Instituto de Computación - Facultad de Ingeniería

Universidad de la República

Montevideo, Uruguay

Gestión de Normativas para Plataformas de Integración Interorganizacionales

Informe de Proyecto de Grado

(Octubre 2017)

Docente: MSc. Ing. Laura González

Autores:

Germán Wolman González

Darío Wolman González

Resumen

Las organizaciones necesitan interactuar para intercambiar datos y funcionalidades de negocio. Las Plataformas de Integración facilitan esta interacción brindando capacidades de conectividad y mediación. Estos ambientes colaborativos e integrados tienen que cumplir con normativas que pueden originarse de leyes, estándares, buenas prácticas y acuerdos, entre otros.

En este contexto, surge la necesidad de que las plataformas de integración brinden mecanismos para la gestión y monitoreo de normativas. Este proyecto propone una solución que brinda mecanismos para la gestión, representación y visualización de normativas a ser controladas en una Plataforma de Integración Interorganizacional.

En primer lugar se analiza la problemática a resolver identificando requerimientos y estudiando cada uno de los conceptos relacionados. Una vez que se realiza este análisis, se relevan los requerimientos y se analizan diferentes formas de especificar los distintos tipos de normativas manejados en el proyecto: calidad de servicio, acuerdos de nivel de servicio, intercambio de mensajes entre organizaciones y protección de datos personales.

En segundo lugar, se propone una solución que permite a organizaciones que interactúan a través de una plataforma de integración gestionar normativas independientemente de su lenguaje de especificación.

Se implementan prototipos cubriendo las herramientas y tecnologías a utilizar. En esta etapa se validan los prototipos y se implementan los casos de uso críticos para la arquitectura de la solución propuesta. Luego se implementan el resto de los casos de uso.

Finalmente, se desarrolla un caso de estudio acorde al contexto de la solución planteada incluyendo múltiples organizaciones y abarcando los tipos de normativas estudiados en el proyecto. Se realizan pruebas para verificar el funcionamiento de la herramienta y evaluar el comportamiento de la plataforma.

***Palabras claves*:** gestión de normativas, plataformas de integración, protección de datos personales, calidad de servicio, acuerdos de nivel de servicio, interacciones interorganizacionales.

Tabla de contenido

1. Introducción

En este documento se presenta el proyecto de grado “Gestión de Normativas para Plataformas de Integración Interorganizacionales”. En esta sección se detalla el contexto y la motivación del proyecto, se plantean sus objetivos, se presentan los aportes del mismo y finalmente se describe la organización del resto del documento.

1.1. Contexto y motivación

En la actualidad cada vez es más necesario que las organizaciones interactúen entre sí para intercambiar datos y funcionalidades de negocio. Esto por lo general se realiza siguiendo el enfoque orientado a servicios que permite la integración y composición de servicios de software distribuidos. Para dar soporte a estas organizaciones resulta fundamental contar con una plataforma de integración. Las plataformas de integración son infraestructuras especializadas que incluyen una capa de procesamiento intermedio para facilitar la provisión e invocación de servicios de software interoperables [1].

En estos contextos de integración a gran escala es cada vez más común que las organizaciones y el sistema en su conjunto tengan que cumplir con normativas que pueden originarse de estándares, buenas prácticas y acuerdos, entre otros [1]. En particular, surge la necesidad de cumplir con normativas en la colaboración entre organizaciones. Estas normativas pueden referirse a Calidad de Servicio (p. ej.: el tiempo de respuesta de un servicio), Acuerdos de Nivel de Servicio (p. ej.: acuerdos entre organizaciones acerca de la cantidad de invocaciones que puede recibir un determinado servicio), intercambio de mensajes entre organizaciones (p. ej.: flujo de interacción para registrar un cliente) y Protección de Datos Personales (p. ej.: autorización para compartir los datos de una persona) [1].

En la Figura 1, se presenta un escenario en el que organizaciones intercambian información a través de una plataforma de integración. En particular se presenta esta realidad en el contexto de gobierno electrónico donde dos organismos del gobierno: Banco de Previsión Social (BPS) y Dirección General Impositiva (DGI), intercambian información mediante servicios. Este escenario muestra al BPS solicitando a la DGI datos sobre un ciudadano uruguayo particular. En este flujo de mensajes se deben cumplir las reglas especificadas en la normativa de protección de datos personales según la ley 18.331 [2].

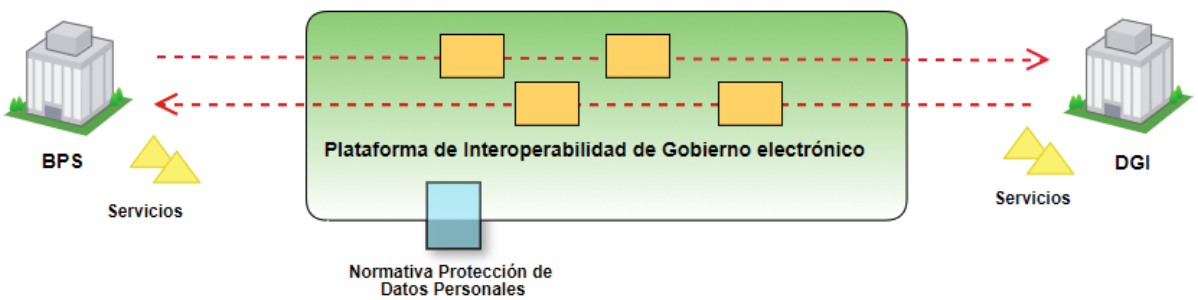

Figura 1 – Plataforma de interoperabilidad del Gobierno electrónico Uruguayo

En este contexto resulta de interés contar con mecanismos para monitorear y controlar esas normativas en plataformas de integración. Una de las principales problemáticas que se debe enfrentar es que existen distintos tipos de normativas (p. ej. Calidad de Servicio, Acuerdos de Nivel de Servicio) expresadas en diferentes lenguajes, incluso para un mismo tipo de normativa. Surge la necesidad entonces, de contar con una herramienta que permita gestionar los distintos tipos de normativas, especificar acciones a tomar en caso de que no se cumplan y expresar las mismas en un lenguaje común para que puedan ser desplegadas en una plataforma de integración. De esta forma, la plataforma puede monitorear y controlar el cumplimiento de distintos tipos de normativas sin la necesidad de conocer distintos lenguajes de especificación.

1.2. Objetivos

El objetivo general del proyecto es desarrollar una solución que permita gestionar normativas a ser controladas por una plataforma de integración interorganizacional.

Para cumplir el objetivo general del proyecto se plantean los siguientes objetivos específicos:

- Analizar los requerimientos que plantea la problemática, en particular, para normativas en las áreas de Calidad de Servicio, Acuerdos de Nivel de Servicio, Protección de Datos Personales e Intercambio de mensajes entre organizaciones.
- Proponer una solución que permita la gestión, especificación en un lenguaje común y visualización de normativas, así como la puesta en producción de las mismas en una plataforma de integración.
- Implementar un prototipo que de soporte a dicha solución para validar la factibilidad técnica de la propuesta.
- Desarrollar un caso de estudio para validar la aplicabilidad de la solución propuesta.

1.3. Aportes

A continuación, se describen los principales aportes del proyecto:

- Análisis de requerimientos en el contexto de la problemática planteada y de las normativas en las áreas mencionadas, así como de trabajos relacionados al área.
- Propuesta de solución que aborda la problemática planteada. La misma cuenta con un diseño extensible que permite importar normativas en distintos lenguajes y transformarlas a un lenguaje común para una plataforma de integración.
- Implementación de una herramienta utilizando la plataforma java empresarial a partir del diseño realizado que cumple con los requerimientos relevados.
- Desarrollo de un caso de estudio en el área *e-business* que permitió validar la aplicabilidad de la propuesta.

1.4. Organización del documento

El resto del documento se organiza de la siguiente manera:

- En el Capítulo 2 se presenta el marco conceptual, donde se describen los conceptos relevantes para el proyecto.
- En el Capítulo 3 se describe el análisis realizado que abarca el relevamiento de requerimientos y de los lenguajes de normativas, y los aportes de los trabajos relacionados encontrados.
- En el Capítulo 4 se presenta la solución propuesta. En particular, se detalla la arquitectura y se describen los aspectos más relevantes de la solución.
- En el Capítulo 5 se brindan detalles de la implementación del prototipo. En particular, se detallan las tecnologías y herramientas utilizadas.
- En el Capítulo 6 se desarrolla un caso de estudio mostrando los procedimientos realizados y los resultados obtenidos.
- En el Capítulo 7 se presentan las conclusiones del proyecto y se mencionan los trabajos a futuro como extensión del mismo.

2. Marco conceptual

En esta sección se presentan los conceptos más relevantes manejados en el proyecto: Gestión del Cumplimiento de Normativas, Estándares y Tecnologías XML, XACML y Tecnologías web y empresariales.

2.1. Gestión del Cumplimiento de Normativas

En entornos integrados y colaborativos, como son los de *e-government, e-health, e-science, e-commerce* y *e-business*, se debe cumplir con un conjunto de reglas a las que comúnmente se les denomina normativas. Estas pueden abarcar desde leyes, normas sectoriales, acuerdos de nivel de servicio, calidad de servicio en el intercambio de mensajes y estándares, entre otros [1].

2.1.1. Descripción general

Las normativas pueden surgir de distintas fuentes, y por lo general están especificadas en lenguaje natural. Para algunas de ellas, se han desarrollado formatos legibles por máquina.

La automatización de cumplimiento de normativas es beneficiosa para los organizaciones ya que por lo general, la frecuencia de las auditorías de cumplimiento, el monitoreo y los reportes, conduce a una mejor administración de cumplimiento. Para esto se requieren métodos y mecanismos de diferentes niveles de abstracción, como son: leyes, políticas y sistemas de software, entre otros [1].

Para el contexto de este trabajo, las categorías de normativas consideradas son: Calidad de Servicio, Acuerdo de Nivel de Servicio, Protección de Datos Personales e Intercambio de Mensajes entre Organizaciones.

2.1.2. Calidad de Servicio

La calidad de servicio (*Quality of Service*, QoS) describe y mide el rendimiento general de un servicio. En particular, los factores de calidad de un servicio web pueden incluir rendimiento, fiabilidad, disponibilidad y seguridad, entre otros. Como ejemplo se presentan algunos de estos factores de calidad de acuerdo a lo definido en [3].

El rendimiento de un servicio web representa la rapidez con que se puede completar una solicitud de servicio. Es posible medirlo en términos de *throughput*, tiempo de respuesta, latencia, entre otros. El *throughput* es el número de solicitudes atendidas del servicio web en un tiempo dado. El tiempo de respuesta es el tiempo requerido para completar una solicitud del servicio web. Por último, la latencia es el retardo de ida y vuelta entre enviar una solicitud y recibir su respuesta. En general, los servicios web de alta calidad deben proporcionar mayor rendimiento, mejor tiempo de respuesta y menor latencia [3].

La fiabilidad representa la capacidad de un servicio web para realizar sus funciones requeridas bajo condiciones establecidas durante un intervalo de tiempo especificado. La

medida general de un servicio web está relacionada con el número de fallos por día, semana, mes o año. La fiabilidad también está relacionada con la entrega asegurada y ordenada de los mensajes que se transmiten y reciben por los solicitantes de servicios y los proveedores de servicios [3].

Un servicio web debe estar listo y disponible para su consumo inmediato. La disponibilidad es la probabilidad de que el sistema esté en condiciones de realizar sus funciones, y está relacionado con la fiabilidad. El tiempo de reparación (TTR, *Time To Repair*) está asociado con la disponibilidad y representa el tiempo que se tarda en reparar el servicio web. El servicio debe estar disponible inmediatamente cuando se invoca [3].

Los servicios web deben contar con la seguridad requerida. Debido al aumento del uso de servicios web que se entregan a través de Internet pública, hay una preocupación creciente por la seguridad. Los proveedores de servicios web pueden aplicar diferentes enfoques y niveles de suministro de políticas de seguridad dependiendo del solicitante del servicio. Brindar seguridad a los servicios web significa proporcionar autenticación, autorización, confidencialidad, trazabilidad, cifrado de datos y no repudio [3].

A modo de ejemplo, una normativa en el área de QoS, podría especificar que un servicio tiene que tener un tiempo de respuesta entre 0.14 y 0.19 segundos.

2.1.3. Acuerdo de Nivel de Servicio

Un Acuerdo de Nivel de Servicio (*Service Level Agreement*, SLA) es un acuerdo negociado entre dos partes, donde una de ellas corresponde a un proveedor de servicios y la otra al cliente. Los Acuerdos de Nivel de Servicio definen un entendimiento común sobre servicios, prioridades, responsabilidades y garantías entre ambas partes. Por lo general un SLA incluye elementos como: definición de los servicios, medición del rendimiento, gestión de los problemas, deberes del cliente, garantías y finalización del acuerdo [4].

La supervisión y el cumplimiento del acuerdo de nivel de servicio son cada vez más importantes en un entorno en el que las aplicaciones y servicios empresariales se basan en servicios que pueden ser publicados de forma dinámica y bajo demanda [4].

A modo de ejemplo, una normativa en el área de SLA podría ser un acuerdo entre organización A y organización B que especifique que el tiempo de respuesta de un servicio de la organización A cuando es invocado por la organización B sea menor que 0.9 segundos por invocación.

2.1.4. Protección de datos personales

Los datos personales hacen referencia a cualquier tipo de información que identifique directamente o haga identificable a las personas. Entre ellos se pueden encontrar: nombre, dirección, teléfono, cédula de identidad, RUT, huella digital, número de socio, número de estudiante, una fotografía o hasta el ADN [5].

En Uruguay, la Ley Nº 18.331 de Protección de Datos Personales y Acción de *Habeas Data* reconoce el derecho a controlar el uso que se hace de los datos personales. La ley aplica a

datos personales registrados en cualquier sitio y que luego serán utilizados de diversas formas ya sea en ámbitos privados o públicos [5].

Un ejemplo es el manejo de datos personales en la Administración pública. Con ayuda de las Tecnologías de la Información, el gobierno electrónico almacena la información brindada por los organismos públicos para mejorar la gestión y los servicios ofrecidos a los ciudadanos. La transparencia de este manejo de información y la protección de los datos de las personas y organizaciones es fundamental para garantizar la seguridad y la confianza [5].

A modo de ejemplo, una normativa en esta área podría ser una ley que establece qué datos son públicos y qué datos son privados, o un consentimiento de un ciudadano para que una organización comparta alguno de sus datos con otra.

2.1.5. Intercambio de mensajes *Business to Business* (B2B)

B2B es un área encargada de mejorar las comunicaciones entre dos participantes, que por lo general son organizaciones. Se enfoca en la eficiencia de las interacciones de las organizaciones permitiendo nuevas formas de negocio e intercambio de información mediante servicios [6].

El desarrollo de modelos de negocio B2B se caracteriza por brindar rapidez y seguridad en las comunicaciones, facilidad de integración de procesos y de comunicación interna, y posibilidad de encontrar otras organizaciones con las que colaborar [6].

En la integración de procesos de negocio basados en tecnologías web se diferencian la coreografía y la orquestación de servicios web. Ambos conceptos se basan en describir las interacciones, cambios de estado y el flujo interno del proceso. La orquestación se enfoca en la vista de un participante en particular describiendo un control central del comportamiento (como un director de orquesta), mientras que la coreografía abarca todos los participantes proporcionando una vista total del sistema y describiendo un control distribuido del comportamiento [7].

Un ejemplo de normativa en esta área sería la especificación de una coreografía entre dos organizaciones. La normativa especifica las reglas de comunicación entre las organizaciones. En particular podría ser que la primera organización envía una petición solicitando datos personales de un ciudadano. La segunda organización puede responder con un mensaje de éxito o de fallo. En caso de fallo se reintenta la solicitud dos minutos después.

2.2. Estándares y Tecnologías XML

En esta sección se describen los principales estándares y tecnologías XML relevantes para el trabajo.

2.2.1. *Extensible Markup Language* (XML)

XML es un formato universal de texto muy flexible y simple derivado de *Standard Generalized Markup Language* (SGML), y es utilizado para datos y documentos estructurados. Al igual que otros lenguajes como *HyperText Markup Language* (HTML), XML utiliza distintas etiquetas para estructurar los datos de un documento [8].

XML sirve para estructurar, almacenar e intercambiar información. Los esquemas XML permiten saber si los datos contenidos en un documento XML son consistentes con los esperados [9].

2.2.2. XML Schema

Los esquemas XML expresan vocabularios compartidos y permiten a las máquinas llevar a cabo las reglas hechas por personas. Un esquema XML es un conjunto de reglas que proporciona un medio para definir la estructura, el contenido y la semántica de los documentos XML [10].

Un esquema permite especificar el formato correcto de los documentos XML. Para esto, el esquema define los elementos que pueden aparecer en el documento, y los atributos asociados a éstos. El esquema también define información estructural, como por ejemplo, enumerar los descendientes de un elemento, sus tipos, la secuencia en que aparecen, etc [11].

Uno de los cometidos principales que tienen los esquemas XML es la validación de un documento correspondiente a un esquema particular, para por ejemplo, verificar el correcto formato en el que se encuentran los datos [11].

Existen distintos estándares que se han propuesto para la especificación de esquemas XML. Entre ellos, se encuentran *Document Type Definition* (DTD) y *XML Schema Definition* (XSD). El XSD es el más utilizado debido a que se definen utilizando documentos XML, mientras que DTD no utiliza un formato XML [11].

La Figura 2 muestra un ejemplo de un esquema XML que puede utilizarse para registrar la fecha de cumpleaños de un cliente.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://ns.com.uy/schema/client" elementFormDefault="qualified"
    xmlns:clt="http://ns.com.uy/schema/client">

    <xs:element name="client" type="clt:Client"> </xs:element>
    <xs:complexType name="Client">
        <xs:sequence>
            <xs:element name="fullName" type="xs:string"/>
            <xs:element minOccurs="0" name="birthday" type="xs:dateTime"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Figura 2 – Ejemplo XML Schema

En la Figura 2 se observa la etiqueta *xsd:schema* del documento XML que permite definir esquemas de acuerdo al estándar W3C. Para poder diferenciar este esquema de otros, se define el atributo *targetNamespace* para asociar el esquema al espacio de nombres indicado. La etiqueta *xsd:complexType* permite definir tipos, indicando los elementos correspondientes a cada dato almacenado [11]. En este caso el Tipo definido es *Client* cuyos elementos son *fullName* (de tipo *string*) y *birthday* (del tipo *dateTime*)

La Figura 3 muestra un ejemplo de XML que cumple con el esquema definido en la Figura 2. El XML muestra el tipo definido *Client* y datos de *fullName* y *birthday* de ejemplos de un cliente.

```
<clt:client xmlns:clt="http://ns.com.uy/schema/client"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="ns.com.uy/schema/client">
    <clt:fullName>Roberto Perez Sanchez</clt:fullName>
    <clt:birthday>1978-09-19T11:51:55-03:00</clt:birthday>
</clt:client>
```

Figura 3 – Ejemplo XML que cumple con esquema de la Figura 2

2.2.3. *Scalable Vector Graphics* (SVG)

SVG es un lenguaje para describir gráficos bidimensionales en XML. SVG permite tres tipos de objetos gráficos: formas de gráficos vectoriales (por ejemplo, trayectorias que consisten en líneas rectas y curvas), imágenes y texto. Los objetos gráficos pueden ser agrupados, estilizados, transformados y compuestos en objetos previamente definidos. El conjunto de características incluye transformaciones anidadas y objetos de plantilla [12].

SVG dispone de un conjunto de etiquetas y atributos para introducir los elementos. La Figura 4 muestra el código de un ejemplo sencillo donde se dibuja un rectángulo con esquinas redondeadas [13].

```
<svg width=200 height=200>
   <rect x=10 y=40 rx=20 ry=20 width=180 height=120
      fill="orange" stroke="purple" stroke-width=2 />
<svg>
```

Figura 4 – SVG Ejemplo Rectángulo [13]

En la Figura 4 se puede observar el uso de la etiqueta *svg*. Este elemento es el contenedor, dentro del cual se incluye el código SVG que representa el resto de los elementos a dibujar [13].

En la Figura 5 se observa el resultado gráfico del SVG.

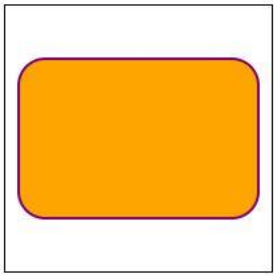

Figura 5 – Resultado ejemplo Rectángulo [13]

Otro ejemplo sencillo se puede observar en la Figura 6, donde se muestra una línea recta [13].

```
<svg height=200 width=200>
   <line x1=5 y1=5 x2=190 y2=190 stroke="red" stroke-width=2/>
</svg>
```

Figura 6 – SVG Ejemplo Línea [13]

En el ejemplo se muestran los atributos *stroke* y *stroke-width* que permiten definir el color y el grosor de la línea dibujada. Es posible dibujar elementos más complejos con etiquetas brindadas por SVG y además incluir estilos CSS para darle distintas propiedades a los elementos. Además se puede aplicar funciones JavaScript para darle movimientos y animaciones a los elementos creados con SVG [13].

En la Figura 7 se muestra el resultado gráfico del SVG anterior.

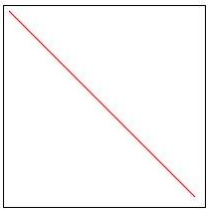

Figura 7 – Resultado ejemplo Línea [13]

2.2.4. Extensible Stylesheet Language Transformation (XSLT)

XSLT es un estándar de W3C para definir la transformación y presentación de documentos XML [14].

Una hoja de estilo XSLT (Transformaciones XSL) especifica la presentación de una clase de documentos XML describiendo cómo se transforma una instancia de la clase. Como resultados se pueden obtener otros archivos XML, archivos HTML, archivos expresados en Extensible Acces Control Markup Language (XACML), etc [15].

La Figura 8 muestra una transformación partiendo de un archivo XML. En el ejemplo el archivo XML contiene información relacionada a un catálogo de discos de música con sus

títulos y artistas representados con etiquetas propias del ejemplo. Al aplicar la transformación que se visualiza en la Figura 8 correspondiente a la hoja de estilo *Extensible Stylesheet Language* (XSL), se obtiene como salida un archivo HTML.

La hoja de estilo XSL consta de una etiqueta: *<xsl:template match="/">*. Esta etiqueta define una forma de reutilizar plantillas con el fin de generar la salida deseada para los nodos de un tipo particular. Se indica con el *"match"* el patrón en el que se va a aplicar la plantilla [16].

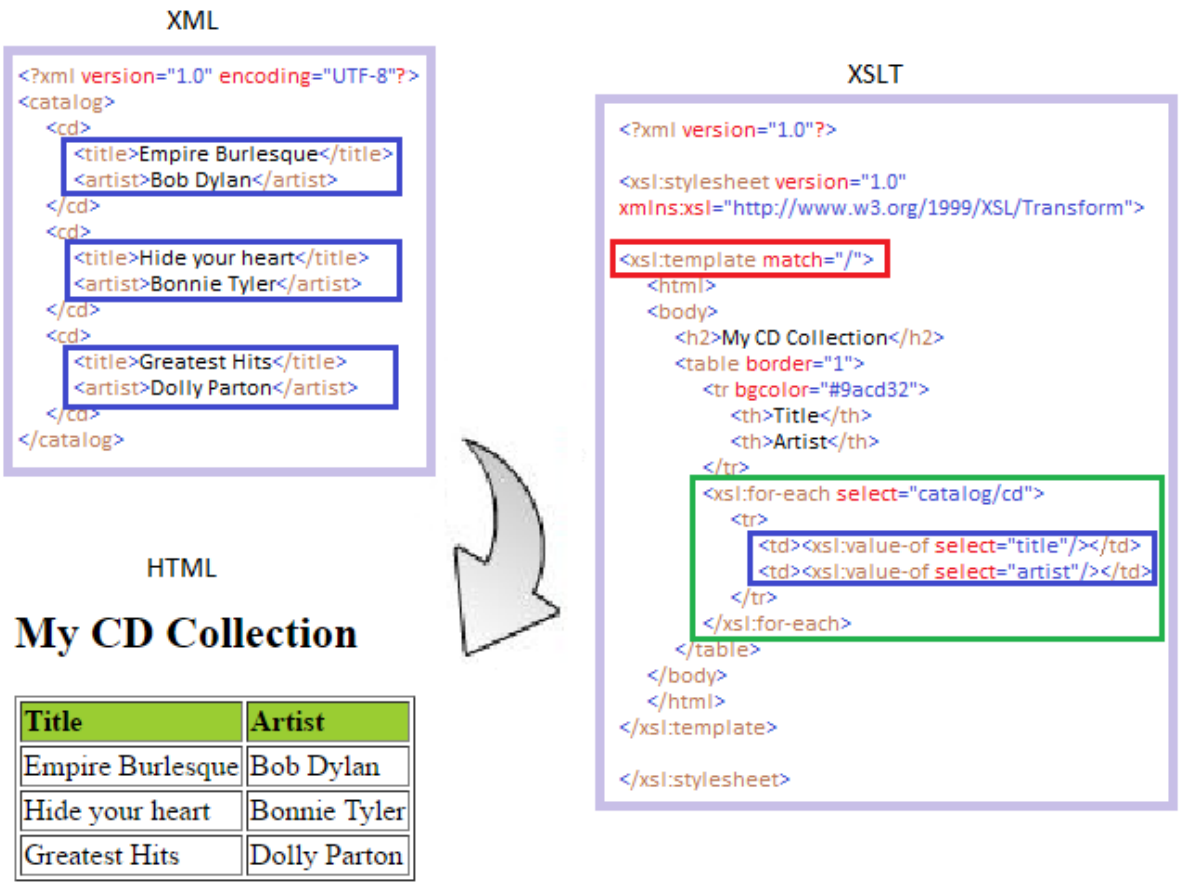

Figura 8 – Ejemplo Transformación XSLT [16]

En el ejemplo de la Figura 8 se va a sustituir por todo el documento y se coloca seleccionando el nodo principal. Luego se comienza a armar el documento HTML y la tabla correspondiente iterando con la etiqueta *<xsl:for-each>.* La iteración es realizada sobre todos los discos que en el HTML se visualizan representados con la etiqueta *<cd>* dentro del nodo padre: *<catalogo>*. Finalmente, con la etiqueta *<xsl:value-of>* se obtienen los valores de los elementos *"title"* y *"artist"* del archivo HTML y se colocan en las etiquetas *<td>* del resultado. El archivo HTML de salida muestra una tabla con la información de los títulos y artistas de los discos antes mencionados.

2.2.5. Base de datos XML

Las bases de datos XML se utilizan para guardar información en formato XML. Existen dos tipos diferentes de Bases de datos XML: *enable* y *native*. La primera es una base de datos

relacional, donde la información XML se almacena en tablas que constan de filas y columnas. La segunda, se basa en formato contenedor, no de tabla, y es más utilizada cuando se desea almacenar gran cantidad de información XML [17].

En particular, eXist-db [1] es una base de datos de documentos NoSQL basada en tecnologías XML, que da soporte a XML, JSON, HTML y documentos binarios. Todas las versiones de eXist-db son de código abierto y pueden utilizarse en aplicaciones académicas, no comerciales y comerciales [18]. Existen otras bases de datos XML, entre ellas, Oracle XML DB. Oracle XML DB es una tecnología de almacenamiento y recuperación nativa XML, y proporciona soporte para otros estándares, entre los que se encuentran: Namespaces, DOM, XQuery, SQL y XSLT. La versión gratuita de Oralce XML trabaja con la versión 1.0 de XQuery, mientras que eXist-db lo hace con la versión 3.0, y por lo tanto la base de datos eXist es más rica en procesamiento [19].

2.3. *Extensible Access Control Markup Language* (XACML)

XACML es un estándar definido por OASIS[2] y describe un lenguaje en el que se declaran políticas de control de acceso (expresado en XML) y un modelo de cómo procesar y evaluar peticiones de acceso según reglas definidas en las políticas. El resultado de dicho proceso depende no sólo de las reglas definidas en las políticas, sino también de la información que contienen las peticiones [20].

El principal objetivo de este lenguaje es promover la interoperabilidad entre los controles de acceso establecidos por las diferentes organizaciones [20].

XACML se basa principalmente en un sistema que se centra en los atributos de los solicitantes, el recurso solicitado, y las acciones sobre el recurso y su entorno. Este sistema se llama *Attribute Based Access Control* (ABAC). Para poder obtener una determinada autorización, los atributos deben cumplir con las condiciones definidas en las reglas [20].

2.3.1. Arquitectura de XACML

La arquitectura se compone de cuatro bloques de construcción: *Policy Decision Point* (PDP), *Policy Enforcement Point* (PEP), *Policy Information Point* (PIP) y *Policy Administration Point* (PAP).

La Figura 9 muestra cómo los bloques se comunican entre sí y cada uno cumple con funciones distintas. PDP evalúa las políticas contra las solicitudes de acceso proporcionados por el PEP. Para proporcionar las decisiones, PDP también puede tener que consultar un PIP para reunir atributos descriptivos sobre el usuario o cualquier otro atributo que falta en la solicitud [21].

[1] http://exist-db.org/exist/apps/homepage/index.html
[2] https://www.oasis-open.org/

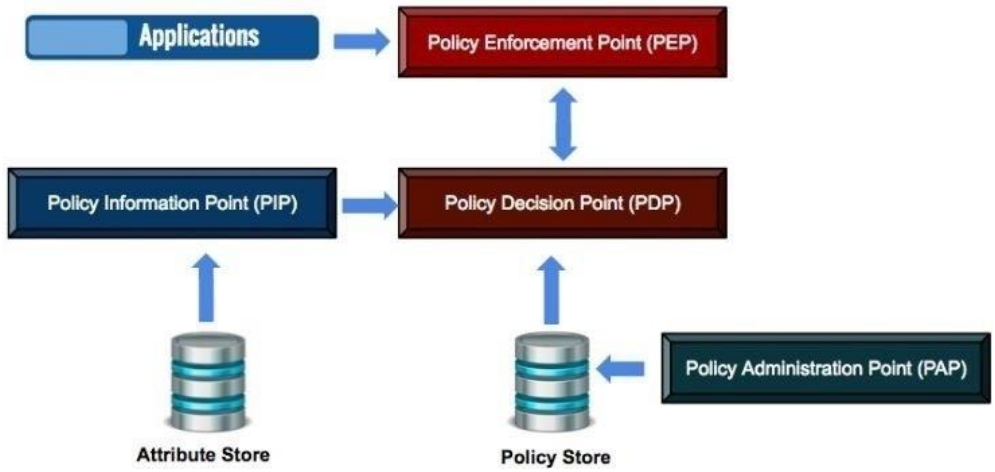

Figura 9 – Arquitectura XACML [21]

2.3.2. Estructura para una política de control de acceso

En la Figura 10 se muestra el modelo conceptual de XACML como política de control de acceso, donde los conceptos principales son *PolicySet, Policy y Rule*. Se puede definir un conjunto de políticas (*PolicySet*) o una única política (*Policy*) en un mismo documento. Estas definiciones pueden contener a su vez un *PolicyCombiningAlgorithm* y un *RuleCombiningAlgorithm*. Los algoritmos de combinación de políticas son procedimientos que combinan múltiples decisiones y obligaciones que provienen de distintas políticas y los algoritmos de Combinación de reglas son procedimientos que combinan múltiples decisiones que provienen de distintas reglas [20].

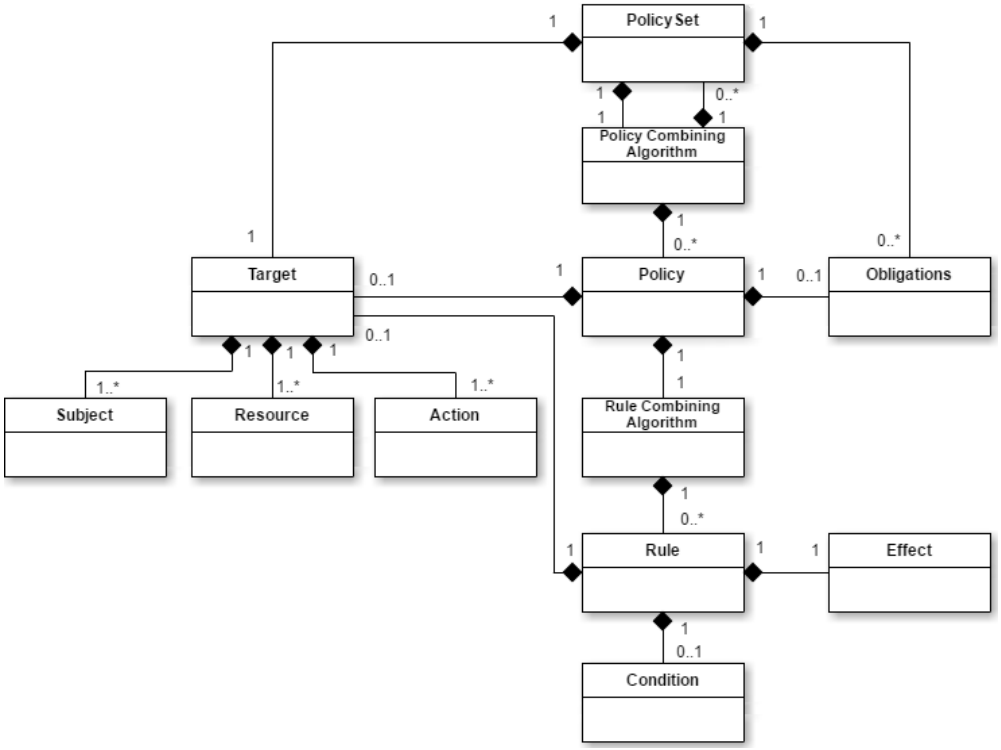

Figura 10 – Modelo de Políticas [20]

Además, tanto los conjuntos de políticas como las políticas pueden poseer *Obligations*. Las obligaciones son operaciones a cumplir cuando un usuario recibe una respuesta de autorización o denegación en una petición de autorización [20].

También se observa en la Figura 10 que las políticas y las reglas pueden poseer un *Target*, que define las condiciones que debe cumplir un solicitante (*Subject*), el recurso solicitado (*Resource*), la acción a realizarse sobre el recurso (*Action*) con la petición de autorización y el entorno (*Environment*) para que esa política o regla sea aplicable [20].

Además, las reglas pueden contener una *Condition* y deben contener un *Effect*. Las *Conditions* son funciones que permiten refinar la aplicabilidad de la regla establecida en el *Target* mencionado. Y el *Effect* corresponde al cumplimiento de una regla, donde su valor establece la autorización (*Permit*) o la denegación (*Deny*) del acceso [20].

2.3.3. Estructura XACML

En esta sección se presentan algunos ejemplos sencillos sobre el uso de reglas y políticas utilizando el lenguaje XACML para describirlas.

La Figura 11 muestra un ejemplo de una regla, donde se especifica que cualquier persona con el rol de "*developer*" puede ejecutar cualquier acción a cualquier recurso. En la Figura 11 se muestra el atributo en el *SubjectAttributeDesignator* y el valor correspondiente en el *AttributeValue*.

```
<Rule RuleId="rul-0001" Effect="Permit">
    <Description>
        Some optional text that explains the purpose of the rule
    </Description>
    <Target>
        <Subjects>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                        developer
                    </AttributeValue>
                    <SubjectAttributeDesignator>
                        role
                    </SubjectAttributeDesignator>
                </SubjectMatch>
            </Subject>
        </Subjects>
    </Target>
</Rule>
```

Figura 11 – Ejemplo de Rule en XACML [22]

Una regla puede contener también una *condition* que debe cumplirse para que la regla pueda devolver el *effect*. Si la condición devuelve indeterminado, la regla también devuelve indeterminado. Si la condición devuelve falso, la regla devuelve *NotApplicable*. Si la condición devuelve verdadera, entonces se devuelve el valor del elemento *effect*, que puede ser: Permitir o Denegar. Si la condición no se especifica, al igual que en el ejemplo anterior, se asume que es verdadera [22].

Las reglas pueden ser evaluadas por separado, pero deben ser parte de una política. Las reglas son la unidad más pequeña en XACML, mientras que las políticas son la unidad más pequeña de la evaluación [22].

La Figura 12 muestra un ejemplo de *Policy*. En el ejemplo se observa que no se especifica un *target* en la regla, pero se podría hacer. En ese caso, la regla sólo se evaluaría por la política si su *target* evalúa verdadero en sus condiciones [22].

En la Figura 12 se muestra la utilización de la regla definida anteriormente, referenciándola con el *RuleId* "rul-0001".

```
<Policy PolicyId="pol-0001" RuleCombiningAlgId=
"urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
    <Description>
        Some optional text that explains the purpose of the policy
    </Description>
    <Target>
        <Subjects>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                        developer
                    </AttributeValue>
                    <SubjectAttributeDesignator>
                        role
                    </SubjectAttributeDesignator>
                </SubjectMatch>
            </Subject>
        </Subjects>
    </Target>
    <Rule RuleId="rul-0001" Effect="Permit"/>
</Policy>
```

Figura 12 – Ejemplo de Policy en XACML [22]

La Figura 13 presenta un ejemplo de *PolicySet*. Al igual que las *reglas* pueden ser reutilizadas en políticas, las políticas enteras pueden ser reutilizadas en otros conjuntos de políticas. Un *PolicySet* contiene un *target*, un *Policy-Combining Algorithm*, un conjunto de *policies*, y algunas *obligations*. El algoritmo de políticas de combinación especifica el procedimiento por el que se combinan los resultados de la evaluación de las políticas [22].

En la Figura 13 se muestra la utilización de la *policy* definida anteriormente, y se referencia con el *PolicyId* "pol-0001". En la Figura 13 también se observa la política referenciada combinada con la regla correspondiente.

```
<PolicySet PolicySetId="pls-0001" PolicyCombiningAlgId=
"urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides">
    <Description>
        Some optional text that explains the purpose of the policy set
    </Description>
    <Target>
        <Subjects>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                        developer
                    </AttributeValue>
                    <SubjectAttributeDesignator>
                        role
                    </SubjectAttributeDesignator>
                </SubjectMatch>
            </Subject>
        </Subjects>
    </Target>
    <PolicyIdReference>
        pol-0001
    </PolicyIdReference>
    <Policy PolicyId="pol-0001" RuleCombiningAlgId=
    "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
        <Target/>
        <Rule RuleId="rul-0001" Effect="Permit"/>
    </Policy>
</PolicySet>
```

Figura 13 – Ejemplo de PolicySet en XACML [22]

2.4. Tecnologías web y empresariales

A continuación se describen tecnologías web y empresariales relevantes para el desarrollo del proyecto.

2.4.1. *Java Platform, Enterprise Edition* (Java EE)

Java EE es una plataforma de programación que permite desarrollar y ejecutar aplicaciones en el lenguaje de programación Java. Java EE se desarrolla a través del *Java Comunity Process* (JCP) que es responsable de todas las tecnologías Java. Grupos de expertos compuestos de partes interesadas han creado solicitudes de especificación de Java (*Java Specification Requests*, JSR) para definir las diversas tecnologías Java EE. El trabajo de la Comunidad Java bajo el programa JCP ayuda a asegurar los estándares de estabilidad de la tecnología Java y la compatibilidad entre plataformas [23].

La plataforma Java EE utiliza un modelo de programación simplificado. Los descriptores de despliegue XML son opcionales. En su lugar, un desarrollador puede simplemente introducir la información como una anotación directamente en un archivo de origen de Java, y el servidor de Java EE configurará el componente en implementación y tiempo de ejecución. Estas anotaciones se utilizan para integrar en un programa datos que serían proporcionados en un descriptor de despliegue XML. Con la ayuda de las anotaciones, se pone la información de especificación en el código junto al elemento del programa que se ve afectado [23].

En la plataforma Java EE, la inyección de dependencia se puede aplicar a todos los recursos que necesita un componente, ocultando de manera efectiva la creación y

búsqueda de recursos del código de la aplicación. La inyección de dependencias se puede utilizar en contenedores de *Enterprise JavaBeans* (EJB), contenedores web y clientes de aplicaciones. La inyección de dependencia permite al contenedor Java EE insertar automáticamente referencias a otros componentes o recursos necesarios, utilizando anotaciones [23].

2.4.2. Web Services

Los servicios web son aplicaciones capaces de interoperar en la web que intercambian información con el objetivo de ofrecer servicios. Están básicamente diseñados para soportar interoperabilidad entre aplicaciones de software. Esta interoperabilidad se obtiene a través de un conjunto de estándares abiertos basados en XML, como *Simple Object Access Protocol* (SOAP), *Web Services Description Language* (WSDL), entre otros. Estas normas proporcionan un enfoque común para definir, publicar y utilizar servicios web [24].

SOAP es un protocolo para intercambiar información con una determinada estructura en un entorno descentralizado y distribuido. Utiliza tecnologías XML para definir un *framework* de mensajería extensible capaz de proporcionar la construcción de mensajes que pueden ser luego intercambiados. El *framework* fue diseñado para que sea independiente de cualquier lenguaje de programación [25].

WSDL es un formato de XML utilizado para describir servicios web. WSDL define una gramática XML para describir los servicios de red como colecciones de *endpoints* de comunicación capaces de intercambiar mensajes. Las definiciones de servicio WSDL proporcionan documentación para sistemas distribuidos y sirven como guía para automatizar los detalles involucrados en la comunicación de aplicaciones [26].

2.4.3. HTML5

HTML es el lenguaje estándar utilizado para crear páginas web y sus elementos forman los bloques de construcción de todos los sitios web [27]. HTML5 trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos que posee un conjunto más amplio de tecnologías permitiendo a sitios web y a las aplicaciones, mayor diversidad y alcance [28].

2.4.4. JavaScript

Un *Script* es un código de programa que no necesita preprocesamiento (por ejemplo, compilación) antes de ejecutarse. En el contexto de un navegador Web, el *scripting* se refiere generalmente al código del programa escrito en JavaScript que es ejecutado por el navegador cuando se descarga una página o en respuesta a un evento desencadenado por el usuario [29].

Las secuencias de comandos pueden hacer que las páginas web sean más dinámicas. Por ejemplo, sin recargar una nueva versión de una página, puede permitir modificaciones en el contenido (*Dynamic* HTML) o permitir que el contenido se agregue o se envíe desde esa página (*Asynchronous JavaScript and XML*, AJAX) [29].

Los *scripts* permiten cada vez más a los desarrolladores crear un puente entre el navegador y la plataforma en la que se está ejecutando, por ejemplo crear páginas Web que incorporen información del entorno del usuario, como ubicación actual, entre otros. Esta interactividad adicional hace que las páginas Web se comporten como una aplicación de software tradicional [29].

3. Análisis

En esta sección se detalla el relevamiento de requerimientos, relevamiento de lenguajes de normativas y trabajos relacionados.

3.1. Relevamiento de requerimientos

El objetivo de esta sección es comprender la realidad y especificar los requerimientos. Para esto se define un marco de trabajo, se describe el problema a resolver, se elabora un modelo conceptual, se determinan los requerimientos funcionales y no funcionales y se definen los casos de uso.

3.1.1. Marco de trabajo

El marco de trabajo consiste en organizaciones que colaboran a través de una plataforma de integración. En este contexto, las organizaciones colaboran publicando y consumiendo servicios.

En particular, la Figura 14 muestra el intercambio de mensajes entre el BPS y la DGI con información de ciudadanos uruguayos a través de la Plataforma de Interoperabilidad de Gobierno Electrónico.

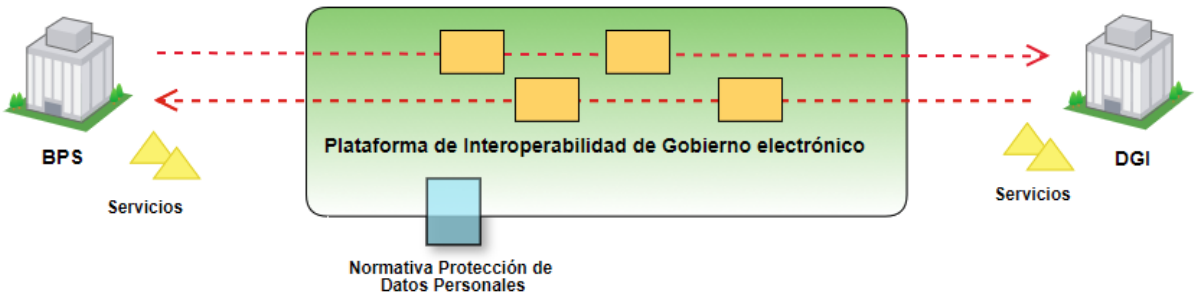

Figura 14 – Marco de Trabajo

En estos contextos colaborativos se deben cumplir con un conjunto importante de normativas que en particular aplican a los intercambios de mensajes entre las organizaciones. Por ejemplo en la Figura 14 los intercambios tienen que cumplir con las normativas de protección de datos personales.

3.1.2. Descripción del problema

En el contexto mencionado en la sección 3.1.1 resulta de interés contar con mecanismos para monitorear y controlar esas normativas en plataformas de integración. Existe la necesidad de administrar y gestionar las normativas referidas a las diferentes áreas. Pero también, surge la necesidad de poder comprender y manejar los diferentes lenguajes de especificación de normativas.

Una de las principales problemáticas que se deben enfrentar es que existen distintos tipos de normativas expresadas en diferentes lenguajes: por ejemplo L1, L2 y L3 como se muestra en la Figura 15, incluso para una misma categoría. Además, las normativas deben poder expresarse en un lenguaje común L para que sean puestas en producción en una plataforma de integración, con el fin de monitorear y controlar su cumplimiento.

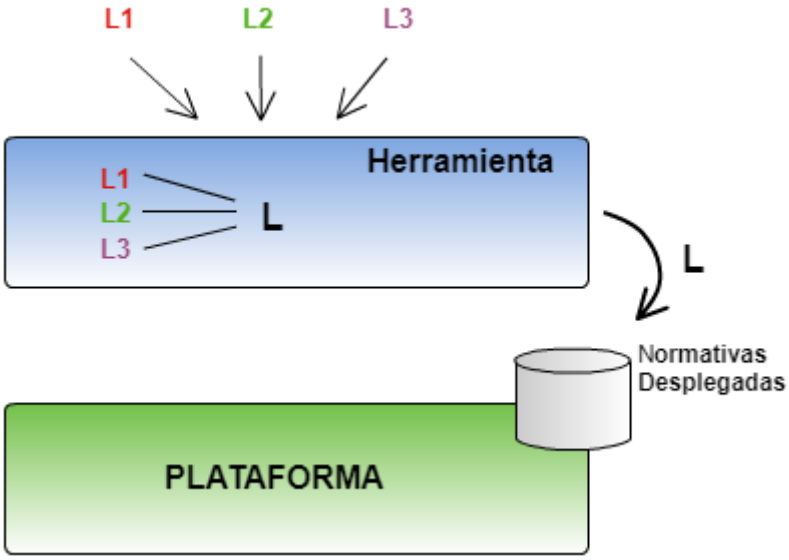

Figura 15 – Descripción del problema

Surge la necesidad entonces, de contar con una herramienta que permita gestionar los distintos tipos de normativas. El problema a resolver es la construcción de una herramienta capaz de representar distintos tipos de normativas en ese lenguaje de puesta en producción, permitiendo a organizaciones que interactúan a través de una plataforma de integración, gestionar normativas independientemente de su lenguaje de especificación.

Las normativas manejan información compleja y tratan con datos críticos para las organizaciones. Por este motivo se debe poder restringir la manipulación, la gestión y la puesta en funcionamiento de las normativas de una organización con respecto a los usuarios de la herramienta.

3.1.3. Modelo conceptual

De la realidad planteada en la descripción del problema surge el modelo conceptual que se presenta en la Figura 16.

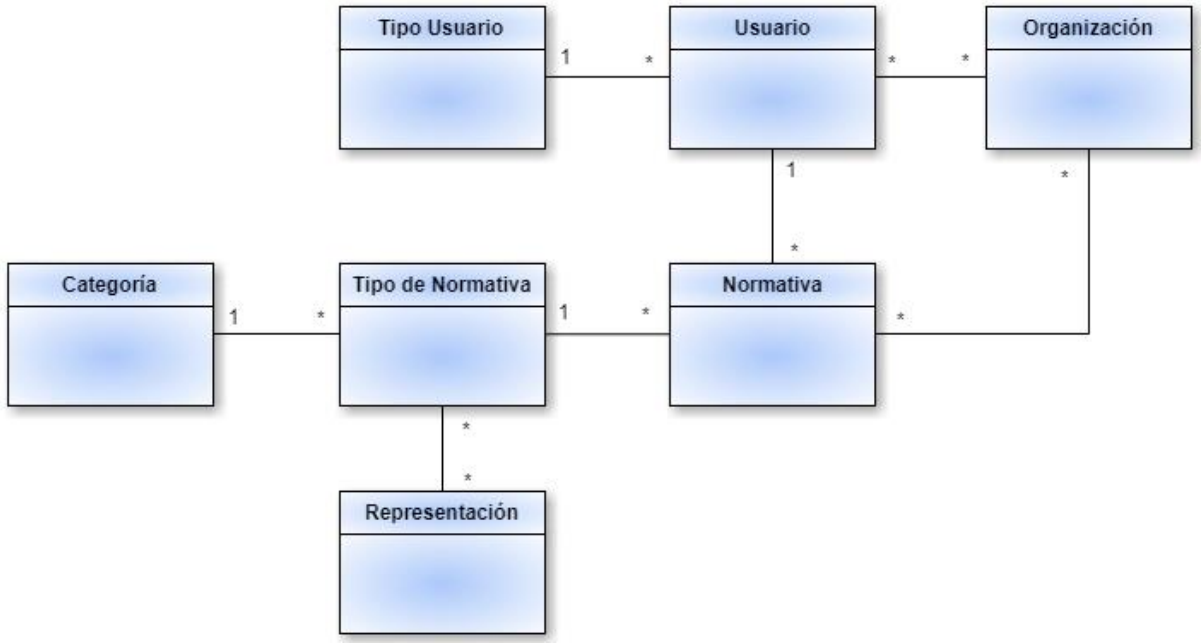

Figura 16 – Modelo Conceptual

A continuación se describen los elementos del modelo:

- **Categoría:** es una forma de clasificar las áreas de normativas trabajadas en el proyecto. Lo que define una categoría es la cantidad de organizaciones que participan y el conjunto de tipos de normativas que agrupa. Ejemplos de categorías son: QoS, SLA, intercambio de mensajes y Protección de Datos Personales.

- **Tipo de Normativa:** es lo que define la sintaxis y estructura con el que se deben validar las normativas especificadas en un determinado lenguaje. Existen varios tipos de normativas para cada Categoría. Por ejemplo para la Categoría intercambio de mensajes, tenemos los siguientes tipos de normativas: *Business Process Model and Notation* (BPMN) y *Web Service Choreography Description Language* (WS-CDL).

- **Normativa:** corresponde a la especificación de las reglas en un determinado lenguaje que es dado por el tipo de normativa. Las normativas son gestionadas por los usuarios e involucran a una cantidad fija de organizaciones definidas en el tipo de normativa.

- **Representación:** hace referencia a las posibles representaciones de la normativa en distintos lenguajes. La representación se define dinámicamente al momento de crear los tipos de normativa. Al definir el tipo de normativa se determinan las representaciones posibles que soportan las normativas correspondientes a dicho tipo. Ejemplos de representaciones pueden ser XACML y HTML.

- **Usuario:** corresponde a los usuarios que utilizan la herramienta. Cada usuario tiene asociado un Tipo de Usuario.

- **Tipo de Usuario:** hace referencia a los diferentes permisos que tienen los usuarios sobre las acciones del sistem. Por ejemplo: un usuario técnico tiene permisos para dar de alta representaciones en el sistema. Otros ejemplos de tipo de usuario son: usuario común, administrador de organización, y administrador del sistema.

- **Organización:** hace referencia a las organizaciones que interactúan con la plataforma de integración. Los usuarios pertenecen a una organización. Las organizaciones están involucradas en las normativas. Por ejemplo: puede existir una normativa sobre acuerdo de nivel de servicio entre la DGI y el BPS y usuarios de estas organizaciones acceder a la herramienta para visualizarla y/o aprobarla.

La Figura 17 muestra un ejemplo de instanciación del modelo presentando relaciones y ejemplos de Tipos de Normativa, Categorías y Representaciones.

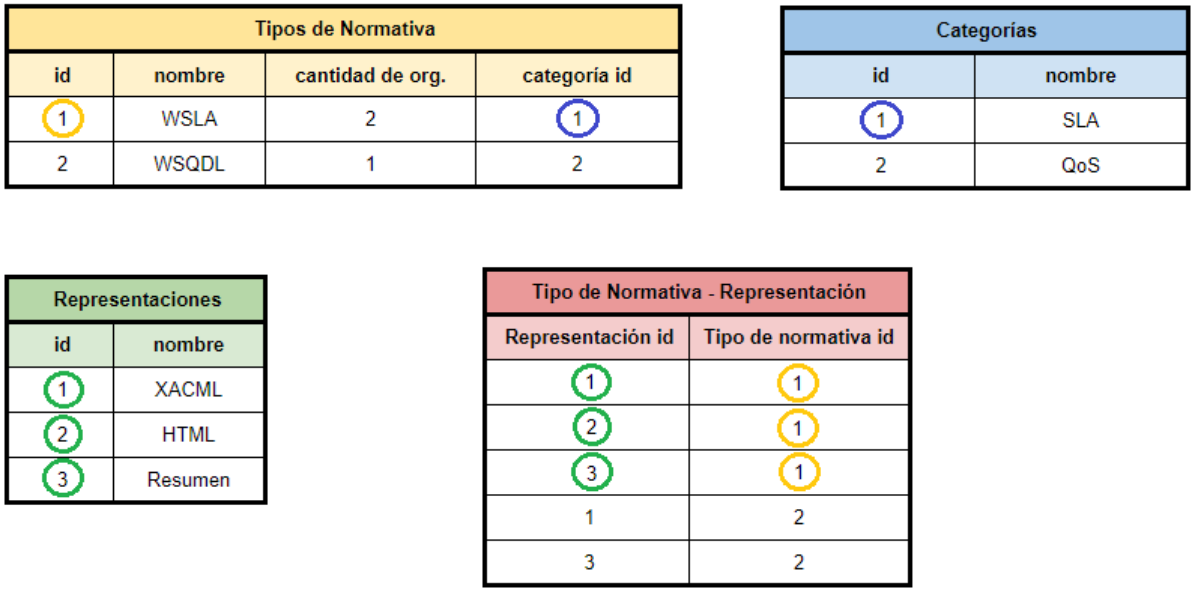

Figura 17 – Ejemplo instancia Modelo Conceptual

En la Figura 17 se muestra como ejemplo dos tipos de normativa: WSLA y WSQDL corrrespondientes a las categorías SLA y QoS respectivamente. Se observa además tres representaciones: XACML, HTML y Resumen. Las tres representaciones están asociadas al tipo de normativa WSLA.

3.1.4. Requerimientos

En base al marco de trabajo definido, la descripción del problema y el modelo conceptual se detallan los requerimientos identificados.

3.1.4.1. Requerimientos funcionales

- Gestión de Organizaciones

El sistema debe permitir la creación, modificación y eliminación de organizaciones. La gestión se debe realizar únicamente por un usuario con permisos de administración.

- Gestión de Categorías de Tipos de Normativas

El sistema debe permitir la creación, modificación y eliminación de categorías. La gestión se debe realizar únicamente por un usuario con permisos de administración.

- Gestión de Tipos de Normativas

El sistema debe permitir gestionar tipos de normativas por un usuario administrador y/o Técnico. El tipo de normativa involucra a una cantidad de organizaciones fija. Cada Tipo de Normativa pertenece a una única categoría.

- Gestión de Normativas

El sistema debe permitir gestionar normativas por usuarios. Éstas corresponden a algún tipo de normativa existente.

- Registro de usuario

El sistema debe contar con la posibilidad de registrar nuevos usuarios.

- Inicio y cierre de sesión

El sistema debe permitir el inicio y cierre de sesión de los usuarios.

- Administración de usuarios

El sistema debe permitir el manejo de diferentes tipos de usuarios con posibilidades de poder efectuar diferentes acciones y con diferentes permisos.

- Representación de normativas

El sistema debe permitir que a partir de normativas especificadas en un lenguaje en particular puedan ser representadas en otros lenguajes. En particular, se debe tener una manera de representar todas las normativas en el lenguaje de puesta en producción.

- Visualización de normativas

El sistema debe permitir la posibilidad de visualizar las normativas gráficamente además de poder verlas en su lenguaje de especificación.

- Aprobación de normativas

El sistema debe permitir un flujo de aprobación de normativas. Los usuarios de las organizaciones pueden crear normativas, pero éstas no quedarán aprobadas automáticamente. Solo usuarios con asignación de determinados permisos podrán aprobar la normativa en la que su Organización está involucrada

- Puesta en producción de las normativas

El sistema debe permitir poner en producción las normativas. El administrador del sistema puede poner en producción las normativas una vez que estén aprobadas por todas las partes involucradas.

3.1.4.2. Requerimientos no funcionales

- Implementación del sistema con la utilización de la plataforma java empresarial (Java EE).
- Extensibilidad: Se debe poder extender la herramienta para soportar nuevas representaciones. Debe ser extensible tanto en los tipos de normativas así como en los lenguajes en que pueden ser especificadas.

3.1.5. Casos de uso

En esta sección se presentan los casos de uso, los cuales se pueden ver con más detalle en el Apéndice 1.

En base a los requerimientos previamente relevados, se identifican los siguientes tipos de usuarios: usuario común, administrador de organización, técnico y súper administrador.

- **Usuario común:** es aquel usuario que se registra en la aplicación y hace uso de las funcionalidades del sistema.
- **Administrador de organización:** es aquel usuario con privilegios de permisos sobre una organización existente.
- **Técnico:** es aquel usuario con permisos enfocados a funcionalidades técnicas.
- **Súper administrador:** es un Súper Usuario, capaz de administrar los datos de cada uno de los diferentes usuarios del sistema.

La Figura 18 muestra el modelo de casos de uso definido a partir de los requerimientos previamente identificados.

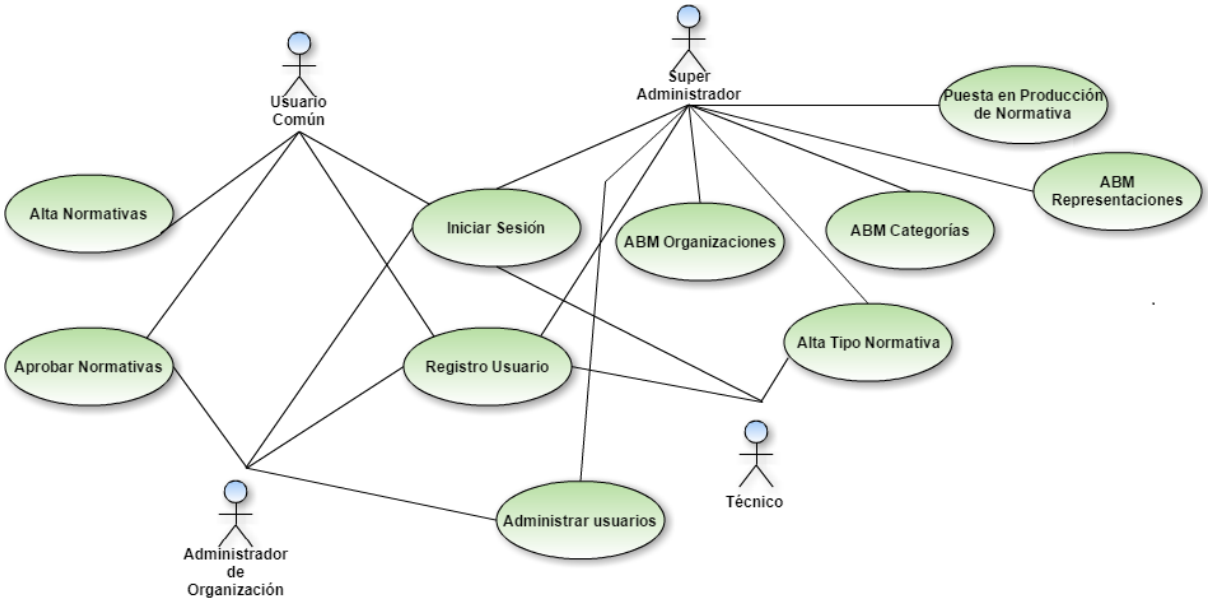

Figura 18 – Vista de Casos de uso

3.1.5.1. Alta, Baja y Modificación de organización

El caso de uso comienza cuando el administrador desea crear, borrar o modificar una organización. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite estas acciones. Para ellos, se debe ingresar únicamente el nombre de la organización.

3.1.5.2. Alta, Baja y Modificación de categoría

El caso de uso comienza cuando el administrador desea crear, borrar o modificar una categoría de tipos de normativas. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite estas acciones. Para ellos, se debe ingresar únicamente el nombre de la categoría.

3.1.5.3. Alta, Baja y Modificación de representación

El caso de uso comienza cuando el administrador desea crear, borrar o modificar una Representación. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite estas acciones. Para ello, se debe ingresar únicamente el nombre de la Representación.

3.1.5.4. Alta tipo de normativa

El caso de uso comienza cuando el administrador desea crear un nuevo tipo de normativa. Se definen la cantidad de organizaciones involucradas y las representaciones que soportará el tipo de normativa. A tales efectos, el sistema proporciona una interfaz gráfica acorde para dicha acción.

3.1.5.5. Alta normativa

El caso de uso comienza cuando el usuario quiere dar de alta una Normativa. Crear la normativa incluye efectuar las transformaciones correspondientes a su tipo de normativa. A tales efectos, el sistema proporciona una interfaz gráfica acorde para dicha acción.

3.1.5.6. Aprobar normativa

El caso de uso comienza cuando el usuario desea aprobar una normativa. A través de interacciones con el sistema, los usuarios con determinados permisos aprueban una normativa.

3.1.5.7. Administrar usuarios

El caso de uso comienza cuando el administrador desea administrar un usuario cambiando su tipo de usuario y modificando su nivel de permisos. A tales efectos, el sistema proporciona una interfaz gráfica acorde para dicha acción.

3.1.5.8. Iniciar sesión

El caso de uso comienza cuando el usuario ingresa su nombre de usuario y contraseña para acceder a la aplicación.

3.1.5.9. Registro usuario

El caso de uso comienza cuando el usuario desea registrarse en el sistema.

3.2. Lenguajes para la Especificación de Normativas

En esta sección, se analizan lenguajes para la especificación de normativas en las categorías que apunta a soportar la herramienta con el fin de seleccionar el más adecuado para ser utilizado en el contexto del proyecto. Con este fin, se preseleccionaron en primera instancia lenguajes con alto grado de utilización en la industria y que contaran con buena documentación.

En segunda instancia se seleccionaron los lenguajes a utilizar en el proyecto de acuerdo a los siguientes criterios:

- **Sintaxis XML:** Dado que en el marco del proyecto es necesario transformar las normativas a un lenguaje común o varios lenguajes, se entiende que es importante que el lenguaje seleccionado cuente con una sintaxis en XML.

- **Correspondencia con elementos del contexto de trabajo:** La especificación debe permitir especificar una normativa, que pueda ser expresada en base a los elementos del contexto de trabajo de la forma más directa posible. Es decir, que sean identificables fácilmente los servicios que se están manejando, los actores participantes en el envío de mensajes, y los datos más relevantes.

Cabe recalcar que la herramienta no queda limitada a la selección de lenguajes realizada. Es decir, la herramienta es extensible brindando soporte a la representación de las normativas en múltiples lenguajes de especificación

3.2.1. Intercambio de mensajes

Existen varios lenguajes para especificar intercambios de mensajes entre organizaciones, como por ejemplo BPMN (*Business Process Model and Notation*), WS-CDL (*Web Service Choreography Description Language*) y WS-BPEL (Web Services Business Process Execution Language). En base a la preselección se analizan dos lenguajes para especificar reglas de intercambio de información en negocio electrónico para B2B: WS-CDL y BPMN. A pesar de que WS-BPEL es muy utilizado en la industria, está más orientado a la especificación de orquestaciones.

3.2.1.1. *Web Service Choreography Description Language* (WS-CDL)

WS-CDL es un lenguaje basado en XML que describe la colaboración entre organizaciones, desde un punto de vista global, de los comportamientos comunes y observables de cada participante de un proceso de negocio. Los Servicios Web ofrecen un puente de comunicación entre entornos computacionales utilizados para desarrollar las aplicaciones. La especificación de coreografía de Servicios Web está diseñada para componer colaboraciones interoperables entre organizaciones, independientemente de la plataforma de soporte o del modelo de programación utilizado [30].

- **Coreografías WS-CDL**

La coreografía ofrece un medio por el cual las reglas de participación dentro de una colaboración pueden ser claramente definidas y acordadas conjuntamente. Cada entidad puede implementar su parte de la coreografía según lo determinado por la visión global.

La Figura 19 muestra un esquema de un ejemplo de uso de WS-CDL donde la DNIC y el BPS deciden integrar sus aplicaciones basadas en Servicios Web. Los respectivos analistas de negocio de ambas organizaciones acuerdan los servicios involucrados en la colaboración, sus interacciones y sus reglas comunes de ordenación y restricción bajo las cuales se producen las interacciones. Luego generan una representación basada en WS-CDL. Es así como una coreografía define las interacciones de servicios entre las entidades organizacionales que garantizan la interoperabilidad, dejando las decisiones reales de implementación en manos de cada organización [30].

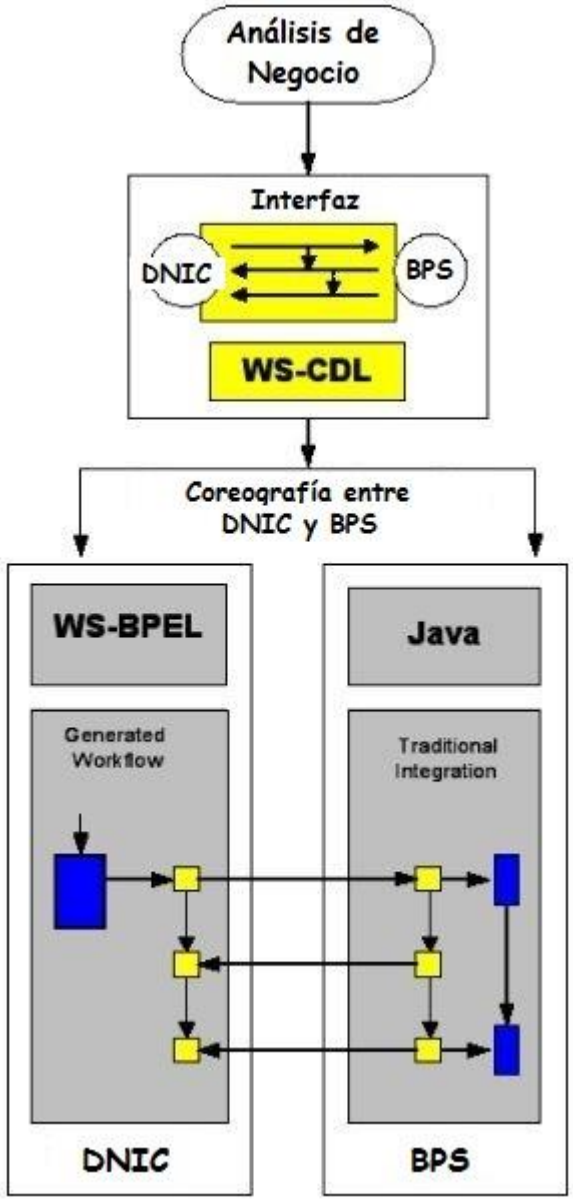

Figura 19 – Coreografía WS-CDL [30]

En la Figura 19 se muestra que la DNIC se basa en una solución de Lenguajes de Ejecución de Procesos de Negocio con Servicios Web (WS-BPEL) para implementar su parte de la coreografía. Mientras que el BPS basa su solución en Java Empresarial para implementar su parte de la coreografía.

En la Figura 20 se presenta un fragmento de código WS-CDL, en donde se describe parte de una coreografía en la que la DNIC y el BPS comparten algún tipo de información. Se definen los dos roles correspondientes a las dos organizaciones que participan en el intercambio, la relación entre ellos y se muestra la primera parte de la especificación de la coreografía [31].

```
<package>
    <role name="DNIC "/>
    <role name="BPS "/>
    <relationship name="Comparte">
        <role type=" DNIC "/>
        <role type=" BPS "/>
    </relationship>
    <channelType/>
    <choreography name="CoreografiaGeneral ">
        <relationship type="Comparte">
            <variableDefinitions>
                <variable name="AcciónCrear" silent-action="true" role=" BPS "/>
            <variableDefinitions/>
```

Figura 20 – Ejemplo de código WS-CDL [31]

3.2.1.2. *Business Procces Model and Notation* (BPMN)

Es un lenguaje gráfico estandarizado, enfocado en el modelado de procesos de negocio y que utiliza un formato de flujo de trabajo (*workflow*). BPMN define un lenguaje sencillo y comprensible, que puede ser utilizado por personal no técnico (particularmente analistas de negocio) y por profesionales de múltiples disciplinas [7].

BPMN da soporte a tres categorías principales de procesos:

1-Orquestación.

2-Colaboración, que puede incluir Orquestación y/o Coreografías.

3-Coreografía.

Para el proyecto se estudia el último: coreografías BPMN.

- **Coreografías BPMN**

Las coreografías BPMN se centran en la manera en que las organizaciones involucradas en el negocio coordinan sus interacciones, es decir, que el foco es el intercambio de información entre estos participantes.

La Figura 21 muestra un ejemplo de una coreografía BPMN. En el ejemplo se visualiza un proceso de intercambio de mensajes entre dos organizaciones: DNIC y BPS. Los mensajes refieren a la solicitud de datos de un ciudadano socialmente vulnerable.

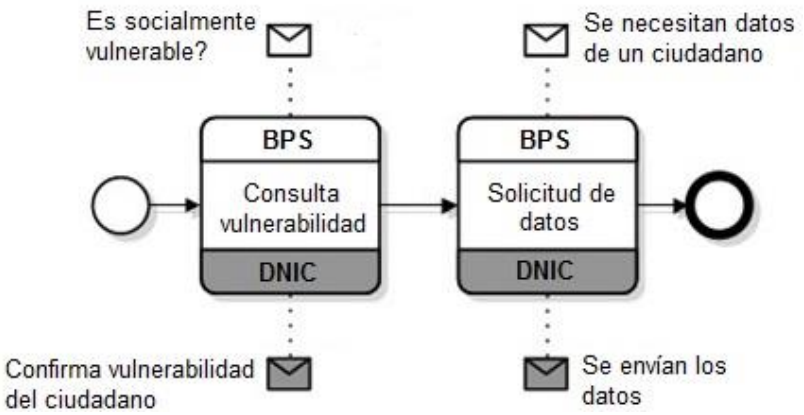

Figura 21 – Coreografía BPMN [7]

Para comprender el estado actual de una coreografía cada organización participante debe observar el comportamiento que el resto tiene sobre la misma. En caso de ser sólo dos los involucrados en el intercambio de información, ambos conocerán al responsable de enviar el próximo mensaje [7].

3.2.1.3. Selección

WS-CDL presenta una representación directa en lenguaje XML para especificar el intercambio de mensajes, y BPMN requiere de un proceso de mapeos para llegar al lenguaje XML. Por lo tanto WS-CDL se acerca más con el cumplimiento del primer criterio de selección.

BPMN consiste en una representación gráfica y en un modelo semántico XML asociado. Cada elemento en el modelo gráfico corresponde a un elemento en el modelo semántico. Tanto BPMN como WS-CDL tienen como objetivo principal proveer una notación estándar que sea fácilmente legible y entendible por todos los participantes involucrados. Por lo tanto ambos lenguajes presentan una correspondencia intuitiva con los elementos del contexto de trabajo, y cumplen así con el segundo criterio de selección.

Debido a lo mencionado anteriormente WS-CDL se acerca más a cumplir con ambos criterios de selección y es el elegido para enfocar el estudio del proyecto.

3.2.2. QoS (*Quality of Service*)

Existen varios lenguajes que permiten especificar normativas que refieren a la Calidad de Servicio. Por ejemplo: WSQDL (*Web Service Quality Description Language*), QML (*QoS Modeling Language*) y DDS (*Data Distribution Service*). WSQDL y QML presentan muy buena documentación con varios ejemplos en la web. Para DDS no se encuentran muchos ejemplos y por lo tanto se decide por analizar WSQDL y QML.

3.2.2.1. *Web Service Quality Description Language* (WSQDL)

WSQDL es un esquema en formato XML para describir los valores de los factores de calidad de servicio web (*Web Service Quality Factors,* WSQF) con métodos de medición y evaluación [32].

Los factores de calidad de servicio web se componen de la calidad de valor de negocio, la calidad de medición de nivel de servicio, la calidad de interoperabilidad, la calidad de procesamiento de negocio, la calidad de gestión y la calidad de seguridad [33].

Si los factores de calidad están relacionados con la perspectiva del negocio, se trabaja con el grupo *Business Quality Group,* y si tienen que ver más con la perspectiva del sistema se trabaja con el grupo *System Quality Group*: ambos observables en la Figura 22 [33]. Por más detalle ver Apéndice 2.

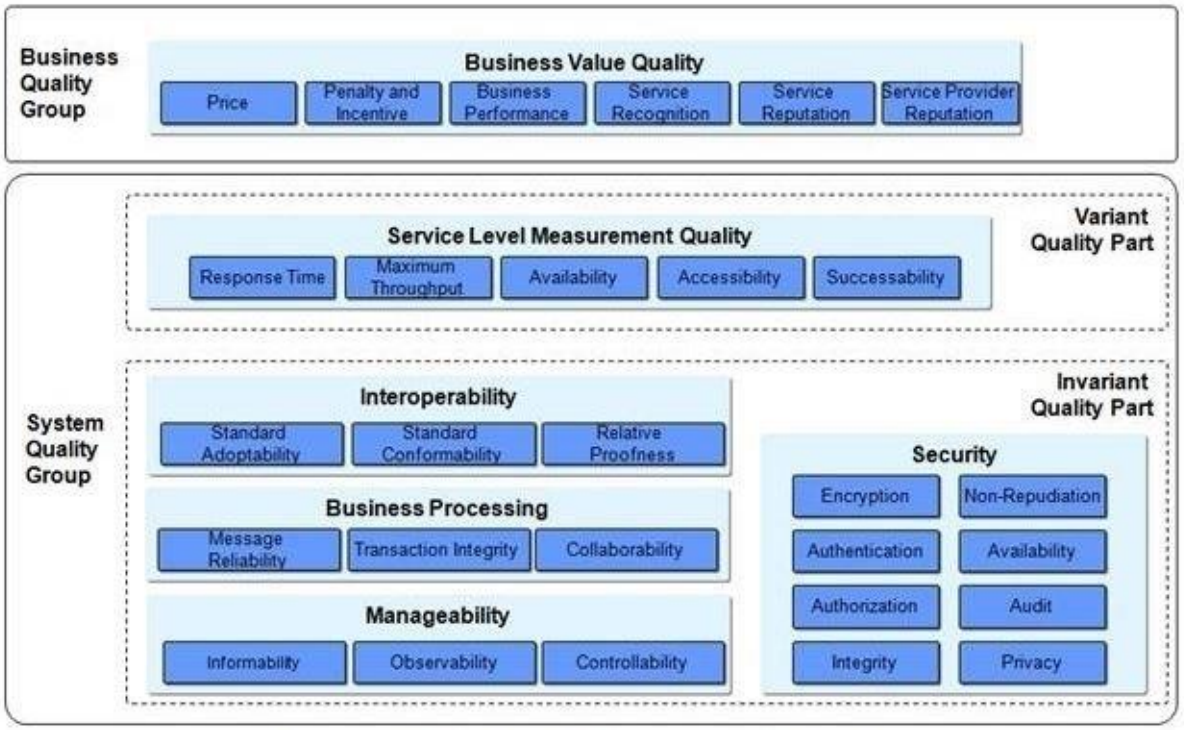

Figura 22 – Estructura de Web Service Quality Factor [33]

La Figura 23 presenta un ejemplo de definición de una instancia de WSQDL, con un factor de calidad vinculado al sistema que es el tiempo de respuesta (definido en el *MeasureFactor*) y otro vinculado al negocio que es costo de servicio (definido en el *BizValueFactor*). Como ejemplo se muestra la definición de las variables, métricas y cómo comienza a definirse la función para una evaluación [34].

```
<QualityFactor>
    <MeasureFactor>
        <ResponseTime>
            <MeasureDirection>
                <ReadingNumber>4132543</ReadingNumber>
            </MeasureDirection>
            <EnvVariables>
                <Variable>
                    <VarName>number of CPU</VarName>
                    <VarValue>4</VarValue>
                </Variable>
            </EnvVariables>
            <MetricValue>
                <Range>0.12 - 0.17</Range>
                <Type>float</Type>
                <Unit>second</Unit>
            </MetricValue>
        </ResponseTime>
    </MeasureFactor>
    <BizValueFactor>
        <ServiceCost>
            <ServicePrice>
                <Price unit="won">100000</Price>
                <ContractDoc>
                    <ContractDocNumber>CN20034324</ContractDocNumber>
                </ContractDoc>
                <PayMethod>quaterly</PayMethod>
            </ServicePrice>
        </ServiceCost>
    </BizValueFactor>
    <EvalFactor>
        <Security>
            <Property name="confidentiality">
                <SubProperty name="message level confidentiality">
                    <Function name="XML ENC Handler" severity="1">
```

Figura 23 – Ejemplo de Código WSQDL [34]

3.2.2.2. QoS *Modeling Language* (QML)

QML está diseñado para integrarse con características orientadas a objetos, como interfaces, clases y herencia. En particular, permite especificar las propiedades de calidad de servicio a través del refinamiento de las especificaciones de calidad de servicio ya existentes [35].

QML tiene tres mecanismos de abstracción principales para la especificación: *contract type*, *contract* y *profile*. QML permite definir tipos de contratos que representan aspectos específicos de QoS, como el rendimiento o la fiabilidad. Por más detalle ver Apéndice 2.

Las definiciones de QML en la Figura 24 incluyen dos tipos de contratos: Fiabilidad y Rendimiento. El tipo de contrato de fiabilidad tiene tres dimensiones. La primera representa el número de fallas por año. La segunda, el tiempo de reparación (TTR), representa el tiempo que tarda en reparar un servicio que ha fallado. Y la tercera, la disponibilidad,

representa la probabilidad de que un servicio esté disponible. En este caso los valores mayores representan restricciones más fuertes, mientras que los valores más pequeños representan probabilidades más bajas y por lo tanto son más débiles [35].

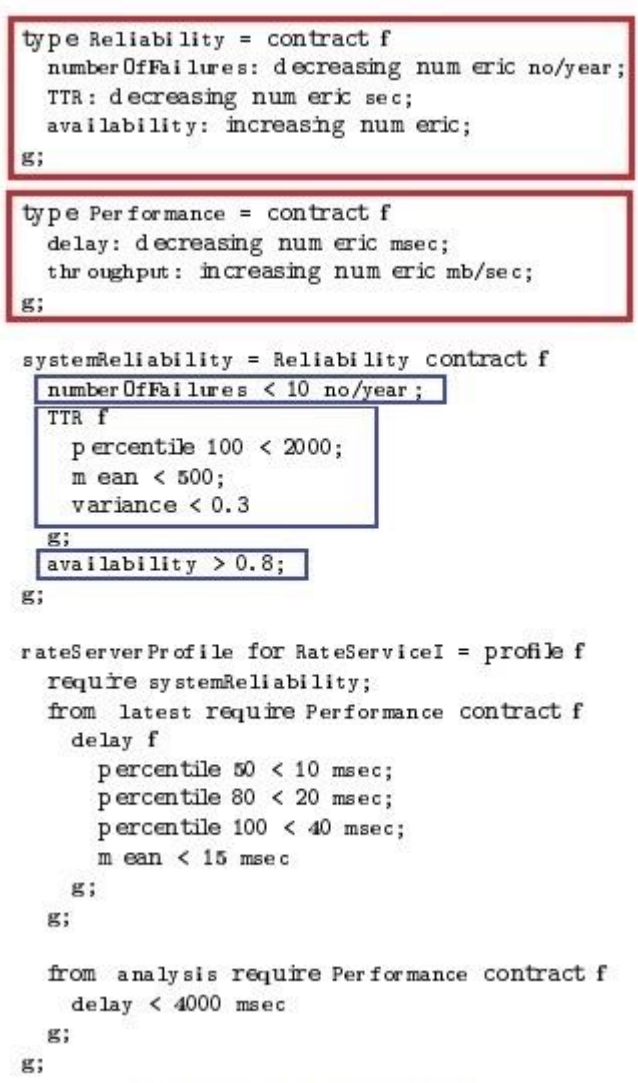

Figura 24 – Ejemplo QML [35]

3.2.2.3. Selección

QML admite la especificación de las propiedades de calidad de servicio de una manera orientada a objetos. Está enfocado más a clases, interfaces y herencia. Aunque QML no está vinculado a ninguna notación de diseño en particular, es posible integrarlo con UML y proporcionar una sintaxis gráfica para las especificaciones de QoS a nivel de componentes. Sin embargo WSQDL es un lenguaje ya especificado en XML que describe valores de calidad de servicio incluyendo factores de medición y evaluación. Por lo tanto WSQDL cumple con el primer criterio de selección.

WSQDL es entendible y proporciona una variedad de propiedades muy útiles, con lo que es muy viable la correspondencia a la información manejada en el contexto del proyecto. WSQDL se acerca mucho más a cumplir con el segundo criterio de selección.

Por lo mencionado anteriormente es que para el proyecto se decide optar por WSQDL como lenguaje de especificación de Calidad de Servicio.

3.2.3. *Service Level Agreement* (SLA)

Existen varias formas para poder especificar Acuerdos de Nivel de Servicio. Entre ellas, se encuentran los lenguajes *Web Service Level Agreement* (WSLA) y SLA *definition language* (SLAng). Ambos presentan buena documentación en la web.

3.2.3.1. *Web Service Level Agreement* (WSLA)

WSLA es un estándar para monitorear el cumplimiento de acuerdos de nivel de servicios web. Entre los parámetros que maneja se encuentra el tiempo de respuesta y rendimiento, y las medidas a tomar en caso de incumplimiento de las garantías del servicio declaradas[36].

El lenguaje de especificación WSLA se define como un esquema XML. WSLA puede ser utilizado tanto por el proveedor de servicios como por el cliente del servicio con el fin de configurar sus respectivos sistemas. Este proceso se llama despliegue. Cada organización utiliza su propia función de despliegue independiente [36].

En la Figura 25 se observan las tres partes importantes de un acuerdo de nivel de servicio.

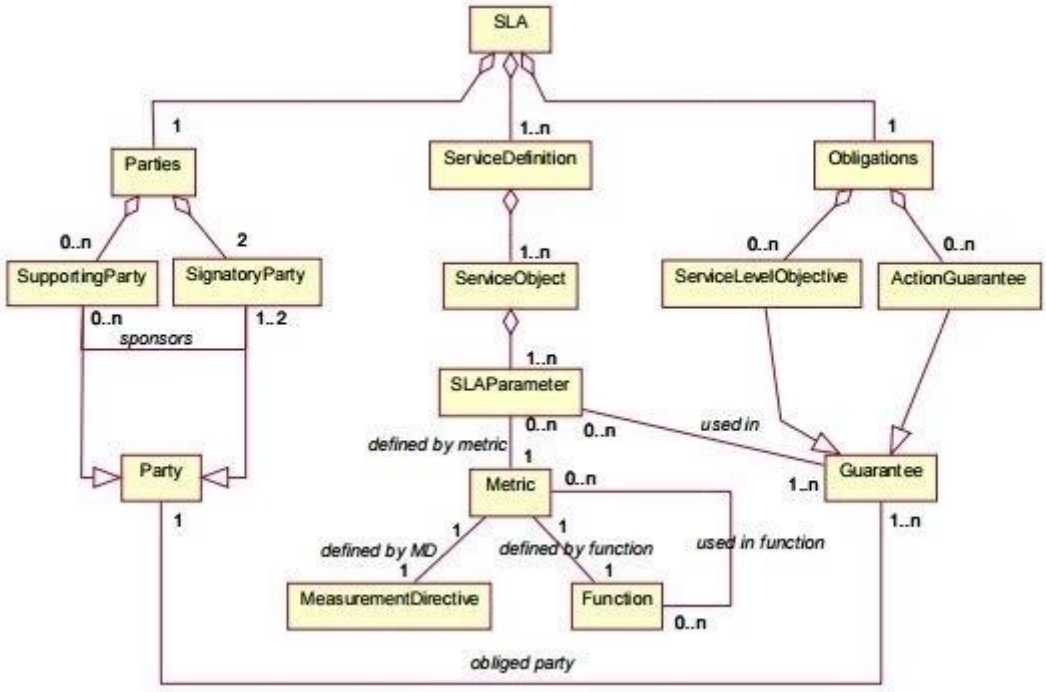

Figura 25 – Esquema Web Service Level Agreement [36]

***Parties*:** describen las partes que intervienen en la gestión del servicio web. Esto incluye las partes firmantes, así como las partes de soporte para actuar en nombre del proveedor de servicios.

***ServiceDefinition*:** describen los servicios sobre los que el acuerdo del WSLA actúa. Las definiciones de servicios se encargan de representar la comprensión común de las partes contratantes, en términos de operaciones y parámetros del servicio y sus métricas.

***Obligations*:** definen el nivel de servicio que se garantiza con respecto a los *SLAParameters* que fueron previamente declarados en la sección de definición de servicio. Por más detalle ver Apéndice 2.

La Figura 26 muestra un ejemplo de código WSLA. En el ejemplo se visualiza la definición de *Service Level Objective* al que se hizo referencia previamente desde una garantía (*Action*Guarantee). El *ServiceLevelObjetive* indica la organización (DNIC) encargada de dicho objetivo, el período de validación y una expresión. La expresión contiene un predicado que define el tiempo medio de respuesta de menos (indicado en el *Predicate* como "wsla:Less") de 5 segundos. En este caso, *AverageResponseTime* es una referencia a una *SLAParameter* definido en la sección de *Service Definition* del SLA. Se evalúa la expresión de acuerdo con el *NewValue* del *EvaluationEvent* [36].

```
<ServiceLevelObjective name="g1">
    <Obliged>DNIC</Obliged>
    <Validity>
        <Start> 2016 -11-30T14:00:00.000-05:00</Start>
        <End> 2016 -12-31T14:00:00.000-05:00</End>
    </Validity>
    <Expression>
        <Predicate xsi:type="wsla:Less">
            <SLAParameter>AverageResponseTime</SLAParameter>
            <Value>5</Value>
        </Predicate>
    </Expression>
    <EvaluationEvent>NewValue</EvaluationEvent>
</ServiceLevelObjective>
```

Figura 26 – Ejemplo de Código WSLA [36]

3.2.3.2. SLA *definition language*

Los acuerdos de nivel de servicio capturan las responsabilidades mutuas del proveedor de un servicio y su cliente con respecto a las propiedades no funcionales que se establezcan. SLAng proporciona no sólo un formato para la negociación de las propiedades de calidad de servicio, sino también un lenguaje apropiado como entrada para sistemas de razonamiento automatizados [37].

SLAng satisface la necesidad de un lenguaje SLA para soportar la construcción de sistemas distribuidos y aplicaciones con características de QoS fiables. La estructura sintáctica y la semántica de SLAng se definen con referencia a un modelo de arquitectura de sistemas distribuidos [37]. Por más detalle ver Apéndice 2.

La Figura 27 muestra el nombre de los acuerdos de nivel de servicio que pueden ser contratados entre organizaciones, debidamente subdivididos en acuerdos verticales y horizontales [37].

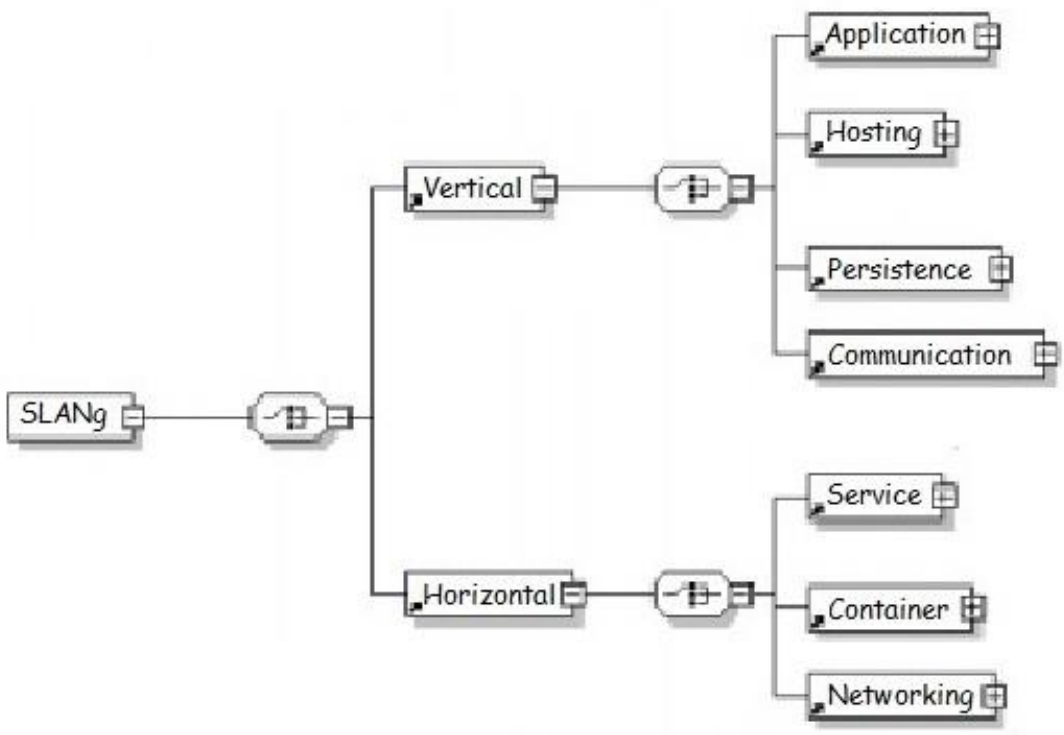

Figura 27 – SLAs Vertical y Horizontal [37]

En la Figura 28 se visualiza un ejemplo de SLAng horizontal en el que se definen condiciones para el servicio. Se muestran condiciones para la performance con un tiempo promedio, máximo y mínimo de respuesta [37].

```
<?xml version="1.0" encoding="UTF-8"?>
<SLAng xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dave/TAPAS/SLAng0_5/SLAng0_5.xsd">
    <Horizontal>
        <Service>
            <Id sls_id="453A" service_id="storage"/>
            <Client>
                <Name>BPS</Name>
                <Place>Montevideo</Place>
                <Availability>96%</Availability>
            </Client>
            <Server>
                <Name>DNIC</Name>
                <Place>Montevideo</Place>
                <Availability>95%</Availability>
                <Maintenance recovery_time="2" scheduled_outages="8"
                    routine_maintenances="5"/>
            </Server>
            <Mutual>
                <Service_schedule start="2016-12-13" end="2020-12-13"/>
                <Performance>
                    <Service_time average="7.3" maximum="16.9" minimum="4.6"/>
                    <Service_rate>26.7</Service_rate>
                </Performance>
            </Mutual>
        </Service>
    </Horizontal>
</SLAng>
```

Figura 28 – Ejemplo de Código SLA Horizontal [37]

3.2.3.3. Selección

Ambos lenguajes analizados permiten la especificación utilizando un formato en XML. Si bien SLAng proporciona una variedad de ventajas ya mencionadas, su estructura en XML no es tan amigable y entendible como lo es la de WSLA para especificar acuerdos de nivel de servicio. Por lo tanto WSLA es el lenguaje seleccionado en base al primer criterio de selección.

Una de las ventajas que presenta WSLA es que es un lenguaje XML, estructurado de tal manera que las cláusulas de seguimiento pueden ser separadas de los términos contractuales para su distribución a un tercero. Esto permite una buena adaptación en la correspondencia a elementos del contexto de trabajo. WSLA se acerca más que SLAng a cumplir con el segundo criterio de selección.

SLAng presenta un diseño mucho más complejo que WSLA y por lo tanto, en base a los criterios establecidos, se decide seleccionar este último para la utilización y estudio en el proyecto.

3.2.4. Protección de datos personales en XACML

Para especificar el conjunto de reglas relacionado a la protección de datos personales se estudia el estándar *eXtensible Access Control Markup Language* (XACML). XACML presenta una estructura basada en XML y un diseño amigable y adaptable a los conceptos manejados en el contexto del proyecto.

En base a las estructuras brindadas por el estándar XACML presentado en el capítulo 2, se establece el diseño utilizado para representar la correspondencia del Conjunto de Políticas, Políticas y Reglas para la Protección de Datos Personales que se desarrolló en el proyecto de grado [38].

Como se menciona en el trabajo [38], Cada ciudadano brinda un conjunto de consentimientos para cada Acción y Propósito que se asocian a una organización. Existe una correspondencia entre el consentimiento y la estructura de una *Policy* de XACML.

La Figura 29 muestra un ejemplo de un ciudadano con cédula de identidad 1.234.567-8 que brinda consentimiento para la Acción *Select_for_use* y el Propósito *MDM_BusinessOriented* al organismo DGI sobre los datos personales Ascendencia y EstadoCivil [38].

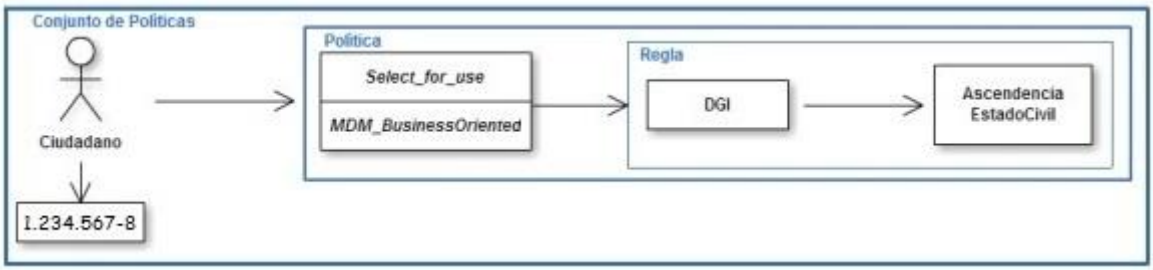

Figura 29 – Ejemplo Consentimiento Ciudadano como elementos XACML [38]

A partir de la agrupación de la Figura 29 se establece la correspondencia con su estructura XACML. El *Conjunto de Políticas* agrupa las políticas de cada ciudadano, las Políticas se definen para cada Acción y Propósito válidos, y el conjunto de Reglas (dentro de cada política) contiene la organización y las condiciones para los datos autorizados [38].

La Figura 30 Muestra un ejemplo del código de la *rule* representada en la Figura 29. En la *Rule* la organización DGI y los datos autorizados: EstadoCivil y Ascendencia.

```
<Rule Effect="Permit" RuleId="Rule0">
  <Target>
    <AnyOf>
      <AllOf>
        <Match>
          <AttributeValue>DGI</AttributeValue>
          <AttributeDesignator AttributeId="subject:subject-id"
            Category="subject-category:access-subject"></AttributeDesignator>
        </Match>
      <AllOf>
    <AnyOf>
  </Target>
  <Condition>
    <Apply>
      <AttributeValue>Ascendencia</AttributeValue>
        <AttributeDesignator AttributeId="resource:Target"
          Category="attribute-category:resource"></AttributeDesignator>
    </Apply>
    <Apply>
      <AttributeValue>EstadoCivil</AttributeValue>
        <AttributeDesignator AttributeId="resource:Target"
          Category="attribute-category:resource"></AttributeDesignator>
    </Apply>
  </Condition>
</Rule>
```

Figura 30 – Ejemplo de Rule a partir del consentimiento [38]

3.3. Trabajo relacionado

En esta sección se analiza trabajo relacionado al proyecto en cuestión y herramientas ya existentes en la actualidad.

3.3.1. Proyectos

En esta sección se describen tres proyectos.

3.3.1.1. Implementación de una Plataforma para la Protección de Datos Personales en Soluciones *Master Data Management* de Gobierno Electrónico

Uno de los trabajos analizados fue el realizado como Proyecto de Grado de la Facultad de Ingeniería de la Universidad de la República entre los años 2015 y 2016 [38].

El proyecto se basa en una realidad en la que se comparten datos personales entre organizaciones. Los datos deben estar actualizándose continuamente, deben ser validados y se deben poder obtener de fuentes confiables [38].

El proyecto busca como objetivo el desarrollo de un prototipo que asegure la protección de datos personales según la Ley Nº 18.331 del estado uruguayo. Las plataformas de integración deben ser capaces de brindar mecanismos para monitorear y hacer cumplir

normativas referentes a la protección de datos personales. El proyecto consistió además en la realización de un prototipo para asegurar la protección de datos personales. El prototipo consta de la combinación entre un sistema que implementa el estándar XACML y tecnologías *Enterprise Service Bus* (ESB) [38].

Si bien este proyecto plantea una solución para monitorear normativas en contexto interorganizacional, se enfoca específicamente en protección de datos personales.

Igualmente se pudo obtener información con respecto al estándar XACML y su utilización para poder cumplir normativas de protección de datos personales que ayudaron a comprender el estándar y a manejar un tipo de normativa que se desarrolla en nuestro proyecto.

3.3.1.2. *Privacy Preserving Event Driven Integration for Interoperating Social and Health Systems*

Este trabajo [39] está más enfocado al cumplimiento de normativas de privacidad de datos personales en un contexto interorganizacional en el área de la salud.

En este trabajo se propone un lenguaje específico y se implementan un conjunto de reglas que se realizan para el control del cumplimiento de normativas mencionado. De manera similar al proyecto mencionado anteriormente, utiliza políticas y reglas que brinda el estándar XACML para definir las normativas que hacen referencia a la protección de datos personales en Salud.

3.3.1.3. QoS *Based Web Service Selection*

Este trabajo [40] aborda temas de calidad de los servicios web. En particular, el trabajo propone una herramienta para monitorear y gestionar los servicios web distribuidos (*Web Services Distributed Management*, WSDM). Por más detalle ver Apéndice 3.

Este trabajo se enfoca en la gestión y administración de normas especificadas en un lenguaje determinado: QoS.

Se pudieron validar los parámetros a tener en cuenta al momento de gestionar normativas correspondientes a calidad de servicio y las formas en que se especifican las normativas de esta categoría.

3.3.2. Herramientas

En esta sección se describen 3 herramientas relacionadas al proyecto

3.3.2.1. *SeaFlows Toolset: Compliance Verification Made Easy*

SeaFlows Toolset se enfoca en la verificación de estructura y datos partiendo de un modelo de proceso especificado previamente en un lenguaje [41]. Por más detalle ver Apéndice 3.

De esta herramienta no se lograron obtener aportes significativos para nuestro proyecto más allá de la posibilidad de poder verificar estructura y contenido sobre repositorios de actividad o modelados de procesos.

Más allá del soporte a múltiples lenguajes de especificación para el modelado de procesos, no posee uno común. Y comparándolo con el punto fuerte de la herramienta Gestor de Normativas llevada a cabo, no posee una representación en un lenguaje común.

3.3.2.2. QUICKER: *A Model-driven QoS Mapping Tool for QoS-enabled Component Middleware*

QUICKER mejora el lenguaje de modelado de componentes independientes de plataforma (*Platform-Independent Component Modeling Language, PICML*) [42]con nuevas construcciones que permiten a los desarrolladores de sistemas especificar y analizar políticas de QoS de aplicaciones en un nivel de abstracción superior al utilizado por lenguajes de programación de tercera generación (como Java o C ++) o por declaraciones textuales (como XML) [43]. Por más detalle ver Apéndice 3.

La herramienta aportó conocimientos en el área de QoS, vinculados a posibles transformaciones para mejorar un modelo existente.

QUICKER es una herramienta enfocada en la gestión de especificaciones de calidad de servicio (una de las áreas en las que se enfoca la herramienta Gestor de Normativas desarrollada), centrándose fuertemente en mejorar el modelado de componentes independientes de una plataforma.

3.3.2.3. PPINOT *Tool Suite: A Performance Management Solution for Process-Oriented Organizations*

La PPINOT *Tool Suite* [44] ofrece dos formas diferentes de definir Indicadores de Desempeño de Procesos. Por un lado gráficamente, junto con la representación BPMN de procesos de negocio a partir de un editor gráfico, que es un editor web que se ha implementado como una extensión de la plataforma Oryx [45]. Y por otro lado, utilizando un editor de plantillas que ofrece la herramienta. En ambos casos, se obtiene como resultado un documento XML con la información del PPI y con el modelo del proceso de negocio. Por más detalle ver Apéndice 3.

En conclusión, PPINOT está fuertemente enfocada a los modelos de procesos de negocio y en particular a la gestión de indicadores de desempeño de procesos. Como aporte a nuestro proyecto se obtuvieron formas de transformar los modelos a documentos XML.

3.4. Resumen

El análisis comienza con una breve descripción del problema, que lleva a definir un modelo conceptual y luego establecer los requerimientos funcionales y no funcionales para determinar el alcance del sistema. En base a estos requerimientos se definen los tipos de usuarios y los casos de uso.

En base a los criterios de selección establecidos, se decide optar por el lenguaje WSQDL, expresado en formato XML, como tipo de normativa correspondiente a la categoría de calidad de servicio. Se selecciona WSLA para la categoría correspondiente a acuerdo de nivel de servicio. Se decide trabajar con WS-CDL como forma de especificar el intercambio de mensajes para una coreografía. Y para la protección de datos personales se toma el lenguaje XACML.

En cuanto al análisis del trabajo relacionado, se puede aprovechar la información enfocada a las normativas y su representación especificada en un lenguaje en particular, y además, a la gestión de alguna de ellas. Pero está faltando una herramienta que permita gestionar múltiples lenguajes de especificación de normativas, y que, como punto sumamente importante, brinde la posibilidad de representar de forma unificada esos lenguajes manejados.

No se encontró ninguna herramienta capaz de cumplir con los requerimientos de nuestro proyecto.

4. Solución Propuesta

En esta sección se presenta la solución propuesta para abordar la problemática planteada en el proyecto. En primer lugar se brinda una descripción general de la solución. Luego se describen con más detalle las características más relevantes de la misma. Por último, se presenta la arquitectura de la solución y se describen decisiones tomadas.

4.1. Descripción general

Se propone el desarrollo de un Gestor de Normativas que permita a organizaciones que interactúan a través de una plataforma de integración gestionar normativas independientemente de su lenguaje de especificación. En la Figura 31 se muestra la herramienta Gestor de Normativas y cómo se posiciona en la realidad planteada en la sección 3.1.1.

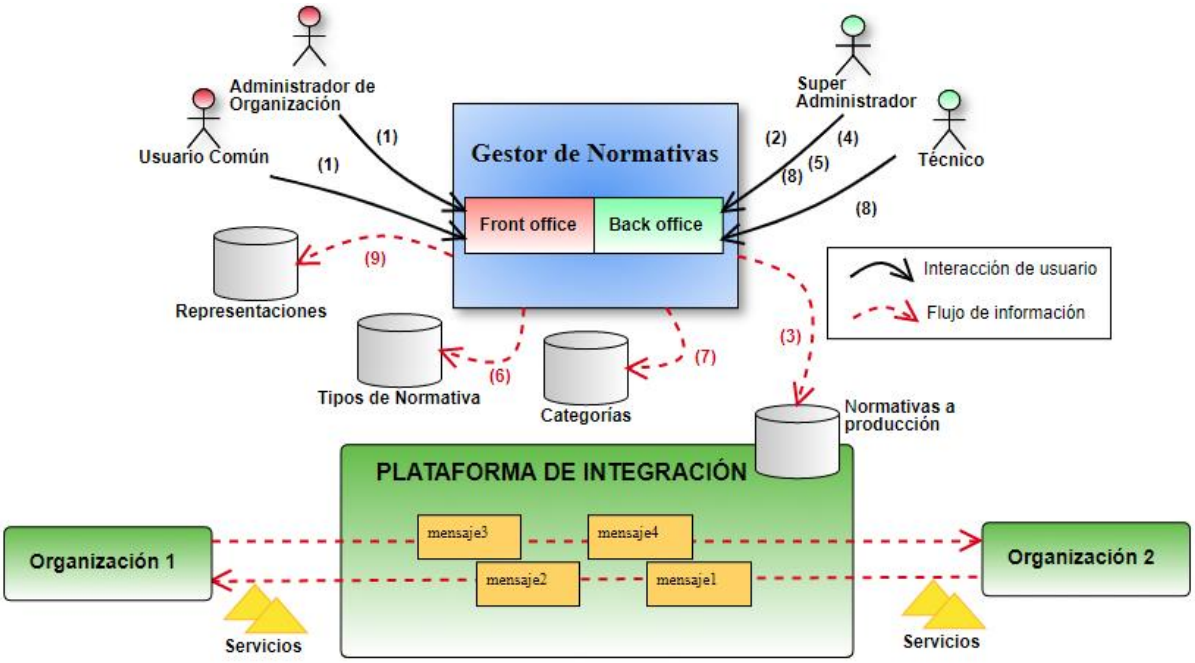

Figura 31 – Solución propuesta. Gestor de Normativas

El gestor permite a usuarios pertenecientes a organizaciones dar de alta nuevas normativas (p. ej.: un acuerdo de nivel de servicio entre BPS y DNIC). La herramienta es la encargada de transformar esas normativas a un lenguaje de especificación común independientemente del lenguaje propio de cada normativa inicial. Una vez realizadas las transformaciones, el gestor brinda un sistema de aprobaciones de normativas en el que los usuarios administradores de organizaciones y usuarios comunes (1), cuyos servicios están involucrados en las normativas especificadas, deben aprobar las mismas. Finalizado el flujo de aprobaciones, la herramienta permite a usuarios Super Administrador (2) poner en producción (desplegar) las normativas en un servidor externo (3).

Para dar soporte al ciclo de vida de estas normativas, el gestor brinda una serie de funcionalidades a los usuarios Super Administrador de la herramienta. Entre ellas se encuentra la de dar de alta tipos de normativas (p. ej.: WSLA o WSQDL) (6) que especifican la estructura y sintaxis de las normativas que se van a querer gestionar por los usuarios Super Administrador (4). Los tipos de normativas corresponden a diferentes categorías (p. ej.: SLA o QoS) (7) que debieron darse de alta previamente en el Gestor de Normativas y que pueden ser agregadas únicamente por usuarios Super Administrador (5). También se debieron haber dado de alta representaciones (p. ej.: HTML o lenguaje común) (9) para que puedan ser asignadas a los tipos de normativas. De esta manera las normativas podrán pasar de una representación a otra a través de una transformación (XSLT o Freemarker). Tanto las representaciones como las transformaciones son gestionadas por administradores de la herramienta o por usuarios técnicos especializados (8).

La solución se compone de un Backoffice y un Frontoffice. El Backoffice brinda funcionalidades relacionadas a la administración de nuevos tipos de normativas, manejo de organizaciones existentes y categorías que maneja la herramienta. El Backoffice permite también la gestión de tipos de usuarios en el Gestor de Normativas. El Frontoffice brinda funcionalidades que manejan Normativas, aprobaciones de las mismas y las distintas representaciones que le correspondan. Los usuarios Administradores de la herramienta y técnicos acceden al Backoffice y los Usuarios Comunes y Administradores dentro de una organización acceden al Frontoffice.

4.2. Funcionalidades

A continuación se describen los mecanismos para cada una de las funcionalidades a resolver.

4.2.1. Soporte a distintos tipos de normativas

En el proyecto se investigaron, estudiaron y analizaron especificaciones de tipos de normativas pertenecientes a diferentes categorías, como son: calidad de servicio, protección de datos personales, acuerdo de nivel de servicio e intercambio de mensajes entre organizaciones. La herramienta soporta, no sólo la gestión de múltiples tipos de normativas correspondientes a diferentes categorías, sino también la creación de nuevas categorías.

En la Figura 32 se muestra un ejemplo de cómo funciona el soporte a distintos tipos de normativas al momento de crear una nueva normativa.

Figura 32 – Validación de normativas

Para soportar esta funcionalidad se almacena la información de los tipos de normativas junto con sus plantillas especificadas mediante XML *schema* que indican la estructura y sintaxis que deben presentar todas las normativas pertenecientes a dicho tipo como se muestra en la Figura 32. Almacenando estos datos se permite que la creación de tipos de normativas sea dinámica, y luego, con la creación de nuevas normativas sólo hará falta validarlas con dichas plantillas.

En el ejemplo de la Figura 32 se observa que al tener una nueva normativa “NormativaA” se intenta validar con cada una de las plantillas correspondientes a los tipos de normativas: “TipoNormativa1” y “TipoNormativa2”. Esa validación se realiza contra la estructura y sintaxis de las plantillas. Al momento en que la “NormativaA” valida con el “TipoNormativa2” (y no así en su primer intento con el “TipoNormativa1”), se establece la relación entre la “NormativaA” y el “TipoNormativa2”, y se almacena esta información. En caso de que una normativa valide con más de una opción, la herramienta selecciona automáticamente el primer tipo de normativa que valide.

Los tipos de normativas también poseen un conjunto de representaciones. Las diferentes representaciones de una normativa son posibles gracias a archivos de transformación que se almacenan dinámicamente al momento de crear cada tipo de normativa.

4.2.2. Soporte a distintas representaciones de normativas

Las normativas que manejan las organizaciones son especificadas en diferentes lenguajes. La idea es que, para la gestión de normativas de la herramienta, todas queden representadas en un lenguaje común. La herramienta es extensible, da soporte a la representación de las normativas en múltiples lenguajes de especificación.

En la Figura 33 se muestra un ejemplo con cuatro representaciones de una normativa. Las diferentes representaciones de una especificación son logradas mediante una transformación.

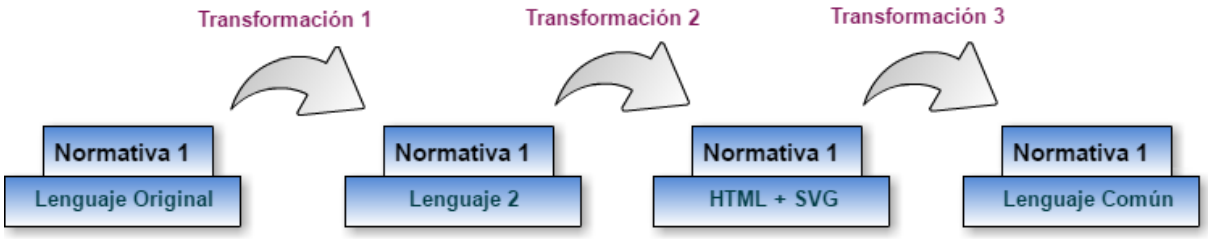

Figura 33 – Representación de múltiples normativas

Las normativas son instancias de Tipos de Normativas existentes, y que al momento de darlas de alta, se debe determinar un conjunto de archivos de transformaciones. Los archivos para transformar normativas son almacenados junto con su información de uso: archivo de entrada especificado en un lenguaje, archivo de salida especificado en otro lenguaje y los tipos de normativas sobre las que se encuentra habilitado para actuar. Por lo tanto, se pueden tener normativas representadas en el lenguaje de especificación original, una cantidad variada de lenguajes determinados por el usuario de la herramienta, y la especificación en lenguaje común.

En la Figura 33 se muestra un ejemplo de diferentes transformaciones. En particular, una de ellas es con la utilización de SVG, que permite representar las normativas gráficamente, utilizando HTML como lenguaje resultado de la transformación.

4.2.3. Acuerdos entre organizaciones: flujo de aprobaciones

La herramienta da soporte a acuerdos entre organizaciones al momento de dar de alta una nueva normativa. La idea es, que si dos organizaciones están involucradas en una normativa, exista un acuerdo entre ambas previo a la puesta en producción de la misma.

La herramienta maneja el concepto de flujo de aprobaciones. El flujo de aprobaciones consiste en que dos o más organizaciones puedan llegar a un acuerdo en lo que se refiere a normativas que las involucra.

La Figura 34 muestra un diagrama del flujo de aprobaciones y cómo sucede la interacción con la herramienta en el tiempo. Un primer usuario desea crear una normativa que involucra a dos organizaciones: una de ellas es a la cual pertenece (Organización1) y la otra es la Organización2. Un segundo usuario perteneciente a la Organización2, ingresa al Gestor de Normativas y aprueba la normativa. Sólo usuarios con el tipo de usuario Administrador de

Organización pueden aprobar normativas. Si el primer usuario era Administrador de Organización, la Normativa queda aprobada por todas las partes. En caso contrario, debe ingresar un usuario con el tipo de usuario Administrador de Organización perteneciente a la Organización1 y aprobarla también.

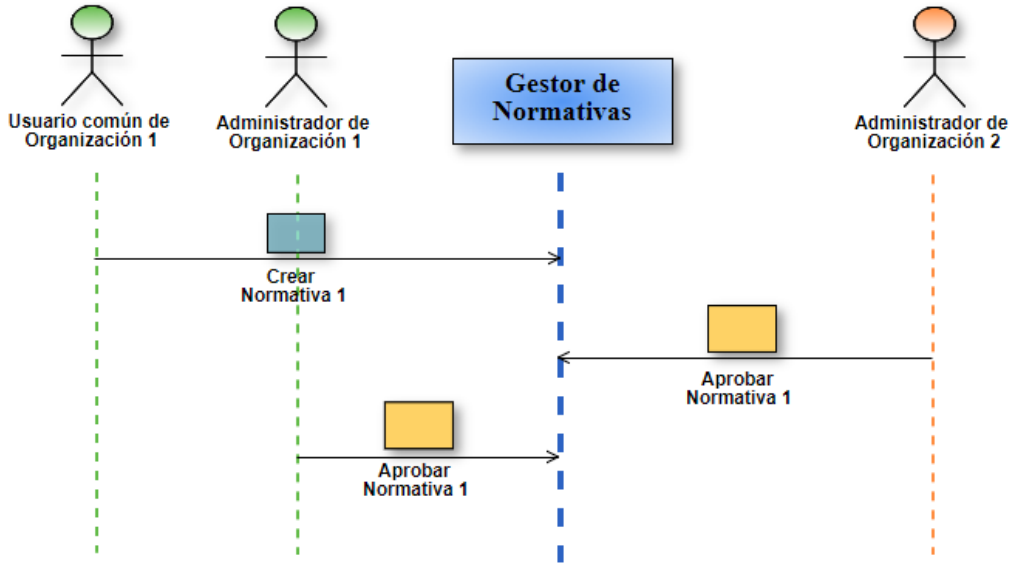

Figura 34 – Flujo de aprobaciones

Una vez llegado al acuerdo de todas las partes involucradas de la normativa, queda definitivamente aprobada y pronta para la puesta en producción.

Para resolver el flujo de aprobaciones de las normativas se define la máquina de estados que se muestra en la Figura 35. Los estados se van almacenando junto con los datos relacionados a las normativas y se van actualizando con las diferentes acciones en el flujo de aprobaciones de normativas.

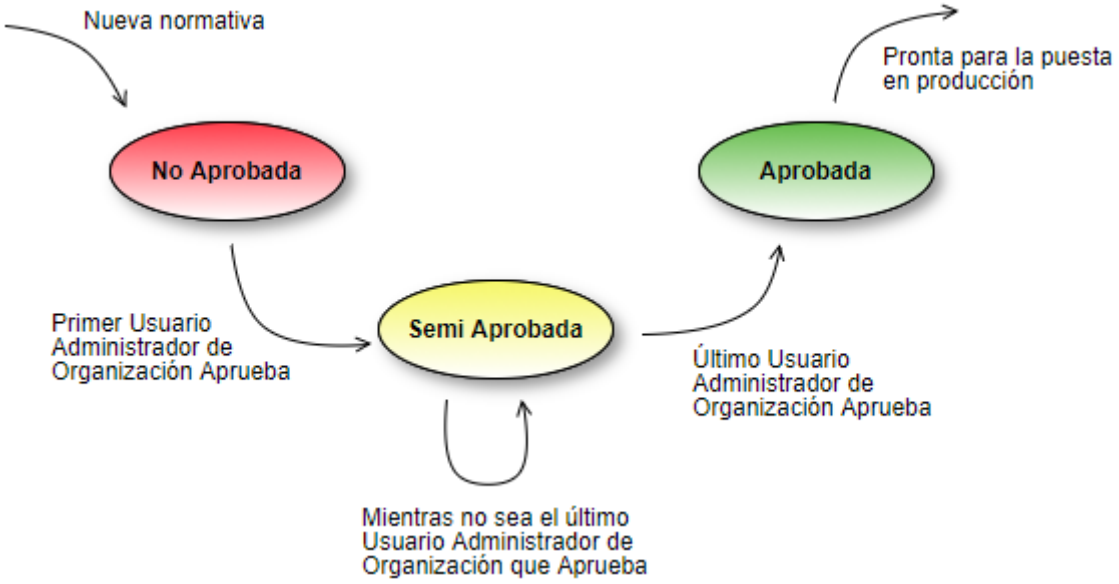

Figura 35 – Máquina de estados para flujo de aprobaciones de una normativa

Como se muestra en la Figura 35, cada normativa pasa por tres estados en el flujo de aprobación. Cuando se crea la normativa, comienza en el estado “No Aprobada”. Cuando el primer usuario Administrador de Organización aprueba la normativa pasa al estado “Semi Aprobada”. Una normativa puede involucrar múltiples organizaciones, por lo tanto deberá ser aprobada por todas las organizaciones participantes. Es así, que hasta no ser aprobada por usuarios de todas las organizaciones, la normativa permanecerá en este estado. Una vez aprobada por el último usuario Administrador de Organización, la normativa pasa al estado “Aprobada” y queda pronta para la puesta en producción.

El sistema de tipos de usuarios utilizado en el Gestor de Normativas es el que permite que sólo usuarios con permisos específicos puedan acceder a la aprobación de las normativas.

4.2.4. Puesta en producción de normativas en la plataforma

La herramienta da soporte a la puesta en producción de normativas. Cuando una normativa es creada, atraviesa un conjunto de transformaciones hasta llegar a su especificación en el lenguaje común. Luego, pasa por el flujo de aprobaciones mencionado en la sección 4.2.3. Una vez aprobada la normativa, un usuario Super Administrador de la herramienta puede ponerla en producción. La puesta en producción es resuelta consumiendo una API a través de una capa de servicios externos, desplegando la normativa en la Plataforma de Integración.

Como lenguaje de despliegue se utiliza XACML, para el cual se van a desplegar en la plataforma un conjunto de políticas para que en *runtime* se haga el *enforcement.*

Junto con cada normativa, se mantiene la información de su estado actual correspondiente al flujo de aprobaciones mencionado en la sección 4.2.3. Si la normativa está aprobada por las organizaciones involucradas se procede a desplegarla en la Plataforma de Integración. Finalmente se actualiza el estado final de la normativa: en producción.

4.3. Arquitectura del Sistema

En esta sección se presenta una visión global y comprensible del diseño general del sistema. Para ello, se presenta la arquitectura del sistema mediante diferentes vistas.

4.3.1. Vista del Modelo de Casos de Uso

En la Figura 32 se presentan los casos de uso más relevantes para la arquitectura del sistema.

Figura 36 – Vista de Casos de Uso Relevantes

Los casos de uso presentados en la Figura 36 son los más relevantes porque son los encargados de los flujos relacionados a Categorias, Organizaciones, Tipos de Normativa, Normativas y Tipos de usuarios, y estos son los elementos fundamentales del contexto de trabajo.

Las descripciones de los casos de uso se pueden visualizar en la sección 3.1.5. Por más detalle ver Apéndice 1.

4.3.2. Vista Lógica

El Gestor de Normativas sigue un modelo arquitectónico dividido en capas. En esta estructura, las capas de más arriba utilizan servicios definidos en las capas de más abajo.

Como se muestra en la Figura 37, el esquema maneja una capa de presentación que consiste de un componente para el Frontoffice y otro componente para el Backoffice. Se decide que la capa de presentación consuma directamente de la capa lógica.

La capa de datos encapsula el acceso a los diferentes tipos de datos y el mapeo a las representaciones.

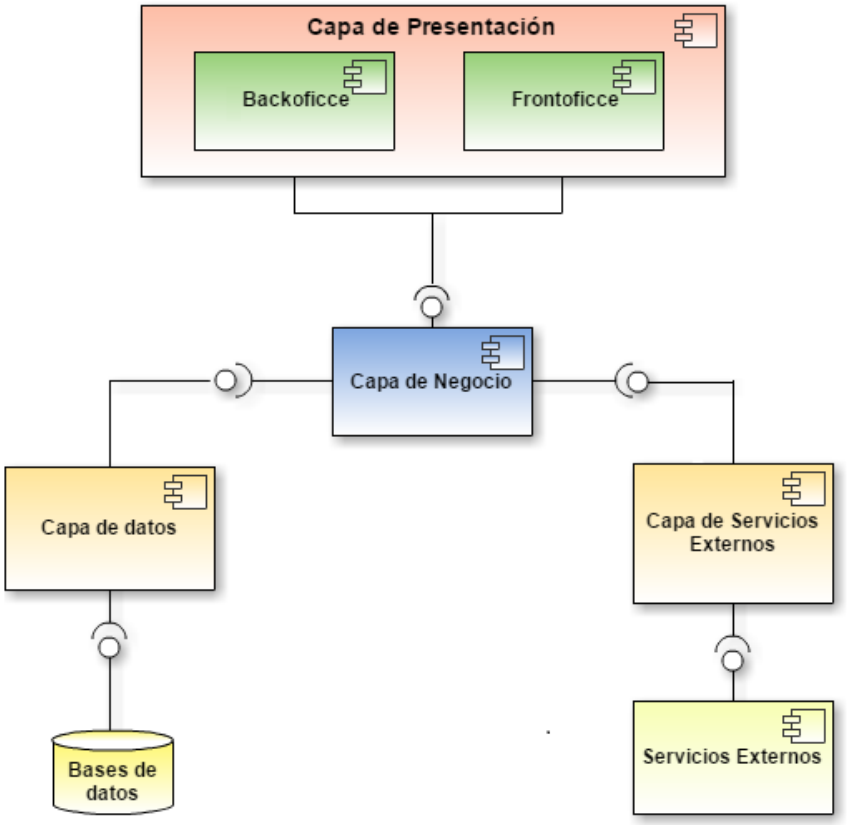

Figura 37 – Estilo arquitectónico

4.3.2.1. Capa de presentación

La Figura 38 muestra la capa de presentación dividida en Frontoffice y Backoffice y cómo los usuarios interactúan con ella.

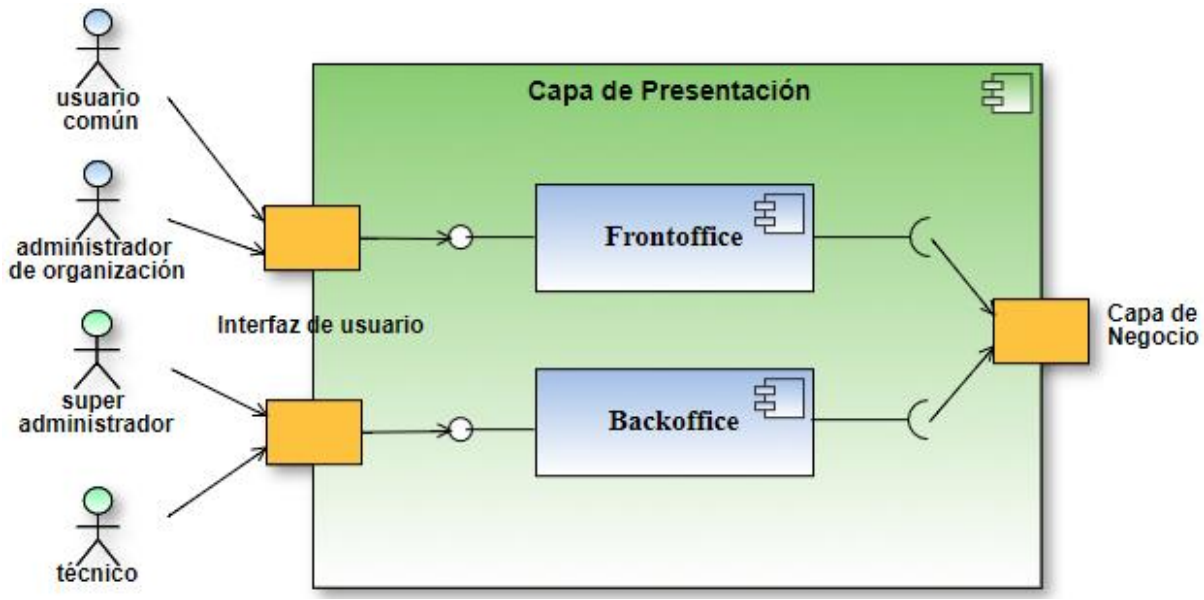

Figura 38 – Componentes Capa de Presentación

Como se comentó en la sección 4.1, la herramienta da soporte a un sistema de roles de usuarios. El Administrador de Organización y el Usuario Común acceden a la parte del Frontoffice de la capa de presentación, y el Super Administrador y el Técnico acceden al Backoffice.

Tanto el Frontoffice como el Backoffice acceden a la Capa de Negocio para resolver los pedidos de la capa de presentación.

4.3.2.2. Capa de negocio

La capa de negocio implementa la funcionalidad principal de la herramienta. Ésta encapsula la lógica de negocio relevante para la aplicación. La capa de negocio consiste en componentes que exponen (en algunos casos) interfaces para que otros componentes consuman. Esta componente controla el acceso a los servicios de negocio desde otras capas, la publicación de los servicios de negocio mencionados e invocación de la capa de persistencia tal como se muestra en la Figura 39.

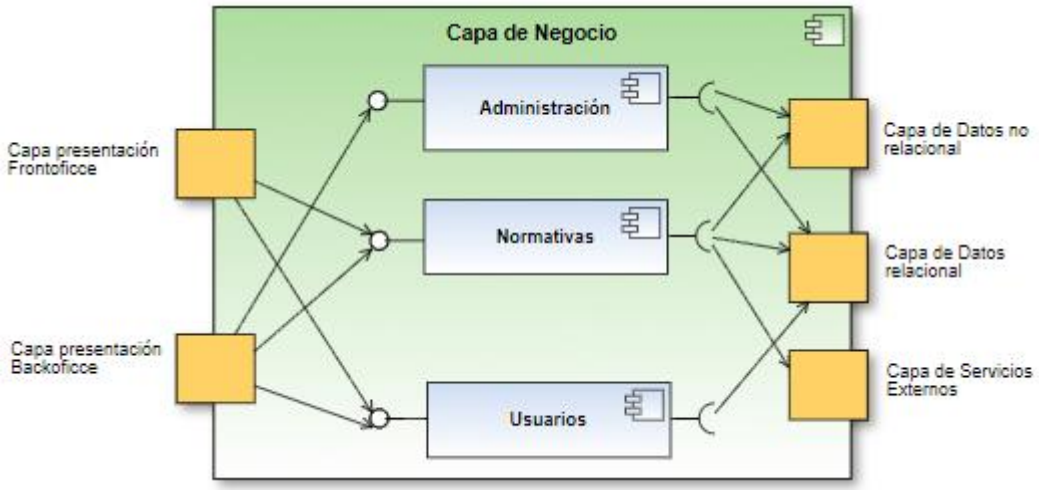

Figura 39 – Componentes Capa de Negocio

Administración: implementa toda la lógica de negocio referente a la administración de categorías, organizaciones y creación de nuevos tipos de normativas.

Normativas: implementa toda la lógica de negocio referente a la administración de transformaciones, creación de nuevas normativas y asignación de representaciones según el tipo de normativa correspondiente.

Usuarios: implementa toda la lógica de negocio referente a la administración de usuarios, tanto para el registro, como la validación de los mismos. Implementa la administración de tipos de usuarios y aprobación de normativas en organizaciones.

4.3.3. Vista de Distribución

En la Figura 40 se muestra la vista de distribución. Allí se observa principalmente los componentes que ejecutan dentro del servidor de aplicaciones y los que ejecutan afuera de él. Estos últimos corresponden a los componentes de persistencia.

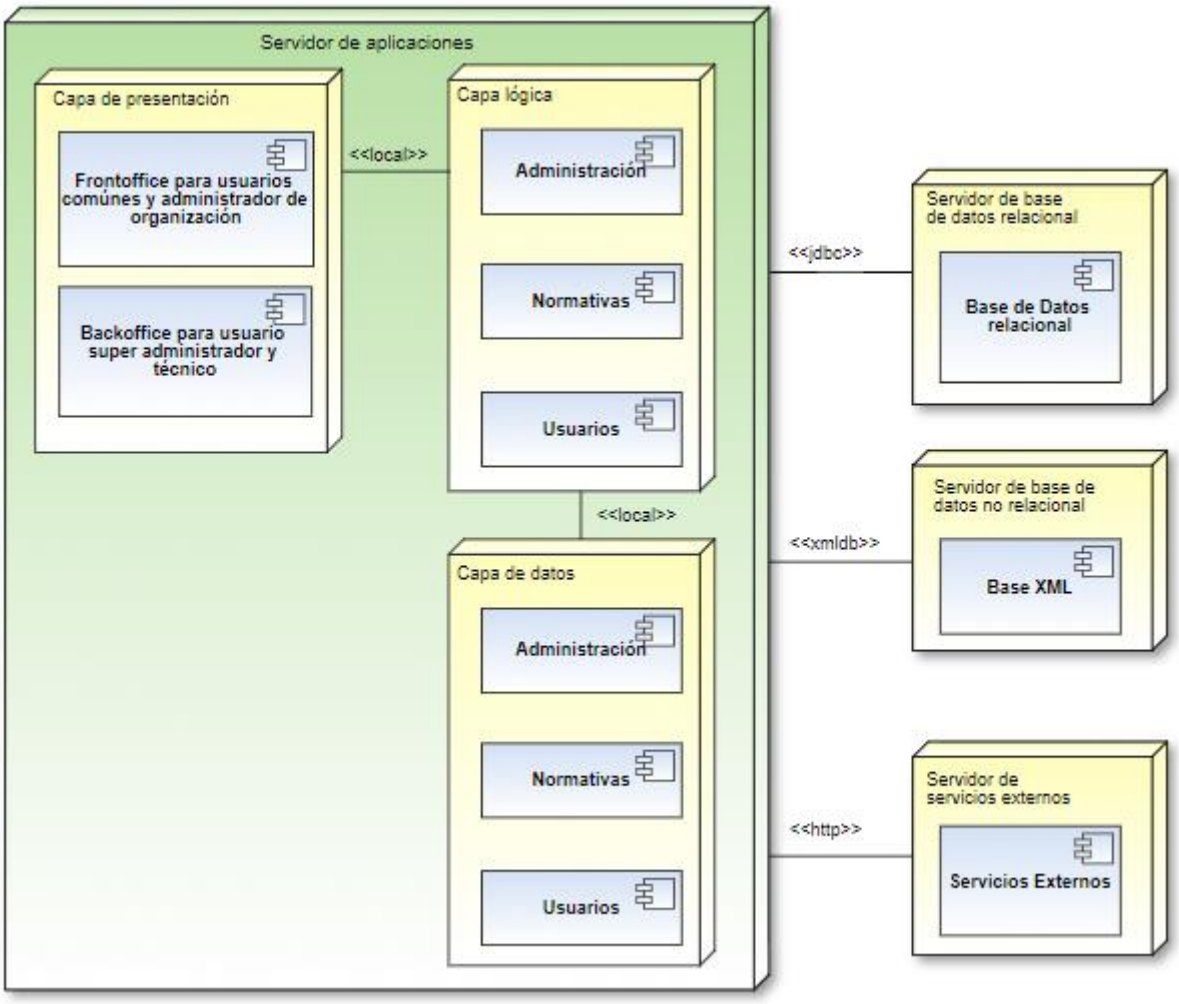

Figura 40 – Vista de Distribución

4.3.4. Vista de Implementación

En la Figura 41 se muestra los componentes en tiempo de ejecución que forman el sistema.

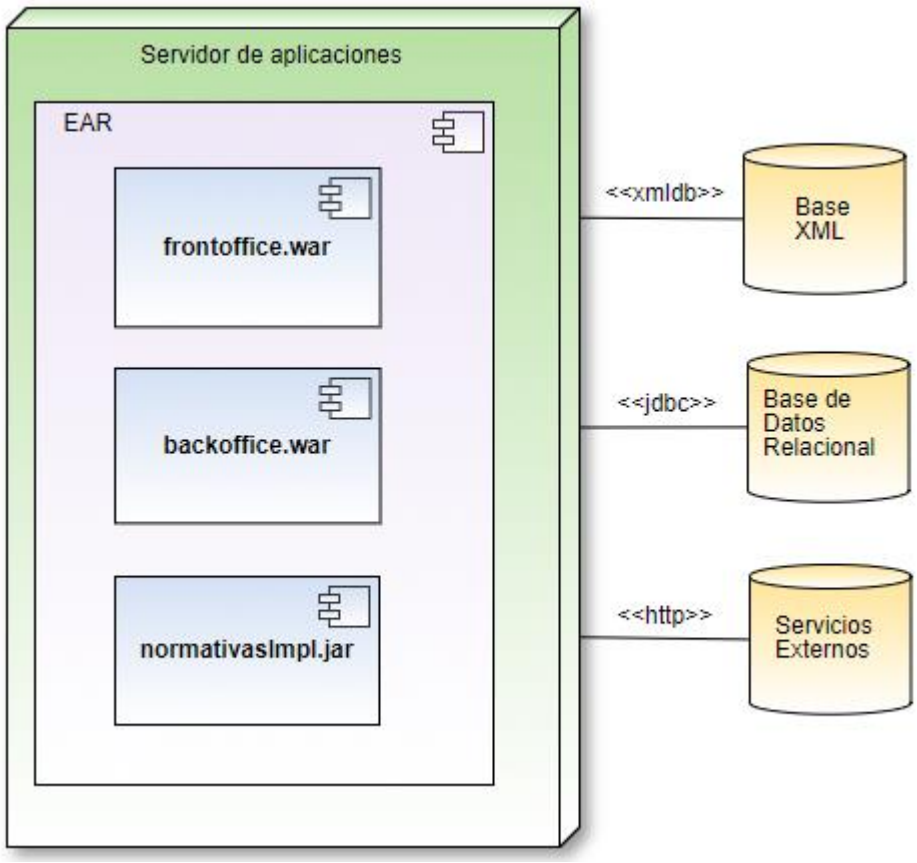

Figura 41 – Vista de Implementación

4.4. Decisiones tomadas

A continuación se comentan las decisiones tomadas al momento del diseño de la herramienta.

4.4.1. Frontoffice y Backoffice

Se decide dividir la capa de presentación en un Backoffice y un Frontoffice. La idea es brindar las funcionalidades relacionadas con la administración de la plataforma en el Backoffice y el resto en el Frontoffice.

Ambas capas se complementan teniendo así una interfaz para usuarios de tipos de usuario relacionados a la administración de la plataforma y otra interfaz para usuarios con permisos más restrictivos que van a poder gestionar normativas y será usada con mayor frecuencia.

4.4.2. Distintos tipos de almacenamiento

Se poseen estructuras de datos con diferentes características para ser almacenados en los distintos flujos de la herramienta. Se tienen las normativas expresadas en archivos XML principalmente de gran tamaño, información acerca de las entidades que maneja la herramienta (por ejemplo: datos de usuarios, organizaciones, normativas, tipos de normativas, etc.) y los archivos, también de gran tamaño, de las normativas en el lenguaje

común. Se decide almacenar los archivos XML en una base de datos XML debido a que su modelo de datos está orientado a este tipo de archivos.

Los datos de las diferentes entidades se almacenan en una base de datos relacional. Y los archivos en lenguaje común se almacenan consumiendo la API brindada por servicios externos.

5. Implementación

En esta sección se presentan detalles de implementación y se describen las tecnologías utilizadas para el desarrollo del proyecto, detallando las ventajas que brinda cada una de ellas y las facilidades que brindan en la implementación. Se describe la organización en módulos de la implementación, los prototipos realizados para tomar algunas decisiones técnicas y evaluar riesgos, y se detalla la implementación de las funcionalidades del sistema. Finalmente se describe el procesamiento de lenguajes de especificación de normativas y se realiza una sección de problemas encontrados y limitaciones que se tuvieron en el desarrollo de la implementación del sistema.

5.1. Descripción general

La herramienta se construye sobre la plataforma Java EE (*Java Platform, Enterprise Edition*). A partir de la arquitectura del sistema separada en capas, elaborada en la sección 4.3, se desarrolla la implementación de la herramienta Gestor de Normativas. En la Figura 42 se muestran las tecnologías que se utilizaron para implementar cada uno de los componentes.

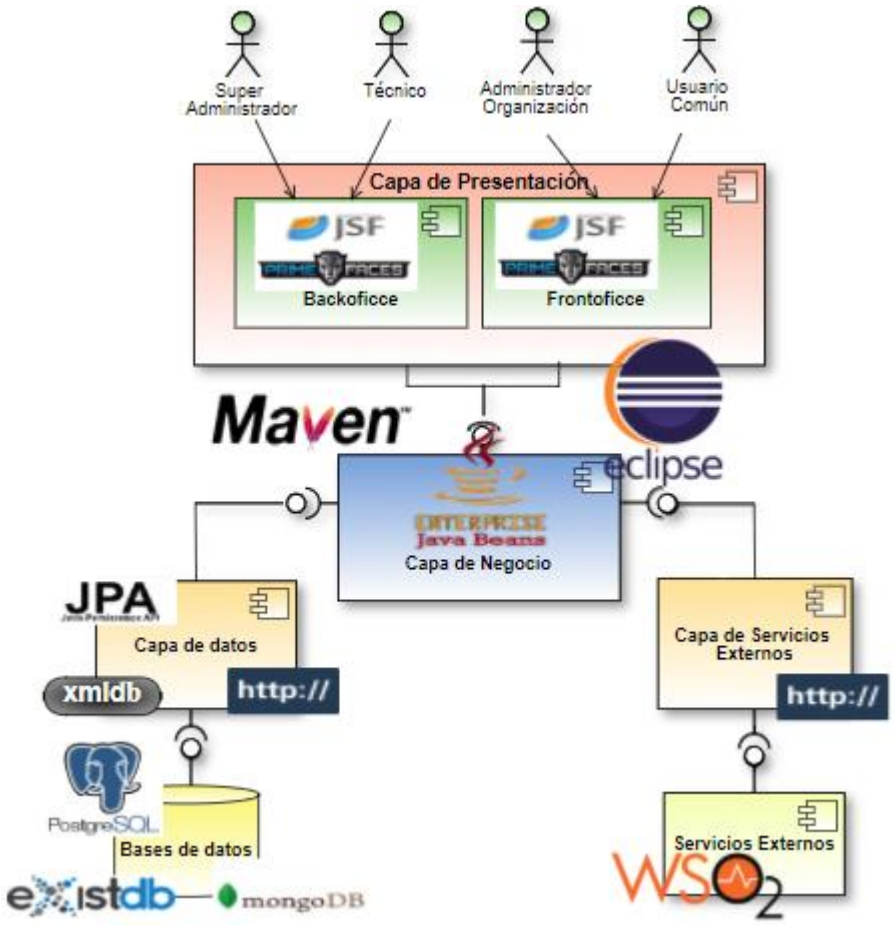

Figura 42 – Herramientas y Tecnologías

Con respecto a la capa de presentación, para implementar las páginas web con sus funcionalidades, tanto para el Frontoffice como para el Backoffice se utiliza JSF 2.2 con la extensión de Primefaces[3]. Las páginas (xhtml) se comunican con los *Managed Beans,* que ejecutan en el servidor de aplicaciones Wildfly[4] en su versión 9.0.2, donde se maneja la lógica de negocio y son estos últimos los que acceden a los EJBs.

Para el almacenamiento de todos los datos de representaciones, normativas, tipos de normativas, organizaciones y usuarios del sistema se utiliza una base de datos relacional PostgreSQL[5] 9.4. Para facilitar la comunicación con dicha base y la persistencia de los datos se utiliza JPA con la implementación de *hibernate*. Las imágenes de los usuarios del sistema son almacenadas en una base de datos no relacional: MongoDB[6] 3.0, con la extensión de Gridfs para facilitar el almacenamiento de archivos. Para guardar archivos XML (tipos de normativas, normativas, transformaciones XSLT) se utiliza una base de datos no relacional eXist 2.2 y su comunicación con xmldb. También las normativas (archivos XML) serán puestas en producción utilizando WSO2[7] que brinda un servidor XACML y consumiendo una API provista por el mismo.

5.2. Tecnologías y herramientas

En esta sección se detallan las tecnologías y herramientas utilizadas para construir la aplicación. Consta de una descripción, ventajas y desventajas, y su uso en la plataforma.

5.2.1. Plataforma base

Se describen las tecnologías y herramientas utilizadas como plataforma base para el desarrollo del Gestor de Normativas.

5.2.1.1. Java

Java es un lenguaje de programación que se ejecuta en múltiples plataformas. Con el avance de Java y su popularidad, se han creado múltiples configuraciones para adaptarse a varios tipos de plataformas. Por ejemplo: J2EE para aplicaciones empresariales, J2ME para aplicaciones móviles [46].

Java se puede ampliar fácilmente ya que se basa en un modelo orientado a objetos. Además se caracteriza por ser independiente de la plataforma debido a que java no se compila en la plataforma de la máquina específica [46].

Para la implementación de la herramienta se utiliza el lenguaje de programación Java, en particular, la configuración J2EE para aplicaciones java empresariales.

[3] https://www.primefaces.org/
[4] http://wildfly.org/
[5] https://www.postgresql.org/
[6] https://www.mongodb.com
[7] http://wso2.com/

5.2.1.2. Eclipse

Eclipse proporciona IDEs (Entornos de Desarrollos Integrados) y plataformas para una gran variedad de lenguajes y arquitecturas [47].

Eclipse brinda una gran colección de herramientas que se pueden instalar fácilmente en el IDE, como por ejemplo: herramientas para modelar, construir la GUI, dibujar gráficas, realizar reportes, entre otros [47].

Para el desarrollo de la herramienta y la construcción del proyecto se utiliza el IDE Eclipse, en particular la versión Mars. Esta versión incluye nuevas características como la visión de proyectos anidados, la capacidad para personalizar perspectivas y herramientas integradas para la construcción [47].

5.2.1.3. Maven

Maven es una herramienta de comprensión y gestión de proyectos de software. Está basado en el concepto de un modelo de objetos de proyecto (POM). Maven puede gestionar la compilación, la elaboración de informes y la documentación de un proyecto a partir de una información central [48].

Esta herramienta facilita la separación del proyecto en módulos y el manejo de librerías, ya que no es necesario agregarlas manualmente en la aplicación. Basta con definir dependencias y las librerías se descargan de un repositorio Maven.

Maven se utiliza como base para la construcción del proyecto en módulos y para la sencilla y útil descarga de librerías (declarándolas en el archivo POM).

5.2.1.4. Wildfly

Wildfly posee versiones de una serie de ofertas de servidores de aplicaciones de código abierto de JBoss (servidor de aplicaciones Java Empresarial implementado en Java puro) [49].

Se utiliza la versión 9.0.2 de Wildfly que incluye facilidades para el manejo de aplicaciones java empresarial, como la utilización de EJB. Posee la dificultad de consumir gran cantidad de memoria a diferencia de un servidor *tomcat*, por ejemplo.

Wildfly es el servidor de aplicaciones donde se realiza el despliegue de la plataforma implementada.

5.2.2. Capa de presentación

Se detallan las tecnologías y herramientas que se utilizan para implementar la capa de presentación.

5.2.2.1. JSF

JSF es una tecnología que simplifica la creación de interfaces de usuario para aplicaciones Java Web [50].

Se utiliza la versión 2.2 de JSF compatible con la extensión de Primefaces. JSF resulta efectivo para la construcción de páginas web y la comunicación con el servidor de aplicaciones.

JSF es empleado en el Gestor de Normativas para la construcción de páginas web y su comunicación con el servidor (*Java Beans*): para los proyectos web normativas-frontoffice y normativas-backoffiece. Los *Java Beans* son componentes de software que facilitan la reutilización de código [51].

5.2.2.2. Primefaces

Primefaces es un *framework* de interfaz de usuario JSF que puede utilizarse para desarrollar rápidamente aplicaciones sofisticadas para una empresa [52].

Se emplea la versión 6.0 de Primefaces para ambos proyectos web. Permite adicionar elementos a la web y utilizar *ajax*: facilita la creación de las aplicaciones web. Aunque puede generar conflictos con librerías de *JQuery* y *Javascript*.

Primefaces es utilizado en las páginas web en conjunto con JSF para facilitar el uso de *ajax*.

5.2.3. Capa lógica

En esta sección se describen las tecnologías que se utilizan en la capa lógica de la herramienta.

5.2.3.1. JAXB

Java Architecture for XML Binding (JAXB) es una API Java que puede facilitar el acceso a documentos XML desde aplicaciones escritas en el lenguaje de programación Java [53].

JAXB proporciona dos características principales con la utilización de la herramienta XJC: la capacidad de serializar las referencias de objetos Java a XML y la inversa, es decir, XML en objetos Java.

En la aplicación, JAXB es utilizado para transformar los archivos XSD (*XML Schema Definition*) de los tipos de normativas en clases Java y poder así luego validarlas de acuerdo a la normativa que se quiera crear.

5.2.3.2. WSO2

WSO2 *Identity Server* (WSO2 IS) es un servidor de gestión de identidades que facilita la seguridad al conectar y administrar varias identidades a través de diferentes aplicaciones [54].

WSO2 IS permite a los arquitectos y desarrolladores de empresas mejorar la experiencia del cliente a través de un entorno de inicio de sesión único seguro [54].

En la aplicación, WSO2 IS se utiliza para el almacenamiento de normativas en el lenguaje XACML. Aquí son guardadas las normativas cuando son puestas en producción por un administrador de la plataforma.

5.2.4. Capa de acceso a datos

En esta sección se describen las tecnologías que se utilizan en la capa de acceso a datos.

5.2.4.1. JPA (*Java Persistence* API)

JPA proporciona un modelo de persistencia POJO (*Plain Old Java Object*: clases simples que no dependen de un framework en especial) para la asignación de objetos relacionales [55].

JPA provee a los desarrolladores de un recurso de mapeo objeto-relacional para administrar datos relacionales en aplicaciones Java. La herramienta brinda alto nivel de abstracción para trabajar con objetos y clases en vez de directamente con las tablas (facilitando también las operaciones con la base de datos) [56].

En la aplicación, JPA se utiliza para la persistencia y consultas de los datos relacionados con normativas, tipos de normativas, representaciones, organizaciones, categorías y usuarios en la base de datos postgreSQL.

5.2.4.2. PostgreSQL

PostgreSQL es un sistema de base de datos relacional de código abierto [57].

Lleva más de 15 años de desarrollo activo y posee una arquitectura probada que le ha valido una sólida reputación de fiabilidad, integridad de datos y corrección [57].

PostgreSQL permite la creación de la base de datos relacional de la aplicación.

5.2.4.3. eXist-db

eXist-db es una base de datos de documentos NoSQL y una plataforma de aplicaciones. Todas las versiones de eXist-db son de código abierto y pueden utilizarse en aplicaciones académicas, no comerciales y comerciales [18].

eXist-db es tanto para consultar como para guardar los documentos y ayuda a no sobrecargar el servidor de aplicaciones.

Se utiliza eXist-db en su versión 2.2 como base de datos no relacional para almacenar archivos XML utilizados en la aplicación: normativas, tipos de normativas y transformaciones XSLT.

5.2.4.4. MongoDB

MongoDB es una base de datos no relacional orientada a documentos que permite realizar búsquedas y guardar estructuras de datos en formato JSON [58].

Se utiliza en la aplicación para guardar las imágenes de los usuarios con la extensión GridFS. GridFS es una especificación implementada por controladores MongoDB que gestiona archivos grandes y sus metadatos asociados como un grupo de archivos pequeños. Cuando se consulta un archivo grande, GridFS vuelve a ensamblar automáticamente los archivos más pequeños en el archivo grande original [59].

5.3. Organización en módulos

Eclipse Mars es el IDE seleccionado para el desarrollo de la herramienta Gestor de Normativas. Se utiliza Maven 3.3.3 como herramienta para la gestión y construcción de proyectos organizados en módulos. El Gestor de Normativas consta de cuatro módulos: normativas-api, normativas-impl, normativas-backoffice y normativas-frontoffice.

El módulo “normativas-api” es el encargado del almacenamiento y gestión de las entidades que se mapean a la base de datos relacional postgreSQL, y además, contiene las interfaces con las firmas de los métodos utilizados por los EJBs (*Enterprise JavaBeans*).

El módulo normativas-impl es donde se encuentran las implementaciones de los métodos de los EJBs que se utilizan para implementar las operaciones que acceden a la base de datos postgreSQL, las que consumen servicios externos a partir de las APIs de WSO2 y las que acceden a la base de datos no relacionales: base XML eXist, y de archivos MongoDB.

El módulo normativas-backoffice contiene las páginas JSF y los *Beans* para implementar las funcionalidades de administración de la plataforma.

Por último, el módulo normativas-frontoffice posee las páginas JSF y *Beans* correspondientes para implementar las funcionalidades de los usuarios del sistema: usuario común y administrador de organización.

5.4. Prototipos

Previo a la implementación de los casos de uso de la plataforma se desarrollaron tres prototipos en base a riesgos previamente identificados. Los riesgos hacen referencia a la integración de diferentes herramientas para llevar a cabo el desarrollo del Gestor de Normativas: utilización de diferentes tipos de almacenamiento para la información, consumo de servicios externos, e implementación de la extensibilidad con respecto a los Tipos de Normativa. Los prototipos implementados para mitigar los riesgos fueron: conexión con la base de datos no relacional eXist, comunicación con servicios externos de WSO2, e integración entre la herramienta y la librería de JAXB. Cada prototipo constó de un objetivo, su implementación y el resultado obtenido.

5.4.1. Conexión con eXist-db

Para mejorar el almacenamiento de archivos XML se decidió que los tipos de normativas y las normativas en formato XML se almacenen en una base de datos no relacional: eXist-db. El objetivo de este prototipo fue implementar un cliente Java simple que logre conectarse con la base eXist, y que sea capaz de guardar y obtener un archivo XML de la misma.

Para implementar el prototipo, se creó un proyecto Maven simple en java. Se creó un archivo con un esquema XML de ejemplo y se guardó en la base eXist con un determinado nombre. Se encontraron problemas de compatibilidad con las librerías “xalan” y “xerces” entre las obtenidas del repositorio de Maven y las disponibles en el servidor de aplicaciones Wildfly donde se estaba ejecutando ese proyecto de ejemplo creado. Una vez solucionado excluyendo las librerías del servidor de aplicaciones, se guardó el XML. Luego se obtuvo el archivo con el nombre guardado y se imprimió en pantalla.

El resultado de la implementación del prototipo fue exitoso. Se logró obtener en salida estándar el archivo que fue seleccionado por el usuario, guardado en eXist y finalmente obtenido con el nombre con el que fue almacenado.

5.4.2. Comunicación con WSO2

Una vez aprobadas las normativas por todas las organizaciones involucradas, se decide utilizar WSO2 y su arquitectura provista para su almacenamiento en el lenguaje común XACML. El objetivo de este prototipo fue realizar una prueba de comunicación entre un cliente Java simple y el Servidor de WSO2 almacenando archivos expresados en XACML.

Para implementar este prototipo, se creó un proyecto Maven simple en Java que ejecuta sobre un servidor de aplicaciones Wildfly 9.0.2. Se importaron del repositorio de Maven las librerías necesarias para consumir la API provista por WSO2 y se levantó el *Identity Server* de WSO2. Luego se procedió a ejecutar el código correspondiente a la conexión con el *Identity Server* y al almacenamiento y recuperación de un archivo de prueba con la sintaxis adecuada que respeta la estructura de un esquema XACML. Se encontraron conflictos de certificados que se describen en la sección 5.6.1.

Una vez realizada esta prueba se obtuvieron resultados con éxito. Mediante la consola de administración que brinda WSO2 se logró visualizar el archivo XACML almacenado durante la ejecución del prototipo. Es así que se procedió a utilizarlo para almacenar las normativas (en la puesta en producción).

5.4.3. Utilización de JAXB

Uno de los puntos importantes de la herramienta es la creación de nuevos tipos de normativas (en Backoffice) y su validación con sus correspondientes normativas (en Frontoffice). El objetivo de este prototipo fue implementar un cliente Java simple que utilice la herramienta xjc de JAXB, que permita la conversión de un archivo XSD a clases Java y poder “compilar” archivos XML con esas clases Java.

Para implementar el prototipo, de la librería JAXB se probó el unmarshaller (para serializar objetos Java a XML y a la inversa) y la herramienta xjc. En primer lugar, se creó un proyecto Maven simple en Java y con la utilización de dependencias Maven se descargaron las librerías correspondientes del repositorio. Se tomó un ejemplo de un archivo expresado en XML del tipo de normativa WSLA y, a partir de su XSD (archivo subido por el usuario del prototipo), utilizando xjc, se generaron las clases Java mapeadas que corresponden. Luego, utilizando un contexto de JAXB y la clase Unmarshaller, se realizó el proceso inverso para chequear si las clases Java "compilaban" con un ejemplo simple de normativa de WSLA, también seleccionado por el usuario del prototipo.

Se obtuvieron con éxito los resultados de este prototipo. Se logró, en tiempo de ejecución, compilar una normativa con alguno de los Tipos de Normativa previamente almacenados (guardados con clases Java en la herramienta). Por lo tanto se decidió utilizar JAXB para implementar la gestión de tipos de normativa.

5.5. Implementación de funcionalidades

A continuación se describe cómo se implementaron las funcionalidades brindadas en el Gestor de Normativas.

5.5.1. Gestión de Organizaciones, Categorías y Representaciones

Para la gestión de Organizaciones (p. ej.: ANTEL, DNIC), Categorías (p. ej.: SLA, QoS) y Representaciones (p. ej.: XACML, HTML) se utiliza JPA en los EJBs de la aplicación (persistiendo, modificando o eliminando la entidad). Esta funcionalidad se encuentra disponible únicamente desde las vistas del proyecto web normativas-backoffice y sólo podrán acceder los usuarios Administradores de la herramienta.

5.5.2. Administración de usuarios

Los usuarios pueden registrarse únicamente desde las vistas de proyecto web normativas-frontoffice. Entre los datos solicitados se encuentra la Organización a la que pertenece y una imagen. La imagen es almacenada en MongoDB utilizando la extensión de GridFS.

Una vez registrados pasan a ser usuarios del Gestor de Normativas con el nivel de permisos más restrictivo: tipo de usuario Usuario Común. Desde el Backoffice existe la posibilidad de poder cambiar el tipo de usuario para cualquiera de los existentes y a cualquier usuario de cualquier organización (funcionalidad disponible sólo para el administrador de la herramienta). Desde el Frontoffice, el Administrador de Organización puede manipular los tipos de usuario Usuario Común y Administrador de Organización de los usuarios pertenecientes a su organización.

Los usuarios tienen diferentes permisos sobre las funcionalidades de la herramienta dependiendo del tipo de usuario al que pertenezcan.

5.5.3. Nuevo Tipo de Normativa

Los tipos de normativas (como por ejemplo WSLA) tienen su tabla correspondiente en la base de datos PostgreSQL. Es decir, que cuando se da de alta un tipo de normativa desde el Backoffice, ya sea por algún administrador de la herramienta o algún técnico, se guardan sus datos en la base de datos. Pero también, cada tipo de normativa posee un XSD correspondiente a la estructura y elementos que soporta. Dicho archivo XSD es solicitado al usuario al momento de crear un nuevo tipo de normativa. La idea es que se puedan "compilar" las normativas en base al XSD del Tipo de Normativa al que pertenece.

Para poder implementar esta compilación, se generan las clases Java a partir del XSD utilizando JAXB. Más precisamente con la utilización de la herramienta xjc, en tiempo de ejecución, para convertir el esquema XML en representaciones de clases. Estas clases Java generadas son almacenadas mismo en el servidor de aplicaciones Wildfly 9.0.2.

Esta funcionalidad brinda una cualidad sobre los tipos de normativas y la relación con las representaciones del sistema. En tiempo de ejecución se permite definir qué representaciones va a soportar. Por lo que se le facilita al usuario la posibilidad de elegir si soporta las diferentes representaciones existentes en el sistema. En caso de seleccionar afirmativamente una Representación (por ejemplo XACML), se le solicita que seleccione desde cuál representación previa se debe realizar la transformación (o si es desde la normativa especificada en el lenguaje original) y el tipo de la transformación. Esta representación es elegida por el usuario entre las soportadas por el sistema: XSLT o FreeMarker. Y finalmente, una vez seleccionado el tipo de transformación, se solicita al usuario subir el archivo correspondiente.

Si el archivo de transformación corresponde a un *template* FreeMarker, se almacena como *String* (tipo LOB) en la base de datos PostgreSQL, en la tabla de Tipos de Normativas. Para el caso en que el archivo sea un XSLT, es almacenado en la base de datos XML eXist para su posterior utilización.

5.5.4. Nueva Normativa

La página web correspondiente a dar de alta una nueva normativa comienza con dos pestañas visibles. Una pestaña se utilizará para confirmar los datos definitivos de la Normativa. En la otra se solicitan los datos de la Normativa, entre ellos, subir el archivo correspondiente. El sistema se encargará de identificar el tipo de normativa al que corresponde la normativa cargada. Es decir, que se "compila" la normativa con los tipos de normativas existentes en el sistema. Para esto se utiliza el Unmarshalling que es provista también por la API de JAXB. Esta herramienta, permite transformar un esquema XML en clases Java, utilizada de manera de identificar el tipo de normativa que se corresponde con la normativa. O sea, a partir de las clases Java generadas con xjc que se habían guardado en el servidor al crear el tipo de normativa, se prueba el Unmarshal con el archivo subido correspondiente a la normativa. Aquí fue necesario también compilar (librería *javax.tools.JavaCompiler*) los .java para generar los .class correspondientes en tiempo de ejecución.

Una vez “compilada” la normativa se identifica el tipo de normativa al que pertenece y se le solicita al usuario, en caso de ser necesario, que seleccione el resto de las organizaciones participantes (ya que es variable y particular para cada tipo de normativa). Además se agregan las pestañas correspondientes a cada representación que fue solicitada al momento de crear el tipo de normativa. Al seleccionar cada pestaña se realiza la transformación y se muestra el resultado. Para dicha transformación se toma en cuenta el archivo desde cual se debe transformar (que fue elegido al crear el tipo de Normativa) y el tipo de transformación. En caso de que el tipo de transformación sea un FreeMarker, se obtiene la plantilla desde la base de datos PostgreSQL y se realiza la transformación. Y para el caso de que sea un XSLT, se accede a la base eXist para obtener el XML correspondiente y se realiza la transformación.

5.5.5. Utilización de SVG

Las normativas se pueden transformar a diferentes lenguajes para obtener distintas visualizaciones de la misma como se mencionó en la sección 5.5.4. Entre ellas se encuentra la visualización final, expresada en XACML, pero también se permite transformaciones a archivos HTML como caso particular. Es decir que es posible transformar la normativa a un HTML (previamente definido en el tipo de normativa) y mediante la utilización de SVG poder visualizar la página generada correspondiente en una de las pestañas mencionadas en la sección anterior. Es así que se brinda esta posibilidad a los usuarios para poder visualizar de manera más sencilla y amigable las normativas y todos sus componentes principales.

5.5.6. Estados de una Normativa

Las normativas deben ser aprobadas por todas las organizaciones participantes. Si un usuario común de una organización crea una nueva normativa, ésta no queda aprobada por dicha organización. En caso de ser creada por un Administrador de Organización, la normativa queda como aprobada por esa organización.

Los estados correspondientes al ciclo de vida de una Normativa son: No Aprobada, Semi Aprobada, Aprobada y Deployada que fueron detallados en la sección 4.2.3.

En el Frontoffice de la aplicación se brinda una funcionalidad para que los usuarios puedan visualizar las Normativas en las que su organización participa y además poder Aprobarlas. En particular, la aprobación sólo estará disponible para los usuarios Administrador de Organización.

El sistema de aprobaciones se efectúa guardando los datos del estado de una normativa en la base de datos PostgreSQL junto con la normativa a la que corresponde y un conjunto de organizaciones faltantes de aprobar.

5.5.7. Puesta en producción de la Normativa

El Backoffice brinda una funcionalidad para los administradores de la plataforma: puesta en producción de una normativa. Consiste en “poner en producción” una Normativa que ya fue aprobada por todas las organizaciones participantes. Aquí se consume una API de WSO2 para almacenar la normativa en un servidor de WSO2. La normativa que se despliega en el

servidor es especificada en el lenguaje de políticas XACML. La normativa se puede visualizar mediante una interfaz brindada por el servidor.

Desde el Gestor de Normativas se consume una API del WSO2 IS que agrega las políticas al *Policy Administration Point* (PAP). Una vez allí, con la consola de administración que brinda el servidor de WSO2, se pueden publicar las políticas (normativas) en el *Policy Decision Point* (PDP). En caso de que se necesite modificar una normativa que se encuentra en producción, se debe crear otra normativa correspondiente al mismo tipo de normativa, con distinto nombre y las modificaciones que se deseen. Esto hace posible que la normativa en el PAP se visualice como una nueva versión. La anterior normativa no se pierde, al publicar en el PDP existe la posibilidad de seleccionar cualquiera de las versiones del PAP.

5.5.8. Despliegue en la nube

Para desplegar la herramienta en la nube se utilizan los servicios brindados por Amazon Web Services[8] (AWS). En particular se utiliza *Elastic Compute Cloud* (EC2) para poder construir una máquina virtual en la nube y poder tener el Gestor de Normativas allí.

Se genera un dominio para el Frontoffice y Backoffice de la herramienta. Además se genera otro dominio para acceder por https a la interfaz de WSO2 IS y poder visualizar las normativas en producción.

5.6. Procesamiento de lenguajes de especificación de normativas

Las especificaciones de los diferentes tipos de normativas (WSLA y WSQDL, entre otros) cumplen un determinado esquema XML. Para ejecutar la funcionalidad correspondiente al alta de normativas de la herramienta, fue necesario seguir el flujo de transformaciones de una normativa y visualizar las representaciones hasta llegar a la del lenguaje de puesta en producción de políticas XACML.

Una de las representaciones fue la de mostrar gráficamente la normativa con la utilización de SVG. Para ello fue necesario procesar las normativas en su lenguaje de especificación original para determinar cómo extraer organizaciones participantes, y todos los elementos que componen el esquema XML. Este procesamiento permitió construir los archivos XSLT y *template Freemarker* para luego generar las transformaciones en la herramienta.

Otra representación fue la de mostrar en un HTML la información destacada de una normativa. Aquí se logró construir un XSLT que procesara los elementos que se consideraban destacados de una normativa y se mostrara como un HTML. Cabe recalcar que el XSLT que muestra el HTML de los elementos destacados de una normativa de WSLA es distinto al XSLT que muestra el HTML de los elementos destacados de una normativa de WSQDL (el procesamiento del esquema XML es distinto).

[8] https://aws.amazon.com

Es así que, la herramienta permite que las normativas puedan tener múltiples representaciones, siempre que se construyan los archivos que transformen a esas representaciones.

5.7. Problemas y limitaciones

A continuación se comentan los problemas encontrados y limitaciones en el desarrollo de la herramienta.

5.7.1. WSO2 almacenes y certificados

Como parte de la implementación se consume una API provista por WSO2 IS, que se utiliza al momento de poner en producción una Normativa para poder guardar el XACML en un servidor. Al intentar consumir la API, ya sea para obtener una normativa o para guardarla, se encontraron problemas de seguridad por temas referentes a certificados digitales.

En primer lugar se intentó detectar los certificados necesarios utilizados en el almacén que tiene el servidor de WSO2, encontrarlos e instalarlos en el almacén de certificados de Java. Al pasar un certificado se observó que dependía de otro también almacenado en el almacén del servidor de WSO2, y así sucedía cada vez que se instalaba un nuevo certificado. La solución implementada para resolver este problema fue agregar el almacén de certificados, que ya estaba incluido en el servidor de WSO2, a la máquina virtual de Java. Cabe destacar que esta solución no sería adecuada para un entorno de producción.

5.7.2. XSD: dinamismo para Tipos de Normativas

Como se explicó en la sección 5.5.3 de la implementación, se utilizan los archivos XSD para validar las normativas con los tipos de normativas. La utilización de Unmarshaller necesita que los archivos java (obtenidos del mapeo del archivo XML) estuvieran compilados, es decir, que no se realiza dicha validación directo con los XSD. Como el XSD se selecciona al momento de crear el tipo de Normativa y la validación se realiza al momento de crear una normativa, es que se necesitan guardar los archivos compilados en algún sitio. Se decide que estos archivos sean almacenados organizadamente en el servidor de aplicaciones Wildfly que utiliza la herramienta.

La limitante, es que si se desea trabajar con muchos tipos de normativas, el servidor se saturaría de archivos almacenados. Igualmente la idea es generalizar los tipos de normativas y que se reutilicen: como mucho 2 o 3 por categoría investigada (QoS, SLA, etc).

5.7.3. Normativas especificadas en XML

Las normativas pueden ser especificadas en lenguajes (desde lenguaje natural hasta archivos XML), o como diagramas conceptuales, entre otros. Debido a la decisión de utilizar como uno de los medios de transformación XSLT, es que las especificaciones de las normativas están limitadas a ser expresadas en XML. Por lo tanto, el sistema se encuentra limitado, ya que para hacer uso del mismo, los usuarios deben utilizar las normativas especificadas en XML.

5.7.4. Bluemix y contenedores Docker

Se intenta utilizar Bluemix como herramienta para desplegar la herramienta en la nube y utilizar el beneficio que tiene con contenedores Docker. El desarrollo de la herramienta consta de un servidor de aplicaciones Wildfly, un servidor MongoDB, un servidor eXist-db, un servidor de WSO2 y un servidor postgreSQL. La gran limitación que presenta Bluemix es que no es posible configurar un servidor exist-db para guardar los archivos XML. Por lo tanto, como segunda opción se intenta crear una máquina virtual en Bluemix con todos los componentes que conforman la herramienta. No se encuentra nada específico que permitiera hacer esto. Solo se encuentra una guía que lleva a crear servidores virtuales pero al momento de crearlo aparece un mensaje de error. Lo que sucede es que Bluemix como herramienta PaaS (*Platform as a Service*) de IBM, no tiene en sus orígenes máquinas virtuales. Fueron agregadas relativamente hace poco, pero en algunas regiones y seguramente no disponible para cuentas de estudiantes.

Finalmente se logra desplegar la herramienta en la nube con los servicios disponibles de AWS.

6. Caso de estudio

En esta sección se detalla el caso de estudio para el presente proyecto. El objetivo es describir una posible realidad en la que sea aplicable la administración de Normativas.

6.1. Descripción del problema

La empresa MoviCel S.A. domina actualmente el mercado de telefonía móvil (celulares) en el Uruguay en muchos rubros. Su directiva ha incursionado en varias áreas con éxito a lo largo de su historia. Desde su concepción como una pequeña empresa de ventas de artefactos de la comunicación, su dueño, el señor Norman Henderson, ha aspirado expandir su empresa. Con el correr de los años, Henderson innovó sus productos adicionando la más alta tecnología en el momento para brindar mantenimiento y desarrollo técnico en la empresa.

El éxito de la empresa ha sido plasmado más que nada en el rubro de venta de celulares a todo el público uruguayo. El objetivo de las ventas siempre se ha enfocado en incluir con el aparato (celular) un contrato que incluya paquetes de datos y minutos gratis. Esto provoca que los clientes no solo compren en el momento, sino que continúan pagando un contrato, por ejemplo a dos años, y durante ese tiempo sigan perteneciendo a la clientela de MoviCel. Pero además, otro punto fuerte de ingreso de la empresa, es la venta de celulares en modalidad prepago. Es decir, clientes que sólo desean comprar el celular sin necesidad de incluir un contrato.

Los contratos requieren una serie de procedimientos legales, los cuales comprometen al cliente a pagar durante un tiempo determinado, el monto acordado, a cambio de las prestaciones de la empresa. Por lo tanto, se solicitan los datos del cliente para registrarlos en la base de datos de clientes de MoviCel. Sin embargo, como la modalidad prepago es simplemente una compra en el momento, durante mucho tiempo no se han solicitado los datos del cliente. Es así que, MoviCel desconoce los datos de los clientes prepagos que posee.

A su vez, mediante reportes anuales generados por la empresa para analizar las ganancias, se constata que los clientes que les generan mayor ganancia son aquellos que poseen un contrato. Por lo que surge la necesidad de convertir los clientes prepagos a clientes contractuales.

Entonces, por un lado se necesita la identificación de los clientes prepagos, para luego poder ofrecerles un contrato acorde a necesidades y capacidades socioeconómicas.

La Dirección Nacional de Identificación Civil (DNIC) posee información centralizada de todos los ciudadanos del Uruguay. Entre otros, posee los datos personales tales como: nombres, apellidos, tipo de documento y número de documento, sexo y fecha de nacimiento. Actualmente la DNIC brinda una gran variedad de servicios. En particular, expone un servicio llamado “Servicio básico de información” [60]. El servicio recibe como parámetro de

entrada el documento de identidad y devuelve, validando dicho documento, los datos filiatorios tales como primer nombre, segundo nombre, primero apellido, segundo apellido, sexo y fecha de nacimiento, entre otros.

El Ministerio de Desarrollo Social (MIDES) de Uruguay es el responsable de contribuir al desarrollo de escenarios de participación social para fortalecer a la ciudadanía uruguaya. El MIDES ofrece una gran cantidad de servicios. En particular existe un servicio relacionado a la vulnerabilidad social de las personas. A partir del documento de identidad, el servicio llamado “esVulnerable” devuelve si dicho documento corresponde a una persona socialmente vulnerable [61].

La Figura 43 muestra una red de empresas comunicándose mediante servicios a través de una plataforma de integración. En particular, en la figura se visualizan las organizaciones: MIDES, DNIC y Movicel, interactuando con la plataforma.

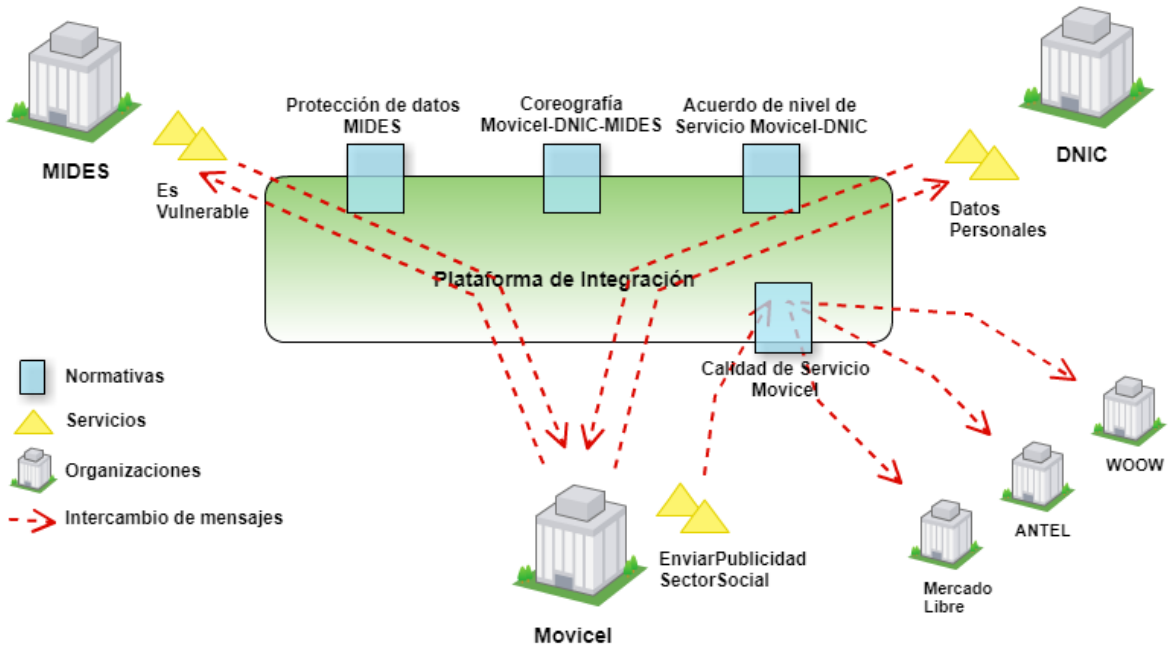

Figura 43 – Caso de estudio – Red de comunicación de organizaciones

Nos encontramos en un contexto de una plataforma de integración que expone comunicación entre las organizaciones públicas y privadas en el área de telecomunicaciones y que además se pueden consumir y/o publicar servicios en otras áreas.

Para cumplir con los requerimientos, Movicel debe consumir, en primer lugar el servicio “Servicio básico de información” de la DNIC y registrar al cliente en su propia base de datos. En segundo lugar, Movicel debe consumir el servicio “esVulnerable” brindado por el MIDES para determinar si el cliente es socialmente vulnerable y guardar la información también en su sistema. Una vez recabados los datos necesarios, Movicel se encuentra en condiciones de poder cumplir con su requerimiento inicial de realizar el envío publicitario a su clientela prepaga ofreciendo contratos acordes a su sector social. Finalmente Movicel expone el servicio “EnviarPublicidadSectorSocial” que dado un sector social de la población, envía por

SMS un mensaje de publicidad a los clientes de Movicel (con contrato o prepago) y que pertenecen a dicho sector social.

Para llevar a cabo toda la comunicación e intercambio de información se debe contar con normativas previamente definidas. Una normativa correspondiente a Acuerdo de Nivel de Servicio (SLA) para establecer un acuerdo en lo referido a cantidad de invocaciones y tiempo de respuesta al momento en que Movicel consuma el servicio de la DNIC. Otra normativa necesaria es una que corresponda a la categoría Protección de Datos Personales en la que se establece el consentimiento del ciudadano para que el MIDES pueda brindar la información de si es o no vulnerable dicho ciudadano a Movicel. Debe existir también una normativa, correspondiente a intercambio de mensajes entre organizaciones, para establecer las reglas de comunicación en el intercambio de mensajes entre el Movicel, DNIC y MIDES. Y por último, una normativa de Calidad de Servicio donde Movilcel establezca las políticas de calidad del servicio “EnviarPublicidadSectorSocial” que expone.

6.2. Especificación de las normativas

Como se mencionó en la sección 6.1, para cumplir con los requerimientos planteados, se debe definir una serie de normativas correspondientes a diferentes categorías.

6.2.1. SLA entre Movicel y DNIC

Mocivel debe consumir el “Servicio básico de información” provisto por la DNIC. Este servicio es altamente demandado por la industria, muchas organizaciones desean acceder a la información de las personas del país. Algunas organizaciones quizás no necesiten invocar el servicio con la misma frecuencia que Movicel ya que no trabajan con clientes nuevos todos los días. Por lo tanto, Movicel y DNIC deben establecer un acuerdo en el nivel del servicio provisto. Para esto, se necesita una normativa correspondiente a un acuerdo de nivel de servicio (SLA) donde se establezcan la cantidad de invocaciones al mes permitidas y un máximo tiempo de respuesta por cada invocación. La normativa es expresada utilizando el lenguaje WSLA.

A continuación en la Figura 44 se muestra la normativa correspondiente a dicho acuerdo.

```
<?xml version="1.0"?>
<SLA xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Parties>
        <ServiceProvider name="Movicel">
            ...
        </ServiceProvider>
        <ServiceConsumer name="DNIC">
            ...
        </ServiceConsumer>
        <SupportingParty name="YMeasurements" role="MeasurementService">
            ...
        </SupportingParty>
        <SupportingParty name="ZAuditing" role="ConditionEvaluationService">
            ...
        </SupportingParty>
    </Parties>
    <ServiceDefinition name="DemoService">
        <Operation name="ObtenerDatosPersona" xsi:type="WSDLSOAPOperationDescriptionType">
            <SLAParameter name="Cant_Invovaciones" type="long" unit="invocaciones/mes">
                ...
            </SLAParameter>
            <SLAParameter name="Tiempo_Respuesta" type="long" unit="ms">
                ...
            </SLAParameter>
            <Metric name="CantidadInvocaciones" type="long" unit="invocaciones/mes">
                ...
            </Metric>
            <Metric name="TiempoRespuesta" type="long" unit="ms">
                ...
            </Metric>
        </Operation>
    </ServiceDefinition>
    <Obligations>
        <ServiceLevelObjective name="CantidadInvocacionesSLO">
            ...
        </ServiceLevelObjective>
        <ServiceLevelObjective name="TiempoRespuestaSLO">
            ...
        </ServiceLevelObjective>
        <ActionGuarantee name="CantidadInvocacionesNotificationGuarantee">
            ...
        </ActionGuarantee>
        <ActionGuarantee name="TiempoRespuestaNotificationGuarantee">
            ...
        </ActionGuarantee>
    </Obligations>
</SLA>
```

Figura 44 – Normativa WSLA Movicel DNIC

En la Figura 44 se observa que DNIC es la Organización que provee el servicio y que Movicel es quien consume. Además se visualizan, dentro de la definición del servicio, las métricas y los parámetros correspondientes a la cantidad de invocaciones y al tiempo de respuesta máximo. Y por último se declaran, dentro de las *Obligations*, objetivos de nivel que se pretenden y las garantías de esos objetivos.

En la Figura 45 se muestra el primer *ServiceLevelObjective* que corresponde al requisito de cantidad de invocaciones.

```
<ServiceLevelObjective name="CantidadInvocacionesSLO">
    <Obliged>Movicel</Obliged>
    <Validity>
        <Start>2017-11-30T14:00:00.000-05:00</Start>
        <End>2017-12-31T14:00:00.000-05:00</End>
    </Validity>
    <Expression>
        <Predicate xsi:type="MenorQue">
            <SLAParameter>Cant_Invovaciones</SLAParameter>
            <Value>800</Value>
        </Predicate>
    </Expression>
    <EvaluationEvent>NewValue</EvaluationEvent>
</ServiceLevelObjective>
```

Figura 45 – ServiceLevelObjective Cantidad de invocaciones

En la Figura 45 se muestra, además de las fechas de validez del objetivo, la expresión que debe cumplirse, es decir, que la cantidad de invocaciones debe ser menor que 800 en el mes.

En la Figura 46 se muestra el segundo *ServiceLevelObjective* que corresponde al requisito de tiempo de respuesta del servicio.

```
<ServiceLevelObjective name="TiempoRespuestaSLO">
    <Obliged>Movicel</Obliged>
    <Validity>
        <Start>2017-11-30T14:00:00.000-05:00</Start>
        <End>2017-12-31T14:00:00.000-05:00</End>
    </Validity>
    <Expression>
        <Predicate xsi:type="MenorQue">
            <SLAParameter>Tiempo_Respuesta</SLAParameter>
            <Value>900</Value>
        </Predicate>
    </Expression>
    <EvaluationEvent>NewValue</EvaluationEvent>
</ServiceLevelObjective>
```

Figura 46 – ServiceLevelObjective Tiempo de respuesta

En la Figura 46 se observa, además de las fechas de validez del objetivo, la expresión correspondiente a que el tiempo de respuesta debe ser menor que 900 ms por invocación.

6.2.2. Protección de Datos Personales MIDES

Movicel debe consultar al MIDES los datos referentes a si las personas son socialmente vulnerables. Para ello consume el servicio esVulnerable provisto por el MIDES. El dato de si

una persona es socialmente vulnerable es sensible, por lo que el MIDES, para poder compartirlo con Movicel, necesita el consentimiento de cada persona. Para esto, el MIDES define las normativas de “Protección de datos personales” en XACML donde se establece el consentimiento de cada persona a compartir el dato.

La Figura 47 muestra un ejemplo de un ciudadano con cédula de identidad 1.234.567-8 que brinda consentimiento para la Acción *Select_for_share* y el Propósito *MDM_BusinessOriented* al organismo MIDES sobre el dato de si es vulnerable socialmente.

```
<PolicySet PoliicySetId="ps1" PolicyCombiningAlgId="deny-unless-permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match>
          <AttributeValue>12345678</AttributeValue>
          <AttributeDesignator AttributeId="resource:NumeroDocumento"
            Category="attribute-category:resource"></AttributeDesignator>
        </Match>
        <Match>
          <AttributeValue>CI</AttributeValue>
          <AttributeDesignator AttributeId="resource:TipoDocumento"
            Category="attribute-category:resource"></AttributeDesignator>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
  <Policy PolicyId="P1" RuleCombiningAlgId="first-applicable">
    <Target>
      <AnyOf>
        <AllOf>
          <Match>
            <AttributeValue>Select_for_share</AttributeValue>
            <AttributeDesignator AttributeId="action:action-id"
              Category="attribute-category:action"></AttributeDesignator>
          </Match>
          <Match>
            <AttributeValue>MDM_BusinessOriented</AttributeValue>
            <AttributeDesignator AttributeId="purpose:purpose-id"
              Category="attribute-category:purpose"></AttributeDesignator>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
    <Rule Effect="Permit" RuleId="Rule0">
      <Target>
        <AnyOf>
          <AllOf>
            <Match>
              <AttributeValue>MIDES</AttributeValue>
              <AttributeDesignator AttributeId="subject:subject-id"
                Category="subject-category:access-subject"></AttributeDesignator>
            </Match>
          </AllOf>
        </AnyOf>
      </Target>
      <Condition>
        <Apply>
          <AttributeValue>EsVulnerable</AttributeValue>
          <AttributeDesignator AttributeId="resource:Target"
            Category="attribute-category:resource"></AttributeDesignator>
        </Apply>
      </Condition>
    </Rule>
  </Policy>
</PolicySet>
```

Figura 47 – Normativa XACML

6.2.3. QoS Movicel

Además de cumplir con su requisito de envío publicitario a los clientes prepagos, Movicel expone un servicio de envío de SMS con publicidad de acuerdo al sector social de todos sus clientes. Para ello decide crear una normativa donde especifica la calidad del servicio expuesto. La normativa será especificada utilizando WSQDL.

En la Figura 48 se muestra cómo queda especificada la normativa de calidad de servicio.

```
<QualityFactor>
    <MeasureFactor>
        <ResponseTime>
            <MeasureDirection>
                <ReadingNumber>123456</ReadingNumber>
            </MeasureDirection>
            <EnvVariables>
                <Variable>
                    <VarName>number of CPU</VarName>
                    <VarValue>4</VarValue>
                </Variable>
            </EnvVariables>
            <MetricValue>
                <Range>0.14 - 0.19</Range>
                <Type>float</Type>
                <Unit>second</Unit>
            </MetricValue>
        </ResponseTime>
    </MeasureFactor>
    <BizValueFactor>
        <ServiceCost>
            <ServicePrice>
                <Price unit="won">100000</Price>
                <ContractDoc>
                    <ContractDocNumber>CN123456789</ContractDocNumber>
                </ContractDoc>
                <PayMethod>quaterly</PayMethod>
            </ServicePrice>
        </ServiceCost>
    </BizValueFactor>
    <EvalFactor>
        <Security>
            <Property name="confidentiality">
                <SubProperty name="message level confidentiality">
                    <Function name="XML ENC Handler" severity="1">
```

Figura 48 – Normativa WSQDL

Se define un factor de calidad vinculado al sistema que es el tiempo de respuesta (definido en el *MeasureFactor*) y otro vinculado al negocio que es costo de servicio (definido en el *BizValueFactor*). La Figura 48 muestra la definición de las variables, métricas y cómo comienza a definirse la función para una evaluación. Se especifica un tiempo de respuesta entre 0.14 y 0.19 segundos y un costo de servicio de 100000.

6.2.4. Intercambio de mensajes

Como se mencionó en la sección 6.1, se necesita una normativa correspondiente a coreografías B2B, para establecer las reglas de comunicación en el intercambio de mensajes entre el Movicel, DNIC y MIDES. La Normativa se especifica utilizando el lenguaje WS-CDL.

La Figura 49 muestra la normativa correspondiente a la coreografía necesaria.

```
<choreography name="PersonalDataChoreography" root="true">
    <description type="documentation">
        Coreografía para comunicación Movicel - DNIC - MIDES
    </description>
    <relationship type="tns:Movicel2DNIC"/>
    <relationship type="tns:Movicel2Mides"/>
    <variableDefinitions>
        ...
    </variableDefinitions>

    <sequence>
        <interaction name="DatosPersona" operation="obtenerDatosPersona" channelVariable="tns:Movicel2DNICC">
            ...
        </interaction>
        <interaction name="VulnerabilidadPersona" operation="esVulnerablePersona" channelVariable="tns:Movicel2MIDESC">
            ...
        </interaction>
    </sequence>
</choreography>
```

Figura 49 – Coreografía WS-CDL Movicel-DNIC-MIDES

En la Figura 49 se observa la relación entre Movicel y DNIC, y entre Movicel y MIDES. Estas organizaciones debieron ser previamente definidas. La coreografía cuenta con una serie de variables y además con las interacciones correspondientes a los intercambios de mensajes. Se observa en la Figura 49 que esas interacciones, en este caso, ocurren en secuencia (y no de forma paralela).

La Figura 50 muestra la primera interacción, que corresponde al intercambio entre Movicel y DNIC.

```
<interaction name="DatosPersona" operation="obtenerDatosPersona" channelVariable="tns:Movicel2DNICC">
    <description type="documentation">
        Obtener datos de persona a partir de Documento de Identidad
    </description>
    <participate relationshipType="tns:Movicel2DNIC" fromRoleTypeRef="tns:MovicelRole" toRoleTypeRef="tns:DNICRole"/>
    <exchange name="DatosRequest" informationType="tns:QuoteRequestType" action="request">
        <description type="documentation">
            Mensaje de Request con Documento de Identidad
        </description>
        <send variable="cdl:getVariable('datosRequest','','')"/>
        <receive variable="cdl:getVariable('datosRequest','','')"/>
    </exchange>
    <exchange name="DatosResponse" informationType="tns:DatosResponseType" action="respond">
        <description type="documentation">
            Mensaje de respuesta Exito con los datos de la persona
        </description>
        <send variable="cdl:getVariable('datosResponse','','')"/>
        <receive variable="cdl:getVariable('datosResponse','','')"/>
    </exchange>
    <exchange name="DatosResponseFault" informationType="tns:DatosResponseFaultType" action="respond" faultName="InvalidProductFault">
        <description type="documentation">
            Mensaje de respuesta Error
        </description>
        <send variable="cdl:getVariable('faultResponse','','')"/>
        <receive variable="cdl:getVariable('faultResponse','','')"/>
    </exchange>
</interaction>
```

Figura 50 – Interacción ObtenerDatosPersona

En la Figura 50 se muestra la especificación de que en primer lugar Movicel debe enviar un *request.* Luego se especifica que la DNIC debe enviar un mensaje de respuesta de éxito o de fallo.

La Figura 51 muestra la segunda interacción, que corresponde al intercambio de mensajes entre Movicel y MIDES.

```
<interaction name="VulnerabilidadPersona" operation="esVulnerablePersona" channelVariable="tns:Movicel2MIDESC">
        <description type="documentation">
                Determinar si la persona es Vulnerable socialmente
        </description>
        <participate relationshipType="tns:Movicel2Mides" fromRoleTypeRef="tns:MovicelRole" toRoleTypeRef="tns:MidesRole"/>
        <exchange name="VulnerableRequest" informationType="tns:VulnerableRequestType" action="request">
                <description type="documentation">
                        Mensaje de Request con Documento de Identidad
                </description>
                <send variable="cdl:getVariable('vulnerableRequest','','')"/>
                <receive variable="cdl:getVariable('vulnerableRequest','','')"/>
        </exchange>
        <exchange name="VulnerableResponse" informationType="tns:VulnerableResponseType" action="respond">
                <description type="documentation">
                        Mensaje de respuesta Exito con booleano de vulnerabilidad social
                </description>
                <send variable="cdl:getVariable('datosResponse','','')"/>
                <receive variable="cdl:getVariable('datosResponse','','')"/>
        </exchange>
        <exchange name="VulnerableResponseFault" informationType="tns:VulnerableResponseFaultType"
                action="respond" faultName="InvalidProductFault">
                <description type="documentation">
                        Mensaje de respuesta Error
                </description>
                <send variable="cdl:getVariable('faultResponse','','')"/>
                <receive variable="cdl:getVariable('faultResponse','','')"/>
        </exchange>
</interaction>
```

Figura 51 – Interacción EsVulnerablePersona

En la Figura 51 se muestra la especificación de que Movicel debe enviar un *request*, y que MIDES le puede responder con éxito o error.

6.3. Desarrollo

En esta sección se desarrolla la solución del caso de estudio mostrando la aplicabilidad del Gestor de normativas y cómo la herramienta da soporte a cargar las normativas especificadas en la sección 6.2 en el contexto descripto en la sección 6.1. Se selecciona una de las normativas especificadas para desarrollar y mostrar en la herramienta: acuerdo de nivel de servicio para el "Servicio básico de información" entre DNIC y Movicel correspondiente a WSLA.

Un usuario Administrador de la plataforma ingresa al Backoffice de la herramienta y mediante una funcionalidad de gestor de Organizaciones que se encuentra disponible como se muestra en la Figura 52, se crea en el sistema las nuevas organizaciones: Movicel y DNIC.

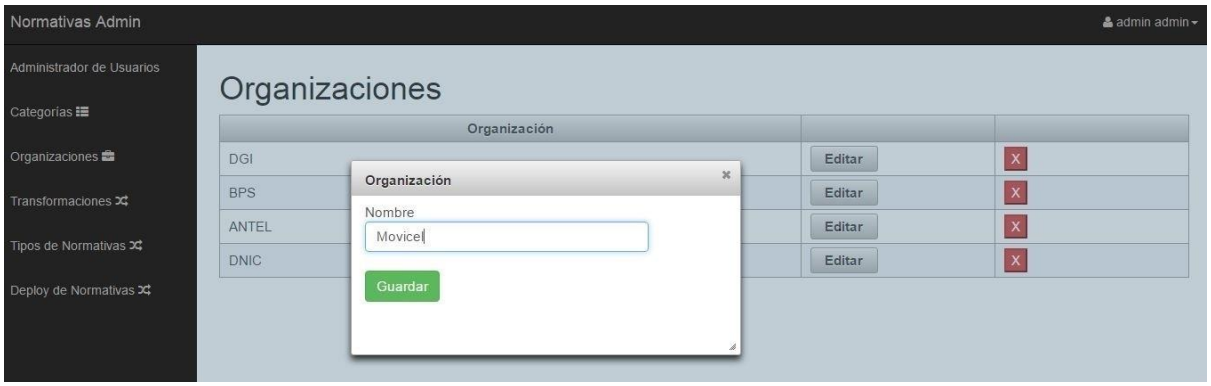

Figura 52 – Organización Movicel

De esta manera ambas organizaciones existen en el sistema.

La normativa que se desea llevar a cabo corresponde al tipo de normativa WSLA y entra en la categoría SLA. Es así que el usuario administrador de la herramienta debe crear en el sistema la Categoría correspondiente (en caso de que no exista) a Acuerdo de Nivel de Servicio (SLA) como se muestra en la Figura 53.

Figura 53 – Categoría SLA

Varios tipos de normativas pueden pertenecer a la categoría SLA, en particular *Web Service Level Agreement* (WSLA). Mediante la misma interfaz de administración, el usuario administrador o un usuario de tipo usuario técnico (permiso asignado únicamente por algún usuario administrador), crea el Tipo de Normativa WSLA.

La Figura 54 muestra la interfaz para gestionar los tipos de normativas y las transformaciones que corresponden realizarse a las normativas de dicho tipo. Se solicitan datos informativos del tipo de normativa como un nombre y una descripción. También se elige la categoría a la que pertenece (SLA) y la cantidad de organizaciones participantes que deberán aprobar las normativas correspondientes a este tipo.

Además se solicita el archivo XSD correspondiente al lenguaje de especificación de WSLA y las transformaciones que van a tener que efectuar todas las normativas que correspondan a este tipo de normativa (las representaciones deben ser especificadas por un usuario administrador previamente). Para seleccionar estas representaciones se solicita el archivo de transformación que puede ser un *template Freemarker* o un XSLT y el archivo de origen para efectuarla (el original especificado con el esquema de WSLA o el resultado de alguna otra transformación).

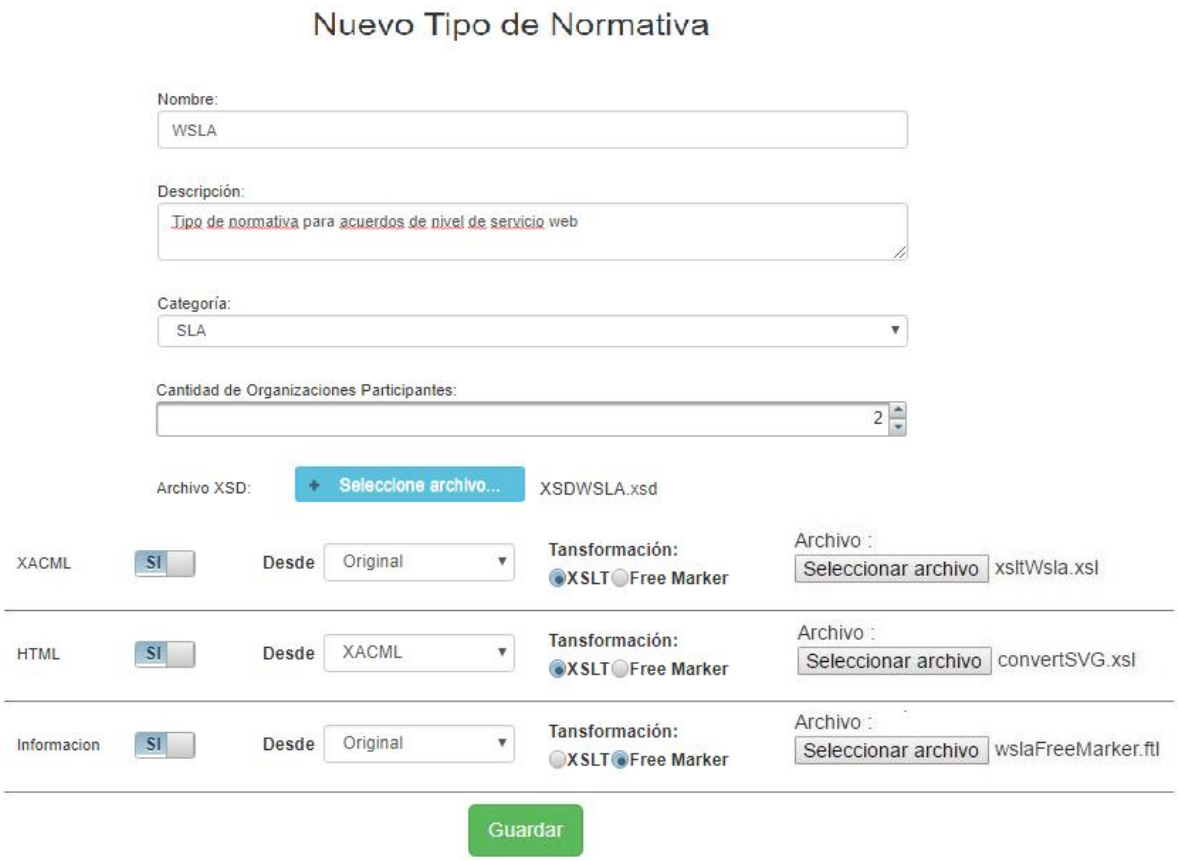

Figura 54 – Tipo de normativa WSLA

En la Figura 54 se observa que las representaciones que se desean para las normativas de WSLA son todas las que en este momento existen en el sistema (“SI”): una denominada XACML, otra HTML y otra Resumen. Para la representación XACML se establece que a

partir de la normativa original especificada utilizando WSLA y utilizando un archivo XSLT, se realice la transformación. Para HTML se establece que a partir del resultado de la transformación XACML y aplicando otro archivo XSLT se realice la transformación. Y para Resumen, se establece que la transformación se realice a partir del archivo original (normativa WSLA) y utilizando un *template Freemarker*.

Una vez creado el tipo de normativa, el técnico o un usuario administrador de una organización participante en normativas de ese tipo, ingresa al Frontoffice de la aplicación para crear la nueva normativa especificada en WSLA, correspondiente al servicio “Servicio básico de información” y que involucra a Movicel y DNIC. Cabe recalcar que una vez cargado el tipo de normativa, para dar de alta nuevas normativas correspondientes a ese tipo, no es necesario volver a dar de alta el tipo de normativa.

Como se muestra en la Figura 55, se solicitan el nombre de la normativa, las organizaciones involucradas y el archivo que contiene la normativa. Al cargar la normativa, se despliega el contenido y se permite la edición del mismo.

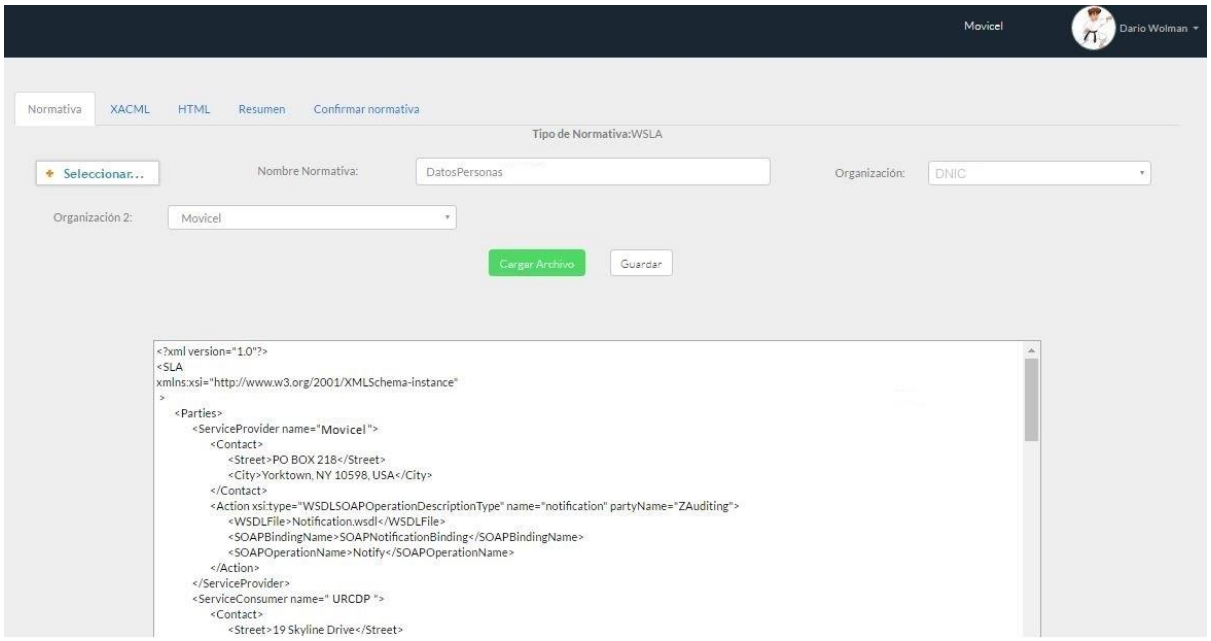

Figura 55 – Normativa DatosPersonaDNIC

Además se visualizan las pestañas que corresponden a todas las transformaciones del tipo de normativa WSLA. Y finalmente una pestaña de confirmación para almacenar todos los datos relacionados a la nueva normativa.

En la Figura 56 se muestra la representación en lenguaje de puesta en producción y la posibilidad de descargar ese archivo generado. Para realizar la transformación se utiliza el archivo de transformación XSLT cargado al crear el tipo de normativa WSLA y las reglas de transformación que se definieron en ese momento (en este caso tomando como archivo a transformar, el original especificado en WSLA).

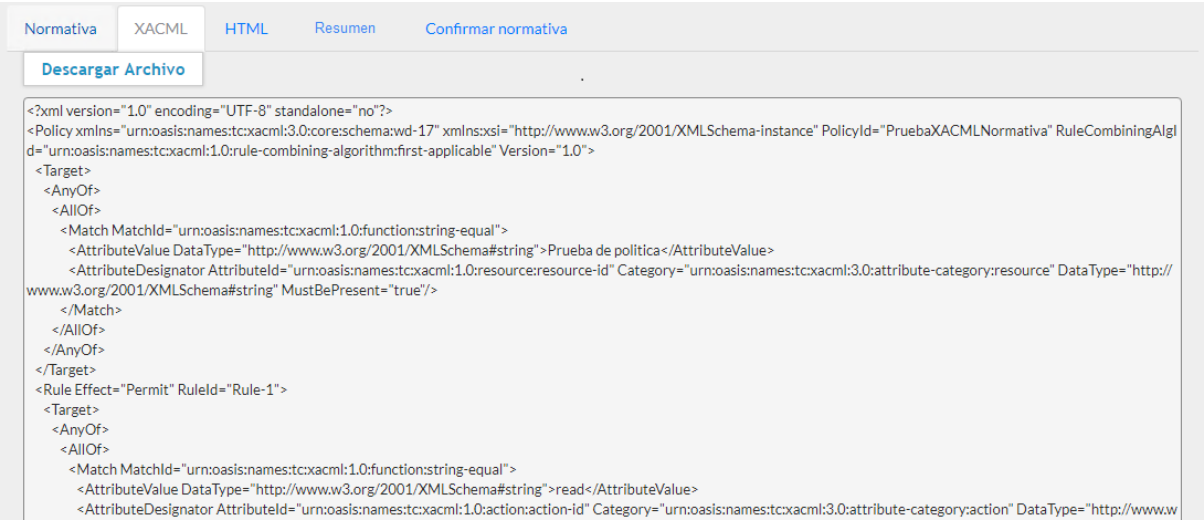

Figura 56 – Representación XACML

En la Figura 57 se visualiza la representación en HTML con la utilización de SVG para obtener la normativa desplegada gráficamente.

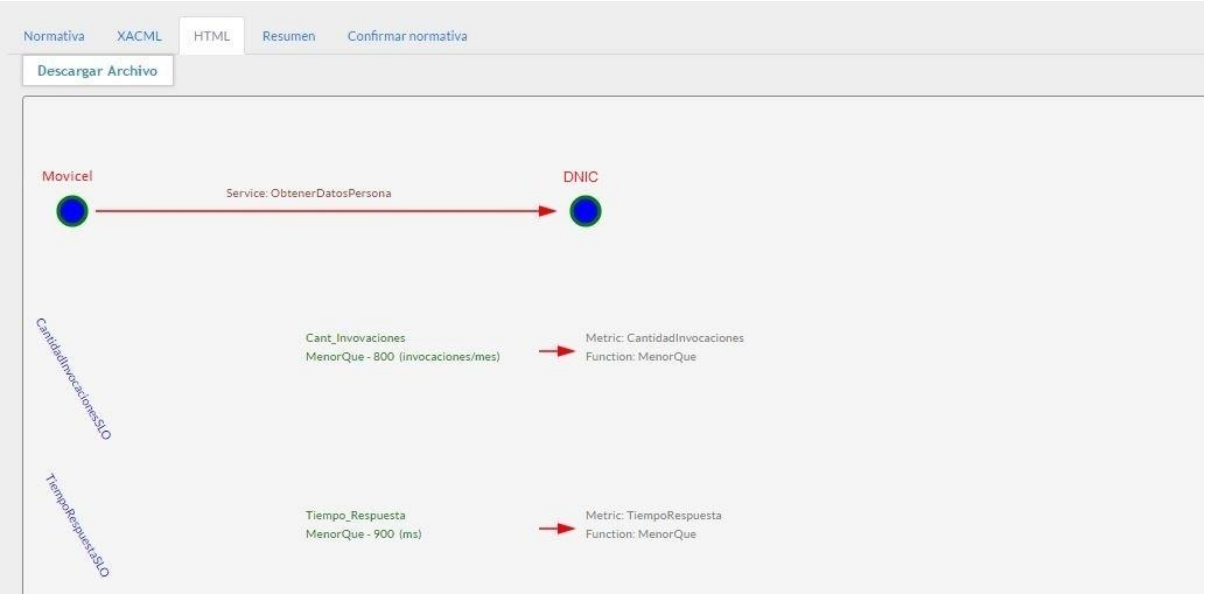

Figura 57 – Representación HTML

En la Figura 58 se muestra otra representación que resume a la normativa y no utiliza un lenguaje tan técnico. Esta transformación, también utilizando XSLT y da como resultado un archivo HTML. La aplicación está configurada para que en esos casos pueda dibujar el contenido HTML.

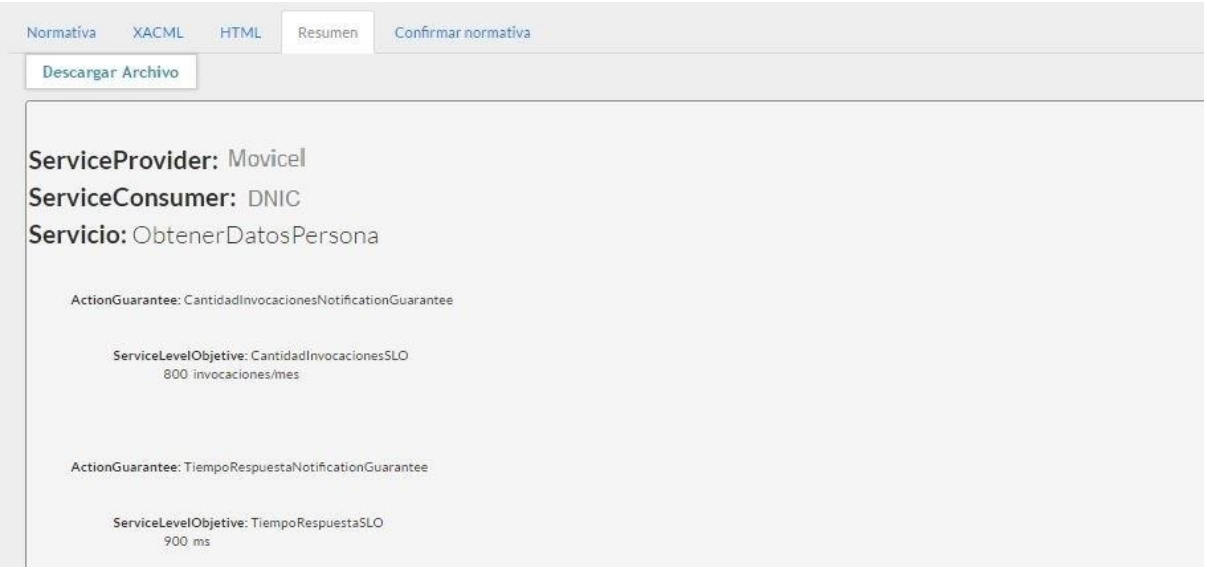

Figura 58 – Representación Resumen

La idea del resultado de esta representación “Resumen” es mostrar los datos principales de la normativa en lenguaje menos técnico y más amigable (La salida también es un archivo HTML).

Finalmente se muestra en la Figura 59 la pestaña de confirmación de la normativa con todos sus datos.

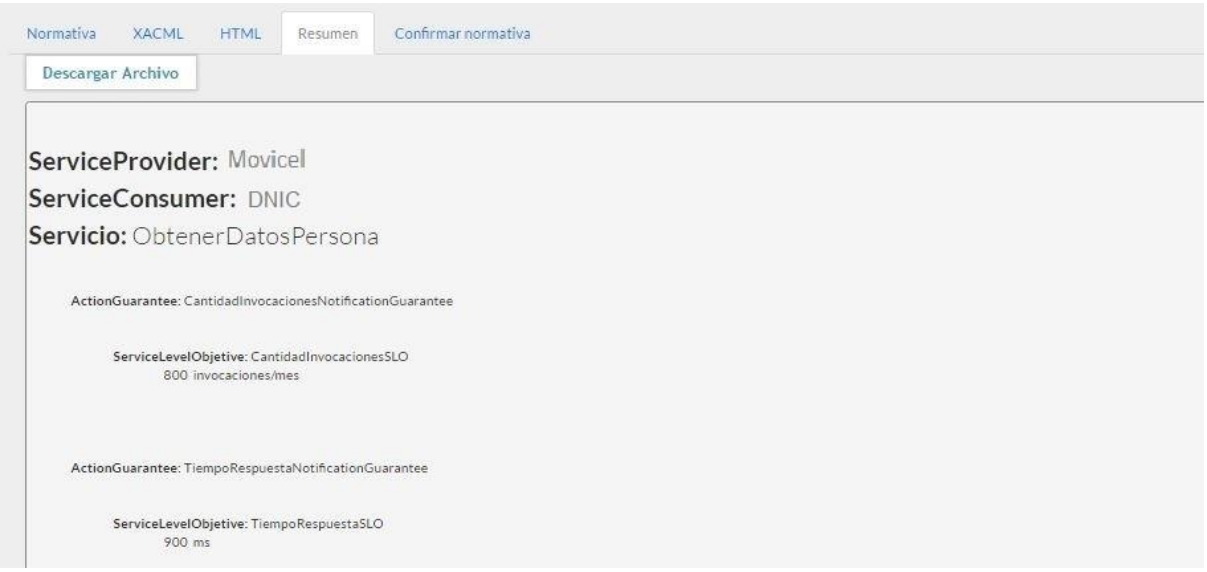

Figura 59 – Confirmación Normativa

Al seleccionar confirmar se crea la nueva normativa guardando los datos en la base de datos postgreSQL y el archivo especificado en el lenguaje de puesta en producción en la base eXist.

La normativa ya está creada y guardada en el sistema, el siguiente paso es que sea aprobada por las organizaciones involucradas. La funcionalidad de aprobación está disponible en el Frontoffice de la herramienta y sólo podrán acceder a aprobar la normativa usuarios con rol Administrador de Organización de cada una de las organizaciones.

Inicia sesión un usuario perteneciente a la organización DNIC (previamente registrado) correspondiente al tipo de usuario Administrador de organización (asignado desde el Backoffice por un usuario Administrador) y accede a la funcionalidad de normativas pendientes para visualizar la normativa y aprobarla como se muestra en la Figura 60.

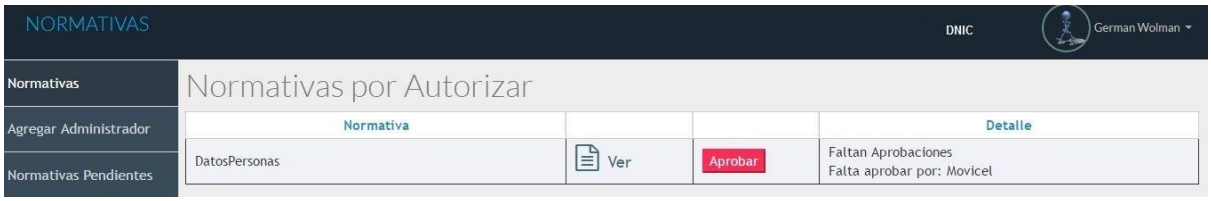

Figura 60 – Aprobación normativa por DNIC

De la misma manera, inicia sesión un usuario perteneciente a Movicel con el mismo tipo usuario para también aprobar la normativa. Una vez que la normativa es aprobada por las organizaciones participantes queda lista para ser desplegada (“poner en producción”).

Finalmente ingresa un usuario administrador de la plataforma que, una vez que la normativa fue aprobada por todas las organizaciones participantes, se le habilita la opción para poner en producción la normativa en el lenguaje de puesta en producción como se muestra en la Figura 61, almacenándola en el servidor de WSO2.

Figura 61 – Puesta en producción

La Figura 62 muestra la consola de administración del WSO2 *Identity Server* y al acceder a *Policy Administration* se visualiza la Normativa en el lenguaje común.

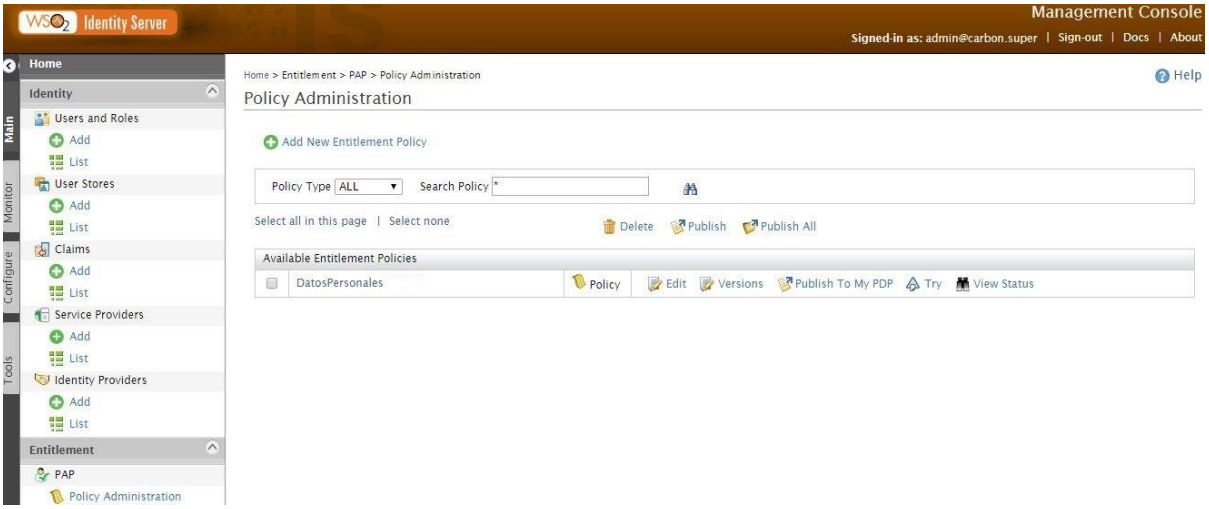

Figura 62 – Consola Administración WSO2 Identity Server

6.4. Conclusión

El caso de estudio permitió verificar la utilidad del Gestor de Normativas como solución a una realidad planteada entre varias organizaciones que se comunican mediante una plataforma de integración. La realidad abarca los tipos de normativas desarrollados en el proyecto, y cada uno de ellos posee variabilidad en cuanto al lenguaje de especificación y la cantidad de organizaciones involucradas. Se logran identificar tipos de normativas y categorizarlos según los requerimientos que surgen a partir de la descripción del problema. A partir de los tipos de normativas se especifican las normativas necesarias para resolver la problemática.

Mediante la ejecución de pruebas se logra demostrar que la herramienta desarrollada soporta cada uno de los tipos de normativas identificados, permitiendo la gestión de normativas en diferentes lenguajes de especificación. Si bien se muestra en detalle la aplicación de la herramienta a uno de los tipos de Normativa (WSLA), se deja claro en la sección 6.2 el proceso de comunicación y las normativas que se deben de desarrollar para resolver todos los puntos de la problemática planteada.

Se detallan las pruebas y aplicabilidad de la herramienta para el tipo de normativa WSLA. Se muestran varios aspectos destacables como son: el hecho de poder representar la normativa en otros lenguajes hasta concluir en el lenguaje de puesta en producción, el soporte multiorganizacional, un sistema de aprobaciones y la puesta en producción de la normativa en la plataforma de integración.

Estas pruebas demostraron que, una vez que se crean las organizaciones y las representaciones posibles para los diferentes tipos de normativas, el flujo de creación, transformación, aprobación y puesta en producción de normativas, es bastante intuitivo y directo.

7. Conclusiones y trabajo a futuro

En esta sección se presentan las conclusiones sobre todo el proyecto y se mencionan posibles mejoras a futuro que enriquecen el trabajo realizado.

7.1. Conclusiones

En un contexto de colaboración entre organizaciones es necesario que la plataforma de integración brinde ciertos mecanismos para monitorear y administrar diferentes normativas. Para abordar esta problemática se desarrolla una herramienta capaz, no sólo de brindar funcionalidades de gestión, sino también funcionalidades que faciliten la colaboración interorganizacional.

En una primera instancia, en el proyecto se realizó un estudio y análisis sobre diferentes categorías de normativas existentes, así como también de los lenguajes de especificación de cada una de ellas. La mayoría de las normativas se especifican en lenguaje natural. Sin embargo, para algunos tipos de normativas se han desarrollado formatos legibles por máquinas, para que se puedan especificar en lenguajes más descriptivos como XML. Se estudiaron costos y beneficios de realizar las transformaciones correspondientes para expresar cada una de las especificaciones en el lenguaje de puesta en producción.

Las categorías que se consideraron en el proyecto fueron Acuerdo de Nivel de Servicio (SLA), Calidad de Servicio (QoS), Intercambio de mensajes entre organizaciones y Protección de Datos Personales con sus respectivos ejemplos de tipos de Normativas: WSLA, WSQDL, coreografías WS-CDL y XACML. Pero el sistema fue diseñado de forma extensible, es decir, para soportar múltiples categorías y tipos de normativas.

Junto con el estudio de normativas y sus distintas especificaciones, también hubo que enfocarse en el contexto en el cual se encuentran. Es decir, un contexto en el que se resalta la importancia de la definición de normativas, reglas que establecen las organizaciones, incluyendo la seguridad en el intercambio de información y el cumplimiento de las mismas.

Se realizó el análisis de los requerimientos y un diseño y desarrollo de prototipos de la plataforma que abarcaron los casos con mayor riesgo. Entre ellos, la creación de tipos de normativas de forma dinámica con la utilización de JAXB como herramienta clave para el mapeo de esquemas XML a clases java y viceversa: a partir de las librerías de unmarshaller y herramienta xjc incluidas. Además se probaron las conexiones con los diferentes tipos de almacenamiento de información, tanto de imágenes con MongoDB y su extensión gridFS para archivos, como el guardado de XML en base eXist y la persistencia con JPA en postgreSQL.

La visualización gráfica de las normativas para representarlas de forma más amigables y agradables a los usuarios, se logra resolver con la utilización de SVG y la transformación correspondiente desde un tipo de normativa para mapear los elementos de la misma a un HTML. Este mecanismo es un caso particular de las múltiples transformaciones que se

pueden efectuar sobre una normativa, previo a su transformación en el lenguaje de puesta en producción.

Como prototipo final, se incluye la comunicación de la aplicación con el servidor de WSO2, consumiendo una API provista por el mismo, para la puesta en producción final de normativas, así como también la recuperación de las mismas. WSO2 *Identity Server* brinda soporte para XACML y permite la integración con otros sistemas brindando funcionalidades a través de Servicios Web.

Las normativas trabajadas en este proyecto hacen referencia a conjuntos de reglas sobre servicios publicados por ejemplo por organizaciones; y dichos servicios pueden manejar información crítica. Es por esto que la solución implementada cuenta con un manejo de tipos de usuarios y permisos sobre las diferentes funcionalidades. Mediante pruebas realizadas, se logró comprobar la efectividad del funcionamiento de los tipos de usuarios y cómo la aprobación de normativas, o la creación de nuevos tipos, puede ser restringido a determinados usuarios de la herramienta.

Se desarrolló un caso de estudio que abarcó el uso de diferentes tipos de normativas y que permitió validar el uso de la herramienta en un contexto de colaboración entre distintas organizaciones. Se partió de normativas especificadas en diferentes lenguajes y se logró representarlas en un lenguaje de puesta en producción, dando soporte y siendo extensible a múltiples organizaciones. Además, el caso de estudio, permitió verificar el sistema de aprobaciones de normativas y la puesta final en producción de las mismas en la plataforma de integración.

Se logró el despliegue del Gestor de Normativas en la nube. Se generó un dominio para el Frontoffice y Backoffice de la herramienta y otro dominio para acceder por https a la interfaz de WSO2 IS y poder visualizar las normativas en producción.

Se comprobó la factibilidad técnica de construir una herramienta que permita gestionar normativas a ser controladas por una plataforma de integración interorganizacional. La extensibilidad de la herramienta en cuanto a los tipos de normativas y categorías que pueden definirse, permite que múltiples organizaciones puedan hacer uso y adecuarse a dicha herramienta.

Con respecto a las herramientas utilizadas, se destaca la facilidad de eXist-db para integrarse con el servidor de aplicaciones Wildfly y su rápido acceso a los archivos XML. A nivel de implementación esta comunicación fue factible gracias a dependencias Maven.

A pesar de la ventaja de que WSO2 *Identity Server* brinda soporte para XACML, resultó complicada la integración con el servidor de aplicaciones Wildfly debido a los conflictos de certificados con la máquina virtual de Java utilizada.

Se entiende que este tipo de problemática es relevante para la gestión de normativas de hoy en día y para cumplir con la necesidad de que las organizaciones colaboren entre sí a través de una plataforma de integración.

7.2. Trabajo a futuro

En esta sección se describen los puntos encontrados como mejora y extensión del trabajo realizado.

7.2.1. Normativas especificadas en lenguaje no XML

Si bien desde un principio se tomó la decisión de partir de especificaciones de normativas en lenguaje XML, una posible extensión es que la herramienta brinde un mecanismo para transformar normativas especificadas en lenguajes textuales que no tienen una representación XML. Esto permitiría poder representar otra gran variedad de normativas además de las especificadas en XML.

Por ejemplo con JAXB se logra transformar clases JAVA a XML. A partir de normativas expresadas en un esquema de clases se le puede asignar una representación en XML, pudiéndose incorporar al Gestor de Normativas implementado.

7.2.2. Notificaciones y alertas

En cuanto al sistema de aprobaciones manejado por la herramienta, en este proyecto sería bueno implementar el hecho de que se notifique a los usuarios Administrador de Organizaciones cuando existan normativas pendientes de aprobación. Y con esta implementación, el agregado de alguna alerta para que estos usuarios se enteren automáticamente (y no tengan que ir a chequearlo exclusivamente) cuando una normativa que involucra a su organización es puesta en producción por algún Administrador de la plataforma en el servidor de WSO2.

Para implementar este punto se propone *Java Message Service* (JMS) para el uso de colas de mensajes.

7.2.3. Comunicación con chat

Al haber implementado una administración de tipos de usuarios que tienen posibilidad de poseer distintos permisos a funcionalidades, sería bueno tener la implementación de un chat para la comunicación entre los mismos. Esto se puede realizar para intercambiar información dentro de las organizaciones o entre organizaciones (por ejemplo que los administradores de cada organización puedan comunicarse). Esto es útil a la hora de aprobar normativas por ejemplo, o simplemente para tratar algún tema sobre reglas, usuarios, clientes, u otras empresas. Además puede haber comunicación entre usuarios técnicos para llegar a distintos acuerdos.

Para lograr este punto se propone la utilización de la API Web Sockets, que permite la comunicación interactiva entre usuarios clientes de la plataforma.

7.2.4. Manipulación de normativas

Es posible realizar la implementación de una extensión a la transformación de normativas. Es decir, una vez transformada la normativa al lenguaje de puesta en producción, con

agregados de políticas y reglas, mediante una interfaz amigable, poder por ejemplo agregarle que un determinado servicio tenga un tiempo de respuesta y en caso de fallar se mande notificación a determinado usuario y/o mail destinatario (información que no venía en la normativa inicial).

La herramienta permite editar normativas modificando directamente el texto de la especificación, ingresando las reglas en la normativa, pero no mediante una interfaz gráfica amigable. Este punto puede ser interesante para agregar a normativas elementos de estilo general a todos los tipos de normativas, como acciones a tomar en determinadas respuestas a los servicios o registros (*logs*) al acceder a los mismos.

7.2.5. Seguridad en el acceso a la web

Para la puesta en producción es requerido incorporar HTTPS para el acceso seguro a la web, en particular para la parte del Backoffice que administra y gestiona las diferentes funcionalidades del Sistema.

7.2.6. Disponibilidad

Debido a que el sistema puede considerarse de uso crítico y de alta importancia ya que podrían utilizarlo múltiples organizaciones, sería conveniente brindar mecanismos de alta disponibilidad. Por ejemplo, manejar una arquitectura con 2 nodos y un balanceador de carga configurado de manera de dirigir los accesos a uno de ellos en caso de que el otro falle. Esto permite que el servidor pueda responder de forma adecuada a un posible uso masivo de la herramienta en forma simultánea.

Esto se puede implementar fácilmente en Wildfly 10 con la utilización de NGINX, por ejemplo.

7.2.7. Aspectos técnicos

A continuación se describen el trabajo a futuro de algunos aspectos técnicos.

7.2.7.1. Validación de normativas

Cuando se da de alta una nueva normativa, el gestor se encarga de identificar el tipo de normativa al que corresponde. Es decir, que se “compila” la normativa con los tipos de normativas existentes en el sistema. Si la normativa valida con más de una opción, se le debería permitir al usuario seleccionar entre todos los tipos de normativa que validaron. Para implementarlo haría falta recorrer todos los tipos de normativas e ir almacenando en memoria aquellas que validan para luego solicitar al usuario que seleccione una.

7.2.7.2. Resolución del conflicto de certificados

Para resolver los conflictos de certificados entre el WSO2 IS y la máquina virtual de Java que utiliza el Gestor de Normativas se debería detectar todos los certificados necesarios utilizados en el almacén que tiene el servidor de WSO2, encontrarlos e instalarlos en el almacén de certificados de Java.

7.2.7.3. Almacenamiento de archivos

Cuando se crean el tipo de normativa, se generan los archivos java a partir del XSD correspondiente. Esos archivos deberían ser almacenados en algún servidor fuera del servidor de aplicaciones Wildfly para no sobrecargarlo.

8. Referencias

[1] Laura González y Raúl Ruggia, «Towards a Compliance-Aware Inter-organizational Service Integration Platform», *Move Meaningful Internet Syst. OTM 2014 Workshop*, pp. 8-17, 2014.

[2] El Senado y la Cámara de Representantes de la República Oriental del Uruguay, Ley de Protección de Datos Personales y Acción, «Habeas Data». .

[3] KangChan Lee, JongHong Jeon, WonSeok Lee, Seong-Ho Jeong, Sang-Won Park, «QoS for Web Services: Requirements and Possible Approaches», 25-nov-2003. [En línea]. Disponible en: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

[4] W3C, «Rule-based Service Level Agreements (SLA) and Web Services». [En línea]. Disponible en: https://www.w3.org/2005/rules/wg/wiki/Rule-based_Service_Level_Agreements_(SLA)_and_Web_Services.html.

[5] U. R. y de C. de D. P. AGESIC, «Los datos personales y su protección». [En línea]. Disponible en: https://datospersonales.gub.uy/inicio/publicaciones/Guias+de+ayuda/.

[6] Pablo García, Urudata, «Proyecto de Grado: B2B para Q-flow». jul-2002.

[7] Juan Federico Gómez Estupiñán, «Análisis de BPMN como herramienta integral para el Modelado de Procesos de Negocio». [En línea]. Disponible en: http://revistasum.umanizales.edu.co/ojs/index.php/ventanainformatica/article/view/274/.

[8] W3C, «Extensible Markup Language (XML)». [En línea]. Disponible en: https://www.w3.org/XML/.

[9] IBM Knowledge Center, «¿Qué es XML?», 27-may-2004. [En línea]. Disponible en: https://www.ibm.com/support/knowledgecenter/es/SSEPGG_8.2.0/com.ibm.db2.ii.doc/opt/c0007799.htm?view=embed.

[10] W3C, «XML Schema». [En línea]. Disponible en: https://www.w3.org/XML/Schema.

[11] W3C, «XML Schema Part 0: Primer Second Edition». [En línea]. Disponible en: https://www.w3.org/TR/xmlschema-0/.

[12] W3C, «Introduction – SVG 1.1 (Second Edition)». [En línea]. Disponible en: https://www.w3.org/TR/SVG/intro.html.

[13] Aprende Web, «SVG: dibujos y gráficos mediante vectores escalables». [En línea]. Disponible en: http://aprende-web.net/NT/svg/svg_1.php.

[14] W3C, «Transformation». [En línea]. Disponible en: https://www.w3.org/standards/xml/transformation.

[15] W3C, «The Extensible Stylesheet Language Family (XSL)». [En línea]. Disponible en: https://www.w3.org/Style/XSL/.

[16] tutorialspoint.com, «XSLT <template>», *www.tutorialspoint.com*. [En línea]. Disponible en: https://www.tutorialspoint.com/xslt/xslt_template.htm.

[17] Microsoft, «Tipo de datos XML y columnas (SQL Server)». [En línea]. Disponible en: https://docs.microsoft.com/es-es/sql/relational-databases/xml/xml-data-type-and-columns-sql-server.

[18] eXistdb, «The Open Source Native XML Database». [En línea]. Disponible en: http://exist-db.org/exist/apps/homepage/index.html.

[19] Oracle, «XML DB home». [En línea]. Disponible en: http://www.oracle.com/technetwork/database/database-technologies/xmldb/overview/index.html.

[20] OASIS, «eXtensible Access Control Markup Language (XACML) Version 3.0», 22-ene-2013. [En línea]. Disponible en: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[21] IBM, «XACML Architecture - Identity Server 5.1.0 - WSO2 Documentation». [En línea]. Disponible en: https://docs.wso2.com/display/IS510/XACML+Architecture.

[22] EMC Community Network - DECN, «XML and Security: Real world examples of XACML security policies», 17-jun-2010. [En línea]. Disponible en: https://community.emc.com/docs/DOC-7410.

[23] Oracle, «Java Platform, Enterprise Edition: The Java EE Tutorial», 2014. [En línea]. Disponible en: https://docs.oracle.com/javaee/7/tutorial/overview.htm#BNAAW.

[24] W3C, «Guía Breve de Servicios Web». [En línea]. Disponible en: https://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

[25] W3C, «SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)», 27-abr-2007. [En línea]. Disponible en: https://www.w3.org/TR/soap12/.

[26] W3C, «Web Service Definition Language (WSDL)», 15-mar-2001. [En línea]. Disponible en: https://www.w3.org/TR/wsdl.

[27] W3C, «HTML». [En línea]. Disponible en: https://www.w3.org/html/.

[28] Mozilla y colaboradores individuales, «HTML5», *Mozilla Developer Network*. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/HTML/HTML5.

[29] W3C, «JavaScript Web APIs», 2016. [En línea]. Disponible en: https://www.w3.org/standards/webdesign/script.

[30] W3C, «Web Services Choreography Description Language Version 1.0», 09-nov-2011. [En línea]. Disponible en: https://www.w3.org/TR/ws-cdl-10/.

[31] Jorge Giraldo, Jaime A. Guzmán, y Demetrio A. Ovalle, «Revista Ingenierías Universidad de Medellín - INTEGRACIÓN DE PROCESOS DE NEGOCIO BASADOS EN SERVICIOS WEB: COREOGRAFÍA Y SATISFACCIÓN DE RESTRICCIONES», 14-abr-2008. [En línea]. Disponible en: http://www.scielo.org.co/pdf/rium/v7n12/v7n12a08.

[32] Youngkon Lee Korea Polytechnic University, «WSQDL (Web Service Quality Description Language)», 16-abr-2007. [En línea]. Disponible en: https://documentslide.com/documents/1-wsqdl-web-service-quality-description-language-16-th-april-2007-youngkon.html.

[33] OASIS, «Web Services Quality Factors Version 1.0», 01-jul-2010. [En línea]. Disponible en: http://docs.oasis-open.org/wsqm/wsqf/v1.0/WS-Quality-Factors-v1.0-cd02.html.

[34] Marc Oriol Hilari, «Quality of Service (QoS) in SOA Systems. A Systematic Review», 06-sep-2009. [En línea]. Disponible en: https://upcommons.upc.edu/bitstream/handle/2099.1/7714/Master%20thesis%20-%20Marc%20Oriol.pdf?sequence=1.

[35] Svend Frølund, Jari Koistinen y Software Technology Laboratory, «Quality-of-Service Specification in Distributed Object Systems», sep-1998. [En línea]. Disponible en: http://www.hpl.hp.com/techreports/98/HPL-98-159.pdf.

[36] IBM, «Web Service Level Agreement (WSLA) Language Specification», 28-ene-2003. [En línea]. Disponible en: https://www.researchgate.net/publication/200827750_Web_Service_Level_Agreement_WSLA_Language_Specification.

[37] D.Davide Lamanna, James Skene y Wolfgang Emmerich, «SLAng: A Language for Defining Service Level Agreements». [En línea]. Disponible en: http://tapas.sourceforge.net/private/Workshops/Papers/SLAng.pdf.

[38] Acuña Ignacio y Serra Diego, «Proyecto de Grado: Implementación de una Plataforma para la Protección de Datos Personales en Soluciones Master Data Management de Gobierno Electrónico», may 2016.

[39] G. Armellin, D. Betti, F. Casati, A. Chiasera, G. Martinez, y J. Stevovic, «Privacy Preserving Event Driven Integration for Interoperating Social and Health Systems», en *Secure Data Management*, 2010, pp. 54-69.

[40] Dr. IIavarasan Egambaram, G. Vadivelou, S. Prasath Sivasubramanian, «QOS BASED WEB SERVICE SELECTION». [En línea]. Disponible en: http://www.ubicc.org/files/pdf/Paper%20ID%20460_460.pdf.

[41] Linh Thao Ly, David Knuplesch, Stefanie Rinderle-Ma, Kevin Goser, Manfred Reichert, Peter Dadam, «SeaFlowsToolset - Compliance Verification MadeE asy». [En línea]. Disponible en: http://ceur-ws.org/Vol-592/PaperDemo05.pdf.

[42] Balasubramanian K, Balasubramanian J, Parsons J, Gokhale A, Schmidt DC, «A platform-independent component modeling language for distributed real-time and embedded systems», 2005.

[43] Kavimandan A, Balasubramanian K, Shankaran N, Gokhale A, Schmidt DC, «QUICKER: A Model-Driven QoS Mapping Tool for QoS-Enabled Component Middleware», pp. 62-70, 2007.

[44] Adela del-Río-Ortega, Antonio Ruiz-Cortés, Cristina Cabanillas, Manuel Resinas, «PPINOT Tool Suite: A Performance Management Solution for Process-Oriented Organisations». [En línea]. Disponible en: http://www.isa.us.es/sites/default/files/DEL_RIO_Ortega-ICSOCToolDemo_2013.pdf.

[45] Tambe S, Dabholkar A, Gokhale A, «CQML: Aspect-Oriented Modeling for Modularizing and Weaving QoS Concerns in Component-Based Systems», pp. 11-20, 2009.

[46] tutorialspoint.com, «Java Tutorial», *www.tutorialspoint.com*. [En línea]. Disponible en: https://www.tutorialspoint.com/java/.

[47] E. Foundation, «Eclipse». [En línea]. Disponible en: http://www.eclipse.org/.

[48] Apache Maven Proyect, «Maven». [En línea]. Disponible en: https://maven.apache.org/.

[49] Project Documentation Editor, «Documentation - WildFly 10». [En línea]. Disponible en: https://docs.jboss.org/author/display/WFLY10/Documentation.

[50] Oracle, «javaserverfaces-spec», 21-may-2013. [En línea]. Disponible en: https://javaee.github.io/javaserverfaces-spec/.

[51] Oracle, «JavaBeans(TM) (The Java™ Tutorials)». [En línea]. Disponible en: https://docs.oracle.com/javase/tutorial/javabeans/index.html.

[52] Oracle, «PrimeFaces in the Enterprise», abr-2014. [En línea]. Disponible en: http://www.oracle.com/technetwork/articles/java/java-primefaces-2191907.html.

[53] Oracle, «Java Architecture for XML Binding (JAXB)», mar-2003. [En línea]. Disponible en: http://www.oracle.com/technetwork/articles/javase/index-140168.html.

[54] WSO2 Identity Server 5.1.0, «WSO2 Identity Server Documentation». [En línea]. Disponible en: https://docs.wso2.com/display/IS510/WSO2+Identity+Server+Documentation.

[55] Oracle, «Java Persistence API». [En línea]. Disponible en: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html.

[56] Oracle, «Introduction to the Java Persistence API - The Java EE 6 Tutorial». [En línea]. Disponible en: https://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html.

[57] The PostgreSQL Global Development Group, «PostgreSQL». [En línea]. Disponible en: https://www.postgresql.org/about/.

[58] mongoDB, «Reinventando la gestión de datos», *MongoDB*. [En línea]. Disponible en: https://www.mongodb.com/es.

[59] mongoDB, «Building MongoDB Applications with Binary Files Using GridFS: Part 1», *MongoDB*. [En línea]. Disponible en: https://www.mongodb.com/blog/post/building-mongodb-applications-binary-files-using-gridfs-part-1.

[60] Guzmán Llambías, «AGESIC - Plataforma de interoperabilidad - Servicio básico de información - Descripción funcional del servicio», 29-jun-2012. [En línea]. Disponible en: https://www.agesic.gub.uy/innovaportal/file/1799/1/documentacion_servicio_dnic_ci.pdf.

[61] MIDES, «Misión», 09-oct-2009. [En línea]. Disponible en: http://www.mides.gub.uy/4376/mision.

[62] D. of M. and C. S. W.M.P. van der Aalst Eindhoven University of Technology, «Process-Aware Information Systems: Lessons to be Learned from Process Mining». [En línea]. Disponible en: http://wwwis.win.tue.nl/~wvdaalst/publications/p522.pdf.

Apéndice 1: Detalle de casos de uso

Alta de Categoría

Caso de Uso	Alta de Categoría
Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea crear una categoría nueva de tipos de normativas. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite esta acción. Para ellos, se debe ingresar únicamente el nombre de la categoría.
Pre-Condiciones	• El actor debe estar logueado al sistema.
Flujo de Eventos	**Flujo Normal** 1. El usuario ingresa nombre para la categoría. 2. No existe una categoría con el mismo nombre, el sistema almacena la nueva categoría. 3. Fin caso de uso. **Flujo alternativo 1 del paso 2**: Existe una categoría con ese nombre en el sistema. 2.1. Se vuelve al paso 1 del flujo normal.
Post-Condiciones	Se da de alta la categoría.

Modificación de Categoría

Caso de Uso	Modificación de Categoría
Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea modificar una categoría existente. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite esta acción.
Pre-Condiciones	• El usuario debe estar logueado al sistema.
Flujo de Eventos	**Flujo Normal**

	1. El sistema lista las categorías existentes. 2. El usuario selecciona una categoría a modificar. 3. El usuario realiza modificación en el nombre. 4. No existe una categoría con el mismo nombre, el sistema almacena la categoría modificada. 5. Fin caso de uso. **Flujo alternativo 1 del paso 4**: Existe una categoría con ese nombre en el sistema. 4.1. Se vuelve al paso 1 del flujo normal.
Post-Condiciones	Se modifica la categoría.

Baja de Categoría

Caso de Uso	Baja de Categoría
Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea eliminar una categoría. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite esta acción.
Pre-Condiciones	• El usuario debe estar logueado al sistema. • No existen tipos de normativas pertenecientes a la categoría a eliminar.
Flujo de Eventos	**Flujo Normal** 1. El sistema lista las categorías existentes. 2. El usuario selecciona una categoría a eliminar. 3. El sistema elimina la categoría. 5. Fin caso de uso.
Post-Condiciones	Se elimina categoría.

Alta de Organización

Caso de Uso	Alta de Organización

Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea crear una organización nueva. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite esta acción. Para ellos, se debe ingresar únicamente el nombre de la organización.
Pre-Condiciones	• El actor debe estar logueado al sistema.
Flujo de Eventos	**Flujo Normal** 1. El usuario ingresa nombre para la organización. 2. No existe una organización con el mismo nombre, el sistema almacena la nueva organización. 3. Fin caso de uso. **Flujo alternativo 1 del paso 2**: Existe una organización con ese nombre en el sistema. 2.1. Se vuelve al paso 1 del flujo normal.
Post-Condiciones	Se da de alta la organización.

Modificación de Organización

Caso de Uso	Modificación de Organización
Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea modificar una organización existente. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite esta acción.
Pre-Condiciones	• El usuario debe estar logueado al sistema.
Flujo de Eventos	**Flujo Normal** 1. El sistema lista las organizaciones existentes. 2. El usuario selecciona una categoría a modificar. 3. El usuario realiza modificación en el nombre. 4. No existe una organización con el mismo nombre, el sistema

	almacena la organización modificada. 5. Fin caso de uso. **Flujo alternativo 1 del paso 4**: Existe una organización con ese nombre en el sistema. 4.1. Se vuelve al paso 1 del flujo normal.
Post-Condiciones	Se modifica la organización.

Baja de Organización

Caso de Uso	Baja de Organización
Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea eliminar una organización. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite esta acción.
Pre-Condiciones	• El usuario debe estar logueado al sistema. • No existen normativas pertenecientes a la Organización a eliminar.
Flujo de Eventos	**Flujo Normal** 1. El sistema lista las organizaciones existentes. 2. El usuario selecciona una organización a eliminar. 3. El sistema elimina la organización. 5. Fin caso de uso.
Post-Condiciones	Se elimina organización.

Alta de Representación

Caso de Uso	Alta de Representación
Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea crear una Representación nueva. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite esta acción. Para ellos, se debe ingresar únicamente el nombre de la Representación.
Pre-Condiciones	• El actor debe estar logueado al sistema.

Flujo de Eventos	**Flujo Normal** 1. El usuario ingresa nombre para la Representación. 2. No existe una Representación con el mismo nombre, el sistema almacena la nueva Representación. 3. Fin caso de uso. **Flujo alternativo 1 del paso 2**: Existe una Representación con ese nombre en el sistema. 2.1. Se vuelve al paso 1 del flujo normal.
Post-Condiciones	Se da de alta la Representación.

Modificación de Representación

Caso de Uso	Modificación de Representación
Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea modificar una Representación existente. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite esta acción.
Pre-Condiciones	• El usuario debe estar logueado al sistema.
Flujo de Eventos	**Flujo Normal** 1. El sistema lista las Representaciones existentes. 2. El usuario selecciona una Representación a modificar. 3. El usuario realiza modificación en el nombre. 4. No existe una Representación con el mismo nombre, el sistema almacena la Representación modificada. 5. Fin caso de uso.

	Flujo alternativo 1 del paso 4: Existe una Representación con ese nombre en el sistema. 4.1. Se vuelve al paso 1 del flujo normal.
Post-Condiciones	Se modifica la organización.

Baja de Representación

Caso de Uso	Baja de Representación
Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea eliminar una Representación. A tales efectos, el sistema proporciona una interfaz gráfica acorde que facilite esta acción.
Pre-Condiciones	• El usuario debe estar logueado al sistema. • No existen tipos de normativas que posean la Representación a eliminar.
Flujo de Eventos	**Flujo Normal** 1. El sistema lista las epresentaciones existentes. 2. El usuario selecciona una Representación a eliminar. 3. El sistema elimina la Representación. 5. Fin caso de uso.
Post-Condiciones	Se elimina Representación.

Administrar Usuario

Caso de Uso	Administrar Usuario
Actor	Usuario Super Administrador.
Descripción	El caso de uso comienza cuando el administrador desea administrar un usuario cambiando su tipo de usuario. A tales efectos, el sistema proporciona una interfaz gráfica acorde para dicha acción.
Pre-Condiciones	• El usuario debe estar logueado al sistema.
Flujo de Eventos	**Flujo Normal** 1. El sistema lista los usuarios existentes con sus tipos de usuario actuales correspondientes.

	2. El usuario selecciona uno y elige su nuevo tipo de usuario. 3. El sistema asigna los permisos correspondientes y modifica al usuario con su nuevo tipo de usuario. 4. Fin del caso de uso.
Post-Condiciones	Se asigna nuevo tipo de usuario al usuario.

Alta Tipo Normativa

Caso de Uso	Alta Tipo Normativa
Actor	Usuario Super Administrador/Técnico
Descripción	El caso de uso comienza cuando el administrador desea crear un nuevo tipo de normativa. A tales efectos, el sistema proporciona una interfaz gráfica acorde para dicha acción.
Pre-Condiciones	• El usuario debe estar logueado al sistema. • Debe existir al menos una Representación en el sistema. • Debe existir al menos una Organización en el sistema. • Debe existir al menos una Categoría en el sistema.
Flujo de Eventos	**Flujo Normal** 1. El usuario ingresa nombre y descripción del tipo de normativa y selecciona la categoría a la cual pertenece. Selecciona cantidad de organizaciones participantes y el archivo xsd. 2. El sistema lista las representaciones existentes. 3. El usuario selecciona una representación, desde cual archivo se realiza, el tipo de transformación a utilizar (XSLT o Freemarker) y el archivo que realiza la transformación. 4. El usuario selecciona confirmar. 5. El sistema guarda en el sistema el nuevo Tipo de Normativa 6. Fin de caso de uso. **Flujo Alternativo 1 del paso 4**: Se desea agregar otra representación. 4.1. El usuario selecciona una representación, desde cual archivo se realiza, el tipo de transformación a utilizar (XSLT o Freemarker) y el archivo que realiza la transformación.

	4.2. Se vuelve al paso 4 del flujo normal.
Post-Condiciones	Se crea un Tipo de Normativa

Login Usuario

Caso de Uso	Login Usuario
Actor	Usuario Común/Administrador de Organización
Descripción	El caso de uso comienza cuando el actor ingresa su nombre de usuario y contraseña para acceder a la aplicación.
Pre-Condiciones	
Flujo de Eventos	**Flujo Normal** 1. El usuario ingresa su nombre de usuario y contraseña. 2. El sistema verifica la existencia del usuario y valida la contraseña. 3. El usuario accede a la página principal de la aplicación. 4. Fin del caso de uso. **Flujo Alternativo 1 del paso 2:** Nombre de usuario y/o contraseña no existente en el sistema. 2.1. Se vuelve al paso 1 del flujo normal.
Post-Condiciones	El usuario se loguea e ingresa a la aplicación.

Registro Usuario

Caso de Uso	Registro Usuario
Actor	Usuario Común/Administrador de Organización
Descripción	El caso de uso comienza cuando el actor desea registrarse en el sistema.
Pre-Condiciones	
Flujo de Eventos	**Flujo Normal** 1. El usuario ingresa un nombre de usuario, una contraseña y datos

	personales. 2. El usuario repite la contraseña para verificarla, 3. El sistema almacena el nombre de usuario y contraseña ingresados por el usuario. 4. Fin del caso de uso. **Flujo alternativo 1 del paso 3:** Nombre de usuario ya existente 3.1.1. Vuelve al paso 1 del flujo normal. **Flujo alternativo 2 del paso 3:** la contraseña y la verificación de contraseña ingresadas no coinciden. 3.2.1. Vuelve al paso 1 del flujo normal.
Post-Condiciones	El sistema registra un nuevo usuario.

Alta Normativa

Caso de Uso	Alta Normativa
Actor	Usuario Común/Administrador de Organización
Descripción	El caso de uso comienza cuando el actor quiere dar de alta una Normativa en su Organización. A tales efectos, el sistema proporciona una interfaz gráfica acorde para dicha acción.
Pre-Condiciones	• El usuario debe estar logueado al sistema. • Debe existir Tipo de Normativa compilable para la nueva Normativa.
Flujo de Eventos	**Flujo Normal** 1. El actor ingresa nombre y descripción para la normativa. Selecciona la/las organización/organizaciones participante/participantes. 2. El actor selecciona el archivo con el contenido de la normativa. 3. El sistema compila la normativa con algún tipo de normativa existente y lo valida. Y realiza las transformaciones correspondientes a dicho tipo de normativa.

	4. El actor agrega/modifica alguna representación y guarda. 5. El actor confirma la Normativa. 6. El sistema crea la nueva Normativa. 7. Fin del caso de uso.
	Flujo Alternativo 1 del paso 3: El sistema no valida con ningún tipo de Normativa. 3.1.1. El sistema compila con los tipo de normativa pero no valida con ninguno. 3.1.2. Se vuelve al paso 2 del flujo normal.
	Flujo Alternativo 2 del paso 3: Alguna de las transformaciones no se pueden realizar. 3.2.1. Alguna de las transformaciones no se pueden efectuar. 3.2.2. El sistema no muestra representación correspondiente. Se comunica al actor. 3.2.3. Se vuelve al paso 2 del flujo normal.
	Flujo Alternativo 1 del paso 5: Se quiere agregar/modificar representación. 5.1. El actor quiere agregar/modificar alguna representación. 5.2. Se vuelve al paso 4 del flujo normal.
Post-Condiciones	• Se crea una nueva Normativa para la Organización.

Aprobar Normativa

Caso de Uso	Aprobar Normativa
Actor	Administrador de Organización
Descripción	El caso de uso comienza cuando el actor desea aprobar una normativa.
Pre-Condiciones	• El actor debe estar logueado al sistema.
Flujo de Eventos	

<table>
<tr><td></td><td>
<table>
<tr><td>Flujo Normal</td></tr>
<tr><td>1. El sistema lista las normativas pertenecientes a la organización del Administrador.</td></tr>
<tr><td>2. El Administrador selecciona una normativa y la aprueba.</td></tr>
<tr><td>3. El sistema modifica la información sobre la aprobación de la normativa.</td></tr>
<tr><td>4. Fin de caso de uso.</td></tr>
</table>

<table>
<tr><td>Flujo Alternativo 1 del paso 2: Visualizar Normativa.</td></tr>
<tr><td>2.1. El Administrador selecciona una normativa y la visualiza.</td></tr>
<tr><td>2.2. Se vuelve al paso 2 del flujo normal.</td></tr>
</table>
</td></tr>
<tr><td>Post-Condiciones</td><td>• Se actualiza el estado de aprobación de la normativa.</td></tr>
</table>

Apéndice 2: Lenguajes de especificación de normativas

1. *Web Service Quality Description Language* (WSQDL)

El *Business Quality Group* incluye sólo el factor de calidad de valor de negocio, mientras que el *System Quality Group* está compuesto por la *Variant Quality Part* y por la *Invariant Quality Part*. La *Variant Quality Part* incluye factores de calidad cuyos valores pueden variar dinámicamente en tiempo de ejecución mientras se utiliza un servicio. La *Invariant Quality Part* se refiere a factores de calidad cuyos valores se determinan tan pronto como se completa el desarrollo del servicio: aquí se incluyen la calidad de interoperabilidad, la calidad de procesamiento de negocios, la calidad de gestión y la calidad de seguridad [33]. Estos elementos se muestran en la Figura 63.

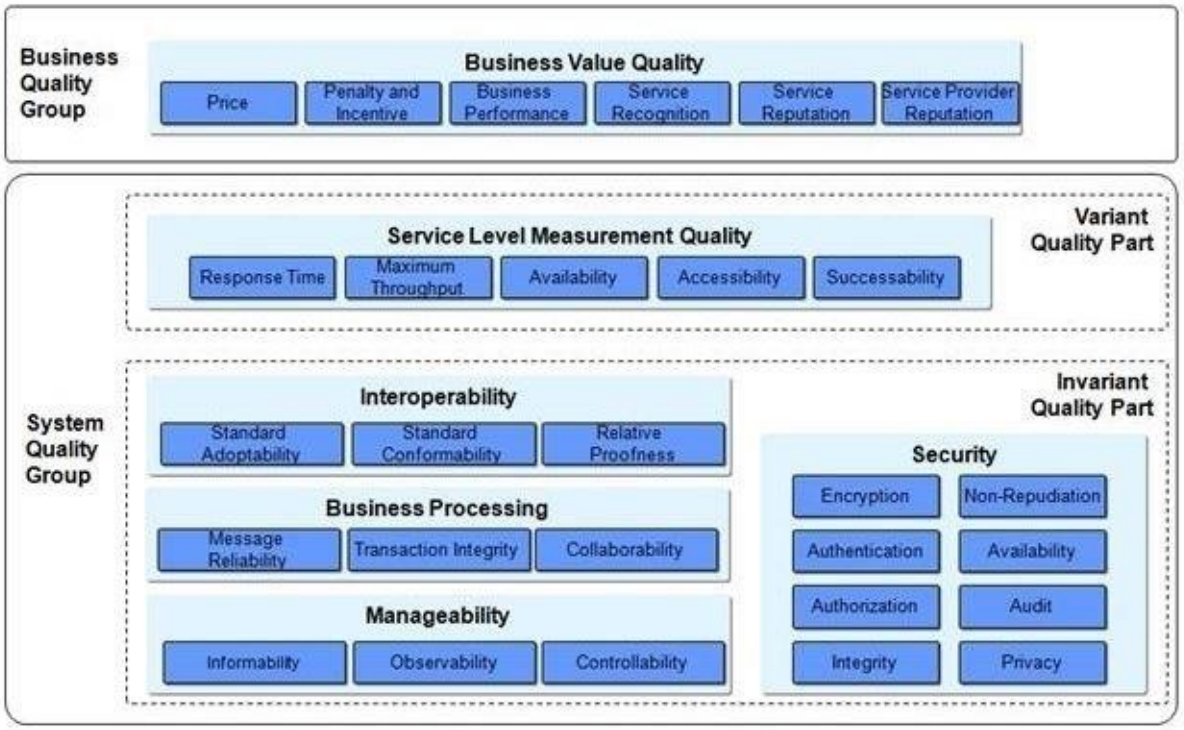

Figura 63 – Estructura de Web Service Quality Factor [33]

2. QoS *Modeling Language* (QML)

QML tiene tres mecanismos de abstracción principales para la especificación: *contract type*, *contract* y *profile*. QML permite definir tipos de contratos que representan aspectos específicos de QoS, como el rendimiento o la fiabilidad.

Un tipo de contrato define las dimensiones que pueden usarse para caracterizar un aspecto de QoS particular. Una dimensión tiene un dominio de valores que puede ordenarse. Hay

tres tipos de dominios: dominios establecidos, dominios enumerados y dominios numéricos. Un contrato es una instancia de un tipo de contrato y representa una especificación de QoS particular. Finalmente, los contratos de QML *profile* asocian contratos con interfaces, operaciones, argumentos de las operaciones y resultados de las mismas [35].

Las definiciones de QML en la Figura 64 incluyen dos tipos de contratos: Fiabilidad y Rendimiento. El tipo de contrato de fiabilidad tiene tres dimensiones. La primera representa el número de fallas por año. La segunda, el tiempo de reparación (TTR), representa el tiempo que tarda en reparar un servicio que ha fallado. Y la tercera, la disponibilidad, representa la probabilidad de que un servicio esté disponible. En este caso los valores mayores representan restricciones más fuertes, mientras que los valores más pequeños representan probabilidades más bajas y por lo tanto son más débiles [35].

```
type Reliability = contract f
  numberOfFailures: decreasing num eric no/year;
  TTR: decreasing num eric sec;
  availability: increasing num eric;
g;

type Performance = contract f
  delay: decreasing num eric msec;
  throughput: increasing num eric mb/sec;
g;

systemReliability = Reliability contract f
  numberOfFailures < 10 no/year;
  TTR f
    percentile 100 < 2000;
    mean < 500;
    variance < 0.3
  g;
  availability > 0.8;
g;

rateServerProfile for RateServiceI = profile f
  require systemReliability;
  from latest require Performance contract f
    delay f
      percentile 50 < 10 msec;
      percentile 80 < 20 msec;
      percentile 100 < 40 msec;
      mean < 15 msec
    g;
  g;

  from analysis require Performance contract f
    delay < 4000 msec
  g;
g;
```

Figura 64 – Ejemplo QML [35]

3. *Web Service Level Agreement* (WSLA)

El lenguaje de especificación WSLA se define como un esquema XML. WSLA puede ser utilizado tanto por el proveedor de servicios como por el cliente del servicio con el fin de configurar sus respectivos sistemas. Este proceso se llama despliegue. Cada organización utiliza su propia función de despliegue independiente [36].

En la Figura 65 se observan las tres partes importantes de un acuerdo de nivel de servicio.

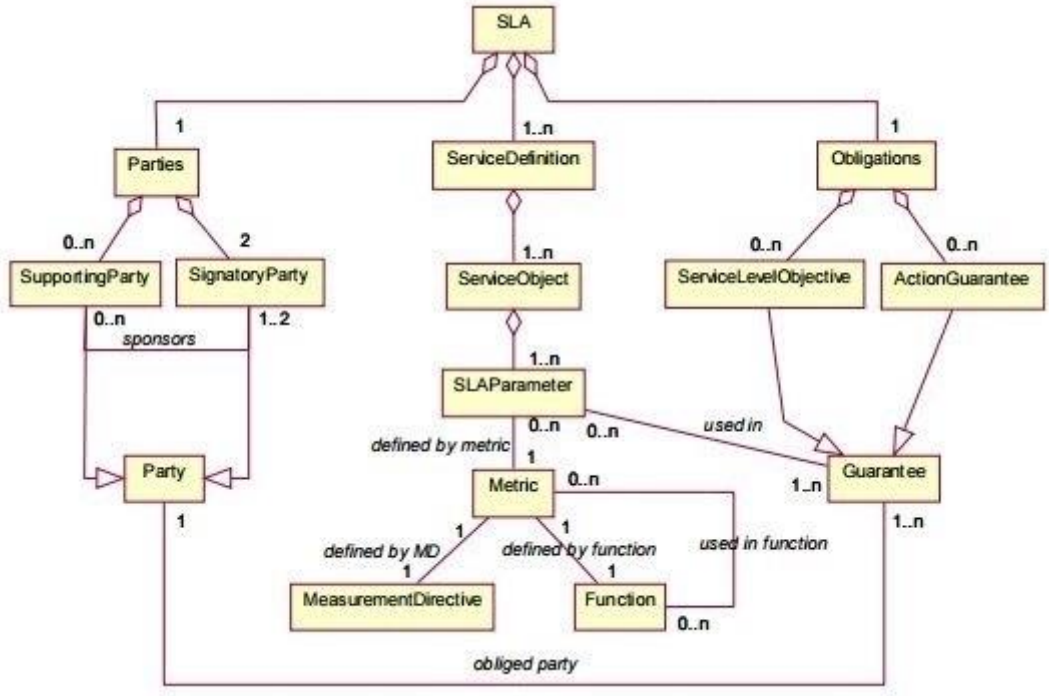

Figura 65 – Esquema Web Service Level Agreement [36]

***Parties*:** describen las partes que intervienen en la gestión del servicio web. Esto incluye las partes firmantes, así como las partes de soporte para actuar en nombre del proveedor de servicios.

***ServiceDefinition*:** describen los servicios sobre los que el acuerdo del WSLA actúa. Las definiciones de servicios se encargan de representar la comprensión común de las partes contratantes, en términos de operaciones y parámetros del servicio y sus métricas.

***Obligations*:** definen el nivel de servicio que se garantiza con respecto a los *SLAParameters* que fueron previamente declarados en la sección de definición de servicio.

Otros componentes importantes que se observan en la Figura 65 son:

***SLAParameter*:** Los parámetros son los principales elementos de descripción de un servicio. Un parámetro describe una propiedad observable de un servicio cuyo valor puede ser obtenido de una fuente de medición. Un *SLAParameter* se puede utilizar en un *ActionGuarantee* (garantía) del SLA.

Un *SLAParameter* tiene una métrica asignada. Se refiere a él utilizando su nombre único. La métrica define cómo se calcula el valor del *SLAParameter*. Esta métrica se utiliza para determinar el valor de *SLAParameter* en tiempo de ejecución.

***Metrics*:** son las definiciones de los valores de las propiedades del servicio que se miden a partir de una prestación de servicios, sistema o calculado a partir de otros parámetros y constantes. Las métricas son el instrumento clave para describir exactamente lo que significa *SLAParameters* especificando la forma de medir o calcular el valor del parámetro.

***Functions*:** específica la forma de calcular el valor de una métrica a partir de los valores de otras métricas y constantes.

En cuanto a las *obligations,* se visualizan en la Figura 65, dos conceptos relevantes:

ServiceLevelObjetive: expresa el compromiso de mantener un estado particular del servicio en un período determinado.

ActionGuarantee: expresa un compromiso de acción para llevar a cabo una actividad, en particular si se dio una determinada condición previa [36].

4. SLA *definition language*

SLAng satisface la necesidad de un lenguaje SLA para soportar la construcción de sistemas distribuidos y aplicaciones con características de QoS fiables. La estructura sintáctica y la semántica de SLAng se definen con referencia a un modelo de arquitectura de sistemas distribuidos [37].

La sintaxis SLAng se define mediante XML y el uso de estos esquemas favorece la integración con los lenguajes de descripción de servicios existentes. Por ejemplo, SLAng se puede combinar con WSDL y BPEL (que son especificados en XML) para obtener una solución completa de automatización de negocio [37].

SLAng define siete tipos diferentes de SLA, cuatro de los cuales son verticales y tres horizontales. Regulan los posibles acuerdos entre los diferentes tipos de partes identificadas en el modelo de referencia, es decir, Aplicación, Servicio Web, Componente, Contenedor, Almacenamiento y Red [37].

La Figura 66 muestra el nombre de los acuerdos de nivel de servicio que pueden ser contratados entre organizaciones, debidamente subdivididos en acuerdos verticales y horizontales [37].

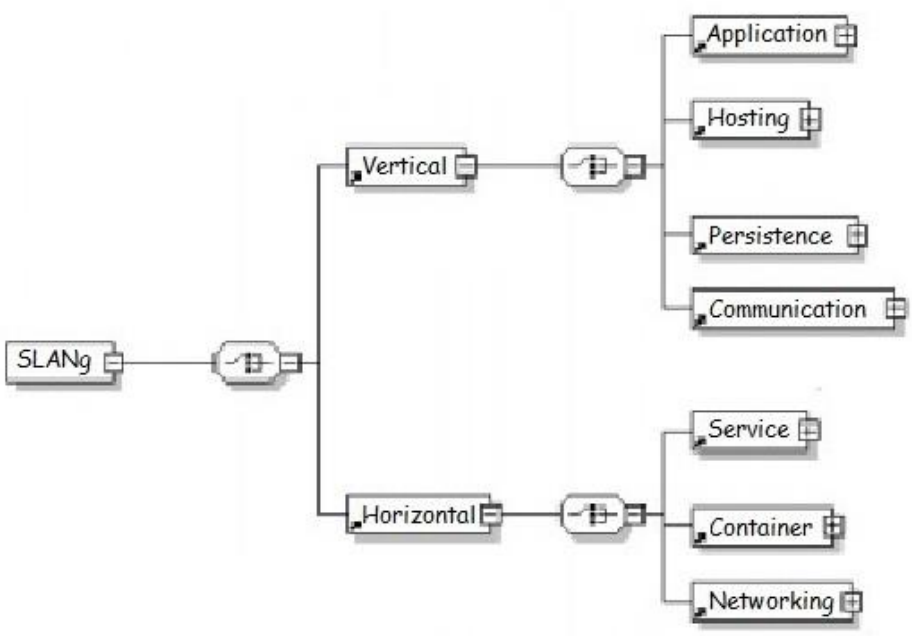

Figura 66 – SLAs Vertical y Horizontal [37]

Los SLA verticales son:

Aplicación: entre aplicaciones o servicios web y componentes.

Hosting: entre proveedores de contenedores y componentes.

Persistencia: entre un proveedor de contenedores y un *Storage Service Provisioning* (SSP).

Comunicación: entre proveedores de servicios de contenedores y de red [37].

Los SLA horizontales son:

Servicio: entre proveedores de servicios web y componentes.

Contenedor: entre proveedores de contenedores.

Redes: entre proveedores de red [37].

Apéndice 3. Trabajo relacionado

1. QoS *Based Web Service Selection*

Este trabajo [40] aborda temas de calidad de los servicios web. En particular, el trabajo propone una herramienta para monitorear y gestionar los servicios web distribuidos (*Web Services Distributed Management*, WSDM). Se propone una division en dos partes: *Management of Web Services* (MOWS) y *Management using Web Services* (MUWS).

MUWS, la gestión que utiliza web services, se puede utilizar para administrar y supervisar los recursos de TI y sus parámetros de QoS. Mientras que MOWS, se creó para la gestión de los endpoints de los web services. Los valores que se pueden utilizar como parámetros de QoS o para calcular parámetros son: número de consultas, tiempo de respuesta, entre otros [40].

2. SeaFlows Toolset: Compliance Verification Made Easy

SeaFlows es una herramienta que extiende el *process-aware information system* (PAIS) [62]mediante la funcionalidad de verificación de cumplimiento. Ejemplos de infraestructura proveniente de PAIS son: repositorio de actividad, herramienta de modelado de proceso y repositorio de modelado de proceso [41].

SeaFlows permite modelar la *compliance rule* como gráficas independientemente de los modelos de proceso específicos partiendo de un repositorio de actividades. Los modelos de proceso se pueden utilizar en las *compliance rules* para propósitos de documentación y verificación. En conclusión, la herramienta posee dos verificadores de *compliance*, el Comprobador de Cumplimiento Estructural y el Comprobador de Cumplimiento de Dato [41].

3. QUICKER: *A Model-driven QoS Mapping Tool for QoS-enabled Component Middleware*

QUICKER mejora el lenguaje de modelado de componentes independientes de plataforma (*Platform-Independent Component Modeling Language, PICML*) [42] con nuevas construcciones que permiten a los desarrolladores de sistemas especificar y analizar políticas de QoS de aplicaciones en un nivel de abstracción superior al utilizado por lenguajes de programación de tercera generación (como Java o C ++) o por declaraciones textuales (como XML) [43].

PICML permite especificar políticas de QoS de aplicación en un alto nivel de abstracción, es decir, centrarse en las características de QoS deseadas en una granularidad de componentes individuales. Estas especificaciones son las entradas para QUICKER [43].

El lenguaje de salida (*Component QoS Modeling Language,* CQML) de la transformación QUICKER captura opciones de configuración de QoS específicas de CCM (*CORBA component model*) para aplicaciones basadas en componentes [43].

4. PPINOT *Tool Suite: A Performance Management Solution for Process-Oriented Organizations*

PPINOT es un conjunto de herramientas y técnicas para la definición y análisis automatizado de indicadores de desempeño de procesos [44].

Los requisitos de desempeño de los procesos de negocios (*Business Process*) generalmente se especifican en términos de Indicadores de Desempeño de Procesos (*Process Performance Indicators,* PPI). Estos indicadores son métricas cuantificables que pueden ser medidas por datos generados dentro del flujo del proceso y que tienen como objetivo evaluar la eficiencia de los procesos de negocio [44].