Universitas Esa Unggul

Smart, Creative and Entrepreuerial

Modul : 10

CQI – 611 – Arsitektur Berbasis Layanan

Oleh:
7841 – Diah Aryani,ST.,M.Kom
Prodi : Teknik Informatika

ESB CONCEPTS

A. Tujuan Pembelajaran

Mahasiswa Mampu Memahami (Enterprise Service Bus) ESB CONCEPTS

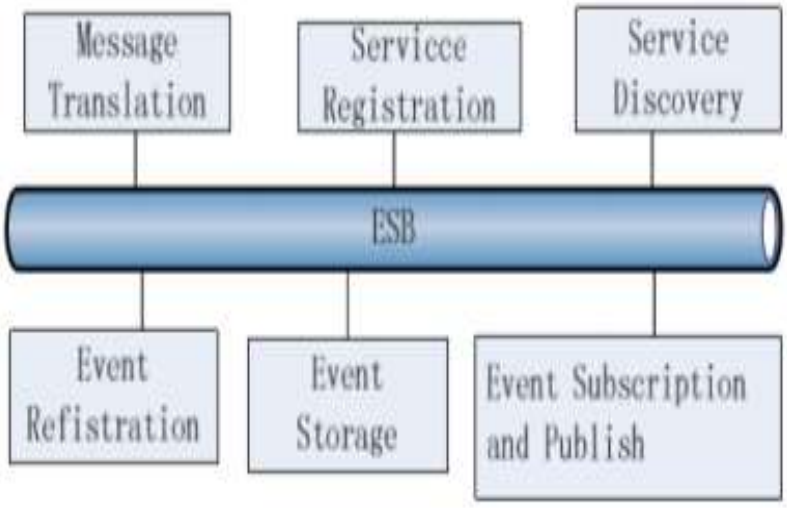
B. Pengertian ESB CONCEPTS

Enterprise Service Bus atau ESB adalah arsitektur yang fleksibel dan dapat dioperasikan untuk integrasi antar apliksi dan layanan, ini dapat mengurangi jumlah dan kompleksitas antarmuka dalam arsitektur SOA, meningkatkan proses bisnis, mengitegrasikan aplikasi yang heterogen dan juga dapat meningkatkan nilai aset data.

ESB merupakan salah satu pilar SOA, pilar lainnya adalah WS dan BPEL. ESB merupakan infrastruktur untuk koneksi layanan SOA dan pertukaran pesan. Fungsionalitas utama ESB adalah melakukan rute, transformasi protokol serta transformasi pesan atau data. Dengan adanya fungsi transformasi protokol dan pesan pada ESB ini maka ketidaksesuaian protokol dan data dapat diatasi. ESB juga memudahkan koneksi dan mediasi, menyederhanakan integrasi serta memudahkan penggunaan ulang komponen-komponen layanan, sehingga skalabilitas integrasi menjadi tinggi.

ESB merupakan kombinasi dari teknologi middleware tradisional, termasuk XML dan web service, ini menyediakan mekanisme pertukaran pesan berbasis standar dan komponen lainnya melalui adaptor dan antarmuka standar untuk memenuhi kebutuhan integrasi aplikasi untuk perusahaan besar (Ma, Yao, & Shan, 2017). Pendekatan service bus untuk integrasi menggunakan teknologi yang mnyediakan bus agar aplikasi satu dengan yang lainnya dapat terintergrasi. 16 Beragam aplikasi tidak berkomunikasi satu sama lain secara langsung melainkan berkomunikasi melalui backbone middleware SOA. Fitur aristektur ESB yang paling membedakan adalah sifat terdistribusi dari topologi integrasi (Kurniawan & Ashari, 2016).

Prinsip dari ESB adalah melalui teknologi integrasi standar, SOA, web service dan XML digabungkan menjadi arsitektur terdistribusi yang mudah digunakan dan dikelola. ESB menyediakan funsionalitas aplikasi perangkat lunak yang ada untuk menghubungkan bisnis internal dan lintas perusahaan dan juga berfungsi sebegai jembatan antara layanan konsumen dan penyedia layanan seperti yang ditunjukkan pada Gambar Prinsip teknik dasar ESB berikut:



Gambar Prinsip Teknik Dasar ESB

Web Service

World Wide Web Concortium (W3C) menjelaskan dalam dokumen resminya bahwa web service adalah sebuah sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas antar mesin dan dapat berinteraksi melalui sebuah jaringan. Pengertian lain web service adalah layanan yang disediakan melalui media internet dengan menggunakan sistem pengiriman pesan XML sebagai standarisasinya. (Kurniawan & Ashari, 2016).

Web service menggunakan model arsitektur SOA yang memiliki sub fungsi dari perangkat lunak, teknologi yang dapat digunakan untuk framework web service mencakup Extensible Markup Language, Simple 17 Object Access Protocol, Web Service Description Language dan Universal Description, Discovery, and Integration. (Kaewrattanapat & Nookhong, 2017).

a. Extensible Markup Language Extensible Markup Language atau XML, bahasa markup berdasarkan pengertian dari W3C, XML akan mendefinisikan data terstruktur yang dapat ditransplantasikan. Ini adalah titik kunci dari layanan Web yang menggunakan XML untuk proses transaksi dan pengolahan data.
b. Simple Object Access Protocol Simple Object Access Protocol atau SOAP adalah protokol komunikasi sederhana berdasarkan XML dalam lingkungan distribusi. Ini digunakan untuk berkomunikasi antar data sesuai dengan bentuk objek antara program aplikasi. Pada protokol komunikasi SOAP, HTTP dianggap sebagai protokol komunikasi, RPC dianggap sebagai jalur panggilan konsistensi, XML dianggap

sebagai format transmisi data. Layanan penyedia dan pelanggan layanan melakukan interaksi komunikasi di Internet melalui firewall untuk mendukung protokol standar SMTP, FTP, TCP, POP3 dan seterusnya. SOAP benar-benar independen dari pabrikan manapun.

c. Web Service Description Language Web Service Description Language atau WSDL dapat mendeskripsikan layanan Web dengan menggambarkan antarmuka layanan web melalui format yang telah memiliki standarisasi berupa 18 XML. WSDL menstandardisasi bagaimana layanan web merepresentasikan parameter input dan output dari suatu permintaan.

d. Universal Description, Discovery, and Integration Universal Description, Discovery, and Integration atau UDDI sebuah kerangka kerja platform yang independent untuk mendeskripsikan layanan- layanan, menemukan, dan mengintegrasikan layanan dengan menggunakan internet (Kumar, 2016). UDDI berkomunikasi melalui SOAP. UDDI adalah sebuah direktori dari web services dimana antarmuka UDDI adalah WSDL.

Sesuai dengan definisinya, web service merupakan kumpulan fungsi yang dapat diakses oleh sistem lain, dengan demikian web service merupakan suatu sistem yang tidak memiliki user interface. Sebagai konsekuensinya, untuk mengimplementasikan suatu web service diperlukan aplikasi lain yang dapat mengakses fungsi-fungsi yang terdapat dalam sebuah web service. Dalam hal ini aplikasi tersebut dapat disebut dengan agent. Agent inilah yang melakukan pengiriman dan penerimaan massage dari suatu web serivce ketika melakukan akses terhadap web service.

STUDY KASUS ESB

IJCCS, Vol.10, No.2, July 2016, pp. 137~148
ISSN: 1978-1520

Penerapan *SOA* menggunakan *Enterprise Service Bus* pada Proses Pemeriksaan Status Perizinan Pemerintah Kabupaten Sleman

Kabul Kurniawan*[1], Ahmad Ashari[2]
[1]Jurusan Ilmu Komputer, FSM Universitas Diponegoro, Semarang
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
e-mail: ***[1]kabulkurniawan@gmail.com**, [2]ashari@ugm.ac.id

Abstrak

Penerapan web service secara point-to-point dinilai sudah tidak relevan jika jumlah service yang digunakan semakin banyak dan kompleks. Standar penerapan web service yang berbeda-beda sering menjadi masalah apabila dibutuhkan integrasi dan komunikasi antar web service. Salah satu cara yang dapat diterapkan untuk melakukan pengelolaan integrasi sistem informasi adalah menggunakan Enterprise Service Bus (ESB). ESB merupakan infrastruktur perangkat lunak yang dapat menjadi solusi masalah kompleksitas integrasi n-to-n, dimana n adalah jumlah aplikasi yang diintegrasikan. Konsep ESB sangat mendukung implementasi Service Oriented Architecture (SOA).

Penelitian ini membahas mengenai penerapan ESB di lingkungan pemerintahan khususnya di Pemerintah Kabupaten Sleman, Indonesia. Beberapa fungsionalitas ESB seperti routing dan transformasi diterapkan untuk membantu menyelesaikan permasalahan integrasi antara instansi pemerintah dengan masyarakat yaitu pada proses pemeriksaan status permohonan izin oleh pemohon (masyarakat) pada sistem informasi perizinan pada Badan Penanaman Modal dan Perizinan Terpadu (BPMPT) Pemerintah Kabupaten Sleman secara realtime.

Hasil pengujian menunjukkan bahwa ESB dapat digunakan sebagai mediator integrasi informasi pada proses pemeriksaan status permohonan izin. Selain itu hasil pengujian unjuk kerja menunjukkan bahwa rata-rata waktu eksekusi (Avarage Execution Time) untuk semua operasi pada service mencapai 0,077 detik. Selanjutnya setiap detik rata-rata melakukan eksekusi sebanyak 12,89 kali dengan data sebesar 11513,75 byte atau setara dengan 11,24 KB. Hasil tersebut menunjukkan bahwa waktu eksekusi dianggap kecil sehingga service yang dikembangkan dianggap tidak mengganggu proses transaksi yang sedang berjalan.

Kata kunci—*Enterprise Service Bus (ESB), Integrasi Sistem Informasi, E-Government, Web Service*

Abstract

The Implementation of web services using point-to-point method considered to be irrelevant weather the number of services are growing up rapidly and more complex. In the other hand, the differences of web service implementation standard is also being a serious problem when the integration and communication among web services are needed.One way that can be applied to provide management information system integration among others is the implementation of Enterprise Service Bus (ESB). ESB is an infrastructure that can be the solution of n-to-n integration complexity problem, where n is the number of applications. ESB concept strongly supports the implementation of Service Oriented Architecture (SOA).

This research will discuss about the implementation of ESB in government area, especially in Sleman, sub-district of Daerah Istimewa Yogyakarta local government, Indonesia. Some of ESB functionality such as Routing and Transformation are applied to solve the integration

Received October 26th,2016; Revised January 4th, 2017; Accepted January 10th, 2017

problem between Government to Public (G2P) in case of checking lisence process at Badan Penanaman Modal dan Izin Terpadu (BMPT) Sleman Sub-Distric.

The results show that the ESB can be used as the mediator of information integration in case of checking lisence process. Beside that, the results of performance testing shows that the average execution time (AET) for service operations reach up to 0,085 seconds. Furthermore, the system has an average amount of execution about 11.99 times and produces data about 15746.86 bytes or equivalent to 15.38 KB per second. These results indicate that the execution time is considered to be small enaugh so that the services will not interfere the process of the transactions.

Keywords—*Enterprise Service Bus (ESB), System Information Integration, E-Government, Web Service*

1. PENDAHULUAN

Badan Penanaman Modal dan Perizinan Terpadu (BPMPT) merupakan satu dari sekian banyak instansi di Kabupaten Sleman yang menjalankan tugas dalam melayani masyarakat yaitu melayani permohonan perizinan di Kabupaten Sleman. Dalam proses pelayanannya instansi ini menggunakan perangkat lunak sistem informasi untuk melakukan pengolahan data perizinan. Setiap permohonan yang masuk setiap harinya dicatat dan dimasukkan pada sistem informasi tersebut untuk selanjutnya diproses sesuai dengan prosedur yang ada hingga proses perizinan tersebut selesai dilakukan.

Permasalahan yang muncul adalah bagaimana mendapatkan informasi mengenai status proses yang dilakukan oleh BPMPT sehingga masayarakat (pemohon) dapat mengetahui status permohonan izin mereka. Status tersebut antara lain status permohonan diregistrasi, status permohonan diverifikasi,status survey lokasi, status pembuatan surat keterangan retribusi daerah (SKRD) dan status penerbitan SK. Untuk dapat mengetahui status tersebut diperlukan sebuah aplikasi yang dapat terintegrasi dengan sistem informasi perizinan. Proses integrasi tersebut dapat dilakukan dengan menyediakan web *service* untuk dapat mengakses data perizinan tersebut.

Seiring dengan sistem informasi dilingkungan Pemerintah Kabupaten Sleman, maka kuantitas dan kompleksitas sistem informasi di lingkungan Pemerintah Kabupaten Sleman mengalami peningkatan. Hal ini meningkatkan pula kuantitas serta kompleksitas *web service* yang dibutuhkan. Penerapan *web service* yang selama ini dikembangkan secara *point-to-point* dinilai sudah tidak relevan karena jumlah *service* yang semakin banyak dan kompleks menjadi sulit untuk dimonitor dan dikelola. Standar penerapan *web service* yang berbeda-beda juga sering menjadi kendala apabila dibutuhkan integrasi dan komunikasi antar *web service.*

Salah satu cara yang dapat digunakan adalah menggunakan *Enterprise Service Bus* (ESB). ESB merupakan infrastruktur perangkat lunak yang dapat menjadi solusi dari masalah kompleksitas integrasi *n-to-n*, dimana n adalah jumlah aplikasi. Integrasi n-aplikasi dengan n-aplikasi mengakibatkan n x n usaha integrasi. Konsep ESB sangat mendukung *implementasi Service Oriented Architecture* (SOA). Interaksi antar komponen *service* dilakukan melalui mediator ESB, sehingga hal ini dapat menghadirkan sifat *loose-coupling* pada interaksi antar layanan dan memudahkan pengelolaan pada sistem terdistribusi. Beberapa fungsionalitas ESB seperti *routing* dan transformasi dapat diterapkan untuk mengatasi permasalahan integrasi aplikasi dengan database perizinan.

2. METODE PENELITIAN

2.1 Tinjauan Pustaka

Penelitian yang dilakukan oleh [1] menerapkan *Enterprise Service Bus* (ESB) sebagai *middleware* berbasis SOA. Penelitian tersebut membahas fungsional ESB dalam melakukan rute, transformasi protokol serta transformasi pesan atau data. Penelitian tersebut juga

mengatakan bahwa dengan fungsi transformasi protokol dan pesan pada ESB ini maka ketidaksesuaian protokol dan data dapat diatasi. ESB juga memudahkan koneksi dan mediasi, menyederhanakan integrasi serta memudahkan penggunaan ulang komponen-komponen layanan, sehingga skalabilitas integrasi menjadi tinggi. Pada penelitian tersebut telah berhasil melakukan integrasi beberapa *web service*s yang berasal dari *eBay*, *Yahoo*, *Amazon* dan *Paypal* dengan menggunakan ESB sebagai *middleware* integrasi.

Selanjutnya, [2] melakukan penelitian untuk mengintegrasikan aplikasi *Android* dan komputer *server* sebagai solusi *mobile commerce* dan CRM. Penelitian tersebut menerapkan ESB menggunakan WSO2 ESB. WSO2 ESB dapat berfungsi sebagai *mediation* dan *transformation* di dalam integrasi antar aplikasi. *Transformation* merupakan peran fungsi yang dimiliki ESB di dalam mentransformasikan *database* suatu aplikasi ke dalam format standar yang dapat dipahami oleh berbagai jenis aplikasi secara umum. Format standar yang digunakan di sini adalah XML atau *Extensible Markup Language*. *Mediation* merupakan fungsi dari ESB untuk mengkomunikasikan *database* dari dua atau lebih aplikasi dengan platform yang berbeda. Di sini, *database* aplikasi pada komputer *server* yang telah diubah menjadi XML akan diakses oleh aplikasi *Android* menggunakan protokol SOAP sehingga dapat dibaca oleh aplikasi *Android* tersebut. Pada penelitian tersebut disebutkan bahwa pemanfaatan teknologi ESB sebagai perantara aplikasi *m-commerce* Android dengan aplikasi *desktop* merupakan solusi integrasi yang efektif untuk komunikasi data antara dua aplikasi dengan platform berbeda melalui jaringan internet.

Penelitian yang dilakukan oleh [3] membahas mengenai sistem informasi yang bervariasi dari sisi platform, *database* dan bahasa pemrograman yang ada di lingkungan pemerintah kabupaten Sleman. Sistem-sistem tersebut kurang memperhatikan aspek integrasi. Sistem yang tidak terintegrasi mengakibatkan terjadinya ketidaksinkronan ketika beberapa instansi melakukan input data yang sama (redudansi data). Dari permasalahan tersebut peneliti melakukan penelitian untuk mengintegrasikan data penduduk di Kantor Kependudukan dan Catan Sipil Kabupaten Sleman sehingga bisa diakses oleh instansi lain yang membutuhkan data tersebut secara sinkron. Peneliti merancang *web service* dengan format XML sebagai format pertukaran data dengan pendekatan *Service Oriented Architecture* (SOA).

2.2 Service Oriented Architecture (SOA)

Erl, mendefinisikan *Service Oriented Architecture* (SOA) adalah sebuah permodelan perangkat lunak yang dibangun dengan pendekatan *service oriented*. *Service oriented* sendiri merupakan sebuah pendekatan yang memiliki visi ideal di mana setiap *resource* dari perangkat lunak terpartisi secara bersih satu sama lain. Setiap *resource* ini disebut dengan *service*. *Service* ini merepresentasikan sebuah *business logic* atau *automation logic* dalam sebuah sistem besar [4]. Setiap *service* memiliki otonomi sendiri yang membuatnya tidak tergantung satu sama lain. Setiap *service* dapat berkomunikasi satu sama lain melalui sebuah protokol yang sudah terstandardisasi sehingga memudahkan untuk melakukan integrasi *service* baru dan penyusunan ulang kumpulan *service* disebabkan proses bisnis yang berubah

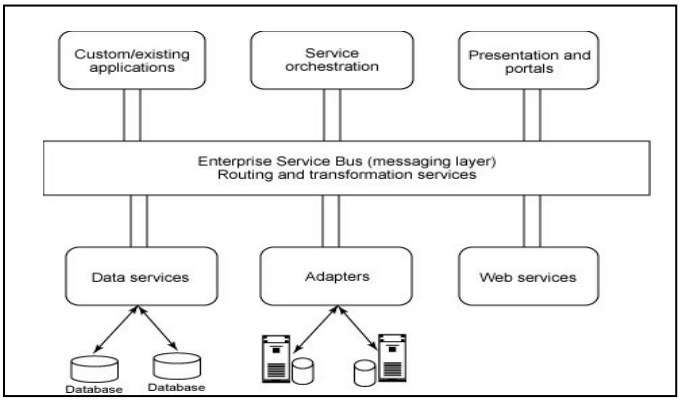
2.3 Enterprise Service Bus (ESB)

Enterprise Service Bus adalah sebuah platform integrasi berbasis standar yang mengabungkan pesan, layanan web, transformasi data dan routing cerdas untuk menghubungkan dan dan mengkoordinasikan interaksi sejumlah aplikasi yang beragam di suatu perusahaan dengan integritas transaksional [5].

Pendekatan *service bus* untuk integrasi adalah menggunakan teknologi yang menyediakan *bus* untuk integrasi aplikasi. Aplikasi-aplikasi yang berbeda tidak berkomunikasi satu sama lain secara langsung melainkan berkomunikasi melalui *backbone middleware* SOA. Fitur arsitektur ESB yang paling membedakan adalah sifat terdistribusi dari topologi integrasi. ESB merupakan sekumpulan *middleware* layanan-layanan yang menyediakan kemampuan integrasi. *Middleware* layanan-layanan ini merupakan jantung arsitektur ESB yang menempatkan pesan untuk dapat diarahkan dan ditransformasikan [6].

2.3.1. Arsitektur ESB

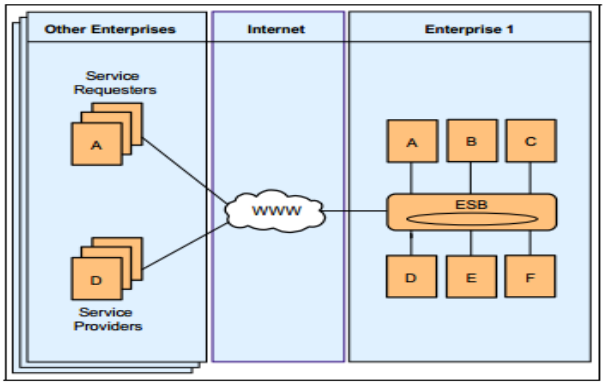
Arsitektur umum dari ESB dengan komponen yang terkoneksi dapat dilihat pada Gambar 1. Komponen dapat mengambil peran penghasil layanan atau pemakai layanan. Layanan-layanan dapat berupa komponen spesial seperti mesin orkestrasi, adapter untuk sumberdaya data atau adapter untuk sistem eksternal dengan transformasi pesan atau konversi transport protokol. ESB melakukan mediasi pesan antar komponen, memutuskan lokasi untuk rute pesan, dan transformasi pesan. ESB memerlukan memori persisten seperti terkoneksi dengan basisdata [7].



Gambar 1. Arsitektur ESB secara umum [6,7]

2.3.2. Pola Akses Layanan ESB

Dalam penelitian yang ditulis oleh Keen, dijelaskan bahwa ESB memiliki pola akses layanan seperti yang ditunjukkan pada Gambar 2. Pada Gambar tersebut diasumsikan bahwa internet (WWW) digunakan sebagai media komunikasi dan interaksi antar *enterprise*. Namun pada Gambar tersebut internet dapat juga diganti dengan jaringan lain misalnya jaringan LAN (*Local Area Network*). ESB dapat diekspose oleh eksternal *service requeste*r maupun *service* provider. Sehingg ESB harus memiliki kemampuan *routing, security* serta transformasi protokol yang baik [8].



Gambar 2 Pola Akses Layanan ESB [8].

2.4. Web Service

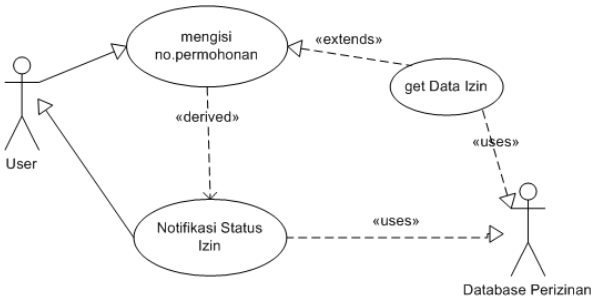
World wide web concortium (W3C) menjelaskan dalam dokumen resminya bahwa *web service* adalah sebuah sistem perangkat lunak yang didesain untuk mendukung interoperabilitas antar mesin dan dapat berinteraksi melalui sebuah jaringan. Pengertian lain *web service* adalah layanan yang disediakan melalui media internet dengan menggunakan sistem pengiriman pesan XML terstandarisasi [9].

Sesuai dengan definisinya, *web service* merupakan kumpulan fungsi yang dapat diakses oleh sistem lain. Dengan demikian *web service* merupakan suatu sistem yang tidak memiliki

user interface. Sebagai konsekuensinya, untuk mengimplementasikan suatu *web service* diperlukan aplikasi lain yang dapat mengakses fungsi-fungsi yang terdapat dalam sebuah *web service*. Dalam hal ini aplikasi tersebut dapat disebut sebagai *agent*. Agent inilah yang melakukan pengiriman dan penerimaan massage dari suatu *web service* ketika melakukan akses terhadap *web service* [10].

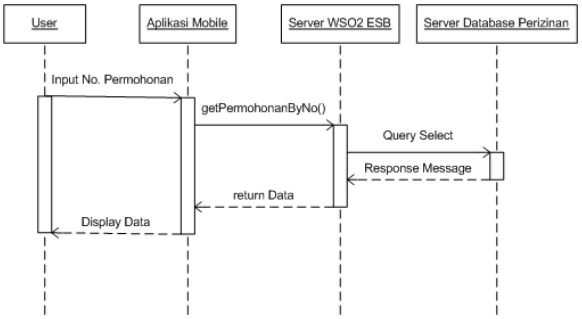
2.5. Rancangan Integrasi Aplikasi Cek Status Izin dengan Data Perizinan

Untuk dapat melakukan pemeriksaan status setiap permohonan yang diproses oleh Badan Penanaman Modal dan Perizinan Terpadu (BMPT), diperlukan sebuah layanan / *service* yang dapat mengakses data dari *database* perizinan. *Service* tersebut nantinya dapat diakses secara *online* dan *realtime* oleh aplikasi berbasis *mobile*. Berikut *Use Case* diagram proses integrasi aplikasi cek status izin dengan data perizinan seperti yang ditunjukkan pada Gambar 3.



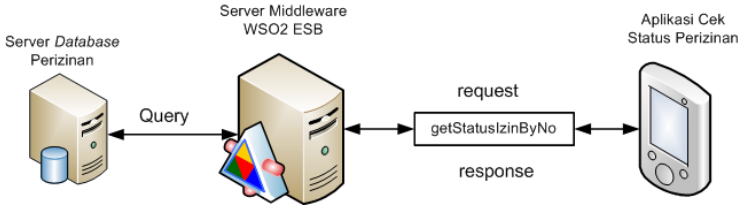
Gambar 3 *Use Case Diagram* Diagram Proses Integrasi Aplikasi Cek Status Izin dengan Data Perizinan

Layanan *service* tersebut dapat dipanggil dengan mengirimkan parameter berupa nomor permohonan sebagai kode unik setiap permohonan izin. Nomor permohonan merupakan nomor yang diperoleh ketika pemohon sudah mengajukan dan mendaftar pada satu perizinan tertentu. Berikut *Sequence Diagram* proses integrasi aplikasi cek status izin dengan data perizinan seperti yang ditunjukkan pada Gambar 4.



Gambar 4 *Sequence Diagram* Proses Integrasi Aplikasi Cek Status Izin dengan Data Perizinan

Adapun arsitektur sistem yang digunakan untuk merealisasikan model integrasi ini dapat dilihat pada Gambar 5.



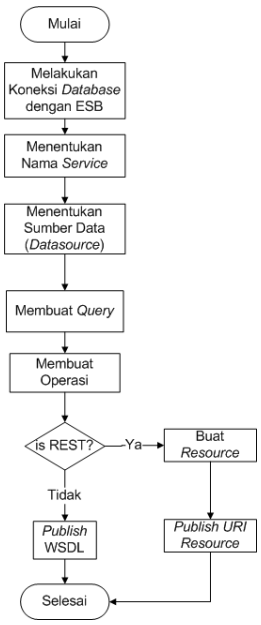
Gambar 5 Rancangan Arsitektur Integrasi Data Perizinan dan Aplikasi Cek Status Perizinan

Gambar diatas menunjukkan bahwa aplikasi cek status perizinan dapat mengakses data Perizinan melalui ESB dengan mengakses *service* yang dibuat oleh ESB. ESB membuat koneksi ke *server database* perizinan untuk dapat melakukan *query* yang dibutuhkan untuk mendapatkan data status perizinan, *query* tersebut kemudian di-*generate* menjadi sebuah *service* dengan sebuah *operation*.

Misalnya untuk menampilkan data status permohonan izin berdasarkan nomor permohonan, maka dibuatkan satu *operation* dengan nama misalnya "getPermohonanByNo". Selanjutnya aplikasi cek status perizinan dapat melakukan *request* pada *service* dan operasi tersebut kemudian ESB memberikan respon sesuai dengan operasi yang dipanggil oleh aplikasi aplikasi cek status perizinan.

2.6. Diagram Alur Pengembangan Service menggunakan SOAP / REST

Untuk mengembangkan sebuah *service* di dalam ESB ada beberapa tahapan yang perlu dilakukan antara lain melakukan koneksi *database* dengan ESB, menentukan nama *service*, menentukan sumber data (*datasource*), membuat *query* dan membuat operasi. Kemudian jika *service* tersebut mau dibuat secara REST maka terlebih dahulu harus membuat *resource* dengan *method* tertentu misal (GET/POST/PUT/DELETE). Jika tidak maka *service* tersebut sudah bisa dipublish menggunakan WSDL yang sudah di-generate oleh ESB. Berikut diagram alur pengembangan *service* baik SOAP maupun REST seperti yang ditunjukkan pada Gambar 6.

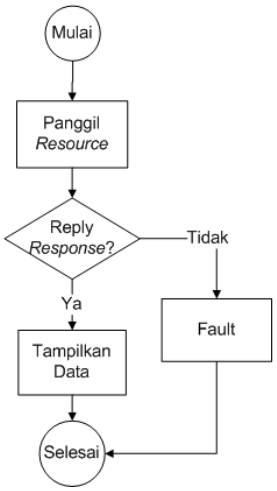


Gambar 6 Diagram Alur Pengembangan *Service* Baik SOAP Maupun REST

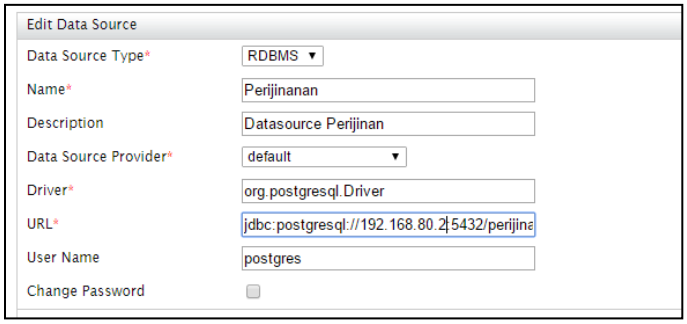
2.7. Diagram Alur Pemanggilan / Penggunaan Service berbasis REST

Selanjutnya untuk alur penggunaan *service* berbasis REST seperti yang ditunjukkan pada Gambar 7. Berbeda dengan service berbasis SOAP, *service* berbasis REST tidak membutuhkan *client* sendiri untuk melakukan pemanggilan. Namun cukup melakukan pemanggilan secara langsung melalui HTTP dengan method tertentu missal GET atau POST. Kemudian jika *service* mengembalikan respon pesan makan respon tersebut dapat ditampilkan sebagai data. Namun jika tidak maka dianggap sebagai kesalahan (*fault*).

Implementasi ini melibatkan *database* perizinan sebagai data sumber yang dikoneksikan ke ESB. ESB harus terlebih dahulu dikoneksikan dengan *database* perizinan. Setting koneksi *database* didalam ESB seperti yang ditunjukkan pada Gambar 8.



Gambar 7 Diagram Alur Pemanggilan Service REST



Gambar 8 Konfigurasi Koneksi d*atabase* Perizinan Di ESB

Dalam konfigurasi tersebut harus didefinisikan nama koneksi, *driver*, url dan *username* serta *password* yang digunakan untuk terhubung ke *database* perizinan tersebut.

2.8. Pembuatan Service Data Status Perizinan

Sama seperti yang dijelaskan dalam analisis kebutuhan dan rancangan sistem terdapat sebuah operasi yang diperlukan untuk mendapatkan data status perizinan yaitu operasi untuk mendapatkan data perizinan berdasarkan nomor permohonan. Berikut nama *service*, operasi dan Jenis *Query* yang digunakan seperti yang ditunjukkan pada Gambar 9.

```
SELECT
  a."NOMOR_URUT",a."TGL_PERMOHONAN", a."ALAMAT_USAHA",
  b."NAMA_PEMOHON",f."NAMA_JENIS_IZIN", c."NAMA_KELURAHAN",d."NAMA_KECAMATAN",
  h."REFSTATUSPROSESNAMA"
FROM "PERMOHONAN_IZIN" a
LEFT JOIN "PEMOHON" b ON a."ID_PEMOHON" = b."ID_PEMOHON"
LEFT JOIN "REF_KELURAHAN" c ON c."ID_KELURAHAN" = a."ID_KELURAHAN"
  AND c."ID_KECAMATAN" = a."ID_KECAMATAN"
LEFT JOIN "REF_KECAMATAN" d ON d."ID_KECAMATAN" = a."ID_KECAMATAN"
LEFT JOIN "REF_JENIS_IZIN" f ON f."ID_JENIS_IZIN" = a."ID_JENIS_IZIN"
LEFT JOIN "REF_STATUS_PROSES" h ON h."REFSTATUSPROSESID" = a."ID_STATUS_PROSES"
WHERE a."NOMOR_URUT" = ?
```

Gambar 9 *Service*, Operasi dan Jenis *Query* yang digunakan pada *service* Data Status Perizinan

Berikut pengkodean *service* untuk mendapatkan data status perizinan seperti yang ditunjukkan pada Gambar 10.

```
<data name="mobileapp" serviceNamespace="http://perijinan.slemankab.go.id">
   <description>Service Perijinan untuk keperluan cek status izin via mobile app
</description>
   <config id="default">
      <property name="carbon_datasource_name">Perijinanan</property>
   </config>
   <query id="select_permohonan_by_no_permohonan" useConfig="default">
      <sql>select ..sql>
      <result element="Entries" rowName="Entry">
                …
      </result>
      <param name="no_permohonan" sqlType="STRING"/>
   </query>
   <operation name="get_permohonan_by_no">
      <call-query href="select_permohonan_by_no_permohonan">
         <with-param name="no_permohonan" query-param="no_permohonan"/>
      </call-query>
   </operation>
   <resource method="GET" path="get_permohonan_by_no">
      <call-query href="select_permohonan_by_no_permohonan">
         <with-param name="no_permohonan" query-param="no_permohonan"/>
      </call-query>
   </resource>
</data>
```

Gambar 10 Pengkodean *Service* Perizinan didalam ESB

Pada gambar diatas didefinisikan *query* yang digunakan untuk menampilkan data status perizinan berdasarkan nomor permohonan. Kemudian didefinisikan pula *query* untuk mendapatkan data status nomor permohonan. Selanjutnya didefinisikan Id Query dan Query SQL lalu menentukan kolom-kolom mana saja yang dikeluarkan sebagai respon. Pada pengkodean tersebut juga didefinisikan parameter "no_permohonan" sebagai parameter masukan pada saat *query* dilakukan. operasi "get_permohonan_by_no" menunjuk pada Id Query "select_permohonan_by_no_permohonan".

Salah satu kelebihan dari ESB adalah kemampuannya untuk melakukan transformasi protokol dan format pesan misalnya melakukan transformasi dari protokol SOAP menjadi protokol http yang digunakan oleh REST. Dengan menggunakan protokol REST nantinya aplikasi *client* tidak perlu melakukan parsing seperti pada protokol SOAP. Dalam REST *output* pesan langsung dapat digunakan karena sudah merupakan data utuh. Dengan menabahkan sintaks "*resource*" dengan method "GET" pada ESB maka *service* tersebut sudah dapat diakses secara REST menggunakan URL yang digenerate oleh ESB.

Data dalam format XML dapat juga ditranformasikan ke dalam format lain misalnya JSON dengan menggunakan *proxy service*. Berikut sintaks yang digunakan untuk mengubah format XML kedalam format JSON didalam ESB seperti yang ditunjukkan pada Gambar 11.

```
<inSequence>
         <property name="uri.var.no" expression="$url:no"/>
         <filter xpath="$url:no">
            <then>
               <send>
                  <endpoint key="cekStatusIzinEp"/>
               </send>
            </then>
         </filter>
 </inSequence>
 <outSequence>
         <property name="messageType" value="application/json" scope="axis2"/>
         <send/>
 </outSequence>
```

Gambar 11 Proses Transformasi Pesan ke Format JSON

Setelah dilakukan transformasi pesan, maka respon pesan akan berubah menjadi format JSON ketika *service* tersebut dipanggil seperti yang ditunjukkan pada Gambar 12.


```
{"Entries":{"Entry":{"NOMOR_URUT":"000007.02.14","TGL_PERMOHONAN":"02-01-
2014","ALAMAT_USAHA":{"@nil":"true"},"NAMA_PEMOHON":"Yohana
Setyani","NAMA_JENIS_IZIN":"Izin Perubahan Penggunaan Tanah","NAMA_KELURAHAN":"-
","NAMA_KECAMATAN":{"@nil":"true"},"REFSTATUSPROSESNAMA":"Proses penjadwalan
tinjau lokasi dan sidang"}}}
```


Gambar 12 Respon Pesan dalam Format JSON

2.9. Pemanggilan Service Perizinan pada Aplikasi Cek Status Permohonan Izin Berbasis Mobile

```
protected Void doInBackground(Void... arg0) {
       // Creating service handler class instance
       ServiceHandler sh = new ServiceHandler();
       URL ="
ttp://192.168.80.2:8281/services/G2B/Simperijinan/to/mobileapp/get_permohonan_by_no?no_permohona
n="
       Log.d("new URL", URL);
       // Making a request to url and getting response
       String jsonStr = sh.makeServiceCall(URL, ServiceHandler.GET);
       Log.d("Response: ", "> " + jsonStr);
           if (jsonStr != null) {
              try {
                 JSONObject jsonObj = new JSONObject(jsonStr);
                    for (int i = 0; i < jsonObj.length(); i++) {
                 ..
                  Log.d("Kontak: ", "> " + status);
                 // tmp hashmap for single contact
         HashMap<String, String> contact = new HashMap<String, String>();
         // adding each child node to HashMap key => value
                          …
                 Log.d("tgl",tgl);
                 // adding contact to contact list
                 contactList.add(contact);
                   }
            } catch (JSONException e) {
                e.printStackTrace();
           }
          } else {
           Log.e("ServiceHandler", "Couldn't get any data from the url");
          }
          return null;}
```

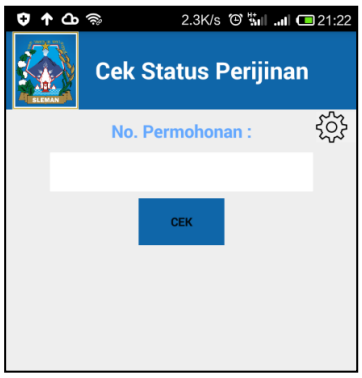
Gambar 13 Pengkodean Proses Pemanggilan *Service* Perizinan Menggunakan Java

Aplikasi cek status permohonan izin dikembangkan menggunakan platform Android dengan bahasa pemrograman berbasis java. Gambar 13 menunjukkan pengkodean proses pemanggilan *service* menggunakan bahasa pemrograman Java.

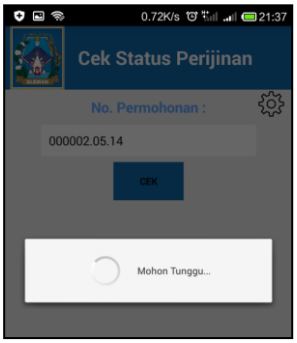
3. HASIL DAN PEMBAHASAN

3.1. Hasil Pengujian Fungsional

Aplikasi cek status permohonan perizinan dikembangkan berbasis *mobile* dengan *Android* menggunakan bahasa *Java*. Aplikasi ini digunakan untuk mengetahui status dari proses perizinan yang diajukan oleh pemohon berdasarkan nomor permohonan. Tampilan antar muka aplikasi cek status permohonan izin seperti yang ditunjukkan pada Gambar 14. Dalam aplikasi tersebut pemohon harus memasukkan nomor permohonan terlebih dahulu pada *text-box* yang disediakan. Ketika pemohon menekan tombol "Cek" maka aplikasi tersebut akan melakukan pemanggilan pada *service* yang telah dibuat berbasis REST API. Berikut tampilan aplikasi pada saat melakukan pemanggilan *service* seperti yang ditunjukkan pada Gambar 15.

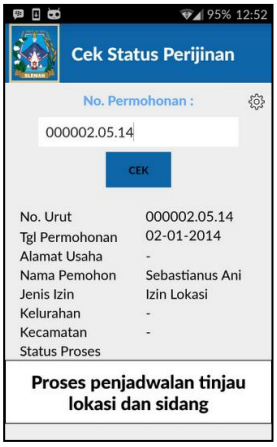


Gambar 14 Antarmuka Aplikasi Cek Status Perizinan Berbasis *Mobile*



Gambar 15 Tampilan Aplikasi pada saat Melakukan Pemanggilan *Service*

Ketika proses pemanggilan *service* tersebut sukses maka *service* akan mengembalikan respon berupa data status permohonan izin berdasarkan nomor permohonan yang diminta. Respon data tersebut dikembalikan dalam format JSON. Data tersebut kemudian ditampilkan pada aplikasi cek status permohonan seperti yang ditunjukkan pada Gambar 16.



Gambar 16 Tampilan Aplikasi Cek Status Perizinan setelah Mendapatkan Respon Data Status Permohonan Izin.

Berdasarkan hasil pengujian fungsional proses integrasi data perizinan dengan aplikasi pemeriksaan status perizinan menunjukkan bahwa Aplikasi Cek Status Permohonan dapat memperoleh data status permohonan secara *real time* menggunakan *service* berbasis REST yang dihasilkan oleh ESB. *Service* tersebut memberikan respon data sesuai dengan nomor permohonan yang diminta oleh aplikasi *client*.

ESB dapat melakukan proses transformasi protokol dan format pesan dari sebuah *service*. Pada bab sebelumnya dijelaskan bahwa respon dari *service* yang dikeluarkan ESB adalah respon dalam protokol SOAP. Kemudian ESB dapat melakukan transformasi protokol pesan menjadi REST. Selanjutnya format pesan juga dapat ditransformasikan dari format XML menjadi format JSON.

Oleh karena itu dengan *service* ini data status perizinan berhasil diperoleh dalam format JSON yang kemudian ditampilkan dengan baik kedalam aplikasi cek status permohonan izin. Sehingga dengan demikian pemohon tidak perlu datang langsung ke Kantor Badan Penanaman Modal dan Perizinan Terpadu untuk mendapatkan informasi status permohonannya. Pemohon cukup menjalankan aplikasi cek status permohonan izin melalui *smartphone* yang dimilikinya kapan saja dan dimana saja.

3.2. Hasil Pengujian Unjuk Kerja

Pengujian unjuk kerja dilakukan untuk mengetahui performa dari *service-service* yang sudah dikembangkan. Pada pengujian unjuk kerja ini penulis menggunakan perangkat lunak SoapUI, sebuah perangkat lunak *open source* yang digunakan khusus untuk melakukan

pengujian terhadap *service*. SoapUI menyediakan fasilitas untuk melakukan pengujian terhadap target service. Pengujian dilakukan pada sebuah komputer dengan spesifikasi CPU dengan kecepatan 1.80 GHz, RAM sebesar 4.0 GB, sistem operasi *Microsoft Windows* 8 dan dijalankan pada jaringan lokal.

Pengujian dilakukan untuk mengetahui rata-rata waktu eksekusi transaksi atau *Average Execution Time* (AET), rata-rata jumlah transaksi perdetik atau *Average Transaction per Second* (ATPS), rata-rata jumlah *byte* perdetik atau *Average Byte per Second* (ABPS) dan rata-rata jumlah kesalahan yang terjadi tiap detik (*Error*). Berikut hasil pengujian operasi dari *service* yang sudah dikembangkan. Pengujian dilakukan selama 60 detik dan diberikan *virtual user (Thread)* sebanyak 10 kemudian diulang hingga 10 kali percobaan. Selanjutnya dicari rata-rata dari 10 kali percobaan tersebut sedemikian hingga didapatkan hasil pengujian seperti yang ditunjukkan pada Tabel 1

Tabel 1 Hasil Pengujian Unjuk Kerja *Service*

Test No.	ET (detik)	TPS	BYTES	BPS	Error(s)
1	0,076	13,02	697433	11623,88	0
2	0,082	12,80	685824	11430,40	0
3	0,078	12,73	682252	11370,87	0
4	0,078	12,97	694754	11579,23	0
5	0,076	13,13	703684	11728,07	0
6	0,077	12,85	688503	11475,05	0
7	0,077	12,85	688503	11475,05	0
8	0,078	12,78	684931	11415,52	0
9	0,077	12,97	694754	11579,23	0
10	0,077	12,83	687610	11460,17	0
Rata-rata	**0.077**	**12,89**	**690824,8**	**11513,75**	**0**

4. KESIMPULAN

Berdasarkan hasil pengujian fungsional proses integrasi data perizinan dengan aplikasi pemeriksaan status perizinan menunjukkan bahwa Aplikasi Cek Status Permohonan dapat memperoleh data status permohonan secara *real time* menggunakan *service* berbasis REST yang dihasilkan oleh ESB. *Service* tersebut memberikan respon data sesuai dengan nomor permohonan yang diminta oleh aplikasi *client*. Selain itu Hasil pengujian unjuk kerja menunjukkan bahwa rata-rata waktu eksekusi untuk semua operasi service sebesar 0,077 detik. Selanjutnya setiap detik rata-rata melakukan eksekusi sebanyak 12,89 kali dengan data sebesar 11513,75 byte atau setara dengan 11,24 KB. Hasil tersebut menunjukkan bahwa waktu eksekusi dianggap kecil sehingga *service* yang dikembangkan dianggap tidak mengganggu proses transaksi yang sedang berjalan

5. SARAN

Saran yang diberikan untuk perbaikan penelitian ini kedepan antara lain perlu dilakukan penelitian lebih lanjut mengenai keterlibatan *service* yang bukan berasal dari ESB namun dimasukkan ke dalam ESB. Kemudian perlu juga dilakukan penelitian untuk melakukan orkestrasi *service* berbasis sekuensial proses yang dapat menyelesaikan permasalahan proses bisnis yang berbeda namun saling terkait.

UCAPAN TERIMA KASIH

Direktorat Pendidikan Tinggi (Dikti) yang telah memberikan beasiswa BPPDN sehingga penulis dapat menyelesaikan penelitian ini. Selain itu penulis mengucapkan terimakasih kepada Dinas Perhubungan Komunikasi dan Informatika Kabupaten Sleman atas izin yang diberikan untuk melakukan penelitian pada sistem informasi perizinan BPMPT Kabupaten Sleman.

DAFTAR PUSTAKA

[1] Utomo, W.H. 2012 *Penerapan Enterprise Service Bus (ESB) Sebagai Middleware Integrasi Berbasis SOA,*

[2] Hidayat, A. N., 2012, *Integrasi Aplikasi Android dan Komputer Server sebagai Solusi Mobile Commerce dan CRM Studi Kasus Toko Game XYZ*. Institut Teknologi Sepuluh Nopember, Surabaya.

[3] Prasojo W. B., 2014, *Integrasi Informasi Menggunakan Pendekatan Service Oriented Architecture untuk Mendukung Layanan Publik Pemerintah kabupaten Sleman*. Universitas Gadjah Mada, Yogyakarta.

[4] Erl, T., 2005, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall PTR, Upper Saddle River, New Jersey 07458

[5] Chappell, D. 2004, *Enterprise Service Bus*. Gravenstein Highway North, O'Reilly Media, Inc.

[6] Andary, J.F. and Sage, A.P., 2010, *The role of service oriented architectures in systems engineering*, Information Knowledge Systems Management 9, IOS Press.

[7] Juric, M.B., Loganathan, R., Sarang, P., dan Jennings, F., 2007, *SOA Approach to Integration*, Packt Publishing, Birmingham, B27 6PA, UK.

[8] Keen, M., 2004, *Patterns: Implementing an SOA Using an Enterprise Service Bus.* USA, IBM RedBooks.

[9] Cerami, E., 2002, *Web Services Essentials Distributed Applications with XMLRPC, SOAP, UDDI & WSDL* O'Reilly: North Sebastopol.

[10] Tabrani, T., 2008, *Web Service Sebagai Media Komunikasi Pertukaran Data Elektronik*, FMIPA UGM, Yogyakarta.