STUDI KELAYAKAN RINCI PEMBANGUNAN LAYANAN SISTEM OTENTIKASI DI BADAN PENGELOLAAN PENDAPATAN, KEUANGAN DAN ASET DAERAH KABUPATEN BREBES

22 Februari 2019

Priyanto Tamami, S.Kom.

1 RUANG LINGKUP PEKERJAAN

Ruang lingkup dari pekerjaan membangun sebuah sistem informasi otentikasi di Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah Kabupaten Brebes yaitu membangun sebuah sistem yang mampu melayani otentikasi termasuk otorisasi di dalamnya, untuk seluruh aplikasi yang berkaitan dengan Pajak Daerah di Kabupaten Brebes, adapun aplikasi atau sistem informasi yang akan dibangun berbasis *web*, artinya untuk menggunakan aplikasi atau sistem informasi ini dibutuhkan *browser* untuk menampilkan halaman sistem informasi.

Sistem informasi yang akan dibangun akan memiliki arsitektur dimana aplikasi yang dibangun akan dipisahkan menjadi 3 (tiga) bagian, yaitu bagian *front-end* dan 2 (dua) bagian *back-end*, bagian *back-end* akan terdiri dari layanan sistem yang menyediakan otentikasi dan otorisasi itu sendiri, yang disebut *Auth Server*, dan bagian lain akan melayani data yang digunakan untuk melakukan otentikasi dan otorisasi seperti informasi pengguna, informasi hak akses, dan informasi mengenai aplikasi klien yang dapat terhubung. Bagian *front-end* dan bagian *back-end* akan berkomunikasi melalui cara *web service* dengan model REST. Data yang dikirim dari peladen *web service* berupa JSON.

Pemilihan arsitektur ini karena nantinya aplikasi klien yang dapat terhubung

dan menggunakan fasilitas otorisasi dan otentikasi bukan hanya aplikasi yang berbasis *web*, namun termasuk juga aplikasi *mobile* yang terpasang dalam ponsel pintar dan aplikasi yang berbasis *desktop*.

Komunikasi data yang sifatnya universal dan mampu menghubungkan ketiga perangkat tersebut, yaitu peladen, klien berbasis aplikasi *mobile* dan aplikasi berbasis *desktop* adalah menggunakan arsitektur REST.

Bagian *front-end* nantinya akan memiliki fungsi sebagai tampilan tatap muka untuk melakukan konfigurasi untuk tiap pengguna dan konfigurasi untuk tiap klien yang nantinya akan dihubungkan. Bagian *front-end* nantinya akan terhubung ke 2 (dua) layanan di bagian *back-end*, layanan yang pertama (yang disebut dengan *OAuth Server*) tentu saja akan melakukan otorisasi dan otentikasi terhadap pengguna yang akan melakukan akses, layanan ini hanya menyediakan halaman masuk (*login*) saja, yang kemudian, setiap aplikasi klien akan mendapatkan sebuah token yang digunakan untuk melakukan akses ke layanan yang kedua (yang diistilahkan sebagai *resource server*). *Resource server* inilah yang nantinya akan berkomunikasi dengan bagian *front-end* (diistilahkan juga sebagai aplikasi klien) dalam penyediaan datanya.

Untuk memastikan bahwa layanan pada *Resource server* terlindungi, tiap permintaan (*request*) yang dilakukan oleh aplikasi klien harus menyertakan token yang didapat dari proses otentikasi dengan *OAuth Server*, nantinya token yang diterima oleh *Resource server* dari aplikasi klien akan diverifikasi dengan melakukan verifikasi token.

Cara yang dilakukan *Resource server* untuk memverifikasi token dari aplikasi klien adalah dengan melakukan dekripsi token menggunakan *public key* yang disediakan oleh *OAuth Server* yang dapat diakses secara bebas.

Token yang diimplementasikan sebagai tanda pengenal bahwa pengguna memiliki hak akses tertentu akan berbentuk JWT (*JSON Web Token*) yang dijabarkan

pada dokumen IETF (*Internet Engineering Task Force*) dengan nomor RFC (*Request for Comment*) 7519 yang telah diperbarui dengan dokumen IETF dengan nomor RFC 7797.

Sebagaimana dijelaskan dalam dokumen tersebut (IETF RFC No. 7519), bahwa JWT sebetulnya hanyalah sebuah teks (*string*) yang mewakili beberapa pernyataan dari objek JSON yang terkode. Pernyataan dalam objek JSON sendiri adalah pasangan antara *key* dan *value*, dimana *key* harus berupa teks (*string*) dan *value* dapat berupa nilai apapun yang dapat didefinisikan dalam JSON.

JSON sendiri sebetulnya adalah singkatan dari *JavaScript Object Notation*, sebagaimana didefinisikan pada dokumen IETF (*Internet Engineering Task Force*) dengan nomor RFC (*Request for Comment*) 8259, bahwa JSON adalah sebuah format pertukaran data yang ringan, berbasis teks, dan independen terhadap bahasa pemrograman.

JSON sendiri dapat merepresentasikan 4 (empat) tipe data primitif seperti *string*, *number*, *boolean*, dan *null*, serta 2 (dua) tipe struktur, yaitu objek dan larik (*array*).

String sendiri adalah barisan dari 0 (nol) atau lebih karakter *Unicode* yang tertuang pada laman http://www.unicode.org/versions/latest/. Harap dicatat bahwa petikan tersebut mengacu pada versi paling akhir dari *Unicode* tanpa menyebutkan secara spesifik versinya. Karena hal ini, sehingga tidak menjamin bahwa perubahan yang terjadi pada spesifikasi *Unicode* dimasa mendatang akan memiliki pengaruh terhadap sintak dari JSON.

Number tentu saja adalah sebuah angka, baik bilangan bulat maupun pecahan, dan *boolean* adalah tipe data primitif yang terdiri dari hanya 2 (dua) nilai, yaitu *true* atau *false*, sedangkan *null* adalah tipe data yang menyatakan bahwa nilainya adalah kosong, atau data tanpa nilai.

Struktur Objek di JSON adalah himpunan yang tidak terurut yang terdiri dari

0 (nol) atau lebih pasangan *name* dan *value*, dimana *name* adalah sebuah *string* dan *value* dapat berupa *string*, *number*, *boolean*, *null*, *objek*, atau larik (*array*).

Sedangkan struktur *array* atau larik di JSON adalah barisan berurut yang terdiri dari 0 (nol) atau lebih nilai.

Bagian *front-end* nantinya akan dibangun menggunakan *framework* Angular, dan kedua bagian *back-end*, baik *OAuth Service* dan *Resource Service* akan dibangun menggunakan Kotlin dan Springboot.

Hal-hal yang mendukung studi kelayakan dibangunnya sistem otentikasi dan otorisasi di Badan Pengelolaan Pendapatan, Keuangan, dan Aset Daerah ini yaitu dengan menggunakan metode analisis kelayakan TELOS (*Technical*, *Economic*, *Legal*, *Operational*, *Schedule*).

Detail dari masing-masing faktor pada metode analisis kelayakan TELOS ini adalah seperti berikut :

1. Faktor Kelayakan Teknis (*Technical*)

   Kelayakan teknis menyoroti kebutuhan sistem yang telah disusun dari aspek teknologi yang digunakan, jika teknologi yang dikehendaki untuk pengembangan sistem merupakan teknologi yang mudah didapat, murah, dan tingkat pemakaiannya mudah, maka secara teknis usulan kebutuhan sistem bisa dinyatakan layak

2. Faktor Kelayakan Ekonomi (*Economic*)

   Aspek yang paling dominan dari aspek kelayakan yang lain adalah kelayakan ekonomi. Tidak dapat disangkal lagi, motivasi pengembangan sistem informasi pada perusahaan atau organisasi adalah motif keuntungan. Dengan demikian aspek untung rugi jadi pertimbangan utama dalam pengembangan sistem. Kelayakan ekonomi berhubungan dengan *return investmen* atau berapa lama biaya investasi dapat kembali, namun pada instansi pemerintah,

yang secara umum bertugas melayani masyarakat, maka hubungannya bukan lagi untung rugi, tetapi lebih kepada imbas administrasi data yang tercatat dengan baik dan sedikit banyak mungkin berpengaruh pada peningkatan kepatuhan wajib pajak dalam memenuhi kewajiban pajaknya.

3. Faktor Kelayakan Hukum (*Legal*)

   Menguraikan secara hukum apakah sistem informasi yang akan dikembangkan tidak menyimpang dari hukum yang berlaku (tidak melanggar hukum jika diterapkan di objek penelitian). Misal: bagaimana kelayakan perangkat lunak yang digunakan, bagaimana kelayakan hukum informasi yang dihasilkan oleh program aplikasi yang dibuat. Apakah melanggar hukum atau tidak.

4. Faktor Kelayakan Operasional (*Operational*)

   Penilaian terhadap kelayakan operasional digunakan untuk mengukur apakah sistem yang akan dikembangkan nantinya dapat dioperasikan dengan baik atau tidak untuk melayani kebutuhan otentikasi dan otorisasi.

5. Faktor Kelayakan Jadwal (*Schedule*)

   Penilaian kelayakan jadwal ini digunakan untuk menentukan bahwa pengembangan sistem akan dapat dilakukan dalam batas waktu yang telah ditetapkan.

Adapun penilaian untuk masing-masing faktor kelayakan TELOS ini adalah sebagai berikut :

1. Menilai kelayakan Teknik (*Technical*)

   Kelayakan teknik menyoroti kebutuhan sistem yang telah disusun dari teknologi yang akan digunakan, untuk penerapan sistem otentikasi di Badan

Pengelolaan Pendapatan, Keuangan dan Aset Daerah. Sistem ini merupakan sistem berbasis layanan otentikasi yang digunakan oleh aplikasi yang akan dibangun yang datanya menggunakan data-data Pajak Daerah dan akses datanya terbatas untuk tiap jabatan pengguna, sehingga di masa mendatang, aplikasi yang dibangun tidak memerlukan bagian otentikasi, proses otentikasi cukup diserahkan ke sistem OAuth yang akan dibangun ini.

Sistem ini nantinya akan menerima permintaan sebuah token dari aplikasi klien, yang apabila belum pernah masuk sama sekali, sistem akan menampilkan halaman masuk (*login*) untuk memastikan bahwa pengguna (*end-user*) sudah terdaftar dan memiliki hak untuk melakukan akses data. Apabila proses *login* telah berhasil, maka aplikasi klien akan mendapatkan sebuah token yang dapat digunakan untuk melakukan akses data ke *resource server*.

Token yang diberikan oleh *OAuth server* berbentuk JWT (*JSON Web Token*) yang akan digunakan oleh aplikasi klien melakukan akses ke *resource server*. Token dalam bentuk JWT ini sebetulnya sudah memiliki informasi yang cukup bagi aplikasi klien melakukan pengaturan akses bagi pengguna yang sedang melakukannya, tentunya aplikasi klien perlu melakukan *decode* terlebih dahulu pada JWT yang diterima dari *OAuth Server*.

Token dalam bentuk JWT ini wajib selalu dikirimkan oleh aplikasi klien pada saat melakukan akses ke *resource server*, hal ini untuk memastikan bahwa data yang berada di *resource server* diakses oleh pengguna yang memiliki hak.

(a) Kebutuhan Perangkat Keras, Perangkat Lunak, dan Perangkat Jaringan

i. Perangkat Keras

Kebutuhan akan perangkat keras, sebetulnya tidak diperlukan pe-

ngadaan alat baru, melainkan dengan menggunakan perangkat yang sudah ada, sistem informasi ini akan dapat berjalan sebagaimana mestinya. Adapun perangkat keras yang digunakan adalah seperti berikut :

No	Perangkat keras	Keterangan
Server Aplikasi		
1	Prosesor	Intel Xeon 2,4G 8 Core
2	Memori	8 GB
3	Harddisk	
4	Jaringan	4 ethernet slot
Server Basis Data		
1	Prosesor	Intel Xeon 2,4G 8 Core
2	Memori	32 GB
3	Harddisk	
4	Jaringan	4 ethernet slot

ii. Perangkat Lunak

Kebutuhan akan perangkat lunak baru tentunya ada, tetapi tidak memerlukan biaya tambahan, karena beberapa perangkat lunak ada yang sudah terpasang, dan perangkat lunak yang akan di*install* tidak memerlukan biaya tambahan.

No	Perangkat Lunak	Kegunaan
1	Windows Server 2008	Sistem Operasi
2	Linux Ubuntu	Sistem Operasi
3	IntelliJ IDEA	IDE
4	Visual Studio Code	Editor
5	Docker	Container
6	Chrome, Firefox	Web Browser

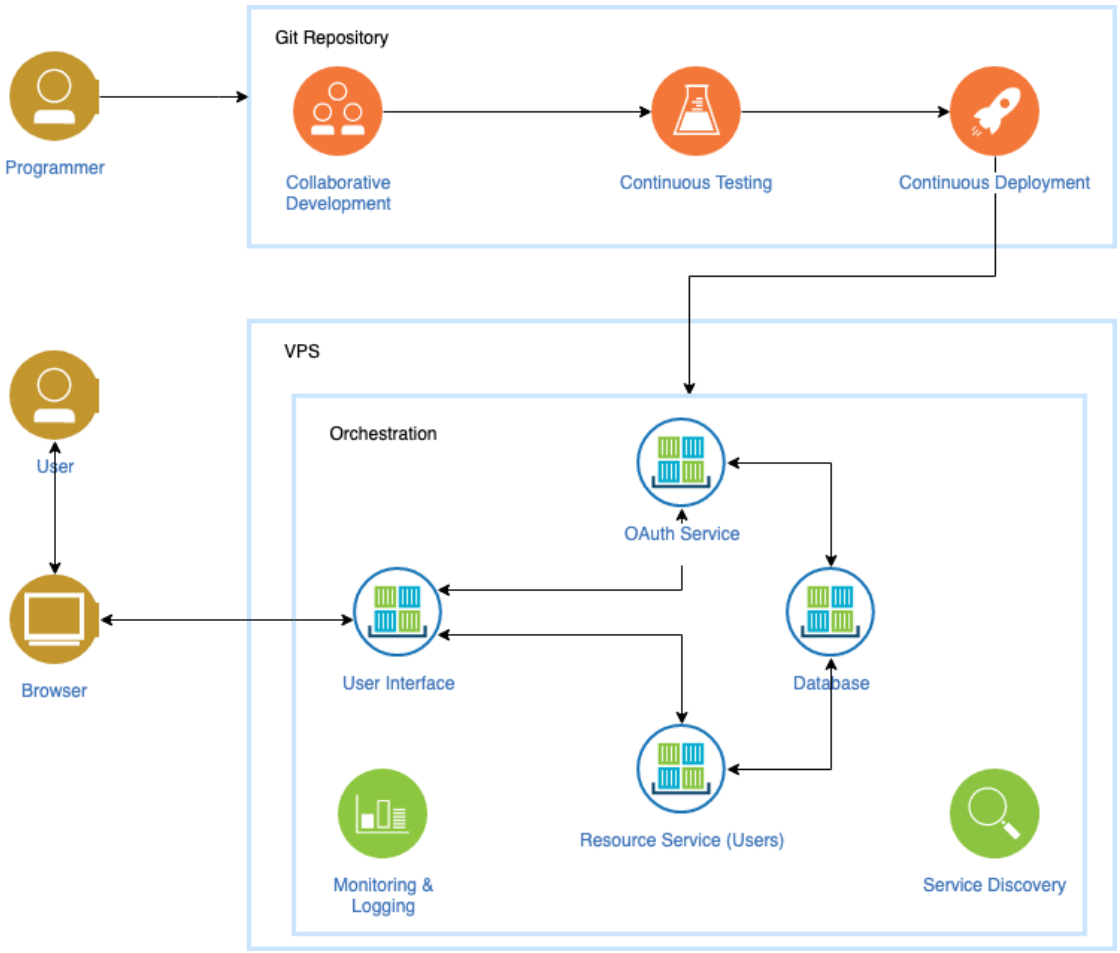
iii. Perangkat Jaringan

Kebutuhan akan perangkat jaringan pun tidak memerlukan pengadaan kembali, karena kondisi jaringan sudah terhubung dengan baik dengan peralatan seperti berikut :

No	Perangkat	Kegunaan
1	Switch	Penghubung antar kabel jaringan
2	Kabel UTP	Media penghubung antar perangkat
3	Konektor RJ-45	Media penghubung antara kabel dengan *socket* perangkat
4	Modem	Sebagai perangkat penghubung jaringan lokal dan internet
5	Router	Penghubung antar 2 atau lebih jaringan (berdasarkan alamat jaringan)

(b) Arsitektur Jaringan Komputer

Arsitektur jaringan sistem informasi ini adalah seperti pada gambar 1. Bila ada seorang pengguna yang melakukan akses dengan *browser*, maka akan ditampilkan halaman utama dari sistem informasi ini.



Gambar 1: Desain Arsitektur Sistem

Nantinya, saat pengguna akan melakukan akses terhadap beberapa aplikasi yang terproteksi, pengguna akan diarahkan ke halaman *login* milik *OAuth Service* untuk mengisikan *username* dan *password*.

Setelah proses otentikasi berhasil dilakukan, selanjutnya halaman pada peramban (*browser*) akan dikembalikan ke halaman aplikasi yang akan diakses, dimana proses yang berjalan dibelakang layar adalah *OAuth Service* akan memberikan sebuah *token* ke aplikasi klien untuk disimpan dan digunakan setiap akan melakukan proses permintaan data (*request*)

ke *Resource Service*.

Selanjutnya, saat pengguna akan melakukan akses data melalui halaman aplikasi *web* klien, aplikasi ini akan melakukan permintaan (*request*) data ke *resource service* dengan membawa data dan *token*, yang kemudian *resource service* akan melakukan verifikasi hak akses melalui *token* tersebut, apabila hak akses tidak sesuai, maka akan mengembalikan pesan kesalahan bahwa permintaan atau *request* tersebut tidak dapat dipenuhi, namun bila hak aksesnya sesuai, maka data akan dikirimkan ke aplikasi klien untuk ditampilkan ke pengguna.

Adapun kondisi sistem informasi yang digunakan saat ini adalah seperti berikut :

(a) Sistem Aplikasi

Aplikasi adalah sebuah *tools* untuk membantu dan mempermudah suatu aktifitas administrasi maupun yang lainnya. Tanpa adanya aplikasi, tentu suatu aktifitas tidak bisa diselesaikan dengan cepat, segala sesuatunya dilakukan dengan pencatatan manual. Aplikasi atau sistem informasi yang digunakan di Bidang Pendataan dan Penetapan pada Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah Kabupaten Brebes saat ini adalah :

No	Aplikasi	Keterangan
1	SISMIOP	Melakukan manajemen objek pajak PBB-P2
2	Simda Pendapatan	Melakukan pengelolaan pajak daerah
3	Microsoft Excel	Digunakan untuk merekam dan mengolah data peralihan hak

Dengan adanya sistem informasi atau aplikasi SISMIOP, sudah membantu pengelolaan pajak daerah untuk jenis Pajak Bumi dan Bangunan

sektor Perdesaan dan Perkotaan dari mulai manajemen pelayanan, pendataan, penetapan, penagihan, hingga ke prosedur pembayaran.

Sedangkan aplikasi Simda Pendapatan, sudah dapat membantu melakukan pengelolaan pajak daerah untuk jenis pajak selain Pajak Bumi dan Bangunan sektor Perdesaan dan Perkotaan, namun belum sepenuhnya dapat melakukan pengelolaan yang terjadi pada jenis Bea Perolehan Hak Atas Tanah dan Bangunan.

Dengan semakin bertambahnya kebutuhan organisasi akan aplikasi atau sistem informasi kedepan, maka akan semakin bertambah pula otentikasi dan otorisasi untuk tiap penggunanya, untuk mengantisipasi kondisi seperti ini, maka kebutuhan untuk membangun sebuah sistem OAuth diperlukan sehingga nantinya tiap pengguna tidak perlu melakukan *login* berulang-ulang untuk melakukan akses ke beberapa aplikasi yang terproteksi.

(b) Sistem Basis Data

Basis data merupakan suatu tempat penyimpanan data-data yang biasanya dari sebuah aplikasi, adapun basis data yang sudah ada pada Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah adalah seperti berikut :

No	Basis data	Keterangan
1	Oracle DB	Digunakan untuk menyimpan data dari aplikasi SISMIOP
2	SQL Server	Digunakan untuk menyimpan data dari aplikasi Simda Pendapatan

Basis data yang digunakan masih belum terintegrasi dengan baik, karena penerimaan pembayaran pada SISMIOP belum dapat secara otomatis tercatat pada Simda Pendapatan.

Sebetulnya sumber data yang digunakan pada pengelolaan Pajak Bumi dan Bangunan sektor Perdesaan dan Perkotaan ada pada 2 (dua) tempat, yang pertama berada di Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah Kabupaten Brebes, yang kedua berada di BPD Jateng.

Kondisi data yang berada di Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah berfungsi sebagai pendataan dan penetapan, namun realisasi pembayaran menggunakan data dari BPD Jawa Tengah. Selisih nilai antara 2 (dua) sumber data ini pasti terjadi, maka seharusnya dibutuhkan personil untuk melakukan pencocokan / rekonsiliasi data antara 2 (dua) sumber data tersebut agar data yang ditampilkan pada sistem informasi yang akan dibangun menjadi lebih lengkap.

Untuk membangun sistem OAuth nantinya membutuhkan sistem basis data yang tentunya terpisah dari sistem yang telah ada, pengunaan sistem basis datanya pun tidak memerlukan biaya tambahan karena menggunakan sistem basis data yang gratis dan sudah teruji kehandalannya. Basis data ini nantinya akan merekam daftar pengguna dan kata kuncinya (*password*) termasuk statusnya apakah sebagai pengguna yang statusnya aktif, atau pengguna yang statusnya di non-aktifkan, yang artinya tidak dapat melakukan akses masuk ke data-data yang terproteksi.

Pada sistem basis data yang digunakan OAuth ini pun akan merekam data aplikasi klien yang terdaftar yang dijinkan melakukan akses ke *resource server* yang terproteksi.

Dilihat dari sisi teknis sistem basis data yang akan digunakan cukup handal, terbukti dengan banyaknya perusahaan besar menggunakan basis data ini dan tersedia secara gratis, maka syarat kebutuhannya sangat

mencukupi untuk membangun sistem OAuth.

(c) Infrastruktur

Infrastruktur merupakan sarana yang sangat dibutuhkan dalam sebuah aktifitas apapun termasuk aktifitas pengelolaan Pajak Daerah. Adapun infrastruktur yang ada pada Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah Bidang Pendataan dan Penetapan yaitu :

No	Infrastruktur	Keterangan
1	Komputer	Memiliki 13 komputer dan 4 *server*
2	Printer	Memiliki 4 *printer multi-purpose*, 4 *printer dot-matrix*, 3 *printer line-matrix*
3	Jaringan Internet	Memiliki 2 (dua) sumber internet, dengan 1 (satu) internet *shared bandwidth* dan 1 (satu) internet *dedicated*

Dari kondisi infrastruktur di atas, sebetulnya Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah secara infrastruktur, syarat kebutuhannya sangat mencukupi untuk membangun sistem OAuth yang dapat digunakan sebagai pendukung aplikasi yang akan dibangun kedepannya.

Melihat kondisi teknis demikian, dimana sistem belum pernah dibuat, teknologi yang digunakan terbilang cukup baru, sehingga nilai kelayakan teknik yang dapat diberikan adalah 9.0.

2. Menilai kelayakan Ekonomi (*Economic*)

Pembangunan sistem baru tentunya membutuhkan investasi ataupun dana yang tidak sedikit, untuk mendapatkan manfaat dimasa yang akan datang, sumber daya dan sumber dana diperlukan dalam pembangunan sistem baru sebagai bentuk investasi.

Untuk menganalisis kelayakan ekonomi digunakan kalkulasi analisis biaya dan manfaat (*cost benefit analysis*). Adapun tujuan dari analisis biaya dan manfaat adalah untuk memberikan gambaran kepada internal Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah, apakah manfaat yang diperoleh dari sistem baru "lebih besar" dibandingkan dengan biaya yang dikeluarkan.

Pada analisis biaya dan manfaat ada beberapa metode kuantitatif yang digunakan untuk menemukan standar kelayakan proyek.

Analisis Biaya dan Manfaat

Untuk melakukan analisa biaya dan manfaat diperlukan dua komponen, yaitu komponen biaya dan komponen manfaat.

(a) Komponen Biaya

Biaya yang berhubungan dengan pembuatan sistem ini dapat diklasifikasikan ke dalam 3 (tiga) kategori utama, yaitu :

i. Biaya pengadaan, yaitu biaya pembelian perangkat keras, yang digunakan pada awal pembuatan sistem, sebelum sistem dioperasikan.

ii. Biaya Pengembangan, yaitu biaya pembuatan perangkat lunak sistem yang meliputi biaya konsultasi, biaya tahap analisa sistem, biaya tahap desain sistem dan biaya tahap penerapan sistem.

iii. Biaya operasi dan biaya perawatan, yaitu biaya yang dikeluarkan untuk menjalankan sistem, yaitu biaya *overhead*, biaya perawatan terhadap perangkat keras dan perangkat lunak.

(b) Komponen Manfaat.

Manfaat yang didapat dari sistem informasi diklasifikasi sebagai berikut :

i. Keuntungan berwujud adalah keuntungan yang berupa penghematan atau peningkatan di dalam pengelolaan pajak daerah yang dapat diukur dalam bentuk satuan nilai uang. Keuntungan berwujud antara lain :

   A. Pengurangan biaya cetak
   B. Pengurangan biaya operasi
   C. Pengurangan biaya perlengkapan

ii. Keuntungan tak berwujud, adalah keuntungan yang sulit atau tidak mungkin diukur dalam bentuk satuan uang. Keuntungan tersebut antara lain :

   A. Peningkatan efisiensi waktu pengerjaan
   B. Peningkatan efektifitas pegawai
   C. Peningkatan pelayanan pajak daerah

Adapun metode untuk melakukan analisis biaya dan manfaat adalah :

i. Metode Periode Pengembalian

   Metode ini adalah uji kuantitatif yang digunakan untuk menghitung jangka waktu yang diperlukan untuk membayar kembali biaya investasi dalam pembuatan aplikasi yang telah dikeluarkan. Penilaian kelayakan untuk pengembalian

$$periode = \frac{investasi}{proceed} xtahun$$

   A. Layak jika waktu pengembalian lebih kecil dari umur investasi
   B. Tidak layak jika waktu pengembalian lebih besar dari umur investasi.

   Perhitungan Periode Pengembalian adalah seperti berikut :
   Nilai Investasi = Rp0,- Proses Th 1 = Rp100.364.100.000,-

Nilai investasi memang bernilai 0 (nol) rupiah karena seluruh sarana dan prasarana telah tersedia, hanya tinggal membangun sebuah sistem informasi untuk digunakan dan dijalankan. Sedangkan nilai Rp100.364.100.000,- (Seratus milyar tiga ratus enam puluh empat juta seratus ribu) didapat dari nilai target penerimaan untuk seluruh jenis Pajak Daerah di Kabupaten Brebes.

$$PP = \frac{0}{100.364.100.000}$$

$$PP = 0tahun$$

Artinya, karena nilai investasi yang dikeluarkan nihil sama sekali, atau dengan kata lain tidak memerlukan nilai investasi, namun sistem informasi secara tidak langsung memberikan andil terhadap realisasi sebesar kurang lebih 100 Milyar Rupiah kepada Kas Daerah. Artinya sistem ini sangat layak untuk dikembangkan karena pengembalian tidak membutuhkan waktu untuk mencapai titik impas.

ii. Metode Pengembalian Investasi

Metode pengembalian investasi digunakan untuk mengukur presentase manfaat yang dihasilkan proyek dibandingkan dengan biaya yang dikeluarkan.

Return On Investmen (ROI) dari suatu proyek dapat dihitung dengan rumus penilaian kelayakan untuk ROI :

A. Layak jika ROI > 0

B. Tidak layak jika ROI < 0

$$ROI = \frac{TotalManfaat - TotalBiaya}{TotalBiaya} x100\%$$

Karena nilai biaya tidak akan pernah muncul, maka bisa disebut bahwa Total Biaya = 0 (nol), Total Manfaat adalah penerimaan realisasi untuk seluruh jenis pajak daerah dalam satu tahun pajak, yaitu senilai 100.364.100.000 rupiah. Maka hasil yang didapatkan adalah seperti berikut :

$$ROI = \frac{100.364.100.000 - 0}{0} x100\%$$

$$ROI = \infty$$

Ternyata karena kondisi pembaginya adalah 0 (nol), maka hasil yang didapatkan menghasilkan Tak Terhingga. Ini dikarenakan bilangan pembagi adalah 0 (nol) sehingga menghasilkan nilai demikian, artinya proyek apapun itu bila tidak memerlukan biaya sama sekali, namun bisa mendapatkan manfaat maksimal tentunya ini proyek dengan ROI tak terhingga.

Jadi, karena nilai ∞ itu lebih dari 0 (nol) maka proyek ini layak untuk dikerjakan.

iii. Metode Nilai Sekarang Bersih

Metode nilai sekarang bersih merupakan metode yang memperhatikan nilai waktu dari uang. Suku bunga diskonto mempengaruhi arus dari uangnya (*proceed*). Metode Nilai Sekarang Bersih atau lebih dikenal dengan *Net Present Value* (NPV) dapat dihitung dari selisih nilai proyek pada awal tahun dikurangi dengan *proceed* tiap tahun yang dinilai uangkan ketahun awal dengan tingkat bunga diskonto.

Kriteria NPV ini adalah seperti berikut :

- NPV > 0 : *Feasible*

- NPV $= 0$: *Indifferent*
- NPV < 0 : *Unfeasible*

Rumus untuk menghitung NPV ini adalah seperti berikut :

$$NPV = -NilaiProyek + \frac{proceed1}{(1+i)^1} + \frac{proceed2}{(1+i)^2} + \frac{proceedn}{(1+i)^n}$$

dimana :

- i adalah tingkat bunga diskonto diperhitungkan
- n adalah umur proyek investasi
- proceed adalah selisih biaya dan manfaat

Sehingga perhitungannya dapat kita cari seperti berikut :

$$NPV = -0 + \frac{100.364.100.000}{(1+6,0\%)^1}$$

$$NPV = -0 + \frac{100.364.100.000}{1,06}$$

$$NPV = 94.683.113.207,55$$

Pada perhitungan di atas, nilai waktu dari bunga uang yang ditanamkan adalah 6,0% berdasarkan suku bunga dari www.bi.go.id pada tanggal 17 Januari 2019. Karena nilai NPV > 0 berarti investasi dapat diterima dan layak untuk dikembangkan.

Dengan melihat ketiga metode untuk kelayakan ekonomi maka dapat dikatakan bahwa sistem informasi yang akan dibangun layak dan dapat dikerjakan sebagaimana mestinya. Nilai yang diberikan untuk kelayakan ekonomi ini adalah 9,0.

3. Menilai kelayakan Hukum (*Legal*)

   Kelayakan hukum adalah kelayakan yang berkaitan dengan legalitas atau kekuatan hukum. Berarti bahwa sistem informasi yang akan dibangun tidak boleh melanggar hukum yang berlaku, baik hukum yang ditetapkan oleh pemerintah maupun hukum yang ditetapkan berdasarkan peraturan-peraturan organisasi.

   Proyek pembangunan sistem informasi yang akan dikembangkan secara hukum dinilai layak karena perangkat lunak (*software*) yang digunakan resmi sesuai dengan perijinan yang ada, karena sebagian besar menggunakan perangkat lunak yang bersifat *free* atau gratis.

   Adapun rincian perangkat lunak secara hukum adalah sebagai berikut :

No	*Software*	Lisensi
1	Windows Server 2008 R2	Lisensi tersedia
2	Linux Ubuntu	Gratis
3	Intellij IDEA Community Edition	Gratis, Apache 2.0
4	Visual Studio Code	Gratis, MIT
5	Docker	Gratis, Apache 2.0
6	Git	Gratis, LGPLv2.1

   Karena sistem informasi yang akan dibangun bertujuan untuk melakukan proteksi terhadap data-data sensitif milik Pemerintah Daerah, dalam hal ini adalah data-data yang berkaitan dengan data diri wajib pajak untuk kegiatan preventif penyalah gunaan data, maka nilai untuk kelayakan hukum dari dibangunnya sistem OAuth ini adalah 9,0.

4. Menilai kelayakan Operasional (*Operational*)

Kelayakan operasional dinilai dengan menggunakan kerangka kerja PIECES yang merupakan singkatan dari *Performance*, *Information*, *Economy*, *Control*, *Efficiency*, dan *Services*. Kerangka kerja PIECES ini dikembangkan oleh James Wetherbe yang bertujuan mengukur apakah sistem yang akan dikembangkan dapat dioperasikan dengan baik atau tidak di dalam organisasi.

Penjelasan untuk masing-masing poin dalam kerangka kerja PIECES ini adalah seperti berikut ini :

- *Performance* atau kinerja adalah untuk mengetahui apakah sistem menyediakan *throughput* dan *response time* yang cukup. Perbandingannya adalah seperti berikut ini :

Sistem Lama	Sistem Baru
Waktu yang dibutuhkan untuk mengembangkan beberapa aplikasi yang memerlukan otentikasi akan selalu berulang menghadapi fase untuk membangun sebuah halaman otentikasi / *login*	Waktu yang dibutuhkan untuk membangun beberapa sistem aplikasi yang membutuhkan otentikasi menjadi lebih singkat karena fase untuk membangun halaman otentikasi sudah tidak diperlukan, hanya melakukan konfigurasi pendaftaran aplikasi klien pada *OAuth Service*, sementara nama penggunanya, cukup dilakukan konfigurasi tambahan tanpa perlu mendaftarkannya dari awal.

Dari sisi *performance* atau kinerja tentu saja layak untuk menjadi pertimbangan dibangunnya sistem ini karena menghemat waktu dan tena-

ga pada fase membangun sistem otentikasi yang membutuhkan otentikasi kedepannya, fase ini dapat kita lewati tanpa mengurangi pembatasan akses terhadap beberapa pengguna yang terdaftar sesuai dengan kewenangannya pada aplikasi klien.

- *Information* atau informasi adalah untuk mengetahui apakah sistem menyediakan informasi yang berkualitas bagi pengguna.

Sistem Lama	Sistem Baru
Bila ingin melihat informasi mengenai status dan hak akses pengguna, karena masing-masing aplikasi memiliki daftar nama pengguna dan masing-masing hak aksesnya, maka seorang administrator yang akan melihat informasi daftar nama pengguna dan hak aksesnya perlu melakukan akses terhadap aplikasi yang sedang digunakan, apabila ingin melihat daftar pengguna dan hak akses diaplikasi lainnya, maka perlu melakukan akses ke aplikasi tersebut	Bila ingin mendapatkan informasi mengenai daftar nama pengguna serta status dan hak aksesnya, maka cukup melakukan akses ke satu aplikasi saja, informasi yang disajikan akan lengkap dari daftar nama pengguna yang sudah ada, berikut hak akses pada masing-masing aplikasi klien yang juga dapat dilihat daftarnya di aplikasi yang sama

Dari sisi *Information* atau informasi, sistem ini layak dibangun karena akan mempermudah pekerjaan yang ditugaskan kepada seorang administrator sistem, personil ini tidak perlu melakukan akses ke seluruh aplikasi apabila ada perubahan struktur organisasi, cukup melakukan akses ke satu aplikasi *OAuth Service* ini, kemudian melakukan perubahan nama pengguna atau hak aksesnya untuk seluruh aplikasi klien yang terdaftar.

- *Economy* atau ekonomi adalah untuk mengetahui apakah sistem menawarkan tingkat dan kapasitas pelayanan yang memadai untuk mengurangi biaya dan meningkatkan keuntungan. Berikut adalah perbandingan dari sistem lama dan baru :

Sistem Lama	Sistem Baru
Biaya yang dikeluarkan lebih besar karena daftar nama pengguna serta hak aksesnya akan tersimpan dalam setiap basis data yang digunakan oleh masing-masing aplikasi, tentunya akan mempengaruhi penggunaan ruang pada *storage* atau *harddisk* yang lebih besar	Biaya yang dikeluarkan relatif rendah, karena daftar nama pengguna serta hak aksesnya berada dalam satu ruang basis data untuk seluruh aplikasi yang dibangun, tentunya karena daftar pengguna berikut hak aksesnya berada dalam satu basis data, maka penggunaan ruang pada *storage* atau *harddisk* akan lebih kecil.

  Dari sisi *economy* atau ekonomi, sistem ini pun layak dibangun karena akan melakukan penghematan tanpa mengurangi fiturnya sebagai pengaman aplikasi, karena seluruh daftar pengguna dan hak aksesnya, serta daftar aplikasi klien yang dapat terhubung berada dalam satu sistem basis data saja, ini akan mempengaruhi penggunaan yang lebih hemat dari perangkat *storage* atau *harddisk*.

- *Control* atau pengendalian adalah untuk mengetahui apakah sistem informasi yang dibangun menawarkan kontrol atau pengendalian untuk mengatasi kecurangan-kecurangan dan untuk menjamin keakuratan dan keamanan data. Berikut adalah perbandingan sistem yang lama dan sistem yang baru :

Sistem Lama	Sistem Baru
Data hanya dapat diakses oleh pengguna yang memiliki hak atas sistem tersebut karena masing-masing aplikasi memiliki daftar pengguna beserta hak aksesnya, namun di beberapa sistem lama seperti SISMIOP, saat ada pengguna atau person yang melakukan akses terhadap sistem basis datanya secara langsung melalui SQL (*Structured Query Language*), maka kata kunci untuk masing-masing pengguna akan terlihat jelas apa adanya dan rawan untuk disalahgunakan oleh pihak yang tidak bertanggung jawab	Sistem yang akan dibangun pun tidak akan mengurangi fungsinya sebagai pengaman untuk melakukan proteksi terhadap data yang terrekam, masing-masing pengguna yang terdaftar nantinya akan mendapatkan hak akses sesuai dengan kewenangannya pada organisasi, apabila ada pengguna atau person yang melakukan akses langsung pada sistem basis data, termasuk seorang adminsitrator sistem, tidak akan dapat melihat secara langsung kata kunci bagi setiap pengguna, langkah yang dilakukan apabila ada pengguna yang lupa akan kata kuncinya, adalah dengan cara melakukan *reset* terhadap nama pengguna tersebut.

Terlihat dari sisi *Control* atau pengendalian adalah bahwa sistem OAuth yang akan dikembangkan layak untuk dibangun karena selain menawarkan kontrol atau pengendalian yang lebih baik yang berada pada satu tempat, juga melakukan penjagaan terhadap keamanan data dengan lebih baik, termasuk data mengenai kata kunci pengguna yang telah terdaftar itu sendiri.

- *Efficiency* atau efisiensi adalah untuk mengetahui apakah sistem menggunakan secara maksimum sumber yang tersedia termasuk orang, waktu

aliran *form*, meminimalkan penundaan proses. Berikut adalah perbandingan sistem informasi yang lama dan sistem informasi yang baru :

Sistem Lama	Sistem Baru
Dinilai kurang efisien karena terdapat data *redundant* terhadap data pengguna beserta hak aksesnya yang tersimpan pada setiap sistem basis data dari aplikasi yang digunakan. Dengan nama pengguna yang sama saja, apabila ada 2 (dua) atau lebih aplikasi yang dibangun, maka tiap pengguna akan didaftarkan pada 2 (dua) atau lebih aplikasi sesuai dengan kewenangannya pada organisasi	Terlihat lebih efisien karena tiap pengguna hanya akan memiliki satu data nama pengguna dan hak aksesnya. Satu data ini, informasi nama pengguna dan hak aksesnya dapat digunakan untuk seluruh aplikasi klien yang terdaftar, kemudian aplikasi klien yang dibangun tidak perlu lagi membuat modul otentikasi lagi, karena prosesnya akan ditangani oleh OAuth Service

Dari sisi efisiensi pun aplikasi otentikasi ini layak untuk dibangun dan dikembangkan karena akan menggunakan sumber daya yang tersedia dengan lebih efisien tanpa redundansi data, selain itu tidak mengurangi performa sehingga tidak akan menjadi kendala dalam melakukan proses datanya.

- *Services* atau pelayanan adalah untuk mengetahui apakah sistem menyediakan layanan yang diinginkan dan handal pada siapa saja yang menginginkannya, dan apakah sistem fleksibel dan dapat dikembangkan. Berikut adalah perbandingannya antara sistem lama dan sistem baru :

Sistem Lama	Sistem Baru
Sistem telah menyediakan layanan yang handal namun terbatas pada masing-masing aplikasi yang berkaitan saja dengan sistem otentikasi tersebut, dengan kata lain tidak dapat digunakan untuk berbagai macam aplikasi klien yang berbeda	Sistem akan menyediakan layanan yang handal dan fleksibel untuk dapat dikembangkan ke berbagai aplikasi yang nantinya membutuhkan sistem otentikasi, tentu saja dengan semakin bertambahnya data yang terrekam dalam sistem informasi pajak daerah, akan semakin bertambah pula kebutuhan akan pengolahan dan pengelolaan datanya, maka akan semakin banyak aplikasi yang dibutuhkan dalam rangka kemudahan akses data,

Dari sisi *services* atau layanan tentu saja sistem otentikasi yang akan dibangun jauh lebih fleksibel dan dapat dikembangkan dibanding sistem lama yang cenderung layanan otentikasinya akan spesifik terhadap masing-masing aplikasi yang dilayaninya, dan dapat kita berikan status layak terhadap pengembangan aplikasi otentikasi ini.

Setelah kita kaji kelayakan dari sisi operasional secara keseluruhan, dari setiap kerangka kerja PIECES yang menyatakan bahwa sistem otentikasi ini layak untuk dibangun dan dikembangkan dari tiap bagiannya, maka dari faktor kelayakan operasional pun akan kita berikan nilai 9,0.

5. Menilai kelayakan Jadwal (*Schedule*)

   Kelayakan jadwal digunakan untuk menentukan bahwa pengembangan sistem dapat dilakukan dalam batas waktu yang telah ditetapkan. Pengem-

bangan sistem direncanakan selesai dalam waktu maksimal ± 11 (sebelas) minggu. Adapun perkiraan tahap-tahap pengembangan sistem dijadwalkan sebagai berikut :

Dalam proyek pengembangan sistem otentikasi di Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah Kabupaten Brebes dilakukan dalam 9 (sembilan) tahap, yaitu :

(a) Tahap analisa kebutuhan sistem

(b) Tahap studi kelayakan

(c) Tahap desain *user interface*

(d) Tahap desain data

(e) Tahap desain *proses*

(f) Tahap Pengodean

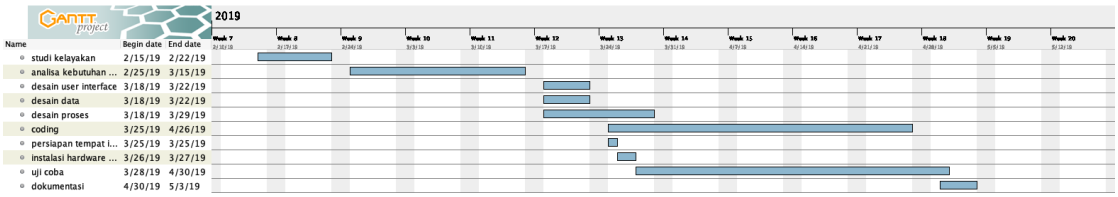
(g) Tahap persiapan tempat *instalasi*

(h) Tahap *instalasi* perangkat keras dan perangkat lunak

(i) Tahap uji sistem

(j) Tahap dokumentasi

Grafik *Gantt* untuk tahapan-tahapan tersebut adalah seperti pada gambar 2 berikut ini :



Gambar 2: Grafik *Gantt*

Karena pengembangan diukur dalam jam, hari, minggu, dan bulan maka kesalahan perkiraan (*error estimation*) yang dibutuhkan untuk perancangan dan implementasi menjadi kecil. Maka nilainya 8.0

Dari keseluruhan faktor kelayakan TELOS tersebut, jumlah dari semua faktornya adalah

$$Total = 9 + 9 + 9 + 9 + 8$$

$$Total = 44$$

Total *score* dari kelayakan ini adalah :

$$score = 44/5 = 8,8$$

Artinya perancangan pengembangan sistem informasi yang dievaluasi adalah LAYAK dengan resiko pengembangan sistem informasi yang cukup rendah.

2 SARANA DAN PRASARANA YANG MELIPUTI PERANGKAT KERAS DAN PERANGKAT LUNAK YANG DIPERLUKAN

Kebutuhan akan sarana dan prasarana untuk jenis perangkat keras yaitu :

1. Peladen (*server*)

   Peladen atau *server* berfungsi sebagai basis jalannya layanan-layanan aplikasi yang nantinya akan dibangun, nantinya tiap layanan aplikasi akan berjalan pada *container* Docker.

   Secara keseluruhan, inti dari aplikasi ini akan berjalan pada 4 (empat) *container*, yang secara ringkas dapat dijelaskan seperti berikut :

(a) *Container* pertama akan melayani *service* otentikasi, dimana halaman *login* akan berada pada layanan ini, termasuk pemberian token yang digunakan oleh aplikasi klien untuk melakukan akses pada *resource server*.

(b) *Container* kedua berupa layanan basis data yang secara langsung akan terhubung dengan *OAuth Service* atau layanan otentikasi pada *container* pertama dan *resource server*. Dalam layanan basis data ini tersimpan daftar pengguna, hak akses, serta daftar klien yang diijinkan melakukan koneksi ke beberapa *resource server* yang nantinya dibangun.

(c) *Container* ketiga berupa layanan dari *resource server* yang nantinya digunakan untuk melakukan pengelolaan baik itu daftar pengguna atau daftar aplikasi klien yang diijinkan untuk terhubung. *Resource server* ini akan berbagi bersama layanan basis data pada *container* kedua dengan layanan *OAuth Server* pada *container* pertama.

(d) *Container* keempat berupa layanan aplikasi klien yang berbentuk aplikasi *web*. Aplikasi klien ini akan menggunakan layanan dari *container* pertama, yaitu *OAuth service* sebagai layanan yang akan melakukan otentikasi terhadap pengguna, kemudian atas dasar hak dan kewenangan pengguna yang dapat melakukan akses ke pengelolaan daftar nama pengguna dan daftar aplikasi klien, mampu untuk melakukan akses ke *resource server*, apabila pengguna tidak memiliki hak dan kewenangan tersebut, maka layanan terhadap *resource server* secara sistematis akan ditolak.

Kebutuhan perangkat keras peladen (*server*) ini tidak perlu membeli karena nantinya akan menggunakan peladen (*server*) yang sudah ada sebagai

peladen (*server*) aplikasi *web* yang sudah beroperasi.

2. Kartu Jaringan

   Kartu jaringan digunakan atau terpasang pada peladen (*server*) agar peladen (*server*) dapat terhubung dengan jaringan, baik jaringan LAN (*Local Area Network*) maupun jaringan *internet*.

   Kartu jaringan ini tidak perlu membeli karena sudah terpasang lebih dari 1 (satu) pada tiap peladen (*server*) yang dapat dikonfigurasi secara mandiri seluruhnya.

3. Kabel *Unshielded Twisted Pair* (UTP)

   Kabel ini terdiri dari 4 (empat) pasang warna yang digunakan sebagai penghubung atau media komunikasi antar kartu jaringan yang terdapat pada tiap peladen (*server*) agar aplikasi atau data pada peladen dapat diakses melalui jaringan.

   Kabel ini juga tidak perlu membeli karena telah terpasang dan saling terhubung dari tiap peladen (*server*) ke area jaringan.

4. Konektor RJ45

   Konektor ini digunakan atau dipasangkan pada kabel UTP agar kabel UTP dapat dipasangkan pada kartu jaringan di tiap peladen (*server*) dan tiap perangkat jaringan seperti *switch*, *router*, ataupun *modem*.

   Masing-masing helai kabel UTP akan terpasang pada tiap pin yang ada pada konektor RJ45, yang kemudian apabila konektor RJ45 dipasangkan pada kartu jaringan, atau slot RJ45 yang terdapat pada *switch*, *router* ataupun *modem*, maka akan terhubung dengan pin pada RJ45 sehingga aliran data dapat disampaikan dari 1 (satu) peladen (*server*) ke perangkat jaringan yang lain.

5. *Switch*

Masing-masing komponen yang terkoneksi / terhubung ke jaringan dapat diistilahkan sebagai *host*, tentunya untuk membuat agar 1 (satu) *host* terhubungan dengan 1 (satu) *host* yang lain maka diperlukan sebuah kartu di masing-masing *host* dan sebuah kabel jaringan yang sudah terpasang konektor RJ45. Akan menjadi masalah apabila ada lebih dari 2 (dua) *host* yang perlu terhubung.

Rumusnya akan menjadi demikian :

(a) Untuk mengetahui jumlah kebutuhan kartu jaringan yaitu menggunakan rumus :

$$JumlahKartuJaringan = n(n-1)$$

dimana `n` adalah jumlah *host*

(b) Sedangkan untuk mengetahui jumlah kabel jaringan yaitu menggunakan rumus :

$$JumlahKabelJaringan = \frac{n(n-1)}{2}$$

dimana `n` adalah jumlah *host*

Ini akan lebih efektif dan efisien apabila menggunakan *star schema* dengan *switch* sebagai penghubung atau pengatur lalu lintas data antar *host*. Sebagai contoh bila ada 2 (dua) *host* yang akan dihubungkan dengan jaringan menggunakan *switch* ini, maka diperlukan 1 (satu) buah *switch*, 2 (dua) buah kabel jaringan (UTP), dan 2 (dua) buah kartu jaringan.

Bagaimana bila ada 3 (tiga) atau lebih *host* yang akan dihubungkan dalam sebuah jaringan yang sama, maka cukup menggunakan 1 (satu) buah *switch*,

`n` buah kabel jaringan (UTP) dan `n` buah kartu jaringan sebanyak jumlah *host* yang akan dihubungkan. Tentunya model *switch* perlu disesuaikan jumlah *port* dengan jumlah *host* yang akan dihubungkan.

Switch ini selain menjadikan jaringan komputer lebih efektif dan efisien, juga mengatur lalu lintas data secara lebih baik (dibandingkan *hub*). Salah satu kelebihan yang dimiliki *switch* dibanding *hub* adalah data akan disampaikan tepat kepada alamat yang dituju, sehingga pertukaran data nyaris aman dalam jalur komunikasinya, sedangkan *hub* lebih rentan karena setiap data yang data yang terkirim dan dilewatkan melalui *hub*, maka alat ini akan menyebarkan berita ke setiap *host* yang terhubung dengan *hub* yang kemudian diserahkan kepada kartu jaringan apakah data yang diterima sesuai dengan alamat yang dimiliki atau tidak. Dalam kondisi ini, keamanan data akan terlalu berresiko saat terjadi komunikasi.

Switch ini pun tidak perlu membeli karena sudah terpasang pada sistem yang lama sebagai penghubung antar 1 (satu) *host* dengan *host* yang lain.

6. *Router*

   Alat ini dalam istilah lain juga disebut sebagai gerbang (*gateway*), yaitu sebuah jembatan yang menghubungkan 2 (dua) atau lebih jaringan yang berbeda alamat, cara kerja alat ini yaitu dengan memiliki peta *routing* yang diprogram, peta inilah yang dibaca oleh *router* kemana paket data yang diterima harus dikirimkan, apabila tujuan paket data hanya dikirimkan untuk *host* yang berada di jaringan lokal, maka *router* tidak akan mengirimkan paket datanya keluar, melainkan akan dikirim ke alamat yang berada di dalam jaringan LAN (*Local Area Network*), tetapi bila tujuan paket data berada di luar jaringan, maka paket tersebut akan dikirimkan ke jaringan lain dalam lingkupnya, atau diteruskan ke *router* lain yang menghubungkan jaringan di

luar lingkupnya.

Tentunya cara kerja tersebut di atas dengan konfigurasi peta jaringan yang sebelumnya sudah diatur dalam *router*. Dan biasanya, pada setiap *router* terdapat *firewall* yang akan memberikan proteksi atau aturan-aturan komunikasi, dimana tidak semua data dapat diteruskan ke jaringan lain di luarnya, tetapi harus memenuhi persyaratan yang ditetapkan dalam tabel *firewall*.

Jaringan internal atau LAN pada bidang Pendataan dan Penetapan memiliki sebuah alamat jaringan lokal baik untuk jaringan nirkabel atau jaringan dengan kabel (UTP) di alamat 192.168.2.0/24, sedangkan untuk terhubung dengan internet digunakan sebuah *router* yang akan menjembatani atau menjadi gerbang antara jaringan internet dengan jaringan pada LAN. *Router* ini pula yang nantinya akan meneruskan data dan melakukan filter data baik yang masuk dari jaringan internet yang aksesnya atau *request*-nya akan diteruskan ke peladen aplikasi *web*.

Dengan kondisi demikian, maka biaya pengadaan untuk *router* sendiri tidak diperlukan, karena sudah terpasang dan sudah digunakan untuk kegiatan komunikasi data pada jaringan yang sudah ada sebelumnya.

7. *Modem*

   Fungsi dari *modem* yaitu untuk menghubungkan 2 (dua) atau lebih media jaringan yang berbeda sehingga data dapat dikomunikasikan pada jaringan yang berbeda media tersebut. Sebagai contoh, karena bidang Pendataan dan Penetapan menggunakan akses serat-optik (*fiber optik*) dari jaringan penyedia layanan internet, dan jaringan internal menggunakan kabel UTP, maka dibutuhkan *modem* untuk menerjemahkan data yang melintas melalui 2 (dua) media jaringan tersebut.

   Karena *modem* ini sudah digunakan pada sistem jaringan sebelumnya yang

sudah digunakan secara baik, maka tidak perlu membelinya kembali.

Melihat kondisi tersebut, kebutuhan perangkat keras secara keseluruhan sebetulnya tidak perlu mengeluarkan biaya tambahan lain karena perangkat-perangkat tersebut sudah tersedia dan cukup untuk mengakomodir berjalannya sistem informasi yang nantinya akan dibangun.

Kemudian, untuk sarana dan prasarana jenis perangkat lunak yang dibutuhkan yaitu :

1. Perangkat Lunak Sistem Operasi

   Perangkat lunak sistem operasi digunakan sebagai dasar dari seluruh sistem yang akan berjalan diatasnya. Pemilihan sistem operasi ini pun harus dapat memenuhi kriteria stabil dalam melakukan tugasnya sebagai peladen (*server*) untuk rentang waktu 24 jam selama 7 hari berturut-turut.

   Tentunya sistem operasi ini sudah terpasang baik pada peladen-peladen (*servers*) yang nantinya akan menjadi peladen (*server*) sistem basis data dan peladen (*server*) aplikasi *web*. Pada peladen (*server*) aplikasi *web* telah terpasang sistem operasi Ubuntu Server 14.04 yang tentu saja, untuk sistem operasi ini berlisensi gratis sehingga kebutuhan akan peladen (*server*) ini tidak perlu mengeluarkan biaya tambahan baik untuk lisensinya maupun pemasangannya.

2. Perangkat Lunak Sistem Basis Data

   Perangkat lunak basis data digunakan sebagai sistem yang nantinya akan mengatur dan menyimpan data-data pengguna, data-data hak akses untuk tiap pengguna, serta daftar detail dari aplikasi klien yang diijinkan untuk melakukan akses terhadap *resource server*.

Pemilihan perangkat lunak basis data ini didasarkan pada lisensinya yang tersedia gratis, namun tidak mengurangi performa dan kehandalannya sebagai tempat simpanan data, sehingga pilihan jatuh pada sistem basis data Postgresql.

Sistem basis data ini sudah lebih dari 30 Tahun dikembangkan dengan lebih dari 400 kontributor, fiturnya pun tidak kalah dengan sistem basis data yang dikembangkan oleh industri-industri besar pada bidang sistem basis data.

Nantinya sistem basis data ini akan berada dalam salah satu *container* docker yang melakukan simpanan data untuk seluruh pengguna berikut hak aksesnya, basis data ini pun nantinya digunakan untuk menyimpan data atau informasi mengenai aplikasi klien yang mampu terhubung dan menggunakan sistem otentikasi yang akan dibangun.

Karena gratis, maka untuk menggunakan sistem basis data Postgresql ini tidak perlu mengeluarkan biaya tambahan untuk pengadaannya.

3. Perangkat Lunak Peladen *Web*

   Perangkat lunak peladen (*server*) *web* digunakan untuk menjalankan atau menyediakan aplikasi *web* yang telah dibangun, yaitu sistem otentikasi di Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah Kabupaten Brebes.

   Peladen (*server*) *web* ini nantinya akan ada 2 (dua) macam, yaitu yang berbentuk *web container* yang nantinya digunakan oleh *OAuth Service* dan *Resource Service*, dan yang lain adalah *web server* seperti pada umumnya yang akan melayani aplikasi klien yang telah dibangun.

   Untuk peladen (*server*) *web* yang berbentuk *web container* sebetulnya sudah ada dalam satu paket proyek yang dibangun menggunakan *framework*

Spring Boot, nantinya paket proyek ini akan dimasukkan dalam sebuah *container* docker, sehingga proses *deployment* akan lebih mudah dengan model *Continuous Integration* dan *Continuous Deployment*.

Alasan kenapa dipilih peladen yang mendukung *servlet* atau *web container* karena memiliki beberapa kelebihan seperti berikut ini :

(a) Efisien dan baik dalam *performance*

Performance servlet dapat dikatakan efisien dan baik karena tidak ada proses pembuatan berulang untuk tiap *request* dari klien (*client*). Setiap *request* ditangani oleh proses *servlet container*. *Servlet* tidak dibuat dan dihancurkan secara berulang-ulang, melainkan tetap tersimpan pada memori untuk menangani *request* yang datang selanjutnya.

(b) *Powerful*

Servlet memiliki kemampuan yang lengkap antara lain mampu melakukan penanganan *request*, *session*, *cookie*, akses ke basis data dengan *Java Database Connection* (JDBC) dan *caching*, serta pustaka yang lengkap untuk pembuatan aplikasi *web*.

(c) Aman

Servlet memiliki fasilitas *security* atau keamanan yang baik dan merupakan bagian dari teknologi Java yang sudah dari asalnya didesain dengan *security* atau keamanan yang baik.

(d) Portabilitas

Teknologi Java *servlet* dapat dijalankan di berbagai *servlet container, application server*, maupun sistem operasi.

(e) Proses *development* yang lebih cepat

Dengan menggunakan *servlet*, kita dapat menggunakan pustaka Java

yang lengkap dan menggunakan komponen yang sudah ada tanpa membangun dari awal kembali.

(f) Tangguh

Servlet merupakan teknologi Java yang memiliki penanganan *memory* yang baik dan *garbage collector* sehingga menjadikan aplikasi *web* yang tangguh dan stabil.

(g) Telah digunakan dan diakui di dunia

Servlet merupakan teknologi Java yang telah digunakan di berbagai belahan dunia. Dapat ditemukan berbagai komponen, solusi, dan dukungan yang ditawarkan baik secara gratis maupun komersial.

(h) Murah

Dikatakan murah karena JDK Java dapat diunduh secara gratis, begitupun dengan *servlet container*.

Pemilihan perangkat lunak peladen (*server*) *web* yang mendukung *servlet* ini pun tidak perlu mengeluarkan biaya, cukup menggunakan Apache Tomcat atau Jetty yang memberikan lisensi gratis dalam penggunaannya, dengan konfigurasi yang tepat, maka peladen (*server*) *web* Apache Tomcat ini cukup untuk melayani permintaan aplikasi dari beberapa klien sekaligus. Sehingga untuk pengadaan perangkat lunak peladen (*server*) *web* ini pun tidak memerlukan tambahan biaya, dan sudah ada dalam paket *framework* Spring Boot untuk kemudahan *debugging* dan *deployment*.

Perangkat lunak peladen (*server*) yang digunakan untuk aplikasi klien nantinya akan menggunakan Nginx yang juga tersedia secara gratis dan sudah teruji kehandalannya karena beberapa perusahaan teknologi informasi besar pun menggunakan Nginx sebagai *web server*-nya.

Ketiga peladen (*server*) ini akan ditempatkan pada masing-masing docker *container* untuk memberikan layanan sesuai dengan tugas dan fungsinya.

4. Perangkat Lunak Desain Aplikasi

   Perangkat lunak desain aplikasi digunakan untuk membuat desain pembangunan aplikasi, mulai dari desain struktur basis data, desain *Unified Modeling Language* (UML) yang digunakan untuk membuat spesifikasi standar untuk mendokumentasikan, menspesifikan, dan membangun sistem perangkat lunak, desain UML sendiri dapat menggambarkan atau memodelkan diagram struktur aplikasi dan diagram sifat atau cara kerja aplikasi.

   Perangkat lunak atau desain aplikasi ini pun tidak perlu mengeluarkan biaya tambahan, ada perangkat lunak untuk desain aplikasi yang gratis dan disediakan secara daring oleh https://www.draw.io. Aplikasi ini memiliki fasilitas yang cukup banyak dan bukan hanya untuk membuat desa UML saja, melainkan desain dari struktur basis data pun mampu dilakukan, kemudian desain jaringan komputer pun dapat dilakukan di aplikasi ini.

5. Perangkat Lunak *Integrated Development Environment* (IDE)

   Perangkat lunak IDE ini digunakan untuk membangun aplikasi *web*, baik dari sisi tampilan tatapmuka pengguna (*user interface*) maupun dari sisi kode. Pengujian *unit testing* pun dapat dilakukan dengan menggunakan perangkat lunak ini.

   Beberapa pilihan untuk perangkat lunak IDE ini pun beragam dan banyak yang menyediakan secara gratis. Sebagai contoh adalah Netbeans dan Eclipse. Untuk pengembangan aplikasi ini dipilih Intellij IDEA sebagai perangkat dalam membangun layanan *OAuth Service* dan *Resource Service* karena tersedianya beragam *plugins* sebagai pendukung dalam membangun sebuah aplikasi, sedangkan untuk membangun aplikasi klien yang digunakan untuk

melakukan pengelolaan atau manajemen pengguna dan aplikasi klien lain, akan menggunakan Visual Studio Code, karena pada Visual Studio Code ini pun terdapat beragam *plugins* yang mampu mendukung dan mempermudah pengerjaan aplikasi klien nantinya.

Kedua aplikasi itu, baik Intellij IDEA dalam hal ini, yang digunakan adalah versi komunitas (*community edition*) dan Visual Studio Code tersedia secara gratis dan bebas diunduh dari masing-masing laman penyedianya, sehingga untuk pengadaan perangkat lunak IDE ini tidak perlu mengeluarkan biaya tambahan.

6. Perangkat Lunak Docker

   Perangkat lunak docker ini digunakan untuk membangun *container-container* kecil yang nantinya berjalan secara independen yang tiap *container*-nya akan menjalankan sebuah layanan (*service*) tertentu yang spesifik, contohnya, sebuah *container* hanya akan menjalankan layanan (*service*) basis data saja, sedangkan *container* yang lain hanya akan menjalankan layanan (*service*) *(*OAuth Service), keduanya akan berjalan secara independen dan paralel yang apabila ingin kedua layanan atau *container* ini saling terhubung, akan melalui sebuah sambungan yang didefinisikan secara khusus.

   Bisa diistilahkan bahwa docker ini adalah bentuk virtualisasi yang lebih ringan, karena sebetulnya, didalam sebuah *container* akan terdapat sebuah sistem operasi yang berjalan, serta beberapa aplikasi pendukung yang tujuannya spesifik untuk menjalankan layanan (*service*) utamanya. Contohnya, saat ada *container* yang memiliki layanan (*service*) basis data Postgresql, maka disana akan terdapat sebuah sistem operasi, basis data Postgresql, dan beberapa pustaka pendukung agar basis data Postgresql dapat berjalan.

   Untuk penggunaan docker ini kita pun dapat memilih menggunakan versi

Enterprise atau *community edition*, untuk membangun aplikasi ini, sebetulnya kebutuhan untuk menggunakan versi *community edition* sudah cukup, sehingga kita tidak memerlukan biaya tambahan lain untuk mendapatkannya.

7. Perangkat Lunak Git

   Perangkat lunak Git digunakan untuk melakukan *Continuous Integration* dengan lebih sempurna. Fungsi dari perangkat lunak Git ini sebetulnya tergolong dalam aplikasi *Versioning Control* yang melakukan pencatatan atau *log* dari kondisi perubahan berkas atau kandar, sehingga saat melakukan perubahan-perubahan yang terjadi pada fitur aplikasi yang dibangun dapat dilihat historis perubahannya, dan apabila ada kesalahan implementasi fitur dan perlu untuk dikembalikan kondisi ke beberapa keadaan diwaktu sebelumnya, mampu dilakukan dengan mudah tanpa harus mengubah kondisi berkas saat ini.

   Fungsi lain dari perangkat lunak Git ini sebetulnya adalah untuk kebutuhan kolaborasi, dimana 1 (satu) atau lebih pengkode (*coder*) dapat mengerjakan pekerjaan koding (*coding*) secara bersama-sama dan paralel, setelah perkerjaan satu sub-modul atau sebuah fitur selesai, hasil kodenya dapat digabung sehingga menjadi satu fitur yang lengkap dan dapat digunakan.

   Penggunaan Git ini tidak terlepas dari sebuah peladen (*server*) Git yang digunakan untuk mengumpulkan seluruh hasil penetapan (*commit*) kode dari masing-masing pengkode (*coder*), yang digunakan nantinya sebagai peladen (*server*) Git adalah Gitlab, karena disana sudah terdapat fitur *Continuous Deployment* (CD) yang berfungsi untuk melakukan instalasi atau pemasangan secara otomatis pada peladen (*server*) VPS milik unit kerja Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah.

Pada proses *Continuous Integration* dan *Continuous Deployment* (CI/CD) nantinya, dapat dilakukan konfigurasi fase penyelarasan kode yang dapat diatur sesuai kebutuhan, contohnya, pada saat kode dari tiap pengkode (*coder*) akan digabung, maka otomatis akan melakukan *unit test* untuk memastikan bahwa tidak ada kesalahan logika program per sub-unit kode terkecil sekaligus melakukan *integration test* untuk memastikan bahwa fitur baru atau pembersihan *bugs* yang dilakukan tidak bermasalah apabila sistem berada pada lingkungan yang mengharuskannya terhubung ke beberapa sumber daya (*resource*) seperti sistem basis data, jaringan, atau hal lain yang dibutuhkan sistem untuk melakukan kegiatan operasionalnya.

Penggunaan Git dapat diperoleh dan diunduh dengan gratis dari https://git-scm.com/, sedangkan Gitlab sendiri menyediakan layanannya dalam beberapa versi sesuai kebutuhan, dan fitur CI/CD yang kita butuhkan sudah ada pada versi gratis yang ditawarkan oleh layanan ini, sehingga tidak perlu mengluarkan biaya untuk menggunakan layanan dari GItlab ini.

8. *Framework* Spring

   Framework Spring adalah perangkat yang dapat membantu untuk membangun aplikasi Java berskala besar dengan mudah. *Framework* Spring sendiri sebetulnya terbagi menjadi beberapa modul. Aplikasi yang akan dibangun dapat memilih modul mana saja yang akan digunakan. Modul inti dari *Framework* Spring ini adalah modul *core container*, yang termasuk di dalamnya adalah model untuk konfigurasi dan mekanisme *dependency injection*.

   Dependency Injection adalah salah satu fasilitas yang memudahkan kita untuk tidak perlu membentuk tiap objek baru secara manual, melainkan cukup dideklarasikan *bean*, kemudian *framework* Spring yang akan melakukan pembentukan objeknya secara otomatis.

Selain itu *framework* Spring pun memberikan dukungan dasar terhadap beberapa arsitektur aplikasi seperti *messaging*, transaksi data dan *persistence*, serta *web*. Spring pun menyertakan *servlet-based* Spring MVC *web framework*.

Beberapa spesifikasi yang didukung oleh *Framework* Spring ini adalah seperti berikut ini :

- Servlet API (JSR 340)
- WebSocket API (JSR 356)
- Concurrency Utilities (JSR 236)
- JSON Binding API (JSR 367)
- Bean Validation (JSR 303)
- JPA (JSR 338)
- JMS (JSR 914)
- Common Annotations (JSR 250)

Beberapa waktu tata aturan pada pengembangan aplikasi Java EE telah mengalami perubahan. Pada awal hadirnya Java EE dan Spring, aplikasi *web* yang telah dibangun akan di *deploy* ke peladen aplikasi. Namun sekarang dengan hadirnya Spring Boot, aplikasi dibangun dalam lingkungan *devops* dan lebih ramah dengan teknologi *cloud* dengan adanya *servlet container* yang dipendam dalam paket *framework*. Bahkan pada *Framework* Spring 5, aplikasi WebFlux tidak perlu menggunakan API *servlet* secara langsung dan dapat menggunakan peladen *web* seperti Netty yang tidak memiliki *servlet containers*.

Framework Spring ini pun tersedia secara gratis dan dapat digunakan tanpa perlu mengeluarkan biaya tambahan lagi.

9. Spring Data JPA

   Spring Data JPA sebetulnya adalah bagian dari *framework* Spring yang tujuan dibentuknya adalah untuk mengurangi jumlah kode *boilerplate* yang dibutuhkan untuk mengimplementasikan lapisan akses data ke berbagai sistem basis data.

   Kelebihan dari penyederhanaan ini adalah mengurangi jumlah *artifact* yang harus dideklarasikan dan di *maintained*, konsistensi pada pola data akses, dan konsistensi pada konfigurasi. Satu-satunya yang perlu dideklarasikan pada Spring Data JPA adalah *interface* DAO.

   Untuk dapat mengimplementasikan Spring Data JPA, *interface* DAO terlebih dahulu harus melakukan *extends* terhadap salah satu antar-muka *repository* yang disediakan oleh Spring Data JPA, nantinya Spring Data akan mencari antar-muka ini dalam pustakanya, kemudian membuatkan implementasinya secara otomatis.

   Pustaka atau *framework* Data JPA ini pun disediakan gratis tanpa perlu mengeluarkan tambahan biaya untuk pengadaannya.

10. Pustaka *Java Database Connection* (JDBC)

    Pustaka *driver* JDBC diperlukan sebagai jembatan antara kode aplikasi yang dibangun dengan data yang berada dalam sistem basis data. *Driver* JDBC ini spesifik dan berbeda untuk tiap sistem basis data, karena sistem basis data yang digunakan adalah Postgresql, maka *driver* yang diperlukan adalah *driver* untuk basis data Postgresql yang dapat diunduh di laman https://jdbc.postgresql.org/download.html, karena kita menggunakan *Maven* sebagai *build-tools*, sehingga kita cukup mendeklarasikan skrip XML seperti berikut di berkas konfigurasi untuk Maven :

    <dependency>

```
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
</dependency>
```

Biasanya tiap-tiap sistem basis data akan menyediakan *driver* JDBC masing-masing sebagai salah satu bentuk dukungan bahwa sistem basis data tersebut mampu ikut terintegrasi bersama aplikasi yang dibangun dengan bahasa pemrograman Kotlin.

Lisensi untuk pustaka *driver* JDBC ini pun gratis sehingga tidak perlu mengeluarkan biaya untuk pengadaannya.

11. Pustaka Spring Security

    Pustaka ini digunakan untuk melakukan pembatasan akses terhadap beberapa sumber daya (*resource*) yang disediakan oleh *framework* Spring untuk pengguna yang telah terdaftar dan memiliki kewenangan.

    Spring Security sebetulnya juga adalah sebuah *framework* yang memberikan fitur otentikasi dan otorisasi pada aplikasi Java, dan seperti seluruh *project* yang dibangun oleh *Spring*, bahwa *framework* Spring Security ini dapat secara mudah dikembangkan sesuai dengan kebutuhan aplikasi yang akan dibangun.

    Penggunaan pustaka Spring Security ini pun gratis sehingga tidak perlu mengeluarkan biaya tambahan untuk pengadaannya.

12. Pustaka Spring Security OAuth

    Pustaka ini sebetulnya adalah hasil perluasan atau pustaka pendukung untuk Spring Security, sehingga mampu melayani menggunakan protokol OAuth dan OAuth 2, dan yang akan diimplementasikan pada aplikasi yang dibangun nantinya menggunakan protokol OAuth 2 sehingga penggunaan

pustaka Spring Security OAuth ini sudah dapat mencukupi kebutuhan pengembangan.

Pustaka Spring Security OAuth ini pun gratis, sehingga tidak perlu mengeluarkan biaya tambahan untuk pengadaannya.

13. Pustaka Spring Security JWT

    Pustaka ini digunakan untuk mengimplementasikan konsep *Authorization Grant Type* dengan mode implisit (*implicit*), dimana nantinya token yang dihasilkan oleh layanan (*services*) *OAuth Service*) akan berbentuk JWT (JSON Web Token), dimana token ini adalah hasil enkripsi dari sebuah objek JavaScript yang memuat informasi-informasi seperti nama pengguna, hak akses, informasi aplikasi klien, dan beberapa detail lainnya.

    Token ini juga yang nantinya digunakan oleh aplikasi klien untuk melakukan permintaan data (*request*) ke *resource server* beserta detail data yang akan diminta.

    Pustaka Spring Security JWT ini pun tersedia secara gratis sehingga tidak perlu mengeluarkan biaya tambahan untuk pengadaannya.

14. Pustaka Spring Web

    Pustaka ini digunakan pada saat membangun layanan (*service*) *OAuth Service*, dimana nantinya pengembangan layanan akan menggunakan konsep MVC (Model-View-Controller), karena konsep ini selain mempermudah proses pengembangan di awal, juga akan mempermudah dalam pemeliharaan kode yang telah dibangun.

    Pustaka ini akan digunakan bersama dengan pustaka *templating engine* Thymeleaf di layanan (*service*) *OAuth Service*, fungsi dari Thymeleaf sendiri nantinya akan membentuk tampilan (*user interface*) dari halaman *login* untuk

proses otentikasi pengguna.

Pustaka Spring Web ini pun tersedia dan dapat digunakan secara gratis sehingga tidak perlu mengeluarkan biaya tambahan untuk pengadaannya.

15. Pustaka Flyway

    Pustaka Flyway ini digunakan mirip seperti fungsi Git dalam hal *versioning*-nya saja, namun penggunaannya spesifik pada pembentukan struktur data pada basis data.

    Pemanfaat khususnya yang dibutuhkan oleh pustaka ini yaitu untuk membentuk struktur data untuk daftar pengguna, hak akses, dan daftar aplikasi klien, dimana nantinya pustaka ini akan secara otomatis membentuk struktur tabelnya pada sistem basis data, dan mengisikan datanya jika diperlukan, yang perlu dilakukan seorang pemrogram (*coder*) hanya cukup mendeklarasikan struktur tabel dan isian data pada berkas dengan penamaan yang telah ditentukan oleh pustaka ini.

    Pustaka Flyway ini pun tersedia secara gratis sehingga tidak memerlukan tambahan biaya. Untuk menggunakannya, kita cukup mendeklarasikannya pada berkas konfigurasi yang digunakan oleh Maven.

16. Pustaka Jackson

    Pustaka ini sangat terkenal dan efisien untuk mengubah objek Java ke dalam format JSON atau sebaliknya. Pustaka ini akan digunakan untuk layanan REST nantinya, ketika API akan mengambalikan data sebuah objek Java, maka Jackson akan mengubah secara otomatis objek tersebut menjadi JSON, begitu pula saat menerima data dari aplikasi klien yang berbentuk JSON, akan mengubahnya secara otomatis menjadi bentuk objek Java.

17. Kotlin *Compiler*

Compiler ini digunakan untuk melakukan kompilasi kode yang ditulis dengan bahasa pemrograman Kotlin. Pemilihan bahasa pemrograman Kotlin ini sebagai dasar pembangunan aplikasi atau sistem Otentikasi karena mampu bersinergi dengan sistem yang dibangun menggunakan bahasa Java, dengan kata lain, karena *framework* yang digunakan adalah Spring Boot yang dibangun menggunakan bahasa pemrograman Java, penggunaan bahasa pemrograman Kotlin dengan menggunakan *framework* Spring sangat memungkinkan dengan ditambah kehandalan dari bahasa pemrograman Kotlin yang mampu mendeteksi atau mengurangi timbulnya *Null Exception* pada aplikasi.

Kotlin *Compiler* sendiri tersedia secara gratis dan dapat diunduh dari lamah https://kotlinlang.org/, sehingga penggunaannya tidak memerlukan biaya tambahan.

18. Maven

    Maven digunakan sebagai *build-tools* untuk mempermudah pengelolaan proyek yang akan dibangun, untuk menggunakan berbagai macam pustaka seperti diatas, cukup dideklarasikan pada sebuah berkas konfigurasi Maven dengan nama `pom.xml`, untuk melakukan pemaketan (*packaging*) pun sangat mudah, nantinya dengan Maven, sebelum sebuah proyek didistribusikan atau di *deploy* ke peladen (*server*), akan dilakukan serangkaian tes (*unit test* dan *integration test*) untuk memastikan bahwa kode yang dibangun telah memenuhi kelayakan untuk digunakan.

19. *Framework* Angular

    Angular digunakan untuk membangun tampilan tatap muka (*user interface*) dari aplikasi yang dibangun. Nantinya, tiap tampilan tatap muka (*user interface*) akan terhubung ke satu atau lebih *resource server* yang menyediakan

layanan datanya.

Penggunaan *Framework* Angular ini dipilih karena sudah memiliki fitur PWA (*Progressing Web Application*) dimana tampilan akan menyesuaikan diri saat diakses dari beberapa perangkat dengan resolusi yang berbeda-beda, contohnya yaitu, aplikasi akan memiliki perbedaan tampilan saat diakses menggunakan *browser* di komputer atau laptop, dan tampilan saat diakses menggunakan aplikasi *mobile* di *smartphone*.

Keunggulan lain yang diperoleh dari *Framework* Angular adalah karena hasilnya berupa aplikasi dengan konsep SPA (*Single Page Application*), semua isi tampilan akan dikirimkan secara keseluruhan ke *browser* di komputer klien atau *smartphone*, untuk menampilkan atau melakukan operasi data, aplikasi cukup berkomunikasi dengan bagian *back-end*, dalam hal ini *resource server* dengan menggunakan arsitektur REST dengan format pengiriman datanya berbentuk JSON (*JavaScript Object Notation*), tentu saja berbeda dengan teknologi aplikasi web yang setiap melakukan *request* atau permintaan data ke peladen (*server*), peladen (*server* akan mengembalikannya dalam bentuk satu halaman penuh beserta datanya, namun dengan menggunakan arsitektur REST dengan implementasi konsep SPA (*Single Page Application*) , penghematan *bandwidth* dapat ditekan dan aplikasi dapat berjalan dalam kondisi dimana lalu lintas data terbatas.

Framework Angular ini pun tersedia secara gratis dan bisa diunduh dengan cara yang telah disediakan pada laman https://angular.io/, sehingga tidak memerlukan biaya tambahan untuk pengadaannya.

3 SUMBER DAYA MANUSIA YANG TERLIBAT DALAM SISTEM APLIKASI

Karena sistem otentikasi ini melakukan tugasnya sebagai pengaman atau pelindung terhadap akses data yang ditujukan ke *resource service*, maka sumber daya manusia yang terlibat pada penggunaan sistem ini nantinya adalah seluruh personil yang akan melakukan akses terhadap data yang berada di Bidang Pendataan dan Penetapan Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah Kabupaten Brebes.

Nantinya setiap personil yang terlibat atau menggunakan aplikasi klien akan memiliki nama pengguna dan kata kunci masing-masing yang tersimpan dalam sistem basis data. Termasuk nantinya saat ada aplikasi klien yang dapat digunakan oleh pihak eksternal organisasi atau unit kerja, maka perlu didaftarkan terlebih dahulu dalam sistem ini.

Personil yang memiliki kemampuan untuk melakukan akses dan melakukan perubahan data terhadap data pengguna dan aplikasi klien adalah seorang administrator, seorang administrator nantinya akan diberikan kewenangan dan diserahi tanggung jawab sebagai pengelola sistem otentikasi ini.

4 ORGANISASI SISTEM APLIKASI

Organisasi dari sistem otentikasi ini seperti terlihat pada gambar 1 dimana secara internal, akan terdiri dari 4 (empat) bagian, yaitu :

1. *Container* docker yang berisi layanan *OAuth Service*, dimana layanan ini akan berisi halaman otentikasi tempat pengguna melakukan otentikasi ketika akan mengakses dengan aplikasi klien (*user agent*). Layanan (*service*) ini pun akan terhubung dengan sistem basis data tempat daftar pengguna dan

daftar aplikasi klien (*user agent*) disimpan. Sistem basis data akan berada pada *container* docker yang berbeda dan berjalan secara independen.

2. *Container* docker yang berisi layanan sistem basis data Postgresql yang digunakan untuk menyimpan data mengenai daftar pengguna dan daftar aplikasi klien (*user agent*). *Container* ini akan terhubung ke layanan *OAuth service* dan *Resource service*.

3. *Container* docker yang berisi layanan (*service*) dari *Resource Server* ini digunakan untuk melakukan pengelolaan daftar pengguna (*resource owner*) dan daftar aplikasi klien (*user agent*). *Container* ini akan terhubung dengan layanan sistem basis data dan aplikasi klien (*user agent*) yang menjadi tampilan tatap muka (*user interface*) untuk pengelolaan pengguna (*resource owner*) dan aplikasi klien (*user agent*).

4. *Container* docker yang berisi layanan aplikasi klien (*user agent*) aplikasi *web* yang berbentuk *Single-Page Apps*, fungsinya adalah sebagai tampilan tatap muka (*user interface*) dari manajemen daftar pengguna (*resource owner*) dan daftar apilkasi klien (*user agent*). Kedepannya, aplikasi klien (*user agent*) ini tidak hanya menyediakan tampilan tatap muka (*user interface*) untuk pengelolaan daftar pengguna dan aplikasi klien (*user agent*) saja, melainkan dapat dikembangkan menjadi aplikasi apapun untuk kebutuhan tampilan tatap muka dari aplikasi-aplikasi yang dibangun di organisasi.

5 WAKTU DAN BIAYA YANG DIBUTUHKAN DALAM PEMBUATAN / PENGEMBANGAN SISTEM APLIKASI SECARA MENYELURUH

Dari sisi biaya, maka tidak diperlukan pengeluaran biaya untuk pembuatan / pengembangan ini, karena sebetulnya seluruh perangkat keras sudah terpasang dan berjalan normal, dan dari pengadaan perangkat lunaknya pun semuanya menggunakan versi gratis.

Sedangkan dari waktu yang dibutuhkan, akan menghabiskan waktu sekitar 11 (sebelas) minggu yang tentu saja ini waktu yang tidak terlalu lama untuk membangun sebuah aplikasi.

6 MANFAAT DAN DAMPAK DARI PENGEMBANGAN SISTEM APLIKASI

Beberapa manfaat dan dampak yang didapat dari pengolahan data nantinya adalah sebagai berikut :

1. Pengguna cukup mengingat 1 (satu) bagian informasi saja berupa nama pengguna dan kata kuncinya untuk melakukan akses terhadap seluruh aplikasi yang tersedia, karena otentikasi akan terjadi pada satu tempat saja, yaitu di *OAuth Server*, yang kemudian hak aksesnya akan didelegasikan ke aplikasi klien (*user agent*) untuk kemudian digunakan sebagai pendukung permintaan (*request*) data ke *resource server*.

2. Waktu pengembangan aplikasi selanjutnya lebih singkat karena fase atau

bagian waktu yang digunakan untuk membangun halaman otentikasi sudah dapat ditangani oleh *OAuth Server*.

3. Administrator sistem atau aplikasi lebih mudah untuk mendaftarkan atau mengelola pengguna berikut hak aksesnya karena hanya dilakukan dari 1 (satu) tempat saja.