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

15 Februari 2019

Priyanto Tamami, S.Kom.

1 PENENTUAN MASALAH DAN PELUANG YANG DITUJU SISTEM

1.1 Penentuan Masalah

Sudah lama sejak penerapan Undang Undang Nomor 28 Tahun 2009 tentang Pajak Daerah dan Retribusi daerah, beberapa kewenangan atas jenis pajak yang sebelumnya dikelola oleh Pemerintah Pusat, dengan Undang-Undang ini kewenangannya dilimpahkan ke Pemerintah Daerah dengan tujuan memberikan kekuatan dan kemandirian daerah dengan penopang yang bersumber dari Pendapatan Asli Daerah.

Beberapa ketentuan mengenai Retribusi pun diatur dalam Undang Undang Nomor 28 Tahun 2009 yang dikenal dengan Undang Undang PDRD (Pajak Daerah dan Retribusi Daerah) ini.

Dengan perluasan jenis pajak yang dilimpahkan ke Pemerintah Daerah, terkhusus Pemerintah Kabupaten / Kota, maka data yang perlu dikelola pun semakin bertambah, terlebih saat melakukan pengelolaan Pajak Bumi dan Bangunan Perdesaan dan Perkotaan.

Pada pelimpahan jenis pajak Pajak Bumi dan Bangunan Perdesaan dan Per-

kotaan, Pemerintah Kabupaten Brebes menerima data sejumlah kurang lebih 1 juta objek pajak. Penyerahan jenis pajak ini tidak hanya data, melainkan aplikasi untuk pengelolaannya pun dilimpahkan juga, sehingga Pemerintah Daerah diharapkan dapat langsung melakukan kegiatan operasionalnya pada Tahun pelimpahan terjadi.

Serupa dengan jenis pajak Pajak Bumi dan Bangunan Perdesaan dan Perkotaan, jenis pajak yang lain pun dituntut untuk lebih tertib administrasi dalam pengelolaannya sehingga penggunaan aplikasi pun diterapkan untuk mempermudah prosesnya.

Untuk jenis pajak selain Pajak Bumi dan Bangunan Perdesaan dan Perkotaan sudah beberapa kali terjadi perubahan implementasi dalam penggunaan aplikasi atau sistem informasi, perubahan ini terjadi karena adanya perubahan struktur pada Badan atau Dinas yang melakukan pengelolaan Pajak Daerah.

Imbas yang terjadi karena perubahan implementasi sistem atau aplikasi yang digunakan ini adalah perlunya adaptasi dan pelatihan terhadap penggunaan aplikasi tersebut sehingga terkadang mengganggu proses pelayanan.

Selain perubahan dari sistem informasi atau aplikasi untuk pengelolaan Pajak Daerah, pada Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah Kabupaten Brebes pun menggunakan banyak aplikasi yang berbeda-beda, sehingga masing-masing pengguna terkadang akan memiliki beberapa akun yang berbeda untuk masing-masing aplikasi atau sistem yang digunakan sesuai dengan tugas dan fungsinya masing-masing.

Seiring berjalannya waktu, kebutuhan akan data yang disimpan dan dikelola tentunya akan semakin bertambah, untuk itu akan diperlukan beberapa aplikasi lagi yang semakin berkembang menyesuaikan kebutuhannya.

Dari hal tersebut, tentunya masing-masing administrator sistem pun akan membuat akun untuk masing-masing pengguna yang jelas akan banyak sekali re-

dundansi data.

1.2 Peluang Yang Dituju

Dari kondisi tersebut di atas diperlukan sebuah aplikasi yang mampu memberikan layanan otentikasi dan otorisasi tunggal, dimana masing-masing pengguna nantinya hanya cukup membutuhkan sebuah akun dan *password* untuk melakukan akses data ke berbagai aplikasi yang ada, sesuai dengan tugas dan fungsi serta kewenangannya masing-masing.

Mengacu kepada Konsep OAuth2 yang dijelaskan pada IETF dengan nomor RFC 6749, bahwa kerangka kerja OAuth2 mampu memberikan otentikasi dan otorisasi terhadap pengguna dalam melakukan akses ke *resource server*.

Nantinya akan ada 3 (tiga) bagian yang berperan dalam implementasi ini, bagian tersebut adalah seperti berikut :

1. *Authorization Server* yang berfungsi untuk melakukan otentikasi dan otorisasi terhadap pengguna, nantinya *Authorization Server* akan memberikan sebuah token akses ke Aplikasi Klien setelah melakukan otentikasi terhadap pengguna.

2. *Resource Server* yang nantinya berperan untuk memberikan data yang dibutuhkan atau sebagai *back-end* dari aplikasi klien dalam mengelola data, nantinya *Resource Server* ini akan melakukan akses langsung ke sistem basis data tempat sumber data berada. *Resource Server* pun akan dilindungi oleh mekanisme akses data terbatas dimana otentikasi yang diberikan kepada pengguna akan difilter dan di verifikasi oleh *Resource Server* yang dikomunikasikan dengan *Authorization Server*.

3. Aplikasi Klien yang dibangun untuk melakukan akses terhadap *Resource Server* dengan menggunakan konsep otorisasi yang diberikan oleh *Authori-*

zation Server berupa sebuah token dari hasil otentikasi Pengguna di *Authorization Server*, dimana pertukaran data atau informasi antara Aplikasi klien dengan *Resource Server* akan selalu mengirimkan sebuah token sebagai bahan otentikasi. Aplikasi Klien ini tidak terbatas kepada aplikasi *desktop*, atau *mobile*, atau *web*, melainkan aplikasi yang di eksekusi di peladen pun apabila memanfaatkan *Resource Server* yang terhubung dengan *Authentication Server* ini akan dianggap sebagai Aplikasi Klien.

4. Pengguna, ini adalah entitas yang memiliki hak akses ke *Resource Server* untuk melakukan manipulasi atau akses data sesuai dengan kewenangan yang diberikan.

Studi kelayakan ini akan menitikberatkan pada pembentukan *Authorization Server* sebagai sarana untuk melakukan otentikasi dan otorisasi sistem, sehingga diharapkan seluruh sistem yang dibangun berikutnya akan mengacu pada penggunakan *Authorization Server* ini untuk melakukan otentikasi dan otorisasi.

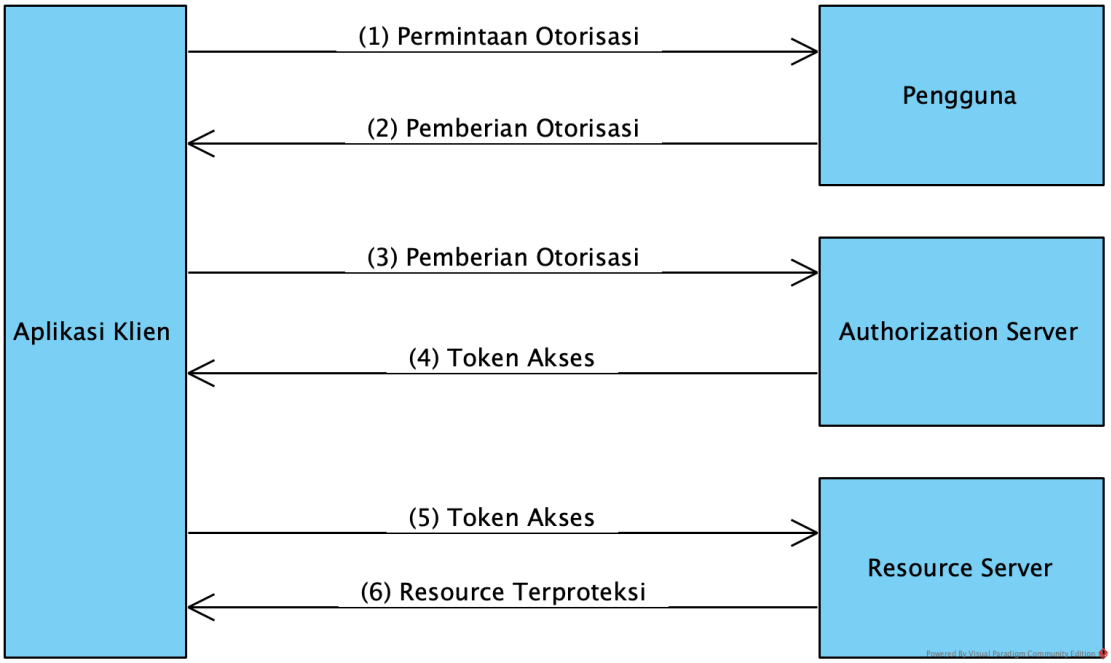
Tidak terbatas pada aplikasi yang akan dibangun yang bersifat baru sesuai kebutuhan organisasi, namun aplikasi-aplikasi atau sistem yang sifatnya adalah migrasi dari sistem lama, diharapkan akan menggunakan *Authorization Server* ini sebagai sarana melakukan otorisasi dan otentikasi.

2 PEMBENTUKAN SASARAN SISTEM BARU SECARA KESELURUHAN

Sasaran dari sistem baru ini secara keseluruhan adalah dapat melakukan otentikasi dan otorisasi terhadap pengguna dan sistem informasi atau aplikasi yang digunakan oleh pengguna.

Pada bagian otentikasinya, pengguna nantinya akan datang dari sebuah aplikasi atau langsung membuka halaman otentikasi, dalam hal ini adalah halaman *login* yang kemudian mengisikan *username* dan *password*, kemudian pengguna akan mendapatkan sebuah token yang dapat digunakan untuk melakukan akses ke beberapa aplikasi yang sudah mendukung dan terhubung dengan sistem *Authorization Server*.

Alur secara umum dapat diilustrasikan seperti pada gambar 1 berikut ini :



Gambar 1: Diagram Alur Otorisasi Secara Umum

Langkah dari gambar 1 tersebut adalah seperti berikut, dengan urutan angka pada diagram menunjukkan proses atau tahapan yang terjadi :

1. Aplikasi Klien akan mengirimkan permintaan otorisasi dari pengguna yang biasanya berbentuk pasangan *username* dan *password*. Permintaan otorisasi ini bisa dilakukan langsung kepada pengguna dari Aplikasi Klien, atau diserahkan ke *Authorization Server* sebagai aplikasi penghubung.

2. Aplikasi Klien akan menerima otorisasi berupa pasangan *username* dan *password* dari Pengguna yang tentunya dengan kewenangan yang telah diberikan. Prosesnya akan menggunakan salah satu dari 4 (empat) bentuk otorisasi yang ada pada protokol OAuth 2. Tipe atau bentuk otorisasinya sudah disepakati bersama antara Aplikasi Klien dengan *Authorization Server*.

3. Aplikasi Klien melakukan permintaan Token Akses ke *Authorization Server* dengan membawa informasi otorisasi dari Pengguna.

4. *Authentication Server* memastikan bahwa informasi otorisasi yang dikirimkan sesuai dengan yang terdaftar, bila informasinya benar, maka akan diberikan Token Akses, namun bila tidak terdaftar atau informasinya salah, maka akan mengirimkan informasi bahwa pengguna tidak terdaftar atau tidak memiliki hak atau kewenangan.

5. Aplikasi Klien akan melakukan permintaan data ke *Resource Server* dengan menyertakan Token Akses sebagai pengenal otorisasinya.

6. Kemudian *Resource Server* akan melakukan verifikasi terhadap Token Akses yang telah diterima dari Aplikasi Klien, apabila Token Akses benar, maka *Resource Server* akan memberikan data atau informasi yang diminta.

Cara yang dianjurkan untuk digunakan saat Aplikasi Klien melakukan permintaan data otentikasi dari Pengguna (sebagaimana pada langkah 1 dan 2), adalah menggunakan *Authorization Server* sebagai perantara. Hal ini diharapkan menghilangkan kecurigaan dari sisi Pengguna bahwa *username* dan *password* yang diberikan tidak disimpan atau disalahgunakan oleh Aplikasi Klien untuk melakukan hal yang tidak diketahui oleh Pengguna, di sisi lain pun menyederhanakan pembangunan sebuah Aplikasi Klien tanpa perlu membuat sebuah halaman otentikasi lagi, karena halaman ini sudah disediakan oleh *Authorization Server*.

Pada tahap otorisasi nantinya akan ada beberapa jenis alur yang dapat dapat digunakan yang biasanya akan melihat dengan teknik apa Aplikasi Klien dibangun. Berikut adalah spesifikasi yang didefinisikan dalam Dokumen IETF RFC 6749 :

1. Authorization Code.

   Authorization code atau kode otorisasi didapat dari *Authorization Server* yang diposisikan sebagai penghubung antara Aplikasi Klien dan *Resource Server*. Aplikasi Klien tidak meminta informasi *username* dan *password* secara langsung dari Pengguna, melainkan Pengguna akan diarahkan ke *Authorization Server* untuk melakukan *login*, setelah berhasil maka Pengguna akan diarahkan kembali ke Aplikasi Klien beserta kode otorisasi.

   Karena Pengguna melakukan otentikasi langsung ke *Authorization Server*, informasi rahasia seperti *username* dan *password* tidak perlu diketahui oleh Aplikasi Klien.

   Skema atau spesifikasi ini memiliki keamanan yang cukup untuk melakukan otentikasi terhadap Aplikasi Klien, termasuk pengiriman Token Akses yang langsung ke Aplikasi Klien yang berpotensi terhadap bocornya Token Akses ke pihak lain.

   Skema atau spesifikasi ini biasanya diterapkan pada Aplikasi Klien yang mampu untuk menyimpan kode otorisasi dengan aman, misalnya seperti aplikasi server yang dibangun dengan PHP, Java, NodeJS, dan sebagainya, atau aplikasi berbasis *Desktop* atau *Mobile* yang dapat di*compile*.

2. User Password

   Pada spesifikasi ini, *username* dan *password* dapat langsung digunakan untuk mendapatkan Token Akses dari *Authorization server*. Karena hal ini tentunya Pengguna memiliki kepercayaan penuh terhadap Aplikasi Klien yang

digunakannya. Yang biasanya terjadi adalah aplikasi klien *mobile*, *desktop* atau *web* yang digunakan dibuat sendiri, sehingga kita sebagai pengguna meyakini bahwa aplikasi tersebut tidak akan berbuat curang.

Meskipun pada spesifikasi ini Aplikasi Klien menerima *username* dan *password* dari pengguna untuk kemudian di otentikasi oleh *Authorization Server* namun tetap saat kondisi *username* dan *password* sudah benar, Aplikasi Klien akan menerima Token Akses untuk melakukan akses ke *Resource Server*.

3. Implicit

   Spesifikasi alur *implicit* ini menyederhanakan alur yang digunakan oleh spesifikasi *Authorization Code* yang biasanya diimplementasikan bagi Aplikasi Klien yang dibangun bahasa interpreter seperti JavaScript.

   Perbedaannya dengan spesifikasi *Authorization Code* adalah bahwa *Authorization Server* tidak perlu memberikan kode otorisasi ke Aplikasi Klien, melainkan langsung diberikan Token Akses secara langsung.

   Ketika memberikan Token Akses ke aplikasi klien, *Authorization Server* tidak melakukan otentikasi terhadap Aplikasi Klien, dimana pada beberapa implementasi, identitas Aplikasi Klien diverifikasi melalui alamat *redirection* URI tempat dikirimnya Token Akses. Token Akses sendiri nantinya dapat dilihat oleh Pengguna atau aplikasi lain yang memiliki akses ke *user-agent*.

   Spesifikasi *implicit* ini meningkatkan responsif dan efisiensi karena mengurangi jumlah lalu lintas data untuk mendapatkan Token Akses.

4. Client Credentials

   Spesifikasi *client credentials* digunakan untuk mengamankan lalu lintas data dari aplikasi ke aplikasi, jadi tidak melibatkan Pengguna untuk melakukan akses data. Akses yang dilakukan dengan spesifikasi alur *client credentials*

memberikan kontrol penuh kepada Aplikasi Klien terhadap akses ke *Resource Server*, oleh karena itu spesifikasi ini tidak dapat melakukan pembatasan terhadap Pengguna tertentu.

3 PENGIDENTIFIKASIAN PARA PEMAKAI SISTEM

Karena tujuan penggunaan aplikasi ini spesifik kepada pemberian hak akses bagi Pengguna aplikasi nantinya, maka target dari pengguna atau pemakai sistem ini adalah Administrator Sistem yang diberikan kewenangan untuk membuat sebuah akun yang nantinya dipasangkan dengan sebuah Aplikasi.

Dengan target pengguna atau pemakai yang demikian, diharapkan aplikasi yang dibangun nantinya membutuhkan sebuah panduan atau prosedur penggunaan bagi Administrator Sistem.

Tidak cukup hanya dibekali dengan buku panduan atau buku petunjuk penggunaan, Administrator Sistem yang nantinya diberikan kewenangan perlu diberikan pelatihan agar sisi teknis yang perlu ditanyakan dapat langsung dicarikan solusinya.

4 PEMBENTUKAN LINGKUP SISTEM

Karena implementasi OAuth 2 ini cukup luas, maka perlu melakukan pembatasan implementasi terhadap alur mana yang digunakan dan bagaimana implementasinya dalam melakukan manajemen pengguna dan manajemen Aplikasi Klien.

Karena aplikasi yang akan dikembangkan kedepannya berbasis aplikasi *web* atau *mobile*, maka skenario yang digunakan akan menggunakan skenario *Authorized Code* dan *Implicit*, walau tidak menutup kemungkinan untuk mengimplemen-

tasikan skenario yang lain seperti *User Password* atau *Client Credentials*.

Untuk melakukan manajemen pengguna nantinya akan dilakukan aplikasi berbasis *web* yang melakukan akses terhadap *Resource Server* yang tentunya pengguna yang melakukan ini akan memiliki kewenangan tertinggi dalam sistem.

5 PENGUSULAN PERANGKAT LUNAK DAN PERANGKAT KERAS UNTUK SISTEM BARU

Beberapa hal terkait perangkat lunak untuk membangun sistem Otentikasi ini akan diuraikan seperti berikut berdasarkan perangkat yang dibutuhkan :

- Container

  Sistem yang akan digunakan akan menggunakan konsep *Microservice* yang menggunakan *Container*, *Container* yang digunakan yaitu Docker. Menggunakan versi *Community Edition* sehingga tidak memerlukan biaya untuk mendapatkannya, cukup mengunduh dari laman `www.docker.com`.

- Sistem Basis Data

  Karena menggunakan Docker, maka sistem basis data yang digunakan akan berasal dari *image* Docker yang bisa diunduh pula dari laman `hub.docker.com`, sistem basis data yang digunakan adalah Postgresql.

  Sistem basis data ini nantinya digunakan untuk menyimpan data-data mengenai Pengguna yang terdiri dari *username*, *password*, dan informasi hak aksesnya, serta menyimpan informasi detail mengenai data dari Aplikasi Klien.

- *Text Editor* atau IDE (*Integrated Development Environment*)

Untuk membangun sistem otentikasi ini, karena terdiri dari *front-end* dan *back-end*, maka kita membutuhkan aplikasi yang berbeda.

Aplikasi *text editor* atau IDE untuk membangun bagian *front-end* adalah yang memiliki karakteristik ringan, memiliki *auto-complete* untuk mempercepat pengetikan kode, dan karena akan dibangun dengan *framework* Angular, maka dibutuhkan *console* yang telah terintegrasi sehingga pada saat uji coba tidak harus berpindah jendela terlalu sering. Kemampuan seperti itu dimiliki oleh editor Visual Studio Code yang tersedia gratis dan dapat diunduh secara bebas di laman https://code.visualstudio.com/.

Membangun sistem pada bagian *back-end* akan dilakukan dengan bahasa pemrograman Kotlin, sehingga dibutuhkan *text editor* atau IDE yang mampu untuk memberikan kemudahan pada saat melakukan *coding* Kotlin. IDE yang digunakan harus cukup ringan dan mudah dioperasikan, sehingga pilihan *text editor* atau IDE yang memenuhi kriteria tersebut ada pada Intellij IDEA versi Community yang dapat diunduh dengan gratis dari alamat https://www.jetbrains.com/idea/

- Git

  Aplikasi ini digunakan untuk melakukan *versioning* sehingga memudahkan untuk melakukan pengembangan dan pemeliharaan. Aplikasi ini dapat diunduh secara gratis dari `git-scm.com`.

- Gitlab dan Gitlab-CI/CD

  Gitlab ini adalah aplikasi atau peladen untuk menempatkan hasil *versioning* yang biasanya digunakan untuk melakukan kolaborasi dalam sebuah Tim pengembang, kali ini kita gunakan sebagai tempat *backup* kode, dan tentunya, karena Gitlab memiliki fitur CI/CD (*Continuous Integration / Continuous*

Delivery), fitur ini digunakan untuk melakukan integrasi secara otomatis dari kode yang sedang dikembangkan ke kode yang telah stabil, fase berikutnya adalah menggunakan fitur ini untuk melakukan *deployment* atau pemasangan secara otomatis di peladen VPS (*Virtual Private Server*) sebagai tempat *production*.

- Pustaka (*Library*)

  Pustaka atau *library* untuk membangun sistem ini pun memiliki 2 (dua) bagian, yaitu bagian *front-end* dan bagian *back-end*, detail dari pustaka yang digunakan pada tiap-tiap bagian ini yaitu seperti berikut ini :

  - Pustaka yang digunakan pada bagian *front-end* yaitu :
    - Angular 5, pustaka ini digunakan pada seluruh bagian *front-end* dari aplikasi, yang menjadi alat untuk berkomunikasi dengan bagian *back-end*, seluruh rangka aplikasi bagian *front-end* akan diatur dengan Angular 5. Termasuk di dalamnya adalah `angular-cli` yang digunakan untuk melakukan otomasi saat uji coba aplikasi yang dibangun dengan Angular.
    - Bootstrap, pustaka ini memberikan kesan yang menarik pada tampilan aplikasi *web* sehingga pengguna nantinya akan lebih nyaman untuk melakukan operasi masukkan data dan membaca informasi pada sistem aplikasi ini.
    - NPM, nama panjang dari aplikasi ini adalah *Node Package Manager*, digunakan untuk otomasi pengumpulan pustaka yang akan digunakan pada aplikasi *front-end*.
  - Pustaka yang digunakan pada bagian *back-end* adalah :
    - Spring Boot

Spring Boot adalah *framework* yang digunakan sebagai dasar aplikasi bagian *back-end* yang memiliki beberapa fitur seperti: *servlet* yang telah terintegrasi di dalamnya, sehingga tidak perlu lagi mengubah sebuah *project* menjadi *file* `war` untuk di *publish*, dapat dikonfigurasi sesuai kebutuhan berdasarkan kegunaan aplikasi, termasuk adanya *dependency injection* yang memungkinkan kita tidak perlu membuat instan sebuah objek (karena sudah diatur oleh *Spring Boot* dan cukup mendeklarasikan dengan menggunakan anotasi.

* Spring Data JPA

  Spring Data JPA adalah paket lain dari *framework* Spring yang fungsinya menghubungkan aplikasi dengan sistem basis data. Dengan pustaka ini akan dimudahkan melakukan operasi basis data dasar tanpa menggunakan bahasa SQL (*Structured Query Language*) sama sekali.

* Spring REST

  Spring REST pun adalah bagian dari paket *framework* Spring yang fungsinya mempermudah dalam membangun sebuah *server webservice* dengan model *Restful*.

* Driver OJDBC

  Pustaka ini digunakan sebagai jembatan penghubung antara aplikasi dengan basis data Oracle, perbedaan dengan penggunaan Spring Data JPA adalah Spring Data JPA digunakan pada saat melakukan operasi transaksi terhadap data pada sistem basis data, sedangkan *driver OJDBC* adalah jembatan yang menerjemahkan perintah yang dikirimkan oleh bahasa pemrograman Kotlin ke bahasa perintah pada sistem basis data.

* Jackson

Pustaka ini berfungsi sebagai penerjemah atau yang melakukan konversi otomatis dari objek Kotlin menjadi data dengan pola struktur JSON (*JavaScript Object Notation*), dan sebaliknya, yaitu dari JSON menjadi objek Kotlin.

Sedangkan kebutuhan akan perangkat kerasnya adalah seperti berikut ini :

- *Server*

  Perangkat keras ini akan menjadi tempat Docker berjalan, karena bentuk *Microservices* ini akan terdiri dari layanan-layanan kecil, maka cukup membutuhkan sebuah peladen (*server*) dimana nantinya Docker akan memiliki banyak *images* yang mampu berjalan sesuai dengan layanannya masing-masing.

- *Router*

  Perangkat keras ini akan mengatur alur data pada jaringan dan menghubungkan 2 (dua) atau lebih alamat jaringan yang berbeda. Perangkat ini yang nantinya akan mengarahkan permintaan akses aplikasi web yang telah dibuat ke alamat *server* aplikasi.

- *Modem* dan Akses Internet

  Perangkat *Modem* ini yang akan menghubungkan jaringan internal dengan jaringan internet agar *server* aplikasi yang berada pada jaringan internal dapat diakses oleh publik.

6 PEMBUATAN ANALISIS UNTUK MEMBANGUN SENDIRI ATAU MEMBELI APLIKASI

Keputusan untuk membangun aplikasi sendiri atau membeli tentunya akan berdasarkan kepada waktu, kualitas, dan layanan.

Dari sisi waktu, tentunya dengan membeli aplikasi yang sudah jadi untuk sistem informasi semacam ini akan lebih cepat selesai karena biasanya sudah tersedia *template* atau bahkan aplikasi sudah jadi dan dipasarkan secara umum. Apabila kita membangun aplikasi ini sendiri, tentu dibutuhkan waktu yang lebih lama karena perlu waktu untuk pembelajaran terhadap model *web services* terlebih dahulu, bagaimana teknis *server web services* menerima data, bagaimana data diolah, dan bagaimana *server* harus melakukan respon data terhadap *request* yang datang. Kita pun perlu mempelajari bagaimana *framework* Angular di sisi *front-end* bekerja. Bagaimana *binding data* pada Angular, *flow* program pada Angular, dan bagaimana Angular berkomunikasi melakukan *request* dan mengolah respon dengan *server web service* di bagian *back-end*. Ditambah lagi adalah pemahaman mengenai implementasi OAuth2 menggunakan *framework* Spring boot.

Dari sisi kualitas, tentunya bisa dilihat dari banyaknya *bugs* atau permasalahan kode program yang nantinya muncul. Beberapa perusahaan yang telah bergerak di bidang pembangunan sistem informasi atau aplikasi komputer biasanya akan memiliki *bugs* atau kesalahan program yang sangat sedikit, yang tentunya ini didapatkan dari pengalaman dalam membangun dan memelihara sebuah sistem informasi. Apabila membangun sistem informasi ini secara mandiri, tentunya akan mengalami beberapa kendala dengan kualitas aplikasi yang dihasilkan, namun akan menjadi investasi jangka panjang bagi proses pemeliharaan sistem aplikasi dan pembaruannya tanpa harus menambahkan biaya tambahan atau biaya peme-

liharaan yang biasanya muncul saat kita membeli aplikasi atau sistem informasi dari pihak lain.

Sedangkan dari sisi layanan, seperti hal yang biasa diajukan oleh sebuah *software house* atau pembuat aplikasi, maka ada 2 (dua) model dasar yang biasanya ditawarkan, yaitu model kontrak putus, artinya begitu sistem sudah jadi dan dipasang, akan ditunjuk 1 (satu) atau lebih orang untuk melakukan *maintenance* dasar, atau bahkan akan diserahkan pula kode sumbernya bila suatu saat akan dilakukan pembaruan atau perbaikan secara mandiri. Model yang kedua yaitu sistem kontrak layanan jasa, yang biasanya sistem aplikasi akan diberikan secara gratis, namun tidak diberikan kode sumbernya, melainkan akan dilakukan perbaikan atau pembaruan secara berkala bila diketahui ada permasalahan pada sistem yang berjalan.

Dengan membangun sendiri sistem otentikasi ini, maka tentunya layanan penuh akan dikendalikan oleh personil yang menangani bidang teknologi informasi tanpa tambahan biaya apapun selain dari menggunakan 2 (dua) jaringan internet atau lebih sebagai strategi untuk menjaga konektivitas.

7 PEMBUATAN ANALISIS BIAYA / MANFAAT

Analisis biaya dan manfaat tentunya tidak jauh dari tujuan dibuatnya sebuah sistem otentikasi ini. Sebagaimana disebutkan pada bagian awal dokumen studi kelayakan pendahuluan ini, bahwa sistem aplikasi yang dibangun harus mampu melakukan otentikasi dan otorisasi terhadap pengguna yang melakukan akses ke *Resource Server* melalui Aplikasi Klien.

Dalam membangun sebuah sistem, tentunya akan menggunakan sumber-sumber daya yang nantinya diharapkan mendapatkan manfaat seperti tujuan dari diba-

ngunnya sistem aplikasi. Jika manfaat-manfaat yang dihasilkan tersebut lebih kecil dari sumber daya yang dikeluarkan, maka sistem aplikasi ini dikatakan tidak bernilai atau tidak layak untuk dibangun, namun bila manfaat yang dihasilkan lebih besar dari sumber daya yang dikeluarkan, maka sistem informasi atau aplikasi ini boleh dikatakan layak untuk dibangun.

Ada dua komponen yang diperlukan untuk melakukan analisis biaya dan manfaat terhadap sebuah sistem aplikasi, sesuai namanya yaitu :

1. Komponen Biaya, dan
2. Komponen Manfaat

Secara detail, masing-masing komponen tersebut adalah seperti berikut ini :

1. **KOMPONEN BIAYA**

   Biaya yang berhubungan dengan pengembangan sistem aplikasi dapat diklasifikasi ke dalam 4 (empat) kategori utama, yaitu :

   (a) Biaya Pengadaan

   Biaya pengadaan tentu saja adalah seluruh biaya yang dikeluarkan untuk membangun sistem informasi yang diinginkan, mulai dari perangkat keras, perangkat lunak, serta dukungan sumber daya lain yang dibutuhkan. Dilihat dari kondisi perangkat keras yang seluruhnya telah tersedia, maka biaya yang perlu dikeluarkan untuk hal ini adalah nihil, dengan kata lain, tanpa biaya pengadaan.

   Begitu juga apabila kita melihat kebutuhan akan perangkat lunak yang akan digunakan, maka dapat dikatakan kebutuhan akan biaya pengadaan perangkat lunak pun nihil. Semua perangkat lunak dapat diunduh secara gratis dan bebas dari internet.

Sumber daya lain yang dibutuhkan adalah, dengan membangun sendiri sistem informasi ini, maka tentunya kebutuhan biaya tidak diperlukan lagi karena personil yang akan membangun sistem informasi ini adalah fungsional pranata komputer yang tugas pokok dan fungsinya memang berada pada bidang tersebut.

Kondisi biaya dari pengadaan perangkat lunak akan berbeda bila dilakukan oleh pihak lain, seperti disebutkan sebelumnya bahwa, secara garis besar, ada 2 (dua) jenis penjualan perangkat lunak yang dilakukan oleh *software house*, yaitu sistem kontrak lepas, artinya membayar di awal masa jual beli, kemudian biasanya akan diberikan rentang waktu tertentu sebagai masa uji coba dan uji kelayakan sebuah sistem aplikasi, setelahnya diserahkan ke Pemerintah Daerah untuk melakukan perawatan mandiri, dan model yang kedua adalah dengan menjual jasa, artinya perangkat lunak yang diberikan gratis tetapi dengan syarat melakukan kontrak jasa dengan rentang waktu minimal yang sering dikatakan sebagai *services*, dimana selama rentang waktu ini akan diberikan perbaikan-perbaikan *bugs* dan pembaruan aplikasi bila ada, namun bila kontrak tidak diperpanjang, maka secara otomatis aplikasi tidak akan dapat digunakan. Dan rata-rata biaya dari pengadaan perangkat lunak yang dikeluarkan akan mencapai puluhan juta, tentu apabila menggunakan model yang kedua, biaya pengadaan menjadi tidak terhingga karena tiap tahunnya harus melakukan perpanjangan kontrak.

(b) Biaya Persiapan Operasi

Biaya persiapan operasi pun tidak diperlukan bila sistem informasi akan dibangun secara mandiri oleh fungsional pranata komputer, karena segala konfigurasi yang dilakukan pada perangkat keras, membangun sistem aplikasi dengan *tools* yang gratis, dapat dilakukan oleh fungsio-

nal pranata komputer itu sendiri.

Pada kondisi dimana sistem informasi dibeli dari pihak lain, maka biasanya biaya persiapan operasi sudah diperhitungkan dan digabungkan dengan biaya pengadaan sebuah sistem informasi.

(c) Biaya Proyek

Biaya proyek adalah biaya yang dikeluarkan untuk pengembangan sistem informasi termasuk penerapannya, namun diluar biaya pengadaan dan biaya persiapan operasi. Yang termasuk kedalam biaya proyek diantaranya adalah biaya dalam tahap analisis sistem, biaya pada tahap desain sistem, dan biaya pada tahap penerapan sistem.

Bila pengadaan sistem informasi dilakukan oleh pihak lain, maka ketiga biaya tersebut akan muncul yang biasanya sudah digabungkan dengan biaya pengadaan diatas, namun bila menggunakan tenaga fungsional pranata komputer, maka seluruh biaya proyek tersebut tidak diperlukan lagi.

(d) Biaya Operasi dan Biaya Perawatan

Biaya operasi dan biaya perawatan adalah biaya-biaya yang dikeluarkan untuk mengoperasikan sistem supaya sistem dapat beroperasi. Pada kasus dimana pihak lain dibeli sistem informasinya dengan sistem kontrak putus, maka biaya operasi tidak diperlukan lagi karena sistem biasanya akan diserahkan ke personil di Pemerintah Daerah untuk melakukan tugas Administrasi Sistem. Namun bila menggunakan sistem pembelian jasa, maka tiap tahunnya akan diperlukan biaya operasi dan biaya perawatan guna perbaikan *bugs* dan pembaruan aplikasi.

Namun bila sistem ini dibangun oleh fungsional pranata komputer, maka kejadiannya akan sama seperti sistem kontrak putus untuk biaya

operasi dan biaya perawatan akan dilaksanakan sepenuhnya oleh seorang fungsional pranata komputer.

2. **KOMPONEN MANFAAT**

Manfaat yang didapat dari sistem otorisasi yang dibangun dapat dijabarkan sebagai berikut :

(a) Manfaat Keamanan Data

Dengan dibangunnya sistem otorisasi ini, maka diharapkan untuk aplikasi-aplikasi yang nantinya dibangun tidak perlu menambahkan modul otentikasi dan otorisasi di tiap aplikasinya, melainkan cukup menggunakan dan memanfaatkan sistem otorisasi ini.

Data yang disimpan pada sistem otorisasi ini pun tidak mudah untuk dilihat bahkan oleh Administrator Sistem sekalipun, karena *password* dan informasi yang bersifat rahasia akan di enkripsi pada sistem basis data.

(b) Manfaat Kesederhanaan

Tentunya dengan adanya sistem otorisasi yang dibangun, nantinya tiap aplikasi akan lebih sederhana melakukan tugasnya tanpa dibebani dengan tugas otorisasi dan otentikasi, tugas ini cukup diserahkan ke sistem otorisasi yang saat ini akan dibangun.

(c) Manfaat Efisiensi

Dengan adanya layanan dari sistem otorisasi ini, akan mengurangi proses penggunaan memori peladen karena otorisasi akan berada dalam *container* Docker sehingga layanan dapat berada pada 1 (satu) peladen fisik yang sama dengan implementasi aplikasi lainnya.

8 PENGKAJIAN TERHADAP RESIKO PROYEK

Sama seperti jenis proyek yang lain, proyek ini pun pasti memiliki beberapa resiko yang apabila dikelola akan menjadikan nilai tambah bagi proyek bersangkutan.

Dengan adanya sistem otentikasi baru ini, diharapkan seluruh aplikasi yang akan dibangun kedepannya dapat memanfaatkan proses otentikasi di sistem otentikasi ini.

Untuk aplikasi atau sistem yang telah terbangun dan menggunakan otentikasi dan otorisasi masing-masing, diharapkan dilakukan migrasi bertahap untuk memanfaatkan sistem otentikasi yang baru ini.

Kondisi jaringan internet yang tersedia pun harus dapat dijaga ketersediaannya, seperti dengan menggunakan 2 (dua) atau lebih jaringan internet dari 2 (dua) atau lebih penyedia jaringan internet, yang dimaksudkan apabila ada kendala pada sebuah jaringan internet, maka jaringan internet yang lain akan melakukan fungsinya sebagai pengganti jaringan yang rusak.

9 PEMBERIAN REKOMENDASI UNTUK MENERUSKAN ATAU MENGHENTIKAN PROYEK

Rekomendasi yang dapat diberikan dengan melihat beberapa pertimbangan dan kebutuhan akan sistem otorisasi menggunakan konsep OAuth 2, maka perlunya meneruskan kegiatan ini sebagai titik awal langkah pengamanan data bagi tiap aplikasi yang dibangun yang nantinya dapat menerapkan atau mengimplementasikan konsep Microservices dimana tiap aplikasi yang telah dibangun dapat dise-

derhanakan kedalam sebuah aplikasi holistik yang menangani segala kebutuhan pada bagian yang menangani Pajak Daerah di Kabupaten Brebes.