Monitoring and Enforcing Data Protection Laws within an E-government Interoperability Platform

Andrés Echevarría, Dahiana Morales, Laura González
Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Uruguay
{juan.echevarria, dahiana.morales, lauragon}@fing.edu.uy

***Abstract*—Public agencies are increasingly required to collaborate with each other in order to provide high-quality e-government services. This collaboration is usually based on the service-oriented approach and supported by interoperability platforms. Such platforms are specialized middleware-based infrastructures enabling the provision, discovery and invocation of interoperable software services. In turn, given that personal data handled by governments are often very sensitive, most governments have developed some sort of legislation focusing on data protection. This paper proposes solutions for monitoring and enforcing data protection laws within an E-government Interoperability Platform. In particular, the proposal addresses the requirements posed by the Uruguayan Data Protection Law and the Uruguayan E-government Platform, although it can also be applied in similar scenarios. The solutions are based on well-known integration mechanisms (e.g. Enterprise Service Bus) as well as recognized security standards (e.g. eXtensible Access Control Markup Language) and were completely prototyped leveraging the SwitchYard ESB product.**

Keywords—data protection; e-government; enterprise service bus; eXtensible Access Control Markup Language; interoperability

I. INTRODUCCIÓN

Durante las últimas décadas, muchos gobiernos han llevado adelante iniciativas de gobierno electrónico para mejorar la calidad de los servicios públicos que ofrecen a los ciudadanos [1]. Para esto, los gobiernos han puesto en marcha sistemas de gobierno electrónico que posibilitan una coordinación inter-organizacional más eficiente entre los organismos gubernamentales. Estos sistemas se apoyan, en general, en plataformas de interoperabilidad que proveen una infraestructura de hardware y software para facilitar la interconexión entre dichos organismos [2].

En Uruguay, por ejemplo, la Agencia de Gobierno Electrónico y Sociedad de la Información (AGESIC) provee una Plataforma de Interoperabilidad (PDI), integrada a su Plataforma de Gobierno Electrónico (PGE) [3], que brinda infraestructura y servicios utilitarios para reducir la complejidad de implementar servicios al público y/o accesibles dentro del Estado. La PDI es la base además para la implementación de una Arquitectura Orientada a Servicios (Service Oriented Architecture, SOA) a nivel del Estado, en la cual los servicios ofrecidos por los organismos pueden ser descriptos, publicados y descubiertos, invocados y combinados a través de interfaces y protocolos estándar.

Por otro lado, dado que los datos personales de los ciudadanos manejados por los gobiernos son en general muy sensibles, muchos países han elaborado algún tipo de ley enfocada en la protección de datos personales [4]. En particular, en Uruguay se promulgó la Ley Nº 18.331 de Protección de Datos Personales y Acción de "*Habeas Data*" [5], la cual establece que el derecho de protección de datos personales es inherente a la persona. La ley declara un conjunto de datos públicos (p. ej. nombres y apellidos, documento de identidad) mientras que al resto se los considera de carácter reservado o como "datos sensibles". En particular, para que un organismo pueda utilizar o compartir con otro organismo un dato sensible de un ciudadano, el ciudadano tiene que haber dado su consentimiento explícito.

En este contexto, resulta de interés que las plataformas de interoperabilidad para gobierno electrónico cuenten con mecanismos que permitan la gestión de estas normativas, así como el monitoreo y aseguramiento de su cumplimiento.

Este artículo propone soluciones que permiten a plataformas de interoperabilidad gestionar normativas asociadas a la protección de datos personales, así como monitorear y asegurar su cumplimiento en las interacciones entre organismos. Las soluciones se basan en mecanismos de integración ampliamente utilizados, como el Enterprise Service Bus (ESB), así como en estándares de seguridad reconocidos, como el eXtensible Access Control Markup Language (XACML). Las soluciones fueron prototipadas completamente utilizando el producto SwitchYard ESB[1].

El resto del artículo se organiza de la siguiente manera. En la Sección II se describen estándares y tecnologías relevantes para el trabajo. En la Sección III se presenta el contexto general de la problemática planteada y los requerimientos identificados. En la Sección IV se describe la solución propuesta para abordar dicha problemática. En la Sección V se detallan algunos aspectos de la implementación del prototipo desarrollado. En la Sección VI se analiza trabajo relacionado y, por último, en la Sección VII se presentan conclusiones y trabajo a futuro.

[1] http://switchyard.jboss.org/

Este trabajo fue parcialmente financiado por la Comisión Sectorial de Investigación Científica (CSIC), Universidad de la República, Uruguay.

978-1-4673-9143-6/15/$31.00 ©2015 IEEE

II. Marco Conceptual

Esta sección describe brevemente los estándares y tecnologías más relevantes para este trabajo.

A. Web Services

Un *web service* es una aplicación de *software* identificada por una URI. Sus interfaces y formas de acceso pueden ser definidas, descriptas y descubiertas como artefactos XML. Además soporta la interacción directa con otros componentes de *software* utilizando mensajes basados en XML y que son intercambiados a través de protocolos basados en internet [6].

Los *web services* se han convertido en la tecnología preferida para la implementación de una Arquitectura Orientada a Servicios (*Service Oriented Arquitecture*, SOA) y son el principal mecanismo de integración de aplicaciones construidas sobre distintas tecnologías [6].

Los principales estándares en los que se basan los *web services* son: *Simple Object Access Protocol* (SOAP) y *Web Services Description Language* (WSDL). Existen también otros estándares que extienden los anteriores para dar soporte a características avanzadas, por ejemplo, *Web Services Security* (WS-Security) y *Web Services Addressing* (WS-Addressing).

SOAP [7] es un formato basado en XML para construir mensajes, de forma independiente al transporte en que se transmiten, y un estándar que especifica cómo el mensaje se debe procesar. Los mensajes SOAP consisten en un *envelope* que contiene un encabezado (*header*) y un cuerpo (*body*). La sección de encabezado de un mensaje SOAP es extensible y puede contener varios elementos que se utilizan para especificar distintos tipos de información, por ejemplo, en relación a seguridad y direccionamiento. El protocolo más utilizado para el envío de mensajes SOAP es HTTP, pero el estándar no está restringido únicamente a éste.

WSDL [8] es un lenguaje de definición de interfaces basado en XML que permite describir de forma estándar a los *web services*. Los documentos WSDL se componen de dos grandes partes: una descripción abstracta y una descripción concreta. En la descripción abstracta, se encuentran elementos que permiten especificar las descripciones funcionales y de estructura de datos asociadas a las operaciones que provee el *web service*. En la descripción concreta se encuentran elementos que: i) proveen instrucciones para interactuar con el *web service* a través de protocolos específicos (p. ej. SOAP sobre HTTP) y ii) especifican una dirección de red concreta a través de la cual el *web service* puede ser invocado.

WS-Addressing [9] se centra en expresar características relativas al procesamiento y entrega de los mensajes. Para este fin, define elementos que permiten especificar estas características de forma independiente al protocolo de trasporte. Por ejemplo, a través del elemento "wsa:to" se puede especificar el destino del mensaje, mientras que el elemento "wsa:action" permite especificar de forma única la semántica del mismo.

WS-Security [10] especifica un conjunto de extensiones a SOAP que posibilitan la integridad, confidencialidad y autenticación a nivel de los mensajes. En particular, WS-Security describe cómo incluir *tokens* de seguridad en mensajes SOAP definiendo un conjunto de *security tokens* y describiendo el mecanismo para adjuntarlos, identificarlos y referenciarlos dentro del mensaje. El UserNameToken, por ejemplo, permite especificar un nombre de usuario y opcionalmente una contraseña. Por otro lado, los XMLTokens permiten adjuntar *security tokens* basados en XML utilizando distintos formatos como Security Assertion Markup Language (SAML).

B. Enterprise Service Bus

Un *Enterprise Service Bus* (ESB) es una plataforma de integración basada en estándares que combina, *web services*, transformación de datos, mensajería y ruteo inteligente para conectar y coordinar, de forma confiable, la interacción de aplicaciones diversas con integridad transaccional [11].

Los ESBs brindan una capa de integración intermedia que provee lógica de integración y comunicación reutilizable, para posibilitar la interacción entre clientes y servicios en una SOA. Los ESBs aceptan pedidos en forma de mensajes, sobre los cuales se pueden realizar operaciones de mediación [11] para solucionar heterogeneidades entre clientes y servicios [12].

La utilización de un ESB promueve el bajo acoplamiento entre clientes y servicios, dividiendo la lógica de comunicación e integración en pequeños componentes de software administrables. Permite además tener una clara separación entre la lógica de integración y comunicación, y la lógica de negocio implementada por los servicios [12].

En la Fig. 1 se pueden observar, cómo distintas aplicaciones y servicios que utilizan tecnologías, formatos de datos y protocolos de comunicación distintos, se comunican con el ESB a través de servicios con interfaces bien definidas. Dichos servicios pueden ser orquestados o utilizados por otras aplicaciones o servicios.

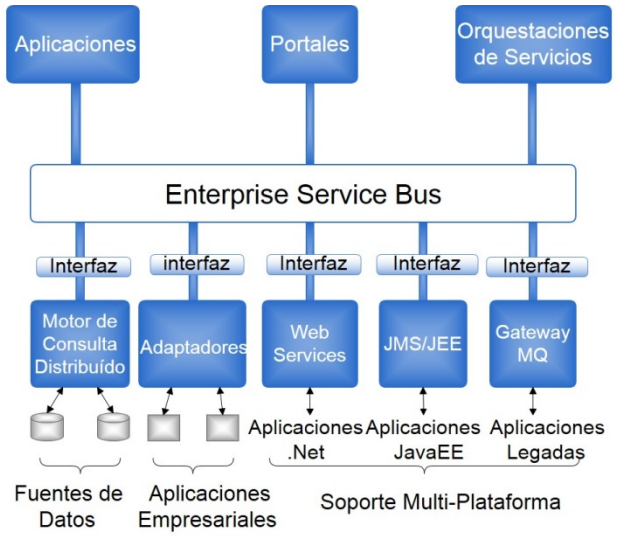

Fig. 1. Enterprise Service Bus [13]

Las funcionalidades provistas por los productos ESB más relevantes para este trabajo son: la transformación y enriquecimiento de mensajes y el ruteo de mensajes [12].

1) Transformación y Enriquecimiento de Mensajes

Los productos ESB brindan una serie de mecanismos para transformar y enriquecer mensajes intercambiados por clientes y servicios, por ejemplo, utilizando el estándar XSLT [14]. Estos mecanismos de transformación pueden utilizarse para afrontar distintos requerimientos, como la resolución de diferencias entre formatos de datos [12].

2) Ruteo de mensajes

Los ESB poseen la capacidad de determinar, en tiempo de ejecución, el destino de un mensaje de acuerdo a distintos factores. Los tipos de ruteo más relevantes para este trabajo son el ruteo basado en contenido y el basado en itinerario.

El ruteo basado en contenido determina el destino del mensaje en base a su contenido. En el caso de mensajes SOAP, la lógica de ruteo puede basarse en el contenido de los cabezales del mensaje o del cuerpo [15]. Por otro lado, el ruteo basado en itinerario (Routing Slip) determina el destino del mensaje en base a un itinerario. El itinerario es incluido en el propio mensaje y especifica al ESB el camino que debe recorrer el mismo [11][15].

C. eXtensible Access Control Markup Language

XACML [16] es un estándar de la OASIS basado en XML que define un lenguaje de políticas de acceso y un lenguaje *request*/*response* para la toma de decisiones de control de acceso. El lenguaje de políticas es utilizado para describir los requerimientos generales de control de acceso, y tiene algunos puntos de extensión para la definición de nuevas funciones y tipos de datos, entre otros. El lenguaje permite realizar consultas para determinar si una cierta acción puede o no realizarse e interpretar el resultado. Las respuestas siempre incluyen uno de los siguientes cuatro valores: *Permit, Deny, Indeterminate* (ocurrió algún error o falta algún valor que hace que la decisión no pueda ser tomada) y *Not Applicable* (la petición no puede ser respondida por este servicio).

En la Fig. 2 se presenta el flujo de información XACML donde se pueden ver los distintos componentes que intervienen: Policy Administration Point, Policy Decision Point, Policy Enforcement Point, Policy Information Point y Context handler.

El *Policy Administration Point* (PAP) es la entidad del sistema que crea las políticas o conjuntos de políticas de acceso. El *Policy Decision Point* (PDP) es la entidad del sistema que evalúa las políticas aplicables y emite una decisión de autorización. El *Policy Enforcement Point* (PEP) es la entidad del sistema que ejecuta el control de acceso, realizando solicitudes de decisión de autorización y haciendo cumplir las decisiones tomadas. El *Policy Information Point* (PIP) es la entidad del sistema que actúa como fuente de atributos. Por último, el *Context Handler* es la entidad del sistema que convierte las solicitudes de decisión del formato nativo de solicitud a la forma canónica XACML y convierte las decisiones de autorización en la forma canónica XACML al formato nativo de respuesta.

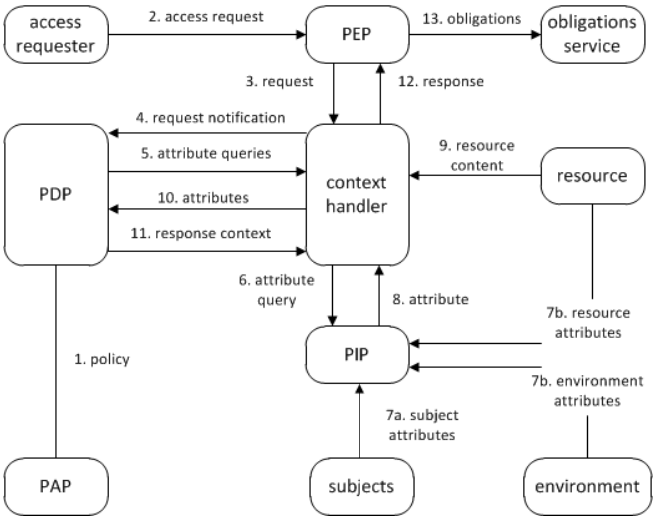

Fig. 2. Flujo de información XACML [16]

La Fig. 3 presenta un ejemplo de una solicitud XACML donde Juan Perez (*subject*) solicita autorización para leer (*action*) su registro médicto (*resource*).

```
<Request>
    <subject>Juan Perez</subject>
    <resource>file://example/med/record/patient/Juan_Perez</resource>
    <action>Read</action>
    <environment></environment>
</Request>
```

Fig. 3. Ejemplo de Request XACML

Por otro lado, la Fig. 4 presenta un ejemplo de respuesta a la solicitud anterior donde se permite (*decision*) el acceso al recurso (*resource*) solicitado por el sujeto (*subject*) para su lectura (*action*).

```
<Response>
    <decision>Permit</decision>
    <status>
        <statusCode>ok</statusCode>
        <statusMessage></statusMessage>
    </status>
    <obligations></obligations>
</Response>
```

Fig. 4. Ejemplo de Response XACML

III. Contexto General y Análisis de Requerimientos

En esta sección se describe el contexto general de la problemática planteada y se presentan los requerimientos identificados. Como se mencionó anteriormente, en este artículo se plantea el interés de incorporar, en plataformas de interoperabilidad, mecanismos que permitan el monitoreo y el control de leyes de protección de datos personales. Más concretamente, este trabajo se enfoca en realidades similares a la uruguaya, es decir, plataformas de interoperabilidad como la provista por la AGESIC y leyes de protección de datos personales análogas a la Ley Nº 18.331 de Uruguay.

A. *Plataforma de Interoperabilidad de Uruguay*

La Plataforma de Interoperabilidad (PDI) de Uruguay tiene como objetivo general facilitar y promover la implementación de servicios de gobierno electrónico en Uruguay. La misma cuenta con dos componentes principales: la Plataforma de Middleware y el Sistema de Seguridad [12].

La Plataforma de Middleware (PM) provee mecanismos para facilitar el desarrollo, despliegue e integración de servicios y aplicaciones. Estos mecanismos brindan a su vez, la infraestructura base para la implementación de una SOA a nivel de Estado. Los organismos pueden utilizar esta plataforma para publicar y consumir servicios. También pueden utilizar las diferentes capacidades de mediación, las cuales permiten desacoplar clientes y servicios. Uno de los principales componentes de la PM es un ESB que provee varios de los mecanismos mencionados.

El Sistema de Seguridad (SS) brinda servicios de seguridad al resto de los componentes y es el encargado de hacer cumplir las políticas de autenticación, autorización y auditoría definidas. En particular, provee mecanismos que permiten realizar el control de acceso a los servicios de la plataforma [12].

Los *web services* provistos por los organismos del estado uruguayo en la PDI tienen un conjunto de métodos. A su vez, cada método tiene un conjunto de elementos que representan los parámetros de entrada o salida del método. En la Fig. 5 se puede observar un modelo con estos conceptos.

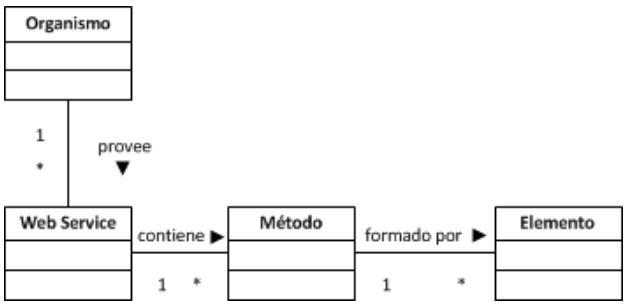

Fig. 5. Modelo conceptual de Web Services

Por ejemplo, el *web service* "CertificadosService" provisto por el Ministerio de Industria, Energía y Minería (MIEM) provee dos métodos: "getCertificadoPymeByRUT" y "getCertificadoCCPByCodigo". Algunos de los elementos del método "getCertificadoPymeByRUT" son: rut, code, desc, razonSocial, tipoDeSociedad, categoría y vigencia [17].

Cuando un organismo desea invocar un servicio de la PDI debe enviar, a dicha plataforma, un mensaje SOAP con los datos de su solicitud. Una vez que el mensaje llega a la PDI el mismo es enviado al SS donde se realiza el control de acceso a los métodos del servicio. Para poder tomar la decisión de autorizar, o no, la invocación a un método de un servicio, el cliente debe especificar el servicio y método que se quiere invocar. Para esto se utiliza el estándar WS-Addressing a través de los elementos "wsa:to" y "wsa:action", respectivamente. Además el cliente debe enviar, utilizando el estándar WS-Security, un *token* de seguridad XML previamente solicitado a la PDI.

Luego de aplicarse los controles de seguridad, el mensaje es dirigido a la PM, donde se le realizan validaciones y, de ser necesario, transformaciones. Por último, el mensaje se envía al servicio destino alojado en un organismo [12].

La Fig. 6 presenta gráficamente la invocación de un servicio en la PDI. En particular, se muestra cómo la Dirección Nacional de Identificación Civil (DNIC) invocaría a un servicio del Banco de Previsión Social (BPS).

Fig. 6. Invocación de un Servicio en la PDI

B. *Ley Nº 18.331 de Protección de Datos Personales y Acción de "Habeas Data"*

La Ley Nº 18.331 de Protección de Datos Personales y Acción de "*Habeas Data*" [5] establece que el derecho de protección de datos personales es inherente a la persona humana y se aplicará por extensión a las personas jurídicas. Un dato personal puede incluir sonidos, imágenes o datos biométricos, entre otros. A modo de ejemplo se consideran datos personales el nombre, apellido, correo electrónico, una fotografía, una huella dactilar, la voz o el ADN. La ley establece que existe un conjunto de datos públicos, es decir, que no es necesario contar con el previo consentimiento del titular de los datos para su tratamiento. Para las personas físicas estos datos son nombres y apellidos, documento de identidad, nacionalidad, domicilio y fecha de nacimiento [5].

Además de datos personales, la ley define los datos sensibles. Son datos personales que revelan el origen racial, étnico, preferencias políticas, convicciones religiosas o morales, afiliación sindical e informaciones referentes a la salud o a la vida sexual. La ley establece que nadie está obligado a proporcionar este tipo de datos y para obtenerlos es necesario contar con el consentimiento expreso del titular. El consentimiento debe ser libre, es decir, lo debe dar de forma voluntaria y los datos podrán ser compartidos dentro de un determinado período de tiempo.

Por otra parte, la ley establece que todo titular tiene derecho a saber qué datos se tienen de sí mismo en cada organismo. Para ello el titular podrá pedir toda la información que se disponga de sí mismo. Dicha información debe ser brindada en un lenguaje claro. El derecho de acceso podrá ser ejercido a intervalos de seis meses y los datos pueden ser suministrados por escrito o por medios electrónicos. El plazo máximo para dar respuesta a estos derechos es de cinco días hábiles.

A partir del análisis de la ley de protección de datos se identificaron los siguientes conceptos:

- **Dato Personal:** cuando se habla de dato personal se hace referencia a información de cualquier tipo referida a personas físicas o jurídicas determinadas o determinables.

- **Ciudadano:** la ley establece que "El derecho de protección de datos personales es inherente a la persona humana". Se utilizará el concepto "ciudadano" para referirse a toda persona física. Pese a que la ley también expresa que el derecho a la protección de datos personales se aplicará por extensión a las personas jurídicas, éstas no serán consideradas en el alcance de este trabajo.
- **Consentimiento:** además de datos personales, se define un subconjunto de los mismos denominados datos sensibles. Se establece que nadie está obligado a brindar estos datos y para obtenerlos es necesario contar con el consentimiento expreso del titular.
- **Organismo:** este concepto hace referencia a los distintos organismos del Estado que se encuentran integrados a la PGE, como por ejemplo la Dirección Nacional de Identificación Civil (DNIC).
- **Finalidad:** cuando un ciudadano brinda su consentimiento para la utilización de sus datos personales, debe conocer para qué serán utilizados. Se define como finalidad aquello para lo que el ciudadano proporciona sus datos personales. Un posible uso de estos datos es la realización de un trámite en un determinado organismo.
- **Acción:** se definen "acciones" como un conjunto de tareas a realizar en caso de que se detecte el incumplimiento de alguno de los aspectos establecidos por la ley de Protección de Datos Personales.

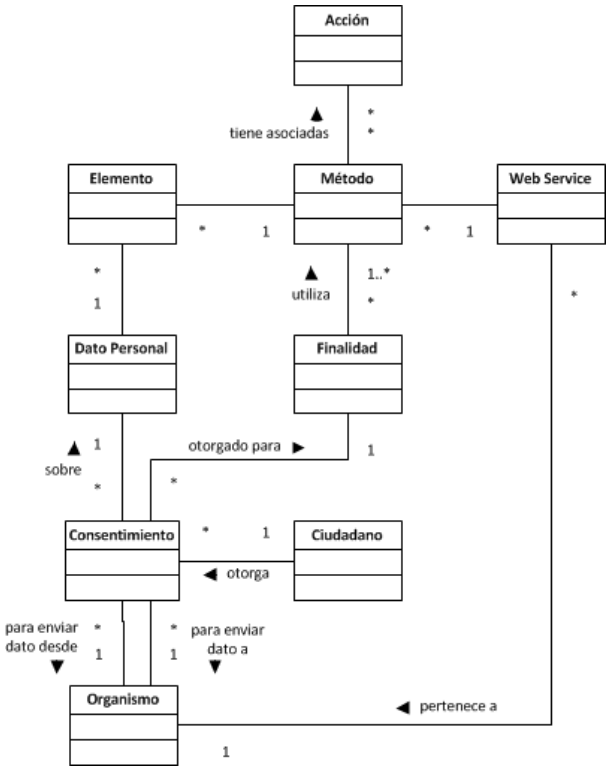

Fig. 7. Modelo Conceptual de la Ley de Protección de Datos

En la Fig. 7 se presenta un modelo conceptual basado en los conceptos anteriores. En el mismo se puede observar que un ciudadano puede otorgar consentimientos para compartir sus datos personales con un organismo en el contexto de una determinada finalidad. La realización de una finalidad puede requerir la invocación de uno o más métodos de distintos *web services*. A cada método es posible asociarle un conjunto de acciones a realizar en caso de que se detecte el incumplimiento de alguno de los aspectos establecidos por la ley al invocar el mismo. La relación entre los elementos y los datos personales se explica en próximas secciones.

C. Requerimientos

A partir de la conceptualización de la ley se definieron los siguientes requerimientos a alto nivel.

La solución deberá controlar los mensajes que se intercambian entre los distintos organismos y garantizar el cumplimiento de la ley de protección de datos. Para esto el sistema debe interceptar todos los mensajes intercambiados entres los organismos que llegan a la PDI, y realizar las validaciones correspondientes para determinar si dichos mensajes satisfacen lo establecido por la ley.

La verificación debe realizarse teniendo en cuenta los organismos que participan en el intercambio de los datos personales, el ciudadano dueño de los mismos, la finalidad bajo la cual se desean intercambiar los datos, la fecha de envío del mensaje y los consentimientos brindados por el titular de los datos.

La solución también debe proveer a los ciudadanos herramientas para poder gestionar los consentimientos brindados a los distintos organismos. Además los ciudadanos deberán contar con mecanismos que les permitan obtener los datos personales que cada organismo dispone de ellos en tiempo y forma de acuerdo a lo especificado en la ley.

Por otra parte, la solución debe permitir a los organismos agregar, modificar o eliminar los consentimientos de los ciudadanos si cuentan con la autorización explícita de los mismos. A su vez, también debe permitir gestionar los distintos pedidos de obtención de datos generados por los ciudadanos.

Por último, la solución debe contar con una herramienta para la configuración del sistema que permita a su vez monitorearlo.

A partir de estos requerimientos se identificaron los siguientes tres tipos de usuarios: Administrador General del Sistema, Administrador de Organismo y Ciudadano.

D. Consideraciones Adicionales

Cabe destacar que si bien el análisis presentado en esta sección se basó en la realidad uruguaya, gran parte del mismo es también aplicable en otros países. Esto es así ya que las PDI son cada vez más utilizadas en contextos de gobierno electrónico[2][18] y que varios países, donde éstas se adoptan, se rigen por normativas de protección de datos personales [19] muy similares a la de Uruguay (p. ej. la Ley Orgánica 15/1999 de España [20]).

IV. Propuesta de Solución

En esta sección se presenta la solución propuesta, denomina *Clouseau,* para abordar la problemática planteada. Más detalles de *Clouseau* se pueden consultar en [21].

En primer lugar, se detalla su arquitectura y se describen sus principales componentes. Luego se presentan y describen las características más relevantes de la misma. Por último, se explica en detalle la interacción entre sus principales componentes, en especial, la que se da entre los módulos PDP y PEP mediante mensajes XACML.

A. Arquitectura General

La solución propuesta consiste de un componente que se incluirá en la PGE y se encargará de brindar funcionalidades a los tres tipos de usuarios identificados en la sección anterior. Por un lado, permite al administrador del sistema configurar los aspectos necesarios para el control de mensajes intercambiados entre organismos así como también monitorear el sistema. Por ejemplo, este usuario puede dar de alta en el sistema a los organismos que se desean controlar. A su vez, permite a los administradores de cada organismo manejar los consentimientos de cada ciudadano y gestionar los pedidos de información creados por los mismos. Por último, cada ciudadano puede gestionar, mediante un sitio web, sus consentimientos y solicitar a cada organismo la obtención de sus datos.

La idea general de *Clouseau* es interceptar todos los mensajes que se reciben y se envían desde y hacia la PDI. Cada mensaje es examinado por la solución la cual verifica su contenido teniendo en cuenta los aspectos de la Ley N° 18.331 previamente analizados. Se realizan validaciones de los tipos de datos personales compartidos teniendo en cuenta los consentimientos disponibles para su utilización y las configuraciones realizadas. Como resultado de ese análisis se realizan distintas transformaciones al mensaje original y eventualmente se toman distintas acciones pre-configuradas (p. ej. transformar los mensajes para quitar elementos que no cuentan con el consentimiento para ser compartidos).

Para poder realizar las tareas antes mencionadas el sistema debe almacenar un conjunto de datos. Entre otras cosas se almacenan los consentimientos brindados por cada ciudadano, la configuración de cada *web service* expuesto por los distintos organismos y las distintas acciones a tomar. En la Fig. 8 se muestra un esquema conceptual de la solución propuesta.

En la figura se presenta un ejemplo de comunicación entre la DNIC y el BPS. Como se mencionó anteriormente dicha comunicación se da a través de la PDI, en particular, utilizando el SS y la PM. La solución se basa en incluir un módulo dentro de la PGE, más concretamente en la PDI, por el cual pasan todos los mensajes intercambiados entre los organismos. Este módulo está formado por cuatro componentes: PEP, PDP, ESB y *web*.

El componente PEP es el encargado de recibir los mensajes y enviar la solicitud de acceso XACML al componente PDP para la evaluación de consentimientos. El componente PDP se encarga de evaluar la petición XACML y generar la respuesta al módulo PEP en base a los consentimientos disponibles. El componente ESB es el encargado de realizar el ruteo de mensajes y aplicarles las transformaciones necesarias (p. ej. quitar elementos del mensaje). Por último, el componente *web* es el componente mediante el cual el administrador del sistema, los administradores de cada organismo y los ciudadanos pueden configurar y gestionar distintos aspectos del sistema.

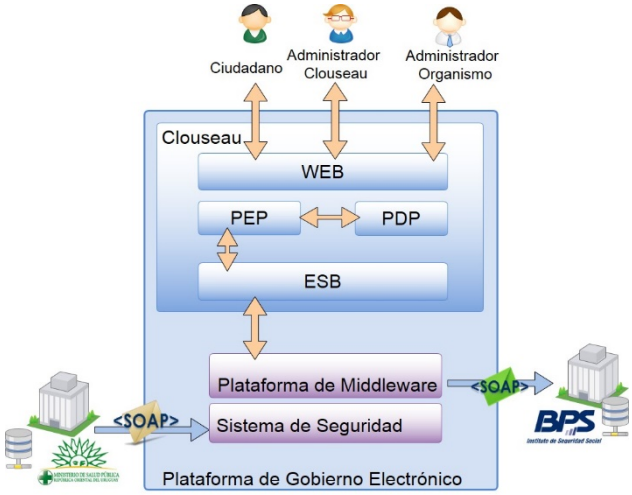

Fig. 8. Solución General

B. Principales Características

A continuación se presentan algunos de los aspectos más importantes de la solución propuesta.

1) Intercepción de requests

Uno de los principales requerimientos del sistema es el análisis de todos los mensajes que se intercambian entre los distintos organismos en la PDI. Para llevar este requerimiento a cabo se optó por exponer un *endpoint* en el cual se reciben todos los mensajes intercambiados, tanto los *requests* como los *responses*. Como respuesta a este análisis se retornará un mensaje que cumpla con las normativas establecidas o, en caso contrario, un mensaje de error.

2) Configuración de web services

Para poder evaluar los distintos tipos de mensajes intercambiados en la PDI era necesario conocer los detalles de los *web services* disponibles. También era necesario conocer qué organismos proveía cada uno de ellos. Para esto se optó por crear un *backoffice* de administración en el cual, entre otras cosas, se puede configurar cada *web service* de la PDI. Analizando el conjunto de *web services* que se encuentran en el catálogo de servicios [22] brindado por la AGESIC se observó que cada *web service* está compuesto por un conjunto de métodos. A partir de esto se decidió que la configuración en el sistema sea a nivel de métodos. Para la configuración de un nuevo método el administrador debe indicar a qué organismo pertenece y subir un archivo WSDL con la descripción del *web service* correspondiente.

3) Modelo de datos canónicos

Dado que cada organismo utiliza su propio modelo de datos para representar los datos personales de los

ciudadanos, fue necesario diseñar un mecanismo que permita unificar esa diversidad. Por ejemplo, el nombre de un ciudadano puede denominarse "nomPersona" en un método de un *web service* de un organismo y "nombre" en otro organismo o incluso en otro método del mismo organismo. Para resolver este problema se decidió crear un modelo de datos canónico para representar los datos personales, siguiendo el patrón Modelo de Datos Canónico (MDC) [23] y teniendo en cuenta el modelo canónico de Persona definido y utilizado por AGESIC [24].

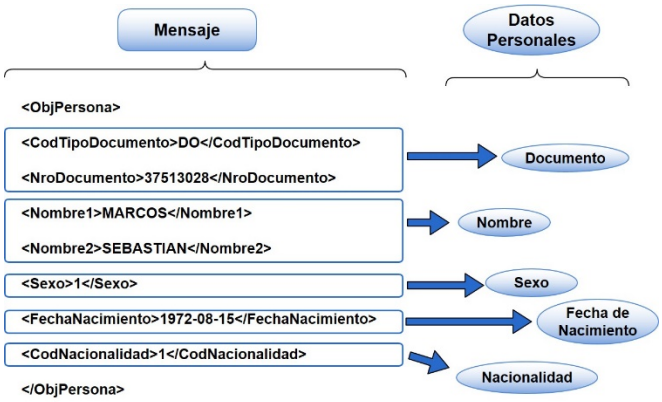

Fig. 9. Correspondencia entre elementos y datos personales

Se optó entonces por definir en el sistema un MDC que incluye los siguientes datos personales: Documento, Nombre, Apellido, Sexo, Fecha de Nacimiento y Nacionalidad. Al momento de que el administrador realice el alta de un nuevo método en el sistema, se debe especificar la correspondencia entre estos datos personales y los elementos del método que hagan referencia a ellos. En la Fig. 9 se puede observar un ejemplo de correspondencia entre los elementos de un mensaje y los datos personales definidos en el sistema.

4) Tipos de Datos Personales

Para identificar qué elementos de un método deben ser verificados a la hora de comprobar los consentimientos disponibles, se deben clasificar los datos personales definidos en el MDC en una de las siguientes categorías:

- *Free*: Este tipo de dato personal no requiere consentimientos para ser compartidos.
- *Limited*: Es necesario contar con el consentimiento explícito del titular para compartir este tipo de datos.
- *Denied*: Este tipo de datos nunca es compartido.

5) Transformaciones y acciones a tomar

Cuando en la invocación a un servicio, o en la respuesta correspondiente, se detecta que un organismo quiere compartir datos de tipo "*Limited*" sin el consentimiento explícito del titular de los mismos, la solución debe tomar acciones para garantizar que la ley de protección de datos personales se cumpla. Para esto, la solución brinda posibilidades de configuración para realizar distintas transformaciones al mensaje y tomar determinadas acciones.

En primer lugar, para cada método configurado en el sistema es posible especificar una transformación XSLT. Esta transformación se aplicará al mensaje en caso de detectar que no se cuenta con todos los consentimientos necesarios para enviarlo a un determinado organismo. Una posible transformación podría filtrar del mensaje los datos no permitidos por el titular de los mismos.

A modo de ejemplo, en la Fig. 10 se observa que al no contar con los consentimientos sobre los datos personales Documento y Sexo, estos datos fueron filtrados (imagen a la derecha) del mensaje original (imagen a la izquierda).

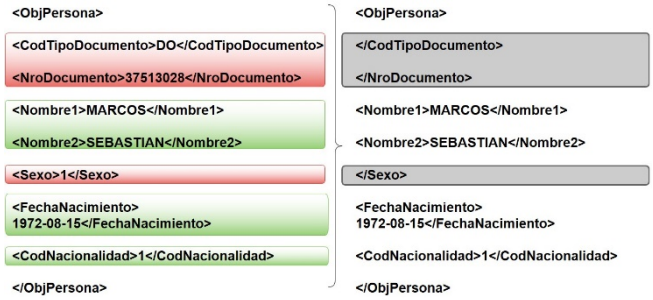

Fig. 10. Ejemplo de filtrado de mensaje

Adicionalmente, para cada método se podrá especificar también un conjunto de acciones, disponibles en el sistema, las cuales se ejecutarán en los mismos casos que la transformación. Un ejemplo de acción podría ser la notificación vía mail de la falta de consentimientos. Además se decidió implementar la creación de acciones de forma extensible para que se puedan registrar nuevas acciones en el sistema cuando se desee.

El conjunto de acciones a ejecutar y el archivo con la transformación XSLT se puede configurar al momento de dar de alta en el sistema cada método.

6) Aplicación Web

Se consideró importante que el ciudadano cuente con una aplicación *web* en la cual pueda consultar y gestionar sus consentimientos brindados a los organismos. De esta forma el ciudadano puede tener un mayor control sobre sus datos personales y por ejemplo dejar de compartirlos cuando lo desee. También desde la aplicación *web* podrá crear solicitudes de accesos a datos sin necesidad de concurrir al organismo correspondiente. Estas solicitudes se enviarán a cada organismo y, una vez que se completen, se le notificará al usuario vía mail que las mismas ya están disponibles.

7) Cálculo de consentimientos

Cuando un ciudadano desea realizar un trámite en un organismo puede ser que el mismo requiera la invocación de uno o más métodos de varios *web services*. A su vez cada uno de estos métodos puede requerir que el ciudadano brinde distintos consentimientos para la transferencia de sus datos personales. Para resolver este problema se decidió contar con una funcionalidad de cálculo de consentimientos.

La idea es que dado un ciudadano y una finalidad el administrador de un organismo pueda obtener los consentimientos que hacen falta en el sistema para poder

realizar la finalidad deseada. Para poder calcular los consentimientos faltantes se obtiene la lista de datos personales que intervienen en todos los métodos de la finalidad (p. ej. un trámite) y se verifica que el ciudadano haya dado el consentimiento para utilizar esos datos personales. De cada dato personal se busca un consentimiento para los organismos involucrados que esté dentro del periodo de validez.

A partir de esta verificación se obtiene la lista de consentimientos faltantes y se brinda la posibilidad al administrador de agregarlos luego de contar con la aprobación explícita del ciudadano.

8) Validez de Consentimientos

Tal cual lo especifica la ley, para cada consentimiento que brinda el ciudadano es posible especificar un período de validez. Por este motivo, pueden existir almacenados en el sistema consentimientos que ya se encuentran expirados. Se implementó entonces un purgador que se encarga de eliminar consentimientos ya expirados del sistema.

C. Principales Componentes y su Interacción

La solución *Clouseau* cuenta con los siguientes módulos: Clouseau-web, Clouseau-service, Clouseau-repo, Clouseau-flow y Clouseau-service-flow. Se decidió separar la aplicación en estos cinco módulos para tener un bajo nivel de acoplamiento y teniendo en cuenta las responsabilidades de cada uno.

1) Clouseau-web

Es el componente *web* de la solución. El mismo puede ser utilizado por los administradores, para tareas de configuración y monitoreo, y por los ciudadanos, para la gestión de sus consentimientos.

2) Clouseau-service

Este módulo contiene la lógica de negocio utilizada por la aplicación *web*. Se comunica con el módulo Clousau-repo para el manejo de datos.

3) Clouseau-repo

Este módulo se encarga de manejar el acceso a la base de datos.

4) Clouseau-service-flow

Este módulo es quien contiene la lógica relacionada con la verificación de consentimientos. Es el componente PDP de la solución ya que evalúa los consentimientos disponibles y emite una decisión de autorización para los mensajes. Se comunica con el módulo Clousau-repo para acceder a los consentimientos almacenados en la base de datos.

Para evaluar los consentimientos disponibles se utilizan los datos provenientes del módulo PEP, los cuales son obtenidos a partir de los cabezales WS-Addressing y WS-Security. Además se define un cabezal denominado Clouseau-Info el cual contiene la finalidad bajo la cual se invoca el servicio e información (documento de identidad) que permite identificar al titular de los datos personales. El módulo PDP obtiene los consentimientos del ciudadano que coinciden con los datos enviados por el módulo PEP y se encuentran dentro del período de validez. A partir de los mismos toma la decisión de autorización que es enviada como respuesta al módulo PEP.

5) Clouseau- flow

Es el módulo PEP de la solución. Está basado en un ESB lo que permite aprovechar varias de sus capacidades. En particular, este módulo consta de un *endpoint* a través del cual se interceptan todos los mensajes SOAP que llegan a la PDI para invocar a los servicios. El módulo también utiliza los patrones “Routing Slip” y “Ruteo basado en contenido” que generalmente incluyen los ESBs. En la Sección V se brinda más detalle del funcionamiento de este componente.

6) Interacción entre Componentes

Como se comentó en la descripción de componentes, los módulos Clouseau-flow y Clouseau-service-flow cumplen la función de PEP y PDP del estándar XACML, respectivamente. En la Fig. 11 se encuentra un diagrama de secuencia en el cual se muestra la interacción que se da entre estos componentes.

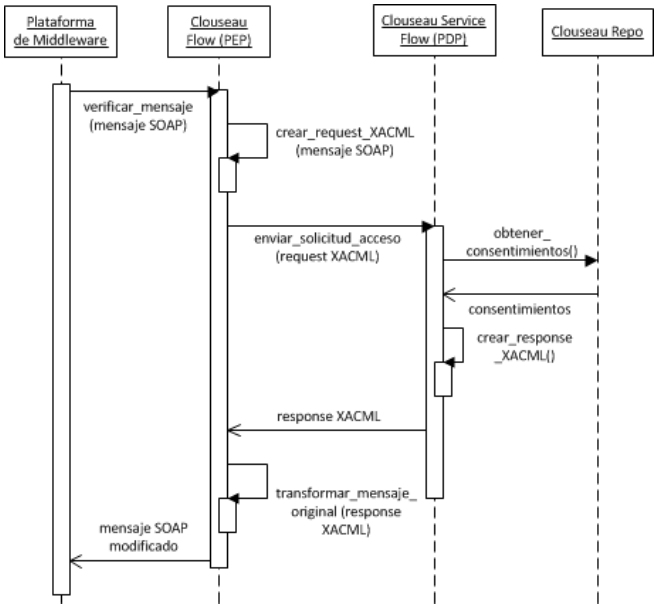

Fig. 11. Diagrama de Secuencia

A continuación se describe a alto nivel cada uno de los pasos que se realizan a la hora de procesar un mensaje:

1. Cuando llega un mensaje para invocar un servicio de la PDI, se envía al módulo Clouseau-flow (PEP)
2. El módulo Clouseau-flow valida la estructura del mensaje y a partir del mismo construye un *request* XACML.
3. El *request* XACML previamente construido es enviado al módulo Clouseau-service-flow para obtener una respuesta de acceso.
4. En módulo Clouseau-service-flow obtiene del módulo Clouseau-repo los consentimientos disponibles en el sistema y en base a éstos construye una respuesta XACML.

5. La respuesta XACML es enviada al módulo Clouseau-flow, el cual a partir de la misma realizará las transformaciones necesarias al mensaje.

El mensaje modificado es retornado hacia la plataforma de middleware.

Como se explicó anteriormente, la comunicación entre los módulos Clouseau-flow y Clouseau-service-flow se da a través de mensajes XACML. A continuación se muestra cómo se utilizan cada uno de los elementos de los mensajes XACML en la solución propuesta.

En el *request* se utilizan los elementos XACML de la siguiente forma:

- *Subject*: es la entidad que solicita el acceso. En este caso será el identificador del organismo que desea enviar el mensaje.
- *Resource*: específica información sobre el recurso para el que se solicita el acceso, en este caso el recurso es el propio mensaje que se desea enviar.
- *Action:* especifica la acción que se realizará sobre el recurso. En esta solución la acción siempre será "Send" ya que lo que se desea es enviar el mensaje.
- *Enviroment:* es un conjunto de atributos que son relevantes para la decisión de autorización independiente de Subject, Resource o Action. Se utiliza esta sección para indicar el propósito por el cual se requiere enviar el mensaje, identificador del organismo destino del mensaje y documento del ciudadano del cual se quiere enviar datos.

En cuanto al *response*, estos son los elementos XACML que se utilizaron:

- *Decision*: decisión de autorización. Se utilizan los valores "Permit" y "Deny" definidos en el estándar XACML.
- *Status*: Indica si se han producido errores durante la evaluación de la solicitud de decisión y opcionalmente información acerca de esos errores.
- *Obligations:* lista de operaciones que deben ser ejecutadas por el PEP en relación a una decisión de autorización. En esta sección se agregan las acciones que se deben realizar en caso de que no se cumpla con alguno de los consentimientos necesarios para enviar el mensaje (p. ej. el envío de un mail notificando la falta de consentimientos). Además se agrega el nombre de la transformación XSLT que se debe realizar al mensaje si es necesario. Por ejemplo dicha transformación podría filtrar los datos personales no autorizados. Por último se envía la lista de parámetros que no pasaron la evaluación de consentimientos.

V. Detalles de Implementación

En esta sección se brindan detalles de implementación del prototipo desarrollado para la solución propuesta. Una demo del prototipo puede verse en [25].

El prototipo se implementó utilizando JavaEE 7 y su despliegue se realizó sobre la plataforma JBossEAP. La implementación de la solución se basó fuertemente en SwitchYard ESB. Además se utilizó Spring Framework para resolver el problema del manejo del ciclo de vida de los objetos, la inyección de dependencias y el manejo de transacciones. Para el mapeo entre objetos y el modelo de base de datos se utilizó Hibernate y MySQL como motor de base de datos. Para el componente web se utilizó JSF junto con Bootstrap y para el *dashboard* de administración se utilizó la librería Morris.js.

La Fig. 12 presenta los componentes implementados y las principales herramientas utilizadas en cada uno de ellos.

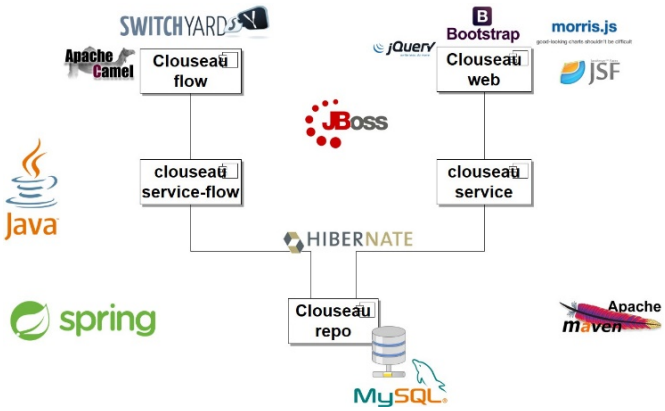

Fig. 12. Herramientas utilizadas

A continuación se describen detalles sobre la implementación de alguna de las funcionalidades.

A. ESB

Como ya se mencionó Clouseau-flow es el componente ESB de la solución. El mismo consta de un *endpoint* por el cual se interceptan todos los mensajes SOAP que llegan a la PM de la PDI. Funciona como PEP ya que es quien realiza las solicitudes de decisión de autorización y se encarga de que se cumplan las decisiones tomadas.

En la Fig. 13 se muestra un esquema con los sub-componentes del módulo.

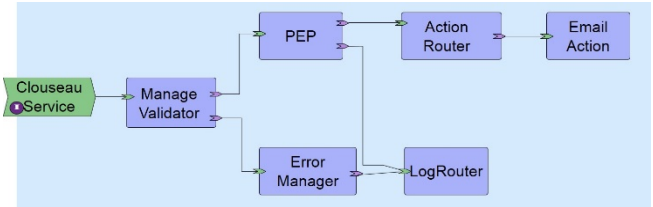

Fig. 13. Flujo ESB

A continuación se describe cada uno de sus componentes.

1) Message Validator Router

Es el encargado de verificar si el mensaje es estructuralmente válido, es decir si posee todos los datos necesarios para realizar el proceso de validación de permisos. En caso de que el mensaje sea correcto es redirigido al componente "PEP", de lo contrario es redirigido al componente "ErrorManagerRouter".

2) Error Manager Router

Es el componente encargado de agregar al cuerpo del mensaje el error correspondiente. Luego, el mensaje es redirigido al componente "LogRouter".

3) PEP

Componente encargado de extraer del mensaje SOAP los datos necesarios para crear el *request* XACML que será enviado a Clouseau-service-flow (PDP).

En la respuesta XACML del módulo PDP se especifica si los organismos involucrados (tanto el emisor como el receptor) están autorizados para intercambiar los datos personales incluidos en el mensaje. En caso de no estar autorizados se incluye en la respuesta las acciones a ejecutar.

Si el módulo PDP determina que los organismos poseen los consentimientos necesarios para el intercambio de los datos personales, se redirige el mismo al componente "LogRouter". De lo contrario es redirigido al componente "ActionRouter".

4) Action Router

Es el encargado de extraer del resultado generado por el módulo PDP las acciones que deben ser ejecutadas y la transformación XSLT que se debe aplicar al mensaje. Tanto las acciones como la transformación que se desea aplicar deben ser configuradas al momento de dar de alta un nuevo método en el sistema.

Una posible acción podría ser un componente que envíe un email. Por otro lado, un ejemplo de transformación podría ser una que, dado un mensaje SOAP y una lista de elementos, retorne el mensaje recibido quitando los valores de los elementos de la lista recibida.

Para ejecutar la transformación y las acciones configuradas se utiliza el patrón de ruteo Routing Slip. Para ello se añade en el *header* del mensaje la lista de los componentes que debe recorrer.

5) Email Action

Componente encargado de enviar emails al administrador de Clouseau cada vez que se detecte la ausencia de consentimientos en los mensajes recibidos por el sistema.

6) Log Router

Componente encargado de registrar en el sistema datos de cada mensaje interceptado. Los datos más importantes almacenados son: organismo emisor, organismo destino, nombre del método invocado, mensaje recibido y mensaje retornado.

B. Procesamiento de archivos WSDL

A la hora de dar de alta un nuevo método en Clouseau, se cuenta con una interfaz que permite al administrador cargar el archivo WSDL del *web service*. A partir del archivo se procesa su contenido y se obtiene automáticamente datos del *web service*, como el identificador de sus métodos, su descripción y la lista de elementos que reciben o devuelven sus métodos. Esta lista se utilizará luego al momento de especificar la correspondencia de estos elementos con los datos personales del MDC definido en el sistema.

La obtención de datos directamente del archivo facilita al administrador realizar el alta de un nuevo método de un *web service* en el sistema y en consecuencia reduce la posibilidad de errores manuales. Para el procesamiento del archivo WSDL se utilizó la librería javax.wsdl. Cabe destacar que en caso de que ocurra un error en el procesamiento del archivo WSDL se da la posibilidad al usuario de que complete los campos de forma manual.

C. Creación de Acciones

Para la creación de acciones que se ejecutarán en caso de detectar la falta de consentimientos, fue necesario diseñar algún mecanismo que permita agregar nuevas acciones fácilmente. Principalmente se buscaba agregar acciones realizando la menor cantidad de cambios en la aplicación. Para resolver este problema se utilizó el soporte de *reflection* provisto por Java.

Reflection es la capacidad que tiene un programa para conocer y en ocasiones modificar su estructura de alto nivel. Cuando el código fuente de un programa se compila, pierde la información sobre su estructura ya que se genera código de bajo nivel. *Reflection* permite, en tiempo de ejecución, volver a conocer la estructura de alto nivel.

De esta forma, para utilizar un nuevo tipo de acción en el sistema se debe crear una nueva clase Java y agregarla a la solución. El único requisito de esta clase es que tiene que extender de una clase genérica la cual contiene un método encargado de obtener los datos que se envían al módulo PEP en la respuesta XACML. Estos datos se utilizan en el componente PEP para ejecutar la acción. Gracias a *reflection* se creará, en tiempo de ejecución, la instancia de esta nueva clase, con solo conocer el nombre de la misma el cual se almacena y se podrá obtener desde la base de datos.

D. Scheduling Tasks

Para implementar el purgador de consentimientos expirados se utilizó el mecanismo de *schedule task* que brinda Spring. Con una simple anotación en el método deseado, Spring ejecuta esa tarea en intervalos de tiempo configurables mediante expresiones *cron*.

A pesar de que se implementó solo una *schedule task*, el diseño se realizó considerando la posibilidad de agregar más tareas fácilmente. Para esto se creó una tabla en la base de datos donde se almacenan todas las tareas. En particular de cada tarea se almacena: nombre, fecha de última ejecución de la tarea, fecha de última ejecución exitosa de la tarea, estado (activo/inactivo) y el último mensaje de error.

Estos datos permiten monitorear las tareas en caso de fallas, así como también habilitar o deshabilitar su ejecución sin necesidad de tener que volver a desplegar la aplicación.

E. Dashboard

El *backoffice* de administración ofrece un *dashboard* donde es posible visualizar varias métricas sobre el tráfico de mensajes tomadas por el sistema. Como se mencionó anteriormente, el sistema guarda un registro con los datos más relevantes de cada mensaje SOAP recibido. Dicho registro es almacenado en una tabla en la base de datos.

La Fig. 14 se presenta un ejemplo de información que se puede obtener en el *dashboard*.

Fig. 14. Información brindada por el Dashboard

VI. Trabajo Relacionado

En los últimos años muchos trabajos han abordado problemáticas asociadas a la protección de datos personales. En [26] se propone un enfoque para controlar el cumplimento de normativas de privacidad de datos en un contexto inter-organizacional en el área de la salud. Los autores proponen un lenguaje específico de dominio para dar soporte a la especificación y control de reglas de privacidad.

Por otro lado, en [27] se formalizan los requerimientos de protección de datos introducidos por las normativas alemanas y se presenta una herramienta semiautomática para ayudar a los proveedores de servicios a que sus servicios cumplan con estas normativas.

Además, recientemente han surgido también trabajos que apuntan a controlar y asegurar el cumplimiento de leyes de privacidad de datos en plataformas *Cloud* [28][29].

Sin embargo, comparado con el enfoque presentado en este artículo, estos trabajos no proveen mecanismos que de forma dinámica tomen acciones para asegurar el cumplimiento de estas leyes en intercambios inter-organizacionales. Por otro lado, ninguno de estos trabajos aborda la problemática de protección de datos en plataformas de interoperabilidad de gobierno electrónico.

Por último, en [30] se plantea la problemática de la protección de datos personales en un contexto de gobierno electrónico. Sin embargo, el trabajo se centra en las interacciones que se dan en soluciones de Master Data Management (MDM).

VII. Conclusiones y Trabajo a Futuro

En este artículo se propusieron soluciones que permiten gestionar, monitorear y asegurar el cumplimiento de normativas asociadas a la protección de datos personales en plataformas de interoperabilidad de gobierno electrónico.

Los principales aportes del trabajo son: i) el análisis de la problemática planteada en el cual se identificaron un conjunto de requerimientos, ii) la propuesta de solución basada tecnologías de integración y estándares de seguridad ampliamente reconocidos y iii) la implementación de la solución con un producto ESB específico que permitió validar la propuesta.

Por otro lado, aunque el análisis y la solución fueron planteados en el contexto de la PGE de AGESIC y la ley de protección de datos personales de Uruguay, los mecanismos propuestos pueden ser aplicados en otros contextos similares. En este sentido, este trabajo representa un paso más para avanzar en la discusión de estrategias que permitan, de forma eficiente, monitorear y controlar el cumplimiento de normativas, en particular de protección de datos personales, en plataformas de interoperabilidad de gobierno electrónico [31].

En cuanto a los estándares y tecnologías utilizadas en la propuesta, se comprobó que el estándar XACML se adecúa en gran medida a las necesidades planteadas por este tipo de leyes. Por otro lado, los ESBs brindan características que facilitan implementar acciones para garantizar el cumplimiento de estas normativas (p. ej. transformaciones). En particular, el producto SwitchYard tiene una gran potencialidad y resuelve nativamente varios de los problemas planteados. Sin embargo, se tuvo algunas dificultades en la implementación por lo que se entiende que tiene algunos puntos en los que mejorar.

Como línea de extensión a este trabajo se identificó la integración de la solución propuesta con un motor de procesamiento de eventos complejos (Complex Event Processing, CEP). Esta integración podría permitir la generación de alertas cuando se detecten determinados eventos en tiempo real. Por ejemplo, cuando se detecte un número determinado de mensajes bloqueados para un mismo organismo. También podría usarse CEP para facilitar el cumplimiento de requerimientos temporales de estas leyes. En particular, la ley de protección de datos de Uruguay establece que todo ciudadano tiene derecho a obtener la información que sobre sí mismo se tenga en cada organismo dentro de los cinco días hábiles de haberla solicitado.

Por último, otras líneas de trabajo a futuro incluyen: i) analizar el uso del Privacy Policy Profile introducido en la versión 3.0 de XACML para el manejo del concepto de finalidad, ii) analizar y proponer soluciones para escenarios donde se utilicen otras características avanzadas de *web services* (p. ej. encriptado y firma digital de los mensajes SOAP) ya que esto podría hacer no viable la transformación de mensajes y iii) analizar y evaluar el impacto que tiene una solución de este tipo en los tiempos de respuesta de los servicios que se controlan.

References

[1] J. Akeroyd, Information Architecture and e-Government INFuture2009:“Digital Resources and Knowledge Sharing, p. 687–701, 2009.

[2] R. Baldoni, S. Fuligni, M. Mecella y F. Tortorelli, The Italian e-Government Enterprise Architecture: A Comprehensive Introduction with Focus on the SLA Issue, de Service Availability, Springer Berlin Heidelberg, 2008, pp. 1-12.

[3] L. González, R. Ruggia, J. Abin, G. Llambías, R. Sosa, B. Rienzi, D. Bello y F. Alvarez, A Service-oriented Integration Platform to Support a Joined-up E-government Approach: The Uruguayan Experience, de Joint International Conference on Electronic Government, the Information Systems Perspective, and Electronic Democracy, Vienna, Austria, 2012.

[4] Y. Wu, Protecting personal data in E-government: A cross-country study, Government Information Quarterly, pp. 150-159, 2014.

[5] Parlamento Uruguayo, Ley Nº 18.331 - Protección de Datos Personales y Acción de "Habeas Data",

http://www.parlamento.gub.uy/leyes/AccesoTextoLey.asp?Ley=18331. [Último acceso: 14 02 2015].

[6] W3C, Web Services Description Requirements, http://www.w3.org/TR/ws-desc-reqs/#definitions. [Último acceso: 14 02 2015].

[7] W3C, SOAP Specifications, http://www.w3.org/TR/soap/. [Último acceso: 16 02 2015].

[8] W3C, WSDL Specifications, http://www.w3.org/TR/wsdl. [Último acceso: 14 02 2015].

[9] W3C, Web Services Addressing Working Group, http://www.w3.org/2002/ws/addr/. [Último acceso: 14 02 2015].

[10] OASIS, WS-Security 1.1, https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf. [Último acceso: 14 02 2015].

[11] D. Chappell, Enterprise Service Bus. O'Reilly., 2004.

[12] AGESIC, Descripción y Guías de uso de la Plataforma de Gobierno Electrónico del Estado Uruguayo, http://www.agesic.gub.uy/innovaportal/file/1295/1/descripcion_y_guias_de_uso_de_la_plataforma_de_gobierno_electronico_del_estado_uruguayo.pdf. [Último acceso: 18 02 2015].

[13] M. P. a. W. Heuvel, Service oriented architectures: approaches.

[14] W3C, XSL Transformations (XSLT), http://www.w3.org/TR/xslt. [Último acceso: 14 02 2015].

[15] L. González, Plataforma ESB Adaptativa para Sistemas, 2011.

[16] OASIS, XACML-Specification, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml. [Último acceso: 16 02 2015].

[17] AGESIC, CertificadosService, http://www.agesic.gub.uy/innovaportal/v/3392/1/agesic/dinapyme_consulta_de_certificados.html. [Último acceso: 30 03 2015].

[18] Gobierno de Canarias, Plataforma de Interoperabilidad del Gobierno de Canarias, http://www.gobiernodecanarias.org/platino/. [Último acceso: 24 May 2015].

[19] Unidad Reguladora y de Control de Datos Personales, Leyes Internacionales de Protección de Datos Personales, http://datospersonales.gub.uy/inicio/normativa/internacional/. [Último acceso: 24 May 2015].

[20] Agencia Española de Protección de Datos, Agencia Española de Protección de Datos - Estatal, http://www.agpd.es/portalwebAGPD/canaldocumentacion/legislacion/estatal/index-ides-idphp.php. [Último acceso: 24 May 2015].

[21] A. Echevarría y D. Morales, Protección de Datos Personales en Plataformas de Integración, 4 2015. http://www.fing.edu.uy/inco/grupos/lins/informes/pg-pdp-pi.pdf. [Último acceso: 25 5 2015].

[22] AGESIC, Catálogo de Servicios, http://www.agesic.gub.uy/innovaportal/v/1602/1/agesic/catalogo_de_servicios.html. [Último acceso: 14 03 2015].

[23] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Prof, 2003.http://www.eaipatterns.com/CanonicalDataModel.html. [Último acceso: 2015].

[24] AGESIC, Modelo persona AGESIC, http://www.agesic.gub.uy/innovaportal/v/2533/1/agesic/metadatos_modelo_de_referencia_persona.html. [Último acceso: 01 03 2015].

[25] A. Echevarría y D. Morales, Demo de Clouseau. http://www.fing.edu.uy/inco/grupos/lins/demos/demo-clouseau.mp4. [Último acceso: 25 5 2015].

[26] G. Armellin, D. Betti, F. Casati, A. Chiasera, G. Martinez y J. Stevovic, Privacy Preserving Event Driven Integration for Interoperating Social and Health Systems, de Secure Data Management, Springer Berlin Heidelberg, 2010, pp. 54-69.

[27] C. Sillaber y R. Breu, Managing legal compliance through security requirements across service provider chains: A case study on the German Federal Data Protection Act., de GI-Jahrestagung, 2012.

[28] P. Yu, J. Sendor, G. Serme y A. S. d. Oliveira, Automating Privacy Enforcement in Cloud Platforms, de Data Privacy Management and Autonomous Spontaneous Security, Springer Berlin Heidelberg, 2013, pp. 160-173.

[29] A. S. D. Oliveira, J. Sendor, A. Garaga y K. Jenatton, Monitoring Personal Data Transfers in the Cloud, de 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), 2013.

[30] F. Piedrabuena, L. González, and R. Ruggia, "Enforcing Data Protection Regulations within e-Government Master Data Management Systems," in 17th International Conference on Enterprise Information Systems, Barcelona, Spain, 2015.

[31] L. González and R. Ruggia, "Towards a Compliance-Aware Inter-organizational Service Integration Platform," in On the Move to Meaningful Internet Systems: OTM 2014 Workshops, 2014, pp. 8–17.