diberikan oleh server REST berupa format JSON, dan pengolahan basis datanya menggunakan MongoDB.

Pustaka

[1] Newman, S. (2015). Building Microservices. O'Reilly Media, Inc.

[2] microservices.io/i/Decomposing-Applications.011.jpg

[3] medium.com/kugen/software-architecture-monolithic-and-microservice-a9ed178ed954

[4] docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

[5] datacommcloud.co.id/monolithic-atau-microservices/

■

TEKNOLOGI INFORMASI

Pengantar Kerangka Pemrograman Flutter untuk Pengembangan Aplikasi *Smartphone*

Oleh
A.Z. Utama | Pussainsa OR-PA BRIN

Dalam beberapa dekade terakhir banyak negara yang memiliki lembaga keantariksaan yang bertanggung jawab untuk melakukan kegiatan observasi, monitoring, analisis, pemodelan, dan prediksi cuaca antariksa serta memberikan mitigasi dini kepada masyarakat ataupun pemangku kepentingan yang terdampak langsung. Pusat Riset Antariksa LAPAN BRIN merupakan lembaga penelitian pemerintah yang bertugas memberikan pelayanan informasi cuaca antariksa di Indonesia. SWIFtS (*Space Weather Information and Forecast Services*) merupakan layanan informasi dan prediksi cuaca antariksa yang dimiliki oleh Pussainsa. Pussainsa memberikan informasi mengenai hasil prediksi kondisi cuaca antariksa dan aplikasinya pada web swifts.sains.lapan.go.id.

Efek dari cuaca antariksa dapat berakibat fatal pada kehidupan modern. Pada tahun 2012 terjadi peristiwa CME yang besar dan hampir mengenai planet Bumi. Para peneliti mengestimasi efek dari peristiwa CME tersebut apabila mengenai planet Bumi akan berakibat pada kegagalan sistem listrik dan akan berdampak pula pada sistem pengairan, akses internet, dan sistem perbankan. Radiasi yang dihasilkan oleh aktivitas Matahari aktif membutuhkan jam bahkan hari untuk sampai ke Bumi. Untuk meminimalisir dampak dari cuaca antariksa tersebut diperlukan sistem peringatan dini yang handal agar pemerintah dan perusahaan yang terkena dampak dapat segera mengambil keputusan untuk melakukan mitigasi secara tepat dan akurat agar infrastuktur dan teknologinya dapat terlindungi.

Pada masa kini kehidupan manusia tidak dapat dipisahkan dengan perangkat seluler pintar (*smartphone*). Pemanfaatan teknologi *smartphone* dapat membantu peneliti melakukan kegiatan desiminasi untuk masyarakat mengenai hasil penelitian ataupun informasi penting secara cepat. Aplikasi *smartphone* merupakan suatu wadah yang efektif untuk menyebarkan informasi tersebut, karena hampir sebagian masyarakat dunia sudah menggunakan perangkat tersebut. Aplikasi *mobile* dapat membantu dan mempermudah para *stakeholder* untuk mendapatkan informasi mengenai cuaca antariksa dan aplikasinya. Flutter dikeluarkan oleh perusahaan raksasa Google untuk membangun aplikasi dalam satu *codebase* yang dapat dikompilasi ke dalam perangkat *mobile*, web, dan *desktop*. Flutter merupakan kerangka pemrograman aplikasi *smartphone* yang bersifat *open-source*, yang artinya pengembang aplikasi *smartphone*