Jurnal Sistem Informasi dan Teknologi
Vol. 6, No. 2 Mei 2024

https://journalversa.com/s/index.php/jsit

IMPLEMENTASI ARSITEKTUR MICROSERVICES DAN TEKNOLOGI DOCKER PADA APLIKASI TRACER ALUMNI

Yudi Fathurrahman[1], Pizaini[2], Novriyanto[3], Teddie Darmizal[4]
[1,2,3,4]Universitas Islam Negeri Sultan Syarif Kasim
Email: 111751101958@students.uin-suska.ac.id[1], pizaini@uin-suska.ac.id[2], noriyanto@uin-suska.ac.id[3], teddie.darmizal@uin-suska.ac.id[4]

Abstrak

Alumni merupakan sekumpulan orang yang telah berhasil menyelesaikan pendidikannya atau telah berhasil lulus dari suatu sekolah ataupun perguruan tinggi. Dalam melihat kualitas suatu institusi pendidikan, Alumni biasanya memegang peranan yang penting untuk berkontribusi dalam memberikan masukan terhadap proses dan perbaikan bagi institusi Pendidikan. Data alumni diperlukan setiap institusi pendidikan untuk keperluan akreditasi, yang dilakukan oleh Badan Akreditasi Nasional serta diperlukan juga oleh institusi pendidikan untuk menjadi evaluasi dari kurikulum yang ada, agar dapat diperbaiki dengan menyesuaikan perkembangan pendidikan dan dunia kerja. Oleh karna itu diperlukan sebuah aplikasi yang dapat digunakan oleh setiap institusi pendidikan, serta dapat membantu menjalin hubungan antara alumni, institusi pendidikan, dan pengguna alumni. Dalam pengembangan aplikasi yang dapat melakukan pendataan alumni pada setiap institusi Pendidikan. Kesulitan melakukan pengembangan aplikasi dengan codebase yang besar yaitu ketika terdapat kegagalan disaat penambahan fitur baru dan disaat terjadi satu kegagalan dalam aplikasi dapat mempengaruhi keseluruhan kinerja aplikasi karena setiap komponen sistem tidak terpisah satu sama lainnya. Kesulitan lainnya yaitu ketika banyak user yang mengakses aplikasi maka performa dari aplikasi akan menurun hal ini disebabkan aplikasi hanya membebankan ke satu service saja. Sehingga untuk mengatasi masalah tersebut, diperlukan yaitu arsitektur microservice. microservices dapat memberikan manfaat yang signifikan, seperti mengembangkan service dengan cepat. Penelitian ini bertujuan untuk mengimplementasikan arsitektur microservices dan pemanfaatan teknologi docker dalam pengembangan aplikasi tracer alumni. Pengujian dalam penelitian ini menggunakan pengujian benchmarking dengan Rakyll/Hey. Dalam hal ini, pengujian membuat 1000 HTTP connections yang kemudian dijalankan dilakukan selama x detik, diketahui bahwa request HTTP yang dapat diterima mencapai 336,24 permintaan per detik. Hal itu menunjukkan seberapa efektifnya pengimplementasian service orchestration pattern yang digunakan pada aplikasi tracer alumni. Pengujian juga mendapatkan informasi mengenai nilai minimum dan maximum jumlah HTTP request/seconds (rps) adalah 160,47 rps hingga 336,24 rps.

Kata Kunci: Alumni, Aplikasi, Docker, HTTP Connections, Microservice, Tracer

Jurnal Sistem Informasi dan Teknologi
Vol. 6, No. 2 Mei 2024

https://journalversa.com/s/index.php/jsit

Abstract

Alumni are people who have completed their education or have graduated from a school or college. In looking at the quality of an educational institution. Alumni data is needed by every educational institution for accreditation purposes, which is carried out by the National Accreditation Body and is also needed by educational institutions to evaluate the existing curriculum, so that it can be improved by adapting to developments in education. Therefore, we need an application that can be used by every educational institution, and can help establish relationships between alumni, educational institutions and alumni users. In developing an application that can collect alumni data at each educational institution. The difficulty in developing applications with a large codebase is when there is a failure when adding new features and when a single failure occurs in the application it can affect the overall performance of the application because each system component is not separate from each other. Another difficulty is that when many users access the application, the performance of the application will decrease, this is because the application only charges one service. So to overcome this problem, a microservice architecture is needed. microservices can provide significant benefits, such as developing services quickly. This research aims to implement microservices architecture and Docker technology in the development of the alumni tracer application. Benchmarking testing using Rakyll/Hey by making 1000 HTTP connections which are then executed for x seconds, it is known that the HTTP requests that can be received up to 336,24 requests per second, which shows how effective the use of the service orchestration pattern applied to the alumni tracer is. Testing also obtained information regarding the minimum and maximum values for the number of HTTP requests/seconds (rps) which were 160.47 to 336,24 rps.

Keywords*: Alumni, Application, Docker, HTTP Connections, Microservices, Tracer*

PENDAHULUAN

Alumni merupakan sekumpulan orang yang telah berhasil menyelesaikan pendidikannya atau telah berhasil lulus dari suatu sekolah ataupun perguruan tinggi. Dalam melihat kualitas suatu institusi pendidikan, Alumni memegang peranan penting dalam memberikan masukan terhadap proses dan perbaikan bagi institusi pendidikan. Data alumni diperlukan setiap institusi pendidikan untuk keperluan akreditasi, yang dilakukan oleh Badan Akreditasi Nasional. Data alumni juga diperlukan oleh institusi pendidikan untuk menjadi evaluasi dari kurikulum yang ada, agar dapat diperbaiki dengan menyesuaikan perkembangan pendidikan dan dunia kerja. Selain itu data alumni diperlukan pengguna alumni untuk melihat jejak pendidikan dari alumni yang bekerja dengannya.

Beberapa institusi pendidikan tidak berhubungan dengan alumninya lagi setelah lulus. Untuk mendapatkan data alumni, institusi pendidikan mendapatkan data dengan cara menghubungi langsung alumninya. Hal ini dinilai kurang efektif, dikarenakan tidak semua alumni dapat dihubungi, sehingga tidak semua alumni dapat terdata, dan akan memakan waktu yang relatif lama dalam mengumpulkan data alumni. Sedangkan institusi pendidikan yang mengurus alumninya dengan baik, memiliki website untuk mendata alumninya. Dengan adanya website tersebut, dapat membantu institusi pendidikan dalam mendata alumninya. Seperti pada penelitian yang telah dilakukan oleh. Pada penelitian ini Diana dan As'ad melakukan perancangan sistem tracer study dan telah dapat digunakan oleh alumni Universitas Prof. DR. Hazairin, SH Bengkulu. Penelitian lainnya telah diteliti oleh. Pada penelitian ini Jonni telah merancang sistem tracer study dan telah dapat digunakan oleh alumni STMIK Lepisi Tangerang. Berdasarkan beberapa penelitian tersebut, dapat disimpulkan bahwa website yang telah dibangun hanya terpaku pada satu institusi pendidikan, dan hanya dapat digunakan oleh alumni institusi pendidikan tersebut. Hal ini menyebabkan terjadinya pembuatan aplikasi yang sama secara berulang-ulang pada institusi pendidikan yang berbeda, sehingga terjadi redudansi data alumni diberbagai institusi pendidikan.

Oleh karna itu diperlukan sebuah aplikasi yang dapat digunakan oleh setiap institusi pendidikan, serta dapat membantu menjalin hubungan antara alumni, institusi pendidikan, dan pengguna alumni. Dalam pengembangan aplikasi yang dapat melakukan pendataan alumni pada setiap institusi pendidikan dari jenjang sekolah dasar, sekolah menengah pertama, sekolah menengah keatas, dan perguruan tinggi memiliki codebase yang besar. Pengembangan aplikasi dengan codebase yang besar sangat sulit karena kompleksitasnya. Kesulitan dalam pengembangan aplikasi dengan codebase yang besar yaitu ketika terdapat kegagalan disaat penambahan fitur baru, atau disaat terjadi satu kegagalan dalam aplikasi, akan mempengaruhi keseluruhan aplikasi, karena setiap komponen sistem tidak terpisah satu sama lainnya. Kesulitan lainnya yaitu ketika banyak user yang mengakses aplikasi maka performa dari aplikasi akan menurun hal ini disebabkan aplikasi hanya membebankan ke satu service saja

Dengan adanya permasalahan diatas, muncul sebuah arsitektur, untuk mengatasi masalah tersebut, yaitu arsitektur microservice. Arsitektur microservices adalah sebuah arsitektur, Dimana

dalam pengembangan aplikasinya dilakukan dan dibagi dalam service-service kecil yang dapat dan saling berkomunikasi satu sama lain. Arsitektur microservice memiliki keunggulan yaitu memberikan kebebasan kepada para pengembang untuk mengembangkan software dalam waktu yang relatif cepat. Dengan adanya service-service kecil yang saling berkomunikasi, maintenance aplikasi akan menjadi lebih mudah, karena jika terjadi suatu kesalahan, perbaikan hanya dilakukan pada satu bagian service saja, begitu pula ketika perlu dilakukan penambahan fitur, tidak perlu dilakukan pada seluruh sistem, dan tidak membebankan ke service lainnya. Oleh karena itu microservices dapat mengelola sistem menjadi lebih adaptif terhadap perubahan. Microservices dapat memberikan manfaat yang signifikan, yaitu merancang, mengembangkan, menguji, dan merilis layanan dengan gesit.

Dalam microservices, setiap service dapat dikembangkan oleh orang yang berbeda, dan dapat dikembangkan dengan bahasa pemrograman yang berbeda atau menawarkan layanan yang terpisah sehingga dapat memberikan sistem yang tetap menjamin availability apabila salah satu layanan perangkat lunak mengalami malfungsi atau dalam tahapan perbaikan. Hal ini menyebabkan runtime library atau package yang dibutuhkan suatu service menjadi tidak konsisten. Oleh karena itu, dalam menggunakan microservices akan menjadi sulit pada saat menyesuaikan antara satu service dan service lainnya. Untuk mengatasi masalah ini diperlukan teknologi Docker untuk memanajemen microservices. Docker container dapat memisahkan antara satu container dengan container lainnya untuk setiap service, yang dimana didalam satu container terdapat semua runtime library yang dibutuhkan service. Pemisahan dengan Docker container mempermudah dalam tahap pengembangan, pengujian, sampai ke tahap produksi.

Berdasarkan permasalahan yang telah diuraikan sebelumnya, penulis mengambil latar belakang pengambilan kasus penelitian dengan judul penelitian "Implementasi Arsitektur Microservices Dan Teknologi Docker Pada Aplikasi Tracer Alumni". Dimana dalam batasan masalah pengambilan kasus yaitu ruang lingkup penggunaan sistem ini hanya berkaitan dengan tracer alumni, institusi pendidikan yang diteliti adalah SD, SMP, SMA, dan Universitas serta aplikasi yang akan dibangun berbasis website.

METODE PENELITIAN

Metodologi penelitian dalam penelitian ini adalah rangkaian tahapan penelitian yang dibentuk sebagai pedoman dalam pelaksanaan penelitian yang dilakukan dalam penelitian. hal ini bertujuan untuk mendapatkan hasil dari penelitian sesuai dengan yang diharapkan berdasarkan perumusan penelitian sebelumnua. Adapun tahapan dalam penelitian ini ini terdiri dari analisis dan skenario, perancangan, implementasi, pengujian serta kesimpulan dan saran.

Analisis dan Skenario

Analisis adalah tahapan untuk mengetahui informasi awal terkait service yang diperlukan dalam pengimplementasian arsitektur microservice pada pengembangan aplikasi tracer alumni. Adapun Service-service ini dipisah-pisahkan sehingga pada tiap-tiap service dapat fokus dengan tugas-tugas yang bersifat ringan serta dapat saling berkomunikasi antar satu service untuk bekerja sama dalam pengembangan aplikasi tracer alumni. Berikut merupakan gambaran umum services yang akan dibuat beserta dengan penjelasan dari masing-masing fungsinya, yaitu:

1) API Gateway

   API gateway dalam penelitian ini sebagai single entry point pada aplikasi microservice yang dibangun. API gateway juga berfungsi sebagai middleware untuk melakukan enkapsulasi pada services di belakangnya.

2) Auth Service

   Auth service dalam penelitian ini merupakan service yang bertugas untuk mengelola data authentication yang akan digunakan pada aplikasi tracer alumni.

3) Alumni Service

   Alumni service dalam penelitian ini merupakan service untuk mengelola data alumni.

4) Institution Service

   Instutition service dalam penelitian ini merupakan service yang bertugas untuk mengelola data institusi pendidikan.

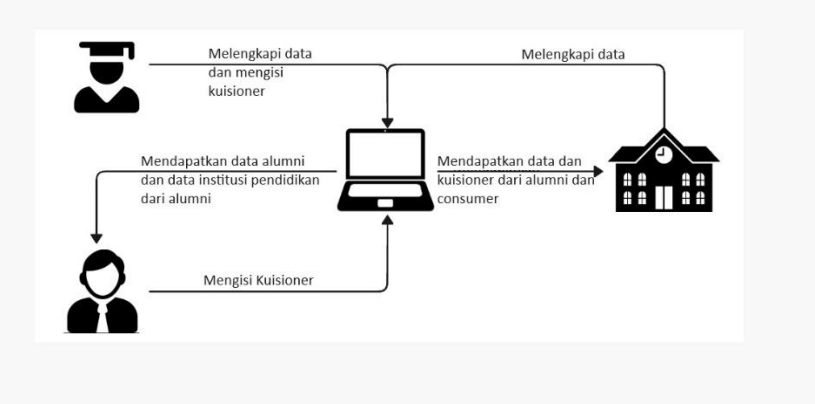
5) Tracer Service

Tracer service dalam penelitian ini merupakan service untuk handle kuisioner kuisioner dari alumni.

6) Database Service

Database service dalam penelitian ini merupakan service yang bertindak sebagai media penyimpanan atau database dari aplikasi.

Skenario dalam penelitian ini merupakan gambaran alur aplikasi atau perangkat lunak yang akan dibangun ataupun dikembangkan. Dimana, aplikasi tracer alumni ini akan menjalankan proses bisnis dari tracer alumni tersebut sehingga layak dan dapat digunakan pada tiap lembaga pendidikan. Berikut merupakan gambaran skenario umum serta proses bisnis inti pada aplikasi yang dibangun dan dikembangkan:



Gambar 1. Skenario Alur aplikasi tracer alumni

Dari gambar 1, dapat dilihat hasil yang ingin dicapai yaitu skenario dari aplikasi tracer alumni ini dimulai dengan pihak institusi mendaftar ke aplikasi. Kemudian alumni perlu mendaftar ke aplikasi tracer alumni dengan cara mengisi form pendaftaran berisi data diri, serta jenjang pendidikan, dan karir yang telah dilaluinya. Selanjutnya alumni akan mengisi kuisioner yang berkaitan dengan institusi pendidikan yang telah dilaluinya. Setelah itu pihak pengguna alumni perlu mendaftar ke aplikasi berupa data perusahaan, kemudian pengguna alumni akan mendapatkan jejak pendidikan, serta data institusi institusi daro pekerjanya yang terdaftar pada

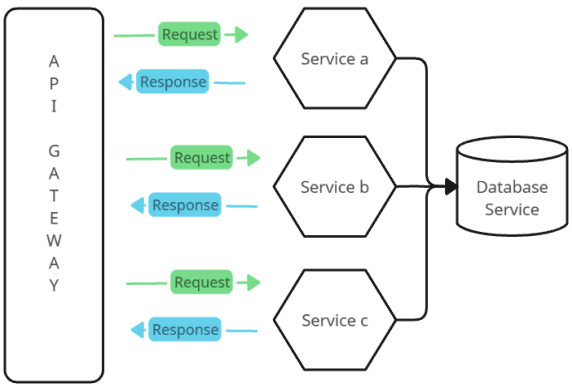
aplikasi. Selanjutnya pengguna alumni mengisi kuisioner dari institusi pendidikan yang memiliki alumni darinya. Setelah tahap tersebut dilalui institusi pendidikan akan mendapatkan data alumni, data pengguna alumni, serta hasil dari kuisioner yang telah diisi oleh alumni, dan pengguna alumni.

Perancangan

Perancangan pada penelitian ini terbagi atas dua tahapan, yakni sebagai berikut:

a. Perancangan Arsitektur

Pada tahap ini dilakukan perancangan arsitektur sesuai dengan kebutuhan dari aplikasi microservice tracer alumni. Adapun arsitektur microservice yang dibangun menggunakan orchestration pattern. Orchestration pattern ini merupakan salah satu pola dari arsitektur microservice, dimana seluruh hubungan antar service terletak pada satu service yaitu service API Gateway. Dengan menggunakan orchestration pattern ini, dapat memberikan kemudahan dalam hal maintainability dan scalability karena seluruh hubungan antar service berada pada API Gateway. Selain itu, hal ini juga akan memudahkan untuk membaca alur dari aplikasi serta akan mengurangi kompleksitas dari aplikasi. Dalam implementasi arsitektur microservices, service-service yang dirancang sebelumnya akan diimplementasikan menggunakan docker menjadi docker image. Setiap service akan menggunakan satu database service yang sama yaitu database service, namun menggunakan database yang berbeda sesuai dengan service yang bersangkutan. Selain itu, setiap service saling terhubung satu sama lain menggunakan docker network yang akan saling terhubung ke API gateway service.



Gambar 2 Metode Komunikasi Menggunakan Representational State Transfer (REST) pada Orchestration Pattern

Berdasarkan Gambar 2, dapat terlihat bahwa dengan penggunaan orchestration pattern, semua service tidak saling terhubung satu sama lain namun hanya dapat berkomunikasi melalui API Gateway. Adapun metode komunikasi yang digunakan yaitu Representational State Transfer (REST). Alur komunikasinya yaitu API Gateway akan berkomunikasi melalui HTTP request terhadap service dan service akan mengembalikan HTTP response ke API gateway service. Selanjutnya, API Gateway akan mengatur arah alur data akan dilanjutkan. Dari gambar tersebut juga terlihat bahwa semua service akan saling terhubung ke suatu database service, dimana setiap service akan mempunyai databasenya tersendiri nantinya.

b. Perancangan Service

Tahapan perancangan services adalah pemberian hasil dari analisis kebutuhan terhadap service-service yang telah ditentukan pada analisis skenario. Dalam tahapan ini juga akan ditentukan teknologi apa saja yang akan digunakan. Adapun pennetuan ini dimulai dari bahasa pemrograman yang akan digunakan, pengimplementasian framework, pemanfaatan teknologi atau platform, runtime hingga operating system pada masing-masing service tersebut. Berikut merupakan perancangan setiap service yang akan dibangun:

Nama Service	Bahasa	Framework	Sistem Operasi
Api Gateway	Python	FastAPI	Alpine Linux
Auth Service	Javascript	Express	Alpine Linux
Alumni Service	Javascript	Express	Alpine Linux
Institution Service	Javascript	Express	Alpine Linux
Tracer Service	Javascript	Express	Alpine Linux
Database Service	PostgreSQL		

Gambar 3 Perancangan Service

HASIL DAN PEMBAHASAN

A. Implementasi

Tahap implementasi merupakan tahap dimana seluruh service dari hasil analisis dan juga perancangan akan diimplementasikan dengan menggunakan teknologi Docker.

1) Implementasi Infrastructure as Code menggunakan Docker File

Setiap service akan dibangun menjadi suatu docker image dengan menggunakan Dockerfile. Dockerfile memungkinkan penyusunan setiap service dengan menggunakan environmentnya tersendiri. Teknik ini dikenal dengan Infrastructure as Code (IaC). Dengan adanya IaC dapat memudahkan proses instalasi dari service karena seluruh Docker File akan beroperasi dalam environment yang terkontainerisasi dan terisolasi yang disebut dengan Docker Container

Berikut merupakan hasil implementasi Infrastructure as Code menggunakan Dockerfile pada aplikasi microservice tracer alumni:

a) API Gateway

API Gateway dibangun menggunakan bahasa pemrograman Python dengan framework fastapi sesuai dengan tahap perancangan service. Gambar 4 menampilkan konfigurasi dari Dockerfile yang digunakan untuk menerapkan service sebagai docker image:

```
FROM python:3.8-alpine

WORKDIR /app

COPY requirements.txt /app/

RUN pip install --no-cache-dir -r requirements.txt

COPY . /app/

EXPOSE 8000

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Gambar 4. Konfigurasi Dockerfile API Gateway Service

API Gateway service diimplementasikan sebagai IaC menggunakan DockerFile seperti yang telah tertera pada gambar 4 diatas dengan keterangan sebagai berikut:

1. API gateway dijalankan melalui operating system Alpine Linux.
2. API gateway mengimplementasikan base image python:3-8.alpine.
3. API gateway diakses pada port 8000.

b) Auth Service

Auth service akan dibangun menggunakan framework Express.js pada Bahasa pemrograman Javascript sesuai dengan tahap perancangan service. Gambar 5(a) menampilkan konfigurasi dari Dockerfile yang digunakan untuk menerapkan service sebagai docker image:

Gambar 5 (a). Konfigurasi Dockerfile Auth Service; (b). Konfigurasi Dockerfile Alumni Service; (c). Konfigurasi Dockerfile Institution Service; (d). Konfigurasi Dockerfile Tracer Service

Auth service diimplementasikan sebagai IaC menggunakan DockerFile seperti yang telah tertera pada gambar 5(a) diatas dengan keterangan sebagai berikut:

1. Auth service dijalankan melalui operating system Alpine Linux.
2. Auth service mengimplementasikan base image node:16-alpine.
3. Auth service diakses pada port 8030.

c) Alumni Service

Alumni service akan dibangun menggunakan framework Express.js pada bahasa pemrograman Javascript sesuai dengan tahap perancangan service. Gambar 5(b) menampilkan konfigurasi dari Dockerfile yang digunakan untuk menerapkan service sebagai docker image. Alumni service diimplementasikan sebagai IaC menggunakan DockerFile seperti yang telah tertera pada gambar 5(b) diatas dengan keterangan sebagai berikut:

1. Alumni service dijalankan melalui operating system Alpine Linux.
2. Alumni service mengimplementasikan base image node:16-alpine.
3. Alumni service diakses pada port 8002.

d) Institution Service

Institution service akan dibangun menggunakan framework Express.js pada bahasa pemrograman Javascript sesuai dengan tahap perancangan service. Gambar 5(c) menampilkan konfigurasi dari Dockerfile yang digunakan untuk menerapkan service sebagai docker image. Institution service diimplementasikan sebagai IaC menggunakan DockerFile seperti yang telah tertera pada gambar 7 diatas dengan keterangan sebagai berikut:

1. Institution service dijalankan melalui operating system Alpine Linux.
2. Institution service mengimplemntasikan base image node:16-alpine.
3. Institution service diakses pada port 8001.

e) Tracer Service

Tracer service dibangun menggunakan framework Express.js pada bahasa pemrograman Javascript sesuai dengan tahap perancangan service. Gambar 5 (d) menampilkan konfigurasi dari Dockerfile yang digunakan untuk menerapkan service sebagai docker image. Tracer service diimplementasikan sebagai IaC menggunakan DockerFile seperti yang telah tertera pada gambar 5(d) diatas dengan keterangan sebagai berikut:

1. Tracer service dijalankan melalui operating system Alpine Linux.
2. Tracer service mengimplementasikan base image node:16-alpine.
3. Tracer service diakses pada port 8004.

2) Implementasi Infrastructure as Code (IaC) Menggunakan Docker Compose

Seluruh Dockerfile akan dibangun dan dirancang menggunakan docker compose. Penggunaan docker compose memberikan kemudahalan dalam proses build Dockerfile menjadi docker image. Serta untuk memudahkan proses instalasi dari setiap dependency, environment dan runtime yang diperlukan untuk setiap service.

Gambar berikut merupakan hasil implementasi build Dockerfile menjadi docker image melalui docker compose:

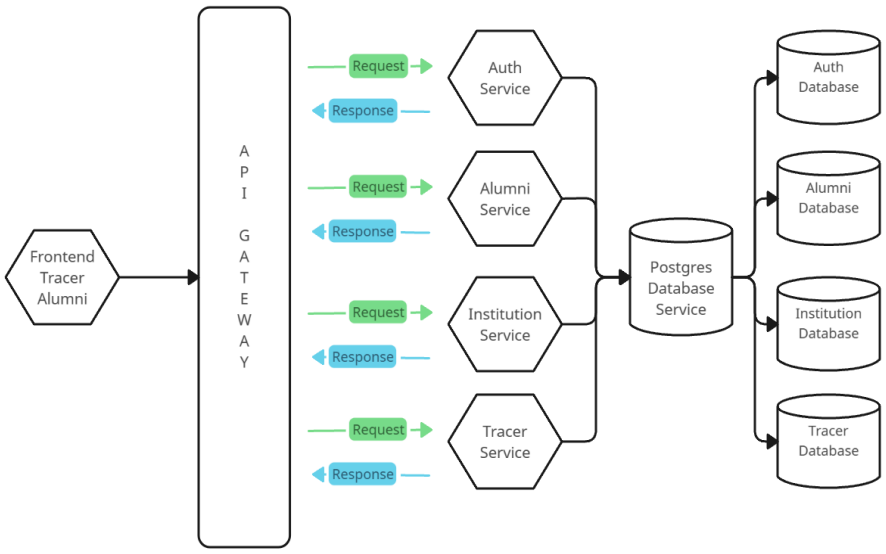
Gambar 6 (a). Konfigurasi Docker Compose; (b). Konfigurasi Docker Compose; (c). Konfigurasi Docker Compose

Melalui Gambar 6 (a); 6 (b) dan 6 (c) yang tertera diatas, service- service akan diinisialisasikan mulai dari nama container, lokasi folder yang akan dijadikan untuk Dockerfile. Hal ini akan digunakan untuk melakukan instalasi dari service, port yang dapat diakses oleh local dan Docker Network, dan juga hubungan antar service. Selanjutnya dilakukan konfigurasi pada database service dengan penjelasan sebagai berikut:

a) Database Service

Database service menggunakan PostgreSQL, yang akan di pull dari Docker Hub dengan docker image postgres:latest, dan instalasi dari seluruh service hanya akan berjalan jika database service telah berjalan dengan baik. Konfigurasi environment lainnya dapat dilihat pada gambar 6.

Ketika docker compose dijalankan menggunakan perintah “docker compose up”, maka seluruh service akan dibangun tepat setelah Database Service siap untuk digunakan berdasarkan instruksi yang telah didefinisikan. Setiap service yang telah dibangun akan menjadi docker container dan menjadi suatu kesatuan aplikasi microservice tracer alumni yang mampu berjalan sesuai dengan proses bisnis inti tracer alumni. Pada gambar 10 merupakan hasil implementasi arsitektur microservice pada aplikasi tracer alumni:



Gambar 7. Implementasi Service Orchestration Pattern pada Arsitektur Microservice Aplikasi Tracer Alumni

3) Perangkat Implementasi

Perangkat yang digunakan untuk melakukan implementasi microservices terhadap aplikasi tracer Alumni adalah:

1. Hardware atau perangkat keras

   Processor : Intel(R) Core (TM) i7-12700 @ 2.10 GHz

   Memory : 32 GB DDR4

   Hard disk : 500 GB

2. Software atau perangkat lunak

   Operating system : Windows 11 Pro 64-bit

   Web server : Uvicorn, Node.js

   Web browser : Microsoft Edge

   Bahasa Pemrograman : Python, JavaScript,

   Tools : Docker Desktop, Visual Studio Code, Postman DBMS

B. Pengujian

Penelitian ini melakukan beberapa tahapan pengujian, yaitu sebagai berikut:

1) Tools yang digunakan

Pada tahapan pengujian ini akan dilakukan serangkaian pengujian menggunakan tools Rakyll/Hey untuk mengetahui benchmark dari aplikasi ketika menggunakan arsitektur microservice. Rakyll/Hey merupakan tools benchmark HTTP modern yang mampu menghasilkan beban signifikan saat dijalankan terhadap endpoint dari aplikasi.

2) Cara Analisis

Aplikasi yang dirancang selanjutnya dilakukan validasi dan akan dieveluasi dengan parameter yaitu jumlah HTTP request/seconds (rps). Aplikasi client memberikan HTTP connection dan mengirimkannya ke server. Software server akan merespon dan mengirimkannya kembali ke client. Jumlah total HTTP connection yang konsisten per detik disebut sebagai RPS (request per second) semakin besar nilainya, menunjukan semakin bagus kualitas dari software yang sedang di test.

Dalam penelitian ini, pengujian dilaksanakan dengan membuat beberapa HTTP connections dan akan diukur berapa lama waktu yang diperlukan untuk menyelesaikan HTTP connections tersebut. Data tersebut nantinya akan ditampilkan ke dalam tabel yang digunakan untuk melihat karakteristik aplikasi yang dibuat dengan menggunakan service orchestration pattern pada arsitektur microservice tracer alumni dalam menangani beban kerja yang telah ditentukan.

1. Hasil Evaluasi

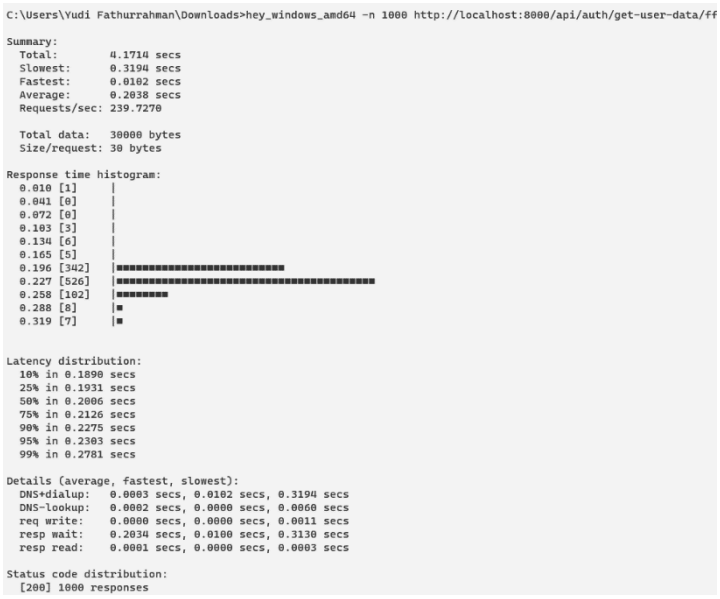
Pada pengujian ini, dilakukan dengan melakukan 1000 HTTP connections terhadap endpoint dari aplikasi. Benchmarking akan dilakukan terhadap endpoint API gateway yang mengarah ke alumni service, endpoint auth/account-user-data pada auth service, serta endpoint get data pada masing-masing service, yakni alumni service, tracer service dan institution service. Tujuan pengujian benchmarking ini adalah untuk mengetahui jumlah HTTP request/seconds (rps) ketika menggunakan service orchestration pattern pada arsitektur microservice guna mengetahui batas yang dapat dicapai oleh aplikasi tracer alumni itu sendiri.

Pada tabel I, ditampilkan secara berurutan hasil benchmarking terhadap endpoint API gateway yang mengarah ke alumni service, endpoint auth/account-user-data pada auth service, serta endpoint get data pada masing-masing service, yakni alumni service, tracer service, dan institution service.

Tabel 1. Pengujian endpoint API

Endpoint	Minimum (s)	Maximum (s)	Rata-rata (s)	*Requests/ Second*	Total *response time* (s)	Rata-rata *response wait* (s)	Rata-rata *response read* (s)
https://localhost:8088/api/alumni/riwayat/id	0,0200	0,2828	0,1631	299,3302	3,3408	0,1621	0,0008
https://localhost:8088/api/account-user-data	0,0102	0,3194	0,2038	239,7270	4,1714	0,2034	0,0003
https://localhost:8088/api/institution	0,0302	0,3788	0,1466	336,2450	2,9740	0,1459	0,0140
https://localhost:8088/api/tracer/kuisioner/institution/alumni/id	0,0835	0,6095	0,3053	160,4705	6,2317	0,3041	0,0005

Tabel 1 menunjukkan bahwa nilai minimum dan maximum jumlah HTTP request/seconds (rps) adalah 160,47 rps hingga 336,24 rps. Dimana jumlah HTTP request/seconds (rps) tertinggi adalah pada institution service dan jumlah HTTP request/seconds (rps) terendah adalah pada institution service. Berikut dijabarkan hasil awal tabel pengujian per end point,



Gambar 8. Pengujian endpoint account-user-data pada auth service

Jurnal Sistem Informasi dan Teknologi
Vol. 6, No. 2 Mei 2024

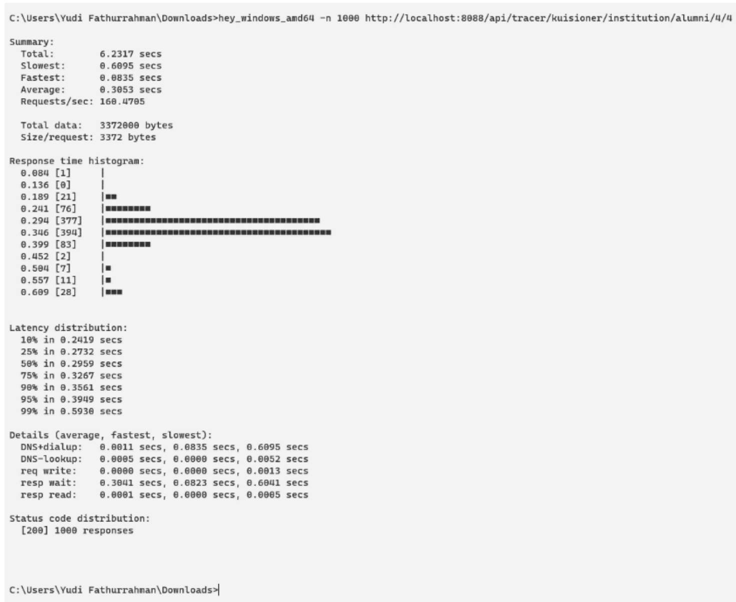
https://journalversa.com/s/index.php/jsit

Gambar 8 menujukkan bahwa dalam endpoint account-user-data pada auth service memiliki rata-rata waktu untuk mempersiapkan request sekitar 0,0102 sec hingga 0,3194 sec. Dalam satu datuan detik, request dapat diperersiapkan sekitar 239,7270 request. Serta melalui gambar tersebut menunjukkan bahwa waktu yang diperlukan untuk mempersiapkan request melalui 1000 HTTP connections adalah sekitar 4,1714 secs.

```
C:\Users\Yudi Fathurrahman\Downloads>hey_windows_amd64 -n 1000 http://localhost:8000/api/institution/

Summary:
  Total:        2.9740 secs
  Slowest:      0.3788 secs
  Fastest:      0.0302 secs
  Average:      0.1466 secs
  Requests/sec: 336.2450

  Total data:   1495000 bytes
  Size/request: 1495 bytes

Response time histogram:
  0.030 [1]     |
  0.065 [6]     |
  0.100 [11]    |■
  0.135 [380]   |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  0.170 [488]   |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  0.205 [67]    |■■■■■
  0.239 [4]     |
  0.274 [9]     |■
  0.309 [23]    |■■
  0.344 [3]     |
  0.379 [8]     |■


Latency distribution:
  10% in 0.1234 secs
  25% in 0.1299 secs
  50% in 0.1383 secs
  75% in 0.1497 secs
  90% in 0.1748 secs
  95% in 0.1986 secs
  99% in 0.3218 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0003 secs, 0.0302 secs, 0.3788 secs
  DNS-lookup:   0.0002 secs, 0.0000 secs, 0.0058 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0003 secs
  resp wait:    0.1459 secs, 0.0301 secs, 0.3725 secs
  resp read:    0.0003 secs, 0.0000 secs, 0.0140 secs

Status code distribution:
  [200] 1000 responses
```

Gambar 9. Pengujian institution service

Gambar 9 menujukkan bahwa pada institution service memiliki rata-rata waktu untuk mempersiapkan request sekitar 0,0302 sec hingga 0,3788 sec. Dalam satu datuan detik, request dapat diperersiapkan sekitar 336,2450 request. Serta melalui gambar tersebut menunjukkan bahwa waktu yang diperlukan untuk mempersiapkan request melalui 1000 HTTP connections adalah sekitar 2,9740 secs.



```
C:\Users\Yudi Fathurrahman\Downloads>hey_windows_amd64 -n 1000 http://localhost:8088/api/tracer/kuisioner/institution/alumni/4/4

Summary:
  Total:        6.2317 secs
  Slowest:      0.6095 secs
  Fastest:      0.0835 secs
  Average:      0.3053 secs
  Requests/sec: 160.4705

  Total data:   3372000 bytes
  Size/request: 3372 bytes

Response time histogram:
  0.084 [1]     |
  0.136 [0]     |
  0.189 [21]    |■■
  0.241 [76]    |■■■■■■■■
  0.294 [377]   |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  0.346 [394]   |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  0.399 [83]    |■■■■■■■■
  0.452 [2]     |
  0.504 [7]     |■
  0.557 [11]    |■
  0.609 [28]    |■■■


Latency distribution:
  10% in 0.2419 secs
  25% in 0.2732 secs
  50% in 0.2959 secs
  75% in 0.3267 secs
  90% in 0.3561 secs
  95% in 0.3949 secs
  99% in 0.5930 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0011 secs, 0.0835 secs, 0.6095 secs
  DNS-lookup:   0.0005 secs, 0.0000 secs, 0.0052 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0013 secs
  resp wait:    0.3041 secs, 0.0823 secs, 0.6041 secs
  resp read:    0.0001 secs, 0.0000 secs, 0.0005 secs

Status code distribution:
  [200] 1000 responses



C:\Users\Yudi Fathurrahman\Downloads>
```

Gambar 10. Pengujian endpoint alumni pada tracer service

Gambar 10 menujukkan bahwa dalam endpoint alumni pada tracer service memiliki rata-rata waktu untuk mempersiapkan request sekitar 0,0835 sec hingga 0,6095 sec. Dalam satu datuan detik, request dapat diperersiapkan sekitar 160,4705 request. Serta melalui gambar tersebut menunjukkan bahwa waktu yang diperlukan untuk mempersiapkan request melalui 1000 HTTP connections adalah sekitar 6,2317 secs

KESIMPULAN

Berdasarkan permasalahan yang ada, penulis melakukan penelitian dengan judul penelitian Implementasi Arsitektur Microservices Dan Teknologi Docker Pada Aplikasi Tracer Alumni. Dimana pada penelitian ini, penulis membuat batasan masalah dalam pengambilan kasus yaitu ruang lingkup penggunaan sistem ini hanya berkaitan dengan tracer alumni, institusi pendidikan yang diteliti adalah SD, SMP, SMA, dan Universitas serta aplikasi yang akan dibangun berbasis website. Adapun kesimpulan dari penelitian yang telah dilakukan ini adalah keberhasilnya dilakukan implementasi dari service orchestration pattern pada arsitektur microservice untuk aplikasi tracer alumni yang terbentuk atas service-service yaitu API gateway, alumni service, auth

service, alumni service, institution service, tracer service dan database service. Dalam pengujian dilakukan benchmarking menggunakan Rakyll/Hey dengan melakukan 1000 HTTP connections terhadap endpoint dari aplikasi, jika dilihat dari Gambar 9 menujukkan bahwa pada institution service memiliki rata-rata waktu untuk mempersiapkan request sekitar 0,0302 sec hingga 0,3788 sec. Dalam satu datuan detik, request dapat diperersiapkan sekitar 336,2450 request. Serta melalui gambar tersebut menunjukkan bahwa waktu yang diperlukan untuk mempersiapkan request melalui 1000 HTTP connections adalah sekitar 2,9740 secs serta jika dilihat pada Gambar 10, maka request HTTP yang diterima dapat mencapai dari 160,47 hingga 336,24 request/ second, yang mencerminkan betapa efektifnya service orchestration pattern yang digunakan pada aplikasi tracer alumni.

DAFTAR PUSTAKA

A. Said et al., “Eksistensi Dan Peran Alumnidalam Menjaga Kualitas Mutu Fakultas Dakwah,” 2011.

M. Rizka, A. Amri, H. Hendrawaty, and M. Mahdi, “Analisis Dan Perancangan Sistem Informasi Tracer Study Berbasis WEB,” Jurnal Infomedia, vol. 3, no. 2, 2018, doi: 10.30811/jim.v3i2.716.

M. Jonni, “Sistem Informasi Pendataan Alumni Berbasis Web Stmik Lepisi Tangerang,” Jurnal Teknik, vol. 4, no. 1, pp. 5–12, 2015.

N. Dragoni et al., “Microservices: yesterday, today, and tomorrow,” Jun. 2016, [Online]. Available: http://arxiv.org/abs/1606.04036

T. Ueda, T. Nakaike, and M. Ohara, “Workload Characterization for Microservices,” IEEE , pp. 1–10, 2016.

F. Arifien and dan Edy Sutomo, “Implementasi Arsitektur Microservices Pada Sistem Informasi Akademik Stmik Jakarta Sti&K Menggunakan Model Enterprise Javabeans (Ejb) Dan Polymer Js,” Universitas Gunadarma Jl. Margonda Raya, vol. 5, no. 1, p. 16424, 2021.

G. Munawar, J. T. Komputer, D. Informatika, P. Negeri, B. Bandung, and A. Hodijah, “Analisis Model Arsitektur Microservice Pada Sistem Informasi DPLK,” Publikasi Jurnal & Penelitian Teknik Informatika, vol. 3, no. 1, 2018.

C. Pahl and P. Jamshidi, “Microservices: A systematic mapping study,” in CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science, SciTePress, 2016, pp. 137–146. doi: 10.5220/0005785501370146.

D. Saputra and R. Fathoni Aji, “Analisis Perbandingan Performa Web Service Rest Menggunakan Framework Laravel, Django Dan Ruby On Rails Untuk Akses Data dengan Aplikasi Mobile (Studi Kasus: Portal E-Kampus STT Indonesia Tanjungpinang),” Bangkit Indonesia, vol. 2, 2018.

T. A. Diajukan, M. Salah, S. Persyaratan, and M. Derajat, “Implementasi Arsitektur Microservice Pada Aplikasi Web Pengajaran Agama Islam Home Pesantren,” 2020.

F. Ariadi, C. Iswahyudi, E. Nurnawati, J. Informatika, and I. Akprind, “Penerapan Docker Container Sebagai Teknologi Ramah Skalabilitas Dibanding Teknik Virtualisasi Untuk Membangun Website Di Ubuntu 18.04.4 LTS,” vol. 8, no. 2, 2020.

F. X. Senduk, X. B. N. , Najoan, and S. R. U. A. Sompie, “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, 2023, [Online]. Available: https://ejournal.unsrat.ac.id/index.php/informatika

D. J. Riyanto, N. H. Safaat, and M. Affandes, “Implementasi Service Choreography Pattern Arsitektur Microservice Classroom Akademik Menggunakan Docker.”

U. Syarif and Pizaini, “Penerapan Event-Driven Microservices Pada Aplikasi Layanan Penerimaan Peserta Didik Baru,” JIPI, vol. 07, no. 03, pp. 745–756, 2022.

A. A. Zabar and F. Novianto, “Keamanan Http Dan Https Berbasis Web Menggunakan Sistem Operasi Kali Linux,” Jurnal Ilmiah Komputer dan Informatika (KOMPUTA), vol. 69, no. 2, 2015.

D. Al Fansha, M. Yusril, H. Setyawan, and M. N. Fauzan, “Load Test pada Microservice yang menerapkan CQRS dan Event Sourcing.”