CLOUD
Jurnal Networking System and Security System
E-ISSN 3090-6601

Penerapan Arsitektur *Microservices* Pada Web *E-Commerce* Menggunakan Metode GRPC

Muhammad Arief[1], Ari Usman Chaniago[2]

[1,2] Universitas Harapan Medan, Medan, Indonesia
[1]arief23@pm.me, [2]ariusman09@gmail.com
[*)]arief23@pm.me

Abstrak − *E-commerce* merupakan salah satu *platform online* yang dapat diakses oleh kalangan masyarakat melalui komputer maupun *handphone.* Saat ini masih banyak *platform e-commerce* di Indonesia yang menerapkan arsitektur monolith. Arsitektur monolith memiliki keterbatasan ketika tim pengembang semakin bertambah dan berganti, arsitektur monolith juga hanya memiliki satu bahasa pemrograman yang dipakai, arsitektur monolith akan semakin sulit di-*maintenance* ketika aplikasi memiliki banyak fitur. Pada penelitian ini bertujuan untuk memisahkan fitur-fitur menjadi layanan mikro yang saling berkomunikasi menggunakan metode *GRPC* (*Google Remote Procedure Call)*. *GRPC* (*Google Remote Procedure Call)* merupakan salah satu bagian dari *remote procedure call*, *GRPC* memanfaatkan HTTP/2 serta *Protocol Buffer* sebagai antarmuka transportasi untuk melakukan pertukaran data. Berdasarkan hasil pada penelitian ini menunjukkan bahwa aplikasi yang berjalan menggunakan arsitektur *microservices* dapat menggunakan dua bahasa yaitu javascript dan golang, adapun hasil lain yang didapat yaitu ketika salah satu *service* yang dibangun sedang bermasalah, fitur-fitur pada website masih berjalan secara baik dikarenakan setiap fitur yang dibangun telah berdiri sendiri.

Kata Kunci: Arsitektur Microservices, E-commerce, GRPC

Abstract − E-commerce is one of the online platforms that can be accessed by the community through computers and mobile phones. Currently, there are still many e-commerce platforms in Indonesia that implement monolith architecture. Monolith architecture has limitations when the development team grows and changes, monolith architecture also only has one programming language used, monolith architecture will be increasingly difficult to maintain when the application has many features. This research aims to separate features into microservices that communicate with each other using the GRPC (Google Remote Procedure Call) method. GRPC (Google Remote Procedure Call) is one part of the remote procedure call, GRPC utilises HTTP/2 and Protocol Buffer as a transport interface to exchange data. Based on the results of this study, it shows that applications that run using microservices architecture can use two languages, namely javascript and golang, as well as other results obtained, namely when one of the services built is having problems, the features on the website still run well because each feature built has stood alone.

Keywords: Microservices Architecture, E-commerce, GRPC

1. PENDAHULUAN

E-commerce merupakan istilah yang digunakan untuk menggambarkan proses pembelian, penjualan, pertukaran jasa melalui internet, salah satu implementasi dari *e-commerce* yaitu dapat berbentuk sebuah website. *E-commerce* adalah platform *online* yang dapat diakses oleh individu melalui komputer. Ini digunakan oleh pelaku bisnis untuk menjalankan kegiatan bisnis mereka dan juga digunakan oleh konsumen untuk mencari informasi dengan bantuan komputer. Di dalamnya, terdapat proses yang dimulai dengan menyediakan layanan informasi kepada konsumen untuk membantu mereka dalam pengambilan keputusan [1].

Pada era saat ini, sudah banyak *e-commerce* yang dibangun menggunakan arsitektur monolitik. Jika sebuah sistem diterapkan arsitektur monolitik, sistem yang dibangun akan sulit dipelihara, sehingga untuk membangun sebuah sistem akan membutuhkan banyak waktu dan sulit untuk dikembangkan jika tim pengembang bertambah dan berganti. Oleh karena itu banyak orang berpikir untuk menggunakan sistem yang masih dibangun dengan arsitektur monolitik.

Microservices memungkinkan adanya skalabilitas individual pada tiap-tiap *service* yang di gunakan, jika pada suatu *service* mengalami kelonjakan *traffic* maka hanya *service* tersebut yang perlu ditingkatkan kapasitasnya, tanpa mempengaruhi layanan yang lain. Pada arsitektur monolitik, hal ini tidak dapat dilakukan. *Microservices* juga dapat menggunakan fleksibilitas bahasa pemrograman yang cukup banyak dikarenakan tiap-tiap layanan berdiri sendiri.

Keunggulan skalabilitas individual yang ditawarkan *microservices* merupakan faktor krusial dalam menghadapi fluktuasi beban kerja aplikasi modern. Bayangkan sebuah platform *e-commerce* di mana layanan pembayaran mengalami lonjakan trafik saat hari belanja *online* nasional (Harbolnas). Dengan arsitektur *microservices*, tim operasional dapat fokus meningkatkan kapasitas layanan pembayaran saja, tanpa perlu

menyentuh layanan lain seperti katalog produk atau manajemen akun. Hal ini menghasilkan penggunaan sumber daya yang lebih efisien dan mencegah gangguan pada fungsi-fungsi aplikasi yang tidak terpengaruh oleh lonjakan trafik.

Selain itu, fleksibilitas dalam pemilihan bahasa pemrograman memungkinkan tim pengembang untuk memilih teknologi yang paling tepat guna untuk setiap layanan. Misalnya, layanan yang membutuhkan pemrosesan data yang kompleks dapat dibangun dengan Python dan library data science-nya, sementara layanan yang menuntut performa tinggi dapat diimplementasikan dengan Go atau C++. Kebebasan ini memungkinkan optimasi performa dan efisiensi pengembangan pada setiap layanan.

Sebagai perbandingan, pada arsitektur monolitik, seluruh aplikasi terikat pada satu stack teknologi. Jika ada kebutuhan untuk mengganti atau meningkatkan satu komponen, seluruh aplikasi harus diubah dan di-deploy ulang. Hal ini menimbulkan risiko, memakan waktu, dan mengurangi agilitas pengembangan.

Dalam dunia yang dinamis dan kompetitif saat ini, kemampuan untuk beradaptasi dengan cepat terhadap perubahan kebutuhan bisnis dan teknologi menjadi sangat penting. Arsitektur microservices, dengan skalabilitas individual dan fleksibilitas bahasa pemrogramannya, memberikan fondasi yang kokoh bagi organisasi untuk mencapai agilitas dan inovasi yang berkelanjutan.

Kelebihan lain dari microservices yang patut disorot adalah kemampuannya dalam mempercepat proses development dan deployment. Tim pengembang dapat bekerja secara paralel pada layanan-layanan yang berbeda, memungkinkan siklus pengembangan yang lebih cepat dan fleksibel. Deployment fitur baru atau perbaikan bug dapat dilakukan pada layanan tertentu tanpa mempengaruhi layanan lain, mengurangi risiko dan downtime. Hal ini berbeda dengan arsitektur monolitik di mana setiap perubahan kecil pun mengharuskan deployment ulang seluruh aplikasi.

Selain itu, arsitektur microservices juga mendukung inovasi dan eksperimen. Tim pengembang dapat dengan mudah mencoba teknologi baru pada satu layanan tanpa mengkhawatirkan dampaknya pada keseluruhan sistem. Hal ini menciptakan lingkungan yang kondusif untuk eksplorasi dan pengembangan solusi yang lebih baik.

Dengan segala keunggulannya, microservices menjadi pilihan arsitektur yang semakin populer untuk aplikasi modern yang menuntut skalabilitas, fleksibilitas, dan kecepatan inovasi. Penelitian-penelitian yang Anda sebutkan sebelumnya menunjukkan bahwa microservices telah berhasil diimplementasikan dalam berbagai konteks, dan tren ini diharapkan akan terus berlanjut seiring dengan perkembangan teknologi dan tuntutan bisnis.

Microservices adalah pendekatan arsitektur perangkat lunak yang memecah aplikasi menjadi layanan-layanan kecil yang saling terhubung. Setiap layanan memiliki tanggung jawab yang spesifik dan dapat dikembangkan, diuji, dan di-*deploy* secara independen. Layanan-layanan ini berkomunikasi satu sama lain menggunakan *API* (*Application Programming Interface*) atau *GRPC* (Google *Remote Procedure Call*). Di samping itu, memecah sistem menjadi komponen-komponen kecil dapat mempermudah perawatan, meningkatkan kecepatan pengujian layanan, dan memungkinkan penerapan teknologi yang berbeda sesuai dengan kebutuhan masing-masing layanan [2].

Pada penelitian terdahulu yang ditulis oleh [3] telah melakukan penelitian berjudul "Perancangan Sistem Informasi Laporan Kegiatan Penanaman Modal Dengan Menggunakan Arsitektur Microservices Pada Kementerian Investasi/Badan Koordinasi Penanaman Modal" arsitektur *microservices* digunakan karena beberapa kelebihannya, seperti aplikasi mudah dikelola karena modul-modulnya dipecah ke volume yang relatif kecil, arsitektur *microservices* berdiri sebagai modul-modul mikro yang bersifat mandiri dan digunakan secara independen, serta tiap modul dapat menggunakan bahasa pemrograman yang berbeda berdasarkan kebutuhan bisnis.

Berdasarkan latar belakang yang telah dijabarkan, maka peneliti membuat penelitian dengan judul "Penerapan Arsitektur *Microservices* pada web *e-commerce* menggunakan metode *GRPC* (*Google Remote Procedure Call*)".

2. METODE PENELITIAN

2.1 *Microservice*

Arsitektur *microservice* adalah sebuah sistem yang terencana untuk membagikan layanan dalam skala yang lebih kecil dan lebih mudah disesuaikan. Perangkat lunak akan dirancang untuk menjalankan fungsi-fungsi secara independen. Dengan kata lain, setiap tantangan teknis dapat di atasi dengan menggunakan metode dan teknik yang beragam, lalu dihubungkan melalui antarmuka pemrograman aplikasi (*API*) [4].

Setiap *microservice* berfokus pada pelaksanaan dan penyelesaian satu tugas spesifik dengan baik. *Microservice* dapat dikembangkan dengan berbagai bahasa pemrograman yang berbeda. Mereka dapat berkomunikasi satu sama lain melalui Antarmuka Pemrograman Aplikasi (*API*), seperti *REST*. Sebuah *microservice* tidak perlu memiliki pengetahuan tentang arsitektur *microservice* lainnya [5].

Microservice memungkinkan pengolaan *service – service* yang ada pada perangkat lunak secara terpisah, dengan begitu jika dilakukan pengembangan *service* maka *service* lain tidak akan terganggu, *microservice* merupakan pengembangan lanjutan dari *Service-oriented Architecture* karena *microservice* terdari dari komponen servis-servis kecil yang terpisah dan fokus pada tugas-tugasnya, *autonomous* untuk tujuan masing-masing namun terkoneksi satu sama lain secara beraturan [6].

2.2 *GRPC* (Google *Remote Procedure Call*)

GRPC adalah bagian dari *remote procedure call* yang bersifat *open source* dan awalnya dikembangkan oleh Google pada tahun 2015. Ini memanfaatkan HTTP/2 serta *Protocol Buffer* sebagai antarmuka transportasi untuk pertukaran data. *GRPC* menyajikan fitur-fitur seperti otentikasi, streaming dua arah, dan *flow control*. Dalam konteks komunikasi data, *GRPC* dapat digunakan dalam berbagai bahasa pemgrograman sebagai perangkat klien dan peladen. *Protocol Buffers* dengan *GRPC* mampu memberikan kecepatan dan performa yang unggul karena metode serialisasi binernya yang sangat cepat [7].

2.3 *API Gateway*

API Gateway merupakan sebuah layanan yang berperan sebagai pintu masuk tunggal dalam sistem, berada di antara klien (*client*) dan layanan *backend* (*server*). Prosesnya dimulai ketika klien mengirimkan permintaan kepada *API* Gateway, lalu *API Gateway* akan menghubungi endpoint yang sesuai untuk mengambil informasi dari klien. Hasilnya, respons dikirimkan kembali kepada klien [8].

API Gateway merupakan layanan yang berfungsi sebagai gerbang utama yang dijangkau oleh klien, sehingga klien tidak harus terkoneksi langsung ke setiap layanan yang ada karena dalam arsitektur *microservices* setiap *service* bersifat *isolated* atau tidak dapat di akses langsung oleh klien melainkan harus melalui *API Gateway*. Dengan hanya mengakses *API Gateway*, klien dapat terhubung ke layanan yang dibutuhkannya [9].

2.4 Docker

Docker adalah sebuah proyek sumber terbuka yang menyediakan platform terbuka bagi pengembang dan administrator sistem untuk membuat, mengemas, dan menjalankan aplikasi di dalam wadah (*container*) di mana pun. Docker menggunakan arsitektur client-server di mana klien dan Docker berinteraksi dengan *daemon* Docker. *Daemon* Docker ini bertanggung jawab untuk melakukan berbagai tindakan seperti membangun, menjalankan, dan mendistribusikan wadah Docker. Klien Docker dan *daemon* dapat beroperasi pada sistem yang sama, dan keduanya berkomunikasi melalui *REST API*, soket *UNIX*, atau antarmuka jaringan [10].

2.5 Web *Service*

Web Service menyediakan interoperabilitas antar berbagai aplikasi perangkat lunak yang berjalan pada *platform* yang berbeda dan tidak terbatas pada perangkat lunak, struktur sistem, atau bahasa pemrograman tertentu. Ini berfungsi untuk menyediakan metode atau layanan yang memungkinkan pertukaran data melalui jaringan [11].

2.6 *Application Programming Interface (API)*

API adalah antarmuka yang memungkinkan aplikasi berinteraksi dengan aplikasi lain. Ini juga bisa didefinisikan sebagai kumpulan fungsi, perintah, dan protokol yang digunakan oleh pengembang saat membuat perangkat lunak untuk sistem operasi tertentu. *API* memungkinkan programmer menggunakan fungsi standar untuk berkomunikasi dengan sistem operasi. Selain itu, terdapat juga konsep seperti *REST API*, yang merupakan aturan atau gaya untuk membuat layanan web. Dengan *API*, pengembang dapat dengan mudah mengintegrasikan fungsionalitas dari berbagai sumber dan sistem, memungkinkan aplikasi berinteraksi dan berbagi data [12].

2.7 Golang

Golang adalah bahasa pemrograman yang diketik secara statis atau *static typing* dan dikompilasi, golang merupakan bahasa pemrograman *open source* yang memudahkan untuk membangun perangkat lunak secara sederhana dan efisien. Golang mempunyai standar paket yang sudah tersedia ketika menginstall bahasa pemrograman go yaitu paket *archive, context, crypto, bytes, compress, encoding* dan lainnya [13].

Golang mempunyai kemiripan dengan bahasa pemrograman C di banyak aspek, golang mengadopsi banyak ide bagus dari banyak bahasa pemgrograman, golang menghindari fitur yang mengarah ke kompleksitas dan tidak dapat diandalkan. Golang mempunyai kemampuan untuk menggunakan *multi-core* CPU untuk memproses *concurrency* menggunakan goroutines dan *channels*, ini memungkinkan golang untuk menjalankan beberapa proses secara bersamaan. Golang tidak mempunyai *class* struktur seperti bahasa pemrograman *Object Oriented Programming* (*OOP*) [14].

2.8 VueJS

Vue.js adalah kerangka kerja JavaScript progresif *open source* yang digunakan untuk membangun antarmuka web dinamis dan interaktif. Kerangka kerja ini berfokus pada lapisan tampilan pengembangan web dan dapat dengan mudah diintegrasikan ke dalam proyek pengembangan web besar dan perusahaan. Vue.js adalah pilihan yang tepat untuk *developer* yang ingin membuat proyek besar dan dapat diskalakan dengan mudah, karena struktur kode dan lingkungan pengembangannya yang ramah untuk *developer* [15].

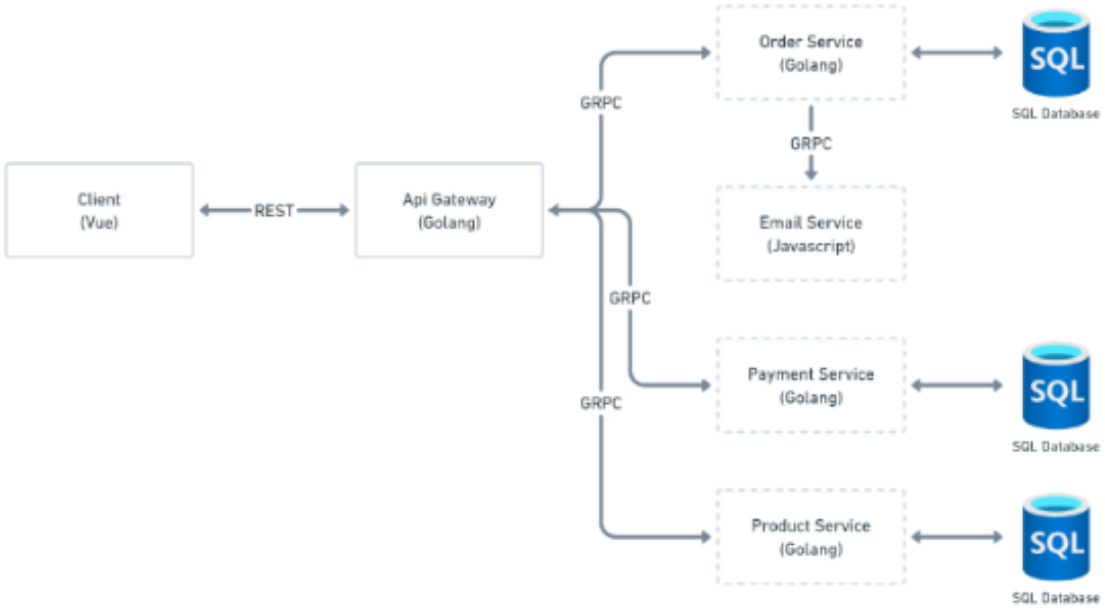
2.9 *Database*

Database adalah kumpulan data yang terorganisir dengan baik, yang saling berhubungan sehingga mudah disimpan, dimanipulasi, dan diakses oleh pengguna. Istilah "berhubungan" dalam konteks ini berarti bahwa data menjelaskan *domain* tertentu, sehingga pengguna dapat dengan mudah menemukan jawaban atas pertanyaan yang diajukan ke dalam basis data tersebut. Sementara itu, sistem basis data merupakan koleksi data yang terorganisasi sedemikian rupa sehingga memudahkan penyimpanan, manipulasi (termasuk pembaruan, pencarian, perhitungan tertentu, dan penghapusan), serta pengolahan data [16].

3. HASIL DAN PEMBAHASAN

3.1 *Microservice Architecture*

Berikut adalah pemodelan sistem pada sistem yang dibuat oleh penulis dapat dilihat pada gambar 1 berikut:



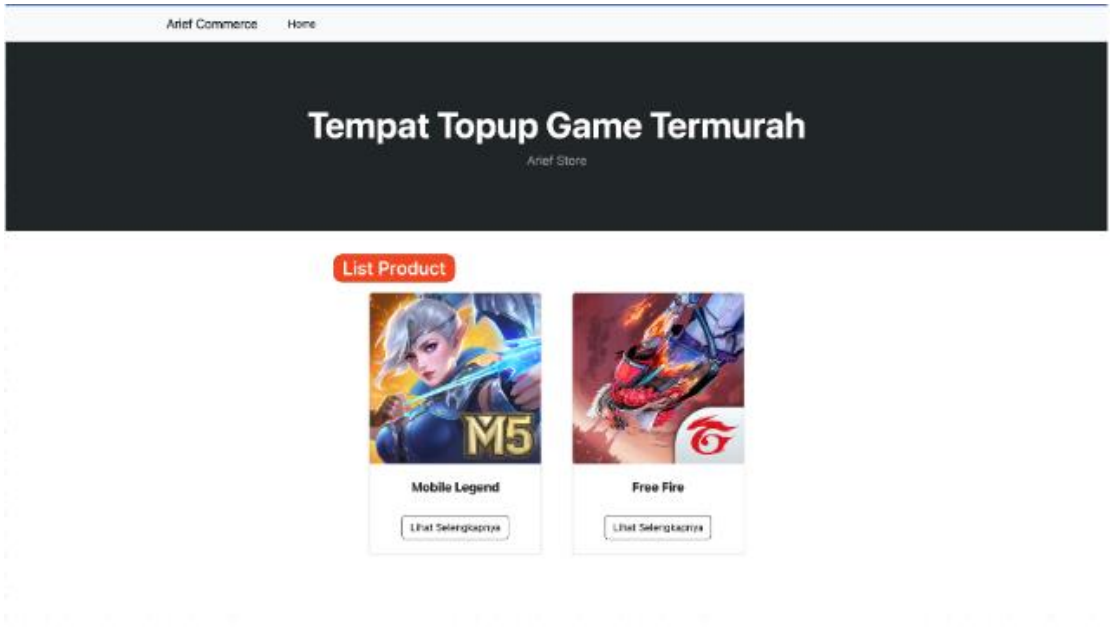
Gambar 1. Sistem *E-Commerce Microservice*

Dapat dilihat bahwa *client* menggunakan *REST API* untuk melakukan komunikasi dengan *API Gateway* dimana *API Gateway* adalah gerbang untuk berkomunikasi antar *service* yang berfungsi sebagai gerbang utama masuk *microservices, API Gateway* juga berfungsi sebagai manajemen *API.* Setelah berhasil melewati proses pada *API Gateway* ada proses *request* yang akan diteruskan ke *service* tujuan seperti, *order service, payment service*, dan *product service*, dan tiap *service* akan menggunakan *GRPC* sebagai protocol komunikasi.

3.2 Implementasi Antarmuka

Antarmuka aplikasi adalah proses implementasi tampilan dari sebuah desain yang telah dibuat sebelumnya pada aplikasi *e-commerce* yang menggunakan arsitektur *microservices* untuk melihat proses komunikasi melalui tampilan visual.

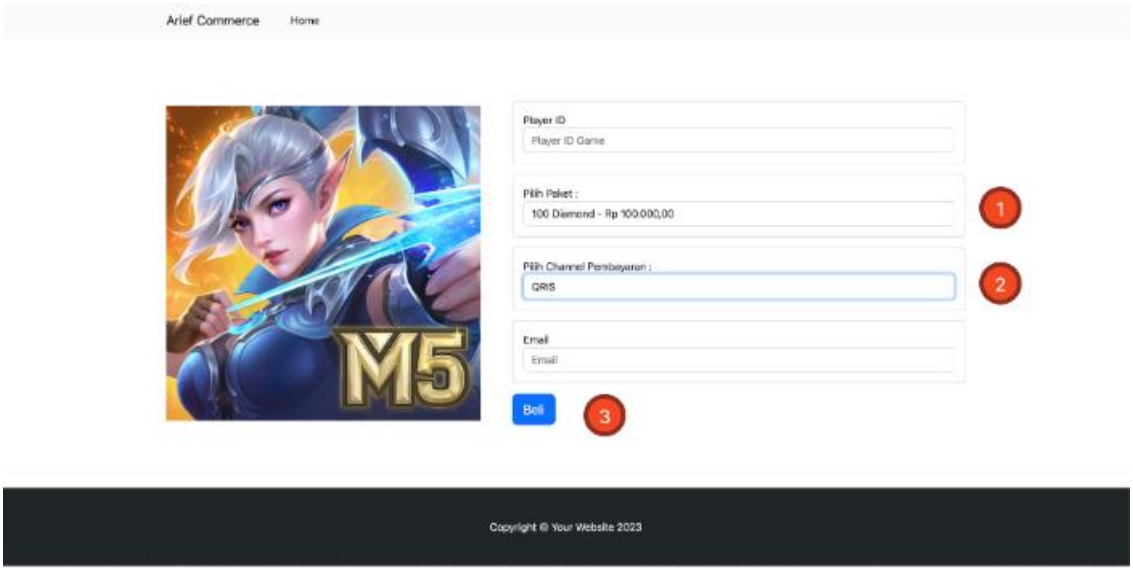
a. Halaman *Home*



Gambar 2. Tampilan Halaman *Home*

Halaman *Home* berisi tentang *website* yang menampilkan *list product* apa saja yang dimiliki. Berikut tampilan halaman *home*. Pada Gambar 2 memperlihatkan 2 produk yang berada pada halaman *home*, produk tersebut didapat dari mengirim *request* kepada *API Gateway*. *Client* akan mengirim *request* kepada *API Gateway* dengan url "/api/v1/products" yang kemudian *API Gateway* akan memproses *request* tersebut dan akan memanggil *product service* dengan *GRPC*. Di bawah ini adalah alur proses pengambilan data dari sisi *client*.

b. Halaman *Order*

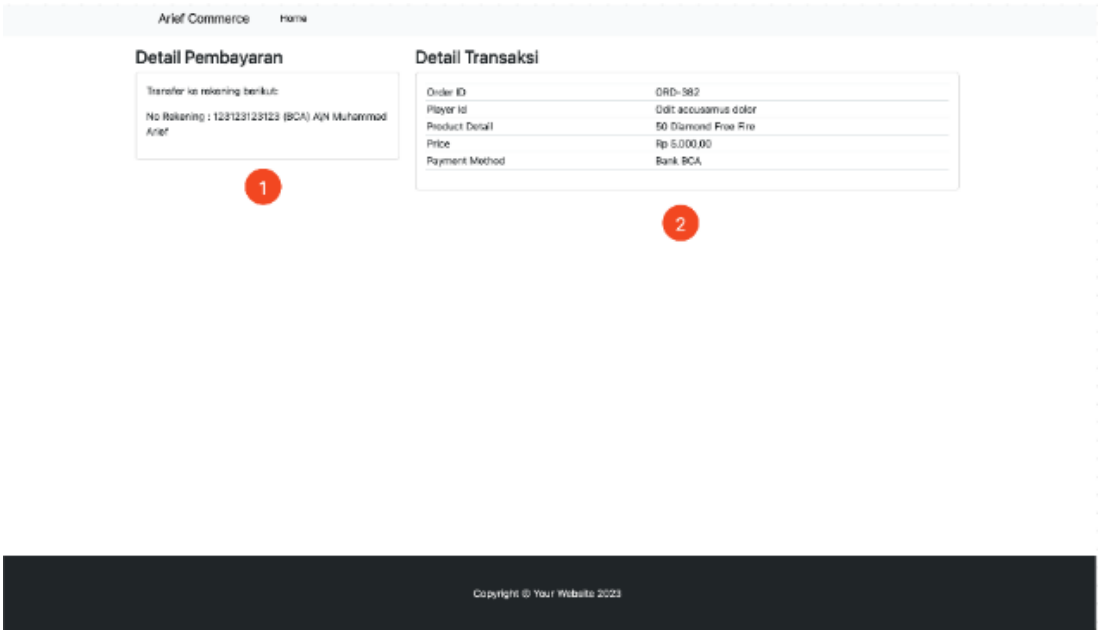


Gambar 3. Tampilan Halaman *Order*

Keterangan

1. *Form* pilih paket merupakan *list* data *product detail* yang diambil dari *API Gateway* dengan *URL* "/api/v1/products/:id", id adalah *parameter* dari id produk yang ada di *database*. *API Gateway* akan mengirim *request* ke *product service* dengan menggunakan *GRPC* yang kemudian akan dikembalikan oleh *API Gateway* ke *client.*
2. *Form* pilih *channel* pembayaran merupakan *list* data *payment channel* yang didapat dari *API Gateway* dengan *URL* "/api/v1/payments". *API Gateway* akan mengirim *request* ke *payment service* menggunakan *GRPC* yang kemudian akan dikembalikan oleh *API Gateway* ke *client.*
3. Tombol "Beli" merupakan tombol yang digunakan untuk membuat *order* pembeli. Pada *action* ini *client* akan mengirim *request* ke *API Gateway* dengan *URL "/api/v1/orders"*, *API Gateway* akan mengirim *request* ke *order service* untuk melakukan *create order*, setelah *create order* sukses *order service* akan mengirimkan *e-mail* kepada pembeli.

c. Halaman Pembayaran



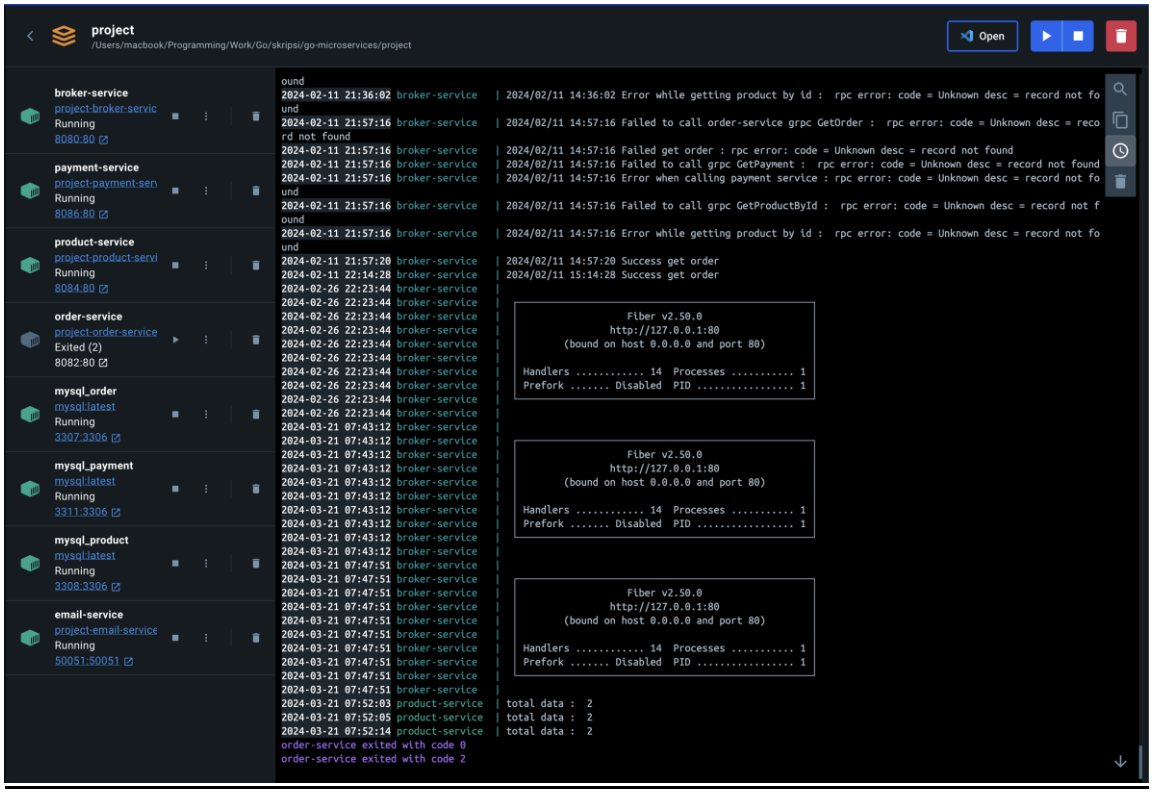
Gambar 4. Tampilan Halaman Pembayaran

Keterangan

1. Pada bagian "Detail Pembayaran" *client* akan mengirim *request* ke *API Gateway* dengan *URL* "/api/v1/payment/:id", kemudian *API Gateway* akan meneruskan *request* ke *payment service* lalu mengembalikan hasilnya ke *client.*
2. Pada bagian "Detail Transaksi" *client* melakukan beberapa *request* ke *API Gateway* yaitu dengan *URL "/api/v1/order/:id"* dan *"/api /v1/product/:id", API Gateway* akan meneruskan *request* ke *order service* dan *product service* lalu mengembalikannya ke *client.*

3.3 Hasil dan Pengujian

Pengujian menghentikan *service* bertujuan untuk melihat perilaku *website client* yang telah dibangun, apakah dengan menghentikan *order service*, *website client* dapat berjalan dengan yang diharapkan atau tidak.



Gambar 5. *Order service* dihentikan

Pengujian arsitektur microservices pada website e-commerce menggunakan metode GRPC telah menunjukkan hasil yang positif dan memuaskan. Sistem berjalan sesuai dengan ekspektasi ketika semua service aktif. Website client dapat menampilkan list produk, memproses pesanan, dan mengirimkan e-mail invoice dengan lancar.

Pengujian juga menunjukkan ketahanan sistem yang baik ketika salah satu service dihentikan. Meskipun terjadi gangguan pada fungsi yang berhubungan dengan service yang dihentikan, website tetap dapat diakses dan fungsi lainnya tetap berjalan normal.

Misalnya, ketika service "Product" dihentikan, website client memang tidak dapat menampilkan list produk dan memproses pesanan, namun pengunjung masih dapat mengakses website dan melihat informasi lainnya. Hal ini menunjukkan bahwa arsitektur microservices mampu meminimalisir dampak kegagalan pada satu service terhadap keseluruhan sistem.

Ketahanan ini merupakan salah satu keunggulan utama arsitektur microservices. Dengan memecah aplikasi menjadi layanan-layanan independen, kegagalan pada satu service tidak akan melumpuhkan seluruh sistem. Hal ini meningkatkan ketersediaan dan keandalan aplikasi, yang sangat penting untuk website e-commerce yang mengutamakan pengalaman pengguna yang positif.

Hasil pengujian ini memperkuat bahwa arsitektur microservices dengan metode GRPC merupakan pilihan yang tepat untuk website e-commerce. Arsitektur ini tidak hanya menawarkan performa yang baik, tetapi juga ketahanan dan skalabilitas yang tinggi, memungkinkan website untuk tumbuh dan berkembang sesuai dengan kebutuhan bisnis.

4. KESIMPULAN

Dalam penerapan arsitektur *microservices* pada sistem *e-commerce*, terdapat beberapa aspek penting yang perlu diperhatikan untuk memastikan sistem berjalan dengan efisien dan aman. Salah satu pendekatan yang diterapkan adalah penggunaan empat service yang masing-masing terhubung dengan *API Gateway* dan berkomunikasi menggunakan *GRPC*. Pendekatan ini memberikan keuntungan signifikan dalam hal pemisahan fungsionalitas dan skala sistem.

GRPC, sebagai protokol komunikasi berbasis *RPC* (*Remote Procedure Call*), memungkinkan *service-service* dalam sistem *e-commerce* untuk berkomunikasi secara efisien dan cepat. Dengan *GRPC*, data dapat ditukar

antara *service*, yang tidak hanya meningkatkan kinerja tetapi juga mempermudah integrasi dan pengelolaan data antar service. Ini penting dalam konteks *e-commerce*, di mana berbagai service seperti manajemen produk, pemrosesan transaksi, dan layanan pelanggan perlu saling terhubung dengan lancar.

Sementara itu, *API Gateway* berfungsi sebagai titik akses tunggal yang menghubungkan *client side* dengan service-service di backend. Dengan menggunakan *API Gateway*, layanan-layanan tidak perlu ter-*expose* secara langsung kepada *client*, yang meningkatkan keamanan sistem secara keseluruhan. *API Gateway* juga memungkinkan manajemen trafik yang lebih baik, *routing* yang lebih fleksibel, dan kontrol akses yang lebih terperinci, serta memberikan fitur tambahan seperti *caching* dan *load balancing*.

Secara keseluruhan, penerapan arsitektur *microservices* dengan *GRPC* dan *API Gateway* ini menciptakan sistem yang lebih modular, skalabel, dan aman. Ini memudahkan pengelolaan serta pengembangan berkelanjutan dari berbagai *service* dalam sistem *e-commerce*, sekaligus meningkatkan pengalaman pengguna akhir dengan memastikan komunikasi antar *service* yang efisien dan aman.

REFERENCES

[1] W. W. Windane and L. Lathifah, "E-COMMERCE TOKO FISAGO.CO BERBASIS ANDROID," *Jurnal Informatika dan Rekayasa Perangkat Lunak*, vol. 2, no. 3, pp. 285–303, Oct. 2021, doi: 10.33365/JATIKA.V2I3.1139.

[2] M. F. Radhiyan, "Analisis dan desain arsitektur microservices dengan graphql sebagai api gateway untuk sistem informasi akademik ais UIN Jakarta studi kasus : ais untuk mahasiswa," Jan. 2020, Accessed: Oct. 01, 2023. [Online]. Available: https://repository.uinjkt.ac.id/dspace/handle/123456789/56187

[3] A. Tjahyono, A. Muslim, and D. Anggraini, "The Perancangan Sistem Informasi Laporan Kegiatan Penanaman Modal Dengan Menggunakan Arsitektur Microservices Pada Kementerian Investasi/Badan Koordinasi Penanaman Modal," *Indonesia Journal on Computing (Indo-JC)*, vol. 7, no. 3, pp. 33–52, Dec. 2022, doi: 10.34818/INDOJC.2022.7.3.674.

[4] S. Atmojo, R. Utami, S. Dewi, and N. Widhiyanta, "Implementasi Sistem-informasi Desa Berbasis Arsitektur Microservices," *SMATIKA JURNAL : STIKI Informatika Jurnal*, vol. 12, no. 01, pp. 55–66, Jun. 2022, doi: 10.32664/SMATIKA.V12I01.658.

[5] D. Purwanto, W. Pramusinto, and G. Utama, "APLIKASI KURSUS ONLINE BERBASIS WEB SERVICE MENGGUNAKAN ARSITEKTUR MICROSERVICES," *Proceeding SENDI_U*, 2021, Accessed: Oct. 09, 2023. [Online]. Available: https://www.unisbank.ac.id/ojs/index.php/sendi_u/article/view/8629

[6] C. Seviro, B. Sakti, and I. Hermawan, "Implementasi Arsitektur Microservice pada Back End Sistem Informasi Atlantas berbasis Website," *Jurnal Teknologi Terpadu*, vol. 6, no. 2, pp. 96–104, Dec. 2020, doi: 10.54914/JTT.V6I2.281.

[7] J. F. Lombogia, A. Syahrina, and A. Musnansyah, "PERANCANGAN ARSITEKTUR PERANGKAT LUNAK MICROSERVICES PADA APLIKASI OPEN LIBRARY UNIVERSITAS TELKOM MENGGUNAKAN gRPC," *Telkatika: Jurnal Telekomunikasi Elektro Komputasi & Informatika*, vol. 1, no. 2, Jun. 2022, Accessed: Oct. 15, 2023. [Online]. Available: https://openlibrarypublications.telkomuniversity.ac.id/index.php/telkatika/article/view/17549

[8] F. X. Senduk, X. B. N. Najoan, and S. R. U. A. Sompie, "Pengembangan Arsitektur Microservices dengan RESTful API Gateway menggunakan Backend-for-frontend Pattern pada Portal Akademik Perguruan Tinggi: Development of Microservices Architecture with RESTful API Gateway using Backend-for-frontend Pattern in Higher Education Academic Portal," *Jurnal Teknik Informatika*, vol. 18, no. 1, pp. 315–324, Mar. 2023, Accessed: Oct. 24, 2023. [Online]. Available: https://ejournal.unsrat.ac.id/v3/index.php/informatika/article/view/50402

[9] U. Syarif and P. Pizaini, "PENERAPAN EVENT-DRIVEN MICROSERVICES PADA APLIKASI LAYANAN PENERIMAAN PESERTA DIDIK BARU," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 7, no. 3, pp. 745–756, Aug. 2022, doi: 10.29100/JIPI.V7I3.3067.

[10] S. Dwiyatno, E. Rachmat, A. P. Sari, and O. Gustiawan, "IMPLEMENTASI VIRTUALISASI SERVER BERBASIS DOCKER CONTAINER," *PROSISKO: Jurnal Pengembangan Riset dan Observasi Sistem Komputer*, vol. 7, no. 2, pp. 165–175, Sep. 2020, doi: 10.30656/PROSISKO.V7I2.2520.

[11] M. I. Aulawi, S. Amini, and S. Mulyati, "Implementasi Web Service dengan Metode Restful API dan QR Code untuk Aplikasi Manajemen Inventori pada Toko Indah Jaya Sport," *Jurnal Ticom : Technology of Information and Communication*, vol. 10, no. 3, pp. 211–217, May 2022, Accessed: Oct. 15, 2023. [Online]. Available: https://jurnal-ticom.jakarta.aptikom.or.id/index.php/Ticom/article/view/40

[12] F. Alfaridzi, J. D. Irawan, and M. Orisa, "Perancangan Sistem Manajemen User Hotspot Berbasis Web Menggunakan Application Programming Interface (API) Mikrotik," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 6, no. 2, pp. 974–981, Jan. 2022, doi: 10.36040/JATI.V6I2.5412.

[13] F. R. Pratama, I. Setyaningrum, and C. V. C. P. Nusantara, *GoLang Programming untuk Sains dan Teknik*. CV Cipta Prima Nusantara, 2021. [Online]. Available: https://books.google.co.id/books?id=xo9NEAAAQBAJ

[14] M. McGrath, *GO Programming in easy steps: Discover Google's Go language (golang)*. in In Easy Steps. In Easy Steps Limited, 2020. [Online]. Available: https://books.google.co.id/books?id=ba4IEAAAQBAJ

[15] S. Eseme, *Architecting Vue.js 3 Enterprise-Ready Web Applications: Build and deliver scalable and high-performance, enterprise-ready applications with Vue and JavaScript*. Packt Publishing, 2023. [Online]. Available: https://books.google.co.id/books?id=6023EAAAQBAJ

[16] G. F. S. Putra, J. Jumadi, and A. Al Akbar, "Implementation of the Turbo Boyer Moore Method in Searching Thesis Titles at the Website-Based Faculty of Economics," *Jurnal Media Computer Science*, vol. 2, no. 2, pp. 203–218–203–218, Jul. 2023, doi: 10.37676/JMCS.V2I2.4377.