Aplicación de Servicios Web SOAP/REST para funcionalidades existentes en sistemas informáticos provinciales

CASTRO Marcelo, SÁNCHEZ RIVERO David, FARFÁN José,
CASTRO Daniel, CÁNDIDO Andrea, VARGAS Alejandro
Investigación + Desarrollo en Gobierno Electrónico / Facultad de Ingeniería /Universidad Nacional de Jujuy
Av. Italia y Av. Martiarena / S. S. de Jujuy / Provincia de Jujuy
mcastro@fi.unju.edu.ar, vdsanchezrivero@fi.unju.edu.ar, jhfarfan@fi.unju.edu.ar
ddcastro@fi.unju.edu.ar, agcandido@fhycs.unju.edu.ar, lavargas@fi.unju.edu.ar

Resumen: En el presente trabajo se desarrollará una propuesta de aplicación de Servicios Web SOAP/REST para utilizar funcionalidades existentes en cualquier sistema informático que se encuentre instalado en el ámbito gubernamental. Para ello se utilizará a modo de caso de estudio, una arquitectura cliente servidor típica con la que cuenta cualquier área de sistemas de la administración pública que tenga la responsabilidad de ejecutar aplicaciones transversales críticas como lo son los Sistemas de Administración Financiera, Liquidación de Haberes, Expedientes y verticales como Impuestos, Catastro, Obras Públicas, Historia Clínica entre otras.

En primer lugar se desarrollará una breve introducción a los conceptos de Arquitectura Orientada a Servicios (SOA) y particularmente a Servicios Web. Posteriormente se expondrá el diseño arquitectónico con que cuentan los organismos informáticos gubernamentales encargados de procesar las aplicaciones mencionadas anteriormente y con posterioridad se desarrollará una propuesta de utilización de Servicios Web SOAP/REST, particularizada para el proceso centralizado de liquidación de haberes, por ejemplo a nivel provincial.

Es importante señalar que la arquitectura de software propuesta puede ser extendida a cualquier solución tecnológica que se encuentre en producción en los sistemas informáticos de los distintos organismos públicos, independientemente del nivel jurisdiccional al que pertenezca; municipal, provincial o nacional.

Palabras clave: Gobierno Electrónico (egov). Tecnologías de Información y Comunicación (TIC). Arquitectura Orientada a Servicios (SOA). Servicios Web. SOAP. REST.

Introducción

Actualmente el concepto de sistemas distribuidos alcanzó gran notoriedad impulsado por el avance de la Web e Internet; lo que implica, en consecuencia, el decrecimiento del uso de sistemas centralizados multiusuario. El auge de los sistemas distribuidos tiene su origen en la aparición de redes de computadoras de alta velocidad, en las enormes prestaciones que brindan las computadoras modernas y el desarrollo de una gran variedad de

software específico, para aplicaciones distribuidas, el que permite coordinar actividades y compartir recursos de los sistemas involucrados.

Coulouris definió a los sistemas distribuidos como “sistemas en los que los componentes hardware y/o software existentes en una red de computadoras, se comunican y coordinan sus acciones mediante el intercambio de mensajes” [Coulouris et al., 2001].

La extraordinaria evolución de la Web y la gran variedad de servicios insatisfechos, como por ejemplo, los servicios que una organización gubernamental (de cualquier orden) debiera brindar al ciudadano común –tema que los responsables de las administraciones públicas tratan de abordar a través de diferentes procesos de solución, siendo Gobierno Electrónico (GE) uno de ellos– hizo que se estudien e investiguen diferentes arquitecturas como alternativas para dar respuesta al problema.

En la actualidad se está empleando SOA (Arquitecturas Orientadas a Servicios), concepto que se definirá más adelante, como medio para implementar sistemas distribuidos.

En este sentido las primeras arquitecturas SOA se basaban en CORBA (Common Object Request Broker Architecture). Pero también merecen mencionarse DCOM (Distributed Component Object Model) y RPC (Remote Procedure Protocol).

Sin embargo, la implementación eficiente de estos sistemas en la Web, acarrea ciertos desafíos o necesidades tales como el rendimiento, la experiencia de los usuarios, la reusabilidad y la compatibilidad de los servicios. Estas necesidades dieron origen a los Servicios Web, que en definitiva brindan mecanismos estándar de interconexión entre los usuarios y la información residente en servidores.

En resumen, estos servicios extienden las características de SOA, sobre todo en lo que respecta a interoperabilidad y reusabilidad [Los Santos, 2009].

Arquitectura Orientada a Servicios

La Arquitectura orientada a Servicios (SOA) no es un concepto fácil de definir. Diversos autores afirman que es un enfoque para diseñar y construir soluciones de negocios; tal que permitan, a las organizaciones, unir sus objetivos con la infraestructura de las Tecnologías de la Información y la Comunicación (TIC).

Microsoft Corporation define SOA como: “una estrategia general de organización de los elementos de IT, de forma que una colección abigarrada de sistemas distribuidos y aplicaciones complejas se pueda transformar en una red de recursos integrados, simplificada y sumamente flexible”. [Microsoft, 2006].

Según Diego Marsili: “SOA no se trata de software o de un lenguaje de programación, SOA es un marco de trabajo conceptual que permite a las organizaciones unir los objetivos de negocio con la infraestructura de TIC integrando los datos y la lógica de negocio de sus sistemas separados” [Marsili, 2007].

Según un trabajo de investigación anterior se debe distinguir falsos supuestos y verdades sobre SOA, donde, básicamente, se insiste que no es una tecnología ni una metodología, además no asegura una adecuada articulación entre TIC y las necesidades del negocio, no requiere una tecnología completa y la revisión de los procesos del negocio, debe ser gradual y construida sobre las aplicaciones actuales mediante procesos iterativos e incrementales; y es un medio para la entrega de soluciones, no un objetivo. Se aclara también que la manera más habitual de implementarla es a través de servicios web, que no significa que estos servicios requieran una arquitectura SOA, y viceversa [Castro et al., 2012].

Las aplicaciones desarrolladas, para las organizaciones, siempre han sido una tarea difícil, ya sea por los avances tecnológicos, por requerimientos de usuario poco exactos y con frecuentes modificaciones o por la alta competitividad existente en el mercado actual.

Mauricio Naranjo, en Oracle Architect Forum, detalla las necesidades comunes de las empresas, entre las cuales se pueden mencionar: el intercambio de información a través de distintos medios, gestión del riesgo a través de información oportuna y consistente, controlar sus procesos de negocios, reducir costos de las operaciones, aumentar los canales de interacción con el cliente y mejorar su experiencia y satisfacción, desarrollar los servicios a través de la utilización efectiva de las nuevas tecnologías, entre otros [Naranjo, 2007].

El manejo ineficiente de la información es el origen principal de muchas deficiencias, debido especialmente a la imposibilidad de poder presentar la información de manera sencilla y coherente, esto repercute principalmente en un aspecto fundamental, para GE por ejemplo, como lo es la interoperabilidad.

Si nos referimos a organizaciones gubernamentales, este grupo de investigación mencionaba en el año 2009, la importancia y los beneficios de la interoperabilidad entre las distintas aplicaciones: integrar la información, obtener información en tiempo real, consolidar las operaciones transaccionales, permitir que la información sea compartida por diversas reparticiones, obtener una única Base de Datos, eliminar las diversas interfaces que se encuentran en las distintas unidades de organización, estandarizar y homogeneizar las distintas herramientas de acceso a la información, facilitar los procesos de reingeniería y documentación, mejorar la gestión de los flujos de trabajo (workflow) y finalmente aumentar la productividad disminuyendo la duplicación de los esfuerzos de los distintos actores [Castro et al., 2009].

La arquitectura SOA permite integrar sistemas y aplicaciones heterogéneas e independientes a través de una metodología que logre una óptima integración como así también facilidad de adaptación ante cambios posteriores surgidos de la evolución de cualquier organización [Microsoft, 2006].

Las principales ventajas de SOA pueden resumirse en: interoperabilidad, flexibilidad, reusabilidad y rentabilidad.

- Interoperabilidad: esta característica, de SOA, permite mejorar la toma de decisiones; ya que al tener las aplicaciones integradas e intercambiando datos se dispone de información más exacta y actualizada.
- Flexibilidad: esta arquitectura logra optimizar y aprovechar los sistemas existentes, y adaptarse de manera rápida a nuevas aplicaciones que surgen por nuevas demandas.
- Reusabilidad: facilita la creación de un repositorio de servicios reutilizables que se pueden combinar en servicios de mayor nivel y aplicaciones compuestas en respuesta a nuevas necesidades de la empresa. Con ello se reduce el costo del desarrollo de soluciones y de los ciclos de prueba, se eliminan redundancias y se consigue su puesta en valor en menos tiempo [Microsoft, 2006].
- Rentabilidad: característica muy relacionada con la reusabilidad y con el uso eficiente de los recursos. Un ejemplo de ella podría ser el aumento de la productividad de los empleados al tener un mejor acceso a los sistemas y a la información.

Servicios Web

Un servicio Web es una tecnología que utiliza diversos estándares y protocolos para intercambiar datos entre aplicaciones [Wikipedia1, 2013]. Estas aplicaciones de software se pueden programar en distintos lenguajes de programación y acceder a distintas bases de datos a la vez, ya sea desde una red local o desde Internet por medio de computadoras o cualquier otro dispositivo que sea compatible con dicha aplicación.

Actualmente, organizaciones como OASIS y W3C, son las entidades encargadas de poner a disposición estándares para definir la arquitectura y reglamentación de los Servicios Web para fomentar la interoperabilidad de los mismos. Así se promueve la interoperabilidad, el uso de estándares y la interconexión de información entre compañías ubicadas, geográficamente, en lugares diferentes.

Como contrapartida, se puede decir que el rendimiento de una aplicación Web distribuida es más lenta que una aplicación cliente-servidor y que al utilizar el protocolo HTTP, es necesario fortalecer las medidas de seguridad mediante el uso de firewall y sistemas de auditorías [Hansen, 2007].

Los Servicios Web han ido evolucionando y se pueden distinguir distintas generaciones:

Primera Generación: En esta etapa se trató de crear estándares y protocolos capaces de compatibilizar el uso de aplicaciones y servicios Web para optimizar el uso y rendimiento de los mismos.

- WSDL (Web Services Description Languaje): se utiliza para describir la interfaz pública de un servicio Web a través de XML; esto describe, por ejemplo, la forma de la comunicación (requisitos del protocolo y formato de los mensajes), necesaria para interactuar con los servicios de un catálogo determinado.
- SOAP (Simple Object Access Protocol): es un protocolo creado por Microsoft, IBM y otros, para definir como dos objetos, en diferentes procesos, pueden comunicarse intercambiando datos en lenguaje XML.
- UDDI (Universal Description Discovery and Integration): es un catálogo de negocios basado en XML para ser accedido por mensajes SOAP y dar paso a documentos WSDL para la descripción de requisitos y formatos de mensajes que permiten interactuar con servicios Web.
- WS-I Basic Profile: especificación que contiene un conjunto de especificaciones para servicios Web, no propietarios, que permiten llevar a cabo la interoperabilidad entre SOAP y WSDL.

Segunda Generación: Esta generación está orientada a reforzar las medidas de seguridad, en los Servicios Web, mediante el uso de librerías de cifrado y encriptado, lo que permite tener capas de transporte de seguridad sobre protocolo HTTP.

- WS-Security: es el protocolo encargado de brindar un medio para aplicar seguridad a los Servicios Web. A través de STL (Security Transport Layer) se apunta a garantizar la confidencialidad y seguridad sobre los Servicios Web.
- Ws-addressing: estandariza el enrutamiento de mensajes de datos a través de las cabeceras de SOAP. Ws-addressing es el encargado de asegurar el despacho y entrega de datos a través de la capa de transporte.
- SOAP (Simple Object Access Protocol): este protocolo permite a un WS (Web Service) comunicar dos objetos, de diferentes procesos, mediante XML; es muy recomendable para entornos formales y donde las funcionalidades de interfaz y datos estén claramente definidas.
- REST (REpresentacional State Transfer): arquitectura recomendada para sistemas distribuidos ideales como los entornos Web, debido a que trabaja con estándares HTTP y XML para transmitir datos sin tener que utilizar una capa de transporte extra como lo hacen los Servicios Web basados en SOAP.

Servicios Web SOAP

Describiremos a los Servicios Web SOAP como el intercambio, entre sistemas, de mensajes basados en SOAP. El Servicio Web basado en SOAP implica envío de mensajes a través de XML (Lenguaje de Marcas eXtensible), tal como lo podemos apreciar en la Figura 1. Los mensajes en las transacciones de las aplicaciones, incorpora muchos niveles de seguridad y aspectos del middleware tradicional que a nivel de aplicaciones empresarial son variables que toman gran importancia, sobre todo en la interoperabilidad entre aplicaciones [Chase, 2011].

```
<SOAPenv:Envelope
      xmlns:SOAPenv="http://schemas.xmlSOAP.org/SOAP/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <SOAPenv:Body>
         <req:getNumberOfArticles xmlns:req="http://daily-moon.com/CMS/">
            <req:category>classifieds</req:category>
         </req:getNumberOfArticles>
      </SOAPenv:Body>
   </SOAPenv:Envelope>
```

Figura 1. Listado de Servicios Web basado en SOAP.

Definiremos a XML como un metalenguaje de 2da generación, ya que es un subconjunto del Lenguaje de Anotaciones Generales (SGML), que lo describe como un lenguaje de marcas que pueden ser interpretadas por software, y ser integradas en otros lenguajes de marcas, tales como HTML (Lenguajes de Marcas para Hipertexto). Podemos caracterizarlo como un lenguaje flexible, escalonable y sobre todo adaptable. A través de XML podemos enviar y recibir información, datos estructurados en la web o entre aplicativos entre sí, y de esta manera eliminar las diferentes restricciones que teníamos en el lenguaje HTML [Tittel, 2003].

Básicamente SOAP deriva de DECOM o CORBA, utiliza HTTP como medio de transporte, dando lugar a lenguajes de alto nivel para implementar el servicio y también hace uso del protocolo XML-RPC (Remote Procedure Call), los que hacen referencia a tecnologías aplicadas en un entorno aislado, donde se tiene un conocimiento particularizado del ambiente. La interoperabilidad se efectúa con la adecuación de cada cliente o usuario, donde se necesita organizar frameworks que permitan evolucionar a la par de las modificaciones, sobre todo cuando el cliente no posee la misma API [Navarro, 2006].

SOAP, al utilizar el lenguaje de Descripción de Servicios WEB (WSDL), cumple con las siguientes especificaciones [Chase, 2011]:

- WS-Security (Seguridad para Servicios Web): hace referencia al encriptado y las firmas digitales.

- WS-Policy (Política de los Servicios Web): donde describe, en forma detallada, quiénes y cómo pueden utilizar el servicio.
- WS-I: proporciona un conjunto de estándares para la interoperabilidad entre aplicaciones.
- WS-BPEL: especifica las interacciones necesarias para crear sistemas con servicios múltiples dentro de un sistema general, a nivel empresarial.

Podemos ejemplificar, como pionera en la aplicación del Servicio Web, a la empresa Amazon, que incluye servicios basados en SOAP, pero también tiene incorporado servicios basado en REST (Transferencia de Estado Representacional).

En la Tabla 1, se pueden visualizar las características, ventajas y desventajas en la utilización de un Servicio Web basado en SOAP [Navarro, 2006].

Tabla 1 - características, ventajas y desventajas en la utilización del servicio web basado en SOAP

Características:	Las operaciones son definidas como puertos WSDL. Dirección única para todas las operaciones. Múltiple instancias del proceso comparten la misma operación. Componentes fuertemente acoplados.
Ventajas:	Fácil (generalmente) de utilizar. La depuración es posible. Las operaciones complejas pueden ser escondidas detrás de una fachada. Envolver API's existentes es sencillo. Incrementa la privacidad. Herramientas de desarrollo.
Desventajas:	Los clientes necesitan saber las operaciones y su semántica antes de su utilización. Los clientes necesitan puertos dedicados para diferentes tipos de notificaciones. Las instancias del proceso son creadas implícitamente.

Un documento XML, incluye toda la información necesaria para procesar el mensaje y llegar al origen. Un mensaje, en un Servicio Web de SOAP, está compuesto por un Header (Encabezado) y un Cuerpo (Body).

La finalidad del Header, en un mensaje SOAP, es proporcionar la información necesaria del mensaje en sí mismo, dónde también se puede incluir datos del ruteo del mensaje.

En Body especificamos la carga útil, que es el motivo de la comunicación, la acción que debe realizar el destinatario o la información que se trata de impartir al servidor.

Servicios Web REST

Los servicios Web REST (Transferencia de Estado Representacional o Representational State Transfer) “es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web” [Wikipedia, 2013]. Este concepto fue introducido, originalmente, en la tesis doctoral escrita por Roy Fielding en el año 2000, quien es uno de los principales autores de la especificación HTTP.

El objetivo que se persigue, en la construcción de aplicaciones distribuidas e inspiradas en las características de la web, es que en lugar de basar la comunicación en protocolos llamados “pesados” como SOAP, REST se basa en el envío de mensajes sobre HTTP y XML.

Un pedido HTTP consta de cuatro métodos de acceso:

- GET: con el cual se pide un recurso dado, generalmente sin efectos secundarios.
- POST: realiza un pedido, acceso o modificación de un recurso.
- PUT: agrega, en el mensaje, una representación de un recurso destinada al servidor.
- DELETE: elimina un recurso dado [Dieguez, 2013].

Estos métodos son comparados con las operaciones asociadas a SQL (Lenguaje de Consulta Estructurado) y con otras analogías como pueden verse en la Tabla 2 [Navarro, 2006]:

Tabla 2. Comparación de los métodos de HTTP

Acción	HTTP	SQL	Copy&Paste	Unix Shell
Create	PUT	Insert	Pegar	>
Read	GET	Select	Copiar	<
Update	POST	Update	Pegar después	>>
Delete	DELETE	Delete	Cortar	Del/rm

Entre las características o principios de REST se destacan:

- Escalabilidad de la interacción con los componentes: entendiendo por escalabilidad la habilidad para manejar el crecimiento continuo de trabajo de manera fluida, inclusive podemos afirmar que la Web ha crecido, exponencialmente, sin degradar su rendimiento.
- Generalidad de interfaces: esto se debe a que el protocolo HTTP puede interactuar con cualquier servidor HTTP lo cual no siempre funciona con SOAP para los Servicios Web.
- Puesta en funcionamiento independiente: lo cual es una realidad en Internet, en donde los clientes y servidores aunque se traten de servidores antiguos son puestos en funcionamiento durante años. Esta es la

ventaja de utilizar HTTP ya que permite la extensibilidad mediante el uso de las cabeceras, a través de las URL's (recursos).

- Compatibilidad con componentes intermedios: es decir con componentes, como proxys para Web, tales como caches para mejorar el rendimiento o firewall para reforzar políticas de seguridad. Otros componentes son los Gateway que permiten encapsular sistemas no propiamente Web. Esta compatibilidad permite reducir la latencia de interacción, reforzar la seguridad y encapsular otros sistemas [Navarro, 2006].

En la Figura 2 podemos observar qué lugar ocupan los principios antes mencionados de REST comparados con SOAP [Hevia, 2011]:

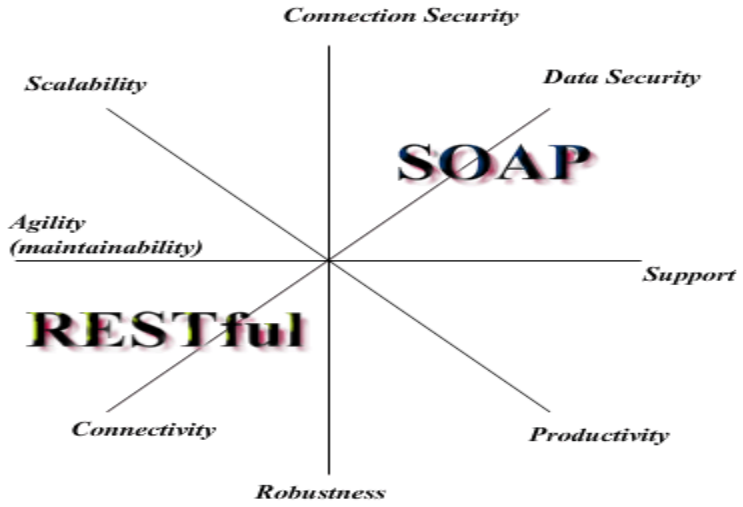

Figura 2 - Comparación de servicios REST vs. SOAP.

Aplicando Servicios Web SOAP y REST en funcionalidades existentes

A continuación se describirá la arquitectura actual con la que cuenta generalmente cualquier área de sistemas en el ámbito gubernamental que tenga la responsabilidad de ejecutar aplicaciones transversales críticas como lo son los Sistemas de Administración Financiera (SIAF), Sistema de Administración de Personal (SIAP) que incluye el proceso de Liquidación de Haberes, Expedientes y aplicaciones verticales como Impuestos, Catastro, Obras Públicas, Historia Clínica entre otras. En general en nuestro país, esta situación se da en la práctica, a nivel Provincial, dónde existe un organismo que aglutina las funciones de ejecución de las aplicaciones anteriormente mencionadas, aunque existen reparticiones que poseen sus propias áreas de sistemas o informática, pero que se encuentran conectadas al organismo informático central.

Por razones de necesidad de automatización de procesos, los primeros Centros de Cómputos o Procesamiento Centralizados Provinciales (CPCP) se encontraban y aún se encuentran, en el ámbito del Ministerio de

Hacienda o Economía. En el transcurso del tiempo el resto de los Ministerios fueron informatizando sus procesos, pero siempre estableciendo una conexión a través de fibra óptica o microondas con el CPCP.

. Ejemplo de Arquitectura Existente

En general los CPCP cuentan con una gran mayoría de aplicaciones Cliente Servidor (dos capas), para este trabajo se tomaron aplicaciones codificadas en Visual Basic 6.0, generadas con una herramienta C.A.S.E. Sin embargo la mayoría de los CPCP se decidieron iniciar un proceso de migración de algunas aplicaciones a entorno Web. La plataforma tecnológica se encuentra basada en procesadores RISC, con clustering, alta disponibilidad y soportados por el sistema operativo AIX. Con respecto al Administrador de Bases de Datos se utilizará Informix.

En relación a la conectividad se utilizará a modo de ejemplo, un paquete de internet denominado Integra que permite velocidades de hasta 100 mbps, a través de fibra óptica creándose una red privada virtual (VPN) para acceso a los usuarios de los distintos sistemas, que se encuentran en las reparticiones fuera del ámbito del CPCP.(Figura3)

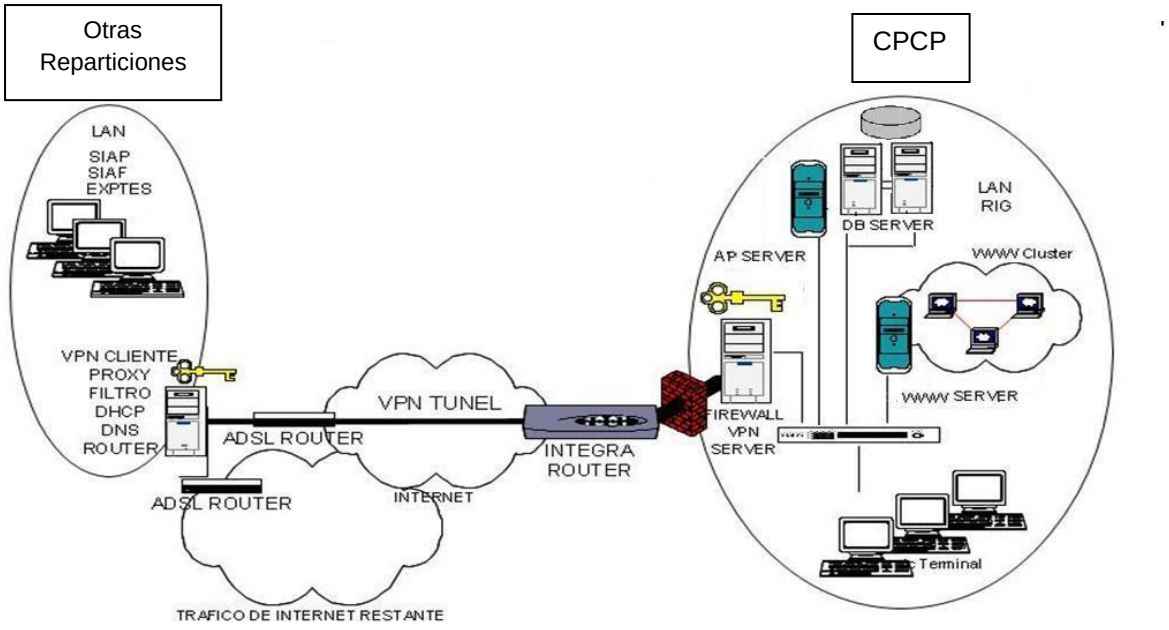

Figura 3. Arquitectura Existente

Tomando como base la arquitectura existente a continuación se realizará una propuesta sobre utilización de Servicios Web SOAP.

.Propuesta para la utilización de Servicios Web SOAP

Se propone utilizar Servicios Web SOAP para acceder a las funcionalidades que no se encuentran desarrolladas para entorno web. Como caso de estudio se utilizará del Sistema de Liquidación de Sueldos, el cual se ejecuta para todas las reparticiones pertenecientes al Poder Ejecutivo en el CPCP. Sin embargo si una determinada repartición quisiera ejecutar su propia liquidación de haberes, no lo podría hacer, en consecuencia se propone que a través de Servicios Web SOAP se permitan ejecutar las funcionalidades que se encuentran desarrolladas en Visual Basic 6.0, y de esta forma permitir a las distintas reparticiones liquidar sus sueldos.

Para lo cual se hace necesario efectuar una invocación de las siguientes funcionalidades que se encuentran alojadas en el Application Server y codificadas en Visual Basic 6.0:

- Ingresar Novedades
- Controlar Novedades ingresadas
- Procesar Liquidación
- Generar planillas de sueldos precontrol
- Generar Recibos
- Generar Listados de Retenciones, embargos y gremios
- Procesar órdenes de Pago
- Procesar Acreditación

En la Figura 4 se puede apreciar el esquema propuesto, basado en la arquitectura actual y descripta en el apartado anterior.

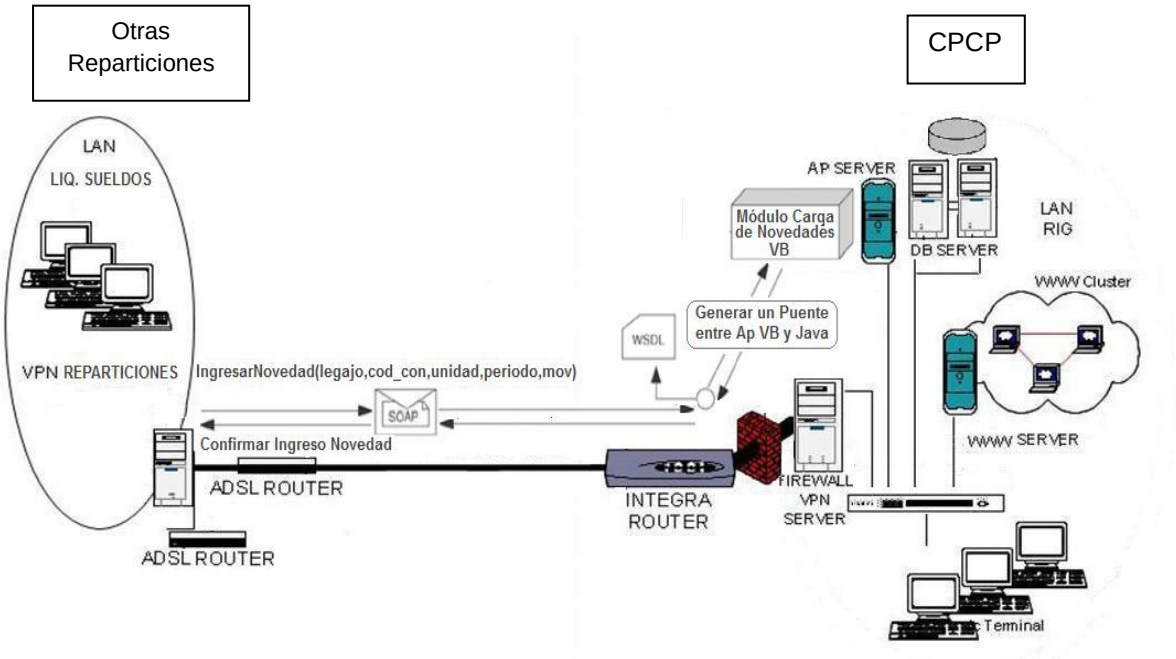

Figura 4. Utilización de funcionalidades a través de Servicios Web SOAP

Finalmente se deberá efectuar un procedimiento que permita mapear las solicitudes de Servicios Web SOAP con las funcionalidades desarrolladas en Visual Basic 6.0. Una manera de implementar dicho procedimiento podría ser a partir de una tabla que permitiera identificar la funcionalidad requerida a partir del servicio invocado por el cliente. Tabla 3. Así lo demuestra su predecesor, el protocolo XML-RPC. Estas tecnologías lograron un éxito limitado antes de ser adaptadas, se debe a que este tipo de tecnologías, las basadas en modelos RPC (Remote Procedure Call) son más adecuadas para entornos aislados. La evolución en este tipo de sistemas es sencilla, se modifica cada usuario para que cumpla con los nuevos requisitos.

Tabla 3. Relación Web Service – Application VB

Web Service (Client Side)	**Param (input)**	**Application VB (Server Side)**	**Param (output)**
ingresarNovedad()	legajo, cod_con, unidad, periodo, mov	AltaNoved()	confirma
ejecutarLiquida()	fecha,tipo	Liquida()	sinError
procesaAcred()	fecha,nroliq	Acredita()	cantEmpleados,montoTotal

.Por qué utilizar Servicios Web basados en SOAP y no implementar la solución a través de REST?

Al tratarse de aplicaciones que se encuentran en un entorno y dominio específico, en el que las operaciones no sufrirán cambios significativos, la implementación de los Servicios Web basados en SOAP puede resultar muy conveniente. Sin embargo podría cuestionarse la conveniencia de utilizar dicha solución, si la cantidad de datos a procesar en la liquidación de sueldos es significativa, lo cual induciría a creer que REST resultaría más oportuna.

Sumado a lo anterior, se encuentra el aspecto relacionado a la cantidad de usuarios que manejarían la aplicación, que en este caso de estudio son aproximadamente 3 usuarios por repartición; siendo 82 las reparticiones que utilizarían el sistema. Se estaría trabajando con aproximadamente 250 usuarios, lo cual no representa un número significativo al momento de cambiar la interfaz. Sin embargo si tenemos en cuenta que se sumarían los usuarios del Sistema Integrado de Administración Financiera (SIAF) o simplemente el sistema de Expedientes, dicho número se multiplicaría por 4, con lo cual se podría concluir que REST resultaría más conveniente.

Otro aspecto a tener en cuenta es el transporte que utilizan los Servicios Web. En la arquitectura actual con la que cuenta el CPD, se puede observar la utilización de Firewalls hay que tener en cuenta que REST utiliza exclusivamente Http como medio de transporte y SOAP añade FTP y fundamentalmente JMS, lo cual puede resultar más conveniente desde el punto de vista de la seguridad, uso de la especificación WS-Security.

También puede pensarse que SOAP puede resultar interesante al momento de simplificar el diseño de la interfaz, a partir del ocultamiento de la complejidad del sistema. No obstante la ventaja fundamental respecto a REST es que permite la utilización de lenguajes de alto nivel para llamar y para implementar el servicio, que es lo que mejor se adapta a nuestros requerimientos para la solución planteada.

Aunque SOAP puede resultar más complejo al momento de la implementación, y utiliza más recursos, posee la ventaja de ser fuertemente acoplado lo cual permite ser probado y depurado antes de poner en funcionamiento la aplicación [Erl, 2008]. Pero si se visualiza que existirá una potencial escalabilidad de las aplicaciones, que no es el caso del modelo planteado en este trabajo, REST puede resultar muchísimo más conveniente, sumado a la optimización en el uso de los recursos.

Es importante señalar que la implementación de SOAP, requerirá que los usuarios del sistema, conozcan de antemano las distintas operaciones que podrán realizar, como así también su semántica. En el caso de los sistemas implementados por el Ministerio de Hacienda, esto no representa un inconveniente.

Otro aspecto importante a tener en cuenta, para el caso de estudio del presente trabajo, es que aunque si bien SOAP no permite la asincronía pura, logra emularla a partir de la utilización del transporte asíncrono lo cual permitirá correr la liquidación (proceso largo y complejo) del lado del servidor y liberar al cliente, hasta que la misma finalice.

Otra característica que representa una ventaja de SOAP, es la fiabilidad en la comunicación, es decir el envío de una única vez del mensaje.

También se puede señalar que SOAP, posee algunas características competitivas respecto de REST, como por ejemplo la conveniencia en la utilización de WSDL, un modelo extensible que permite que los mensajes sean claramente comprensibles, logrando formalidad en el proceso de comunicación.

No obstante REST posee un mecanismo de nombrado de recursos a partir de URI, lo cual es una carencia en SOAP.

Finalmente se puede observar que para el caso de estudio expuesto en el presente trabajo, SOAP se adapta mejor ya que se trata de una tecnología óptima para la integración punto a punto de sistemas internos.

Sin embargo las ventajas de REST para sistemas abiertos puede resultar incomparable con SOAP, ya que su simplicidad en la concepción, uso y optimización de los recursos resulta realmente atractiva.

Conclusiones

Como puede observarse en el desarrollo del presente trabajo, la utilización de SOA y particularmente de los Servicios Web resulta un desafío interesante pero a la vez muy productivo para implementar soluciones de este tipo en el ámbito de la administración pública. Esto se encuentra fuertemente ligado a la gran cantidad de plataformas y arquitecturas de software que poseen los gobiernos en los tres niveles jurisdiccionales municipal, provincial y nacional. Dicha multiplicidad, muchas veces atenta contra objetivos y estándares a los que apunta la formalización de los procesos de gobierno electrónico, como es el caso de la interoperabilidad.

La utilización de arquitecturas avanzadas de software, y especialmente SOA en el ámbito gubernamental permitirá obtener una innumerable cantidad de ventajas que van desde la actualización de aplicaciones que se encontraban basadas en arquitecturas de dos capas para pasar a arquitecturas multicapas con pequeños cambios en el diseño arquitectónico a nivel de software, hasta la implementación de Servicios Web (SOAP o REST) dependiendo de los entornos de las aplicaciones que se encuentran en producción.

Finalmente podemos concluir, que aunque si bien SOA puede resultar una herramienta interesante para aumentar los servicios que brinda un organismo público a los ciudadanos, debe tener en cuenta ciertos aspectos propios del ámbito gubernamental que requiere tomar los recaudos adecuados para evitar inconvenientes que están relacionados con el tiempo de desarrollo, la curva de aprendizaje, la relación entre recursos afectados y costos demandados por la implementación de SOA.

Referencias y Bibliografía

[Castro et al., 2009] Castro, Marcelo; Farfán, José; Sánchez Rivero, Víctor, Cándido, Andrea. “Las aplicaciones transversales en Gobierno electrónico”. Actas de las V Jornadas de Investigaciones en Facultades de Ingeniería del NOA; Tomo I. Salta, 2009.

[Castro, et al., 2012] Castro, Marcelo; Sánchez Rivero David; Reinoso, Elizabeth; Aparicio, María; Aragón, Fabiana; Cazón Liliana. “Servicios digitales comunes reutilizables para gobierno electrónico”. Actas del 6º Simposio de Informática en el Estado y 41º JAIIO; La Plata, Argentina, 2012.

[Chase, 2011] Chase, Nicholas. “Comprender las especificaciones de los servicios web, Parte 1: SOAP”. Disponible en: http://www.ibm.com/developerworks/ssa/webservices/tutorials/ws-understand-web-services1/section2.html. Última visita: Abril 2013.

[Coulouris et al., 2001] Coulouris G., Dollimore, J.& Kindberg, T. “Distributed Systems Concepts and Design”. Third Edition. Ed. Addison-Wesley, 2001.

[Dieguez, 2013] Dieguez, Cristian. “Servicios Web Rest”. Disponible en: http://es.wikipedia.org/wiki/Representational_State_Transfer. Última visita: Abril 2013.

[Erl, 2006] Erl, Thomas. “Service-Oriented Architecture: Concepts, Technology & Design”. Ed. Pearson Education, 2006.

[Erl, 2008] Erl, Thomas. “SOA Principles of Service Design”. Ed. Prentice Hall, 2008.

[Hansen, 2007] Hansen, Mark. “SOA Java Using Web Services”. Ed. Prentice Hall, 2007.

[Hevia, 2011] Hevia, Andrés. “Implementación de SOA con REST”. Disponible en http://pensandoensoa.com/2011/03/19/869/. Última visita: Abril 2013.

[Kumar et al., 2010] Kumar, B.V., Narayan, Prakash & Ng, Tony. “Implementing SOA Using Java EE”. Ed. Addison Wesley, 2010.

[Los Santos, 2006] Los Santos Aransay, Alberto. “Revisión de los Servicios Web SOAP/REST: Características y Rendimiento”. Marzo 2009. Disponible en: http://www.albertolsa.com/?p=187. Última visita: Abril de 2013.

[Marsili, 2007] Marsili, Diego. “¿Qué es SOA, la arquitectura orientada a servicios?”. Disponible en: http://www.iprofesional.com/notas/46399-Que-es -SOA-laarquitectura-orientada- a-servicios.html. Última visita: Abril de 2013.

[Microsoft, 2006] Microsoft White Paper; “La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real”, 2006. Disponible en: http://social.msdn.microsoft.com/Search/es-ES?query=SOA&ac=4. Última visita Abril 2013.

.

[Naranjo, 2007] Naranjo, Mauricio. “Casos empresariales de implementación de Arquitectura orientada a servicios”. Disponible en: http://www.lucasian.com/idocs/Lucasian. OracleArchitectForum.CasosImplSOA_Ago-2007_v1.pdf. Última visita: Abril 2013.

[Navarro, 2006] Navarro Marset, Rafael. “REST vs Web Services”. Disponible en: http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf. Última visita: Abril 2013.

[Tittel, 2003] Tittel, Ed, “Teoría e Problemas de XML”. Traducción por Ralph Miller Jr. Ed. The McGraw Hill, 2003.

[Wikipedia, 2013] Wikipedia 2013. “Representational State Transfer”. Disponible en: http://es.wikipedia.org/wiki/Representational_State_Transfer. Última visita: Abril 2013.

[Wikipedia1, 2013] Wikipedia 2013. “Servicio Web”. Disponible en http://es.wikipedia.org/wiki/Servicio_web. Última visita: Abril 2013.