Implementasi Arsitektur Enterprise Pola Finansial pada Aplikasi Berbasis *Microservices*

M. Hanif Faturohman
Prodi Informatika – Program Sarjana
Universitas Islam Indonesia
Yogyakarta, Indonesia
17523082@students.uii.ac.id

Teduh Dirgahayu
Jurusan Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
teduh.dirgahayu@uii.ac.id

Hanson Prihantoro Putro
Jurusan Informatika
Universitas Islam Indonesia
Yogyakarta, Indonesia
hanson@uii.ac.id

***Abstrak*— Arsitektur enterprise merupakan cetak biru dari enterprise yang mewakili seluruh visi, misi, serta fungsionalitasnya. Sedangkan, *microservices* merupakan arsitektur teknologi yang sedang populer di berbagai enterprise dalam 5 tahun belakangan. Sejauh ini, literatur mengenai implementasi arsitektur enterprise pada aplikasi berbasis *microservices* masih sulit ditemukan. Masalah ini tentu menyulitkan beberapa pihak enterprise yang ingin mengimplementasikan arsitektur enterprisenya menjadi aplikasi berbasis *microservices*. Oleh karena itu, melalui makalah ini penulis bertujuan untuk menambah literatur terkait hal tersebut dengan menyediakan deskripsi mengenai bagaimana proses pengimplementasian arsitektur enterprise menjadi sebuah aplikasi berbasis *microservices*. Subjek penelitian yang digunakan pada makalah ini adalah arsitektur enterprise pola finansial yang sudah tersedia sebelumnya. Adapun metode yang digunakan adalah dengan mengamati proses pengimplementasian arsitektur enterprise pola finansial pada proyek aplikasi baru yang berbasis microservices. Pada implementasinya terdapat 3 tahap, yakni: perancangan ulang pola finansial, pengembangan aplikasi (basis data, API layanan, dan antarmuka) menggunakan kerangka kerja MERN *stack*, dan pengujian aplikasi. Diharapkan makalah ini dapat memberikan referensi terkait pengimplementasian arsitektur enterprise pada aplikasi berbasis *microservices*.**

Kata Kunci*— *arsitektur enterprise, arsitektur enterprise pola finansial, implementasi arsitektur enterprise, microservices, pola microservices, implementasi microservices, MERN stack

I. PENDAHULUAN

Arsitektur Enterprise (AE) menawarkan gambaran tingkat tinggi mengenai sebuah enterprise dari segi bisnis, sistem IT-nya, beserta korelasinya [1]. Gambaran tingkat tinggi tersebut mencakup deksripsi mengenai tujuan, visi, strategi, informasi entitas, proses bisnis, orang-orang yang terlibat, struktur organisasi, sistem pada aplikasi, infrastruktur teknologi atau perangkat keras [2].

Dalam membuat sebuah AE, seorang arsitek akan mulai dengan memodelkan perspektif dari seluruh pihak yang terkait. Selanjutnya untuk mengelola kompleksitasnya, arsitek akan mengkategorikan perspektif tersebut berdasarkan ranah arsitekturnya lalu menyusunnya ke dalam skema lapisan sesuai kerangka kerja AE yang digunakan [3]. Kerangka kerja AE ini merupakan sebuah panduan dalam membuat AE. Saat ini sudah banyak kerangka kerja yang dapat digunakan oleh enterprise untuk memodelkan AE, misalnya: TOGAF, DoDAF, FEAF, TEAF, dan Zachman [4].

Pada situasi dan kondisi tertentu, berbagai enterprise kerap kali mendapat masalah yang sama mengenai ranah arsitektur tertentu [3]. Sebagai contoh, Thierry Perroud menyebutkan bahwa jika kondisi pengolahan finansial enterprise buruk, maka 30% enterprise akan melewatkan potongan pembayaran lebih awal, 27% enterprise akan terkena masalah keterlambatan pembayaran [3]. Dan masih banyak kejadian lain yang serupa pada beberapa enterprise.

Kejadian tersebut memicu penelitian lebih lanjut mengenai implementasi konsep pola pada AE. Konsep tersebut sebelumnya sudah sukses diterapkan pada bidang pengembangan aplikasi untuk menyelesaikan masalah yang kerap kali muncul ketika situasi dan kondisinya terpenuhi saat mengembangkan aplikasi [3]. Walaupun penelitian mengenai penerapan pola pada AE masih cukup dini, pada tahun 2009 TOGAF menyatakan antusiasmenya bahwa konsep ini disinyalir akan semakin mempermudah para arsitek ketika mengembangkan AE ke depannya [3].

Pada beberapa tahun belakangan ini sedang terjadi banyak revolusi arsitektur teknologi di berbagai enterprise dari arsitektur monolitik ke arsitektur *microservices*. Di tahun 2020, sudah terdapat 78% organisasi yang menggunakan *microservices*, dan 61% di antaranya baru menggunakan *microservices* sejak 1-5 tahun belakangan ini [5]. Keterbatasan arsitektur monolitik pada skalabilitas dan ketergantungan kuat antar modulnya membuat beberapa enterprise kesulitan untuk mengimbangi kompleksitas bisnisnya yang makin meningkat dari tahun ke tahun dan beralih ke *microservices* [6].

Arsitektur *microservices* ini merupakan arsitektur yang menekankan kepada pembagian aplikasi menjadi layanan-layanan yang berukuran kecil, bersifat independen, serta dapat saling berkomunikasi [7]. Kecil di sini bermaksud bahwa tiap layanan hanya mewakili proses bisnis tertentu, sedangkan independen di sini bermaksud bahwa tiap layanan dapat berjalan otomatis tanpa terikat dengan layanan lain [7], [8].

Sejauh ini, literatur mengenai implementasi AE pada aplikasi berbasis *microservices* masih sulit ditemukan. Masalah ini tentu menyulitkan beberapa pihak enterprise yang ingin mengimplementasikan AE-nya menjadi aplikasi berbasis *microservices*. Oleh karena itu, disusunlah makalah ini dengan tujuan untuk menambah literatur terkait hal tersebut.

Subjek penelitian yang digunakan pada makalah ini adalah AE pola finansial yang sudah tersedia sebelumnya di buku [3]. AE pola finansial ini merupakan generalisasi dari AE berbagai departemen finansial pada sebuah enterprise. AE pola finansial ini bertujuan untuk mengelola aset dan ekuitas milik enterprise [3].

Dalam makalah ini, disajikan deskripsi tentang bagaimana proses pengimplementasian AE pola finansial menjadi aplikasi berbasis *microservices*. Makalah ini

diharapkan dapat memberikan referensi terkait pengimplementasian AE pada aplikasi berbasis *microservices*.

II. KAJIAN PUSTAKA

A. Arsitektur Enterprise (AE)

Ranah arsitektur dan skema lapisan yang akan digunakan dalam makalah ini akan merujuk kepada kerangka kerja TOGAF, yakni: ranah arsitektur bisnis, ranah arsitektur data atau informasi, ranah arsitektur aplikasi, dan ranah arsitektur teknologi. TOGAF dipilih karena struktur ranah arsitekturnya lebih sederhana dan mudah dipahami. Meski begitu, TOGAF sudah dapat dianggap setara dengan kerangka kerja lainnya [9]. TOGAF sendiri disusun oleh konsorsium bertingkat internasional yang mendorong pengembangan standar teknologi terbuka dan netral.

Berikut ini merupakan 4 skema lapisan AE dalam kerangka kerja TOGAF [3].

1) Ranah Arsitektur Bisnis

Lapisan ini mendeskripsikan struktur dan proses bisnis pada organisasi seperti: strategi penjualan produk atau jasa, struktur organisasi, dan sebagainya. Lapisan ini sangat penting sehingga lapisan-lapisan yang lain bergantung pada lapisan ini.

2) Ranah Arsitektur Data

Lapisan ini mendeskripsikan detail data yang digunakan pada proses bisnis seperti: bentuk data, relasi data, format data, dan sebagainya.

3) Ranah Arsitektur Aplikasi

Lapisan ini mendeskripsikan segala aspek yang berkaitan dengan aplikasi untuk mendukung proses bisnis.

4) Ranah Arsitektur Teknologi

Lapisan ini mendeskripsikan infrastruktur teknologi yang digunakan untuk menjalankan aplikasi dalam rangka mendukung proses bisnis.

Penggunaan AE yang baik dapat mendatangkan beberapa keuntungan, di antaranya adalah: meminimalisir kesalahan dalam membuat keputusan; menurunkan kompleksitas proses bisnis dalam enterprise; meningkatkan performa enterprise secara keseluruhan; dan dapat mengukur visi enterprise [3].

B. AE Pola Finansial

TOGAF memang merekomendasikan penggunaan pola arsitektur enterprise sebagai blok-blok penyusun dalam AE untuk menyelesaikan beberapa masalah yang berulang [10]. Namun sayangnya, TOGAF tidak menyediakan lebih lanjut pola-pola tersebut. Dikarenakan hal itu, pola yang digunakan pada makalah ini akan berasal dari buku *Enterprise Architechture Pattern* [3]. Kebetulan pola-pola dalam buku tersebut juga dibangun di atas skema lapisan TOGAF.

Pola yang akan diimplementasikan pada penelitian ini adalah pola finansial. Finansial merupakan cabang ilmu yang mempelari mengenai sebab-akibat sebuah aset terhadap individu atau organisasi dalam jangka waktu pendek maupun panjang [3]. Pola finansial ini dapat diimplementasikan di berbagai perusahaan, baik besar maupun kecil. Pola ini biasanya dipakai untuk membantu perusahaan mengelola aset beserta ekuitasnya [3].

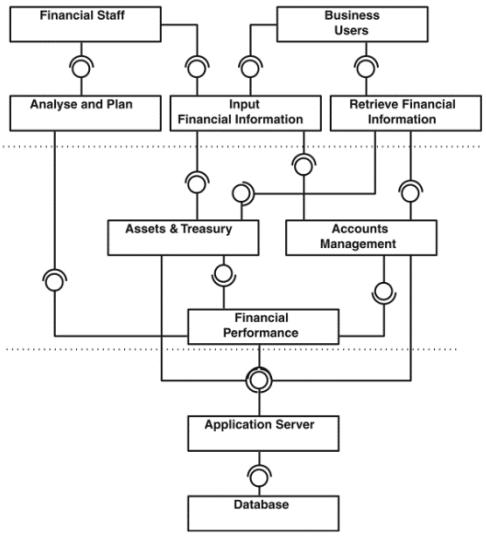
Berikut merupakan beberapa masalah finansial yang dialami banyak organisasi [3]:

- Tagihan dibayarkan terlalu lama, sehingga membuat perusahaan melewatkan potongan pembayaran jangka pendek atau bahkan terkena denda karena melebihi batas pembayaran.
- Proses manajemen informasi finansial secara manual terlalu melelahkan.
- Banyak kesalahan pada laporan finansial.
- Secara keseluruhan biaya manajemen finansial terlalu besar.
- Pengiriman tagihan kepada alamat yang salah.
- Data finansial yang hilang atau rusak.

Berdasarkan permasalahan di atas, berikut ini merupakan AE pola finansial dari 3 perspektif [3].

1) Perspektif Holistik

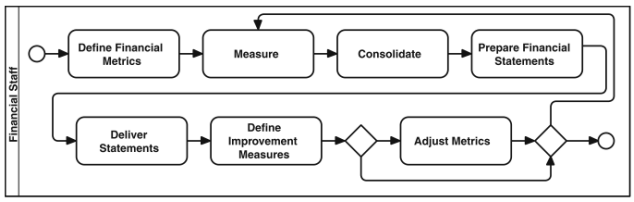
Perspektif holistik pada Gambar 1 menjelaskan garis besar dari 3 ranah arsitektur yang masing-masingnya dipisahkan menggunakan garis putus-putus. Pada lapisan paling atas termuat ranah arsitektur bisnis yang berupa aktor beserta proses bisnisnya. Pada lapisan tengah termuat ranah arsitektur aplikasi. Pada lapisan paling bawah termuat ranah arsitektur teknologi. Detail tiap lapisan akan dijelaskan lebih lanjut pada perspektif selanjutnya.



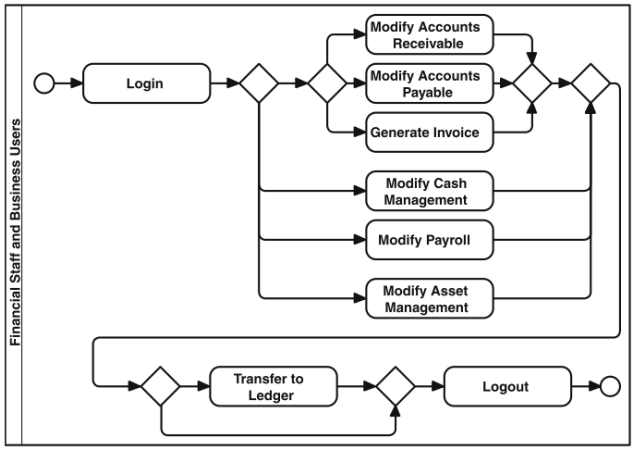
Gambar 1. Perspektif holistik pola finansial [3]

2) Perspektif Proses Bisnis

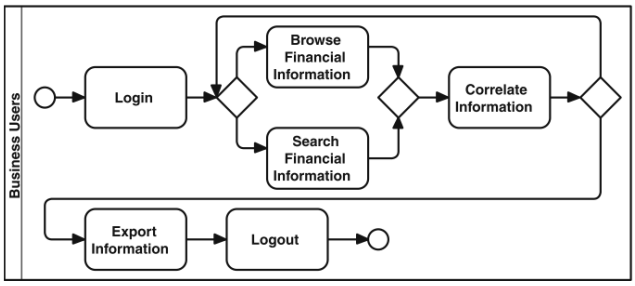
Perspektif ini memuat proses bisnis dari pola finansial secara lebih detail. Pada Gambar 2 termuat proses bisnis analisis dan perencanaan. Pada Gambar 3 termuat proses bisnis masukan informasi finansial. Pada Gambar 4 termuat proses bisnis baca informasi finansial.



Gambar 2. Proses bisnis analisis dan perencanaan [3]

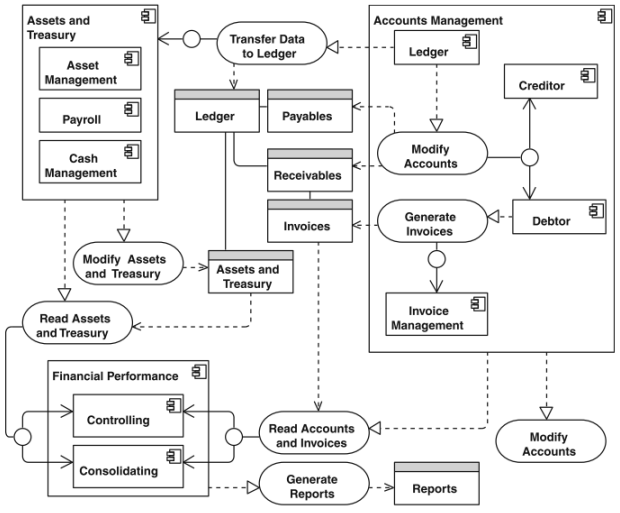


Gambar 3. Proses masukan informasi finansial [3]



Gambar 4. Proses baca informasi finansial [3]

3) Perspektif Data dan Aplikasi



Gambar 5. Perspektif data dan aplikasi [3]

Perspektif ini memuat seluruh aliran pada aplikasi serta skema data yang digunakan. Dalam Gambar 5, terdapat 3 kelompok aplikasi yakni:

a) Asset & Treasury,

Dalam kelompok *asset and treasury* terdapat 3 aplikasi yang dapat mengakses 3 layanan serta 1 basis data. Tiga aplikasi tersebut yakni: *asset management*, *payroll*, *cash management*. Tiga layanan tersebut yakni: *modify asset and treasury*, *read asset and treasury*, *transfer to ledger*. Satu basis data tersebut yakni: *asset and treasury.*

b) Accounts Management,

Kelompok *accounts management* memiliki 4 aplikasi yang dapat mengakses 4 layanan serta 3 basis data. Empat aplikasi tersebut yakni: *ledger*, *creditor*, *debtor*, *invoice management.* Empat layanan tersebut yakni: *transfer data to ledger*, *modify accounts*, *generate invoice*, *read accounts and invoice.* Tiga basis data yakni*: payables, receivable*, *invoices.*

c) Financial Performance

Kelompok *financial performance* memiliki 2 aplikasi yang dapat mengakses 3 layanan serta 2 basis data. Dua aplikasi tersebut yakni: *controlling*, *consolidating.* Tiga layanan tersebut yakni: *read assets and treasury*, *read accounts and invoices*, *generate reports.* Satu basis data tersebut yakni: *reports*.

C. Microservices

Mengembangkan aplikasi sebagai layanan-layanan kecil yang independen memberikan beberapa keuntungan dan kerugian [8], [11]. Keuntungan tersebut yakni: fleksibilitas pemilihan teknologi, resiliensi lebih terhadap *bug*, kemudahan dalam skalabilitas, kemudahan dalam mengembangkan masing-masing layanan. Sedangkan kerugiannya yakni: kerumitan dalam mengelola seluruh layanan, sulit menjaga konsistensi terhadap data, serta risiko kegagalan komunikasi yang cukup besar.

Microservices memiliki pola-pola yang sudah tersedia untuk mempermudah pengembangannya [6], [12]. Pola yang digunakan dalam makalah ini adalah pola saga dan pola orkestrasi. Pola saga merupakan pola yang digunakan untuk menjaga integritas data saat aplikasi melakukan transaksi yang melibatkan lebih dari 1 basis data. Pola orkestrasi merupakan pola yang menyimpan seluruh urutan aktivitas proses bisnis pada 1 layanan. Selanjutnya, 1 layanan tersebut mendelegasikan aktivitas-aktivitasnya kepada layanan lain untuk dikerjakan dan keluarannya akan disimpan kembali pada 1 layanan orkestrasi tersebut.

D. MERN Stack

MERN *stack* merupakan istilah dari susunan 4 kerangka kerja yang digunakan untuk membangun aplikasi berbasis web [13], [14]. Secara berurutan MERN tersusun atas: MongoDB, ExpressJS, NodeJS, dan ReactJS. MongoDB merupakan basis data terbuka yang berorientasi dokumen. MongoDB memiliki keunggulan pada fleksibilitas struktur data. Selanjutnya, ExpressJS merupakan kerangka kerja terbuka berbasis Javascript yang dibangun di atas platform NodeJS. Dalam MERN, ExpressJS digunakan untuk mengembangkan aplikasi bagian *server*. ExpressJS ini memiliki keunggulan pada proses pengembangannya yang sangat mudah dan cepat [15].

Platform berikutnya yaitu NodeJS yang merupakan *platform* terbuka yang digunakan untuk menjalankan kode Javascript di luar *browser*. Hal tersebut membuat Javascript

dapat digunakan untuk menjalankan sebuah *server*. NodeJS ini memiliki keunggulan pada eksekusi kodenya yang dapat dijalankan secara *asynchronous.* Kemudian terdapat ReactJS yang merupakan kerangka kerja terbuka khusus antarmuka yang berbasis bahasa Javascript. ReactJS memiliki keunggulan pada efisiensi kode dalam proses pengembangan aplikasi [14].

E. Penelitian Serupa

Kajian Pustaka ini mengkaji 4 penelitian serupa mengenai implementasi AE. Pustaka nomor [16] membahas mengenai rancangan implementasi AE pada perusahaan tambang. Pustaka nomor [17] membahas mengenai metode perancangan AE pada PT. Bestonindo Central Lestari. Pustaka nomor [18] membahas mengenai metode perancangan AE pada PT. Tin Tin menggunakan kerangka kerja Zachman. Pustaka nomor [3] membahas implementasi AE pada proyek aplikasi.

Selain itu, dikaji juga 5 penelitian serupa mengenai implementasi *microservices.* Pustaka nomor [19], [20] membahas mengenai migrasi proyek monolitik ke *micoservices.* Pustaka nomor [21] membahas mengenai pembuktian keuntungan pada implementasi *microservices*. Pustaka nomor [22] membahas mengenai implementasi *microservices* sebagai solusi dari masalah pada sistem *E-government.* Pustaka nomor [23] membahas mengenai deskripsi implementasi berdasarkan rumusan kebutuhan.

Dari seluruh penelitian serupa yang sudah dikaji, hanya 1 penelitian yang membahas mengenai implementasi AE pada proyek aplikasi. Tetapi sayangnya penelitian pada buku [3] menggunakan arsitektur teknologi yang kurang relevan. Selain itu, belum ditemukan juga penelitian yang membahas mengenai implementasi *microservices* yang didasari oleh AE.

III. METODOLOGI PENELITIAN

A. Kajian Pustaka

Kajian Pustaka memiliki lima bagian yang dikaji berurutan mulai dari AE, AE pola finansial, *microservices*, MERN *stack*, dan penelitian serupa. Literatur terkait AE dikaji dengan kata kunci arsitektur enterprise. AE pola finansial dikaji berdasarkan sebuah buku [3]. *Microservices* dikaji dengan kata kunci *microservices*. MERN *stack* dikaji dengan kata kunci "MERN". Penelitian serupa dikaji dengan kata kunci *implementation of enterprise architechiture in system information* dan kata kunci *implementation of microservices*.

B. Perancangan AE Pola Finansial

Perancangan AE finansial menggunakan pola dasar yang sudah disediakan dalam sebuah buku [3] dan akan dimodifikasi pada beberapa lapisan. Ranah arsitektur bisnis tidak mengalami perubahan. Ranah arsitektur data, aplikasi, dan teknologi dimodifikasi untuk menyesuaikan terhadap kebutuhan teknologi *microservices*.

C. Pengembangan Basis Data

Pada awalnya skema basis data dirancang menggunakan *entity relationship diagram*. Kemudian, diagram tersebut diubah menjadi bentuk dokumen dengan menjadikan entitas yang akan ditampilkan pada antarmuka sebagai dokumen utama. Sedangkan, entitas lain yang memiliki relasi dengan entitas utama akan dijadikan sub dokumen yang ditulis bersarang pada dokumen utama. Adapun teknologi yang digunakan dalam pengembangannya adalah MongoDB.

D. Pengembangan API Layanan

API layanan menggunakan AE pola finansial yang sudah dimodifikasi sebagai panduan dalam pengembangannya. API layanan juga menerapkan pola-pola yang terdapat pada *microservices* seperti: pola saga untuk menjaga integritas datanya, dan pola orkestrasi untuk menurunkan kompleksitas komunikasi antar layanan. Adapun teknologi yang digunakan dalam pengembangannya adalah ExpressJS, serta NodeJS.

E. Pengembangan Antarmuka

Antarmuka juga dikembangkan menggunakan AE pola finansial yang sudah dimodifikasi sebagai panduan pengembangannya. Antarmuka ini menggunakan arsitektur monolitik. Adapun teknologi yang digunakan dalam pengembangannya adalah ReactJS. Selain itu, digunakan paket *Ant Design* sebagai kerangka dalam mendesain antarmukanya.

F. Pengujian

Dalam implementasinya, pengujian akan dilakukan tiap satu fungsionalitas selesai dikembangkan. Pengujian ini dilakukan dengan cara menyiapkan *test case* secara manual lalu mengeksekusinya untuk mengindentifikasi cacat pada aplikasi. Seluruh pengujian akan dilakukan pada tingkat fungsional.

IV. HASIL

A. Arsitektur Aktual

Pada implementasinya terdapat beberapa proses bisnis yang tidak diimplementasikan serta beberapa yang ditambahkan. Berikut gambaran perubahan pada tiap perspektif AE pola finansial:

1) Perspektif Holistik

Mengacu pada Gambar 1, berikut perubahan yang terjadi pada perspektif holistik yang diklasifikasikan sebagai beberapa tingkat:

a) Lapisan 1: Ranah arsitektur bisnis

Tidak ada perubahan pada aktor maupun proses bisnisnya. Hanya saja proses verifikasi penggunanya dengan *login* dan *logout* tidak diimplementasikan karena keterbatasan waktu dan bukan prioritas utama. Dampaknya, tidak ada pembatasan hak akses pada aktor. Seluruh aktor dapat mengakses seluruh modul yang tersedia pada aplikasi.

b) Lapisan 2: Ranah arsitektur data dan aplikasi

Perubahan yang terjadi adalah pemecahan kelompok aplikasi menjadi kelompok yang lebih kecil. Hal ini dilakukan karena kelompok aplikasi yang sebelumnya dinilai kurang spesifik dari segi domain bisnisnya. Kelompok *financial performance* menjadi kelompok laporan. Kelompok *account management* dipecah menjadi kelompok akunting, dan invoice. Kelompok *asset & treasury* dipecah menjadi kelompok arus kas, kelompok logistik, kelompok sumber daya manusia.

c) Lapisan 3: Ranah arsitektur teknologi

Perubahan yang terjadi adalah penerjemahan nama dan pemecahan blok. Blok *application server* dipecah menjadi

blok API layanan dan blok antarmuka. Hal ini dilakukan karena *microservices* hanya akan diterapkan pada blok API layanan. Untuk blok *database* hanya terjadi penerjemahan nama menjadi basis data.

2) Perspektif Proses Bisnis

a) Proses Bisnis analisis dan perencanaan

Mengacu pada Gambar 2, hanya aktivitas *define improvement measure* dan *adjust metrics* yang diimplementasikan. Aktivitas lain diasumsikan dilakukan di luar sistem. Sebagai contoh: melalui rapat internal eksekutif, dan sebagainya.

b) Proses bisnis masukan informasi finansial

Mengacu pada Gambar 3, perubahan yang terjadi adalah penerjemahan nama aktivitasnya serta penghapusan aktivitas *login* dan *logout.*

c) Proses bisnis baca informasi finansial

Mengacu pada Gambar 4, perubahan yang terjadi adalah penghapusan aktivitas *login* dan *logout*. Selain itu, aktivitas *correlate information* dan *export information* tidak diimplementasikan. *Correlate information* diasumsikan dilakukan di luar sistem, sedangkan export information menggunakan fitur *print* pada *web browser*.

3) Perspektif Data dan Aplikasi

Sesuai modifikasi pada perspekif holistik yang aktual, terdapat 6 kelompok aplikasi baru. Tabel I menunjukkan bagian-bagian dari tiap kelompok aplikasi.

TABEL I. KELOMPOK APLIKASI AKTUAL

Kelompok	Antarmuka	Layanan	Basis data
Laporan	Laporan keuangan	-	-
Akunting	Akun	Akun	Akun
	Jurnal	*Ledger*	*Ledger*
	-	Posting ledger	-
	-	Rollback ledger	-
Invoice	*Invoice* penjualan	*Invoice*	*Invoice*
Arus kas	Transaksi kas	Arus kas	Arus kas
	Hutang dagang	Hutang dagang	Hutang Dagang
	Piutang Dagang	Piutang Dagang	Piutang Dagang
Logistik	Stok dagang	Stok dagang	Stok dagang
Sumber Daya Manusia	Karyawan	Penggajian	Penggajian
	Penggajian		

B. Basis Data

Sesuai pada Tabel I, terdapat 8 basis data yang masing-masing berdiri secara independen. Basis data akun menyimpan seluruh data akun beserta kategori dari akun. Basis data *ledger* menyimpan seluruh detail perubahan nominal dari akun dalam format pembukuan berpasangan. Basis data arus kas menyimpan seluruh detail transaksi yang berkaitan dengan kas. Basis data hutang dagang menyimpan seluruh detail transaksi pada proses pembelian stok dagang secara hutang.

Selanjutnya, basis data piutang dagang menyimpan seluruh detail transaksi pada proses penjualan stok dagang secara piutang. Basis data penggajian menyimpan seluruh transaksi yang berkaitan dengan penggajian karyawan beserta daftar karyawannya. Basis data inventaris menyimpan detail mengenai barang dagangan beserta transaksinya. Basis data *invoice* menyimpan data mengenai nota penagihan pembelian beserta detail pembayarannya.

C. API Layanan

Sesuai pada TABEL I, terdapat 10 API layanan yang masing-masing dikembangkan sebagai proyek kecil yang independen.

1) Layanan Akun

Layanan akun merupakan layanan untuk mengelola akun rekening dalam transaksi. Dalam layanan ini terdapat 7 fungsionalitas, yakni: Baca seluruh akun; Baca seluruh akun dengan id yang sesuai; Baca seluruh akun dengan nama akun yang sesuai; Tambah akun baru; Kelola informasi akun; Hapus akun; Baca kategori akun.

2) Layanan Ledger

Layanan *ledger* merupakan layanan untuk mencatat seluruh detail perubahan nominal yang terjadi pada akun setiap terjadi transaksi. Dalam layanan ini terdapat 5 fungsionalitas, yakni: Baca seluruh *ledger*; Baca seluruh *ledger* dengan id yang sesuai; Tambah *ledger* baru; Kelola informasi *ledger*; Hapus *ledger*.

3) Layanan Posting Ledger

Layanan *posting* ledger merupakan layanan orkestrasi proses tambah transaksi baru. Transaksi baru ini akan mempengaruhi layanan akun dan layanan *ledger*. Layanan ini juga mengimplementasikan pola saga. Dalam layanan ini terdapat 1 fungsionalitas saja, yakni: Tambah transaksi baru.

4) Layanan Rollback Ledger

Layanan *rollback ledger* merupakan layanan orkestrasi dari proses menghapus transaksi. Transaksi yang dihapus akan mempengaruhi layanan akun dan layanan *ledger*. Layanan ini juga mengimplementasikan pola saga. Dalam layanan ini terdapat 1 fungsionalitas saja, yakni: Balik transaksi *ledger*.

5) Layanan Arus Kas

Layanan arus kas merupakan layanan yang mengelola seluruh transaksi terhadap akun kas dalam bentuk tunai. Dalam layanan ini terdapat 3 fungsionalitas, yakni: Baca seluruh data transaksi kas; Tambah transaksi kas baru; Hapus transaksi kas.

6) Layanan Piutang Stok Dagang

Layanan piutang stok dagang merupakan layanan yang mengelola proses penjualan stok dagang secara piutang. Dalam layanan ini terdapat 4 fungsionalitas, yakni: Baca seluruh data transaksi piutang dagang; Tambah transaksi piutang dagang baru; Tambah transaksi pembayaran cicilan piutang; Hapus transaksi piutang dagang.

7) Layanan Hutang Stok Dagang

Layanan hutang stok dagang merupakan layanan yang mengelola proses pembelian stok dagang secara hutang. Dalam layanan ini terdapat 4 fungsionalitas, yakni: Baca seluruh data transaksi hutang dagang; Tambah transaksi hutang dagang baru; Tambah transaksi pembayaran cicilan hutang; Hapus transaksi piutang hutang; Layanan Penggajian.

8) Layanan Penggajian

Layanan penggajian merupakan layanan yang mengelola proses penggajian secara tunai. Dalam layanan ini terdapat 4 fungsionalitas yakni: Baca seluruh data tansaksi penggajian; Tambah transaksi penggajian baru; Hapus transaksi penggajian; Baca seluruh data karyawan

9) Layanan Stok Dagang

Layanan stok dagang merupakan layanan yang mengelola stok dagang yang tersedia beserta transaksi pembeliannya. Dalam layanan ini terdapat 3 fungsionalitas, yakni: Baca seluruh transaksi stok dagang; Tambah transaksi stok dagang baru; Hapus transaksi stok dagang.

10) Layanan Invoice

Layanan *invoice* merupakan layanan yang mengelola nota penagihan. Dalam layanan ini terdapat 4 fungsionalitas, yakni: Baca seluruh data *invoice*; Tambah data *invoice* baru; Tambah transaksi pembayaran *invoice*; Hapus data *invoice.*

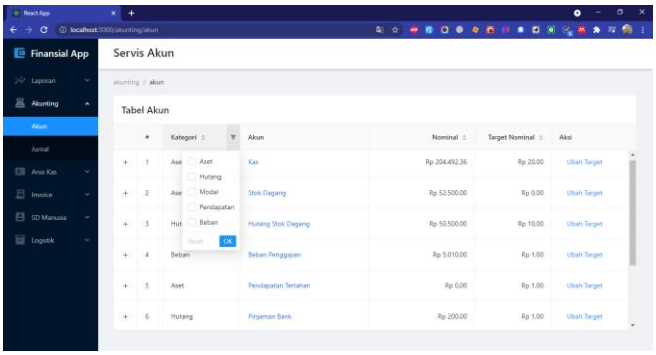
D. Antarmuka

Sesuai pada Tabel I, terdapat 10 antarmuka yang digunakan untuk mengakses masing-masing layanannya. Antarmuka laporan keuangan menampilkan laporan terkait laba rugi sebuah perusahaan.

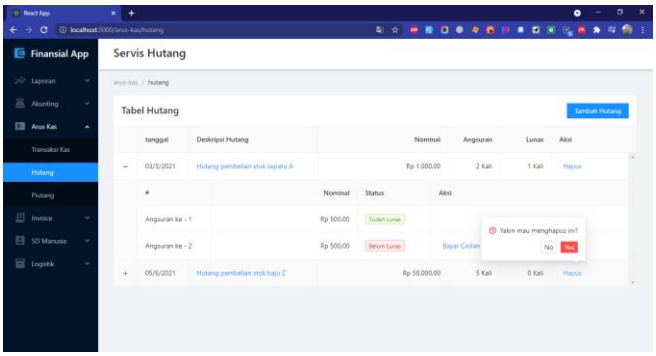
Antarmuka akun pada Gambar 6 menampilkan seluruh akun rekening beserta nominalnya saat ini. Disini ditampilkan juga target nominal dari akun yang ingin dicapai tiap periode. Selain itu, antarmuka ini memiliki fungsionalitas untuk menyaring data akun dan mengurutkan data akun. Antarmuka jurnal menampilkan seluruh detail perubahan nominal yang terjadi pada tiap akun.

Selanjutnya, Antarmuka *invoice* penjualan mengelola seluruh nota penagihan beseta status pelunasannya. Antarmuka transaksi kas mengelola seluruh transaksi yang hanya berkaitan dengan akun kas. Antarmuka hutang dagang pada Gambar 7 mengelola seluruh transaksi yang hanya berkaitan dengan pembelian stok dagang secara hutang. Seperti yang terlihat pada Gambar 7 dalam antarmuka ini pengguna dapat menambah hutang dagang, menghapus hutang dagang, serta melunaskan cicilan hutang.

Kemudian, antarmuka piutang dagang mengelola transaksi yang hanya berkaitan dengan penjualan produk dagangan secara piutang. Antarmuka stok dagang menampilkan seluruh transaksi yang berkaitan dengan stok dagang. Antarmuka karyawan menampilkan daftar karyawan yang bekerja. Antarmuka penggajian menampilkan pengelolaan gaji karyawan tiap periodenya.



Gambar 6. Antarmuka akun



Gambar 7. Antarmuka hutang dagang

E. Pengujian

Pada bagian API servis terdapat total 36 fungsionalitas, dan tiap fungsionalitas diuji minimal 1 kali. Pada bagian antarmuka terdapat total 55 fungsionalitas, dan tiap fungsionalitasnya juga diuji minimal 1 kali. Dalam prosesnya, jika terdapat keluaran fungsionalitas yang tidak sesuai harapan, maka fungsionalitas akan diperbaiki lalu diuji ulang. Hal tersebut dilakukan hingga keluaran sesuai dengan yang diharapkan. Di akhir pengujian, seluruh keluaran pada API servis dan antarmuka sudah sesuai harapan.

V. PEMBAHASAN

Tujuan AE pola finansial untuk mengelola aset dan ekuitas dapat dicapai setelah diimplementasikan. Untuk pembuktiannya, penulis mencoba menggunakan aplikasi untuk menyelesaikan studi kasus sederhana mengenai keuangan enterprise.

Pengembangan aplikasi *microservices* terbukti mudah ketika mengembangkan dan memperbaiki *bug* pada masing-masing layanan. Hal tersebut dikarenakan masing-masing layanan pada *microservices* bersifat independen. Jadi, lingkup pengembangan serta perbaikan *bug* hanya terbatas per layanan saja.

Pengembangan aplikasi *microservices* terbukti sulit ketika mengelola seluruh komunikasi dan menjaga integritas data antar layanannya. Pada *microservices* sering terdapat beberapa duplikasi data antar layanannya. Kasus kegagalan komunikasi antar layanan dapat membuat data yang seharusnya sama antar layanan menjadi berbeda. Diperlukan kode *exception error* yang cukup kompleks untuk mengatasi hal tersebut.

Penggunaan MERN *stack* terbukti menghemat waktu pengembangan. Hal tersebut dikarenakan penggunaan basis

data berorientasi dokumen yang tidak terikat dengan skema mampu menghemat waktu ketika melakukan penyesuaian struktur data. Selanjutnya ExpressJS dan ReactJS yang sama-sama berbasis bahasa Javascript juga dapat menghemat waktu untuk belajar. Selain itu, prinsip penggunaan ulang kode pada ReactJS juga sangat menghemat waktu ketika menulis kode program.

VI. KESIMPULAN

Berdasarkan penelitian yang sudah dilakukan, terdapat 3 tahapan utama dalam mengimplementasikan AE menjadi aplikasi berbasis *microservices*.

A. Tahap perancangan AE atau pola AE

Dalam tahap ini dilakukan proses perancangan AE atau pola AE yang akan diimplementasikan. Tergantung situasi dan kondisinya, perancangan ini dapat berupa membuat AE atau pola AE baru ataupun memodifikasi AE atau pola AE yang sudah tersedia. AE ataupun pola AE yang dirancang di tahap ini setidaknya harus memiliki gambaran yang setara dengan pola AE finansial yang digunakan pada makalah ini.

B. Tahap pengembangan aplikasi

Dalam tahap ini dilakukan proses pengembangan aplikasi berdasarkan AE atau pola AE yang sudah dirancang sebelumnya. Terdapat 3 hal utama yang harus dikembangkan dalam tahap ini, yakni: basis data, API layanan, dan antarmuka. Arsitektur *microservices* ini diterapkan pada API layanan dan berdampak langsung pada struktur basis data. Pada praktiknya, aplikasi berbasis *microservices* ini memiliki beberapa pola yang dapat digunakan untuk mengatasi masalah yang sering terjadi pada proses pengembangannya.

C. Tahap pengujian aplikasi

Dalam tahap ini dilakukan proses pengujian dari aplikasi yang sudah dikembangkan pada tahap sebelumnya. Pengujian ini dilakukan untuk memastikan bahwa seluruh fungsionalitas sudah berjalan sesuai dengan yang diharapkan.

VII. SARAN

Makalah ini hanya menyajikan mengenai deskripsi proses implementasi AE pola finansial menjadi aplikasi berbasis *microservices.* Finansial sendiri merupakan domain enterprise yang sudah ada sejak ratusan tahun yang lalu. Adapun saat ini terdapat banyak domain enterprise. Maka dari itu, untuk menyempurnakan penelitian ini perlu penelitian lebih lanjut mengenai penerapan AE pada domain lain menjadi aplikasi berbasis *microservices.*

DAFTAR PUSTAKA

[1] T. Tamm, P. B. Seddon, G. Shanks, and P. Reynolds, "How does enterprise architecture add value to organisations?," *Commun. Assoc. Inf. Syst.*, vol. 28, no. 1, pp. 141–168, 2011, doi: 10.17705/1cais.02810.

[2] C. M. Pereira, "Enterprise Architecture : Business and IT Alignment," pp. 1344–1345, 2005.

[3] T. Perroud and R. Inversini, *Enterprise Architecture Patterns*. 2013.

[4] B. D. Rouhani, M. N. Z. R. Mahrin, F. Nikpay, R. B. Ahmad, and P. Nikfard, "A systematic literature review on Enterprise Architecture Implementation Methodologies," *Inf. Softw. Technol.*, vol. 62, no. 1, pp. 1–20, 2015, doi: 10.1016/j.infsof.2015.01.012.

[5] M. Loukides and S. Swoyer, "Microservices Adoption in 2020 – O'Reilly," Jul. 15, 2020. https://www.oreilly.com/radar/microservices-adoption-in-2020/ (accessed Jun. 06, 2021).

[6] K. Indrasiri and P. Siriwardena, *Microservices for the Enterprise: Designing, Developing, and Deploying*. 2018.

[7] J. Lewis and M. Fowler, "Microservices," 2014. https://martinfowler.com/articles/microservices.html (accessed May 10, 2021).

[8] S. Newman, *Building Microservices*. 2015.

[9] R. Winter and R. Fischer, "Essential Layers , Artifacts , and Dependencies of Enterprise Architecture," 2006.

[10] "The TOGAF Standard, Version 9.2 - Architecture Patterns." https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap22.html (accessed Jun. 06, 2021).

[11] C. Richardson, "Developing event-driven microservices with event sourcing and CQRS svcc 2015.key."

[12] C. Richards, *Chris Richards*, vol. 2018, no. March. 2018.

[13] M. P. Widodo, "Pengembangan Aplikasi Pelaporan Progress-Plan- Problem untuk Manajemen Tugas dan Penentuan OKR di Krafthaus Indonesia."

[14] A. Samikshya, "MERN STACK WITH MODERN WEB PRACTICES-Developers Connecting Application," 2020.

[15] S. Aggarwal and J. Verma, "Comparative analysis of MEAN stack and MERN stack," *Int. J. Recent Res. Asp.*, vol. 5, no. 1, pp. 127–132, 2018.

[16] I. Ilin, A. Levina, and O. Iliashenko, "Enterprise architecture approach to mining companies engineering," *MATEC Web Conf.*, vol. 106, 2017, doi: 10.1051/matecconf/201710608066.

[17] M. O. Riku and D. B. Setyohadi, "Strategic plan with enterprise architecture planning for applying information system at PT. Bestonindo Central Lestari," *2017 5th Int. Conf. Cyber IT Serv. Manag. CITSM 2017*, 2017, doi: 10.1109/CITSM.2017.8089274.

[18] P. Sidiq and D. Bhakti, "Architecture Planning Information Manufacture Systems Using Enterprise Architecture Planning At PT. Tin Tin," no. January 2020, 2020, doi: 10.4108/eai.11-7-2019.2298090.

[19] A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara, "From Monolithic to Microservices: An Experience Report from the Banking Domain," *IEEE Softw.*, vol. 35, no. 3, pp. 50–55, 2018, doi: 10.1109/MS.2018.2141026.

[20] C. Y. Fan and S. P. Ma, "Migrating Monolithic Mobile Application to Microservice Architecture: An Experiment Report," *Proc. - 2017 IEEE 6th Int. Conf. AI Mob. Serv. AIMS 2017*, pp. 109–112, 2017, doi: 10.1109/AIMS.2017.23.

[21] W. Hasselbring and G. Steinacker, "Microservice architectures for scalability, agility and reliability in e-commerce," *Proc. - 2017 IEEE Int. Conf. Softw. Archit. Work. ICSAW 2017 Side Track Proc.*, pp. 243–246, 2017, doi: 10.1109/ICSAW.2017.11.

[22] H. Suryotrisongko, D. P. Jayanto, and A. Tjahyanto, "Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring

Boot," *Procedia Comput. Sci.*, vol. 124, pp. 736–743, 2017, doi: 10.1016/j.procs.2017.12.212.

[23] M. Wu, X. Ding, and R. Hou, "Design and implementation of B2B E-commerce platform based on microservices architecture," *ACM Int. Conf. Proceeding Ser.*, pp. 30–34, 2019, doi: 10.1145/3339363.3339369.