

APLIKASI APOTEK BERBASIS WEB MENGGUNAKAN ARSITEKTUR MICROSERVICES (STUDI KASUS APOTEK GLEN, KAB.TOBA)

Noplin Siagian¹, Tiarro Elprida Tamba², Hansen Henok O Situmorang³, Hernawati Susanti Samosir^{*4}
D3 Teknologi Informasi, Institut Teknologi Del Laguboti

*Penulis Korespondensi: hernawati@del.ac.id

Received 1 October 2021; Revised 9 November 2021; Accepted 17 November 2021

ABSTRAK

Dalam bidang kesehatan, teknologi informasi berkembang pesat bagi masyarakat khususnya di pedesaan. Salah satu penerapannya di bidang kesehatan pada sistem informasi. Apotek Glen merupakan salah satu toko obat di wilayah Kabupaten Toba. Saat ini sistem layanan apotek seperti transaksi penjualan obat-obatan masih dilakukan dengan cara manual yaitu dengan mencatat ke dalam buku sehingga dibangun aplikasi Apotek Glen berbasis web yang menggunakan arsitektur *microservices*. Alasan Aplikasi Apotek menggunakan arsitektur *microservices* adalah karena lebih skalabel (terukur) dan fleksibel sehingga dapat memenuhi kebutuhan pengguna. Namun pada umumnya beberapa aplikasi dibangun menggunakan arsitektur monolitik karena banyak diimplementasikan oleh *developer*. Tujuan utama penelitian dari aplikasi apotek yang dibangun adalah untuk mengimplementasikan arsitektur *microservices* dan menganalisis kedua arsitektur tersebut berdasarkan beberapa indikator yaitu struktur aplikasi, bahasa pemrograman, *database* dan deployment. Metode penelitian yang digunakan dalam pengembangan aplikasi apotek Glen berbasis web dilihat dari arsitektur *microservices*, dan arsitektur monolitik. Aplikasi yang menggunakan arsitektur monolitik diuji secara manual, sedangkan aplikasi yang menggunakan arsitektur *microservices* diuji menggunakan *Postman*. Hasil dari penelitian ini adalah aplikasi apotek berbasis web menggunakan arsitektur *microservice* dan perbedaan arsitektur *microservices* dengan arsitektur monolitik yang masing-masing memiliki karakteristik tersendiri. Arsitektur monolitik merupakan sebuah arsitektur yang menjalankan semua layanan yang berada dalam sebuah *server* sedangkan arsitektur *microservices* merupakan sebuah arsitektur yang bersifat *independent* yang terdiri dari berbagai *service* kecil yang memiliki tanggung jawab sendiri dan setiap *service* dapat saling berkomunikasi satu sama lain menggunakan API (*Application Programming Interface*).

Kata Kunci: *apotek, monolitik, microservices, API (Application Programming Interface)*

ABSTRACT

In the health sector, information technology is growing rapidly for the community, especially in rural areas. One of its applications in the health sector is in information systems. Apotek Glen is one of the drug stores in the Toba Regency area. Currently, the pharmacy service systems such as drug sales transactions are still carried out manually, namely by recording in a book so that a web-based Apotek Glen application was built that uses the microservices architecture. The reason the Pharmacy Application uses the microservices architecture is because it is more scalable (measured) and flexible so that it can meet user needs. However, in general, some applications are built using a monolithic architecture because many are implemented by developers. The main purpose of research from the pharmacy application that was built is to implement the microservice architecture and analyze the two architectures based on the application structure, programming language, database, and deployment. The research method used in developing web-based Glen pharmacy applications is seen from the microservices architecture and monolithic architecture. Applications using a monolithic architecture are tested manually, while applications using a microservices architecture are tested using Postman. The result of this research is a web-based pharmacy application using microservice architecture and the difference between microservices architecture and monolithic architecture, each of which has its own characteristics. The monolithic architecture is the architecture which runs all the services on the server whereas the microservices architecture is an independent architecture which consists of different small services which have responsibility for each service but each service. can communicate with each other using an API (Application Programming Interface) and if one service has a problem, the problem will not affect other services or the system can continue to work.

Keyword: pharmacy, monolithic, microservices, API (Application Programming Interface)

I. PENDAHULUAN

Perkembangan teknologi informasi yang sangat pesat telah berperan penting sebagai pendorong kemajuan dunia bisnis dalam mengolah data yang lebih baik, akurat dan efisien. Teknologi informasi telah mencakup keseluruhan bidang seperti pendidikan, kesehatan, ekonomi, industri, pemerintahan dan lainnya. Salah satu penerapan teknologi informasi dibidang kesehatan telah menggunakan sebuah sistem informasi pelayanan kesehatan untuk membantu dan mempermudah pihak kesehatan maupun masyarakat sesuai kebutuhan masing-masing. Saat ini, apotek merupakan salah satu yang menggunakan sistem informasi tersebut.

Apotek merupakan toko tempat penjualan dan persediaan obat-obatan baik berupa sirup, tablet/kapsul, salep, injeksi, dan jenis obat lainnya yang diperlukan masyarakat. Pengelolaan apotek menjadi tanggung jawab seorang apoteker yang profesional di bidang farmasi. Studi kasus pada tugas akhir ini adalah apotek Glen. Apotek "Glen" merupakan sebuah toko obat yang ada di Kabupaten Toba. Tepatnya di Jl. Dr.FI. Tobing, Pasar Porsea Kecamatan Porsea. Pada umumnya, kendala yang sering terjadi pada sistem layanan apotek adalah masih menggunakan transaksi penjualan obat-obatan yang dilakukan dengan cara manual yaitu dengan mencatatnya ke dalam buku dan masalah pencatatan stok obat kadaluarsa oleh petugas stok obat masih kurang terkontrol[1]. Selain itu, apoteker harus memeriksa dan menghitung obat secara manual sehingga diperlukan waktu yang cukup lama [2].

Gaya arsitektur suatu sistem memiliki beberapa jenis, contoh paling dikenal adalah monolitik dan *microservices*. Arsitektur monolitik merupakan sebuah arsitektur pengembangan aplikasi yang kode program, *database* dan tampilan program menjadi satu[3], dan semua layanan dijalankan dalam sebuah *server* sehingga satu *server* akan menangani semua layanan dalam aplikasi. Arsitektur monolitik jika diterapkan pada sebuah aplikasi yang semakin kompleks dan pengguna semakin banyak akan memengaruhi kenaikan biaya yang lebih besar, sulit melakukan *scaling* pada bagian tertentu, memiliki kompleksitas yang berukuran besar apabila terjadi kegagalan pada proses pembaruan atau penambahan fitur baru pada aplikasi yang disebut *resilient challenges*. Selain itu, apabila salah satu komponen pada sistem mengalami masalah, masalah tersebut akan memengaruhi komponen lainnya.

Berbeda hal dengan menggunakan arsitektur *microservices* yaitu untuk sistem terdistribusi, dimana arsitektur ini terdapat berbagai *service* kecil yang memiliki tanggung jawab bagi masing-masing *service*[4], akan tetapi *service-service* dapat berkomunikasi satu sama lain menggunakan API (*Application Programming Interface*). Jika salah satu *service* dari sistem mengalami masalah, masalah tersebut tidak akan memengaruhi komponen-komponen lainnya, sehingga komponen-komponen dalam sistem dapat terus bekerja[5]. Arsitektur *microservice* dapat membuat kode aplikasi lebih sedikit dan bersifat *independent* sehingga dapat melakukan pengujian aplikasi dengan cara terpisah, mempermudah pengembang untuk melakukan *scalability*, *developer* dapat bebas untuk mengembangkan aplikasi dengan berbagai jenis bahasa pemrograman ataupun *framework*[6].

Berdasarkan perbandingan antara arsitektur monolitik dan *microservices*, maka penulis sebagai tim pengembang melakukan analisis perbandingan dan mempertimbangkan agar

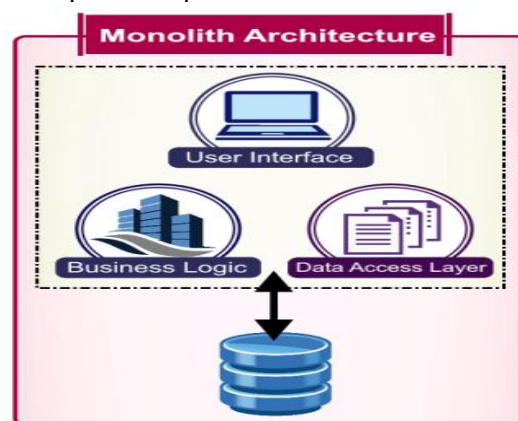
pengelolaan aplikasi apotek berbasis *web* lebih terukur dan secara fleksibel[7] yaitu dengan menggunakan arsitektur *microservices* yang dapat memenuhi kebutuhan dari pengguna yaitu transaksi penjualan obat.

II. STUDI LITERATUR

Pada bab ini dijelaskan mengenai metode penelitian yang digunakan selama penyelesaian pembangunan aplikasi apotek berbasis *web*.

1. Arsitektur Monolitik

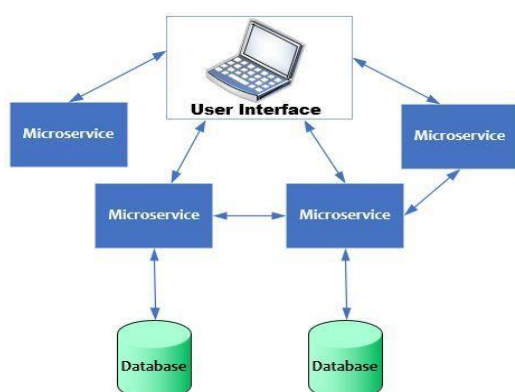
Arsitektur monolitik adalah arsitektur yang menggambarkan sebuah aplikasi yang menjalankan semua layanan yang terdapat dalam satu *server* sehingga satu *server* akan menangani semua layanan dalam aplikasi. Setiap aplikasi akan dijalankan secara bersamaan karena seluruh modul aplikasi dibungkus di dalam sebuah aplikasi besar. Pada arsitektur monolitik akan mengkompilasi keseluruhan kode program sistem menjadi satu aplikasi. Semua UI (antarmuka pengguna), *business*, and *database access logic* dikemas bersama ke dalam artefak aplikasi tunggal dan disebar ke server aplikasi. Diagram arsitektur monolitik dapat dilihat pada Gambar 1.



Gambar 1. Diagram Arsitektur Aplikasi Monolitik[8]

2. Arsitektur *Microservices*

Arsitektur *microservice* adalah suatu arsitektur yang dapat diterapkan pada sistem informasi yang dirancang untuk sistem terdistribusi. Selain itu, dalam arsitektur ini terdapat berbagai *service* kecil yang memiliki tanggung jawab bagi masing-masing *service*[4] akan tetapi masing-masing *service* dapat saling berkomunikasi satu sama lain menggunakan API (*Application Programming Interface*). Jika salah satu *service* dari sistem mengalami masalah, masalah tersebut tidak akan memengaruhi komponen-komponen lainnya, sehingga komponen-komponen dalam sistem dapat terus bekerja. *Microservice* membantu memecahkan masalah kompleksitas tradisional dalam basis kode yang besar dengan menguraikan basis kode besar menjadi potongan-potongan kecil yang terdefinisi dengan baik. *microservice* juga menguraikan dan memisahkan fungsionalitas aplikasi sehingga mereka benar-benar independen satu sama lain. Pada umumnya, dalam arsitektur *microservice*, semua layanan berkomunikasi satu sama lain dengan menggunakan HTTP sebagai media komunikasi API[9]. Setiap layanan berisi databasenya sendiri. Diagram arsitektur menggunakan *microservices* dapat dilihat pada Gambar 2.

Gambar 2. Diagram Arsitektur Aplikasi *Microservices*[8]

3. Apotek

Apotek merupakan toko tempat penjualan dan persediaan obat-obatan baik berupa sirup, tablet/kapsul, salep, injeksi, dan jenis obat lainnya yang diperlukan masyarakat. Apotek memiliki fungsi dan tugas diantaranya untuk sarana penyaluran perbekalan farmasi seperti obat-obatan yang diperlukan masyarakat, serta sebagai sarana farmasi untuk peracikan, pencampuran, pengubahan bentuk, dan penyerahan obat. Di beberapa lokasi yang cukup strategis khususnya di daerah pedalaman memiliki ketentuan dan perizinan untuk mendirikan sebuah apotek sebagai toko penjual obat-obatan. Seorang apoteker dikatakan profesional telah memiliki sertifikat secara resmi dan memiliki keahlian yang diakui oleh lembaga keprofesian di bidang farmasi.

4. API Gateway

API Gateway adalah sebuah server yang *single entry point* pada sistem yang dibangun. API Gateway bertanggung jawab pada arsitektur sistem internal dan sebagai penyedia API yang disesuaikan untuk setiap *client*. Selain itu, API Gateway memiliki fungsi seperti *permission verification, monitoring, load balancing, caching and monitoring* [10]. Semua permintaan dari *client* terlebih dahulu ditangani oleh API Gateway dan mengarahkan permintaan tersebut ke *microservice* yang sesuai. Salah satu contoh Api Gateway adalah Kong. Kong merupakan sebuah API Gateway untuk menjadi *entry point client* agar dapat mengakses *Application Programming Interface* (API) dan digunakan sebagai tempat untuk mendaftarkan *microservices* dan *routing* dari *microservices* yang terdaftar[11]. Contoh dari penggunaan *API Gateway* adalah pada *Netflix API Gateway*.

5. Restful API

Restful API merupakan salah satu model implementasi dari API (*Application Programming Interface*)[12]. API digunakan agar dapat saling berbagi data antar aplikasi yang berbeda. Sementara itu, *Restful* yaitu protokol/aturan untuk melakukan REST (*Representational State Transfer*) yakni suatu metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data. Tujuan *Restful* API membuat sistem memiliki performa yang baik, cepat dan mudah dikembangkan (*scale*) khususnya dalam pertukaran dan komunikasi data. Dalam hal ini, *Restful* API memiliki REST *Client* dan Rest server yang berkaitan apabila mengakses pada masing-masing *resource* (data). Pada data yang diberikan oleh

Rest server biasanya berupa format *text*, JSON maupun XML. Akan tetapi, JSON adalah format yang sering digunakan dan lebih populer. Terdapat beberapa metode HTTP yang dipakai dalam Restful API diantaranya[12]:

1. *GET* digunakan untuk menampilkan dan membaca data pada *REST server*
2. *POST* digunakan untuk memasukkan atau menambahkan sebuah data baru di dalam *REST server*
3. *PUT* digunakan untuk melakukan perubahan terhadap data yang sudah ada di dalam *REST server*
4. *DELETE* digunakan untuk menghapus data dari dalam *REST server*

III. HASIL DAN PEMBAHASAN

1. Analisis dan Hasil Implementasi

1.1. Analisis

1.1.1. Analisis Kondisi *Current System* di Apotek Glen Kab. Toba

Apotek merupakan sebuah toko yang menjual beberapa produk obat-obatan. Salah satunya adalah Apotek Glen yang ada di Kabupaten Toba. Tepatnya di Jl. Dr.FI. Tobing, Pasar Porsea Kecamatan Porsea. Penulis telah melakukan survei dengan melakukan wawancara kepada pemilik apotek Glen. Wawancara dilakukan pada Sabtu 28 November 2020. Dari hasil wawancara diperoleh informasi bahwa pada saat ini sistem yang digunakan pada apotek masih menggunakan cara manual dalam melakukan transaksi penjualan obat-obatan seperti pencatatannya ke dalam buku dan masalah pencatatan stok obat kadaluarsa oleh petugas stok obat masih kurang terkontrol. Selain itu, apoteker harus memeriksa dan menghitung obat secara manual sehingga diperlukan waktu yang cukup lama.

1.1.2. Analisis terhadap Implementasi Arsitektur Monolitik pada Aplikasi Apotek

Arsitektur monolitik merupakan sebuah arsitektur yang menggambarkan semua fungsi atau komponen yang dibangun berada di dalam satu *project*. Aplikasi Apotek menggunakan arsitektur monolitik menggunakan bahasa pemrograman Node.js dan menggunakan jenis *database* MongoDB.

1.1.3. Analisis terhadap Implementasi Arsitektur *Microservices* pada Aplikasi Apotek

Arsitektur *microservices* merupakan arsitektur perangkat lunak yang digunakan untuk mengembangkan suatu aplikasi yang dibentuk dengan rangkaian *service-service* kecil, dimana setiap *service* memiliki tanggung jawab sendiri dan dapat saling berkomunikasi. Aplikasi apotek menggunakan arsitektur *microservices* akan dibangun menggunakan bahasa pemrograman Node.js dan Go serta menggunakan jenis *database* MongoDB dikarenakan pada *database* MongoDB memiliki desain yang fleksibel yang dapat diubah dan memiliki performa yang lebih cepat saat melakukan penyimpanan data tidak perlu melakukan pengecekan struktur data, dan mudah dibaca. Terdapat 5 buah *services* yang membangun aplikasi ini, yaitu *User Service, Notification Service, Product Service, Cart Service, dan Order Service*.

1.2. Hasil Implementasi

1.2.1. Deskripsi Aplikasi Apotek

Aplikasi apotek yang dibangun merupakan aplikasi yang menggunakan arsitektur monolitik dan menggunakan arsitektur *microservices*. Aplikasi apotek menggunakan arsitektur monolitik dibangun dengan bahasa pemrograman *Node.js* sedangkan aplikasi apotek menggunakan *microservices* terdiri dari *setiap service* menggunakan bahasa pemrograman *Node.js* dan *Go*, *frontend* menggunakan *Vue.js* dan *Api Gateway* menggunakan *KONG*. *Tool* yang digunakan untuk membangun aplikasi apotek adalah menggunakan *Visual Studio Code*.

1.2.2. Fungsi pada Aplikasi Apotek

Implementasi fungsi pada pembangunan aplikasi apotek menggunakan arsitektur monolitik dapat dilihat pada Tabel 1 dan *service* yang digunakan pada aplikasi Apotek menggunakan arsitektur *microservices* dapat dilihat pada Tabel 2.

Tabel 1. Implementasi Use Case pada Aplikasi Apotek menggunakan Arsitektur Monolitik

No	Nama Use Case	Keterangan
1	Autentikasi	User dapat melakukan Register untuk mendaftarkan akun lalu dapat melakukan Login ke Aplikasi Apotek.
2	Edit profile	User dapat mengubah akun yang telah didaftarkan, seperti melakukan <i>Reset Password</i> .
3	Notifikasi	User dapat menerima notifikasi seperti pada saat melakukan <i>Reset Password</i> .
4	Mengelola Daftar Produk	Apoteker dapat mengelola produk seperti menambah daftar produk, mengubah produk dan menghapus produk.
5	Melihat Daftar Produk	User dapat melihat daftar produk yang tersedia.
6	Melakukan Pemesanan Produk	User dapat melakukan pemesanan terhadap produk.
7	Mengelola Keranjang Pesanan	User dapat mengelola keranjang pesanan seperti menambah produk atau menghapus produk dari daftar keranjang pesanan.

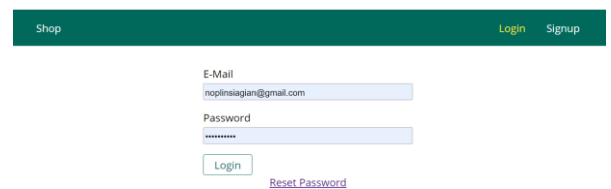
Tabel 2. Implementasi Service pada Aplikasi Apotek menggunakan Arsitektur *Microservices*

No	Nama Service	Keterangan
1	<i>User Service</i>	User dapat melakukan <i>Register</i> , lalu Login ke dalam aplikasi, <i>User</i> juga dapat melakukan edit akun, dan admin dapat menghapus <i>User</i>
2	<i>Notification Service</i>	<i>Notification Service</i> berfungsi untuk mengirimkan notifikasi melalui email, saat melakukan autentikasi dan pemesanan atau pembayaran

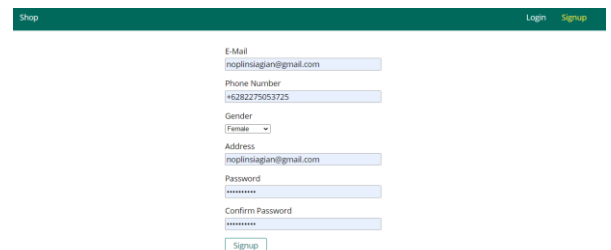
No	Nama Service	Keterangan
3	<i>Product Service</i>	Apoteker dapat mengelola produk dan <i>User</i> dapat melihat seluruh daftar produk yang tersedia.
4	<i>Cart Service</i>	<i>User</i> dapat mengelola keranjang pesanan produk seperti menambah produk atau menghapus produk dari keranjang.
5	<i>Order Service</i>	<i>User</i> dapat melakukan transaksi terhadap produk yang telah dipesan.

1.2.3. Hasil Implementasi Arsitektur Monolitik pada Aplikasi Apotek

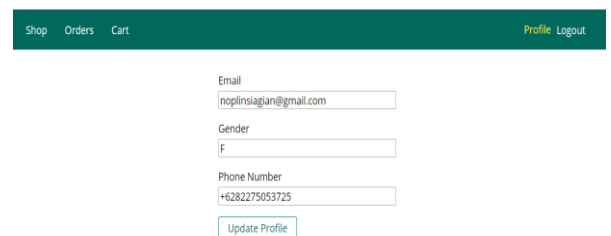
Pada bagian ini akan dijelaskan hasil dari pengimplementasian aplikasi apotek menggunakan arsitektur monolitik yang digunakan sebagai media perbandingan dengan arsitektur *microservices*. Hasil dari implementasi Aplikasi Apotek menggunakan arsitektur monolitik dapat dilihat pada gambar berikut:



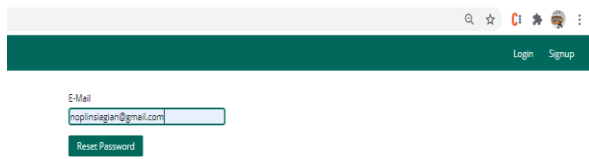
Gambar 3. Implementasi Login menggunakan Arsitektur Monolitik



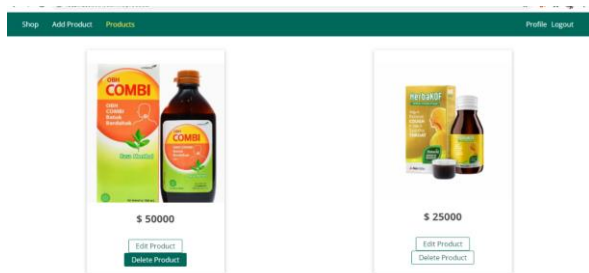
Gambar 4. Implementasi Register menggunakan Arsitektur Monolitik



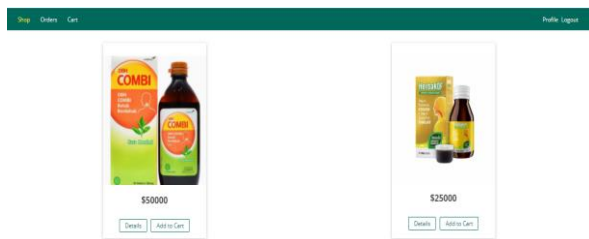
Gambar 5. Implementasi Form Edit Profil menggunakan Arsitektur Monolitik



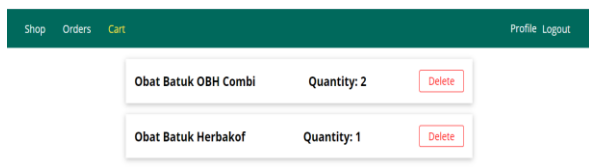
Gambar 6. Implementasi Notifikasi menggunakan Arsitektur Monolitik



Gambar 7. Implementasi Mengelola Daftar Produk



Gambar 8. . Implementasi Melihat Daftar Produk



Gambar 9. Implementasi form pemesanan terhadap Produk menggunakan Arsitektur Monolitik

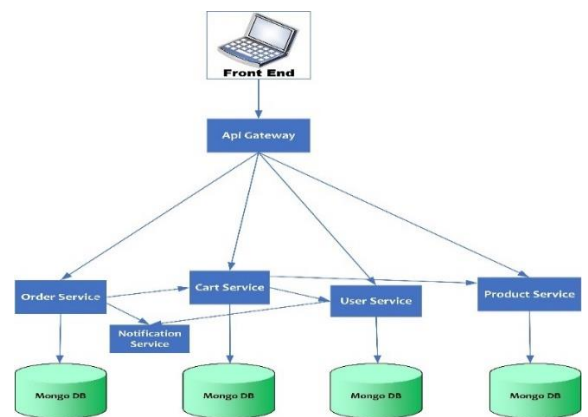
1.2.4. Analisis Implementasi Arsitektur *Microservices* pada Aplikasi Apotek

Aplikasi apotek menggunakan *arsitektur microservices* dibangun menggunakan bahasa pemrograman Node.js dan Go serta menggunakan jenis *database* MongoDB dikarenakan pada *database* MongoDB memiliki desain yang fleksibel yang dapat diubah dan memiliki performa yang lebih cepat saat melakukan penyimpanan data tidak perlu melakukan pengecekan struktur data, dan mudah dibaca. Terdapat 5 buah *service* yang membangun aplikasi ini, yaitu:

- a) *User Service*, Untuk menangani pengelolaan data *User* dan memberikan data yang dibutuhkan *service*.
- b) *Notification Service*, *Service* yang mengirimkan pemberitahuan seperti status *order* atau pembayaran.

- c) *Product Service*, menangani pengolahan terhadap informasi produk yang ada pada aplikasi.
- d) *Cart Service*, Untuk menangani pengolahan informasi produk yang dipilih oleh *user* untuk melakukan pemesanan.
- e) *Order Service*, menangani pengolahan informasi pemesanan produk yang ada pada aplikasi dan menangani pengolahan informasi pembayaran pesanan pada aplikasi.

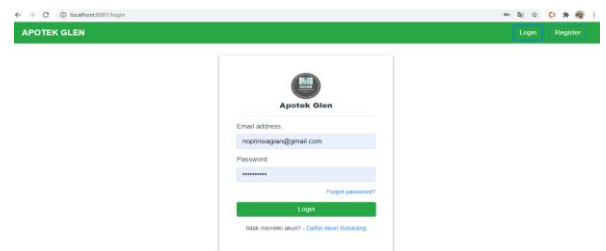
Berdasarkan gambar di bawah ini, *Client* hanya dapat mengakses aplikasi dengan melakukan *request* ke sebuah *API Gateway* yang akan melakukan *routing* terhadap setiap *request*. Setiap *service* akan berhubungan dengan *service* lainnya. Arsitektur aplikasi beserta *database* dari aplikasi apotek menggunakan *microservices* dapat dilihat pada Gambar 10.



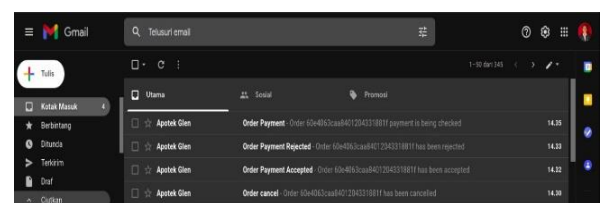
Gambar 10. Arsitektur Aplikasi Apotek dan *database*

1.3. Hasil Implementasi Arsitektur *Microservices* pada Aplikasi Apotek

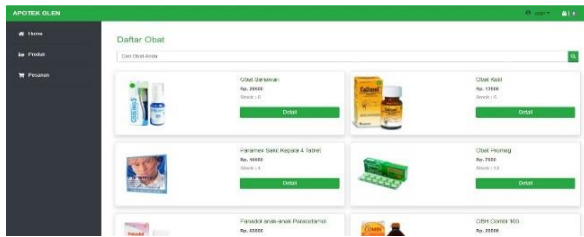
Adapun hasil dari implementasi aplikasi apotek menggunakan arsitektur *microservices* dapat dilihat pada gambar berikut:



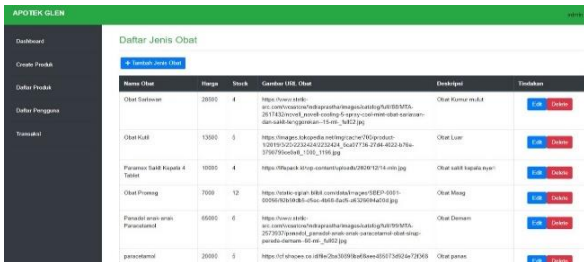
Gambar 11. Implementasi Login pada *User Service*



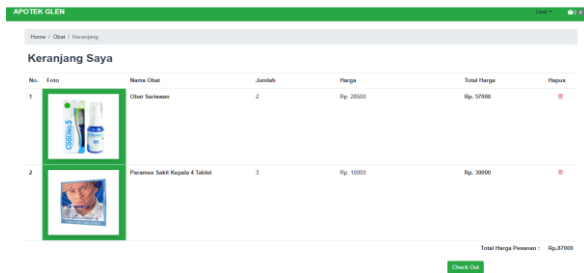
Gambar 12. Implementasi *Notification Service*



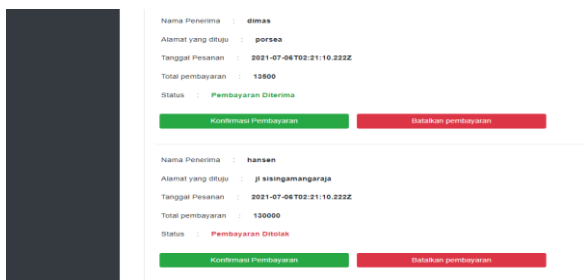
Gambar 13. Implementasi Menampilkan Produk pada Product Service



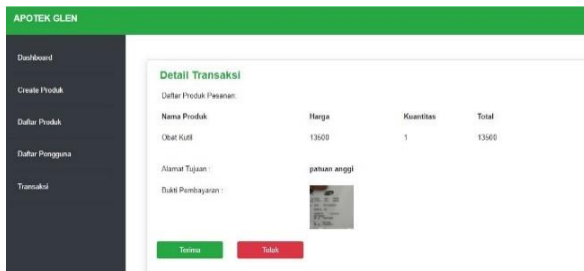
Gambar 14. Implementasi Mengedit dan Menghapus dari sisi Admin pada Product Service



Gambar 15. Implementasi Keranjang pada Cart Service



Gambar 16. Implementasi Status Pembayaran pada Order Service



Gambar 17. Implementasi Detail dan Konfirmasi Transaksi pada Order Service

2. Pengujian

Pengembang melakukan pengujian secara manual terhadap aplikasi apotek menggunakan arsitektur monolitik sedangkan pada aplikasi apotek menggunakan

arsitektur *microservices*, terlebih dahulu pengembang melakukan pengujian terhadap setiap *service* menggunakan *Postman* lalu setiap *service* yang telah diuji dihubungkan dengan *frontend* dan Api Gateway dan dilakukan pengujian kembali.

3. Hasil Perbandingan Arsitektur Monolitik dan Arsitektur *Microservice* pada Aplikasi Apotek

Perbandingan secara umum antara arsitektur monolitik dan arsitektur *microservices* berdasarkan beberapa indikator dapat dilihat pada Tabel 3.

Tabel 3. Perbandingan Arsitektur Monolitik dengan Arsitektur *Microservices*

No	Indikator	Arsitektur Monolitik	Arsitektur <i>Microservices</i>
1	Struktur Aplikasi	Terdapat dalam satu komponen <i>project</i>	Terdapat dalam beberapa <i>service</i> , Api Gateway dan <i>Frontend</i>
2	Bahasa Pemrograman	Menggunakan satu bahasa pemrograman	Dapat menggunakan lebih dari satu bahasa pemrograman
3	Database	Hanya menggunakan satu database	Dapat menggunakan lebih dari satu database
4	Deployment	<i>deploy</i> dilakukan sekaligus dengan waktu yang bersamaan	<i>deploy</i> dilakukan terpisah untuk masing-masing <i>service</i>

Sedangkan, perbandingan yang diperoleh oleh penulis setelah menerapkan arsitektur monolitik dan *microservice* pada aplikasi apotek Glen berbasis web dapat dilihat pada Tabel 4.

Tabel 4. Perbandingan Aplikasi Apotek menggunakan Arsitektur Monolitik dan Arsitektur *Microservices* (Studi Kasus: Apotek Glen, Kab.Toba)

No	Indikator	Aplikasi Apotek menggunakan Arsitektur Monolitik	Aplikasi Apotek menggunakan Arsitektur <i>Microservices</i>
1	Struktur Aplikasi	Terdapat dalam satu komponen <i>project</i>	Terdapat dalam 5 <i>Service</i> , <i>Front End</i> dan <i>Api Gateway</i>
2	Bahasa Pemrograman	Menggunakan satu bahasa pemrograman yaitu Node.js	Pada <i>Service</i> menggunakan 2 bahasa pemrograman yaitu menggunakan Node.js dan Golang sedangkan untuk <i>Front end</i> menggunakan <i>Vue.js</i>
3	Database	Menggunakan satu database	Menggunakan 4 database
4	Deployment	<i>deploy</i> dilakukan sekaligus dengan waktu yang	<i>deploy</i> dilakukan terpisah untuk masing-masing <i>service</i>

No	Indikator	Aplikasi Apotek menggunakan Arsitektur Monolitik	Aplikasi Apotek menggunakan Arsitektur <i>Microservices</i>
		bersamaan	
5	<i>Build Time</i>	Selama 1,3467s	<i>User Service</i> = 1,0887s, <i>Product Service</i> = 1,0135s, <i>Cart Service</i> = 0.9722s, <i>Order Service</i> = 0.9980s, <i>Front End</i> = 11526ms

IV. KESIMPULAN

Berdasarkan penerapan aplikasi apotek berbasis *web* menggunakan arsitektur *microservice*, maka dapat disimpulkan bahwa:

1. Pengembang telah berhasil membangun aplikasi apotek berbasis *web* menggunakan arsitektur *microservices* dan *microservices* ini cocok diimplementasikan pada aplikasi apotek.
2. Berdasarkan perbandingan yang telah dibahas bahwa aplikasi menggunakan arsitektur *microservices* merupakan keputusan yang tepat apabila ingin membangun aplikasi dalam skala yang besar dikarenakan aplikasi dibangun terdiri dari beberapa *service* yang masing - masing memiliki satu tanggung jawab. Berbeda dengan arsitektur monolitik yang semua layanan berada dalam satu komponen sehingga satu program akan menangani semua layanan dalam aplikasi.
3. Menggunakan arsitektur *microservices* pengembang dapat melakukan pembaharuan atau menerapkan teknologi baru tanpa harus memperbaharui *service* yang lain.

UCAPAN TERIMAKASIH

Terima kasih kepada LPPM Institut Teknologi Del yang memberikan dana penyelenggaraan dalam melaksanakan penelitian ini serta individu yang telah membantu penulis dalam penelitian: pembimbing, pendukung keuangan atau pendukung lainnya.

REFERENCES

- [1] P. Novianhiny, H. Nasution, and E. Esyudha Pratama, "Rancang Bangun Aplikasi Penjualan dan Pembelian Berbasis Web Pada Apotek Neofarma Sanggau," *J. Sist. dan Teknol. Inf.*, vol. 6, no. 3, p. 129, 2018, doi: 10.26418/justin.v6i3.25770.
- [2] T. Murwanto, W. A. Syaifei, and R. R. Isnanto, "Rancang Bangun Sistem Informasi WEB (Studi Kasus di Apotik Mutiara Banyumanik Semarang)," *Undip*, vol. 2, NO 3, 2008.
- [3] M. D. Rafiqi, E. Subyantoro, and D. K. W., "Implementasi Arsitektur Microservice Pada Aplikasi Online Travel Tourinc," *Karya Ilm. Mhs. Manaj. Inform.*, vol. 1, no. 1, pp. 1–10, 2019.
- [4] G. Munawar and A. Hodijah, "Analisis Model Arsitektur Microservice Pada Sistem Informasi DPLK," *Sink. J. dan Penelit. Tek. Inform.*, vol. 3, no. 1, pp. 232–239, 2018.
- [5] H. Suryotrisongko, "Arsitektur Microservice untuk Resiliensi Sistem Informasi," *Sisfo*, vol. 06, no. 02, pp. 231–246, 2017, doi: 10.24089/j.sisfo.2017.01.006.
- [6] R. Mufrizal and D. Indarti, "Refactoring Arsitektur Microservice Pada Aplikasi Absensi PT. Graha Usaha Teknik," *J. Nas. Teknol. dan Sist. Inf.*, vol. 5, no. 1, pp. 57–68, 2019, doi: 10.25077/teknosi.v5i1.2019.57-68.
- [7] P. L. Lokapitasari Belluano, P. Purnawansyah, B. L. E. Panggabean, and H. Herman, "Sistem Informasi Program Kreativitas Mahasiswa berbasis Web Service dan Microservice," *Ilk. J. Ilm.*, vol. 12, no. 1, pp. 8–16, 2020, doi: 10.33096/ilkom.v12i1.492.8-16.
- [8] F. Tapia, M. ángel Mora, W. Fuertes, H. Aules, E. Flores, and T. Toulkeridis, "From monolithic systems to microservices: A comparative study of performance," *Appl. Sci.*, vol. 10, no. 17, 2020, doi: 10.3390/app10175797.
- [9] M. Rezaldy, I. Asror, and I. L. Sardi, "Desain dan Analisis Arsitektur Microservices Pada Sistem Informasi Akademik Perguruan Tinggi Dengan Pendekatan Architecture Tradeoff Analysis Method (ATAM) (Studi Kasus : iGracias Universitas Telkom)," vol. 4, no. 2, 2017.
- [10] J. T. Zhao, S. Y. Jing, and L. Z. Jiang, "Management of API Gateway Based on Micro-service Architecture," *J. Phys. Conf. Ser.*, vol. 1087, no. 3, 2018, doi: 10.1088/1742-6596/1087/3/032032.
- [11] Y. Chandra, T. Putra, T. Adi, P. Sidi, and J. E. Samodra, "Implementasi Arsitektur Microservice pada Aplikasi Web Pengajaran Agama Islam Home Pesantren," vol. 1, no. November, pp. 88–97, 2020.
- [12] A. Firdaus, S. Widodo, A. Sutrisman, S. G. F. Nasution, and R. Mardiana, "Rancang Bangun Sistem Informasi Perpustakaan Menggunakan Web Service Pada Jurusan Teknik Komputer Polstri," *J. Inform.*, vol. 5, no. 2407–1730, p. 83, 2019.