Perancangan dan Implementasi Protokol *OAuth 2.0* pada Pengembangan Produk Coofis Verse di PT ARM Solusi

1st Diza Aulia Kusnadi
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
dizaauliakusnadi@student.telkomuniversity.ac.id

2nd Muhammad Iqbal
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
miqbal@telkomuniversity.ac.id

3rd Bramandityo Prabowo
Network, Hardware, & Infrastructure
PT Andal Rancang Multi Solusi
Bandung, Indonesia
bram@armsolusi.com

***Abstrak* — PT Andal Rancang Multi Solusi (PT ARM Solusi) adalah perusahaan teknologi yang berfokus pada pengembangan big data, analisis data, kolaborasi, otomatisasi administrasi, integrasi aplikasi, dan API. PT ARM Solusi memiliki produk aplikasi Coofis (*Collaboration Office*) yang bertujuan untuk meningkatkan efisiensi dengan menghilangkan penggunaan kertas melalui otomatisasi proses administrasi. PT ARM Solusi sedang mengembangkan produk Coofis menjadi Coofis Verse dengan mengubah arsitektur dari monolitik ke microservice. Salah satu layanan penting dalam pengembangan ini adalah layanan *Auth* (*Authentication*), yang memverifikasi identitas pengguna sebelum memberikan akses ke aplikasi. Penerapan *Auth* menggunakan protokol OAuth 2.0, yang menghasilkan otorisasi melalui *request* token, *access* token, dan *refresh* token untuk autentikasi pengguna API. Pada penelitian ini, telah dilakukan perancangan dan implementasi Protokol *OAuth 2.0* dengan menerapkan sistem *Single Sign-On* dalam Pengembangan Produk Coofis Verse di PT ARM Solusi. Penerapan sistem *Single Sign-On* menggunakan protokol *OAuth 2.0* dirancang menggunakan *automation tool* yang disebut Ansible dan menggunakan sebuah *platform open-source* sebagai penyedia layanan otentikasi dan otorisasi bagi aplikasi yang disebut Keycloak. Hasil dari perancangan Protokol *OAuth 2.0*, mencakup pembuatan sistem *Single Sign-On* dengan menggunakan *platform* Keycloak yang memenunjukkan bahwa implementasi Keycloak memungkinkan pengguna untuk mengakses berbagai aplikasi dengan satu set kredensial, sehingga meningkatkan kenyamanan pengguna dan efisiensi operasional.**

***Kata kunci*— PT Andal Rancang Multi Solusi, Coofis, *OAuth 2.0,* Protokol, *Single Sign-On*, Keycloak**

I. PENDAHULUAN

A. Latar Belakang

PT Andal Rancang Multi Solusi (PT ARM Solusi) adalah perusahaan teknologi yang berfokus pada pengembangan *big data*, analisis data, kolaborasi, otomatisasi administrasi, integrasi aplikasi, dan API. Beroperasi di Jakarta, Bandung, dan Surabaya, perusahaan ini berupaya meningkatkan efisiensi proses administrasi perusahaan dengan menghilangkan penggunaan kertas [1]. Saat ini, PT ARM Solusi sedang meningkatkan layanan produk Coofis (*Collaboration Office*) menjadi Coofis Verse dengan mengubah arsitektur dari monolitik menjadi *microservice*. Arsitektur monolitik menggabungkan semua elemen sistem, membuat pengujian dan perbaikan sulit, sementara arsitektur *microservice* memecah proses menjadi layanan kecil yang independen, meningkatkan fleksibilitas, skalabilitas, dan keandalan [2]. Salah satu layanan dalam peralihan ini adalah *Authentication*, yang membutuhkan kerahasiaan pengguna untuk identifikasi dan memerlukan login ulang setiap kali mengakses situs berbeda, untuk mengatasi masalah tersebut dapat menggunakan *authorization*. *Authorization* adalah tindakan pemberian hak akses kepada pihak ketiga untuk mengakses data yang dimiliki, proses ini hanya bisa dilakukan setelah menyelesaikan proses *authentication* sehingga informasi yang diberikan sesuai dengan pengguna yang telah melakukan proses *login*.[3] Penerapan dari *Authentication* adalah penerapan *OAuth 2.0* dalam menghasilkan otorisasi dengan menggunakan *request* token, *access* token, dan *refresh* token untuk autentikasi pengguna API.

B. Rumusan Masalah

Adapun rumusan masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimanakah merancang dan mengimplementasikan Protokol *OAuth 2.0* dalam pengembangan produk Coofis Verse di PT ARM Solusi berbasis arsitektur *microsevrices*?
2. Bagaimanakah cara mengintegrasikan Protokol *OAuth 2.0* dengan arsitektur *microservices*?

C. Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan Protokol *OAuth 2.0* dalam pengembangan produk Coofis Verse di PT ARM Solusi menggunakan arsitektur *microservices*.
2. Menyediakan proses autentikasi dan otorisasi secara aman, efisien, dan sesuai standar keamanan terkini dalam pengembangan produk Coofis Verse di PT ARM Solusi.

D. Manfaat

Adapun manfaat dari penelitian ini adalah sebagai berikut:

1. Menyediakan lapisan keamanan tambahan dengan mengontrol dan membatasi akses pengguna ke sumber daya aplikasi.
2. Meningkatkan kenyamanan pengguna dan mengurangi kebutuhan untuk mengingat banyak kredensial.

E. Batasan Masalah

Adapun Batasan masalah dari penelitian ini adalah sebagai berikut:

1. Fokus pada implementasi Protokol *OAuth 2.0* dalam pengembangan produk Coofis Verse di PT ARM Solusi.
2. Tidak membahas implementasi protokol keamanan lain selain *OAuth 2.0*.
3. Integrasi Protokol *OAuth 2.0* dalam pengembangan produk Coofis Verse akan diimplementasikan dalam aplikasi berbasis *web*.
4. Perancangan Protokol *OAuth 2.0* menggunakan arsitektur *Single Node*.
5. Semua pengguna didaftarkan oleh admin (*pre-register*) karena tidak terdapat fitur *self-register* pada sistem *Single Sign-On* produk Coofis Verse.

II. KAJIAN TEORI

A. *Microservice*

Microservice adalah suatu desain infrastruktur yang dapat digunakan untuk membuat suatu aplikasi menjadi terdiri dari berbagai layanan sendiri tetapi tetap saling terhubung satu sama lain. *Microservice* ini biasanya bertujuan untuk meningkatkan fleksibilitas, skalabilitas, dan efisiensi dalam pengembangan perangkat lunak. Tidak seperti aplikasi monolitik, *microservice* membagi aplikasi menjadi layanan yang lebih kecil yang saling terhubung dan lebih terukur. Arsitektur *microservice* merupakan alternatif arsitektur yang lebih fleksibel dan terukur. Pada arsitektur *microservice*, sistem informasi dirancang untuk terdistribusi dan menyediakan layanan secara lebih fokus dan spesifik [4].

B. Linux

Linux adalah sistem operasi yang menggunakan kernel yang pertama kali ditulis oleh Linus Torvalds, seorang mahasiswa dari Finlandia, untuk arsitektur Intel 80386. Linux didistribusikan di bawah lisensi *General Public License* (GNU), yang merupakan lisensi *copyleft* bebas. Lisensi ini menjamin kebebasan untuk mendistribusikan dan memodifikasi semua versi dari sebuah program, dengan syarat bahwa kode sumber asli harus disertakan. Linux tersedia dalam berbagai distribusi, yang merupakan paket dari kernel Linux, sistem dasar, program instalasi, alat dasar, dan program lainnya yang berguna sesuai dengan tujuan pembuatan distribusi tersebut. Beberapa distribusi Linux yang populer meliputi Ubuntu, Debian, Red Hat, Fedora, dan CentOS. [5].

C. Ubuntu

Ubuntu adalah salah satu distribusi sistem operasi berbasis Debian yang paling populer dan banyak digunakan di dunia serta didistribusikan oleh Canocical Ltd pada tahun 2004 oleh Mark Shuttleworth. Ubuntu digunakan untuk beberapa hal seperti *desktop*, *server*, *cloud*. Dalam pengembangan perangkat lunak dan infrastruktur TI, Ubuntu menyediakan platform yang kuat dan fleksibel. Keamanan juga menjadi salah satu fokus utama Ubuntu. Dengan fitur-fitur seperti AppArmor untuk kontrol akses wajib, pembaruan keamanan yang cepat, dan dukungan penuh terhadap enkripsi data, Ubuntu memastikan bahwa sistem tetap terlindungi dari ancaman keamanan [6].

D. Ansible

Ansible adalah alat *provisioning* yang didistribusikan oleh Red Hat Linux. Alat ini membantu mencatat setiap proses *deployment* dan konfigurasi berulang untuk berbagai server dan jaringan. Dengan demikian, Ansible memungkinkan otomatisasi penyebaran *cloud*, manajemen konfigurasi, penerapan aplikasi, orkestrasi, dan kebutuhan TI lainnya [7]. Ansible termasuk dalam kategori CMT (*Configuration Management Tools*) yang mengubah proses manajemen infrastruktur dan *deployment* dari manual menjadi otomatis. Untuk melakukan konfigurasi, deployment, dan orkestrasi, Ansible menggunakan playbook [8].

E. *OAuth*

OAuth (*Open Authorization*) adalah protokol otorisasi yang memungkinkan pengguna mengakses aplikasi tanpa harus membagikan kata sandi mereka. Aplikasi yang diintegrasikan menggunakan kredensial pengguna dengan teknologi otentikasi dari penyedia API (*Application Programming Interface*). OAuth juga memungkinkan otorisasi API yang terlindungi dari aplikasi desktop atau web melalui metode yang sederhana dan standar [9].

F. *OAuth 2.0*

OAuth 2.0 merupakan pengembangan lanjutan dari protokol *OAuth* yang pertama kali dibuat pada akhir tahun 2006. *OAuth 2.0* lebih menekankan kemudahan bagi klien untuk bertindak sebagai pemilik dan pengembang aplikasi dengan memberikan otorisasi khusus untuk berbagai aplikasi.[9] Secara umum, penerapan protokol *OAuth 2.0* mencakup dua aspek utama, yaitu: 1) *Federated Identity*: Bagian dari *identity management* yang memungkinkan pengguna masuk ke sebuah aplikasi menggunakan akun dari aplikasi lain. 2) *Delegated Authority*: Otorisasi yang didapatkan dari aplikasi lain, yaitu memungkinkan sebuah layanan untuk mengakses sumber daya pada layanan lain atas nama pengguna [10].

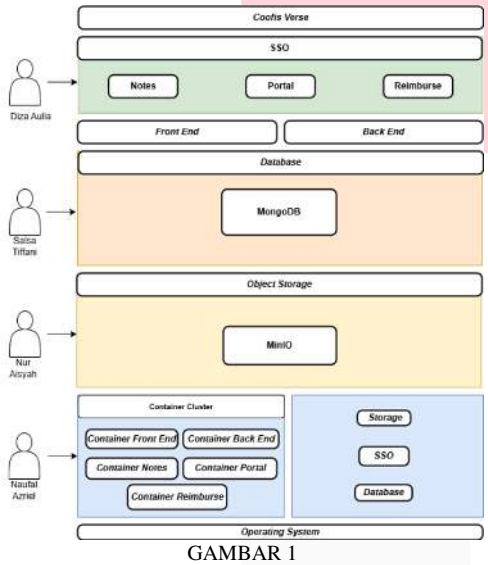
G. Keycloak

Keycloak adalah *platform open-source* untuk *Identity and Access Management* (IAM) yang dirancang untuk aplikasi dan layanan *modern*. *Platform* ini menyediakan berbagai fitur termasuk *single sign-on*, *identity broker*, *social login*, *user federation*, *account management console*, *centralized management*, LDAP and *Active Directory*, *authorization service* dan masih banyak lagi. Keycloak dibuat berdasarkan serangkaian *User Interface* (UI) dan *Representional State Transfer* (REST) *Application Programming Interface* (API) untuk menyediakan cara mengelola hak akses dan kebijakan untuk melindungi sumber daya. Keycloak mengatur

kebijakan otorisasi dan menerapkannya dalam aplikasi dan layanan yang dibuat, agar dapat memastikan perlindungan yang efektif terhadap sumber daya tersebut [11].

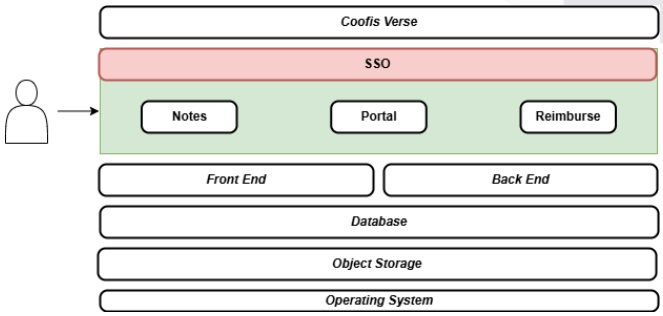
III. METODE

Pada Proyek Akhir ini akan dilakukan perencanaan *Single Sign-On* (SSO) menggunakan protokol *OAuth 2.0* dalam pengembangan produk Coofis Verse di PT ARM Solusi menggunakan *platform open-source* yaitu Keycloak. Coofis Verse adalah salah satu layanan atau produk yang di buat dan dipasarkan oleh PT ARM Solusi kepada perusahaan-perusahaan BUMN. Produk ini merupakan layanan yang digunakan dalam tata kelola persuratan, dan terdapat beberapa fitur seperti dashboard, surat internal, surat eksternal, surat disposisi, dan sebagainya.



GAMBAR 1
Diagram Sistem Infrastruktur Coofis

Pada Gambar 1, ditunjukkan mengenai diagram sistem infrastruktur dari produk aplikasi Coofis Verse. Blok diagram sistem ini menggambarkan secara garis besar mengenai arsitektur perancangan produk Coofis Verse dengan arsitektur *microservice* dengan tujuan dapat lebih mudah dimengerti dan dipahami. Bagian blok diagram yang akan dijelaskan secara rinci adalah mengenai sistem *Single Sign-On* dari produk Coofis Verse yang akan ditunjukkan pada Gambar 2.



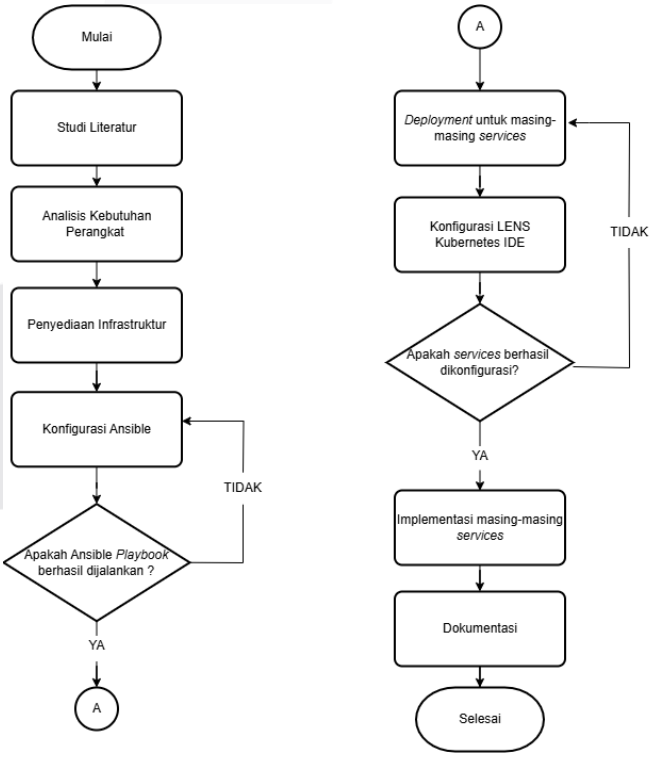
GAMBAR 2
Blok Diagram SSO

Proses perancangan *OAuth 2.0* untuk Coofis Verse dimulai dengan analisis kebutuhan pengguna dan spesifikasi teknis yang diperlukan untuk memastikan keamanan dan kenyamanan. Protokol *OAuth 2.0* dipilih karena dalam konteks *Single Sign-On*, *OAuth 2.0* memungkinkan pengguna untuk mengakses berbagai aplikasi dengan satu set kredensial tanpa perlu berulang kali melakukan *login*. Hal ini dilakukan dengan menggunakan token akses yang bersifat sementara, sehingga meningkatkan keamanan karena tidak perlu berbagi kata sandi langsung dengan aplikasi pihak ketiga. Dengan *OAuth 2.0*, pengguna dapat memberikan izin kepada aplikasi untuk mengakses informasi tertentu tanpa harus mengungkapkan kata sandi mereka, mengurangi risiko pencurian kredensial.

Perancangan *Single Sign-On* (SSO) menggunakan Keycloak melibatkan penggunaan *platform open-source* yang digunakan untuk mengelola autentikasi, otorisasi, dan manajemen pengguna dalam sebuah lingkungan aplikasi yang terintegrasi. Keycloak menyediakan berbagai fitur yang mendukung implementasi SSO dengan *OAuth 2.0* dan *OpenID Connect*, serta menyederhanakan integrasi dengan berbagai aplikasi dan layanan. Keycloak tidak hanya memberikan lapisan keamanan yang diperlukan untuk melindungi data sensitif dan mengelola akses pengguna secara efektif, tetapi juga memberikan fleksibilitas dalam integrasi dengan infrastruktur IT yang ada

A. Diagram Alir Proses Pengerjaan

Pada bagian ini, ditampilkan diagram alir yang menggambarkan ilustrasi proses yang dilakukan dalam perancangan protocol *OAith 2.0* untuk produk aplikasi. Berikut diagram alir perancangan protocol *OAuth 2.0* yang dapat dilihat pada Gambar 3.



GAMBAR 3
Diagram Alir Proses Pengerjaan

Proses perancangan produk Coofis Verse dibagi menjadi enam tahap. Tahap pertama melibatkan studi literatur dan analisis kebutuhan untuk mengidentifikasi kebutuhan

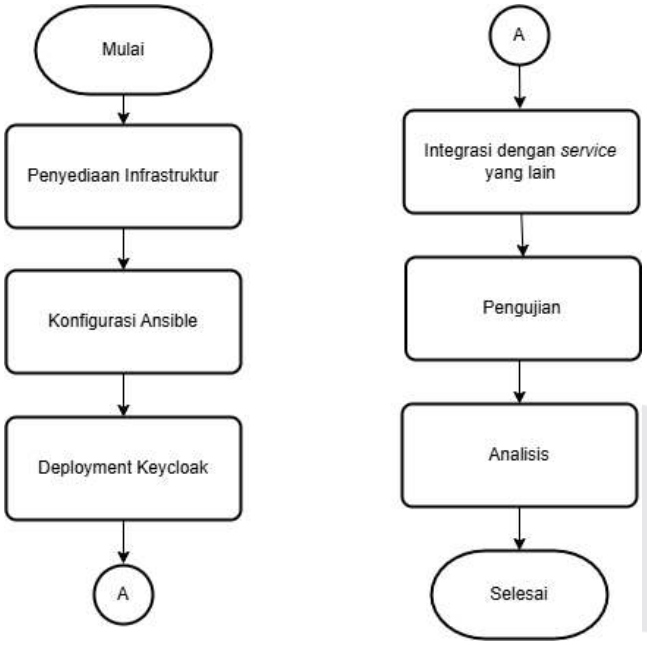
berbagai *services* dari produk Coofis Verse. Tahap kedua berfokus pada analisis kebutuhan perangkat, yang mencakup identifikasi kebutuhan teknis dan non-teknis yang diperlukan untuk mengembangkan dan menjalankan produk aplikasi dengan sukses. Tahap ketiga melibatkan penyediaan infrastruktur, termasuk pengaturan dan penyiapan komponen teknis seperti server fisik atau virtual, instalasi sistem operasi Linux, dan konfigurasi jaringan. Tahap keempat melibatkan konfigurasi Ansible untuk mengatur dan mengelola infrastruktur aplikasi secara otomatis. Tahap kelima menggunakan LENS, sebuah IDE khusus untuk Kubernetes, untuk mengelola dan memantau kluster Kubernetes dengan lebih efisien. Tahap keenam melibatkan implementasi services, di mana layanan-layanan inti yang mendukung fungsionalitas aplikasi dirancang, dikembangkan, dan diintegrasikan ke dalam sistem.

B. Diagram Alir Perancangan Sistem

Pada perancangan sistem mengenai implementasi *Single Sign-On* menggunakan platform Keycloak untuk produk aplikasi Coofis Verse dilakukan dengan dua proses yaitu *deployment* dan *operations*.

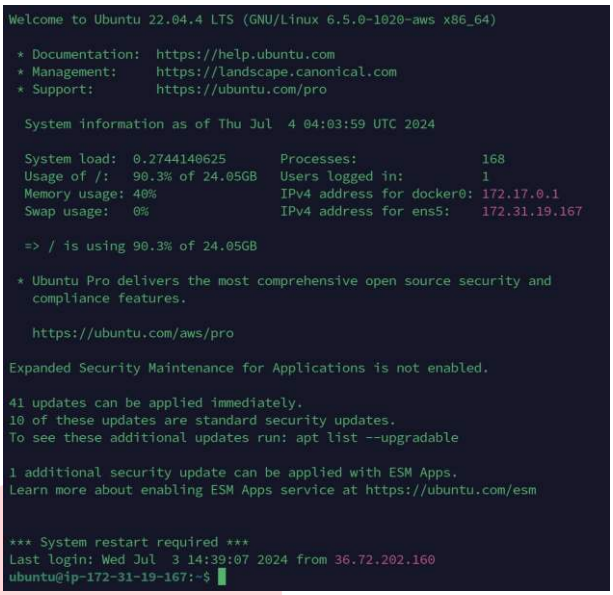
1. Deployment

Tahap *deployment* dari pembuatan aplikasi melibatkan beberapa langkah penting yang memastikan aplikasi berjalan dengan baik. Diagram alir tahapan yang akan dilakukan dapat dilihat pada Gambar 4.



GAMBAR 4
Perancangan Sistem *Deployment*

a. Tahap pertama yaitu melakukan penyediaan infrastruktur berupa penyediaan server *virtual machine* yang sudahh berjalan, lalu lakukan peng*install*-an Ansible.



Gambar 5 Penyediaan Server

b. Tahap pertama Tahap kedua yaitu melakukan konfigurasi pada file *inventory* yang berisi IP *Address* target *tools* yang di*install* dan *source code* file .yaml yang berisikan perintah untuk melakukan instalasi *tools* Keycloak.

```
[sso]
coofis-sso ansible_host=127.0.0.1 ansible_port=61473 ansible_user=xxxx
ansible_become_password=xxxx
```

GAMBAR 6
File Inventory

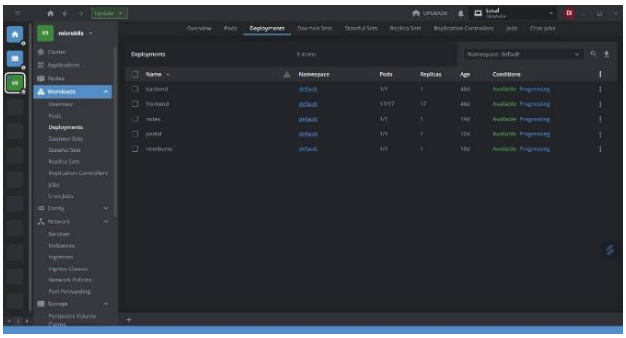
```
root@telkombg:~/ansible/roles# ls
common db kube pg reverseproxy sso storage
root@telkombg:~/ansible/roles# cd sso/
root@telkombg:~/ansible/roles/sso# ls
handlers tasks templates
```

GAMBAR 7
File Instalasi SSO

c. Tahap ketiga, membuat beberapa file konfigurasi untuk proses integrasi berbagai *services* berupa file konfigurasi .yaml yaitu deployment.yaml, ingress.yaml, service.yaml file yang digunakan dalam lingkungan Kubernetes untuk mendefinisikan dan mengatur berbagai aspek aplikasi yang akan di-*deploy*.

d. Tahap keempat, dilakukan *deployment* terhadap MicroK8s. Hal yang dilakukan pada langkah ini yaitu, mempersiapkan kubeconfig untuk diekspor dan IP *Adress* yang akan dikonfigurasikan pada LENS Kubernetes. Untuk mempersiapkan kubeconfig tersebut, command yang harus digunakan adalah "***microk8s config***".

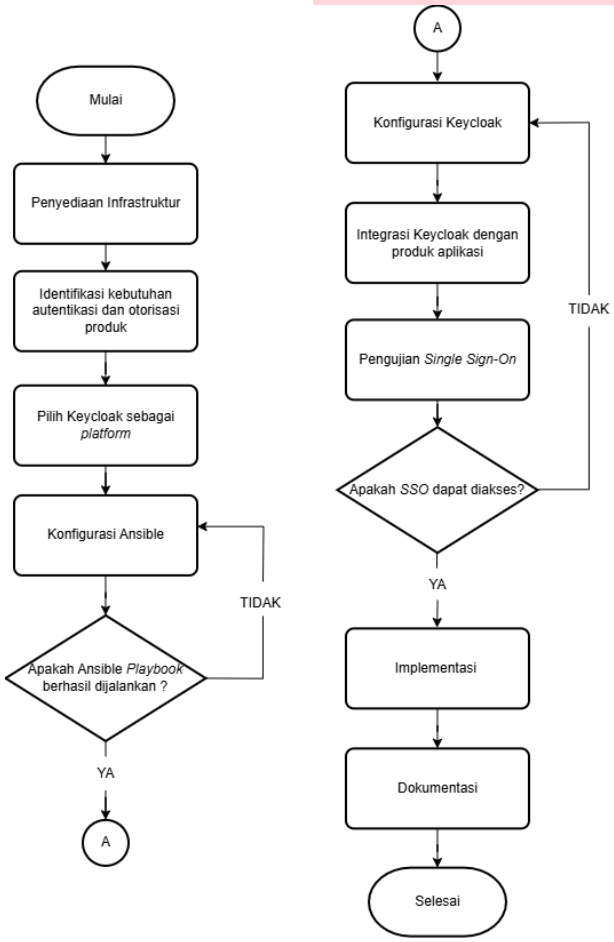
e. Tahap kelima, dilakukan konfigurasi pada LENS Kubernetes dengan menambahkan kubeconfig yang telah disiapkan pada langkah sebelumnya. Berikut adalah tampilan pada LENS jika telah memasukkan kubeconfig.



GAMBAR 8
Tampilan LENS

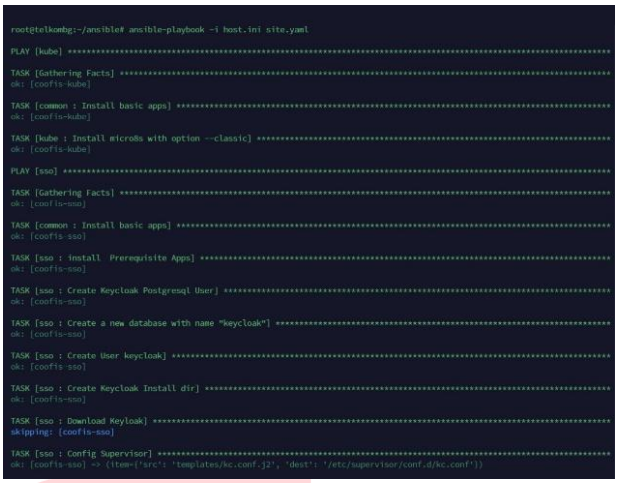
2. Operations

Tahap *operations* sebagai penyediaan infrastruktur untuk produk aplikasi dilakukan dengan mengelola dan memelihara infrastruktur yang telah disiapkan dan di-*deploy* secara berkelanjutan untuk memastikan performa yang optimal dan keandalan sistem. Diagram alir tahapan yang akan dilakukan dapat dilihat pada Gambar 9.



GAMBAR 9
Perancangan Sistem *Operations*

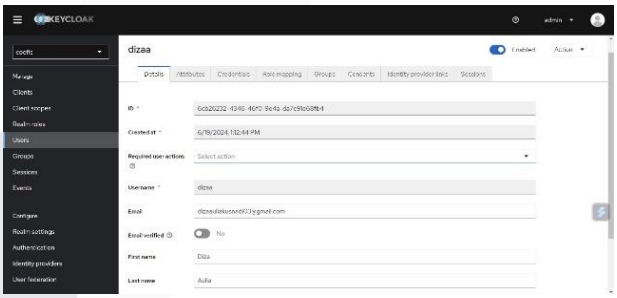
a. Tahap pertama, melakukan *running* terhadap ansible yang telah dikonfigurasi pada tahap deployment. *Running* ansible, dilakukan dengan menggunakan *command* “ansible-playbook -I <file inventory> <nama .yaml yang akan dijalankan>.



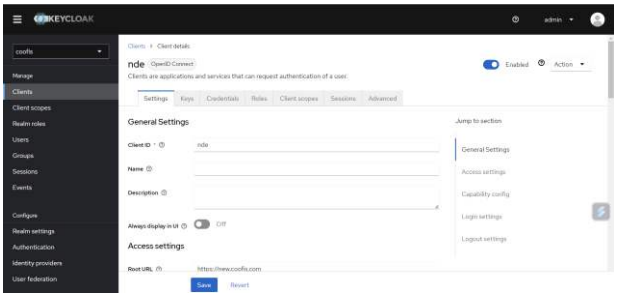
GAMBAR 10
Menjalankan *Ansible Playbook*

b. Tahap kedua, jika telah dipastikan bahwa *tools* Keycloak dapat berjalan di server, selanjutnya adalah konfigurasi pada Keycloak. Tahapan konfigurasi aplikasi klien di Keycloak, termasuk pengaturan redirect URI, jenis klien, dan izin yang diperlukan.

GAMBAR 11
Login newsso.coofis.com

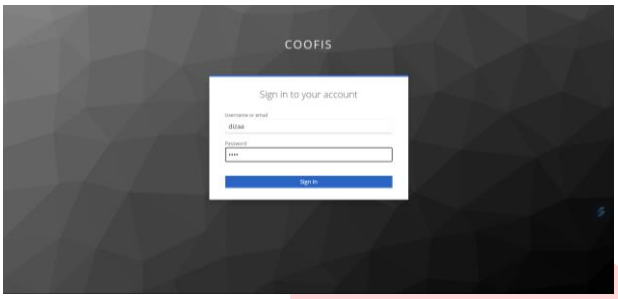


GAMBAR 12
Menambahkan *User*



GAMBAR 13
Mengatur *Client*

c. Tahap ketiga, setelah dilakukan konfigurasi dan integrasi aplikasi pada Keycloak, selanjutnya dilakukan verifikasi kredensial pengguna dengan mengakses *link* **newportal.coofis.com** terlebih dahulu. Lalu, pastikan bahwa pengguna dapat masuk menggunakan kredensial mereka (*username* dan *password*) yang terdaftar di Keycloak.

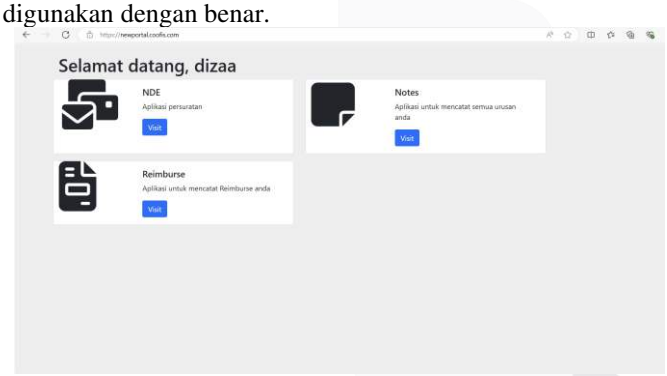


GAMBAR 14
Halaman *web login* newportal.coofis.com

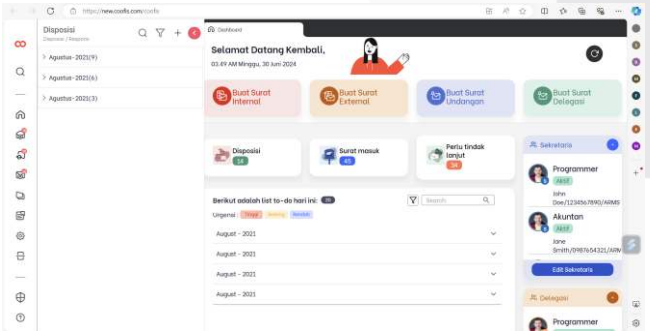
IV. HASIL DAN PEMBAHASAN

A. Hasil Pengujian Sistem *Single Sign-On*

Hasil pengujian yang akan dilakukan berupa pengujian sistem *Single Sign-On* yang menunjukkan bahwa sistem autentikasi dan otorisasi berfungsi dengan baik. Pengguna dapat *login* dan *logout* ke berbagai aplikasi melalui Keycloak dengan sukses, dan token autentikasi diterbitkan serta digunakan dengan benar.

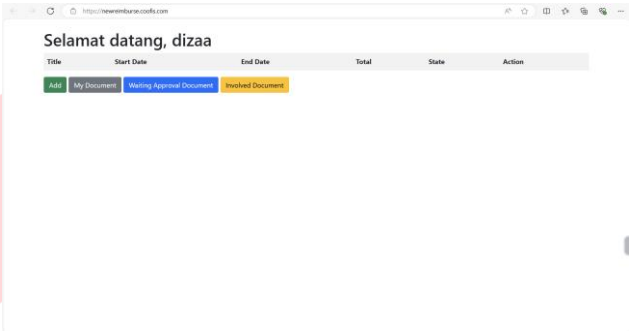


GAMBAR 15
Tampilan *web* newportal.coofis.com



GAMBAR 16
Tampilan *web* newcoofis.com

GAMBAR 17
Tampilan *web* newnotes.coofis.com



GAMBAR 18
Tampilan *web* newreimburse.coofis.com

B. Hasil Pengujian Fungsionalitas

Pengujian fungsionalitas pada penggunaan sistem *Single Sign-On* (SSO) Keycloak bertujuan untuk memverifikasi bahwa setiap fungsi sistem berkerja sesuai dengan persyaratan dan spesifikasi yang telah ditentukan. Adapun hasil pengujian sistem dapat dilihat pada Tabel 1:

TABEL 1
Pengujian Fungsionalitas

No	Skema Pengujian	Hasil yang Diharapkan	Sistem Bekerja (Ya/Tidak)
1.	Keberhasilan *Login*	Pengguna dapat login ke aplikasi menggunakan kredensial Keycloak dengan sukses. Proses login berlangsung cepat dan lancar tanpa adanya kendala teknis.	Ya
2.	Pengujian *Multi-Platform*	Pengujian dilakukan pada berbagai platform seperti laptop dan *smartphone*. SSO Keycloak bekerja konsisten di semua perangkat, memastikan pengalaman pengguna yang mulus dan seragam.	Ya
3.	Keberhasilan *Single Sign-On*	Setelah pengguna *login* di satu aplikasi, mereka dapat mengakses aplikasi lain yang terintegrasi dengan Keycloak tanpa perlu memasukkan kredensial lagi. Hal ini menunjukkan bahwa SSO berfungsi dengan baik.	Ya
4.	Keberhasilan *Single Log-out*	Pengguna dapat melakukan *logout* dari satu aplikasi dan otomatis logout dari semua aplikasi lain yang terintegrasi dengan Keycloak karena belum terdapat *button* "logout" pada aplikasi web sehingga alternatif lain dari proses *logout* adalah dengan menghapus *Session Cookies* dari browser.	Tidak

Berdasarkan pengujian fungsionalitas penggunaan sistem *Single Sign-On* (SSO) pada Tabel 2 dapat dilihat bahwa hasil pengujian menunjukkan sebagain besar fungsional sistem berjalan dengan baik sesuai dengan scenario yang diharapkan, namun terdapat satu pengujian yang tidak berjalan sesuai dengan harapan yaitu pada pengujian Keberhasilan *Single Log-out* karena belum terdapat *button* "logout" pada aplikasi web sehingga alternatif lain dari proses *logout* adalah dengan menghapus *Session Cookies* dari browser.

C. Hasil Pengujian Kinerja Sistem

Pengujian kinerja sistem pada penggunaan sistem *Single Sign-On* (SSO) Keycloak bertujuan untuk menentukan bagaimana sebuah sistem atau aplikasi berfungsi di bawah beban kerja tertentu. Pengujian kinerja sistem yang akan dilakukan pada Proyek Akhir ini berupa pengujian *throughput* dan *latency*.

TABEL 2
Hasil Pengujian Kinerja Sistem

No	Hasil Pengujian			Error
	Users	Throughput (s)	Latency (ms)	
1	100	3.26	233.986	0.00%
2	500	16.18	267.378	0.00%
3	1000	32.34	214.972	0.00%

Berdasarkan Tabel 2, ditunjukkan hasil perbandingan pengukuran kinerja sistem dengan jumlah *users* yang berbeda. Pada pengukuran *throughput* terjadi peningkatan seiring dengan meningkatnya jumlah pengguna yaitu pada pengukuran 100 *users* didapatkan nilai *throughput* senilai 3.26 req/sec, pengukuran 500 *users* didapatkan nilai *throughput* senilai 16.18 req/sec dan pada pengukuran 1000 *users* didapatkan nilai *throughput* senilai 32.34 req/sec yang menunjukkan bahwa server mampu menangani peningkatan beban dengan efisien.

Sedangkan pada pengukuran *latency,* terjadi peningkatan dengan bertambahnya pengguna pada pengujian kedua (500 *users*) dari nilai *latency* 233.986 ms (100 *users*) menjadi 267.378 ms , tetapi menurun pada pengujian ketiga (1000 *users*) dengan nilai *latency* 214.972 ms, yang menunjukkan adanya optimasi server terhadap beban yang tinggi.

V. KESIMPULAN

Berdasarkan hasil pengujian sistem *Single Sign-On* menggunakan Keycloak pada produk aplikasi Coofis Verse dapat disimpulkan bahwa implementasi Keycloak memungkinkan pengguna untuk mengakses berbagai aplikasi dengan satu set kredensial, sehingga meningkatkan kenyamanan pengguna dan efisiensi operasional.

Selama pengujian, Keycloak terbukti efektif dalam mengelola sesi pengguna, menjaga keamanan token, dan memastikan bahwa pengguna hanya dapat mengakses sumber daya yang sesuai dengan hak akses mereka.

Secara keseluruhan pada pengujian fungsionalitas sistem *Single Sign-On* (SSO) menggunakan Keycloak menunjukkan bahwa sistem ini berhasil menyediakan autentikasi yang efisien dan seragam di berbagai aplikasi. Meskipun belum terdapat fitur *logout* pada aplikasi yang diakses, solusi sementara dengan menghapus *session cookies* dari *browser* masih dapat digunakan untuk mengelola sesi pengguna.

REFERENSI

[1] K. F. Ribawanto and D. Pramono, "Pengembangan Sistem Nota Dinas Elektronik dengan Tanda Tangan Elektronik Studi Kasus PT Andal Rancang Multi Solusi (Arm Solusi)," 2022. [Online]. Available: http://j-ptiik.ub.ac.id

[2] Y. Christian and R. Bisma, "Studi Perbandingan Performa Aplikasi Web Monolitik Dan Microservice Berbasis Apache Kafka," *Journal of Informatics and Computer Science*, vol. 03, 2021.

[3] A. Suhardi, E. Fatkhiyah, and M. Sholeh, "PERANCANGAN DAN IMPLEMENTASI SSO (SINGLE SIGN ON) MENGGUNAKAN PROTOKOL OAUTH 2.0," vol. 5, no. 2, 2017, [Online]. Available: https://web.facebook.com

[4] A. Sinambela and F. Farady Coastera, "IMPLEMENTASI ARSITEKTUR MICROSERVICES PADA RANCANG BANGUN APLIKASI MARKETPLACE BERBASIS WEB," 2021. [Online]. Available: http://ejournal.unib.ac.id/index.php/rekursif/1

[5] M. R. Pelatihan PENGENALAN LINUX oleh Anton Raharja Afri Yunianto Wisesa Widyantoro Editor, "Open Source Campus Agreement."

[6] E. Erawan and M. Salman, "PENGUATAN KEAMANAN OTOMATIS PADA SISTEM OPERASI UBUNTU BERBASIS IMAGE MESIN VIRTUAL MENGGUNAKAN SOLUSI PACKER," *CAKRAWALA – Repositori IMWI*, vol. 6, no. 4, pp. 1089–1097, 2023.

[7] I. Putu, A. E. Pratama, P. B. Suarnata, and W. Putra, "Pengujian IaC Berbasis DevOps dan Ansible Menggunakan Metode Black Box Testing," vol. 15, no. 2, pp. 1979–276, 2022, doi: 10.30998/faktorexacta.vx3ix.xxxx.

[8] T. Alfiandi, T. M. Diansyah, and R. Liza, "ANSIBLE DAN SHELL SCRIPT PADA CLOUD SERVER DEPLOYMENT AWS," *JITEKH*, vol. 8, no. 2, pp. 78–84, 2020.

[9] K. A. Lutvianto, "PERANCANGAN SISTEM SINGLE SIGN ON DENGAN METODE OTENTIKASI OAUTH 2.0," Universitas Teknologi Digital Indonesia, Yogyakarta, 2019.

[10] N. Wulandari, A. Wibowo, and B. Susanto, "Penerapan RESTful API untuk Membangun Program Pembayaran Piutang Menggunakan Otentikasi OAuth 2.0," *Jurnal Terapan Teknologi Informasi*, vol. 5, no. 1, pp. 1–10, Apr. 2021, doi: 10.21460/jutei.2021.51.230.

[11] F. R. Pontoh, A. Basuki, and A. Bhawiyuga, "Pengembangan Platform Hands-on Lab untuk Manajemen Identitas dan Akses menggunakan Teknologi Virtualisasi berbasis Container," 2022. [Online]. Available: http://j-ptiik.ub.ac.id