Seguridad en Plataforma de Integración de Servicios Geográficos

Instituto de Computación

Facultad De Ingeniería

Universidad de la República

Autores

Eugenia Díaz

Juan Gonzalez

Leonardo Jorcin

Tutor

Ing. Bruno Rienzi

Resumen

Los Sistemas de Información Geográfica (SIG) han surgido para almacenar, manejar, analizar y presentar datos geográficos tanto a usuarios expertos como a usuarios comunes. En los últimos años las aplicaciones con capacidades geográficas se han incrementado notoriamente, por lo que nuevas soluciones de software, como los Servidores de Mapas o Internet Map Servers (IMS), han sido incorporadas en las empresas. Esto ha llevado al nacimiento de lo que se conoce como “Enterprise GIS”.

Aunque estos sistemas se basan en estándares para el procesamiento geoespacial distribuido e intercambio de datos, a través del uso de Web Services Geoespaciales, dichos estándares han dejado de lado deliberadamente la incorporación de mecanismos de seguridad, por lo que las alternativas de seguridad existentes son generalmente soluciones ad hoc creadas para aplicaciones particulares, o mecanismos muy básicos incorporados a los IMS.

En los últimos años se ha venido trabajando dentro del Laboratorio de Integración de Sistemas (LINS) en la explotación de los Enterprise Service Buses (ESB) como plataformas de middleware idóneas, entre otras cosas, para la incorporación de los Web Services Geoespaciales en aplicaciones empresariales, cuyos requerimientos funcionales y no funcionales son mucho más complejos de lo que puede satisfacer un IMS. Dentro de estos requerimientos, los concernientes a la seguridad, en particular, autenticación y autorización, son cada día más relevantes debido al creciente uso de Web services como el Web Feature Service (WFS), que permiten la edición de la información geográfica, por lo que es vital poder restringir quién puede hacer qué a través de estos Web services.

En este contexto se ha desarrollado una Plataforma Específica del Dominio GIS basada en un ESB, la cual apoyándose en sus capacidades de mediación provee una arquitectura de mecanismos reusables para abordar desafíos comunes que surgen al integrar sistemas empresariales tradicionales con Web Services Geoespaciales. Este trabajo apunta a incorporar mecanismos de seguridad basados en estándares, en particular de autenticación y autorización, en dicha plataforma.

Contenido

1 Introducción

1.1 Contexto y Motivación

Las organizaciones que, actualmente, brindan servicios en los cuales la información geográfica es enriquecida con datos empresariales, ofrecen a los usuarios más información en el momento de poder acceder a los recursos geográficos. Por ejemplo, el poder ubicar en una zona geográfica datos empresariales, que pueden ser públicos o privados, y así brindar un grado más de información.

Este tipo de servicios si bien son más útiles que los servicios enfocados solamente en brindar información geográfica, no proveen la capacidad de ofrecer un control de acceso a ciertas zonas geográficas y/o datos empresariales. Esto limita el tipo de acciones y operaciones que se pueden realizar sobre esta información.

Un ejemplo de esto sería el siguiente. Se considera una empresa en la cual se comparten datos relacionados a sus empleados. Estos datos indican los lugares por donde el empleado estuvo trabajando dentro de una zona geográfica específica. Estas empresas están interesadas en poder brindar la capacidad de acceder a los datos de forma controlada teniendo en cuenta el rol específico de sus empleados. Esto permitirá verificar qué roles tienen permisos sobre qué datos.

Generalmente los roles suelen usarse en aplicaciones empresariales para poder aplicar cierto control sobre datos o transacciones. Por ejemplo, una aplicación puede definir ciertos límites en la clase de datos que se van a ofrecer dependiendo de si el empleado que realiza la solicitud es un miembro del rol especificado. Los empleados podrían tener autorización para obtener datos que son limitados a una zona geográfica específica, los supervisores podrían tener un límite mayor y los gerentes podrían tener un límite aún mayor (o ningún límite).

Para lograr esto, se proveerá de una autenticación para cada empleado, los cuales a su vez tendrán un rol de autorización que les permitirá poder tener acceso a un determinado conjunto de información.

Con el propósito de poder cumplir con las necesidades planteadas en el ejemplo, se propone una solución con base en una implementación previa [21], la cual fue desarrollada utilizando mecanismos de integración orientados a servicios geográficos sobre un ESB [3].

Estos mecanismos brindan una solución a los problemas de integración de servicios empresariales en un contexto georreferenciado. Sin embargo no proveen el soporte necesario para brindar una solución de seguridad en el momento de acceder a los datos, por lo que es necesario agregar este tipo de características a la solución existente teniendo en cuenta lo mencionado anteriormente.

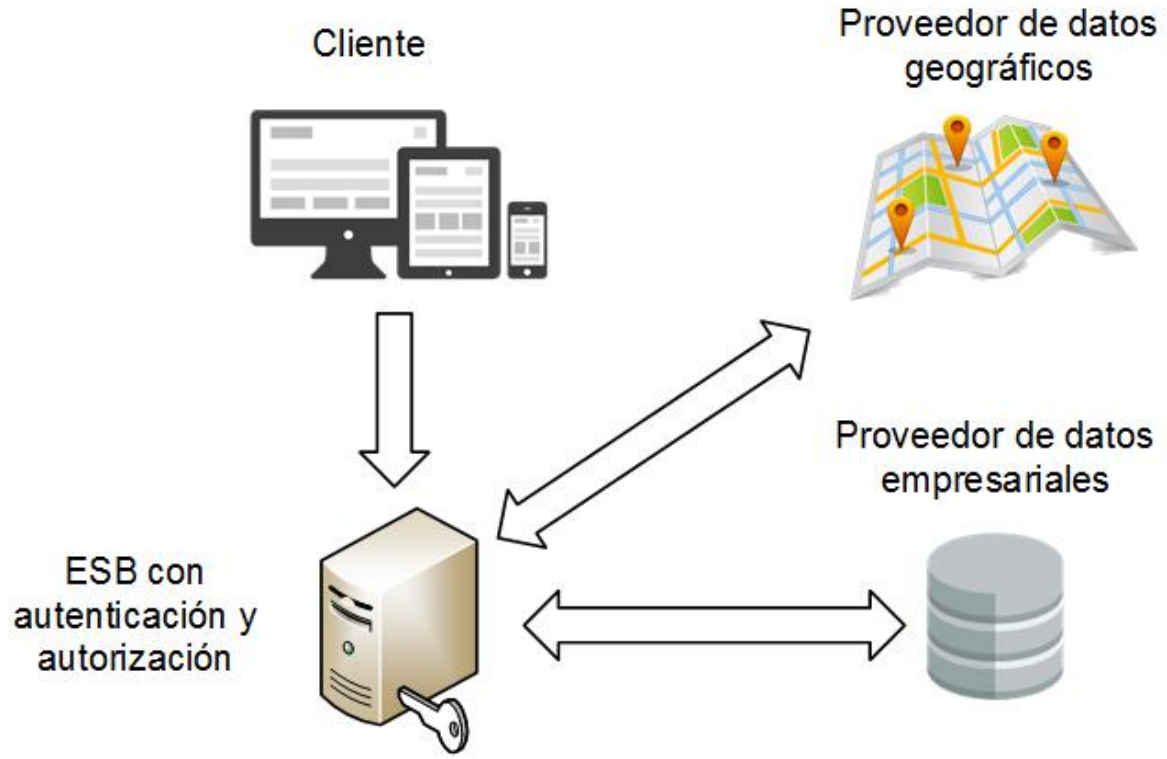

Figura 1.1 - Integración de datos con control de acceso

Teniendo en cuenta la solución planteada en la Figura 1.1 es interesante el poder desarrollar un sistema en el que se puedan extender las funcionalidades ya existentes, poder incorporar nuevas y además agregar una capa de seguridad a los datos geográficos enriquecidos provistos por la plataforma.

1.2 Objetivos

Se plantea como objetivo general de este proyecto incorporar mecanismos de seguridad basados en estándares, en particular de autenticación y autorización, a la Plataforma de Integración para Servicios Geográficos basada en Enterprise Service Buses (ESB). Esta implementación se debe realizar utilizando el producto JBoss ESB. Además se debe

permitir la composición de mecanismos existentes entre sí y con los nuevos mecanismos de seguridad que fueron mencionados anteriormente.

Para ello, se plantean los siguientes objetivos específicos:

- Estudio detallado sobre las alternativas más relevantes para incorporar mecanismos de seguridad a los Web services geoespaciales.
- Analizar, diseñar e implementar la incorporación a la plataforma de mecanismos de seguridad, en particular de autenticación y autorización.
- Realizar un diseño que permita la composición entre diversos mecanismos de manera flexible.
- Realizar un diseño que permita la extensibilidad de la plataforma.
- Reutilizar los mecanismos existentes
- Desarrollo de un caso de estudio que permita evaluar y demostrar la factibilidad de la plataforma propuesta.

1.3 Aportes

Existen una infinidad de sistemas que abarcan todas las áreas funcionales de una empresa con la tarea de ejecutar procesos de negocio, brindando una solución enfocada en la administración eficiente de recursos y servicios al cliente. Por otro lado existen sistemas que permiten analizar, consultar, manipular y desplegar información geográfica.

La solución planteada en al sección 1.1 nos permite poder unir estos dos sistemas, con el propósito de poder brindarle al cliente la posibilidad de acceder a los datos empresariales dentro de un contexto geográfico. Además ofrece la posibilidad de limitar el acceso a estos datos según el rol del cliente.

Por ejemplo, supongamos que tenemos una aplicación que permite para el común de la población identificar las rutas de evacuación, ubicación de centros de refugio y centros de atención médica en caso de emergencias. En la figura 1.2 se muestra un ejemplo.

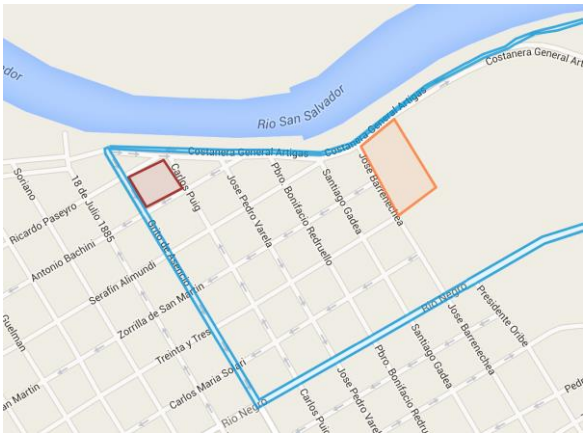

Figura 1.2 - Información geográfica sobre la zona de evacuación

Por otro lado se plantea un sistema que nos brinda la información de los lugares de interés en caso de emergencia como se muestra en la figura 1.3.

Lugar	Dirección
Centro Medico	Costanera General Artigas y Carlos P. Puig
Refugios	Costanera General Artigas y Jose Barrenechea
Ruta de evacuación 1	Costanera General Artigas
Ruta de evacuación 2	Grito de Asencio, Rio Negro

Figura 1.3 - Datos sobre la zona de evacuación

Un aporte sumamente valioso es poder unir estos dos tipos de información en uno solo y además poder mostrar sólo la información necesaria para los médicos y personas dedicadas a dar refugio a los evacuados. A su vez, también es posible brindar toda la información sin ningún tipo de límites para los funcionarios públicos encargados de solucionar el estado de emergencia en la zona.

1.4 Organización del documento

En el capítulo 2 se introducen los conceptos fundamentales que refieren a este trabajo. Estos son los Web services geoespaciales, en particular WMS y WFS, las arquitecturas orientadas a servicio (SOA) y la seguridad enfocada desde la autenticación y autorización. Luego se realiza un relevamiento de las tecnologías relacionadas con el control de acceso a la información y autenticación. Por último se evalúan trabajos relacionados a la temática de este trabajo.

En el capítulo 3 se expone la propuesta de arquitectura, fundamentando el diseño que se aplicó. Se detallan los componentes que están presentes en la solución y los mecanismos de la plataforma planteada.

El capítulo 4 se centra en la propuesta de solución desarrollada. Se detallan todos los mecanismos desarrollados, cómo están compuestos y detalles de implementación. Al final del capítulo se describe el conjunto de tecnologías utilizadas para el desarrollo de todos los componentes de la solución.

En el capítulo 5 se presenta el caso de estudio que se utilizó para realizar la evaluación de factibilidad de la plataforma desarrollada. Para esto se detalla la realidad involucrada, describiendo cada uno de los aspectos relevantes de la solución implementada. Además del caso de estudio se presentan un conjunto de pruebas realizadas sobre el sistema.

En el capítulo 6 se analizan los resultados obtenidos, las dificultades encontradas durante el transcurso del proyecto, las conclusiones globales y los trabajos a futuro que podrían realizarse.

2 Estado del arte

En este capítulo veremos conceptos sobre Web services (ver 2.1.1), Arquitectura orientada a servicios (ver 2.1.2), Enterprise Service Bus (ver 2.1.3) y Seguridad (ver 2.1.4). Luego haremos un relevamiento de tecnologías (ver 2.2), pasando por tecnologías de autenticación, autorización y seguridad en la comunicación (ver 2.3). Para finalizar describiremos los trabajos relacionados a nuestro proyecto (ver 2.4) y una breve síntesis del capítulo (ver 2.5).

2.1 Marco conceptual

A continuación se describen los principales conceptos que están vinculados a los Web services geoespaciales en un entorno empresarial como son los Enterprise Service Bus (ESB). Primero se definirán los Web services SOAP, REST y geoespaciales. Luego se realizará una introducción a las arquitecturas orientadas a servicios (SOA), seguido de una referencia al concepto de ESB. Por último se mencionan conceptos generales de seguridad.

2.1.1 Web services

La definición de Web services ha ido variando en el correr de los años. En el 2002 el Grupo de Trabajo de Arquitectura de Web services de la W3C [8] dio la siguiente definición: Un Web service tiene una interfaz que se describe en un formato procesable por máquina (específicamente WSDL). Otros sistemas interactúan con el Web service en una forma prescrita por su descripción utilizando mensajes SOAP (Simple Object Access Protocol), por lo general transportado utilizando HTTP con una serialización XML junto con otros estándares relacionados con la web[1].

Esta definición fue extendida en el 2004, añadiendo el concepto de REST. En esta actualización se identifican dos grandes clases de Web services:

- Web services REST-compliant, en el que el objetivo principal del servicio es manipular representaciones de recursos web utilizando un conjunto uniforme de las operaciones sin estado.
- Los Web services arbitrarios, en el que el servicio puede exponer un conjunto arbitrario de operaciones [32].

[1] https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice

El término REST (Representational State Transfer) fue definido en un principio como un estilo arquitectónico, desarrollado como un modelo abstracto de la arquitectura Web para guiar un nuevo diseño y definición del protocolo HTTP y de las URIs (Identificador Uniforme de Recursos) en un artículo publicado por Roy Fielding en el año 2000 [33].

Actualmente se utiliza en el sentido más amplio para describir cualquier interfaz entre sistemas que utilicen directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos en algún formato, entre los que se encuentra XML y JSON[2], sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP.

Los Web services RESTful fueron construidos para que funcionen mejor en la Web. REST es un estilo arquitectónico que especifica constantes, con una interfaz uniforme que introduce propiedades deseables al aplicarla a los Web services, como son la performance, la escalabilidad y la capacidad de modificación, que permiten a los servicios trabajar mejor en la Web.

En el estilo arquitectónico REST los datos y las funciones son consideradas recursos y son accedidos usando URIs, típicamente llamados links en la Web. Estos recursos son accedidos utilizando un conjunto simple de operaciones bien definidas. Utilizan la arquitectura cliente/servidor y están diseñados para que el protocolo de comunicación sea sin estado (stateless), típicamente HTTP. En este estilo arquitectónico clientes y servidores intercambian las representaciones de los recursos utilizando una interfaz y protocolo estandarizado.

Los siguientes principios aseguran que las aplicaciones RESTful sean simples, livianas y rápidas [35]:

- *Identificación de recursos a través de URIs*: Los recursos son identificados a través de URIs lo cual provee un direccionamiento global para los recursos y ubicación de los servicios.
- *Interfaz uniforme*: Los recursos son manipulados utilizando un conjunto fijo de operaciones: PUT, GET, POST y DELETE, entre otros. PUT crea un nuevo recurso

[2] Es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript

que luego puede ser eliminado utilizando DELETE. GET devuelve el estado actual de algún recurso. POST transfiere un nuevo estado a algún recurso.

- *Mensajes auto-descriptivos*: Los recursos son desacoplados de su representación por lo cual su contenido puede ser accedido utilizando varios formatos, entre ellos: HTML, XML, texto plano, PDF, JPEG, JSON, etc. La metadata del recurso está disponible y es utilizada por ejemplo para el caché, detección y transmisión de errores, negociación para la representación del recurso mediante un formato en particular y autenticación o control de acceso.

- *Interacciones con estado a través de hipervínculos*: Toda interacción con un recurso es sin estado; queriendo decir, que los pedidos de mensajes son auto-contenidos. Las interacciones con estado están basadas en el concepto de transmisión explícita de estado. Existen varias técnicas para intercambiar el estado, como pueden ser la sobreescritura de las URIs, cookies y campos de formulario ocultos. El estado puede ser embebido en la respuesta de un mensaje para marcar el futuro estado válido de una interacción.

Los Web services del estilo SOAP son capaces de ofrecer algunas ventajas con respecto a los modelos tradicionales de arquitectura distribuidas como *CORBA*[3] *y EJB*[4]. Algunas de las ventajas presentadas por SOAP son:

- Permitir la coexistencia de diferentes tecnologías.
- Comunicarse entre aplicaciones que se han desarrollado mediante diferentes lenguajes de programación.
- Permitir la transmisión mediante el protocolo HTTP.
- Estandarizar la localización de los servicios, entre otros.

Se basan en estándares y protocolos abiertos. A continuación se describen los estándares de forma breve:

- *eXtensible Markup Language (XML)*: Basado en marcas y etiquetas, es muy frecuente el uso de este metalenguaje para crear otros lenguajes con entidad propia.

[3] Es un estándar que permite que diversos componentes de software escritos en múltiples lenguajes de programación y que corren en diferentes computadoras, puedan trabajar juntos.

[4] Es una de las interfaces de programación de aplicaciones (API) que forman parte del estándar de construcción de aplicaciones empresariales de Java.

Su simpleza, facilita su uso fundamentalmente en el intercambio de una gran variedad de datos.

- *Simple Object Access Protocol (SOAP)*: Este protocolo permite realizar intercambios de información entre diversas aplicaciones situadas en entornos que están descentralizados y se encuentran distribuidas. Los diferentes mensajes, codificados en XML, se integran dentro de mensajes SOAP. Hay multitud de tipos de mensajes que se pueden integrar dentro de SOAP (respuestas tras el uso de un servicio, información de errores, estado del servicio, enlaces, datos distintos al formato XML (MIME). SOAP es independiente del sistema permitiéndole comunicar aplicaciones que se encuentren implementadas en diferentes lenguajes de programación. También puede atravesar firewalls corporativos y ofrece la interoperabilidad de las aplicaciones.

- *Universal Description Discovery and Integration (UDDI)*: Se encarga de la publicación, localización y enlazado de los Web services. El principal objetivo que tiene UDDI es permitir, mediante el uso de unos criterios de selección para la búsqueda, el localizarse a las diferentes entidades de negocio.

- *Web Service Description Language (WSDL)*: Es el estándar que se utiliza para describir un Web Service. Está basado en XML y permite especificar cómo deben representarse los parámetros, tanto de entrada como de salida, en una invocación de tipo externo al servicio. Permite comprender cómo operar con el Web service, a los diferentes clientes.

En la figura 2.1 se muestra la arquitectura de los Web services SOAP.

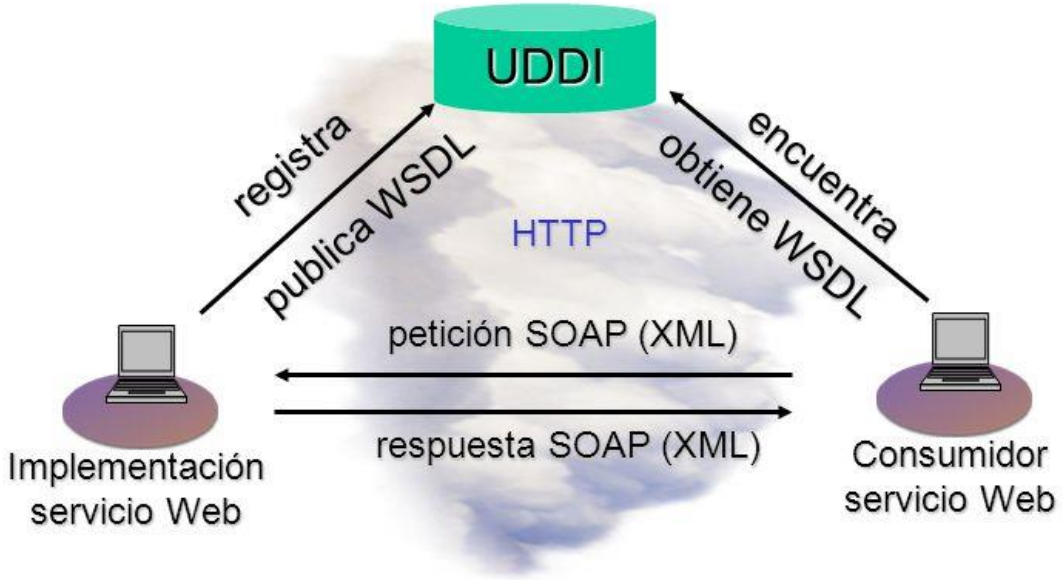

Figura 2.1 - Arquitectura de los Web services [51]

2.1.1.1 Web services geoespaciales

La particularidad de estos Web services es que están enfocados a la información geoespacial. Esto permite que los datos puedan ser ubicados en el espacio, junto con sus características geométricas. Los datos espaciales tienen tres dimensiones: espacial, temporal y temática. La dimensión espacial a su vez agrega la complejidad del ciclo de vida de estos datos [3].

La organización Open Geospatial Consortium (OGC)[5] es la que propone los estándares de tecnología sobre datos geoespaciales. Dentro de los estándares que propone la OGC se encuentran estándares para la representación, almacenamiento e intercambio de información geográfica, como también una Arquitectura de Referencia y múltiples estándares de diferentes tipos de servicio. La mayoría de los Web services que trabajan con datos espaciales son “Data Services”, es decir, poseen interfaces bien conocidas y estandarizadas y solamente cambia el conjunto de datos que brindan.

La OGC define diversos estándares para poder acceder a información geográfica, dentro de los que se encuentran el estándar WMS (Web Map Service), el estándar WFS (Web Feature Service) y WCS (Web Coverage Service) [30]. El protocolo CSW (Catalog Service for the Web) es otro de los estándares, el cual se utiliza para descubrir las características de un

[5] http://www.opengeospatial.org

servicio de información geográfica, en particular las características de la información que provee.

Web Map Service (WMS)

WMS produce dinámicamente mapas de datos georreferenciados a partir de la información geográfica distribuida. Este estándar define un "mapa" representando la información geográfica y lo brinda en un formato de archivo digital, lo cual es conveniente para exhibir en la pantalla de un computador. Los formatos en los cuáles WMS genera los mapas son formato de imágenes bien conocidos como PNG, GIF o JPEG. También se brinda la opción de generarlos en formato SVG (Scalable Vector Graphics) o WebCGM (Web Computer Graphics Metafile).

WMS ofrece tres métodos que permiten realizar las siguientes acciones para publicar los datos geográficos:

- Devolver metadatos del nivel del servicio (GetCapabilites).
- Devolver un mapa cuyos parámetros geográficos y dimensionales han sido bien definidos (GetMap).
- Devolver información de características particulares mostradas en el mapa (GetFeatureInfo).

Estas operaciones pueden ser invocadas utilizando un navegador, realizando dichas peticiones en forma de URLs[6]. El contenido de estas URLs depende de la operación que se está solicitando. Cuando se solicita una capa geográfica, la URL indica qué información debe ser mostrada en el mapa, la porción que se debe dibujar, el sistema de coordenadas de referencia y el ancho y altura de la imagen. Si dos o más capas se producen con los mismos parámetros geográficos se solapan los resultados para producir un mapa compuesto.

Para que todos los mapas subyacentes puedan ser visibles se utilizan los formatos de imagen transparentes. También es posible solicitar mapas individuales de diversos servidores. Esto permite crear una red de servidores distribuidos de mapas para que los clientes puedan crear mapas a medida. Existen también aplicaciones de software libre

[6] Un localizador de recursos uniforme URL es un identificador de recursos uniforme (Uniform Resource Identifier, URI) cuyos recursos referidos pueden ser variables en el tiempo.

(GRASS[7], uDIG[8], gvSIG[9], etc) que permiten el acceso avanzado a la información remota, agregando la posibilidad de poder cruzarla con información local y así disponer de varias herramientas SIG. Un Sistema de Información Geográfica (SIG) es cualquier sistema de información capaz de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada.

Para que sea posible la interoperabilidad de los servicios de mapas, es necesario que los WMS cumplan ciertas normas. La Organización de Estandarización (ISO) ha desarrollado la norma 19128 (Web Map Service Interface) basándose en la especificación *"Web Map Service Implementation Specification of OGC"* [31].

Web Feature Service (WFS)

Define un protocolo para consultar y modificar información geográfica en GML[10].

Pensado para ofrecer acceso a datos geográficos a otras aplicaciones ya que toda la información que se envía o se recibe está en formato XML. Se aplica solamente a información geográfica vectorial.

Los WFS se pueden clasificar según las operaciones que soportan:

- *WFS Básico*: DescribeFeatureType y GetFeature, solamente soporta consulta de datos.
- *WFS con Xlink*: Básico + GetGmlObject, puede solicitar datos adicionales a otros servidores.
- *WFS Transaccional*: Básico + Transaction + LockFeature, permite modificar los datos.
- *WFS Transaccional con Xlink*: puede hacer actualizaciones de datos y además consultar a otros servidores.

---

[7] Es un software SIG. Puede soportar información tanto raster como vectorial y posee herramientas de procesado digital de imágenes. https://grass.osgeo.org/

[8] Aplicación GIS de escritorio. http://udig.refractions.net/

[9] http://www.gvsig.com/

[10] Es un sublenguaje de XML para el modelaje, transporte y almacenamiento de información geográfica.

2.1.2 Arquitecturas orientadas a servicios

Las arquitecturas empresariales han evolucionado paulatinamente a medida que crecieron las demandas del mercado. Esto llevó a que diferentes tipos de aplicaciones e infraestructuras se actualicen para acompasar dichos cambios. Desde el principio, el objetivo detrás de las tecnologías de la información (TI) fue asumir por parte de las organizaciones la realización de un conjunto de actividades humanas complejas (y de gran escala) de una forma eficiente.

Las arquitecturas SOA logran facilitar el desarrollo de las aplicaciones empresariales como servicios de negocio modulares que pueden integrarse de forma fácil y rápida. La reutilización, la facilidad de mantenimiento, y una mejor visibilidad de los negocios, son sus principales beneficios.

A continuación se listan los principios fundamentales de la arquitectura SOA [2]:

- Contratos estandarizados de servicios.
- Servicios con bajo acoplamiento: para mantener baja la dependencia entre cada servicio.
- Reusabilidad de los servicios.
- Abstracción de los servicios: los contratos de los servicios contienen información esencial y la información acerca de los servicios es limitada a lo que se publica en los contratos de servicios.
- Autonomía: los servicios ejercen un alto nivel de control sobre su entorno y tiempo de ejecución subyacente.
- Componibilidad: los servicios son participantes eficaces de composición, independientemente del tamaño y la complejidad de la composición.
- Detección de servicios: los servicios se complementan con metadatos los cuales pueden ser descubiertos e interpretados eficientemente.
- Servicios sin estado: los servicios minimizan el consumo de recursos mediante el aplazamiento de la gestión de la información de estado cuando es necesario.

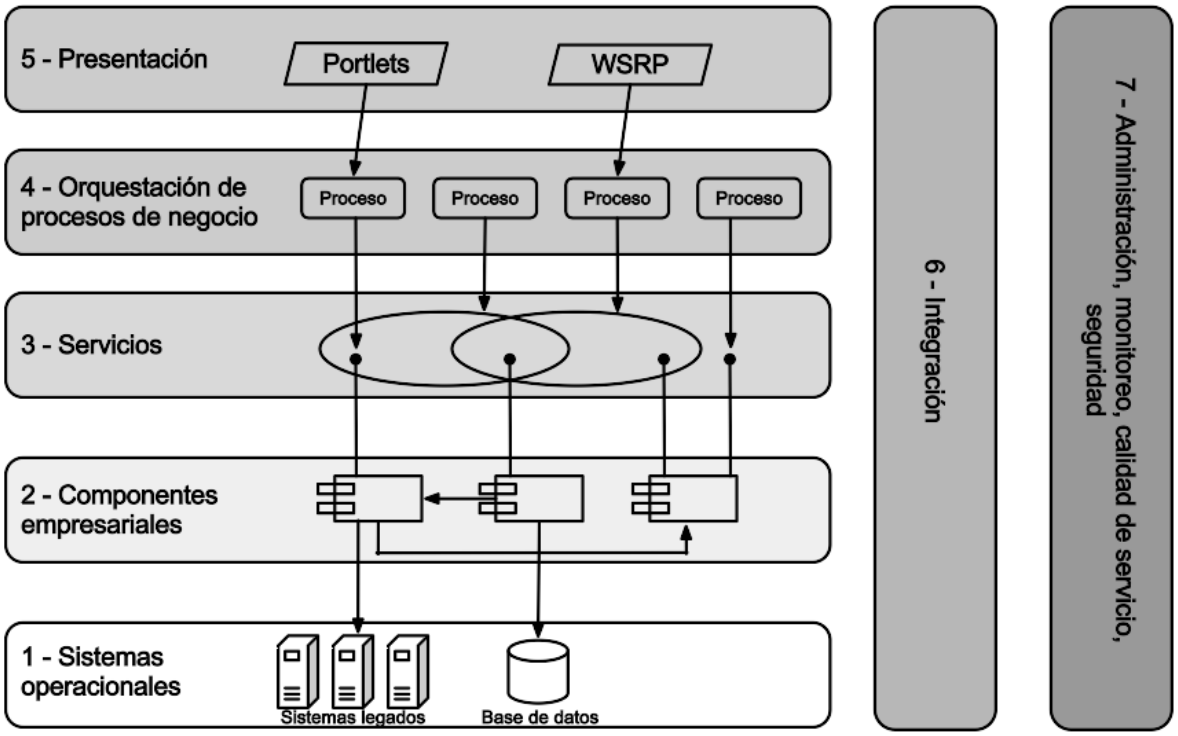

Figura 2.2 - Arquitectura en capas de SOA [2]

En la figura 2.2 se presenta la Arquitectura en capas de SOA. A continuación se realiza una breve descripción de las capas que fueron definidas en la figura:

- *Sistemas operacionales*: Esta capa representa el punto de intersección entre la infraestructura de tiempo de ejecución real y el resto de la arquitectura orientada a servicios que se ejecuta en dicha infraestructura. Además, es el punto de integración para una construcción de una Infraestructura Subyacente como Servicio (IaaS) y el resto de la arquitectura en el contexto más amplio de la computación en la nube.

- *Componentes empresariales*: Es la responsable de asegurar que las funcionalidades que ofrecen los servicios de la capa inferior puedan ser utilizados, y a su vez garantizar cierta calidad de servicio (QoS). Los componentes implementados aquí llevan a cabo los principales flujos del negocio de las organizaciones, por eso en general deben garantizarse ciertos niveles de servicio (SLA).

- *Servicios*: En esta capa residen los servicios que el negocio pretende exponer, que son utilizados para construir los servicios, tales como librerías funcionales y técnicas, interfaces tecnológicas, etc.

- *Orquestación de procesos de negocio*: En esta capa se exponen las orquestaciones de los servicios y se define el proceso en términos del negocio y sus necesidades, que varía en función del negocio.

- *Presentación*: Es la capa en la que los usuarios pueden interactuar con la arquitectura SOA invocando los procesos de negocio. Pueden existir diversas interfaces de usuario para consumir las funcionalidades expuestas por las capas inferiores.

- *Integración*: Es una capa transversal que permite y proporciona la capacidad para mediar, que incluye la transformación, enrutamiento y conversión de protocolos para el transporte de solicitudes de servicio desde el solicitante de servicio al proveedor de servicios correcto. La capa de integración es, entre otros objetivos, responsable de mantener la coherencia de la solución en presencia de un sistema de acoplamiento flexible. La integración que se produce aquí es principalmente la integración entre los componentes de servicio, el servicio, y las capas de proceso.

- *Calidad del servicio (QoS)*: Proporciona los medios para garantizar que una arquitectura cumple sus requisitos con respecto a: monitoreo, confiabilidad, disponibilidad, capacidad de gestión, transaccionalidad, mantenibilidad, escalabilidad, seguridad, etc.

2.1.3 Enterprise Service Bus

Existen diferentes definiciones de que es un Enterprise Service Bus (ESB), dependiendo de la bibliografía consultada o del proveedor del software. No hay acuerdo sobre si se debe definir un bus de servicios de empresa como un estilo de arquitectura, como un producto de software o como un grupo de productos de software. Entre otras cosas, un ESB se define como [3]:

- Un estilo de arquitectura de integración que permite la comunicación a través de un bus de comunicación común que consiste en una variedad de conexiones punto a punto entre proveedores y usuarios de servicios.

- Una infraestructura que una empresa utiliza para la integración de servicios en el entorno de aplicaciones.

- Un patrón de arquitectura que permite la interoperabilidad entre entornos heterogéneos, con orientación de servicio.

Se podría resumir en que el ESB es un modelo de arquitectura de software utilizado para diseñar e implementar la comunicación entre aplicaciones de software en una arquitectura SOA.

Es uno de los exponentes más avanzados dentro del grupo de las plataformas denominadas middleware[11]. Estas plataformas son sistemas de software complejos que combinan las características de los sistemas de middleware más simples. Esto lo consiguen brindando mecanismos de integración de alto nivel y con una gran variedad de formas de conexión [4].

Los ESB pueden ser definidos basándose en las principales características que estos brindan. A continuación se describen algunas de estas capacidades [5].

2.1.3.1 Ruteo de mensajes

Determinar el destino final de un mensaje entrante es una funcionalidad importante de un ESB [6]. En la mayoría de los proyectos de integración, una vez que un mensaje llega al ESB, este puede tomar diferentes rutas (a diferentes destinos) dependiendo de las decisiones que el ESB pueda tomar. Este ruteo de mensajes, se puede definir basándose en el emisor, en reglas estáticas, dinámicas o en una combinación de algunas de ellas. De esta manera, el ESB también puede actuar como filtro, o incluso agregar mensajes dentro del flujo de comunicación [7].

2.1.3.2 Orquestación de servicios

Con esta funcionalidad que brinda el ESB se permite abstraer la lógica de negocio de la lógica de integración. Esto permite separar el servicio de los métodos de comunicación utilizados, los formatos de mensaje y los protocolos. Además, habilita a las invocaciones de servicios a ser independientes de su ubicación, dado que el ESB oculta la ubicación del proveedor.

2.1.3.3 Transformación de mensajes

El ESB provee la funcionalidad para poder transformar mensajes de un formato a otro basado en estándares como XSLT[12] y XPath[13]. Es una capacidad bastante utilizada en los

[11] Es el software que proporciona un enlace entre aplicaciones de software independientes.

[12] Es un estándar de la W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML.

ESB porque es bastante frecuente que el formato del mensaje de entrada no sea el mismo que el que espera la aplicación destino.

2.1.3.4 Seguridad

Un ESB debe proveer mecanismos de autenticación, autorización y funcionalidad de cifrado para prevenir el uso malicioso a través de los mensajes entrantes, así como la asegurar los mensajes salientes para satisfacer los requisitos de seguridad del proveedor del servicio.

2.1.4 Seguridad

En esta sección se presentan algunos conceptos generales de seguridad, en particular nos centraremos en la autenticación y la autorización. La seguridad se puede describir considerando los aspectos funcionales y los no funcionales [9].

A continuación se definen los aspectos funcionales de la seguridad. Para cada uno de ellos se repasa su definición, y las estrategias de diseño e implementación, seguidas por las arquitecturas tradicionales contrastándolas con las estrategias SOA. Se definen las arquitecturas tradicionales como aquellas que no fueron pensadas para dar soporte a la reutilización, ni la composición de servicios.

2.1.4.1 Autenticación

Es el proceso de verificar la identidad de un usuario. La autenticación puede ser separada en 3 tipos o métodos de autenticación, las que son definidas en el cuadro 2.1.

Tipo de evidencia	Descripción	Ejemplos
Lo que se conoce	El usuario presenta uno o más secretos que la aplicación espera que sepas	Usuario y contraseña, preguntas cuya respuestas solo las conoce el usuario
Lo que se tiene	El usuario tiene posesión de algo que la aplicación espera que tenga	Token de seguridad
Lo que se es	El usuario demuestra su identidad con evidencia biométrica	Huellas dactilares, escaneo de retina, detección de voz

[13] Es un lenguaje que permite construir expresiones que recorren y procesan un documento XML.

Cuadro 2.1 - Categorías de autenticación [28]

Estrategia de autenticación tradicional

Existen diversos mecanismos utilizados para realizar la autenticación de un usuario en un sistema. Dos alternativas comúnmente adoptadas son:

- Que el sistema posea toda la información necesaria como para poder autenticar a un usuario.
- Utilizar un protocolo que permita manejar la información de los usuarios de forma centralizada, un ejemplo típico es Lightweight Directory Access Protocol (LDAP) [24].

Más allá de la elección de qué mecanismo utilizar, termina siendo la aplicación la que en última instancia decide qué usuarios pueden acceder.

Estrategia de autenticación en SOA

La estrategia de autenticación para SOA varía según el escenario en el que se esté, los posibles escenarios son mostrados en el cuadro 2.2.

Descripción del escenario	Posible estrategia de autenticación
El servicio es invocado directamente por una aplicación cliente perteneciente al mismo dominio que el servicio.	Autenticarse a través del directorio corporativo de una organización, como por ejemplo LDAP.
El servicio es invocado por otro servicio o una aplicación compuesta de varios servicios pertenecientes al mismo dominio.	Si el invocador es otro servicio con los mismos requisitos de autenticación, el servicio invocado puede confiar en la autenticación ya hecha por el invocador. De otro modo, el servicio invocado puede re-autenticar al usuario contra el directorio corporativo. Si la aplicación que invoca es una aplicación compuesta, el servicio invocado puede confiar en la aplicación compuesta o en su defecto realizar la autenticación del usuario contra el directorio corporativo.
El servicio es invocado por una aplicación que no pertenece a su propio dominio	Confiar en la autenticación del usuario que realizó contra el otro dominio.

Cuadro 2.2: Posibles estrategias de autenticación dependiendo del escenario [28]

Para este requerimiento se ve claramente que la estrategia para el caso SOA, si bien puede variar dependiendo del escenario que se presente, intenta descentralizar la lógica de autenticación de las aplicaciones, tratando de llevar a su máxima expresión la reutilización de dichos servicios.

Más allá de que los escenarios vistos anteriormente son habituales en SOA, no son los únicos, ni tampoco las soluciones planteadas son las óptimas en todos los casos donde se presenten estos escenarios. En la sección 2.2.1.1 se describe el estándar Security Assertion Markup Language (SAML) [18], que entre otras funcionalidades da soporte a la comunicación de información de autenticación entre componentes.

2.1.4.2 Autorización

La autorización es la función de especificar permisos de acceso sobre recursos. Este concepto está relacionado con el de autenticación, ya que primero es necesario autenticarse para luego saber qué permisos tendrá el usuario autenticado sobre los recursos del sistema.

Estrategia de autorización tradicional

Al igual que para la autenticación, en la estrategia de autorización tradicional, las aplicaciones en general tienen su propio control de acceso, o sea, la lógica para decidir o no el acceso a un recurso está dentro de la propia aplicación. Los dos modelos de acceso más comunes son:

- Role-Based Access Control (RBAC) [45]
- Access Control List (ACL) [46]

En RBAC, los permisos para cada acción contra un recurso son concedidos a uno o más roles. Por ejemplo, en una aplicación de gestión de proyectos, el rol de gerente es requerido para escalar una tarea. La información de los roles concedidos a cada usuario puede ser mantenida por ejemplo en un directorio LDAP.

El control de acceso ACL tiene un enfoque diferente. Los administradores asocian una lista de reglas para cada recurso. Una regla declara si se concede o se deniega el permiso para ejecutar determinada acción sobre determinado recurso.

Estrategia de autorización en SOA

Típicamente las aplicaciones con arquitecturas SOA exponen determinadas funcionalidades que se componen de la invocación de otros servicios. Si se eligiera la estrategia tradicional, se debería idealmente chequear el control de acceso de cada servicio que constituye la funcionalidad expuesta previa a su ejecución.

Esto no es posible en general ya que muchas veces no se tiene por qué conocer como los servicios realizan dichos chequeos. Por tanto, para los servicios SOA es aconsejable no utilizar diseños de autorización que incluyan esta tarea dentro del servicio. Esto puede llevar a que la reutilización se vea seriamente afectada, fundamentalmente para los casos en que se quisiera cambiar la estrategia de autorización.

Los Web services requieren objetivos más específicos de seguridad como los que se detallan a continuación:

- Es necesario asegurar que existe una autenticación mutua entre el cliente que accede a los Web services y el proveedor de dichos servicios.

- Se debe mantener una política de autorización del acceso a recursos y, más importante, a operaciones y procesos en un entorno en el que debe administrarse y controlarse el acceso de clientes, proveedores, vendedores, competidores y los posibles ataques que reciban de personal externo.

- Mantener al cliente identificado, de manera que se identifique una sola vez y pueda acceder a servicios en diversos sistemas.

- Controlar y asegurar la confidencialidad de los datos intercambiados. SOAP no tiene la capacidad para cifrar la información para que viaje de forma segura a través de la red. Es necesario asegurar la comunicación con algún estándar que permita crear un canal seguro de comunicación. El estándar ya firmemente establecido de creación de canales seguros SSL y el cifrado de partes específicas de documentos mediante el cifrado XML son las direcciones que se están siguiendo en este terreno.
- Se debe asegurar la integridad de los datos, de manera que estén protegidos a los posibles ataques o a manipulaciones fortuitas. En este campo se está utilizando el estándar de firmas XMLDSIG[14], que permiten la firma de partes específicas del documento XML.

[14] Es una recomendación del W3C que define una sintaxis XML para la firma digital

- Comprobar que no se repudian las operaciones, para lo cual es necesario mantener firmas en XML.

2.2 Relevamiento de tecnologías

En esta sección se detallan las tecnologías que fueron relevadas para diseñar la autenticación y autorización de los Web services.

2.2.1 Tecnologías de autenticación

2.2.1.1 SAML

Es un formato estándar abierto basado en XML para el intercambio de datos de autenticación y autorización entre partes, en particular, entre un proveedor de identidad y un proveedor de servicios.

SAML fue incorporada a WS-Security (se profundiza en este protocolo en la sección 2.3.1) por parte del comité de seguridad de los Web services de OASIS ya que aborda la necesidad de los Web services de identidad portable. La Identidad de los usuarios en un sistema es fundamental para la seguridad. Por lo que una medida para determinar si un sistema es seguro es si las identidades de los todos los usuarios son conocidas y las de los intrusos son bloqueadas [14]. La identidad de los usuarios válidos debe trasladarse cuando la información se mueve de un dominio de seguridad a otro. El hecho de que los Web services se utilizan atravesando dominios de seguridad hace que la confianza portable sea un requisito para la seguridad de los Web services.

El requisito más importante que aborda SAML es la sesión web única en un navegador (Single Sign-On SSO). SSO es un procedimiento de autenticación que habilita al usuario a acceder a varios sistemas con una sola instancia de identificación.

La especificación SAML define tres roles: el Director (normalmente un usuario), el Proveedor de Identidad (IP), y el Proveedor de Servicios (SP). En el caso de uso abordado por SAML, el director solicita un servicio del proveedor de servicios. Los proveedores de servicios obtienen una afirmación de identidad del proveedor de identidad. Sobre la base de esta afirmación, el proveedor de servicios puede tomar una decisión de control de acceso.

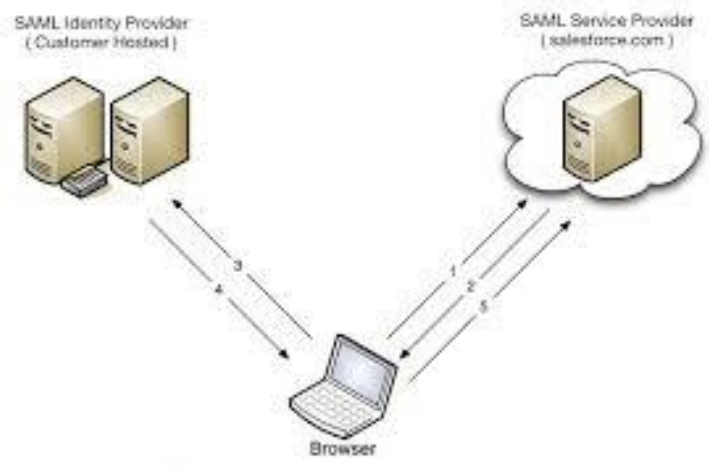

Figura 2.3 - Interacción SAML [27]

SAML es un estándar adecuado para proveer los mecanismos de afirmación (assertion) y protocolo que son necesarios para el eXtensible Access Control Markup Language (XACML). Define esquemas destinados a ser utilizados en la solicitud y la respuesta con varios tipos de afirmaciones de seguridad. XACML es un lenguaje estándar de OASIS [15] que especifica los esquemas para las políticas de autorización y para las solicitudes de decisión de autorización y de respuesta. También especifica cómo evaluar las políticas contra las solicitudes realizadas para calcular una respuesta. En la figura 2.3 se puede apreciar la interacción en SAML entre los distintos componentes.

2.2.1.2 Identity Provider

Un Identity Provider (IP), también conocido com Identity Assertion Provider, es responsable de:

- Proporcionar identificadores para usuarios que quieren interactuar con un sistema.
- Hacer valer a un sistema tal que dicho identificador presentado por un usuario es conocido por el proveedor.
- Proporcionar posiblemente otra información sobre el usuario que es conocida por parte del proveedor.

Esto se puede lograr a través de un módulo de autenticación que verifica un token de seguridad que puede ser aceptada como una alternativa a la autenticación en repetidas ocasiones, explícitamente de un usuario dentro de un dominio de seguridad. Un token es un objeto o elemento que representa el derecho a realizar alguna operación.

En la autenticación perimetral (el proceso de autenticación de la identidad de un usuario externo fuera del dominio de seguridad de la aplicación), un usuario necesita ser autenticado sólo una vez (Single Sign-On). El usuario obtiene un token de seguridad que luego es validado por un IP para cada sistema que el usuario necesita acceder. Algunos IP soportan varios tipos de tokens de seguridad como SAML, SPNEGO[15] y X.509[16].

En la figura 2.4 se muestra un ejemplo de integración de SAML con un Identity Provider.

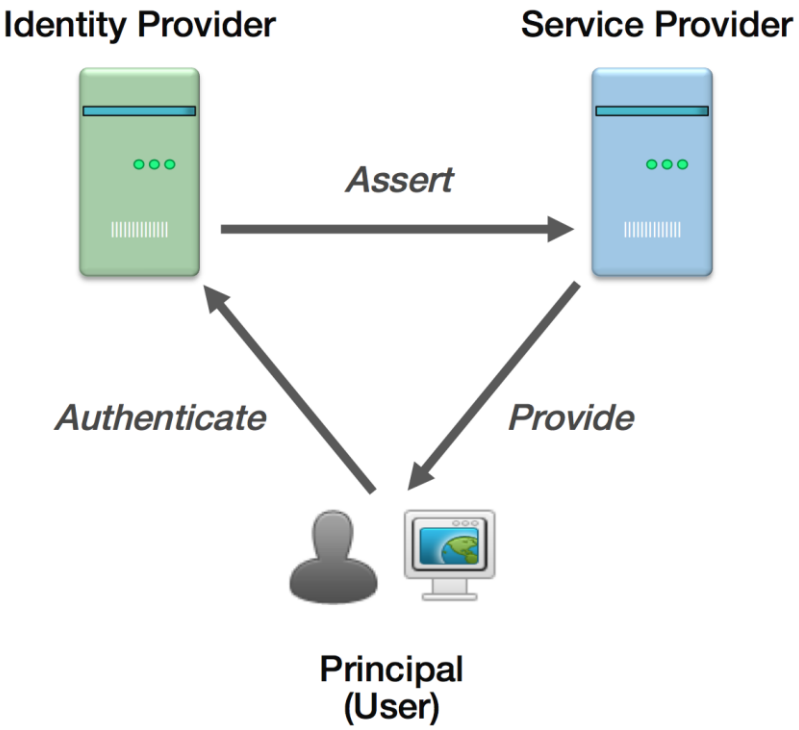

Figura 2.4 - Integración de SAML con un Identity Provider [53]

2.2.2 Tecnologías de autorización

2.2.2.1 XACML

XACML permite establecer un sistema de control de acceso que puede ser utilizado para gestionar las arquitecturas orientadas a servicios (SOA) como es en este caso. Por lo que puede ser usado para proteger información declarando derechos a través de un lenguaje de políticas de acceso.

[15] Es un pseudo mecanismo de GSSAPI (Generic Security Service Application Program Interface) utilizado por el software cliente-servidor para negociar la elección de la tecnología de seguridad.

[16] Es un estándar de criptografía para infraestructuras de claves públicas.

El estándar XACML define un lenguaje de política de control de acceso declarativo implementado en XML y un modelo de procesamiento que describe cómo evaluar las solicitudes de acceso de acuerdo a las reglas definidas en las políticas [15].

Es principalmente un sistema de Control de Acceso Basado en Atributos (ABAC), donde los atributos (bits de datos) asociados a un usuario, acción o recurso son entradas en la decisión de si un usuario puede tener acceso a un recurso determinado de una forma particular.

XACML por un lado es un modelo de autorización y lenguaje de políticas, y además es un modelo de flujo de información. En las siguientes subsecciones se describen ambas facetas.

Lenguaje de Políticas

El modelo de lenguaje de políticas define la codificación XML para expresar restricciones de acceso y puntos de extensión para definir tus propios valores de atributo, funciones, etc. Una Política XACML es el conjunto de restricciones de acceso. El siguiente diagrama UML (Unified Modeling Language) de la figura 2.5 detalla la estructura.

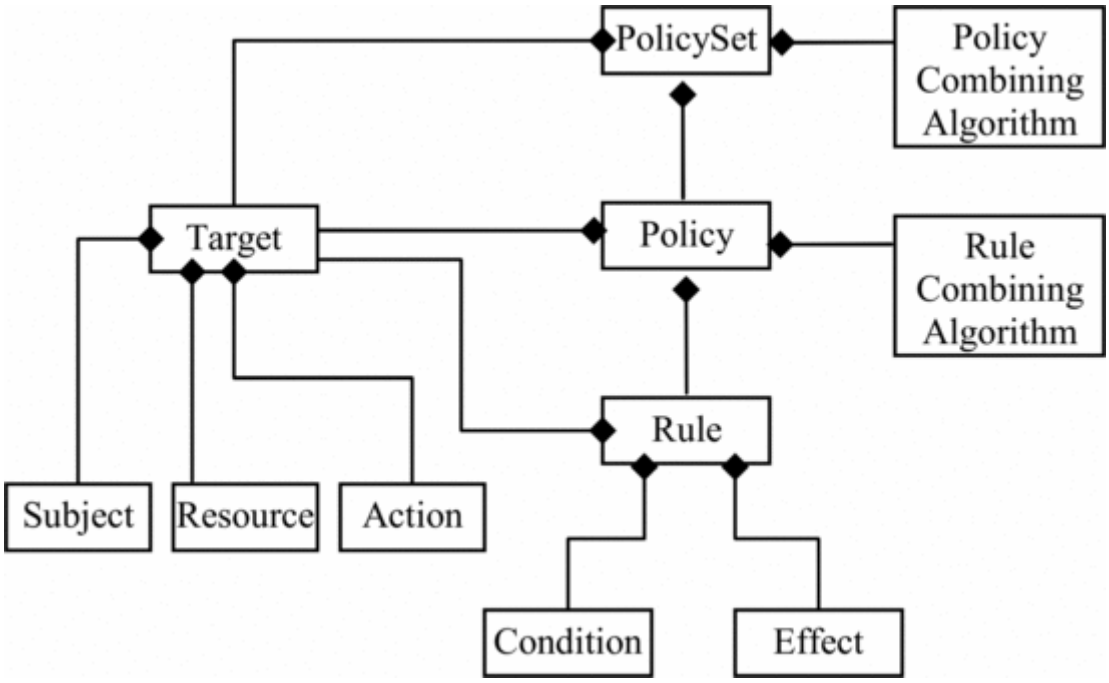

Figura 2.5 - Modelo del lenguaje de políticas [11]

- *Rule*: Una regla es la unidad más elemental de la política. Puede existir aisladamente sólo dentro de uno de los principales actores del dominio XACML. Con el fin de intercambiar reglas entre los principales actores, que deben ser encapsulados en una política. Una regla puede ser evaluada sobre la base de su contenido. Los principales componentes de una regla son: un objetivo (Target), un efecto (Effect) o una condición (Condition).

- *Target*: Cada elemento PolicySet, Policy y Rule tienen asociado un objetivo (Target) que puede ser utilizado para definir condiciones de correspondencia simples para un asunto (Subject), recurso (Resource) y acción (Action).

- *Effect*: El efecto de una regla indica la consecuencia de una regla cuando se evalúa “Verdadero”. Se permiten dos valores: "Permitir" y "Denegar".

- *Condition*: La condición representa una expresión booleana que define la aplicabilidad de la regla más allá de los predicados implicados por el objetivo (Target). Por lo tanto, puede estar ausente.

- *Policy*: Cada elemento Policy contiene el conjunto de reglas (Rule). Dentro de cada regla se pueden formar restricciones de acceso más complejas a través de las condiciones (Condition). Cada regla tiene un efecto (Effect) que determina si la regla es para permitir o denegar el acceso.

- *PolicySet*: Cada elemento PolicySet puede contener cero o más elementos de PolicySet. Esto permite que se puedan reutilizar políticas predefinidas y combinar las mismas. A su vez puede contener 1 o más elementos de Policy (políticas).

Dada una solicitud, para obtener una decisión de autorización, la políticia de XACML es recorrida desde la parte superior (el conjunto de políticas <PolicySet>) hasta las hojas (las reglas <Rule>). Para todas las reglas que se verifican, su efecto, ya sea permitir o denegar, es tomado como su controlador más básico para la decisión de autorización. Al atravesar hasta la política los efectos de todas las reglas, asociadas a un elemento <Policy> son combinadas usando el Algoritmo Combinado de Reglas (ACR). Los efectos resultantes de todos los elementos <Policy> se combinan en el siguiente nivel más alto, hasta llegar al elemento tope <PolicySet>; el Algoritmo de Combinación de Políticas (PCA) crea el efecto final sobre la política, que representa la decisión de autorización (ver figura 2.6).

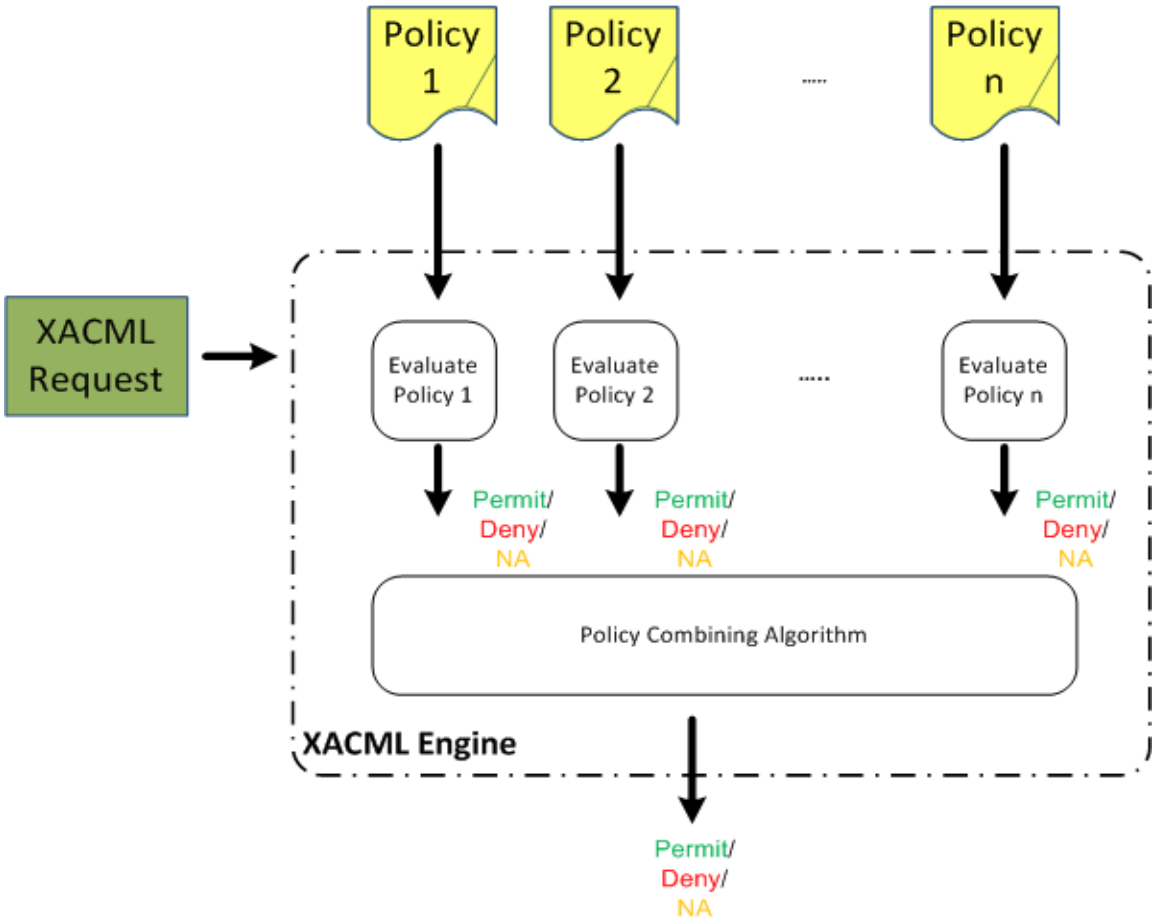

Figura 2.6 - Algoritmo de combinación de políticas [23]

Modelo de Flujo de Información

El modelo de flujo de información define la arquitectura de un sistema de control de acceso distribuido y modular. La figura 2.7 ilustra la arquitectura y la secuencia de mensajes entre los componentes del sistema.

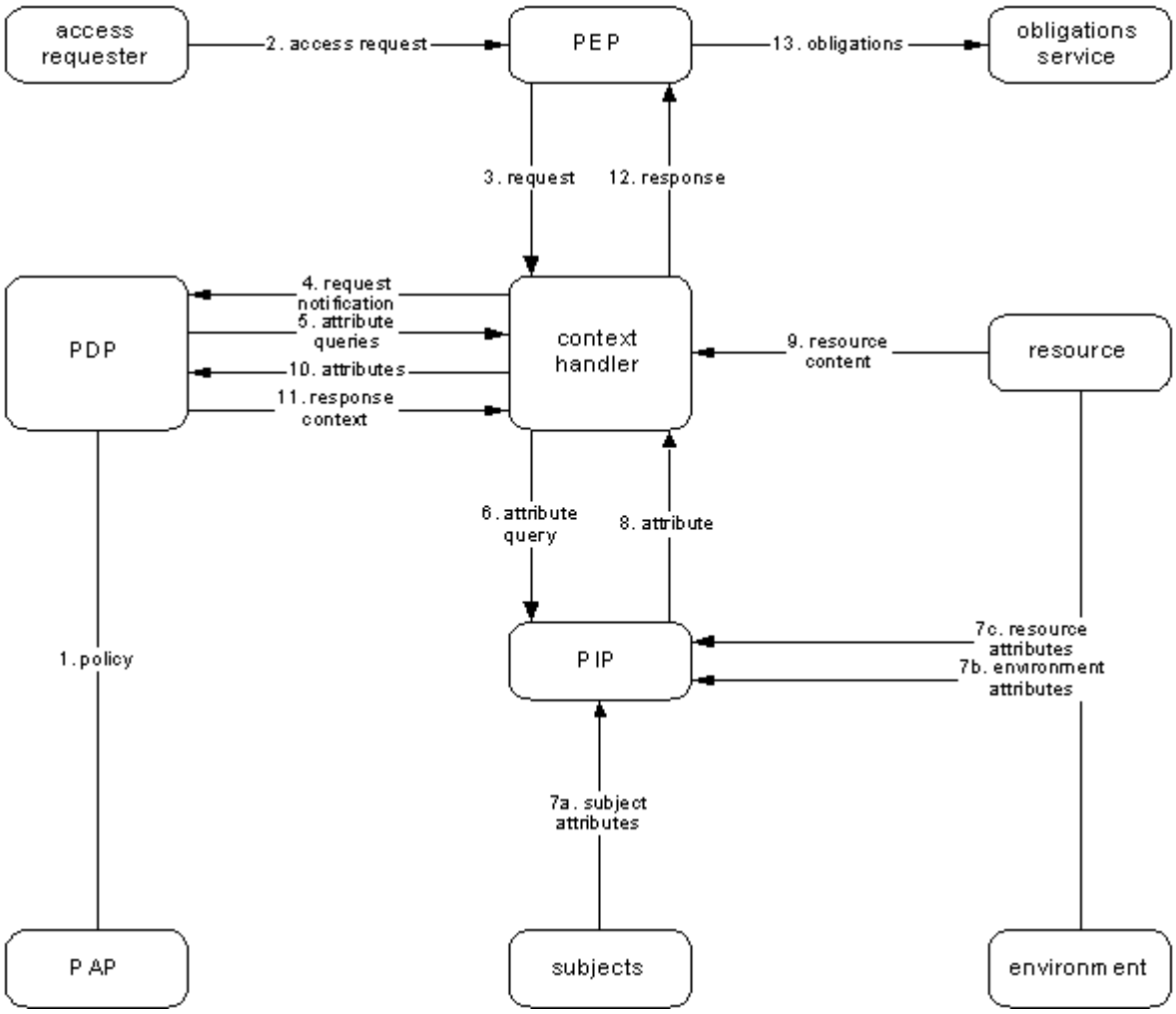

Figura 2.7 - Modelo de flujo de información de XACML [11]

Estos son los componentes que conforman la arquitectura del modelo de flujo de información:

- El Punto de Administración de la Política (PAP) es el punto que permite a uno o más administradores de política mantener los derechos de acceso en un conjunto de políticas.
- El Punto de Decisión de la Política (PDP) es el componente que obtiene una decisión de autorización, basado en una solicitud, recibido de uno o múltiples Puntos de Ejecución de la Política (PEP).
- El punto PEP intercepta la solicitud de un usuario de acceder a un recurso, hace una solicitud de decisión a PDP para obtener una decisión de acceso (para determinar si el acceso es permitido o denegado) y actúa según la decisión recibida.

- El Punto de Información de la Política (PIP) es la entidad fuente de los valores de los atributos como son los recursos (Resource), asuntos (Subjects) y entorno (Environment)

- El controlador de contexto (Context Handler) es la entidad del sistema que convierte las solicitudes de decisión en el formato nativo al formato canónico XACML y convierte las decisiones de autorización en la forma canónica XACML al formato de respuesta nativa

Cuando un usuario solicita el acceso a un recurso ocurre el siguiente flujo [15]: Los PAPs escriben políticas y conjuntos de política y las ponen a disposición del PDP. Estas políticas o conjuntos de políticas representan la política completa para un objetivo específico. El que solicita el acceso envía una solicitud de acceso al PEP. El PEP envía la solicitud de acceso al controlador de contexto en su formato nativo de solicitud, que incluye opcionalmente los atributos de los asuntos, recursos, acción, entorno y otras categorías.

El controlador de contexto construye un contexto de solicitud XACML y lo envía al PDP. El PDP solicita cualquier asunto (Subject) adicional, recurso, acción o entorno del controlador de contexto. El controlador de contexto, pide los atributos al PIP. El PIP obtiene los atributos solicitados y devuelve los atributos solicitados al controlador de contexto. Opcionalmente, el controlador de contexto incluye el recurso en el contexto. El controlador de contexto envía los atributos solicitados y (opcionalmente) el recurso al PDP.

El PDP evalúa la política y devuelve el contexto de respuesta (incluyendo la decisión de autorización) al controlador de contexto y éste traduce el contexto de respuesta al formato nativo del PEP y le envía la respuesta. Si se permite el acceso, entonces el PEP permite el acceso a los recursos; de lo contrario, se deniega el acceso.

2.2.2.2 GeoXACML

GeoXACML es una extensión geográfica de XACML para la declaración y aplicación de políticas de acceso a información geográfica. Se puede resumir que el estándar GeoXACML define [16]:

- El modelo geométrico en el que los tipos de datos geométricos de las reglas de acceso deben estar basadas.

- Los distintos lenguajes de codificación para los tipos de datos geométricos.

- Las funciones de prueba para las relaciones topológicas entre geometrías.
- Las funciones geométricas.

Al ser una extensión de XACML, GeoXACML proporciona soporte para tipos de datos espaciales y funciones de decisión de autorización espaciales. Los tipos de datos y funciones se pueden utilizar para definir restricciones espaciales adicionales para las políticas basadas en XACML.

La especificación XACML define puntos extensibles [15]. Para esta especificación se debe notar que pueden ser extendidos los tipos de datos (DataType), FunctionId y AttributeId. GeoXACML define extensiones geoespaciales para estos puntos.

A continuación, en la figura 2.8, se muestra un ejemplo de una política de acceso a un recurso geoespacial. Esta política autoriza el acceso al usuario *"usuario_zona"* a un recurso que contenga el texto *''calles''* para la operación *"GetMap"* y un espacio geográfico determinado por un conjunto de coordenadas.

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" PolicyId="2"
  RuleCombiningAlgId= "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Politica para un usuario, limitando el acceso por zona</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">usuario_zona
          </AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>                                                                          1
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">.*calles.*</AttributeValue>
          <ResourceAttributeDesignator AttributeId="layers"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
    </Resources>                                                                         2
  </Target>
  <Rule RuleId="2" Effect="Permit">
    <Target>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">GetMap</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>                                                                         3
    </Target>
    <Condition>
      <Apply FunctionId="urn:ogc:def:function:geoxacml:1.0:geometry-Intersects">
        <ResourceAttributeDesignator AttributeId="bbox" DataType="gml:geometry"/>
        <AttributeValue DataType="gml:geometry">
          <gml:geometry xmlns:gml="http://www.opengis.net/gml">
            <gml:Polygon>
              <gml:exterior>
                <gml:LinearRing>
                    <gml:coordinates cs="," decimal="." ts=" ">286681.8446823484,5892788.716326414
                  286681.8446823484,6505879.6017696615 873206.6183040687,6505879.6017696615
                  873206.6183040687,5892788.716326414 286681.8446823484,5892788.716326414
                  </gml:coordinates>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:geometry>
        </AttributeValue>
      </Apply>
    </Condition>                                                                         4
  </Rule>
</Policy>
```

Figura 2.8 - Ejemplo de política GeoXACML

Esta política tiene dos grandes secciones. En la primera se define a qué usuario o rol aplica y a los recursos incluidos en la política. La segunda establece la regla que se va a aplicar.

En la parte 1 se determina el Subject (asunto), en este caso el valor del atributo SubjectId debe ser igual a *"usuario_zona"* para que aplique la política. En el punto 2 de la imagen se declara el recurso a que está sujeto, es para el tipo de dato layers y el valor del atributo debe emparejar con el texto *"*.calles.**"*.

En la parte 2 se define la regla a esta política que es identificada con el atributo RuleId y tiene como efecto "Permit", esto quiere decir que si la solicitud cumple con las condiciones (Conditions) y cumple con el objetivo (Target) definido en la sección 1 se permite el acceso al recurso.

En la parte 3 declara las acciones que van a ser consideradas en la regla. En este ejemplo el tag ActionMatch indica que esta regla se aplica si la acción es igual al texto *"GetMap"*.

En la parte 4 se detalla la condición para que se cumpla esta regla. Está definida por una función que es una extensión de GeoXACML que determina si dos figuras geométricas se intersectan. Estas dos figuras son las definidas por las coordenadas en el elemento <gml:coordinates> y la que está en la solicitud a la que se aplica la política. Un elemento <Function> tiene un atributo llamado FunctionId, que es de tipo xs:anyURI. Se pueden definir nuevas funciones asociando un FunctionId que sea único. Esta capacidad permite la definición de funciones geoespaciales, como son definidas en GeoXACML [16].

Como parte de la función se define el atributo al que se le aplica la condición, en este caso al atributo *"bbox"* que es el *bounding box* (rectángulo delimitador) con el tipo de datos *gml:geometry*. Este es otro ejemplo de extensión que aplica GeoXACML, en el que define un nuevo tipo de dato geométrico.

El tipo de dato definido es GML (Geography Markup Language) que es un sublenguaje de XML descrito como una gramática en XML Schema para el modelado, transporte y almacenamiento de información geográfica. GML sirve como un lenguaje de modelado de sistemas geográficos, así como un formato de intercambio abierto para transacciones geográficas en Internet. Se diseñó a partir de la especificación abstracta producida por OGC, y de la serie de documentos ISO 19100. GML no contiene información específica sobre cómo se debe hacer la visualización de los datos representados. Para ello se utilizan estilos que se relacionan a GML y se describen en otros sublenguajes de XML [43].

GML permite definir un polígono que está formado por un conjunto de coordenadas. Este polígono debe intersectar con el polígono al que se le aplica la política para que se cumpla la condición.

2.3 Tecnologías de seguridad en la comunicación

2.3.1 WS-Security

La especificación WS-Security, describe la forma de asegurar los Web services en el nivel de los mensajes, en lugar de en el nivel del protocolo de transferencia o en el de la conexión [58] [59]. Para ello, tiene como objetivo principal describir la forma de firmar y de encriptar mensajes de tipo SOAP. Las soluciones en el nivel de transporte actuales, como SSL/TLS, proporcionan un sólido cifrado y autenticación de datos punto a punto, aunque presentan limitaciones cuando un servicio intermedio debe procesar o examinar un mensaje. Por ejemplo, un gran número de organizaciones implementan un firewall que realiza un filtrado en el nivel de la aplicación para examinar el tráfico antes de pasarlo a una red interna.

Si un mensaje debe pasar a través de varios puntos para llegar a su destino, cada punto intermedio debe reenviarlo a través de una nueva conexión SSL. En este modelo, el mensaje original del cliente no está protegido mediante cifrado puesto que atraviesa servidores intermedios y para cada nueva conexión SSL que se establece se realizan operaciones de cifrado adicionales que requieren una gran cantidad de programación.

WS-Security se basa en estándares y certificaciones digitales para dotar a los mensajes SOAP de los criterios de seguridad necesarios. Se definen cabeceras y usa XML Signature para el manejo de firmas en el mensaje. La encriptación de la información la realiza mediante XML Encryption haciendo uso del intercambio de credenciales de los clientes [60].

WS-Addressing es una especificación de mecanismo de transporte neutral que permite a los Web services comunicarse información de direccionamiento [61]. En esencia, consiste en dos partes: una estructura para comunicar una referencia a un extremo de servicio Web, y un conjunto de propiedades de direccionamiento de mensajes que asocian información de direccionamiento con un mensaje en particular. Desempeña un papel fundamental en la seguridad en el nivel de los mensajes puesto que proporciona los mecanismos para enviar los mensajes de un modo independiente del transporte. Esto permite enviar un mensaje

seguro a través de cualquier transporte y enrutarlo con facilidad. La protección del mensaje en lugar de la utilización del protocolo de transporte ofrece varias ventajas [62]:

- Resulta más flexible puesto que se pueden firmar o cifrar partes del mensaje en lugar del mensaje completo. De este modo, los intermediarios pueden ver las partes del mensaje destinadas a ellos. Un ejemplo de esto sería un servicio Web que enruta mensajes SOAP y puede inspeccionar las partes no cifradas de los mismos para determinar a dónde enviarlos, mientras que otras partes permanecen cifradas.
- Los intermediarios pueden agregar sus propios encabezados al mensaje y firmarlos para llevar a cabo el registro de auditorías. Por último, esto implica que el mensaje protegido se puede enviar a través de diferentes protocolos, como SMTP, FTP y TCP sin necesidad de basarse en el protocolo para la seguridad.

WS-Security define la forma de conseguir integridad, confidencialidad y autenticación en los mensajes SOAP [14]. Se realiza de la siguiente manera:

- La autenticación se ocupa de identificar al invocador.
- WS-Security utiliza tokens de seguridad para mantener esta información mediante un encabezado de seguridad del mensaje SOAP.
- La integridad del mensaje se consigue mediante firmas digitales XML, que permiten garantizar que no se han alterado partes del mensaje desde que lo firmó el originador. La confidencialidad del mensaje se basa en la especificación XML Encryption y garantiza que sólo el destinatario o los destinatarios a quién va destinado el mensaje podrán comprender las partes correspondientes.
- Se apoya en WS-Addressing para asegurar el no repudio.

2.3.2 WS-Trust

La especificación WS-Trust permite definir extensiones al estándar WS-Security con el objetivo claro de dotar a este de nuevos mecanismos de seguridad [63]. En esta especificación se incluye el proceso de solicitud, emisión y control sobre tokens de seguridad y se permite la gestión de las relaciones de confianza entre los servicios

WS-Security, realiza una definición amplia de los mecanismos básicos para proporcionar un entorno de trabajo seguro en el intercambio de mensajes. Esta especificación, partiendo de los mecanismos básicos, va añadiendo primitivas adicionales junto con extensiones para

estandarizar el intercambio de tokens de seguridad. Con ello se busca optimizar la emisión y propagación de las credenciales de los servicios dentro de diferentes dominios de confianza.

Esta especificación da soporte a un modelo de confianza destinado a los WS, se le denomina Web Services Trust Model. Para ello, se define un proceso a través del cual, un servicio, puede solicitar que cualquier petición que le llegue cumpla con una serie de reclamaciones. En la situación, que un solicitante no disponga de todos los requerimientos necesarios, los solicita al Servicio de tokens de seguridad (STS). Existe una relación de confianza entre el STS y el servicio.

WS-Trust define varios mecanismos para verificar relaciones de confianza entre dos partes. Sin embargo, no se restringe solo a ellos, pudiendo un servicio verificar la relación de confianza con la otra parte como considere necesario. Los métodos que definen son los siguientes:

- *Fixed Trust Roots*: El más simple. El servicio mantiene un conjunto fijo de relaciones de confianza.
- *Trust hierarchies*: El servicio confiara en los tokens siempre que vengan de una jerarquía de confianza que lleve a un trust root.
- *Authentication Service*: Es un servicio con el cual el servicio mantiene una relación de confianza. Cuando llega un security token, el servicio lo envía al Authentication Service el cual enviará probablemente otro token que aprobará o no la autenticación.

2.3.3 WS-Federation

Con frecuencia se produce la situación de que participantes en el consumo y la prestación de un servicio pueden utilizar diferentes tecnologías de seguridad. Por ejemplo, una de las partes podría utilizar Kerberos[17] y otro certificados X.509, y sería necesaria una traducción de los datos que afectan a la seguridad entre ambas partes.

Una federación es una colección de dominios de seguridad que han establecido relaciones en virtud del cual un proveedor de uno de los dominios puede proporcionar acceso autorizado a los recursos que gestiona sobre la base de la información de los participantes (como puede ser su identidad) la cual debe ser afirmada por un Proveedor de Identidad (IP).

[17] Es un protocolo de autenticación.

WS-Federation es la especificación que describe la forma de llevar a cabo la intermediación entre los participantes [64]. Esta especificación tiene como objetivo principal ayudar a la definición de los mecanismos de federación de dominios de seguridad, ya sean distintos o similares. Para ello, define, categoriza e intermedia con los niveles de confianza de las identidades, atributos, y autenticación de los servicios Web de todos los colaboradores.

En la especificación WS-Federation se definen perfiles asociados a las entidades que servirán para clasificar los solicitantes que pueden adaptarse al modelo.

2.4 Trabajos relacionados

Como se mencionó en el capítulo anterior el objetivo del proyecto a grandes rasgos es el de incorporar mecanismos de seguridad basados en estándares, en particular de autenticación y autorización en una plataforma de integración basada en ESB, la cual pretende abordar el desafío de integrar sistemas empresariales tradicionales con Web services geoespaciales.

En los últimos años se ha venido trabajando dentro del Laboratorio de Integración de Sistemas (LINS) en estos temas y han surgido varios proyectos que intentan abordar algunos de estos desafíos. Más puntualmente, en el artículo "*Towards an ESB-Based Enterprise Integration Platform for Geospatial Web Services*" [1] es donde inicialmente se propone la elaboración de dicha plataforma y se destacan algunos de los proyectos que se relacionan y pueden ayudar a la realización de este proyecto. A continuación se mencionan los más relevantes.

2.4.1 Plataforma de Integración de Información Geográfica sobre JBoss ESB

Este trabajo implementa mecanismos de integración en una plataforma ESB que permiten dar solución a los siguientes escenarios. Permitir el acceso a Web services geoespaciales mediante estándares tradicionales de Web services. Específicamente el acceso mediante el protocolo SOAP a Web services que utilizan el estilo REST [21]. Por otro lado provee una solución al enriquecimiento de los datos geográficos con datos empresariales.

La utilización y adaptación de estos mecanismos de integración ya implementados son parte de los requerimientos del presente trabajo.

2.4.2 Seguridad en Sistemas de Información Geográfica

Se investigó este trabajo como ejemplo de un antecedente en la utilización de una implementación de GeoXACML para controlar el acceso de Web services WMS y WFS.

El objetivo principal de este trabajo fue estudiar la usabilidad y aplicabilidad del estándar de control de acceso GeoXACML sobre los protocolos WMS y WFS. Para dicho fin se buscó desarrollar una solución basada en el estándar GeoXACML que incorporara mecanismos de control de acceso y pudiese ser integrable con soluciones empresariales existentes [20]. El resultado fue la implementación de un servidor GeoXACML, encargado de filtrar los pedidos de acuerdo a las políticas previamente generadas. Así mismo se desarrolló un administrador de políticas GeoXACML y un sistema de autenticación que permitiera identificar a los usuarios en los distintos sistemas, aunque no fueron utilizados en nuestro proyecto.

2.4.3 Orquestación de Servicios en la Plataforma de Interoperabilidad de Gobierno Electrónico

La orquestación de servicios y la integración de estos con mecanismos de seguridad en una arquitectura ESB fueron los motivos que propiciaron el interés por este trabajo. Como parte del trabajo el prototipo fue implementado en base al producto JBoss ESB por lo que fue otra razón para considerarlo.

El objetivo de este proyecto era explorar soluciones y desarrollar prototipos que permitieran guiar hacia una solución para realizar orquestaciones de servicios en la Plataforma de Interoperabilidad de la Agencia para el Desarrollo del Gobierno de Gestión Electrónica y la Sociedad de la Información y del Conocimiento (AGESIC) [37], con énfasis en la integración con el sistema de seguridad de la misma [38].

2.4.4 Mecanismos de Seguridad en Enterprises Service Bus

La implementación de mecanismos de seguridad en una plataforma ESB es uno de los objetivos principales del proyecto.

El trabajo Mecanismos de Seguridad en ESB [39] se centró en analizar las problemáticas de integración haciendo foco principalmente en la seguridad, utilizando como mediador un

ESB. Plantea una solución genérica de mecanismos de seguridad que extienda al ESB de forma de resolver la integración de seguridad de manera simple, extensible y reutilizable. Dicha solución se implementó en JBoss ESB utilizando la librería PicketLink [52]. Dicha librería es un proyecto open source de JBoss Community para la gestión de identidades en aplicaciones Java.

2.4.5 Sistema de Gestión de Identidades

La gestión de las identidades de los usuarios es uno de los puntos importantes de la autenticación. Esta funcionalidad puede ser compleja de implementar, por lo que una posibilidad es delegar esta responsabilidad a un componente externo al ESB. Se investiga el trabajo Sistema de Gestión de Identidades [40] como una opción viable. Este proyecto se planteó como objetivo proponer soluciones al problema de gestión de identidades en el contexto de una organización en la que conviven aplicaciones web donde cada una de ellas administra las identidades de sus usuarios independientemente.

2.5 Síntesis

Uno de los requisitos fijados fue la necesidad de poder acceder a los recursos geoespaciales mediante Web services convencionales y Web services RESTful. Los servidores de mapas de Internet (IMS) tienen por definición estándar de la OGC proveer la información cartográfica mediante Web services RESTful. Por lo que el consumo de los datos geoespaciales del IMS se deben realizar mediante Web services REST. Para dar soporte a las peticiones SOAP se considera el mecanismo de integración implementado en el proyecto "*Plataforma de Integración de Información Geográfica sobre JBoss ESB*" [21] (GeoEEIP a partir de ahora). Este mecanismo realiza la transformación de un mensaje SOAP en una petición WMS (REST). Esto permite exponer un Web service SOAP al cliente y que internamente se realice la transformación.

Como mencionamos anteriormente en la sección 2.4.1 la utilización de los mecanismos implementados en el proyecto GeoEEIP forman parte de los requerimientos. Por lo tanto se plantea la necesidad de incorporar los otros mecanismos implementados en GeoEEIP, que permiten agregar información empresarial a los Web services geoespaciales. Estos posibilitan el enriquecimiento de la información proveída por el IMS para una consulta WMS o WFS con datos empresariales.

El proyecto "*Seguridad en Sistemas de Información Geográfica"* [20] descrito previamente en la sección 2.4.2, tenía como uno de sus objetivos principales implementar un servidor GeoXACML que habilita el control de acceso, permitiendo o denegando el acceso a un recurso aplicando un conjunto de políticas. La implementación de esta solución no fue realizada bajo el entorno de un ESB por lo que sería necesario migrar parte de la misma a mecanismos ESB para agregarlos a la plataforma.

A partir de la investigación del proyecto "*Mecanismos de Seguridad en ESB"* (ver 2.4.4) se profundizó en el análisis del uso de la librería Picketlink. Se evalúa la utilización de este framework de seguridad de aplicaciones para Java EE[18] porque brinda funciones para la autenticación y gestión de usuarios. Además provee soluciones para la federación de identidades, autorización y seguridad para aplicaciones REST. Dentro de las tecnologías soportadas se encuentran varias de las mencionadas anteriormente en esta sección como son SAML, XACML y STS. Otra ventaja es que puede ser instalado en cualquier contenedor de aplicaciones como puede ser JBoss AS[19]. Por lo tanto para implementar el Identity Provider se plantea como opción el uso del framework Picketlink.

[18] Java EE es el estándar de la industria para desarrollar aplicaciones empresariales Java.

[19] Es un servidor de aplicaciones Java EE de código abierto implementado en Java puro, más concretamente la especificación Java EE.

3 Análisis

En este capítulo se realiza un análisis para resolver la problemática planteada. En la sección 3.1 se describen los componentes involucrados en el sistema y cómo interactúan entre sí. La arquitectura evaluada se presenta en la sección 3.2. Los componentes que son identificados en el análisis son abordados en la sección 3.3. Finalmente la composición de servicios como patrón de diseño fundamental para el diseño de los mecanismos se considera en la sección 3.4.

3.1 Componentes

El sistema que se propone tiene varios componentes, como se puede ver en el diagrama de la figura 3.1. Estos son: el ESB, el Identity Provider, el IMS, los Servicios Empresariales y el PDP.

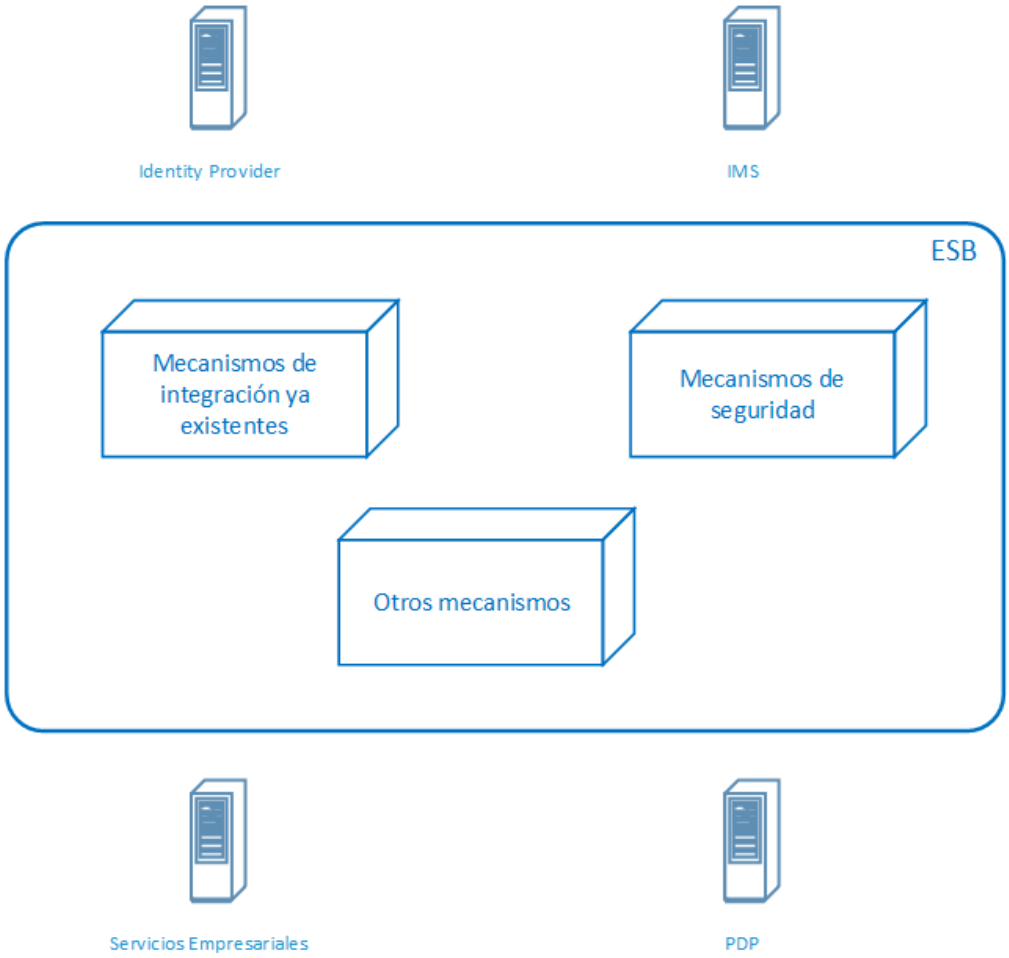

Figura 3.1 - Diagrama de componentes

Un ESB, como fue descrito en la sección 2.1.3, es un modelo de arquitectura de software utilizado para diseñar e implementar la comunicación entre aplicaciones de software en una arquitectura SOA (ver sección 2.1.2). En nuestro trabajo, este componente está integrado por mecanismos que se pueden separar en tres categorías: Mecanismos de integración ya existentes, mecanismos de seguridad y otros mecanismos. En la sección 3.1.1 se detallan cada uno de estos.

El Identity Provider ofrece como servicio la gestión de la identidad de los usuarios. El IMS es un servidor de mapas que va a ser consultado por el ESB para obtener los datos geográficos. Los Servicios Empresariales son los servicios que se consumen para enriquecer los datos geográficos provistos por el IMS con datos de negocio. En tanto que el PDP es un servidor de decisión de políticas GeoXACML. En la sección 3.1.2 se describen en mayor detalle estos componentes externos.

3.1.1 ESB

En la subsección 3.1.1.1 se detallan los mecanismos de integración ya existentes en trabajos anteriores, en la 3.1.1.2 los mecanismos de seguridad que se agregan a la plataforma y finalmente en la subsección 3.1.1.3 otros mecanismos que pueden ser agregados en el futuro siguiendo los lineamientos planteados en este trabajo.

3.1.1.1 Mecanismos de integración existentes

Los mecanismos ya existentes fueron diseñados para ser utilizados de forma independiente por lo que no interactúan entre sí. La solución planteada debe contemplar que estos mecanismos puedan ser utilizados de forma aislada pero que además puedan ser combinados entre sí.

Hay tres mecanismos ya existentes que fueron parte de la implementación de la plataforma GeoEEIP [21] y que fueron reutilizados para el trabajo actual que son: SOAP-WMS Wrapper, WMS Enricher y WFS Enricher. Estos mecanismos, que atacan problemas de enriquecimiento de mensajes e incompatibilidad de formatos, serán analizados en la sección 3.3.1.

3.1.1.2 Mecanismos de seguridad

La incorporación de mecanismos de seguridad para los Web services geoespaciales es una de las bases fundamentales de este trabajo, en particular la autenticación y autorización.

Como fue explicado en la sección 2.1.4.1, la autenticación es el proceso de verificar la identidad de un usuario. No es uno de los objetivos de este trabajo la gestión de la identidad de los usuarios de los clientes de la plataforma. En base a esto, se evalúa como una posible solución el uso de un Identity Provider (IP, ver sección 2.2.1.2) como un servicio externo a la plataforma. Para la autenticación de los clientes sería deseable que el método de autenticación sea el mismo sin tener en cuenta si la petición es del estilo SOAP o REST para que la solución sea homogénea.

Para el caso de la autorización se podría pensar en utilizar un PDP externo y un PEP dentro de la plataforma ESB (ver sección Modelo de Flujo de Información). A través del PDP podemos controlar qué datos son los que están permitidos para un cliente dado teniendo en cuenta las políticas de acceso previamente definidas. Una vez que el PEP pide acceso al PDP para un cliente en particular, el PDP decide si el mismo tiene la autorización correspondiente para poder obtener la información requerida. Luego el PEP actúa según la decisión recibida por el PDP.

3.1.1.3 Otros mecanismos

Uno de los objetivos de este trabajo es permitir la integración de mecanismos aún no implementados que cumplan con requerimientos funcionales o no funcionales y que se puedan combinar con cualquiera de los mecanismos ya existentes.

Se propone un componente de caché como un ejemplo de implementación. Este componente tiene como función almacenar los mensajes de respuesta del IMS para reducir el tiempo de respuesta de los servicios. Siguiendo la especificación del patrón de infraestructura “Mediación de caché” [25] nos enfocamos en mejorar la performance del tiempo de respuesta de una solicitud. El uso de un caché para los mensajes tiene algunos inconvenientes a considerar [25], que se detallan a continuación:

- *Correlación de los mensajes*: El ESB debe proporcionar un mecanismo para correlacionar los mensajes de solicitud con los mensajes de respuesta.

- *Identificación del mensaje*: Los mensajes ESB deben poder ser identificados de forma única para poder almacenarlos en el caché y compararlos entre sí.

- *Caché subyacente*: Este patrón no es un patrón de diseño para propósitos generales de caché. Por el contrario, aprovecha las capacidades de un caché básico existente

proporcionada por el middleware de mensajería u otros sistemas de caché. JBoss tiene como implementación estándar para JBoss Application Server el producto JBoss Cache [26].

3.1.2 Componentes externos

Entre las posibles opciones de un Identity Provider se encuentra el STS (Security Token Service), que es un proveedor de identidad basada en un software responsable de emitir tokens de seguridad, como parte de un sistema de identidad basada en notificaciones. Estos tokens de seguridad pueden ser generados por ejemplo utilizando el estándar SAML (ver sección 2.2.1.1).

El módulo PDP va a estar desplegado fuera del ESB. Este componente también fue implementado como parte del proyecto mencionado anteriormente "*Seguridad en Sistemas de Información Geográfica"* [20] y es reutilizado aquí. Será el que tendrá la responsabilidad de determinar el acceso de parte de un cliente a un determinado recurso georreferenciado.

El componente Servicios Empresariales está desplegado fuera del ESB y permite el enriquecimiento de los datos geográficos con datos empresariales disponibles a través de un Web service.

Los distintos productos de tipo IMS existentes respetan las especificaciones definidas por la OGC para Web services geoespaciales, por lo que el uso de cualquiera de ellas no debería de impactar en el diseño e implementación de los mecanismos ESB.

3.2 Propuesta de Arquitectura

El objetivo principal de este proyecto es incorporar mecanismos de seguridad basados en estándares, en particular de autenticación y autorización, partiendo de la base de la plataforma propuesta en el artículo "*Towards an ESB-Based Enterprise Integration Platform for Geospatial Web Services*" [1] y de su implementación GeoEEIP desarrollada en el proyecto "*Plataforma de Integración de Información Geográfica sobre JBoss ESB"* [21].

Para cumplir con este objetivo se diseñaron mecanismos básicos y mecanismos complejos que pudieran ser reusados. Este enfoque permite la implementación de módulos independientes como pueden ser el de autorización y el de autenticación.

Como parte de la solución de GeoEEIP se desarrollaron los mecanismos que fueron mencionados en la sección 3.1.1.1 y son explicados en mayor profundidad en la sección 3.2.1. Se plantea que estos mecanismos solo sean ajustados para que puedan ser integrados con los mecanismos de seguridad. Esto permite que además de poder ser utilizados de forma independiente puedan componerse con otros mecanismos.

La plataforma SegGeoESB es propuesta como una extensión de GeoEEIP, incorporando los mecanismos de seguridad planteados. La arquitectura definida sigue el principio de diseño Composición de Servicios [29] que tiene como beneficios la reusabilidad, modificabilidad, extensibilidad, entre otras ventajas (en la sección 3.4 se da una descripción más detallada). Este enfoque facilita la integración entre los mecanismos de integración ya existentes y los mecanismos de seguridad planteados previamente.

La figura 3.2 muestra la arquitectura propuesta con los mecanismos complejos, los básicos y los que están disponibles en JBoss ESB. Además se exponen las dependencias que tienen entre sí.

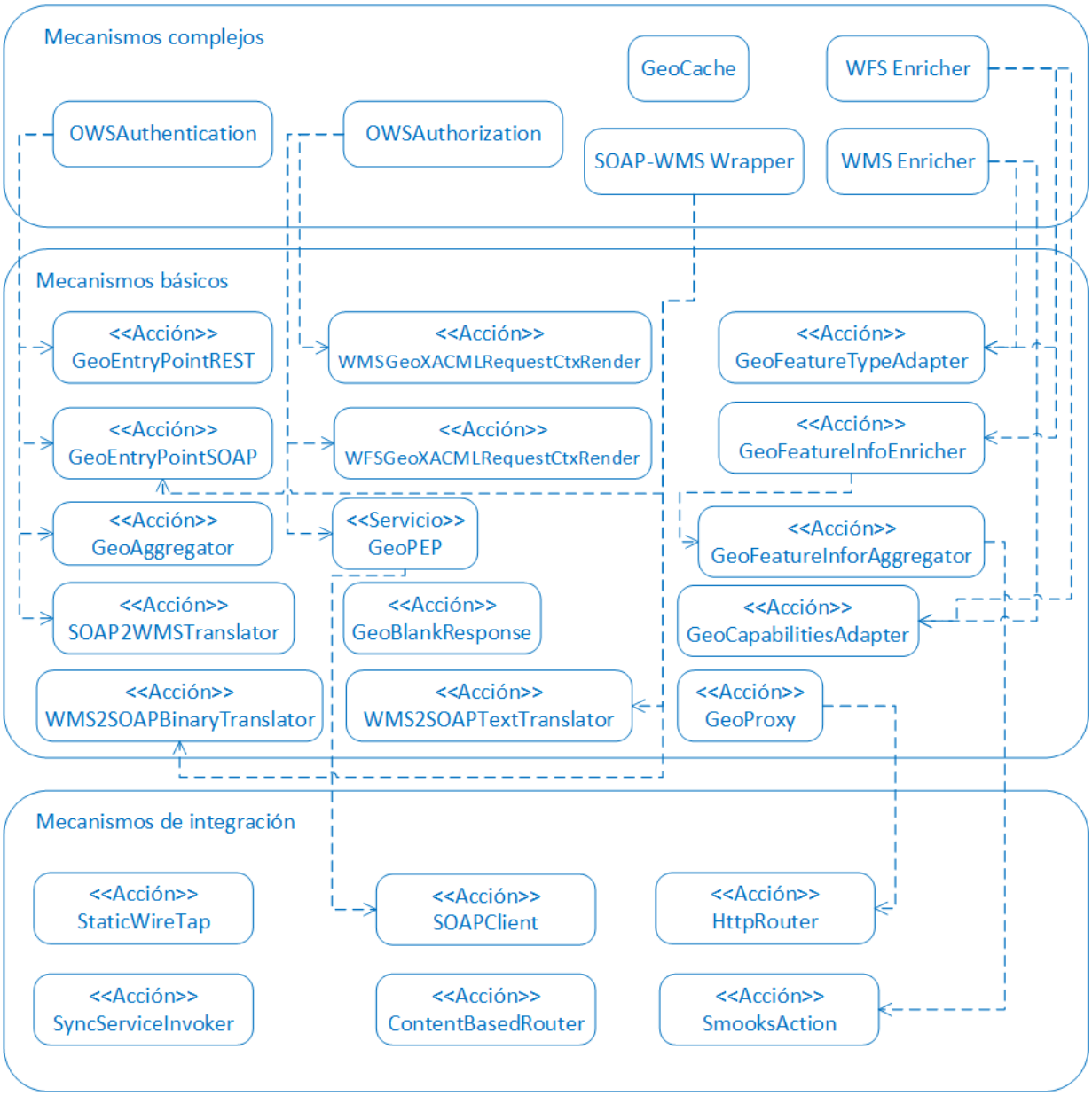

Figura 3.2 - Arquitectura de SegGeoESB

Los mecanismos complejos OWSAuthentication, OWSAuthorization y GeoCache son analizados en detalle en la sección 3.3.1. El resto de los mecanismos complejos, reutilizados del proyecto GeoEEIP, son explicados en la sección 3.4.1

3.3 Mecanismos identificados a desarrollar

A continuación se detallan los componentes ESB identificados. En la sección 3.3.1 se describen los mecanismos complejos y en la sección 3.3.2 se detallan los básicos. Estos mecanismos básicos pueden formar parte de mecanismos complejos.

3.3.1 Mecanismos complejos

A continuación se describen los mecanismos de seguridad OWSAuthentication (3.3.1.1) y OWSAuthorization (3.3.1.2) y el mecanismo de cache GeoCache (3.3.1.3).

3.3.1.1 OWSAuthentication

El mecanismo de autenticación tiene la función de verificar que las credenciales del cliente son válidas. La responsabilidad de la gestión de las identidades es delegada al Identity Provider. OWSAuthentication es presentado en la figura 3.3.

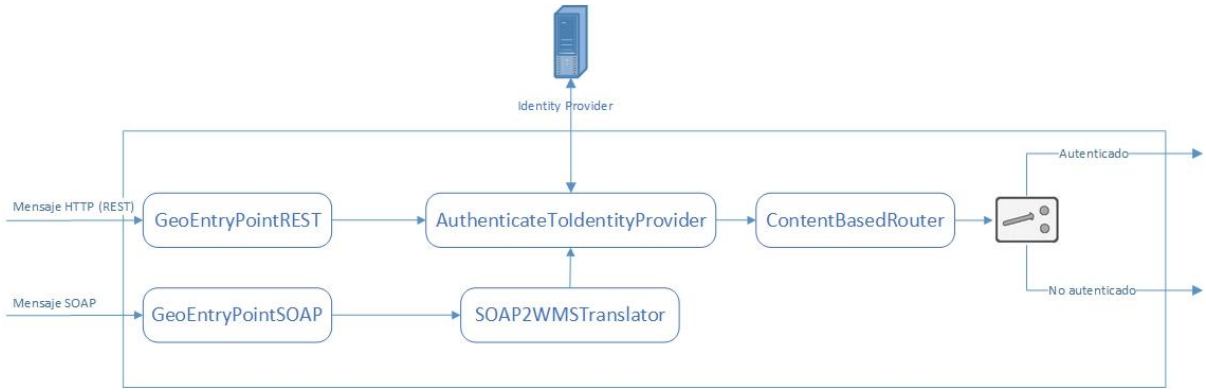

Figura 3.3 - Mecanismo de autenticación

Para las solicitudes REST las acciones realizadas son las siguientes:

1. GeoEntryPointRest recibe el mensaje HTTP y es transformado a un mensaje ESB. Es enrutado a AuthenticateToIdentityProvider.
2. AuthenticateToIdentityProvider obtiene las credenciales del cliente y se comunica con el IP.
3. El IP valida las credenciales y envía un mensaje de respuesta.
4. AuthenticateToIdentityProvider guarda dentro del mensaje ESB si fue autenticado o no, considerando la respuesta del IP.
5. El mensaje es enrutado al siguiente mecanismo en el flujo.

En caso de que la solicitud sea SOAP se realizan los siguientes pasos:

1. El mecanismo GeoEntryPointSOAP recepciona la solicitud.

2. SOAP2WMSTranslator realiza la transformación del mensaje SOAP en una petición WMS, que es guardada en el mensaje ESB. Este es mecanismo básico, que es parte del mecanismo compuesto SOAP2MWSWrapper.

3. El mensaje es enrutado a AuthenticateToIdentityProvider. Luego sigue los pasos a partir del punto 2 del flujo de las solicitudes REST.

3.3.1.2 OWSAuthorization

El mecanismo de autorización aplica el control de acceso a la información geoespacial mediante el uso del estándar GeoXACML. Utilizando como base la implementación de un servidor GeoXACML desarrollado como parte del trabajo *"Seguridad en Sistemas de Información Geográfica"* (ver 2.4.2), se diseñó el mecanismo de la figura 3.4.

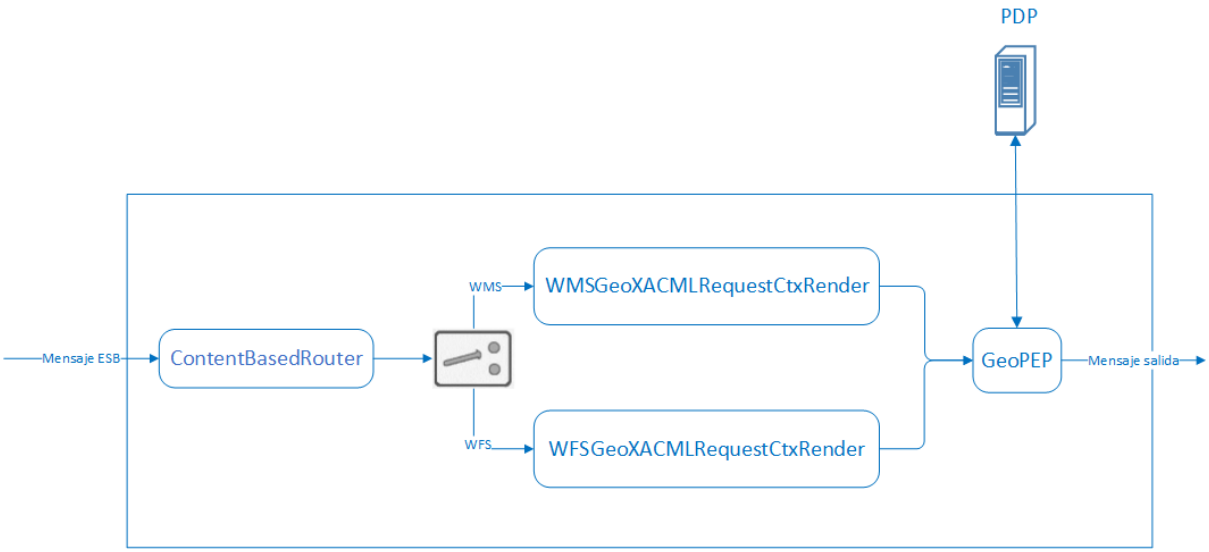

Figura 3.4 - Mecanismo de autorización

1. El mecanismo OWSAuthorization recibe un mensaje ESB y lo enruta basado en una propiedad del mensaje indicando si es una solicitud WMS o WFS.

    a. En caso de ser WMS el mensaje es enrutado hacia el componente WMSGeoXACMLRequestCtxRender.

    b. En caso de que sea WFS será enrutado hacia el componente WFSGeoXACMLRequestCtxRender.

2. Luego el mensaje es enrutado hacia el mecanismo GeoPEP, que invocará un Web service expuesto por el PDP, éste envía una respuesta si el acceso es permitido, denegado o indeterminado.

3. El GeoPEP evalúa la respuesta recibida por el PDP (3.1.4) y guarda como propiedad del mensaje si el acceso es autorizado o no.

Los mecanismos WMSGeoXACMLRequestCtxRender y WFSGeoXACMLRequestCtxRender tienen la responsabilidad de crear la solicitud que será enviada al PDP (en la sección 4.2.3 se detalla el mensaje que se envía como solicitud al PDP).

El mecanismo GeoPEP es una implementación de un Punto de Ejecución de Políticas (PEP, ver sección 2.2.2.1.2) de GeoXACML.

3.3.1.3 GeoCache

GeoCache es un mecanismo que tiene como propósito mejorar los tiempos de respuesta para obtener los recursos geoespaciales, almacenando los resultados de solicitudes anteriores obtenidos del IMS. Implementa un caché de datos geoespaciales obtenidos de un IMS siguiendo el patrón de infraestructura “Mediación de caché” [25].

GeoCache cumple con el requerimiento no funcional de que se puedan incorporar nuevos mecanismos y que puedan ser integrados con los mecanismos ya existentes como fue analizado en la sección 3.1.3. En la figura 3.5 se puede ver el diseño de dicho mecanismo.

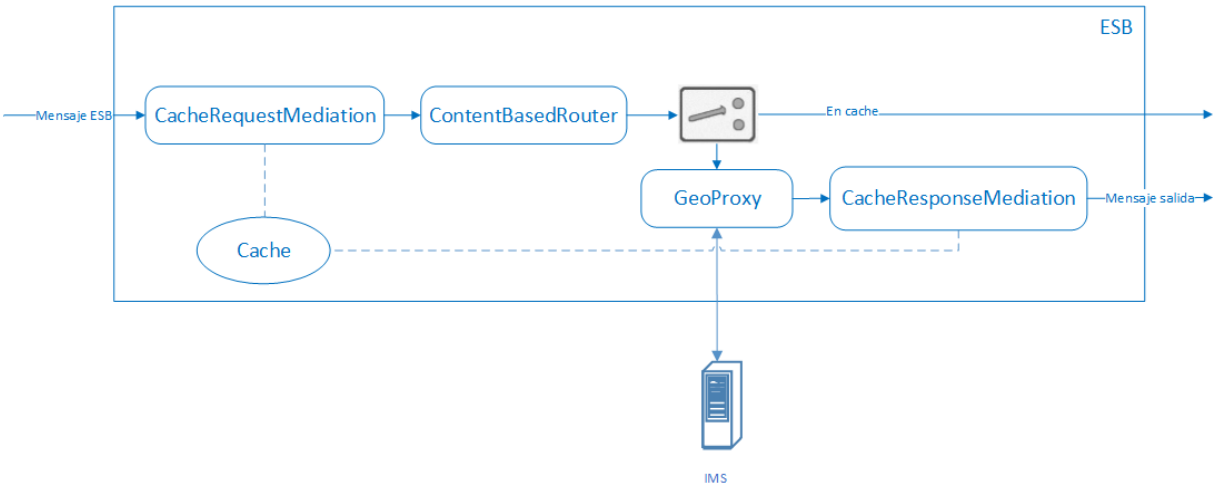

Figura 3.5 - Diagrama del mecanismo GeoCache

A continuación se describe cómo funciona el GeoCache cuando recibe un mensaje ESB:

1. El componente CacheRequestMediation consulta el cache para chequear si contiene un resultado de IMS para la solicitud WMS/WFS. Si está en el caché se obtiene la respuesta almacenada. Este resultado es guardado en el cuerpo del mensaje ESB.
2. Luego el mensaje será enrutado basado en si este contiene el resultado obtenido del caché.
    a. En caso de que contenga la respuesta se enruta el mensaje de salida hacia el servicio que invocó al GeoCache.
    b. En caso de que la respuesta no esté en el caché se enruta el mensaje al componente GeoProxy.
3. GeoProxy obtiene a través del IMS la información geográfica correspondiente a la petición que se tiene en el mensaje de entrada. La respuesta del IMS se guarda como parte del cuerpo del mensaje ESB.
4. El componente CacheResponseMediation guarda una copia de la respuesta en el cache y enruta el mensaje de salida al servicio que invocó al GeoCache.

3.3.2 Mecanismos básicos

En esta sección se describen los mecanismos básicos surgidos del análisis de los mecanismos complejos ya mencionados. Los mecanismos GeoEntryPointREST, GeoEntryPointSOAP, GeoProxy y GeoBlankResponse son reutilizados de GeoEEIP.

3.3.2.1 GeoEntryPointREST

Componente que recibe un mensaje ESB conteniendo la petición HTTP recibida por SegGeoESB. Luego se encarga de guardar en el mensaje los parámetros de la consulta como texto y un diccionario conteniendo dichos parámetros. El texto se utiliza como parte de la URL de la solicitud que se realiza al IMS. Pero además puede tener otros usos como son: clave para buscar en el cache, logs para testing o auditoría. El diccionario se utiliza para tener acceso a los parámetros de forma individual.

3.3.2.2 GeoEntryPointSOAP

Recibe un mensaje ESB conteniendo la petición SOAP recibida por la plataforma. Luego se encarga de guardar el cuerpo del mensaje SOAP en una propiedad.

3.3.2.3 GeoAggregator

Agrega una propiedad en el mensaje indicando si es de tipo WMS o WFS. Esto permite etiquetar el mensaje para que en los componentes subsiguientes se pueda enrutar el mensaje dependiendo del caso.

3.3.2.4 GeoBlankResponse

Prepara el mensaje para responder la solicitud en los casos en que no se pudo obtener la respuesta deseada, ya sea porque no se validaron las credenciales o no está autorizado a acceder al recurso.

3.3.2.5 WMSGeoXACMLRequestCtxRender

Recibe un mensaje ESB y crea un mensaje que será enviado como solicitud hacia el PDP con el WMS. Agrega esta estructura como parte del cuerpo del mensaje ESB para ser utilizada por el mecanismo GeoPEP.

3.3.2.6 WFSGeoXACMLRequestCtxRender

Tiene el mismo propósito que WMSGeoXACMLRequestCtxRender pero es invocado para las solicitudes WFS.

3.3.2.7 GeoPEP

Tiene la responsabilidad de invocar el Web service que expone el PDP y ejecuta la decisión. Los resultados posibles del PDP son Permit, Deny, Indeterminate y Not Applicable. El mecanismo guarda en una propiedad del mensaje si el acceso es autorizado o no dependiendo del resultado del PDP. Si el resultado tiene el valor Permit, la propiedad se guarda como autorizado. En el resto de los casos el acceso es denegado, por lo que la propiedad se guarda con un valor que indica que no fue autorizado.

3.4 Composición de servicios

En un escenario típico dentro de un contexto empresarial, el acceso a un recurso geoespacial implica la ejecución de diversos servicios, tales como: la autenticación, la

autorización y la transformación de SOAP a WMS (si fuera necesario). La composición de estos servicios hace posible este flujo y mejora la re-usabilidad.

Este principio de diseño desempeña un papel importante en las implementaciones SOA proporcionando los siguientes beneficios [29]:

- *Mejora de la re-usabilidad*: proporciona una manera natural de reutilización de los servicios existentes permitiendo a los proveedores de servicios agregar valor a través de la composición de los mismos.

- *Mayor rapidez de desarrollo*: nuevas soluciones se pueden construir con menor tiempo y esfuerzo.

- *Mejora de la seguridad*: El servicio compuesto representa un único punto de acceso al conjunto de servicios subyacentes. Proporciona una forma sencilla de hacer cumplir los contratos de invocación de servicios, permitiendo controlar y dosificar el acceso a los servicios de los componentes.

- *Reducción en la duplicación de código*: La redundancia se reduce o elimina. En lugar de replicar la misma funcionalidad de negocio varias veces, un único servicio de negocio que implementa la funcionalidad requerida se puede reutilizar en múltiples composiciones.

- *Mejora de la modificabilidad*: Debido a que el mismo servicio subyacente puede ser parte de muchos servicios compuestos hay un lugar único para la modificación del servicio. Los cambios en la aplicación útil de los componentes están disponibles para todos los consumidores que utilizan este servicio.

3.4.1 Mecanismos existentes

La composición de los mecanismos existentes con los mecanismos de seguridad (analizados en la sección 3.3.1) es parte de los objetivos de este trabajo. En pos de entender las funcionalidades brindadas, explicamos con más detalle los mismos.

3.4.1.1 Mecanismos complejos

Como fue mencionado en la sección 3.1.1.1, mecanismos complejos ya existentes de GeoEEIP son reutilizados para el trabajo actual. Estos son: SOAP-WMS Wrapper, WMS Enricher y WFS Enricher. A continuación se realiza una descripción de cada uno de ellos.

SOAP-WMS Wrapper

Permite publicar operaciones de WMS siguiendo los estándares de la W3C para Web Services (SOAP y WSDL). Esto permite integrar WMS con sistemas que tienen esos estándares como restricción. Para brindar esta funcionalidad, primero el mecanismo GeoEntryPointSOAP recibe la petición SOAP realizada a un Web Service publicado. Luego el mecanismo básico SOAP2WMSTranslator realiza la conversión del mensaje SOAP en una petición WMS. A continuación el mecanismo GeoProxy obtiene a través del IMS la información geográfica correspondiente a la petición que se tiene en el mensaje de entrada. El mensaje es enrutado al mecanismo básico WMS2SOAPBinaryTranslator si la operación es GetMap y en caso contrario se direcciona al mecanismo básico WMS2SOAPTextTranslator. Ambos generan como resultado un mensaje SOAP que contiene la respuesta WMS.

WMS Enricher y WFS Enricher

Estos mecanismos permiten el enriquecimiento de la información geográfica con datos empresariales. La información geográfica se obtiene del IMS y los datos empresariales del servidor de Servicios Empresariales. En el caso de WMS, la operación GetFeatureInfo permite consultar características particulares mostradas en el mapa y GeoFeatureInfoEnricher (que es parte del WMS Enricher) permite enriquecer los datos con información empresarial. Para WFS la operación GetFeature posibilita la consulta de datos y el mecanismo GeoFeatureEnricher permite enriquecer los mismos.

3.4.1.2 Mecanismos básicos

Los mecanismos básicos existentes de GeoEEIP son los listados y definidos a continuación.

GeoEntryPointSOAP

Componente que recibe un mensaje ESB conteniendo la petición SOAP recibida por la plataforma. Luego se encarga de guardar en el mensaje ESB el cuerpo del mensaje SOAP.

SOAP2WMSTranslator

Este componente se encarga de transformar la petición SOAP que se encuentra en el mensaje ESB a una petición WMS, para luego guardarla en el mensaje.

WMS2SOAPBinaryTranslator

Mecanismo que procesa una respuesta WMS, generando como resultado un mensaje SOAP conteniendo la respuesta WMS. En particular es aplicado a respuestas que contienen datos binarios.

WMS2SOAPTextTranslator

Cumple la misma función que WMS2SOAPBinaryTranslator pero se aplica a respuestas que contienen únicamente texto.

GeoFeatureInfoEnricher

Es el componente encargado de obtener los datos empresariales y luego enriquecer con éstos la información del IMS almacenada previamente en el mensaje ESB (mediante transformaciones XSLT).

GeoFeatureEnricher

Mecanismo encargado de enriquecer el resultado de la operación de WFS GetFeature con la información de negocio (resultado de la consulta a Web services empresariales).

3.4.2 Flujo de mediación para soportar la composición de servicios

En esta subsección se describe el flujo de mediación más completo, en el que intervienen todos los mecanismos complejos analizados.

Las peticiones REST van a ser recepcionadas por el servicio GeoEntryPointREST, para ser enrutadas hacia el componente OWSAuthentication que brinda la funcionalidad de autenticación del cliente comunicándose con el Identity Provider. Si el cliente es autenticado se enruta el mensaje ESB hacia el módulo de autorización OWSAuthorization. Este mecanismo contiene la implementación de un PEP (ver 2.2.2.1) que evaluará la respuesta del PDP. En caso de que esté autorizado el acceso al recurso, se enruta el mensaje hacia el componente GeoCache. Si la solicitud está en el caché, se enruta el mensaje hacia el mecanismo WMS Enricher o WFS Enricher (dependiendo si era un Web service WMS o

WFS). En caso de no estar en el caché, se redirige el mensaje hacia el componente GeoProxy que obtendrá el recurso desde el IMS.

Para la consulta empresarial el mensaje se enruta dependiendo del tipo (WMS/WFS), en caso de que sea WMS se enruta hacia el mecanismo WMS Enricher, de lo contrario se enruta hacia el WFS Enricher. Los componentes de enriquecimiento empresarial realizan la consulta a un Web service expuesto por parte del servidor Servicios Empresariales, y agregan los datos empresariales al mensaje. A continuación el componente es enrutado al mecanismo RestoreRESTMessage que se encarga de cargar el cuerpo del mensaje ESB con el mensaje REST que va a ser respondido al cliente.

Para las peticiones SOAP solo se van a considerar las que sean del estilo WMS (WFS no está soportado). Estas solicitudes van a ser recibidas por el servicio GeoEntryPointSOAP y transformadas a WMS por el componente SOAP2WMSTranslator. Luego de transformado el mensaje, se enruta al mecanismo de autenticación OWSAuthentication y luego sigue el mismo flujo que una petición REST hasta que se obtienen los datos empresariales. En este punto se realiza un ruteo basado en el tipo de contenido, si es una solicitud por un recurso binario (para la operación GetMap) se enruta el mensaje al mecanismo WMS2SOAPBinaryTranslator y si es de tipo texto (para la operación GetCapabilities) se enruta al mecanismo WMS2SOAPTextTranslator.

El flujo para las solicitudes REST y SOAP descritas anteriormente se pueden ver en la figura 3.6.

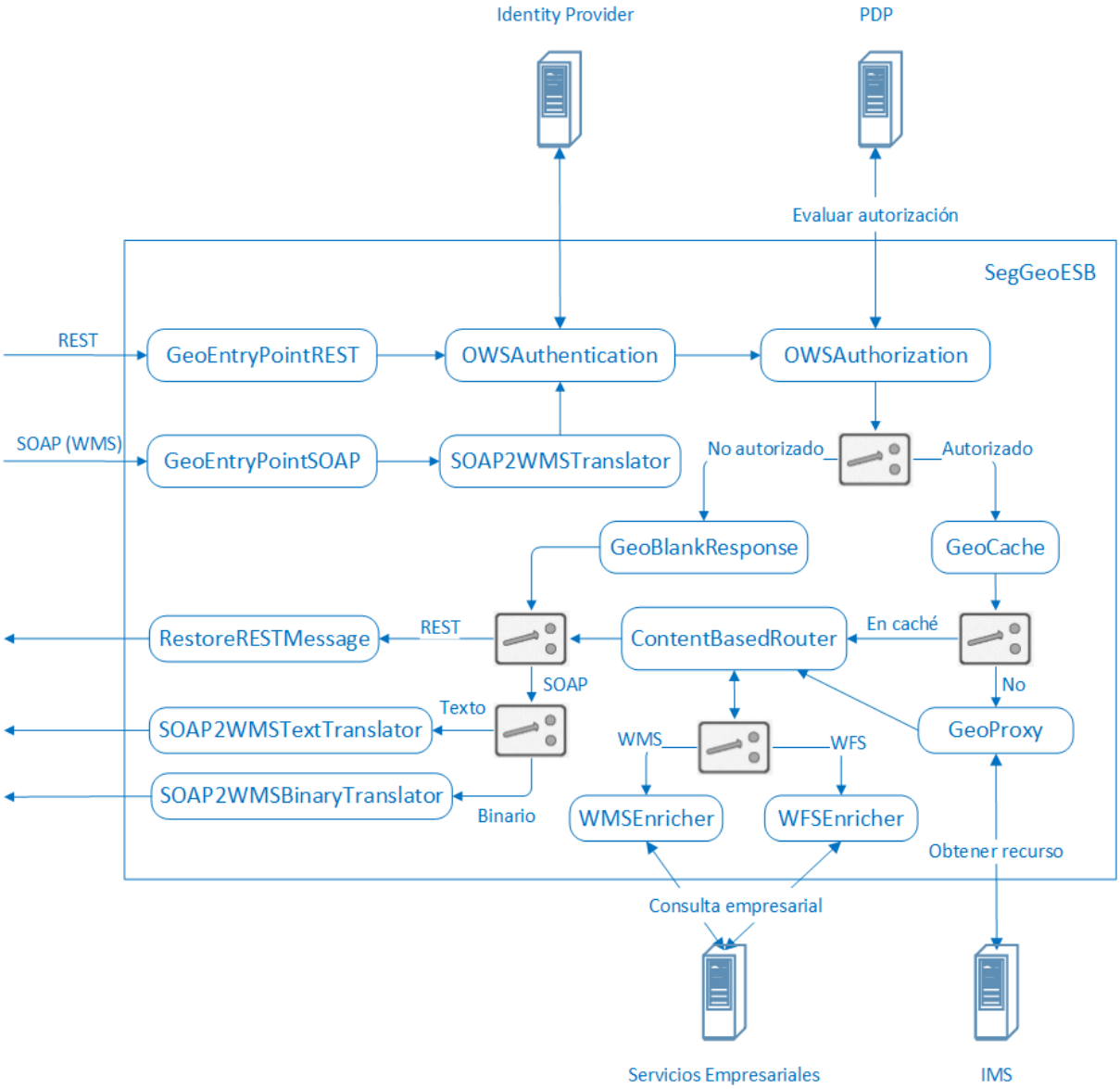

Figura 3.6 - Diagrama del flujo de las solicitudes REST y SOAP

4 Propuesta de solución

La solución planteada tiene como propósito agregar a la plataforma los mecanismos de seguridad que proveen autenticación y autorización, así como brindar la capacidad de combinarlos con otros mecanismos, ya sea reutilizando los que ya existían como desarrollando otros para satisfacer nuevos requerimientos. En la figura 4.1 se muestran los mecanismos y las dependencias que tienen entre sí.

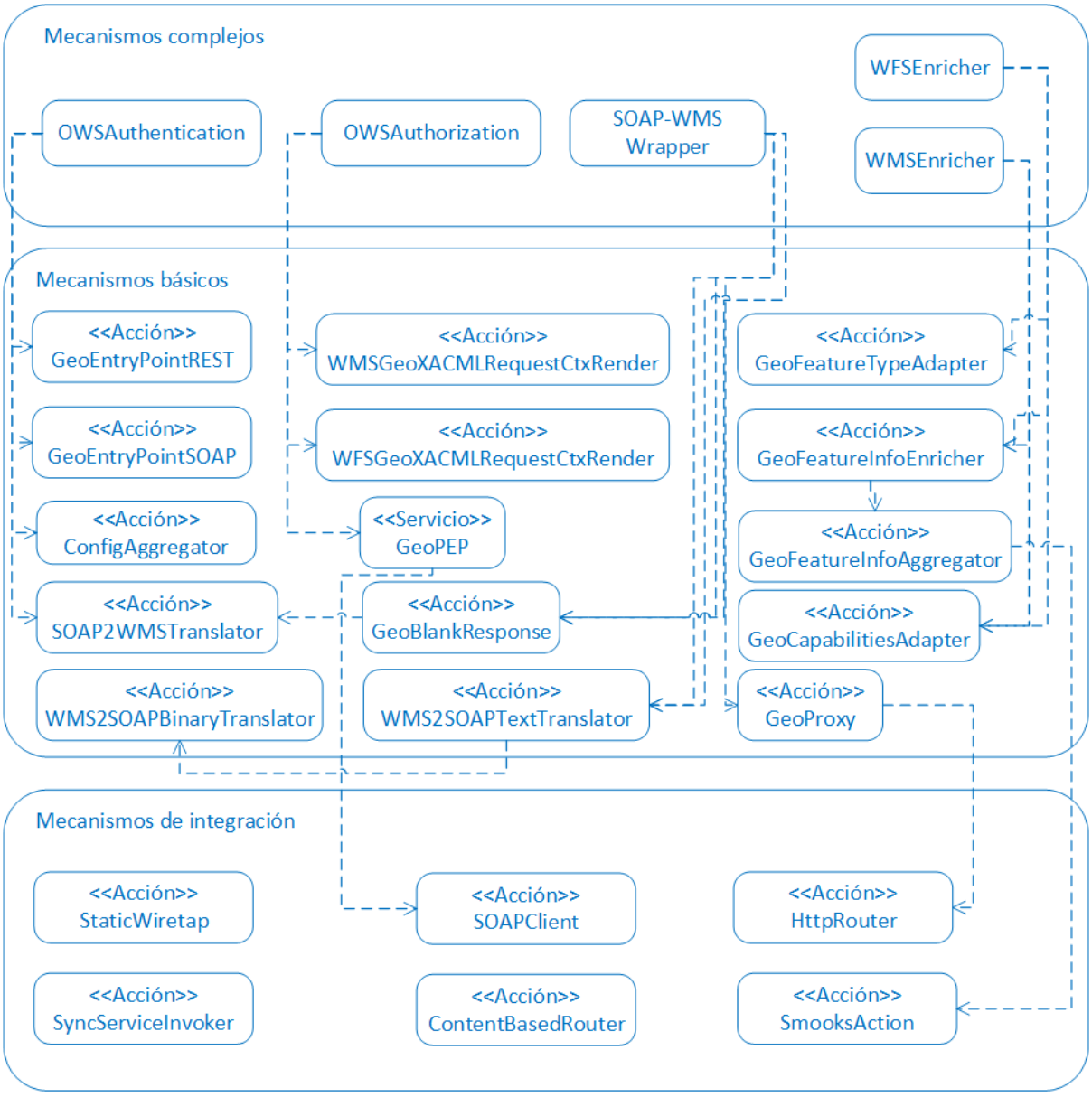

Figura 4.1 - Arquitectura de la solución

OWSAuthentication es el mecanismo de autenticación implementado. Un STS externo al ESB será la autoridad responsable de la gestión de la identidad de los clientes. En la sección 4.1.1 se realiza una explicación más en detalle de este mecanismo.

El mecanismo de autorización *OWSAuthorization* realiza el control de acceso utilizando una implementación del estándar GeoXACML que está contenida en el componente externo PDP. Este componente publica un Web service que es consumido desde la aplicación SegGeoESB cuando se quiere solicitar el acceso a un dato geoespacial de parte de un cliente. En la sección 4.1.2 se especifica en mayor profundidad la composición de servicios de este mecanismo y cómo es la comunicación con el PDP.

Para los casos de los mecanismos de integración ya existentes como son SOAP-WMS Wrapper, WMS Enricher y WFS Enricher se pueden seguir invocando como servicios aislados pero además se extendió la usabilidad para que puedan incorporar los mecanismos de seguridad. Esto permite, por ejemplo, que se puedan enriquecer los datos geográficos de un recurso solicitado a través desde un WMS y que además se ofrezca autenticación y control de acceso.

Los mecanismos de integración que están disponibles en JBoss ESB y que son utilizados por parte de los mecanismos implementados son descritos en la sección 4.4.2.

4.1 Mecanismos complejos

A continuación se detallan todos los mecanismos complejos que fueron implementados describiendo su funcionalidad y la composición de mecanismos que los componen.

En la sección 4.1.1 se describe el componente de autenticación y en la sección 4.1.2 se realiza una exposición del módulo de autorización.

4.1.1 OWSAuthentication

El componente de autenticación cumple como primera función la de ser el punto de entrada de los Web services REST y SOAP expuestos por la aplicación SegGeoESB que requieren autenticación. La responsabilidad de la autenticación es delegada a un STS que está desplegado como un componente externo. En la figura 4.2 se muestra un diagrama con la composición de los servicios que están incluidos en el mecanismo.

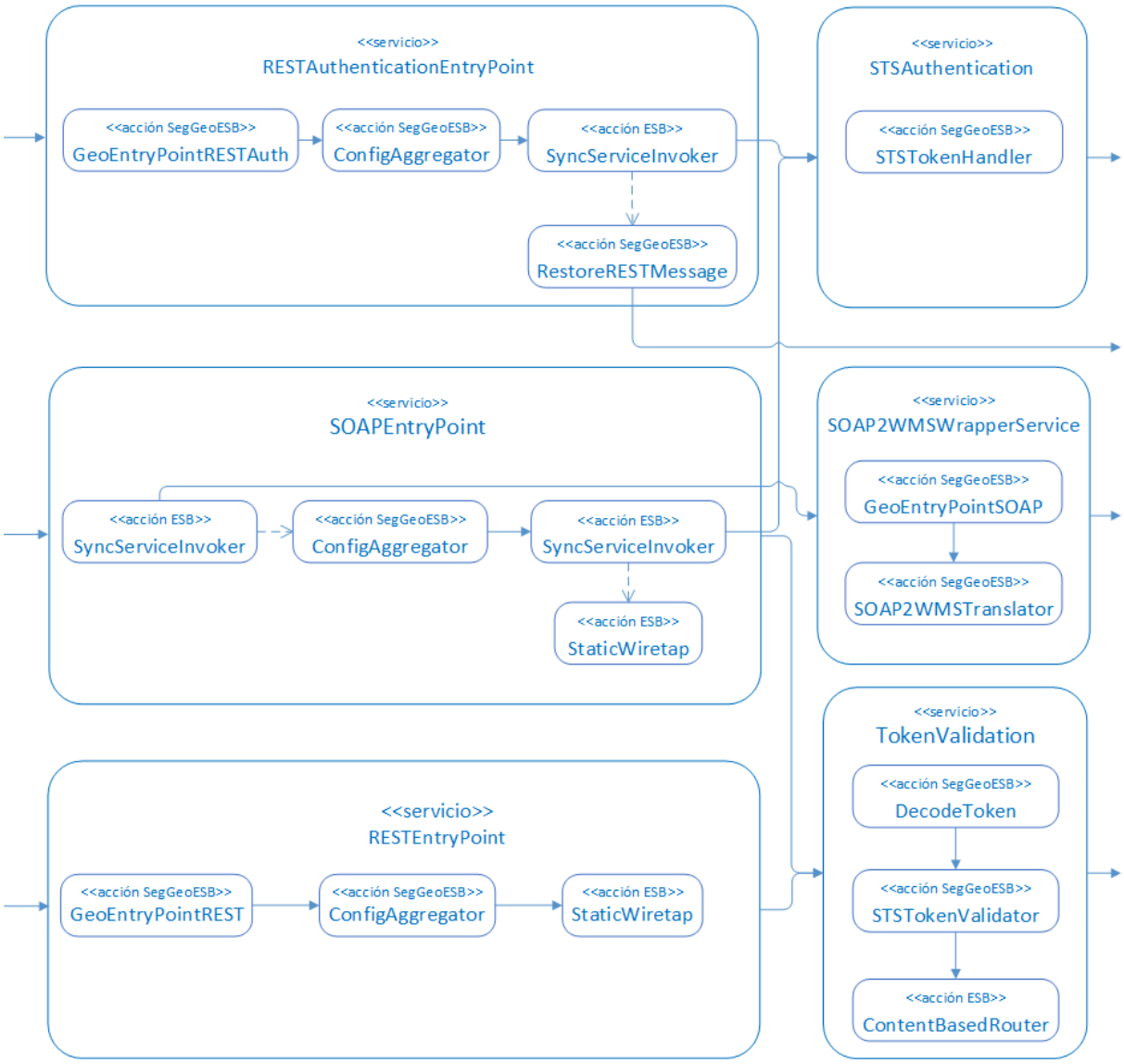

Figura 4.2 - Diagrama de flujo de *OWSAuthentication*

En el caso de una solicitud de autenticación REST el flujo es el siguiente:

1. El servicio *RESTAuhtenticationEntryPoint* expone un Web service Restful para el intercambio de las credenciales del cliente y envía como respuesta un token de seguridad SAML. Esta comunicación se realiza utilizando el protocolo HTTP sobre SSL para brindar seguridad.

2. El mecanismo *GeoEntryPointRestAuth* obtiene las credenciales enviadas por el cliente y las guarda en las propiedades USER_NAME y USER_PASSWORD del mensaje.

3. Luego el mecanismo *ConfigAggregator* agrega las propiedades de configuración de SegGeoESB.

4. La acción *SynServiceInvoker* enruta el mensaje de forma sincrónica al servicio *STSAuthentication*.

5. La acción *STSTokenHandler* le envía una solicitud al servlet (que está configurado en la propiedad del mensaje STS_TOKEN_GENERATOR_URI) con las credenciales del cliente esperando como respuesta el token generado. El token es codificado en Base64[20] y se guarda como cuerpo del mensaje ESB.

6. El mensaje retorna al servicio *RESTAuhtenticationEntryPoint* en el que la acción *RestoreRESTMessage* prepara el mensaje para que el cliente REST pueda interpretarlo.

Para acceder a un recurso utilizando una solicitud REST se aplican los siguientes pasos:

1. El servicio *RESTEntryPoint* expone un Web service Restful para proveer el acceso a los datos geoespaciales a los clientes REST.

2. La acción *GeoEntryPointREST* obtiene los parámetros de la solicitud WMS o WFS y el token SAML y los guarda dentro del mensaje ESB.

3. A continuación la acción *ConfigAggregator* carga propiedades de configuración de SegGeoESB. Estas propiedades son básicamente habilitar la autorización y/o el enriquecimiento de la información geoespacial con datos empresariales y las URLs para acceder al STS. Esto permite configurar de forma dinámica el flujo de los mensajes. En la sección 4.2.2 se describe en más detalle este mecanismo.

4. El mensaje, mediante la acción *StaticWiretap* es enrutado al servicio *TokenValidation*. El token es decodificado de Base64 en la acción *DecodeToken*.

5. La acción *STSTokenValidation* es la responsable de validar el token realizando una petición al STS y guardar la respuesta en la propiedad del mensaje VALIDATED_TOKEN.

6. La acción *ContentBasedRouter* rutea el mensaje basado en el valor de dicha propiedad. Si el valor es AUTORIZADO se enruta al servicio de entrada del componente de autorización *AuthorizationEntry*, si el valor es NO_AUTORIZADO se

[20] Base64 es una codificación que tiene la capacidad de convertir cualquier dato definido a nivel de bytes en un formato seguro de transportar por Internet

enruta el mensaje al servicio *BlankResponse* y si es SALTEAR_AUTORIZACION se enruta al servicio *GeoServerResponse* (ver sección 4.3).

Para las solicitudes SOAP el flujo del mensaje es:

1. El servicio *SOAPEntryPoint* es el punto de entrada para las solicitudes SOAP.
2. El mensaje es enrutado de forma sincrónica utilizando la acción del ESB *SyncServiceInvoker* al servicio *SOAP2WMSService*.
3. La acción *GeoEntryPointSOAP* obtiene las credenciales del cliente y el cuerpo del mensaje SOAP y las guarda como propiedades del mensaje ESB.
4. Luego la acción *WMS2SOAPTranslator* transformará el cuerpo del mensaje SOAP en una solicitud WMS y el mensaje retorna al servicio *SOAPEntryPoint*.
5. A continuación la acción *SyneServiceInvoker* enruta el mensaje al servicio *STSAuthentication*. Una vez recibido el token SAML el mensaje será enrutado al servicio *TokenValidation*.

El intercambio de credenciales entre el cliente y el ESB requiere de una comunicación bajo alguna herramienta de seguridad. Se tomaron distintos enfoques para SOAP y REST. En SOAP, WS-Security es el protocolo de facto en seguridad por lo que fue una decisión natural utilizar este estándar para las solicitudes SOAP. REST no es compatible con WS-Security pero sí soporta SSL [36], por lo que se opta por cifrar la comunicación utilizando este protocolo.

Por esta razón, en el momento de implementar la autenticación del cliente se resolvió implementar dos enfoques según el protocolo de comunicación a utilizar.

Para el caso REST en primer lugar se obtiene el token de seguridad para luego, utilizando el mismo, poder hacer peticiones al sistema. A continuación se muestran los diagramas de secuencia del sistema para el caso de uso login y el pedido de un recurso para las peticiones REST.

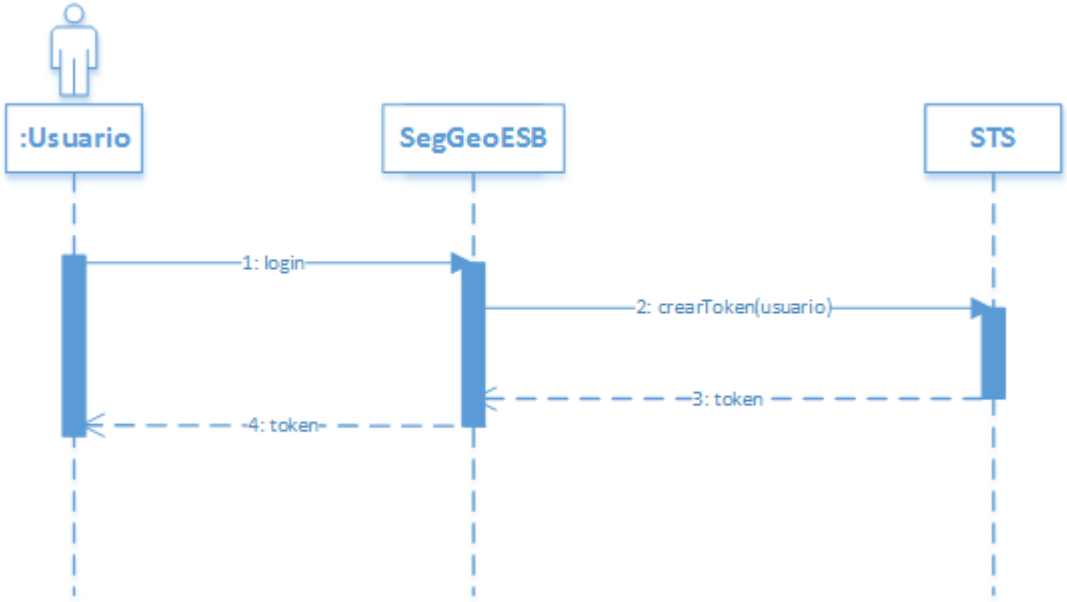

Figura 4.3 - Diagrama de secuencia de inicio de sesión REST

En la figura 4.3 se describe el flujo para la obtención de un token para un usuario. El usuario envía una solicitud de login con las credenciales a SegGeoESB. Este se comunica con el STS para generar el token para el usuario. El STS genera el token y se lo envía como respuesta a SegGeoESB que a su vez le enviará el token creado al usuario que lo solicitó.

Luego de obtenido el token de acceso el usuario podrá obtener los datos geoespaciales a través de sucesivas invocaciones al Web service REST expuesto por SegGeoESB. Cada vez que el usuario realice una consulta la aplicación SegGeoESB se comunicará con el STS para que valide el token. Luego invocará el servicio responsable de obtener el recurso solicitado en caso de que el token sea validado y que el control de acceso lo autorice. Por último la respuesta será enviada al usuario.

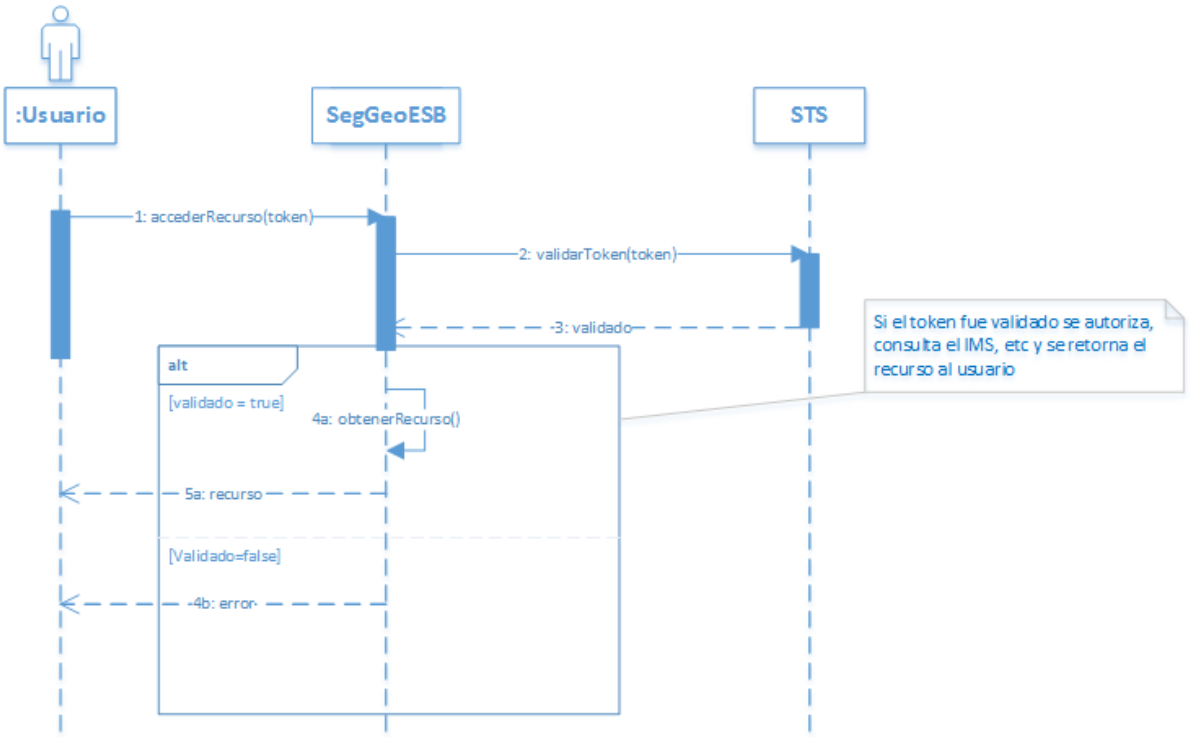

Figura 4.4 - Diagrama de secuencia al solicitar un recurso con el estilo REST

Para el caso de SOAP cuando el usuario requiere acceso a un recurso se envían las credenciales del usuario junto con la petición del recurso. SegGeoESB solicita la creación de un token de seguridad al STS para luego poder acceder a los recursos del sistema. En el diagrama 4.5 se puede observar el flujo descrito.

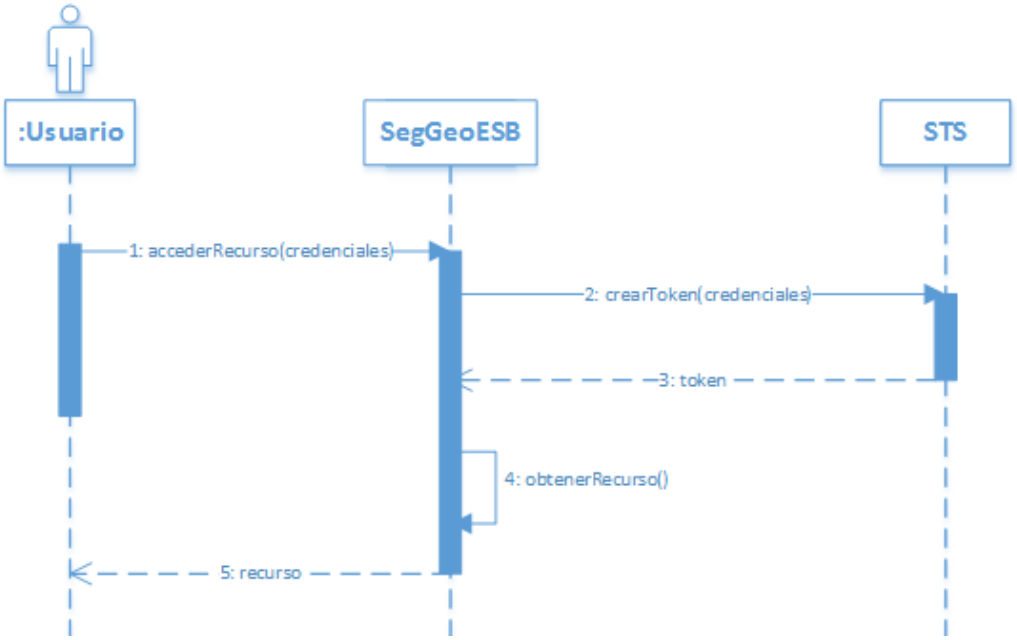

Figura 4.5 - Diagrama de secuencia al solicitar un recurso con el estilo SOAP

4.1.1.1 Implementación del STS

Como se mencionó en la sección 2.5 se había propuesto el uso de la librería Picketlink [52] para implementar el Identity Provider (IP), a ser utilizado como el responsable de la gestión de identidades de los usuarios. Como parte del análisis realizado para la propuesta de la arquitectura de la plataforma se había evaluado como implementación del IP un STS (ver sección 3.1.2).

Se descartó el uso de la librería Picketlink debido a que las versiones de dicha librería que a la fecha tienen soporte no son compatibles con la versión de JBoss AS utilizada. Se tomó la decisión de usar un STS que está basado en la implementación del proyecto de grado "*Integración GIS-PGE"* [19]. Este componente es el encargado de generar el token SAML, que luego debe ser enviado en cada solicitud por parte del cliente. Además de crear el token tiene la responsabilidad de controlar si un token dado es válido. Tanto la generación como la validación de los tokens son expuestos a través de Servlets[21].

El STS consta de un componente que es el *STSServer*. Este componente expone los servicios de generación y validación de los tokens y la librería *STSClient*, la cual luego es utilizada por la aplicación SegGeoESB para consumir esos servicios.

Se debieron realizar algunas modificaciones para adaptarlo a los requerimientos del sistema, que se listan a continuación:

- Se agregó el usuario asociado al token como un elemento del token. Esto permite validar el usuario. Además es un requisito del PDP que se le proporcione el sujeto (en este caso el usuario) al que se le va a aplicar la evaluación de las políticas de seguridad para acceder a un recurso

- El STS no tenía implementado el manejo de validación de los tokens, por lo que se decidió extender la aplicación para que soporte esta función. Esto se hizo siguiendo el standard de OpenSAML [22]. Se agregó las siguientes validaciones:
    - Validación de la clave enviada en el token. Controla que el token haya sido firmado por la misma entidad que lo emitió.
    - Validación del tiempo de expiración del token. Si el token ha superado el tiempo de expiración, el servidor responde que el token no es válido. El tiempo de expiración es configurable y permite que luego de cierto tiempo el

[21] Los Servlets son módulos escritos en Java que se utilizan en un servidor, que puede ser o no ser servidor web, para extender sus capacidades de respuesta a los clientes al utilizar las potencialidades de Java.

token no sea válido creándolo nuevamente. La regeneración de este token corre por cuenta del cliente.

4.1.2 OWSAuthorization

El componente de autorización recibe como entrada un mensaje ESB y en el servicio *AuthorizationEntry* realiza un ruteo basado en si la propiedad del mensaje GEO_REQ_TYPE es igual a WMS o WFS. En caso de que sea WMS es enrutado al servicio *GetWMSTargets*, que será el responsable de crear la solicitud que va a ser enviada al PDP. En caso de que sea WFS el servicio *GetWFSTargets* realiza la misma acción pero es específica para WFS. En la sección 4.2.3 se describe en mayor profundidad cómo funcionan estos componentes.

El mecanismo *GeoPEP* (ver sección 4.2.4) es el responsable de comunicarse con el PDP y evaluar la decisión que fue enviada por parte del componente. Esta comunicación es a través de un Web service que expone el PDP y se configura en la acción del ESB *SOAPClient*.

Luego de obtenida la decisión de acceso al recurso, el mensaje es enrutado al servicio *ResponseEntry* que es el punto de entrada para la composición de servicios, responsable de obtener el recurso del IMS y enviar la respuesta al cliente (ver sección 4.3).

En la siguiente figura se detalla el componente desde el punto de vista de la composición de servicios y cómo están relacionados entre sí los mecanismos.

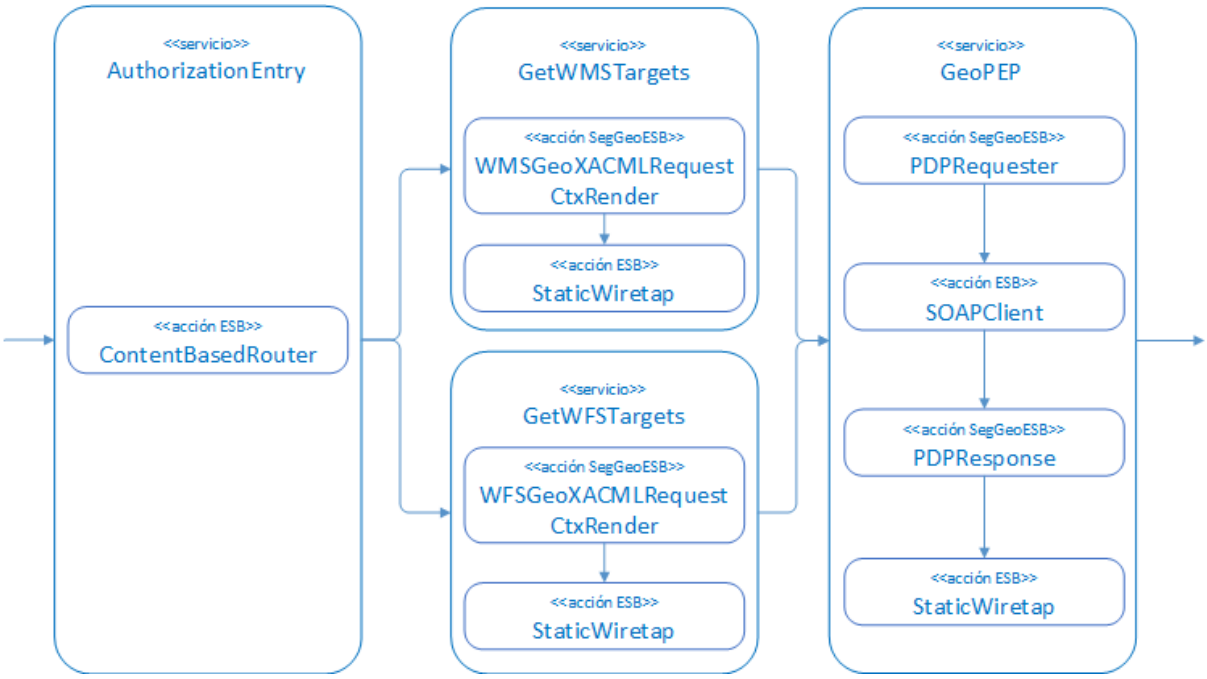

Figura 4.6 - Diagrama de flujo del componente *OWSAuthorization*

4.1.2.1 Implementación del PDP

El PDP y el PEP componían el servidor GeoXACML implementado por el proyecto descrito en la sección 2.4.2. El módulo PEP fue adaptado e incorporado al proyecto SegGeoESB. En tanto que el PDP quedó como el único módulo del servidor GeoXACML.

Este PDP fue desarrollado sobre una implementación libre de XACML publicada por Sun Microsystems [44]. Cuenta con un repositorio de políticas que está implementado como un conjunto de archivos (uno por cada política) disponibles dentro de una carpeta en el servidor. Al iniciar el PDP se carga en memoria el conjunto de políticas existentes.

4.2 Mecanismos básicos

En esta sección se describen los mecanismos básicos que son reutilizados por los mecanismos complejos. Para cada mecanismo se realiza una descripción, un diagrama de clases, las pre y post condiciones y una tabla con el listado de los parámetros requeridos, los parámetros de configuración del componente y los parámetros del mensaje de salida.

4.2.1 GeoEntryPoint

El punto de entrada para acceder a los servicios que provee SegGeoESB son GeoEntryPointREST, GeoEntryPointSOAP y GeoEntryPointRESTAuth.

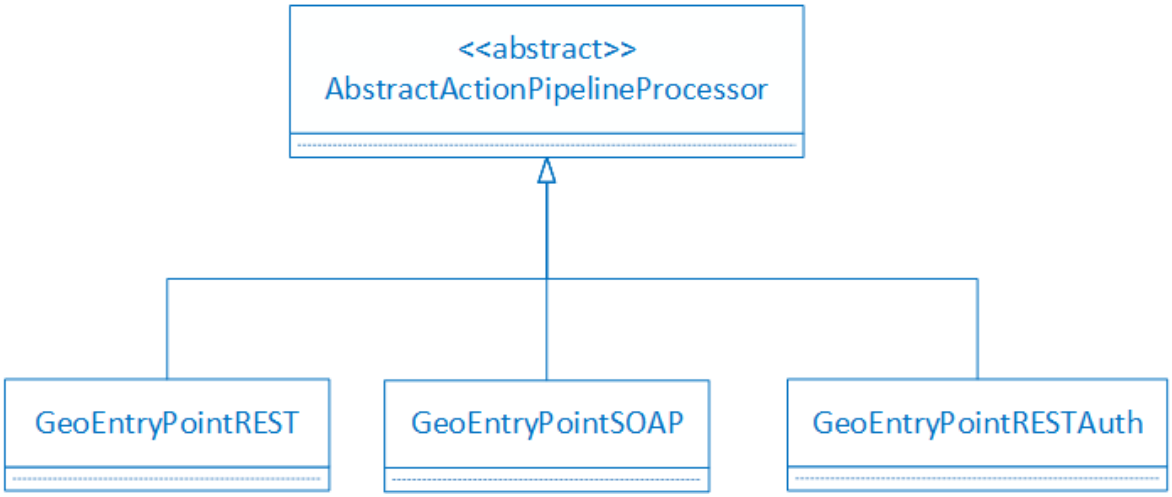

Figura 4.7 - Diagrama de clases de los *GeoEntryPoint*

4.2.1.1 GeoEntryPointREST

Recibe un mensaje ESB conteniendo la petición HTTP recibida por SegGeoESB. Luego se encarga de guardar en el mensaje ESB los datos obtenidos en la solicitud HTTP.

Condiciones

Pre	Post
Petición WMS/WFS REST luego de autenticado el cliente	Se genera un nuevo mensaje con la información recibida en el cuerpo del mensaje. Entre estos datos se identifican: URL, nombre de la operación WMS/WFS y parámetros de consulta

Parámetros requeridos

Nombre del parámetro	Descripción
Mensaje de entrada	
HTTP_REQUEST	Petición recibida por el *HTTPEntryPoint* del ESB para el caso de una petición WMS/WFS REST
Configuración del componente	

Mensaje de salida	
LOCAL_ADDRESS	La dirección del servicio en el cuerpo del mensaje
QUERY_STRING	Los parámetros de la consulta como texto
QUERY_PARAMS	Diccionario conteniendo dichos parámetros de la consulta como propiedad
WS_TYPE	Valor fijo indicando que es de tipo REST y se guarda como propiedad
GEO_REQ_TYPE	Indica si es WMS o WFS, se guarda como propiedad del mensaje
TOKEN	Se parsea el token SAML de la consulta y se guarda en el cuerpo del mensaje

4.2.1.2 GeoEntryPointRESTAuth

Este *GeoEntryPoint* es utilizado para obtener las credenciales del usuario en una autenticación a través de una solicitud REST. Recibe un mensaje ESB conteniendo la petición HTTP recibida por SegGeoESB. Las credenciales se obtienen del header HTTP Authorization y se guardan como propiedad del mensaje.

Condiciones

Pre	**Post**
Petición REST del cliente enviando las credenciales para iniciar la sesión	Se obtienen las credenciales enviadas de parte del cliente y se guardan como propiedad del mensaje

Parámetros requeridos

Nombre del parámetro	**Descripción**
Mensaje de entrada	
HTTP_REQUEST	Petición recibida por el *HTTPEntryPoint* del ESB para el caso de una petición REST de inicio de sesión

Configuración del componente	
Mensaje de salida	
USER_NAME	Usuario del cliente que realiza la petición. Esta se guarda como propiedad del mensaje
PASSWORD	Contraseña del usuario que se guarda como propiedad del mensaje

4.2.1.3 GeoEntryPointSOAP

Recibe un mensaje ESB conteniendo la petición SOAP recibida por la plataforma. Luego se encarga de guardar en el mensaje el cuerpo del mensaje SOAP.

Condiciones

Pre	**Post**
Petición SOAP con información para invocar una operación WMS	Se genera un mensaje ESB con el cuerpo del mensaje SOAP

Parámetros requeridos

Nombre del parámetro	**Descripción**
Mensaje de entrada	
SOAP_REQUEST	Petición recibida por el *HTTPEntryPoint* del ESB para el caso de una petición SOAP.
Configuración del componente	
Mensaje de salida	
SOAP_MESSAGE	Es el cuerpo del mensaje SOAP y se guarda como propiedad del mensaje
WS_TYPE	Valor fijo indicando que es de tipo SOAP y se guarda como propiedad

GEO_REQ_TYPE	Indica si es WMS o WFS. Para esta propiedad siempre se asume que es WMS porque la transformación SOAP-WFS no está soportada

4.2.2 ConfigAggregator

Este componente fue implementado con el objetivo de poder configurar de forma dinámica el flujo de los mensajes. Esto permite por ejemplo habilitar/deshabilitar el control de acceso, habilitar/deshabilitar el enriquecimiento de los datos geográficos con datos empresariales y configurar las urls para acceder al STS. Esto facilita que la plataforma sea más flexible. La extensión de este componente es sencilla, pudiendo agregar otros parámetros de configuración que se consideren necesarios. En la figura 4.8 se muestra el diagrama de clases para este componente.

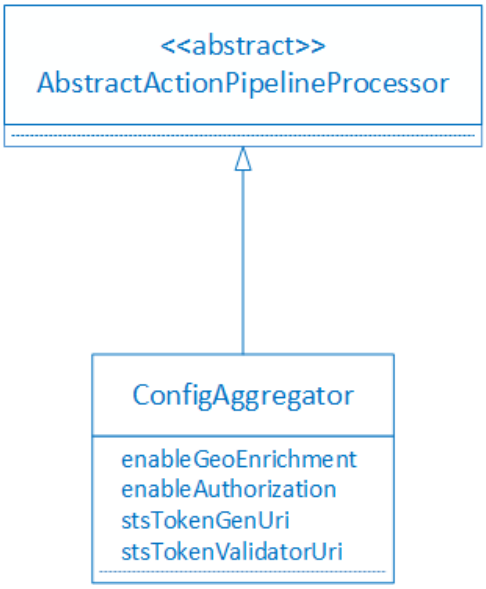

Figura 4.8 - Diagrama de clases de *ConfigAggregator*

Cuando la acción *ConfigAggregator* es invocada, lee todos los atributos que tiene configurados y guarda cada uno como una propiedad en el mensaje ESB para que puedan ser consultados por el resto de las acciones.

Condiciones

Pre	Post
Petición para acceder a un recurso por parte de un cliente	Se cargan como propiedades del mensaje los atributos del componente configurado

Parámetros requeridos

Nombre del parámetro	Descripción
Mensaje de entrada	
Mensaje ESB	
Configuración del componente	
enableGeoEnrichment	Atributo que permite habilitar/deshabilitar la opción de enriquecimiento geográfico. Valor por defecto: true
enableAuthorization	Atributo que permite habilitar/deshabilitar el control de autorización de acceso a un recurso por parte de un usuario. Valor por defecto: true
stsTokenGeneratorUri	Atributo que contiene la URL del Servlet del STS que es responsable de generar el token SAML
stsTokenValidatorUri	Atributo que contiene la URL del Servlet del STS que es responsable de validar el token SAML enviado por el usuario en una solicitud
Mensaje de salida	
ENABLE_AUTHORIZATION	Se agrega como propiedad del mensaje. Si está configurado el atributo *enableAuthorization* se guarda ese valor, de lo contrario se guardar el valor por defecto
ENABLE_GEO_ENRICHMENT	Se agrega como propiedad del mensaje. Si está configurado el atributo *enableGeoEnrichment* se guarda ese valor, de lo contrario se guardar el valor por defecto
STS_TOKEN_GENERATOR_URI	Si el atributo *stsTokenGeneratorUri* está configurado se guarda como propiedad del mensaje
STS_TOKEN_VALIDATION_URI	Si está configurado el atributo *stsTokenValidatorUri* se guarda como propiedad del mensaje

4.2.3 AbstractGeoXACMLRequestCtxRender

La autorización es realizada por el componente externo PDP. Este componente debe recibir como solicitud una instancia serializada[22] del objeto *RequestCtx* , que es construido a partir del WMS o WFS.

El *RequestCtx* contiene los conjuntos de subjects, resources, actions y environment que son las condiciones de correspondencia en XACML (ver sección Lenguaje de Políticas) que son derivados de la petición.

La clase abstracta *AbstractGeoXACMLRequestCtxRender* obtiene de las propiedades del mensaje, el usuario y el diccionario que contiene los parámetros de la consulta. Luego, utilizando el patrón Template Method [42], invoca la implementación del método *getGeoXACMLRequest()* que retorna como resultado una instancia del ya mencionado *RequestCtx*. El objeto es serializado y se guarda en el cuerpo del mensaje.

Hay una implementación concreta para WMS que es *WMSGeoXACMLRequestCtxRender* y una para WFS que es *WMSGeoXACMLRequestCtxRender* que se detallan en las siguientes subsecciones.

En la figura 4.9 se muestra el diagrama de clases de las acciones que fueron descritas anteriormente.

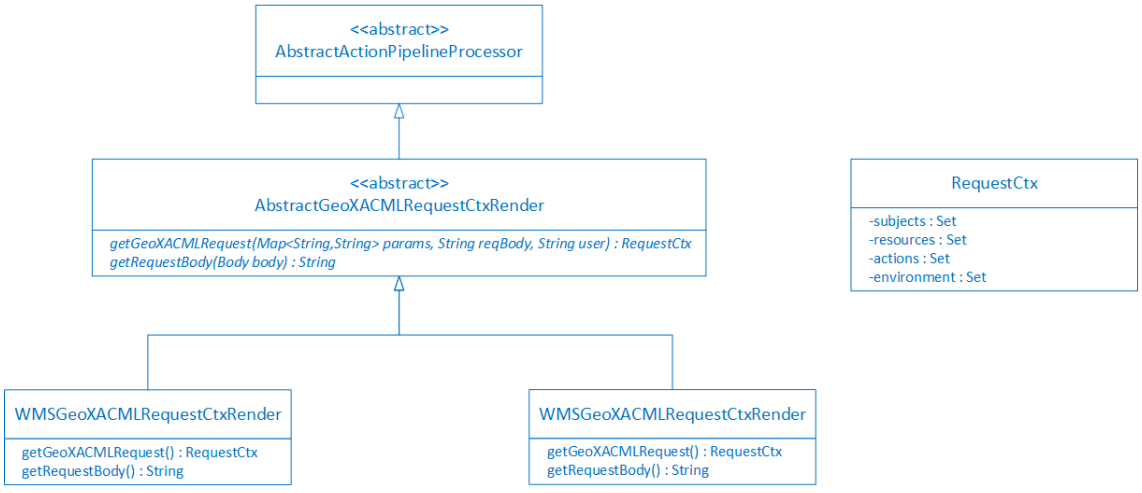

Figura 4.9 - Diagrama de clases de la creación de las solicitudes al PDP

[22] La serialización consiste en un proceso de codificación de un objeto en un medio de almacenamiento con el fin de transmitirlo a través de una conexión en red como una serie de bytes o en un formato humanamente más legible como XML.

4.2.3.1 WMSGeoXACMLRequestCtxRender

Este componente tiene la responsabilidad de crear la solicitud PDP de las operaciones WMS a partir de los parámetros que están en la propiedad QUERY_STRING del mensaje ESB.

Condiciones

Pre	Post
El mensaje tiene la propiedad GEO_REQ_TYPE con el valor WMS	Se ha creado la solicitud PDP, se ha serializado y fue agregado al cuerpo del mensaje

Parámetros requeridos

Nombre del parámetro	Descripción
Mensaje de entrada	
QUERY_STRING	Los parámetros de la consulta como texto como propiedad
USER_NAME	Usuario del cliente que realizó la petición
Configuración del componente	
Mensaje de salida	
GEO_XACML_REQ_CTX	El *RequestCtx* serializado se agrega en el cuerpo del mensaje

4.2.3.2 WFSGeoXACMLRequestCtxRender

Este componente tiene la responsabilidad de crear la solicitud PDP de las operaciones WFS a partir de los parámetros que están en la propiedad QUERY_STRING del mensaje ESB.

Condiciones

Pre	Post
El mensaje tiene la propiedad	Se ha creado la solicitud PDP, se ha serializado y fue

GEO_REQ_TYPE con el valor WFS	agregado al cuerpo del mensaje

Parámetros requeridos

Nombre del parámetro	Descripción
Mensaje de entrada	
QUERY_STRING	Los parámetros de la consulta como texto
USER_NAME	Usuario del cliente que realizó la petición
Configuración del componente	
Mensaje de salida	
GEO_XACML_REQ_CTX	El *RequestCtx* serializado se agrega en el cuerpo del mensaje

4.2.4 GeoPEP

El mecanismo GeoPEP invoca el Web service del PDP con el *RequestCtx* creado previamente y luego procesa la respuesta indicando si la solicitud fue autorizada, denegada o no aplica. La acción *PDPRequester* guarda la solicitud a ser enviada al PDP en el cuerpo del mensaje ESB y la acción *SOAPClient* invoca el Web service del PDP para consultar por el control de acceso al recurso requerido por el cliente. La acción *PDPResponse* evalúa la decisión enviada del PDP y la guarda en la propiedad IS_AUTHORIZED del mensaje que es enrutado por la acción *StaticWiretap* al servicio *ResponseEntry*. En la figura 4.6 está incluido el diagrama de flujo de este componente.

Condiciones

Pre	Post
El mensaje tiene como parte del cuerpo a GEO_XACML_REQ_CTX	Tiene la decisión si el cliente puede acceder al recurso o fue denegada

Parámetros requeridos

Nombre del parámetro	Descripción
Mensaje de entrada	
GEO_XACML_REQ_CTX	El *RequestCtx* serializado se obtiene del cuerpo del mensaje
Configuración del componente	
wsdl	*URI* del wsdl que expone el PDP para invocar el Web service
SOAPAction	Se configura con el valor *evaluate*, para indicar que es la operación que se quiere ejecutar. Se invoca esta operación porque se quiere evaluar el acceso al recurso solicitado
Mensaje de salida	
IS_AUTHORIZED	Se guarda como propiedad del mensaje indicando si el acceso fue autorizado o no

4.3 Composición de servicios encargados de enviar la respuesta

Para obtener el recurso geoespacial del IMS y enviar la respuesta al cliente se implementó una composición de servicios. En la figura 4.10 se muestra un diagrama con los servicios y acciones que lo componen.

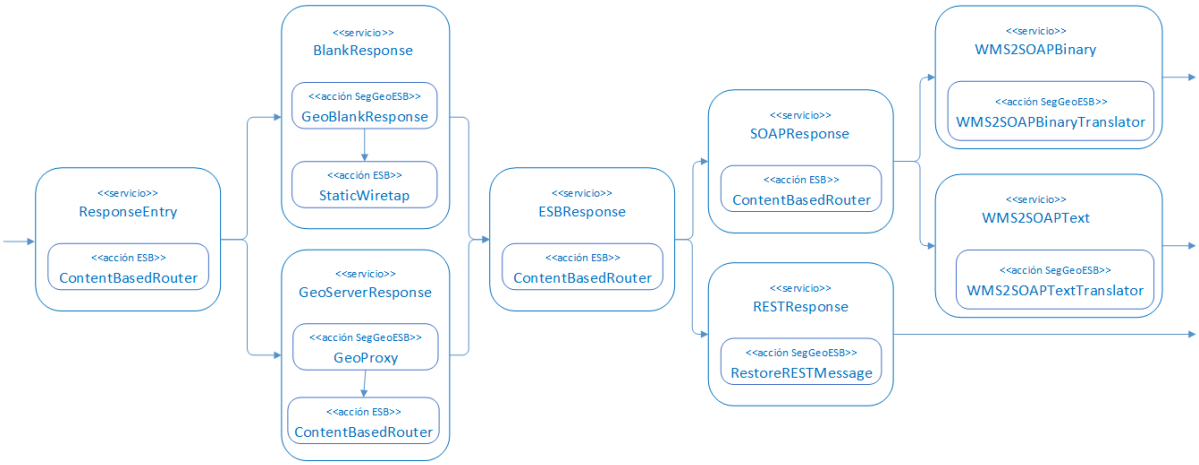

Figura 4.10 - Composición de servicios para el retorno de la respuesta al cliente

El flujo de un mensaje es el siguiente:

1. El servicio ResponseEntry recibe un mensaje ESB y la acción ContentBasedRouter enruta el mensaje basado en la propiedad IS_AUTHORIZED.

2. Si el acceso al recurso fue autorizado se direcciona el mensaje al servicio *GeoServerResponse* y en caso que no haya sido autorizado se direcciona al servicio *BlankResponse*.

    a. El servicio *BlankResponse* mediante el mecanismo *GeoBlankResponse* crea una respuesta “vacía” y se guarda en la propiedad IMS_RESPONSE del mensaje. El formato de la respuesta depende del “content-type” que es uno de los parámetros enviados en la solicitud WMS/WFS. Si el tipo de contenido es una imagen se carga en la propiedad una imagen por defecto, en otro caso se carga un texto vacío. La acción *StaticWiretap* enruta el mensaje al servicio *ESBResponse*.

    b. El servicio *GeoServerResponse* tiene la responsabilidad de realizar la consulta al IMS por el recurso solicitado. El mecanismo *GeoProxy* crea la URL de consulta al IMS a partir de la propiedad QUERY_STRING del mensaje ESB y luego instancia la acción de ESB *HttpRouter* para la cual se configuran las siguientes propiedades: “*endpointUrl*” conteniendo el valor de la URL obtenida y “*method*” con valor “*GET*”, que especifica el método HTTP a utilizar. La respuesta del IMS es almacenada en la propiedad IMS_RESPONSE del mensaje original. Además de esta información se guarda el tipo de contenido de la respuesta en la propiedad del mensaje CONTENT_TYPE. A continuación a través de la acción *ContentBasedRouter* se enruta el mensaje basado en el valor de la propiedad “*enableGeoEnrichment*” que fue cargada en el mecanismo *ConfigAggregator* (ver 4.3.2) previamente. Si el valor de la propiedad es “true” implica que se quiere enriquecer los datos geográficos con información empresarial, por lo que se enruta el mensaje al servicio *GeoEnricher*. Si el enriquecimiento no está habilitado el mensaje es enrutado al servicio *ESBResponse*.

3. El servicio *ESBResponse* enruta el mensaje basándose en la propiedad del mensaje WS_TYPE.

a. Si es de tipo REST lo direcciona al servicio *RESTResponse*. La acción *RestoreRESTMessage* carga la respuesta del IMS que está cargada en la propiedad IMS_RESPONSE como cuerpo del mensaje.

b. Si la solicitud es SOAP direcciona el mensaje al servicio *SOAPResponse*. El mensaje es enrutado basado en el valor de la propiedad CONTENT_TYPE. Si es de tipo binario o de aplicación se enruta al servicio *WMS2SOAPBinary* y si es de tipo texto se enruta el mensaje al servicio *WMS2SOAPText*.

El servicio *WMS2SOAPBinary* está compuesto por el mecanismo *WMS2SOAPBinaryTranslator* que procesa el mensaje ESB codificando a Base64 la respuesta WMS para ser retornada en un mensaje SOAP y que pueda ser interpretado por el cliente Web.

El servicio *WMS2SOAPText* está compuesto por el mecanismo *WMS2SOAPTextTranslator*, que procesa el mensaje insertando la respuesta de WMS en el XML con formato SOAP del mensaje resultante.

4.4 JBoss ESB

JBoss ESB [48] es un producto de Red Hat[23] que implementa una arquitectura ESB en Java, basado en el servidor de aplicaciones JBoss de la misma compañía. Utiliza estándares tanto de Java como de la Web para brindar conectividad y comunicación con muchos otros tipos de tecnologías. En la figura 4.11 se puede ver un esquema de todas las tecnologías de comunicación soportadas por este producto.

[23] https://www.redhat.com/en

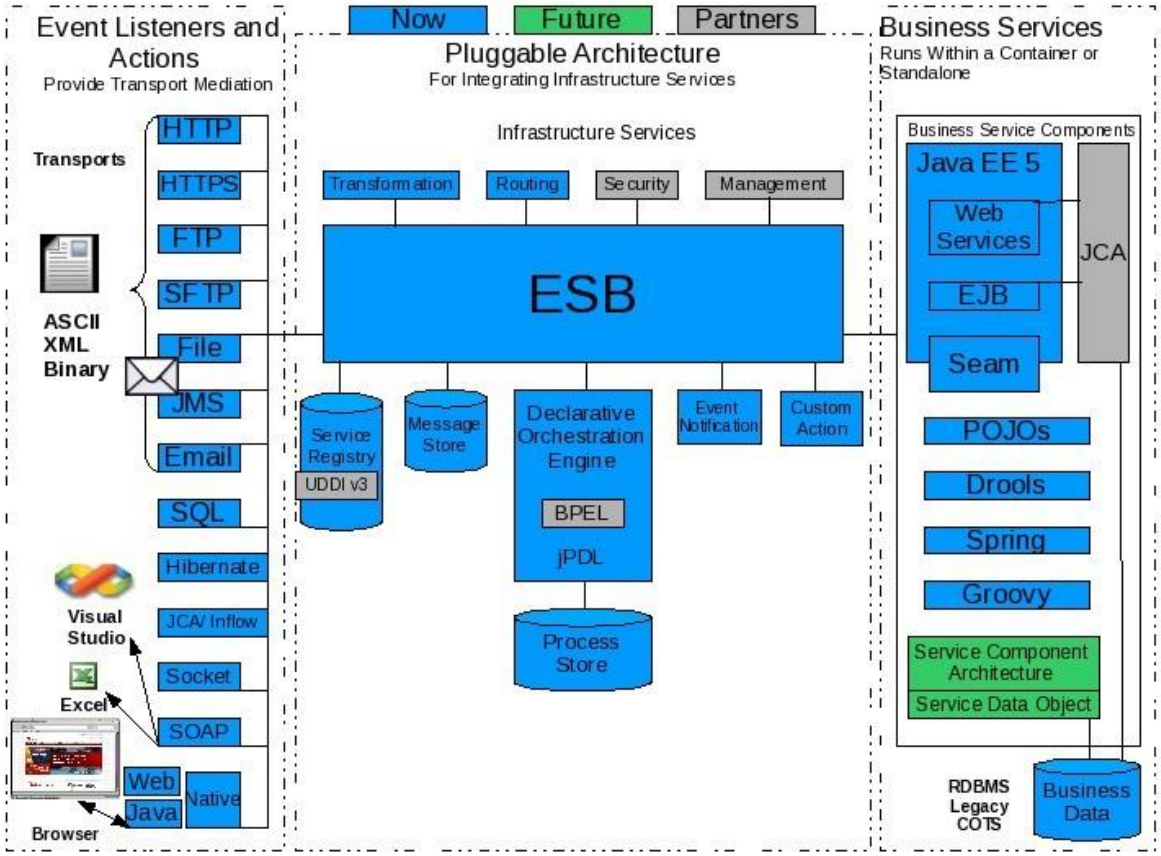

Figura 4.11 - Arquitectura de JBoss ESB [48]

4.4.1 Servicios y Mensajes

Un Service (servicio) en JBoss ESB es definido como una lista de Actions (acciones) que procesan un mensaje ESB de forma secuencial. El servicio puede recibir una lista de Listeners que actúan como routers de entrada del servicio. Los puntos de entrada de un servicio permiten el acceso a los servicios utilizando distintas tecnologías de transporte como pueden ser JMS[24], FTP[25] o HTTP.

Cada servicio tiene declarado una categoría y un nombre como atributos. Estos atributos son utilizados por JBoss ESB cuando es desplegado el servicio para registrarlo en el Registro de servicios. Este registro permite publicar los servicios para que puedan ser localizados por otros servicios.

[24] JMS (Java Message Service) es la solución para las colas de mensajes de Sun Microsystems.

[25] FTP (File Transfer Protocol) es un protocolo de red para la transferencia de archivos.

Las acciones son ejecutadas cuando un mensaje es recibido por el servicio y son las que realizan las tareas del servicio. Las acciones son utilizadas para enrutar los mensajes, aplicar transformaciones de un dato a otro, ejecutar scripts y más tipos de actividades.

Como parte de un servicio, las acciones se combinan generalmente en una cadena de acciones, donde un solo mensaje será procesado por múltiples acciones. Cada acción suele realizar una sola acción atómica en un mensaje, pero la combinación de ellas dentro de una cadena permite que los servicios puedan llevar a cabo tareas complejas.

Hay dos tipos de acciones, las que son parte del paquete de acciones que están incluidas en JBoss ESB y las personalizadas que son diseñadas y creadas por parte de los desarrolladores que utilicen el ESB.

Todas las interacciones entre los clientes y servicios dentro de JBoss ESB son a través del intercambio de mensajes. Cada mensaje está compuesto principalmente por un cabezal, un contexto, un cuerpo y un conjunto de propiedades. El cabezal contiene información de ruteo y de direccionamiento. El contexto contiene la información relacionada a la sesión como puede ser el contexto de la transacción o de la seguridad. El cuerpo generalmente contiene la carga del mensaje pero puede contener también una lista de objetos de distintos tipos. Las propiedades pueden ser utilizadas para definir metadata adicional del mensaje.

4.4.2 Acciones disponibles del ESB

JBoss ESB provee una serie de acciones ya implementadas y que pueden ser categorizadas por su funcionalidad. Hay acciones de ruteo, específicas para Web services, de transformación y de notificación entre otras. En esta sección se mencionan sólo las acciones que fueron evaluadas en el análisis. Se describirán las acciones de ruteo *HttpRouter*, *ContentBasedRouter*, *StaticWiretap* y *SyncServiceInvoker*, la acción *SOAPClient* que está dentro de las específicas para el procesamiento de Web services y la acción *SmooksAction* que está incluida dentro del grupo de acciones de transformación.

4.4.2.1 HttpRouter

Permite la invocación a un Http endpoint externo al ESB desde una acción que está dentro del contexto del ESB mediante el uso de la librería HttpClient[26] de Apache Commons.

[26] HttpClient es una implementación de un agente HTTP compatible con la versión 1.1 del protocolo.

4.4.2.2 ContentBasedRouter

Rutea el mensaje basado en el contenido. Es una implementación del patrón Content Based Router[27] Esta acción soporta las siguientes reglas de ruteo:

- Reglas XPath[28] definidas como atributos de la acción o en un archivo externo.
- Drools[29] mediante archivos con reglas DSL[30].

4.4.2.3 StaticRouter

Es una acción de ruteo estático a diferencia de *ContentBasedRouter*. Se configura como propiedad de la acción del servicio al que serán ruteados los mensajes. Es una implementación del patrón de mensajería Message Router[31].

4.4.2.4 StaticWiretap

La acción *StaticWiretap* permite realizar un ruteo estático de un mensaje y posibilita que las acciones que están por debajo de ella en el pipeline puedan ser ejecutadas.

4.4.2.5 SyncServiceInvoker

Esta acción hace una invocación síncrona en el servicio configurado y pasa la respuesta de invocación de nuevo al pipeline de acciones para su procesamiento por las acciones posteriores (si las hubiera) o como la respuesta si el servicio es de tipo *RequestResponse*[32].

---

[27] http://www.enterpriseintegrationpatterns.com/patterns/messaging/ContentBasedRouter.html

[28] XPath (XML Path Language) es un lenguaje que permite construir expresiones que recorren y procesan un documento XML, similares a expresiones regulares.

[29] Drools es un sistema de gestión de reglas de negocio de JBoss que utiliza un motor de reglas basado en inferencia de encadenamiento hacia adelante y hacia atrás.

[30] DSL (Domain Specific Language) es un lenguaje de programación o especificación dedicado a resolver un problema en particular.

[31] http://www.enterpriseintegrationpatterns.com/patterns/messaging/MessageRouter.html

[32] Este tipo de servicios al finalizar el pipeline de acciones redirige el mensaje al servicio que lo invocó

4.4.2.6 SOAPClient

Utiliza el servicio de cliente SoapUI [49] para construir y publicar un mensaje para el servicio de destino. Esta acción entonces enruta ese mensaje a ese servicio.

4.4.2.7 SmooksAction

SmooksAction puede procesar (utilizando las funcionalidades de Smooks) distintos tipos de carga (payload) de los mensajes ESB como pueden ser strings, arreglos de bytes, objetos Java, entre otros. El procesamiento de esta carga puede ser aplicar una transformación de un tipo de datos a otro o enriquecer un mensaje utilizando una fuente de datos (Datasource).

Smooks [47] es una librería y motor de JBoss desarrollado en Java para procesar tanto datos XML como no XML como son CSV, JSON.

4.5 Tecnologías utilizadas

En esta sección se presentan todas las tecnologías y herramientas utilizadas en el proyecto.

Para el entorno de desarrollo se utilizó Ubuntu[33] en su versión 14.04 como sistema operativo y la plataforma Java SDK 1.7.0_80[34].

Se utilizó el servidor de aplicaciones JBoss AS 6.1.0.Final[35]. El mismo tiene incorporado el servidor JBoss ESB 4.12[36]. El IMS usado fue GeoServer 2.7.1.1, el cual accede al motor de base de datos PostgreSQL 9.3[37]. Para publicar las capas en GeoServer se utiliza la extensión de PostgreSQL PostGIS 2.1[38].

El IDE (Integrated Development Environment) de desarrollo utilizado fue Eclipse Kepler[39]. Para el versionado del código se utilizó la solución de Git Bitbucket[40].

[33] http://www.ubuntu.com/

[34] http://www.oracle.com/technetwork/es/java/javase/downloads/jdk7-downloads-1880260.html

[35] http://jbossas.jboss.org/

[36] http://jbossesb.jboss.org/docs/4_12

[37] http://www.postgresql.org/

[38] http://postgis.net/

[39] https://eclipse.org/kepler/

En tanto para el Caso de Estudio se utilizó OpenLayers 2.12[41] para la visualización del mapa e invocación de Web services geoespaciales y JQuery 1.10.2[42] para la manipulación de los datos geográficos y el HTML.

El cliente accede al Caso de Estudio utilizando el navegador web Mozilla Firefox 37.0.1[43]. Para la ejecución de pruebas de solicitudes SOAP se utilizó la herramienta de testing SoapUI 5.1.3[44].

[40] https://bitbucket.org/

[41] http://openlayers.org/two/

[42] https://jquery.com/

[43] https://www.mozilla.org/

[44] https://www.soapui.org/

5 Caso de estudio

En este capítulo se presenta un caso de estudio que tiene como propósito utilizar en conjunto la gama de mecanismos que ofrece el sistema SegGeoESB.

El caso de estudio tiene como objetivo la elección en un mapa de un local de cobro del Banco de Previsión Social[45] (BPS), que es el organismo de seguridad social que paga las jubilaciones y pensiones en Uruguay. El cliente de la aplicación puede seleccionar el local de cobro más cercano para el pago de pasividades.

Los locales de cobro son una de las capas disponibles en la aplicación, pero no tienen toda la información necesaria para el usuario, como ser la dirección, barrio, nombre del local, etc. Por este motivo se accede a un Web service de negocio para brindar la información restante. Esta información es brindada por un Servidor de datos empresariales.

El sistema SegGeoESB controlará en cada consulta WMS y WFS que el usuario esté autorizado. Se definieron dos usuarios que permiten evaluar los mecanismos de autorización:

- Usuario Administrador
- Usuario Normal

El administrador está autorizado a ver todas las capas y el usuario normal tiene restringido a una zona geográfica el acceso a la capa de locales de cobro.

Esta aplicación Web está basada en la que se desarrolló para la evaluación de la plataforma GeoEEIP [21]. Está implementada mediante el uso de OpenLayers[46] y JQuery[47] como tecnologías base para el front-end. La información geográfica será obtenida a través de GeoServer [50].

En la sección 5.1 se detalla el despliegue de los componentes del Caso de Estudio. La descripción y configuración de los componentes son abordados en la sección 5.2. Los escenarios de prueba son explicados en la sección 5.3. Las pruebas realizadas para las

[45] http://www.bps.gub.uy/

[46] Es una biblioteca de JavaScript para mostrar mapas interactivos en los navegadores web.

[47] Es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML.

peticiones SOAP están planteadas en la sección 5.4. Finalmente en el punto 5.5 se detallan los resultados obtenidos.

5.1 Despliegue de los componentes

En la figura 5.1 se muestran los artefactos que son parte del diagrama de despliegue para el Caso de Estudio. Dentro de la figura podemos apreciar cuatro artefactos que interactúan entre sí para llevar a cabo el caso de estudio. Tenemos por un lado el cliente (browser), luego el servidor de mapas (GeoServer), la base de datos (PostgreSQL) y por último el servidor de aplicaciones (JBoss AS).

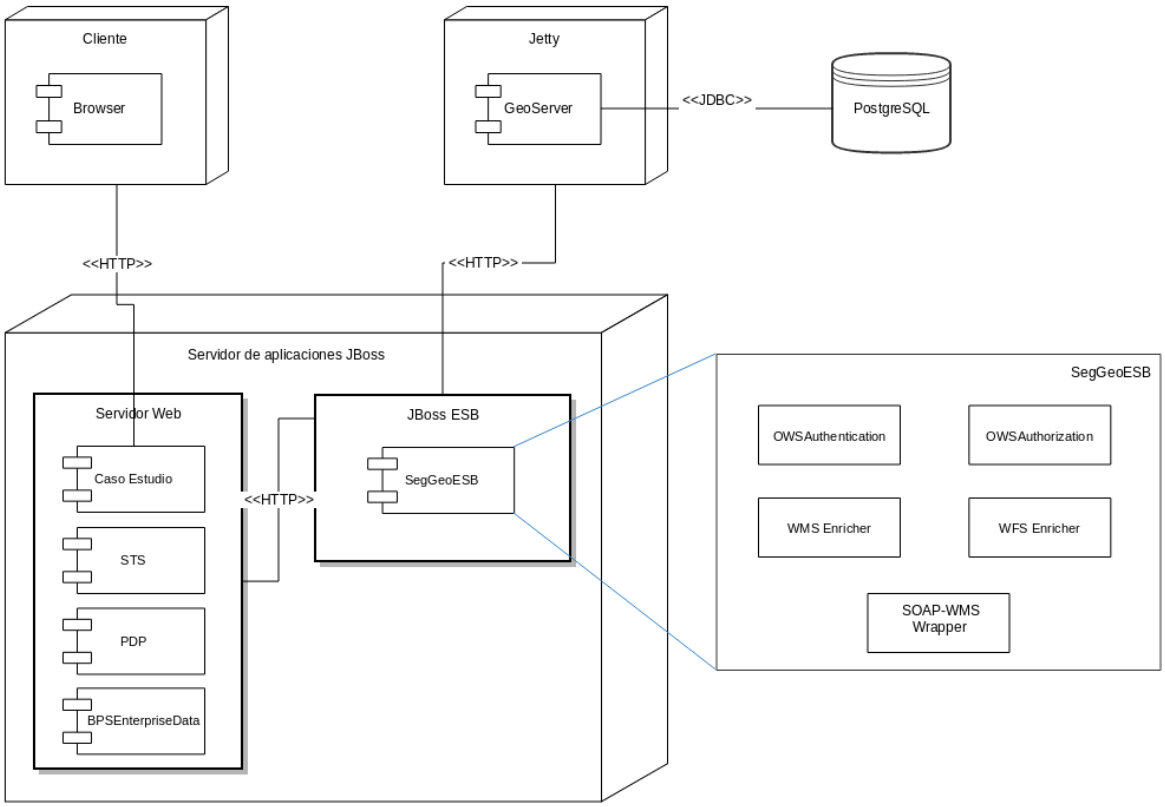

Figura 5.1 - Diagrama de despliegue de los componentes

El Cliente permite al usuario realizar consultas al servidor de aplicaciones desde el cliente utilizando el protocolo HTTP.

GeoServer corre sobre un servidor Jetty[48] con los datos geográficos a ser consultados. Jetty es un servidor HTTP y contenedor de Servlets escrito en Java. El GeoServer se comunica con la base de datos PostgreSQL por JDBC[49].

Por último está el servidor de aplicaciones JBoss AS [56] que contiene al JBoss ESB y al Servidor Web. En JBoss ESB está desplegada la aplicación SegGeoESB, que es la que contiene todos los mecanismos de integración. En el Servidor Web están desplegados los componentes externos Caso Estudio, STS, PDP y BPSEnterpriseData. Estos componentes se comunican por medio de HTTP con el JBoss ESB.

5.2 Descripción de los componentes

Los componentes involucrados en el caso de estudio son descritos en las siguientes subsecciones.

5.2.1 PDP

Como se explicó en capítulos anteriores el servidor PDP nos permite obtener una decisión de autorización, para lograr este propósito se debieron configurar un conjunto de políticas que cubrieran los escenarios de prueba definidos. Estas políticas se cargan en sesión para luego ser utilizadas en el momento de autorizar un usuario a un recurso dado.

Cada política aplica específicamente a un usuario de los citados anteriormente. A continuación se explica brevemente cuales son las políticas que se han creado.

- *policy-admin*: Administra el acceso de un administrador. En este caso está permitido el acceso a todos los recursos independientemente de la operación invocada.
- *policy-serv_por_zona*: Permite el acceso a las capas de ejes, manzanas y espacios libres para todas las operaciones y todo el espacio geográfico.
- *policy-serv_comerciales*: Permite el acceso para la capa de servicios comerciales, limitando el acceso a las operaciones WMS para una región geográfica en particular.

[48] http://www.eclipse.org/jetty/

[49] Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute.

La combinación de las últimas dos políticas definidas son aplicadas al usuario normal, ya que el administrador no tiene restricciones de autorización. En el apéndice se aprecia una información más detallada de estas políticas.

5.2.2 GeoServer

Aquí se publican las capas necesarias para el Caso de Estudio. Estas son de manzanas, calles, espacios libres y servicios comerciales. GeoServer obtiene información asociada a las capas desde la base de datos PostgreSQL, mediante el uso de PostGIS. PostGIS es un módulo que añade soporte de objetos geográficos a PostgreSQL

A continuación se detallan las capas mencionadas anteriormente:

- *manzanas*: Es una capa de polígonos con las manzanas de Montevideo y número de carpeta catastral[50]. Se toma como la capa base y están disponibles para las consultas WMS.
- *calles*: Es una capa de líneas con las vías de tránsito de Montevideo[51] y están publicadas por WMS.
- *servicios comerciales*: Es una capa de puntos que identifica a los locales de cobro publicadas por WMS.
- *espacios libres*: Es una capa polígonos de los espacios públicos (plazas, parques, canteros, separadores viales, etc)[52]. Están publicadas por WFS.

Todas las capas fueron obtenidas del Catálogo de Datos Abiertos de la Intendencia de Montevideo[53] y están en el Sistema de Referencia UTM 21S con datum WGS84 (código EPSG:32721).

Además de las capas mencionadas previamente se deben cargar los estilos que tienen asociadas.

[50] https://catalogodatos.gub.uy/dataset/manzanas

[51] https://catalogodatos.gub.uy/dataset/vias-de-transito-montevideo

[52] https://catalogodatos.gub.uy/dataset/espacios-libres-montevideo

[53] https://catalogodatos.gub.uy/organization/intendencia-montevideo

5.2.3 BPSEnterpriseData

Esta aplicación fue desarrollada dentro del proyecto GeoEEIP y permite simular datos de negocio que tienen valor para el caso de estudio. Es una implementación de un servidor de Servicios Empresariales como fue descrito en la sección 3.1.2. La aplicación expone dos Web services (*EspaciosLibresWS* y *BPSLocalesPagoWS*, ver sección 2 del documento anexo) ofreciendo las operaciones definidas a continuación:

- *infoLocal*, que retorna los datos de un local a partir de un identificador (obtenido de la entidad geográfica).
- *infoEspaciosLibres*, la cual retorna información acerca del espacio libre consultado a partir del identificador. Para obtener el identificador es necesaria una función de correlación, ya que el valor de la entidad geográfica es diferente del valor del identificador de negocio requerido por el servicio.

5.2.4 SegGeoESB

En SegGeoESB se debe configurar la comunicación con cada uno de los componentes. Esta configuración se realiza en el archivo *jboss-esb.xml* que es parte de la aplicación SegGeoESB. La configuración completa de todos los mecanismos está disponible en la sección 1 del documento anexo.

El mecanismo *ConfigAggregator* (ver 4.2.2) permite habilitar la autorización, el enriquecimiento de los datos geográficos con datos empresariales y además es donde se configura la comunicación con el STS. Un ejemplo de esta configuración se muestra en el cuadro 5.1.

```
<action class="uy.edu.fing.seggeoesb.mecanismos.basicos.ConfigAggregator"
        name="ConfigAggregator">
   <property name="enableAuthorization" value="true"/>
   <property name="enableGeoEnrichment" value="true"/>
   <property name="sts_token_generator_uri"
             value="http://localhost:8080/STSService/STSServerServlet"/>
   <property name="sts_token_validation_uri"
             value="http://localhost:8080/STSService/STSValidateTokenServlet"/>
</action>
```

Cuadro 5.1 - Configuración de *ConfigAggregator*

Para la comunicación con el PDP se debe configurar dentro del mecanismo GeoPEP, la acción *SOAPClient* que está disponible en JBoss ESB. Un ejemplo de esta configuración se puede apreciar en el cuadro 5.2.

```
<action class="org.jboss.soa.esb.actions.soap.SOAPClient" name="ClienteEvaluatePDP">
   <property name="wsdl" value="http://localhost:8080/PDP/services/WSPDP?wsdl" />
   <property name="SOAPAction" value="evaluate" />
   <property name="responseAsOgnlMap" value="true" />
</action>
```

Cuadro 5.2 - Configuración de comunicación con el PDP

Para comunicación con el IMS se debe establecer la url en la propiedad *imsName* del mecanismo GeoProxy, como se puede ver en el cuadro 5.3.

```
<action class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoProxy"
        name="GeoProxy">
   <property name="imsName" value="http://localhost:8081/geoserver/" />
</action>
```

Cuadro 5.3 - Configuración de la comunicación con GeoServer

Para la comunicación con el servidor de Servicios Empresariales se debe configurar la url del wsdl que se quiere invocar. En el cuadro 5.4 se muestra un ejemplo en donde se establece la url en la propiedad *wsdl* del mecanismo *GeoFeatureInfoEnricher*.

```
<action class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoFeatureInfoEnricher"
         name="GeoFeatureInfoEnricher">
   ...
   <property name="wsdl"
             value="http://localhost:8080/BPSEnterpriseDataA/BPSLocalesPagoWS?wsdl"/>
   <property name="SOAPAction" value="infoLocal"/>
   ...
</action>
```

Cuadro 5.4 - Configuración de la comunicación con BPSEnterpriseData

5.3 Pruebas del sistema

Los escenarios de prueba están basados en los usuarios definidos en la descripción del Caso de Estudio. Para llevar a cabo los casos de uso mencionados anteriormente fue necesario generar una serie de políticas de acceso que serán consultadas por el PDP en el

momento de la autorización. Estas políticas fueron explicadas previamente en la sección 5.2.1.

A continuación se describe el flujo desde que se recibe una petición por parte del cliente hacia el SegGeoESB. Dicho flujo se puede ver en la figura 5.2.

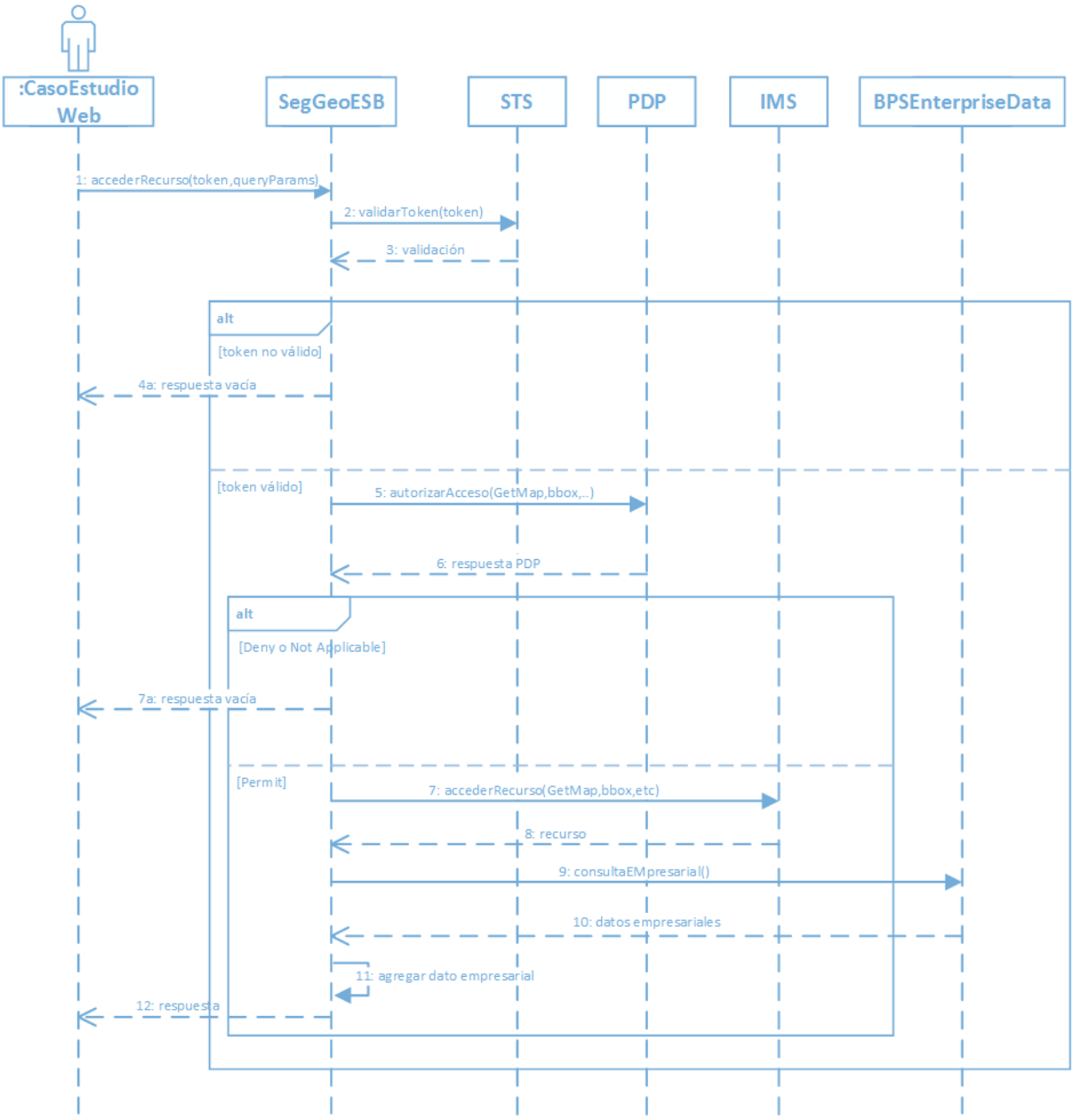

Figura 5.2 - Diagrama de secuencia de caso de estudio para solicitudes REST

En primer lugar, como se muestra en la figura 5.3 la aplicación web solicita al usuario que ingrese sus credenciales. Los datos del usuario son enviados al ESB, que se encarga de generar un mensaje para enviarlo al STS. El usuario es autenticado por el STS el cual

genera el token de seguridad que es enviado al ESB. Un ejemplo de dicho token se puede ver en la sección 4.1 del documento anexo.

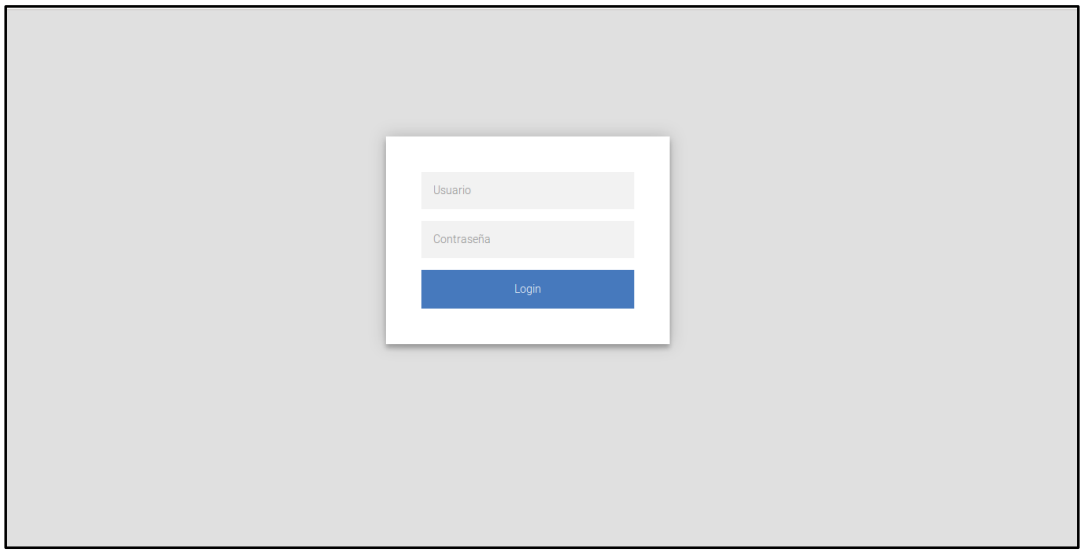

Figura 5.3 - Página de login

Luego que el usuario está autenticado, la aplicación realizará las consultas WMS y WFS necesarias para poder desplegar el mapa. El token de seguridad obtenido en el inicio de sesión es agregado en cada consulta realizada. SegGeoESB se encarga de evaluar los pedidos a los Web services geoespaciales (WMS/WFS) contra el PDP para determinar si el usuario autenticado en la aplicación tiene autorizado el acceso a los recursos solicitados.

El PDP envía una respuesta con la decisión, ya sea que no tiene acceso (valor “Deny”), que no pudo tomar una decisión (“Not applicable”) o que tiene acceso (“Permit”). En caso de que sea denegado, SegGeoESB envía una respuesta vacía al cliente. Por otro lado si el PDP responde con la decisión de que puede acceder, SegGeoESB autoriza el acceso a los recursos para obtener los recursos se consulta al IMS, una vez que se tenga la respuesta del IMS.

En este momento, en caso de que la capa y la operación lo tengan habilitado, se hará la comunicación con el Web service de negocio correspondiente para enriquecer los datos geográficos con la información obtenida del servidor de datos empresariales *BPSEnterpriseData*. Finalmente SegGeoESB enviará la respuesta al cliente, que desplegará la información solicitada como se ve en la figura 5.4. A continuación se describe la información que es desplegada al cliente.

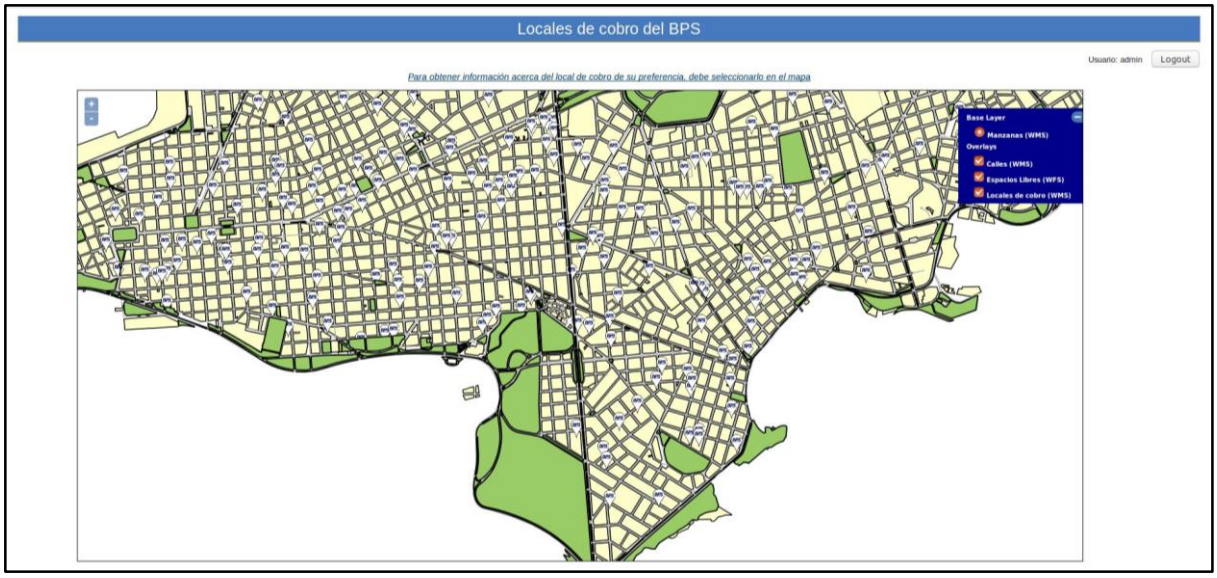

Figura 5.4 - Página de mapas

La capa de servicios comerciales muestra los puntos en el mapa donde están posicionados los locales de cobro de BPS. Al seleccionar uno de los locales, SegGeoESB obtiene los datos empresariales (nombre del local, dirección, barrio) de *BPSLocalesPagoWS*. En la imagen 5.5 se puede apreciar un ejemplo de lo descrito previamente.

Figura 5.5 - Local de cobro con datos empresariales

La capa de Espacios libres permite visualizar los espacios públicos de Montevideo (plazas, parques, canteros, separadores viales, etc). Al seleccionar uno de estos, se obtiene la

información de negocio obtenida del Web Service empresarial *EspaciosLibresWS* que retorna las prestaciones que ofrece dicho espacio libre (bancos, baños, basureros, hamacas, etc), un ejemplo de esto se puede ver en la figura 5.6.

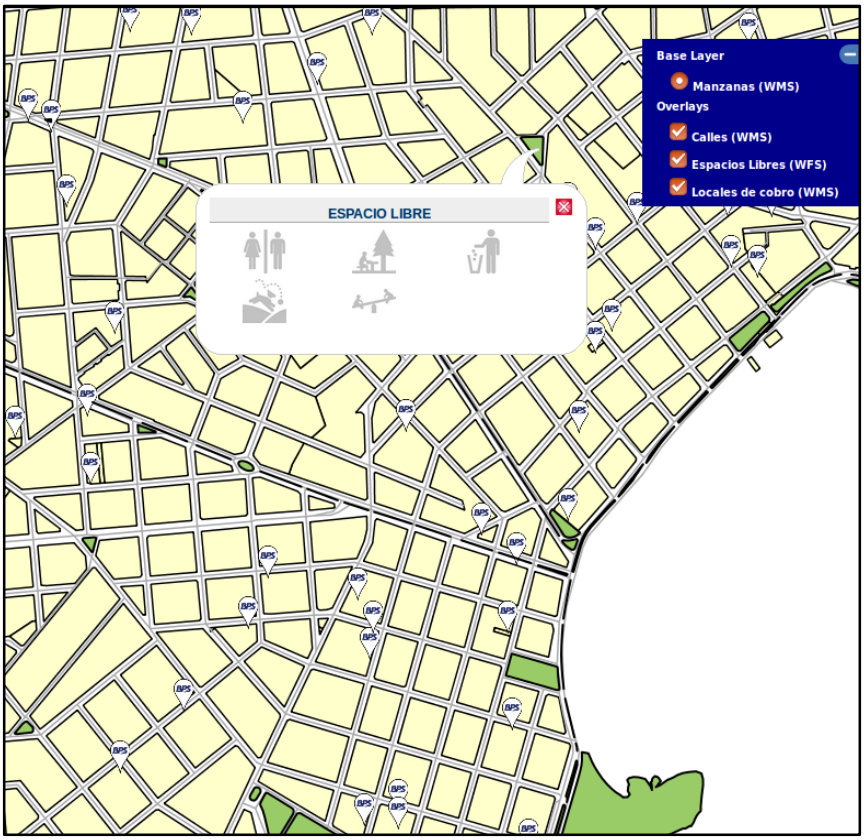

Figura 5.6 - Capa de espacios libres

5.4 Cliente SOAP

Otro aspecto importante de la implementación es la capacidad de consumir Web services geoespaciales a través de una petición SOAP. Para ejemplificar esto se utiliza como herramienta la aplicación de escritorio SoapUI [49]. Estos mensajes son enviados hacia el SegGeoESB que será el encargado de procesar estos datos y generar luego la comunicación con el IMS.

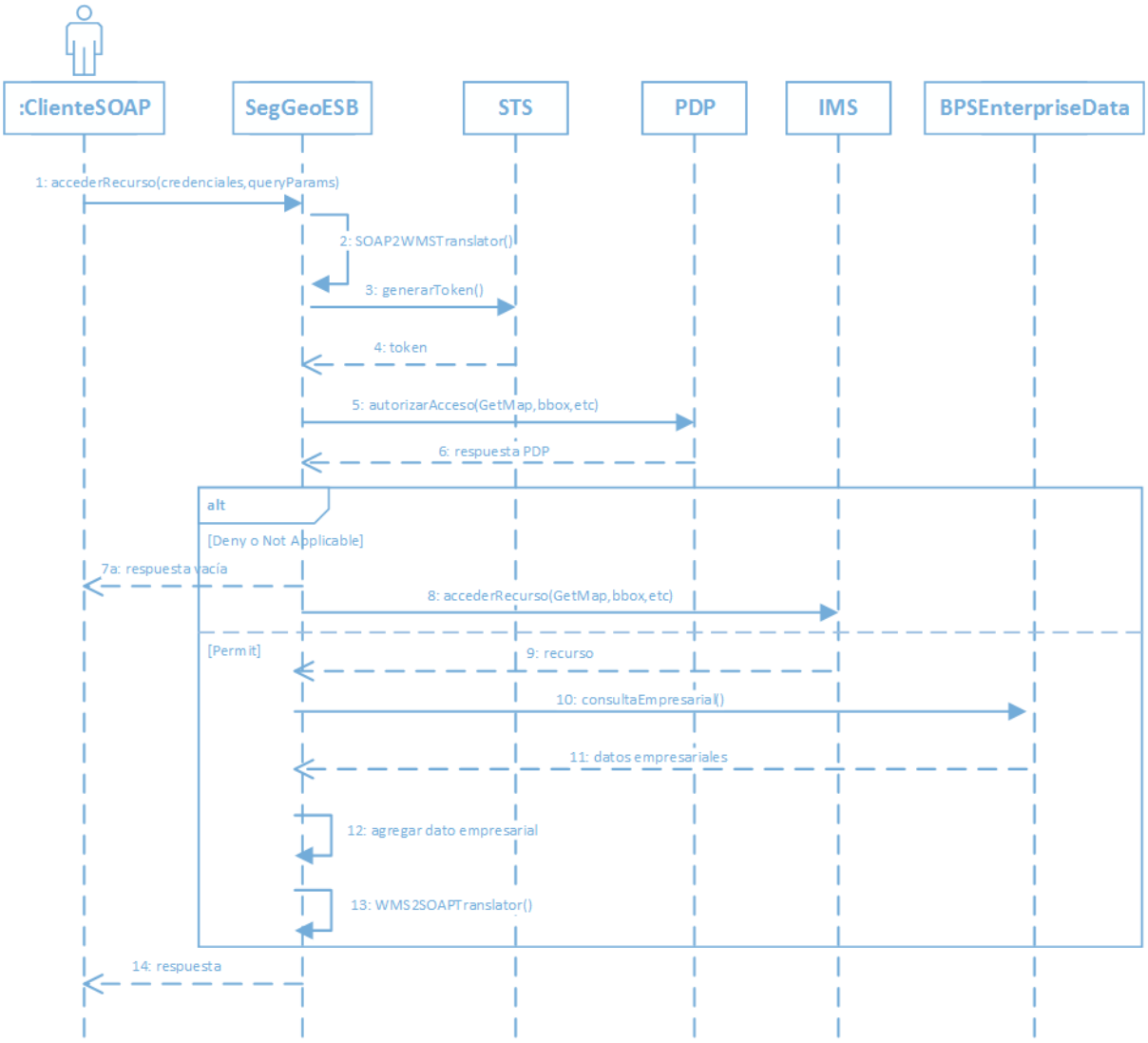

Figura 5.7 - Diagrama de secuencia para solicitudes SOAP

A continuación y como se puede ver en la figura 5.7 se detalla el flujo de una petición SOAP hacia el SegGeoESB.

Como se describió en la sección 3.5, en el caso de la autenticación las credenciales del cliente van a estar contenidas en el mensaje SOAP junto con la petición del recurso. Dicha petición llega al SegGeoESB que es quién se encarga de enviar las credenciales del usuario al STS para generar el token de acceso.

Luego que el usuario es autenticado, SegGeoESB se encarga de realizar la transformación de SOAP a REST. A partir de este momento se realiza la autorización del usuario al recurso solicitado. Igual a como se describió en la sección anterior el SegGeoESB consulta al PDP

para permitir o denegar el acceso al mismo. Una vez que se determina que el usuario tiene acceso al recurso, el SegGeoESB se encarga de realizar el pedido al IMS.

Luego que el SegGeoESB tiene la respuesta del IMS realiza otra transformación, generando una respuesta SOAP que contiene la respuesta original recibida por el IMS. Dicha respuesta es la que luego es enviada al cliente SoapUI.

Un ejemplo de una solicitud SOAP de un WMS para la operación GetCapabilities se puede ver en el cuadro 5.5.

```
<soapenv:Envelope xmlns:eeip="http://www.fing.edu.uy/eeip"
                  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
                               wss-wssecurity-secext-1.0.xsd"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
                 utility-1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-E47F24B934295442CF14621441331294">
        <wsse:Username>user</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
                             wss-username-token-profile-1.0#PasswordText">
          password
        </wsse:Password>
        <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
                                  wss-soap-message-security-1.0#Base64Binary">
          CUUTTS5FCEVYZAoY10Xc/A==
        </wsse:Nonce>
        <wsu:Created>2016-05-01T23:08:53.129Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <eeip:GetCapabilities>
      <eeip:version>1.3.0</eeip:version>
      <eeip:service>wms</eeip:service>
      <eeip:exceptions>application_vnd_ogc_se_xml</eeip:exceptions>
      <!--Optional:-->
      <eeip:section name="/">
        <eeip:RegEntry regEntryId="?" updateSequence="?"/>
      </eeip:section>
      <eeip:style>full</eeip:style>
    </eeip:GetCapabilities>
  </soapenv:Body>
</soapenv:Envelope>
```

Cuadro 5.5 - Solicitud SOAP de la operación GetCapabilities

La solicitud SOAP está formada por dos grandes partes, el encabezado (que es la etiqueta *<soapenv:Header>*) y el cuerpo del mensaje (que es la etiqueta *<soapenv:Body>*).

La sección de encabezado contiene el encabezado de WS-Security (ver 2.3.1) para el envío del usuario (*<wsse:Username>*) y la contraseña (*<wsse:Password>*).

En la sección del cuerpo del mensaje están las etiquetas propias de GeoEEIP. La etiqueta envolvente *<eeip:GetCapabilities>* indica la operación de la solicitud, en este caso GetCapabilities. El resto indican la versión utilizada de GeoXACML, el servicio (wms) y otros parámetros que pueden ser opcionales. Estas etiquetas van a ser interpretadas por el mecanismo de WMS-SOAP Wrapper para realizar la transformación a WMS.

Un ejemplo de respuesta de la solicitud anterior se puede ver en el cuadro 5.6.

```
<eeip:GetCapabilitiesResponse
       xsi:schemaLocation="http://www.fing.edu.uy/eeip/wms_schemas.xsd"
       xmlns:eeip="http://www.fing.edu.uy/eeip"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <eeip:GetCapabilitiesResult>
    <![CDATA[<?xml version="1.0" encoding="UTF-8"?>
      <WMS_Capabilities version="1.3.0" updateSequence="135"
                xmlns="http://www.opengis.net/wms"
                xmlns:xlink="http://www.w3.org/1999/xlink"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.opengis.net/wms
              http://localhost:8081/geoserver/schemas/wms/1.3.0/capabilities_1_3_0.xsd">
        <Service>
          <Name>WMS</Name>
          <Title>GeoServer Web Map Service</Title>
          <Abstract>A compliant implementation of WMS plus most of the SLD extension
                    (dynamic styling). Can also generate PDF, SVG, KML, GeoRSS
          </Abstract>
          <KeywordList>
                 <Keyword>WFS</Keyword>
                 <Keyword>WMS</Keyword>
                 <Keyword>GEOSERVER</Keyword>
          </KeywordList>
          <OnlineResource xlink:type="simple"
                  xlink:href="http://geoserver.sourceforge.net/html/index.php"/>
          <ContactInformation>
            ...
          </ContactInformation>
          <Fees>NONE</Fees>
          <AccessConstraints>NONE</AccessConstraints>
        </Service>
        <Capability>
          <Request>
            <GetCapabilities>
              <Format>text/xml</Format>
              <DCPType>
                <HTTP>
                  <Get>
                    <OnlineResource xlink:type="simple"
                   xlink:href="http://localhost:8081/geoserver/ows?SERVICE=WMS&amp;"/>
                  </Get>
                  <Post>
                    <OnlineResource xlink:type="simple"
                   xlink:href="http://localhost:8081/geoserver/ows?SERVICE=WMS&amp;"/>
                  </Post>
                </HTTP>
              </DCPType>
            </GetCapabilities>
            <GetMap>
              <Format>image/png</Format>
```

```
            ...
            <Format>text/html; subtype=openlayers</Format>
            <DCPType>
              <HTTP>
                <Get>
                  <OnlineResource xlink:type="simple"
                  xlink:href="http://localhost:8081/geoserver/ows?SERVICE=WMS&amp;"/>
                </Get>
              </HTTP>
            </DCPType>
          </GetMap>
          <GetFeatureInfo>
            <Format>text/plain</Format>
            ...
            <Format>application/json</Format>
            <DCPType>
              <HTTP>
                <Get>
                  <OnlineResource xlink:type="simple"
                   xlink:href="http://localhost:8081/geoserver/ows?SERVICE=WMS&amp;"/>
                </Get>
              </HTTP>
            </DCPType>
          </GetFeatureInfo>
        </Request>
        <Exception>
          <Format>XML</Format>
          <Format>INIMAGE</Format>
          <Format>BLANK</Format>
        </Exception>
        <Layer queryable="1" opaque="0">
          <Name>seggeo:avenidas</Name>
          <Title>avenidas</Title>
          <Abstract/>
          <KeywordList>
            <Keyword>avenidas</Keyword>
            <Keyword>features</Keyword>
          </KeywordList>
          <CRS>EPSG:32721</CRS>
          <CRS>CRS:84</CRS>
          <EX_GeographicBoundingBox>
            <westBoundLongitude>-56.35579919675954</westBoundLongitude>
            <eastBoundLongitude>-56.030599000528795</eastBoundLongitude>
            <southBoundLatitude>-34.93034799339506</southBoundLatitude>
            <northBoundLatitude>-34.718249758275576</northBoundLatitude>
          </EX_GeographicBoundingBox>
          <BoundingBox CRS="CRS:84" minx="-56.35579919675954"
                       miny="-34.93034799339506" maxx="-56.030599000528795"
                       maxy="-34.718249758275576"/>
          <BoundingBox CRS="EPSG:32721" minx="558984.098820564"
                       miny="6134490.68376212" maxx="588538.116199611"
                       maxy="6157775.71193034"/>
          <Style>
            <Name>line</Name>
            <Title>Default Line</Title>
            <Abstract>A sample style that draws a line</Abstract>
            <LegendURL width="20" height="20">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                            xlink:type="simple"
xlink:href="http://localhost:8081/geoserver/ows?service=WMS&amp;request=GetLegendGraphic&
amp;format=image%2Fpng&amp;width=20&amp;height=20&amp;layer=seggeo%3Aavenidas"/>
            </LegendURL>
          </Style>
          <Style>
```

```
            <Name>avenidas</Name>
            <Title>Default Styler for simple road segments</Title>
            <Abstract>Light red line, 2px wide</Abstract>
            <LegendURL width="20" height="20">
            <Format>image/png</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                            xlink:type="simple"
xlink:href="http://localhost:8081/geoserver/ows?service=WMS&amp;request=GetLegendGraphic&
amp;format=image%2Fpng&amp;width=20&amp;height=20&amp;layer=seggeo%3Aavenidas&amp;style=a
venidas"/>
            </LegendURL>
          </Style>
        </Layer>
        ...
      </Capability>
    </WMS_Capabilities>]]>
  </eeip:GetCapabilitiesResult>
</eeip:GetCapabilitiesResponse>
```

Cuadro 5.6 - Respuesta SOAP de operación GetCapabilities

La respuesta SOAP está contenida dentro de etiquetas propias de EEIP que envuelven la el elemento <*WMS_Capabilities:GetCapabilities*>, que es la respuesta del IMS.

Dentro de este elemento se define principalmente el servicio (<*Service*>) al que corresponden las capacidades que provee el IMS, en este caso WMS, y el elemento <*Capability*> que contiene la lista de operaciones soportadas por el servicio, las excepciones y las capas disponibles.

Cada operación tiene definido un elemento con su nombre, ya sea GetCapabilties, GetMap o GetFeatureInfo. Cada uno de estos elementos contiene una lista de formatos que soporta (<*Format*>) como puede ser html o imagen con tipo de datos específico.

Para cada capa (<*Layer*>) se declara su nombre, los sistemas de referencia de coordenadas (<*CRS*>) soportados y la respectiva delimitación geográfica (<*BoundingBox*>) y los estilos (<*Style*>) asociados. Cada estilo está identificado por un nombre, un formato y una URL direccionando al archivo que es donde están definidos propiamente los estilos.

5.5 Resultados obtenidos

El caso de estudio presentado en este capítulo, permite ver el funcionamiento de los mecanismos desarrollados junto con los mecanismos obtenidos de trabajos anteriores. Está fuera del alcance del caso de estudio abarcar todos los escenarios posibles pero como resultado de componer estos mecanismos se pudo mostrar cómo funcionan en conjunto

para generar una plataforma que proporcione además de información geográfica enriquecida con información de negocio, la posibilidad de autenticar a los usuarios y crear control de acceso a los recursos.

En detalle lo que se pudo demostrar a través del caso de estudio fueron las siguientes funcionalidades:

- Autenticación de un usuario mediante un STS.
- Autorización de un usuario a los recursos geográficos mediante un PDP.
- Despliegue de información geográfica enriquecida por datos empresariales.

5.5.1 Funcionalidades no consideradas en el caso de estudio

Para el caso de estudio no fueron considerados escenarios en los cuáles los mecanismos funcionen de forma aislada. Uno de los objetivos del proyecto es que los mecanismos reutilizados de GeoEEIP tengan la capacidad de ser utilizados de forma independiente. Los escenarios de prueba planteados (ver sección 5.3) no cubren esta opción.

Queda por fuera del caso de estudio la evaluación de la plataforma SegGeoESB para las solicitudes SOAP. En la sección 5.4 se detallaron las pruebas que se realizaron para estos casos utilizando la herramienta SoapUI.

6 Conclusiones

En este capítulo se presentan las conclusiones del trabajo realizado y los posibles trabajos a futuro. En la sección 6.1 se describe la gestión del proyecto, presentando la metodología de trabajo y el análisis de estimación. En la sección 6.2 se presentan las dificultades encontradas a lo largo del proyecto. Las conclusiones son abordadas en la sección 6.3. Por último en la sección 6.4 se presentan los posibles trabajos a futuro.

6.1 Gestión del proyecto

En la presente sección se presentan las etapas que fueron parte del proyecto y la estimación mediante un diagrama de Gantt en la figura 6.1.

La primera etapa del proyecto fue realizar el estado del arte de las principales tecnologías involucradas en el proyecto. Luego se realizó un estudio de la plataforma del proyecto de grado GeoEEIP. En base a este estudio y la investigación de estándares en el diseño de arquitecturas SOA se analizó la arquitectura de la plataforma propuesta. Una vez que la arquitectura quedó definida, se realizó el diseño e implementación del módulo de autorización. Paralelamente se diseñó e implementó el módulo de autenticación. Luego se integró el mecanismo de autenticación con el STS. Como fue mencionado en la sección 4.1.1.1 el STS está basado en la implementación del proyecto de grado "*Integración GIS-PGE*". Los mecanismos existentes de GeoEEIP fueron reutilizados e integrados a los mecanismos de seguridad. El desarrollo del caso de estudio permitió el testing de la plataforma SegGeoESB. Por último se realizó el informe final del trabajo.

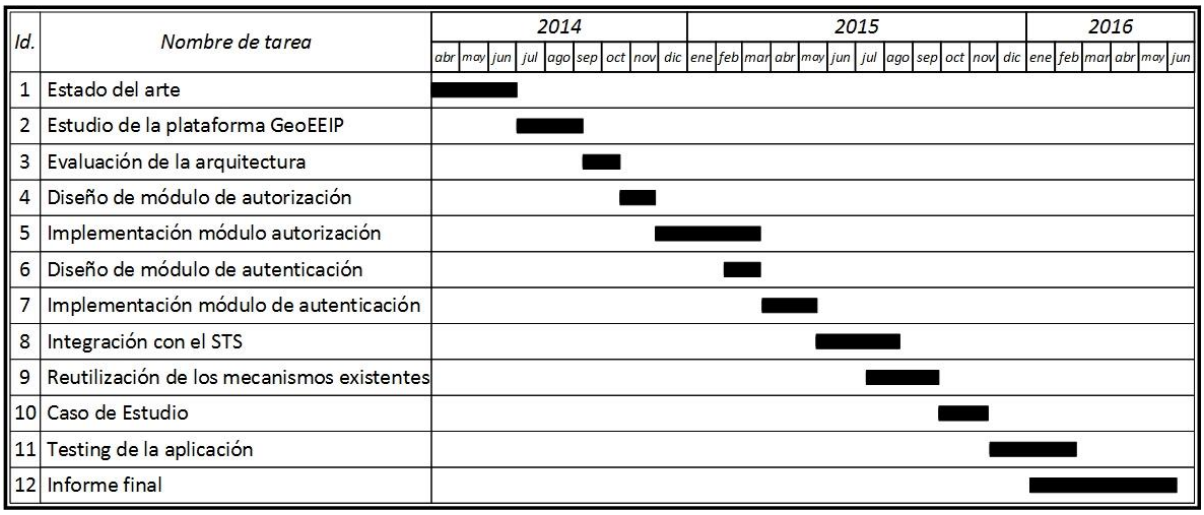

Diagrama 6.1 - Diagrama de Gantt

En el Apéndice A se detallan en profundidad todas las etapas que fueron parte de este trabajo y fueron introducidas previamente en esta sección.

6.2 Dificultades encontradas

Las dificultades que se encontraron durante el desarrollo del presente trabajo son las siguientes:

- Por un lado habíamos analizado la posibilidad de usar Picketlink como solución al manejo de autenticación y gestión de usuarios. Contemplando esta opción iniciamos un trabajo de investigación para determinar si este framework era lo suficientemente flexible como para ser integrado a nuestro proyecto. Luego de hacer pruebas de concepto para evaluar si era factible su uso, nos dimos cuenta de que la versión de JBoss AS (versión 6.1.0[54]) soportada por JBoss ESB no era compatible con ninguna de las versiones de Picketlink disponibles. Debido a esto se tomó la decisión de no utilizar Picketlink como parte de la solución para el mecanismo de autenticación.

- Como alternativa al uso de Picketlink como librería para implementar el IP se decidió analizar la implementación de un STS por parte del proyecto *"Integración GIS-PGE"* [19]. Más allá de que se logró integrar esta implementación con nuestra solución, hubieron ciertas dificultades para llevarlo a cabo. Por ejemplo se realizaron algunos ajustes y agregados para poder obtener una correcta validación de tokens. Como consecuencia se extendió la librería para soportar estos cambios.

- La reutilización y adaptación de tres prototipos realizados durante el desarrollo de proyectos de grado anteriores tuvo sus dificultades. Del trabajo GeoEEIP [21] se reutilizaron los mecanismos existentes y fueron integrados con los mecanismos de seguridad desarrollados en el presente proyecto. La implementación del servidor GeoXACML del trabajo "*Seguridad en Sistemas de Información Geográfica"* no estaba desarrollado para un ESB. Por esta razón se tuvo que adaptar el código del módulo PEP, lo que derivó en el desarrollo del mecanismo GeoPEP (ver 4.2.4). Como fue mencionado en el punto anterior la integración con el STS del trabajo *"Integración GIS-PGE"* fue compleja y tuvo que ser adaptada para que pudiera ser utilizable.

[54] https://developer.jboss.org/wiki/AS610FinalReleaseNotes

6.3 Contribuciones y Resumen

Se realizó un estudio detallado de los Web services geoespaciales (2.1.1.1) y las tecnologías de seguridad en autenticación y autorización (2.1.4) para arquitecturas SOA. En particular se evaluaron y realizaron pruebas de concepto con WS-Security (2.3.1), SAML (2.2.1.1) y GeoXACML (2.2.2.2).

Se desarrollaron los mecanismos de autenticación OWSAuthentication (4.1.1) y autorización *OWSAuthorization* (4.1.2). La autenticación de los clientes es delegada a un STS externo (4.1.1.1), que por intermedio del uso de tokens de seguridad SAML valida la identidad de los usuarios. En las solicitudes SOAP, la integridad y seguridad en el envío de las credenciales del cliente es suministrada por el protocolo WS-Security (ver el estudio realizado en la sección 2.3.1). El mecanismo *OWSAuthorization* aplica el control de acceso a la información geoespacial mediante el uso del estándar GeoXACML. Las solicitudes de acceso son enviadas al módulo PDP (ver 4.1.2.1) y la respuesta evaluada por el mecanismo GeoPEP.

La plataforma SegGeoESB fue diseñada siguiendo el patrón de diseño Composición de Servicios [29], que tiene como beneficios la reusabilidad, modificabilidad, extensibilidad, entre otras ventajas (ver 4.3). Este enfoque facilita la integración entre los mecanismos de integración ya existentes y los mecanismos de seguridad.

Los mecanismos de integración existentes (3.4.1) fueron reutilizados para que fueran compatibles con el objetivo de que se puedan componer con otros mecanismos. A su vez se mantuvo la posibilidad de que estos puedan ser consumidos de forma independiente, como estaban planteados en la plataforma GeoEEIP.

Se hizo el diseño de un mecanismo pensando cómo se podría agregar a futuro otro tipo de mecanismo que soportara o solucionara un requerimiento no funcional, no necesariamente vinculado a la seguridad. Con ese objetivo se diseñó el GeoCache (3.3.1.3), que permite mejorar los tiempos de respuesta al obtener los recursos geoespaciales, almacenando los resultados de solicitudes anteriores obtenidos directamente del IMS.

Como forma de evaluar y demostrar la factibilidad técnica y uso de la plataforma se desarrolló un caso de estudio. Los distintos escenarios de prueba (5.3) hicieron posible que fueran utilizados todos los mecanismos desarrollados, validando a su vez la estrategia de composición de los mismos.

Se constató que no existe un estándar de implementación de los Enterprise Integration Patterns[55] en los distintos productos ESB. Si bien son soportados por todos, cada uno los implementa a su manera. Eso implica que la implementación planteada deba ser modificada para que pueda ser utilizada en otro ESB. Sin embargo, el diseño fue pensado independiente del ESB elegido para el desarrollo de la plataforma. Esto permite que el diseño no sea alterado al cambiar de ESB.

Para finalizar se puede concluir que se pudo realizar un diseño y una implementación que cumple con los objetivos planteados y que se realizó un caso de estudio que abarca la mayoría de los mismos. Además se pudieron identificar un conjunto de mejoras las cuales se describen en la siguiente sección.

6.4 Trabajo a futuro

- Es recomendable que a futuro se realice una migración a otro producto ESB, porque JBoss ESB está discontinuado en su desarrollo. SwitchYard [57] sería el "más indicado" a nivel de impacto. SwitchYard es el sucesor de JBoss ESB y es el que en principio debería tener más similitudes. Para esto hay que evaluar la complejidad de realizar la migración dados los mecanismos implementados.

- Implementación de un mecanismo diseñado para caché de respuestas [25]. El tiempo de respuesta del IMS puede ser elevado, en especial cuando se utiliza WFS, por lo cual sería deseable tener un sistema de caché para que las consultas del cliente no dependieran tanto de la velocidad de respuesta del IMS. Una opción podría ser el uso de la librería JBoss Cache [26].

- Agregar soporte a las solicitudes WFS en SOAP. Si bien la transformación de SOAP a WFS no era un requisito del proyecto y la transformación de SOAP a WMS y viceversa es heredada del proyecto GeoEEIP, se considera que es recomendable implementar esta transformación para una completitud de la solución. La implementación de la transformación de WFS no es tan compleja como la desarrollada para WMS.

- Sería deseable que el flujo de invocación de los Web services de la plataforma, tanto para el caso de SOAP y REST, sea el mismo. Como fue explicado en la sección 3.5

[55] http://www.enterpriseintegrationpatterns.com/

los clientes REST deben seguir dos pasos. Primero deben enviar sus credenciales y el token SAML es enviado como respuesta. Luego para cada solicitud de acceso a un recurso se debe enviar ese token. En SOAP en cambio se deben enviar las credenciales en cada solicitud de acceso a un recurso. Es deseable que se realicen modificaciones para que los clientes SOAP tengan la misma interacción con SegGeoESB que los clientes REST.

Referencias

[1] B. Rienzi y L. Gonzalez, *Towards an ESB-Based Enterprise Integration Platform for Geospatial Web Services. 2013*

[2] T. Erl, *SOA Principles of Service Design*. Prentice Hall. 2007.

[3] En línea: J. Kress, http://www.oracle.com/technetwork/articles/soa/ind-soa-esb-1967705.html. Julio 2013

[4] R. Sosa, *Web Services Geográficos y Gobierno*. Pedeciba Informática.2011

[5] D. Chappell, *Enterprise service bus*. O'Reilly. Julio 2004

[6] T. Rademarkers, *Open Source ESBs In Action*. Manning. 2009

[7] S. Arambur y, P. Chaer, *Mecanismos de Seguridad en ESB*. FIng, UdelaR. 2012

[8] D. Booth y H. Haas, *Web Service Architecture*. W3C Working Group Note. Feb. 2004

[9] P. A. C. Ramarao Kanneganti, *SOA Security*. Manning Publications. Enero 2008.

[10] En línea: *Open Geospatial Consortium*. http://www.opengeospatial.org/. Marzo 2016

[11] L. Jiayuan, T. Heping y G. Zhu, *An Access Control Mechanism for Geospatial Information Services*. Information Science and Engineering (ICISE), 2009 1st International Conference on, vol., no., pp.1971,1974, 26-28 Dec. 2009

[12] M. Shilcher y A. Donaubauer, *OGC Specifications for Access to Distributed Geospatial Data*. Photogrammeric Week 05. Wichmann Verlag, Heidelberg, pp 217-227,2005

[13] En lìnea: J. Harrison y C. Reed, *Authentication IE*, http://www.opengeospatial.org/projects/initiatives/authie. Agosto 2015

[14] J. Rosenberg y D. Remy, *Securing Web Services with WS-Security*. May 12, 2004

[15] En línea: *OASIS, eXtensible Access Control Markup Language (XACML) Version 3.0*. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf. 22 Enero 2013

[16] A. Matheus, *OGC Geospatial eXtensible Access Control Markup Language (GeoXACML) 3.0 Core*. 11 Junio 2013

[17] A. Matheus, *Geospatial eXtensible Access Control Markup Language (GeoXACML) Version 1 Corrigendum*. 12 Mayo 2011

[18] En línea: *OASIS*, https://www.oasis-open.org/standards#samlv2.0. Marzo 2015

[19] L. Canales, M. Felix y A. Remiro, *Integración GIS-PGE*. FIng, UdelaR. Mayo 2015

[20] J. Pintos, R. Ordeix y M. Souto, *Seguridad en Sistemas de Información Geográfica*. FIng, UdelaR. Octubre 2011

[21] I. Assandri, M. Villero y N. Lasarte, *Plataforma de Integración de Información Geográfica sobre JBoss ESB*. FIng, UdelaR. Junio 2014

[22] En línea: *OpenSAML*. https://wiki.shibboleth.net/confluence/display/OpenSAML/Home. Marzo 2016

[23] En línea: http://techspiro.blogspot.com.uy/2015/08/xacml-to-fore.html. Febrero 2016

[24] En línea: J. Sermersheim. https://tools.ietf.org/rfc/rfc4511.txt. Marzo 2016

[25] En línea: *Cache mediation pattern specification: an overview* http://www.ibm.com/developerworks/library/ws-soa-cachemed/. Marzo 2016

[26] En línea: *JBossCache http://jbosscache.jboss.org/*. Marzo 2016

[27] En línea: *Implementing Single Sign-On Across Multiple Organizations* https://developer.salesforce.com/page/Implementing_Single_Sign-On_Across_Multiple_Organizations. Agosto 2015

[28] P. A. C. Ramarao Kanneganti, *SOA Security* .Manning Publications. Enero 2008

[29] En línea: A. Arsanjani, *Toward a pattern language for Service-Oriented Architecture and Integration*. http://www.ibm.com/developerworks/webservices/library/ws-soa-soi2. Junio 2015

[30] A. Whiteside y J. Greenwood y Ramarao Kanneganti, *OGC Web Services Common Standard*. Abril 2010

[31] J. de la Beaujardiere, *OpenGIS Web Map Server Implementation Specification*. Marzo 2006

[32] En lìnea: *Relationship to the World Wide Web and REST Architectures.* https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest. Marzo 2016

[33] R. Fielding y R. Taylor, *Principled design of the modern Web architecture*. 2000

[34] L. Richardson y S. Ruby, *RESTful web services*. Farnham. O'Reilly. 2007

[35] En línea: *What Are RESTful Web Services?* http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html. Marzo 2016

[36] En línea: *The Secure Sockets Layer (SSL) Protocol Version 3.0*, https://tools.ietf.org/html/rfc6101. Marzo 2016

[37] En línea: *AGESIC*, http://www.agesic.gub.uy/. Marzo 2016

[38] M. Steffen y E. Penna, *Orquestación de Servicios en la Plataforma de Interoperabilidad de Gobierno Electrónico*. FIng, UdelaR. Agosto 2013

[39] S. Aramburu, P. Chaer y S. Grattarola, *Mecanismos de Seguridad en Enterprises Service Bus*. FIng, UdelaR. Diciembre 2012

[40] B. Argenta, L. Galusso y F. Maldonado, *Sistema de Gestión de Identidades*. FIng, UdelaR. Julio 2014

[41] En línea: About SAML-Based Single Sign-On, https://support.cloudpassage.com/entries/25297643-About-SAML-Based-Single-Sign-On . Marzo 2016

[42] E. Gamma, R. Helm, R. Johnson, J. Vlissides, y G. Booch, Design Patterns: Elements of Reusable Object-Oriented Software, 1 edition. Reading, Mass: Addison-Wesley Professional, 1994

[43] En línea: Geography Markup Language, http://www.opengeospatial.org/standards/gml. Abril 2016

[44] En línea: Sun's XACML Implementation, http://sunxacml.sourceforge.net. Abril 2016

[45] En línea: Role-based Access Control, http://csrc.nist.gov/groups/SNS/rbac/ . Abril 2016

[46] En línea: Access Control List, http://msdn.microsoft.com/en-us/library/windows/desktop/aa374872(v=vs.85).aspx. Abril 2016

[47] En línea: Smooks, http://www.smooks.org/mediawiki/index.php?title=Main_Page. Abril 2016

[48] En línea: JBoss ESB, http://jbossesb.jboss.org/. Abril 2016

[49] En línea: SoapUI, http://www.soapui.org/. Abril 2016

[50] En línea: GeoServer, http://geoserver.org/. Abril 2016

[51] En línea: Arquitectura Web services, http://slideplayer.es/slide/1057007/. Abril 2016

[52] En línea: Picketlink, http://picketlink.org/. Abril 2016

[53] En línea: About SAML-Based Single Sign-On, https://support.cloudpassage.com/entries/25297643-About-SAML-Based-Single-Sign-On. Abril 2016

[54] En línea: Apache Tomcat, http://tomcat.apache.org/. Mayo 2016

[55] En línea: Apache Software Foundation, http://www.apache.org/. Mayo 2016

[56] En línea: JBoss AS, http://wildfly.org/. Mayo 2016

[57] En línea: SwitchYard, http://switchyard.jboss.org/. Mayo 2016

[58] En línea: http://www.cgisecurity.com/ws.html. Mayo 2016

[59] En línea: WS-Security, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss. Mayo 2016

[60] En línea: https://msdn.microsoft.com/en-us/library/ms951257. Mayo 2016

[61] En línea: Web Services Addressing, https://www.w3.org/Submission/ws-addressing/. Mayo 2016

[62] En línea: WS-Addressing, http://www.ibm.com/support/knowledgecenter/es/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac64300_.htm . Mayo 2016

[63] En línea: WS-Trust, http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html. Mayo 2016

[64] En línea: WS-Federation, http://docs.oasis-open.org/wsfed/federation/v1.2/ws-federation.html. Mayo 2016

Glosario

GeoXACML	Geospatial eXtensible Access Control Markup Language
GML	Geographic Markup Language
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
OGC	Open Geospatial Consortium
OWS	OpenGIS Web Services
SAML	Security Assertion Markup Language
SOA	Service Oriented Architecture
SSO	Single Sign-On
STS	Security Token Service
UML	Unified Modeling Language
WFS	Web Feature Service
WMS	Web Map Service
XACML	eXtensible Access Control Markup Language

Figuras

Cuadros

Diagramas

Apéndice A - Etapas del proyecto

De modo de organizar el trabajo a realizar se definieron etapas, cada una con sus particularidades descritas a continuación.

- **Estado del arte**

  En esta etapa cada integrante investigó una tecnología generando un reporte de la información recabada. Se llevaron a cabo reuniones de grupo semanales de seguimiento e intercambio de conocimientos. Al final de esta etapa se realizó una presentación oral a los tutores del proyecto para compartir los conocimientos adquiridos. Con esta información y la investigación de trabajos relacionados con el presente proyecto se generó el documento del Estado del Arte incluido en este informe en la sección 2.

- **Estudio de la plataforma GeoEEIP**

  Se realizó una investigación de la plataforma GeoEEIP, con el fin de analizar las soluciones con respecto al enriquecimiento de mensajes e incompatibilidad de formatos, dando lugar a la reutilización de los mecanismos ya existentes en esta plataforma.

- **Evaluación de la arquitectura**

  En esta etapa se realizó un análisis de la plataforma aplicando los conocimientos adquiridos en la etapa anterior y en el estado del arte. Se evaluaron distintos enfoques y se diseñó una propuesta de la plataforma a implementar.

- **Diseño del módulo de autorización**

  Se diseñó el módulo de autorización considerando el uso del servidor PDP y el diseño planteado por el proyecto descrito en la sección 2.4.2.

- **Implementación del módulo de autorización**

  Una vez generado el diseño se comienza con la etapa de implementación de la arquitectura de autorización definida. Se implementó el mecanismo de autorización, incluido el mecanismo GeoPEP. La integración con el servidor PDP también fue parte de esta etapa.

- **Diseño de módulo de autenticación**

  En esta etapa se realizó el diseño de la solución para autenticar a los clientes de la plataforma. El Identity Provider se pensó desde un primer momento como un componente externo al ESB dado que no era un requerimiento la implementación del mismo.

- **Implementación módulo de autenticación**

  En esta etapa se comenzó con la implementación del mecanismo de autenticación, teniendo en cuenta que los mensajes SOAP utilizan una transformación a WMS con el fin de ser coherentes con la solución planteada para los mensajes REST.

- **Integración con el STS**

  El STS fue reutilizado y adaptado de un proyecto de grado anterior, como fue explicado previamente. Se tuvo que realizar varios cambios (listados en la sección 4.1.1.1) para que se pudiera integrar con la plataforma SegGeoESB. Cabe aclarar que no era un objetivo del proyecto la implementación de un STS.

- **Reutilización de los mecanismos de GeoEEIP**

  Se reutilizaron y adaptaron los mecanismos de integración del trabajo GeoEEIP. Como parte de la adaptación, estos fueron integrados con los mecanismos de seguridad desarrollados en el presente trabajo.

- **Caso de Estudio**

  Se decidió reutilizar el caso de estudio del proyecto *"Plataforma de Integración de Información Geográfica sobre JBoss ESB"*. Esta es una aplicación para la elección de un local de cobro del BPS, como caso de prueba para esta plataforma. De esta manera se verificó el funcionamiento de las principales operaciones de la plataforma.

- **Testing de la aplicación**

  Se verificaron todos los mecanismos complejos disponibles en la plataforma. Esto fue posible mediante la definición de distintos escenarios de prueba ejecutados por el Caso de Estudio y utilizando la herramienta SoapUI.

- **Informe final**

  En esta etapa se finalizó la documentación del proyecto, la cual comenzó en paralelo con la etapa de implementación. Esta documentación comprende el documento final y anexos.

Seguridad en Plataforma de Integración de Servicios Geográficos

Anexos

Autores

Eugenia Díaz

Juan Gonzalez

Leonardo Jorcin

Tutores

Bruno Rienzi

1 Configuración de SegGeoESB

1.1 OWSAuthentication

1.1.1 Servicios REST

```
<service category="GeoEntryPoint" invmScope="GLOBAL"name="RESTAutenthicationEntryPoint">
   <listeners>
      <http-gateway name="RESTAuthHTTP_GATEWAY" urlPattern="RESTAuthentication"
                    payloadAs="STRING">
         <property name="synchronousTimeout" value="30000" />
         <property name="allowedPorts" value="8443" />
      </http-gateway>
   </listeners>
   <actions mep="RequestResponse">
      <action name="GeoEntryPoint"
              class="uy.edu.fing.seggeoesb.mecanismos.basicos.GeoEntryPointRESTAuth"/>
      <action name="ConfigAggregator"
              class="uy.edu.fing.seggeoesb.mecanismos.basicos.ConfigAggregator">
         <property name="enableGeoEnrichment" value="false" />
         <property name="enableAuthorization" value="false" />
         <property name="sts_token_generator_uri"
                   value="http://localhost:8080/STSService/STSServerServlet" />
         <property name="sts_token_validation_uri"
                   value="http://localhost:8080/STSService/STSValidateTokenServlet" />
      </action>
      <action class="org.jboss.soa.esb.actions.SyncServiceInvoker" name="routeAuth">
         <property name="service-category" value="Authentication" />
         <property name="service-name" value="STSAuthentication" />
      </action>
      <action name="RestoreMessage"
              class="uy.edu.fing.seggeoesb.mecanismos.basicos.RestoreRESTMessage">
         <property name="propertyRestored" value="token" />
      </action>
   </actions>
</service>

<service category="GeoEntryPoint" invmScope="GLOBAL" name="RESTEntryPoint">
   <listeners>
      <http-gateway name="RESTHTTP_GATEWAY" urlPattern="REST" payloadAs="STRING">
         <property name="synchronousTimeout" value="30000" />
         <property name="allowedPorts" value="8080" />
```

```
      </http-gateway>
   </listeners>
   <actions mep="OneWay">
      <action name="GeoEntryPoint"
              class="uy.edu.fing.seggeoesb.mecanismos.basicos.GeoEntryPointREST" />
      <action name="ConfigAggregator"
              class="uy.edu.fing.seggeoesb.mecanismos.basicos.ConfigAggregator">
         <property name="enableGeoEnrichment" value="true" />
         <property name="enableAuthorization" value="true" />
         <property name="sts_token_validation_uri"
                   value="http://localhost:8080/STSService/STSValidateTokenServlet" />
      </action>
      <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeValidacionSTS">
         <property name="destinations">
            <route-to service-category="Authentication" service-name="TokenValidation"/>
         </property>
      </action>
   </actions>
</service>
```

1.1.2 Servicios SOAP

```
<service category="GeoEntryPoint" invmScope="GLOBAL" name="SOAPEntryPoint">
         <listeners>
            <http-gateway name="WMS_SOAP_WRAPPER_HTTP_GATEWAY" urlPattern="SOAP"
                          payloadAs="STRING">
               <property name="synchronousTimeout" value="900000" />
               <property name="allowedPorts" value="8080" />
            </http-gateway>
         </listeners>
         <actions mep="OneWay">
            <action class="org.jboss.soa.esb.actions.SyncServiceInvoker"
                    name="routeWrapper">
               <property name="service-category" value="Wrapper" />
               <property name="service-name" value="SOAP2WrapperService" />
            </action>
            <action name="ConfigAggregator"
                    class="uy.edu.fing.seggeoesb.mecanismos.basicos.ConfigAggregator">
               <property name="enableGeoEnrichment" value="false" />
               <property name="enableAuthorization" value="false" />
               <property name="sts_token_generator_uri"
                     value="http://localhost:8080/STSService/STSServerServlet" />
               <property name="sts_token_validation_uri"
```

```
                    value="http://localhost:8080/STSService/STSValidateTokenServlet" />
            </action>
            <action class="org.jboss.soa.esb.actions.SyncServiceInvoker"
                    name="routeAuth">
               <property name="service-category" value="Authentication" />
               <property name="service-name" value="STSAuthentication" />
            </action>
            <action class="org.jboss.soa.esb.actions.StaticWiretap"
                    name="routeValidacionSTS">
               <property name="destinations">
                  <route-to service-category="Authentication"
                            service-name="TokenValidation" />
               </property>
            </action>
         </actions>
      </service>
```

1.1.3 Servicios de STS

```
<service category="Authentication" name="STSAuthentication" invmScope="GLOBAL">
         <actions mep="RequestResponse
            <action name="Autenticacion con STS"
       class="uy.edu.fing.seggeoesb.mecanismos.basicos.autenticacion.STSTokenHandler"/>
         </actions>
      </service>

      <service category="Authentication" invmScope="GLOBAL" name="TokenValidation">
         <actions mep="OneWay">
            <action name="DecodeToken"
        class="uy.edu.fing.seggeoesb.mecanismos.basicos.autenticacion.DecodeToken" />
            <action name="Validacion"
    class="uy.edu.fing.seggeoesb.mecanismos.basicos.autenticacion.STSTokenValidator" />
            <action class="org.jboss.soa.esb.actions.ContentBasedRouter"
                    name="Authentication_Router">
               <property name="cbrAlias" value="Regex" />
               <property name="ruleSet" value="/regex-authentication-rules.properties"/>
               <property name="ruleLanguage" />
               <property name="ruleReload" value="true" />
               <property name="destinations">
                  <route-to destination-name="authorization"
                    service-category="Authorization" service-name="AuthorizationEntry" />
                  <route-to destination-name="blank_response"
                    service-category="Response" service-name="BlankResponse" />
```

```
                <route-to destination-name="geoserver"
                  service-category="Response" service-name="GeoServerResponse" />
            </property>
            <property name="object-paths">
               <object-path esb="properties.validatedToken" />
            </property>
         </action>
      </actions>
   </service>
```

1.2 OWSAuthorization

```
<service category="Authorization" description="Entrada a Authorization"
         invmScope="GLOBAL" name="AuthorizationEntry">
   <!-- filtrar mensaje si es WMS/WFS -->
  <actions mep="OneWay">
     <action class="org.jboss.soa.esb.actions.ContentBasedRouter" name="WMS_WFS_Router">
        <property name="cbrAlias" value="Regex" />
        <property name="ruleSet" value="/regex-georeqtype-rules.properties" />
        <property name="ruleLanguage" />
        <property name="ruleReload" value="true" />
        <property name="destinations">
           <route-to destination-name="wmsMessage" service-category="GetTargets"
                  service-name="GetWMSTargets" />
           <route-to destination-name="wfsMessage" service-category="GetTargets"
                  service-name="GetWFSTargets" />
        </property>
        <property name="object-paths">
           <object-path esb="properties.geo_req_type" />
        </property>
     </action>
  </actions>
</service>

<service category="GetTargets" description="Obtener los parametros WMS"
         invmScope="GLOBAL" name="GetWMSTargets">
  <actions mep="OneWay">
     <action class="uy.edu.fing.seggeoesb.mecanismos.basicos.WMSGeoXACMLRequest"
            name="WMSGeoXACMLRequest" />
     <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeAction">
        <property name="destinations">
           <route-to service-category="PEP" service-name="GeoPEP" />
        </property>
```

```
    </action>
  </actions>
</service>

<service category="GetTargets" description="Obtener los parametros WFS"
         invmScope="GLOBAL" name="GetWFSTargets">
  <actions mep="OneWay">
    <action class="uy.edu.fing.seggeoesb.mecanismos.basicos.WFSGeoXACMLRequest"
            name="WFSGeoXACMLRequest" />
    <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeAction">
       <property name="destinations">
          <route-to service-category="PEP" service-name="GeoPEP" />
       </property>
    </action>
  </actions>
</service>
```

1.2.1 Servicios GeoPEP

```
<service category="PEP" name="GeoPEP" description="Geo PEP" invmScope="GLOBAL">
  <actions mep="OneWay">
    <action class="uy.edu.fing.seggeoesb.mecanismos.basicos.PDPRequester"
            name="request-mapper" />
    <action class="org.jboss.soa.esb.actions.soap.SOAPClient"
            name="ClienteEvaluatePDP">
       <property name="wsdl" value="http://localhost:8080/PDP/services/WSPDP?wsdl" />
       <property name="SOAPAction" value="evaluate" />
       <property name="responseAsOgnlMap" value="true" />
    </action>
    <action class="uy.edu.fing.seggeoesb.mecanismos.basicos.PDPResponse"
            name="response-mapper" />
    <action class="org.jboss.soa.esb.actions.StaticWiretap" name="route">
       <property name="destinations">
          <route-to service-category="Response" service-name="ResponseEntry" />
       </property>
    </action>
  </actions>
</service>
```

1.3 Archivo de configuración jboss-esb.xml

```
<?xml version="1.0"?>
<jbossesb parameterReloadSecs="5"
xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb-1.3.1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb-1.3.1.xsd
http://anonsvn.jboss.org/repos/labs/labs/jbossesb/trunk/product/etc/schemas/xml/jbossesb-1.3.1.xsd">
   <providers>
      <jms-provider connection-factory="ConnectionFactory"name="SOAPWrapperMQ">
         <jms-bus busid="route_to_geoFeatureTypeAdapter">
            <jms-message-filter dest-name="queue/queue_route_to_getFeatureTypeAdapter"
                     dest-type="QUEUE"                    />
         </jms-bus>
         <jms-bus busid="route_to_geoFeatureEnricher">
            <jms-message-filter dest-name="queue/queue_route_to_getFeatureEnricher"
                     dest-type="QUEUE" />
         </jms-bus>
         <jms-bus busid="route_to_specificGeoFeatureEnricher">
            <jms-message-filter
                     dest-name="queue/queue_route_to_getSpecificFeatureEnricher"
                     dest-type="QUEUE" />
         </jms-bus>
         <jms-bus busid="wfs_AggregatorListenQueue">
            <jms-message-filter
                     dest-name="queue/queue_route_to_getAggregatorListenQueue_WFS"
                     dest-type="QUEUE" />
         </jms-bus>
         <jms-bus busid="route_to_geoFeatureInfoEnricher">
            <jms-message-filter dest-name="queue/queue_route_to_getFeatureInfoEnricher"
                     dest-type="QUEUE" />
         </jms-bus>
         <jms-bus busid="route_to_specificGeoFeatureInfoEnricher">
            <jms-message-filter
                 dest-name="queue/queue_route_to_getSpecificFeatureInfoEnricher_WMSTest"
                dest-type="QUEUE" />
         </jms-bus>
         <jms-bus busid="wms_AggregatorListenQueue">
            <jms-message-filter
                     dest-name="queue/queue_route_to_getAggregatorListenQueue_WMSTest"
                     dest-type="QUEUE" />
         </jms-bus>
```

```
        <jms-bus busid="soapAuthentication">
           <jms-message-filter dest-type="QUEUE dest-name="queue/seggeoesb_soap_esb"
                    selector="serviceName='SoapService'" />
        </jms-bus>
     </jms-provider>
  </providers>

  <services>

  <!-- ############################ WMS SOAPWRAPPER ########################### -->
  <service category="GeoEntryPoint" invmScope="GLOBAL" name="SOAP2WMSEntryPoint">
     <listeners>
        <http-gateway name="SOAP_WMS_WRAPPER_ONLY_HTTP_GATEWAY"
                      urlPattern="SOAP2WMSWrapper" payloadAs="STRING">
           <property name="synchronousTimeout" value="900000" />
           <property name="allowedPorts" value="8088" />
        </http-gateway>
     </listeners>
     <actions mep="OneWay">
        <action class="org.jboss.soa.esb.actions.SyncServiceInvoker"
                name="routeWrapper">
           <property name="service-category" value="Wrapper" />
           <property name="service-name" value="SOAP2WrapperService" />
        </action>
        <action name="ConfigAggregator"
                class="uy.edu.fing.seggeoesb.mecanismos.basicos.ConfigAggregator">
           <property name="enableGeoEnrichment" value="false" />
           <property name="enableAuthorization" value="false" />
        </action>
        <action name="SkipSecurityMechanisms"
              class="uy.edu.fing.seggeoesb.mecanismos.basicos.SkipSecurityMechanisms"/>
        <action class="org.jboss.soa.esb.actions.StaticWiretap"
                name="routeValidacionSTS">
           <property name="destinations">
              <route-to service-category="Response" service-name="ResponseEntry" />
           </property>
        </action>
     </actions>
  </service>
  <service category="Wrapper" invmScope="GLOBAL" name="SOAP2WMSWrapperService">
     <actions faultXsd="/lib/fault.xsd" inXsd="/lib/wms_schemas.xsd"
              mep="RequestResponse" outXsd="/lib/wms_schemas.xsd" webservice="false">
        <action name="contract-publisher"
            class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.ContractPublisher">
           <property name="wsdl" value="classpath:///lib/wms.wsdl" />
```

```
        </action>
        <action name="GeoEntryPoint"
                class="uy.edu.fing.seggeoesb.mecanismos.basicos.GeoEntryPointSOAP" />
        <action name="SOAP2WMSTranslator"
         class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.SOAP2WMSTranslator" />
    </actions>
</service>
<!-- ############################ FIN WMS SOAPWRAPPER ########################## -->

<!-- ################################## ENRICHER ################################## -->
<service category="Enricher" name="WMSEnricherEntryPoint" invmScope="GLOBAL"
          description="Entrada a Enricher de WMS">
    <listeners>
      <http-gateway name="WMSTest_HTTP_GATEWAY" urlPattern="WMSEnricher"
                    payloadAs="STRING">
         <property name="synchronousTimeout" value="30000" />
         <property name="allowedPorts" value="8080" />
      </http-gateway>
    </listeners>
    <actions mep="RequestResponse">
        <action name="GeoEntryPoint"
                class="uy.edu.fing.seggeoesb.mecanismos.basicos.GeoEntryPointREST" />
        <action name="GeoProxy"
                class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoProxy" >
           <property name="imsName" value="http://localhost:8081/geoserver/" />
        </action>
        <action class="org.jboss.soa.esb.actions.SyncServiceInvoker" name="routeAuth">
           <property name="service-category" value="Enricher" />
           <property name="service-name" value="EntryPoint" />
        </action>
        <action name="RestoreMessage"
                class="uy.edu.fing.seggeoesb.mecanismos.basicos.RestoreRESTMessage">
           <property name="propertyRestored" value="token" />
        </action>
    </actions>
</service>
<service category="Enricher" name="WFSEnricherEntryPoint" invmScope="GLOBAL"
          description="Entrada a Enricher de WFS" >
    <listeners>
        <http-gateway name="WFSTest_HTTP_GATEWAY" urlPattern="WFSEnricher"
                      payloadAs="STRING">
           <property name="synchronousTimeout" value="30000" />
           <property name="allowedPorts" value="8080" />
        </http-gateway>
    </listeners>
```

```
      <actions mep="RequestResponse">
         <action name="GeoEntryPoint"
                 class="uy.edu.fing.seggeoesb.mecanismos.basicos.GeoEntryPointREST" />
         <action name="GeoProxy"
                 class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoProxy" >
            <property name="imsName" value="http://localhost:8081/geoserver/" />
         </action>
         <action class="org.jboss.soa.esb.actions.SyncServiceInvoker" name="routeAuth">
           <property name="service-category" value="Enricher" />
           <property name="service-name" value="EntryPoint" />
         </action>
         <action name="RestoreMessage"
                 class="uy.edu.fing.seggeoesb.mecanismos.basicos.RestoreRESTMessage">
            <property name="propertyRestored" value="token" />
         </action>
      </actions>
   </service>
   <service category="Enricher" invmScope="GLOBAL" name="EntryPoint">
      <actions mep="OneWay">
         <action class="org.jboss.soa.esb.actions.ContentBasedRouter"
                 name="WMS_WFS_Router">
            <property name="cbrAlias" value="Regex" />
            <property name="ruleSet" value="/regex-georeqtype-rules.properties" />
            <property name="ruleLanguage" />
            <property name="ruleReload" value="true" />
            <property name="destinations">
               <route-to destination-name="wmsMessage" service-category="Enricher"
                         service-name="WMSEnricher" />
               <route-to destination-name="wfsMessage" service-category="Enricher"
                         service-name="WFSEnricher" />
            </property>
            <property name="object-paths">
               <object-path esb="properties.geo_req_type" />
            </property>
         </action>
      </actions>
   </service>
   <service category="Enricher" description="" invmScope="GLOBAL" name="WMSEnricher">
      <actions mep="OneWay">
         <action class="org.jboss.soa.esb.actions.ContentBasedRouter" name="GeoRouter">
            <property name="cbrAlias" value="Regex" />
            <property name="ruleSet" value="/regex-georouter-rules.properties" />
            <property name="ruleReload" value="true" />
            <property name="object-paths">
               <object-path esb="properties.query_string" />
```

```
            </property>
            <property name="destinations">
               <route-to destination-name="geoIdentity" service-category="GeoIdentity"
                         service-name="GeoIdentity" />
               <route-to destination-name="geoCapabilitiesAdapter"
                         service-category="GeoCapabilitiesAdapter"
                         service-name="GetCapabilitiesAdapter"/>
               <route-to destination-name="geoFeatureInfoEnricher"
                         service-category="GeoFeatureInfoEnricher"
                         service-name="GetFeatureInfoEnricher"/>
            </property>
         </action>
      </actions>
   </service>
   <service category="Enricher" description="" invmScope="GLOBAL" name="WFSEnricher">
      <actions mep="OneWay">
         <action class="org.jboss.soa.esb.actions.ContentBasedRouter" name="GeoRouter">
            <property name="cbrAlias" value="Regex" />
            <property name="ruleSet" value="/regex-georouter-rules.properties" />
            <property name="ruleReload" value="true" />
            <property name="object-paths">
               <object-path esb="properties.query_string" />
            </property>
            <property name="destinations">
               <route-to destination-name="geoIdentity" service-category="GeoIdentity"
                         service-name="GeoIdentity" />
               <route-to destination-name="geoCapabilitiesAdapter"
                         service-category="GeoCapabilitiesAdapter"
                         service-name="GetCapabilitiesAdapter" />
               <route-to destination-name="geoFeatureEnricher"
                         service-category="GeoFeatureEnricher"
                         service-name="GetFeatureEnricher" />
               <route-to destination-name="geoFeatureTypeAdapter"
                         service-category="GeoFeatureTypeAdapter"
                         service-name="GetFeatureTypeAdapter" />
            </property>
         </action>
      </actions>
   </service>
   <service category="GeoCapabilitiesAdapter" description="route to geo capabilities"
            invmScope="GLOBAL" name="GetCapabilitiesAdapter">
      <actions mep="RequestResponse">
         <action name="GeoCapabilitiesAdapter"
      class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoCapabilitiesAdapter" />
         <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeAction">
```

```
            <property name="destinations">
               <route-to service-category="Response" service-name="ESBResponse" />
            </property>
         </action>
      </actions>
   </service>
   <service category="GeoIdentity" name="GeoIdentity" invmScope="GLOBAL">
      <actions mep="OneWay">
         <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeAction">
            <property name="destinations">
               <route-to service-category="Response" service-name="ESBResponse" />
            </property>
         </action>
      </actions>
   </service>

   <!-- WMS -->
   <service category="GeoFeatureInfoEnricher" name="GetFeatureInfoEnricher">
      <listeners>
         <jms-listener busidref="route_to_geoFeatureInfoEnricher"
                       name="route_to_getFeatureInfoEnricher" />
      </listeners>
      <actions mep="OneWay">
         <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeAction">
            <property name="destinations">
               <route-to service-category="EnricherSource"
                         service-name="WMSEnricherSource" />
            </property>
         </action>
      </actions>
   </service>
   <service category="EnricherSource" name="WMSEnricherSource">
      <listeners>
         <jms-listener busidref="route_to_specificGeoFeatureInfoEnricher"
                       name="route_to_specificGeoFeatureInfoEnricher" />
      </listeners>
      <actions mep="OneWay">
         <action name="GeoFeatureInfoEnricher"
        class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoFeatureInfoEnricher">
            <property name="sourceName" value="WMSTest" />
            <property name="wsdl"
               value="http://localhost:8080/BPSEnterpriseDataA/BPSLocalesPagoWS?wsdl" />
            <property name="SOAPAction" value="infoLocal" />
            <property name="ws_query_params_xpath"
                      value="ws_query_params_xpath_sourceA.properties" />
```

```
            <property name="get-payload-location" value="ws_parameters_in_message" />
         </action>
         <action class="org.jboss.soa.esb.actions.StaticRouter"name="routeToAggregator">
            <property name="destinations">
               <route-to destination-name="aggregatorService"
                service-category="Aggregation" service-name="WMSEnrichmentAggregrator"/>
            </property>
         </action>
      </actions>
   </service>
   <service category="Aggregation" name="WMSEnrichmentAggregrator">
      <listeners>
        <jms-listener busidref="wms_AggregatorListenQueue"name="AggregatorListenQueue"/>
      </listeners>
      <actions mep="OneWay">
         <action class="org.jboss.soa.esb.actions.Aggregator" name="Aggregator">
            <property name="timeoutInMillis" value="600000" />
         </action>
         <action name="GeoFeatureInfoAggregator"
       class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoFeatureInfoAgregator">
            <property name="sourceName" value="WMSTest" />
         </action>
         <action class="org.jboss.soa.esb.smooks.SmooksAction" name="WMSTest_transform">
            <property name="smooksConfig" value="smooks-res_WMSTest.xml" />
            <property name="get-payload-location" value="ims_response" />
            <property name="set-payload-location" value="ims_response" />
            <property name="mappedContextObjects" value="java.lang.String" />
         </action>
         <action class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.RestoreMessage"
                 name="RestoreMessage" />
         <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeAction">
            <property name="destinations">
               <route-to service-category="Response" service-name="ESBResponse" />
            </property>
         </action>
      </actions>
   </service>

   <!-- WFS -->
   <service category="GeoFeatureEnricher" name="GetFeatureEnricher">
      <listeners>
         <jms-listener busidref="route_to_geoFeatureEnricher"
                       name="route_to_geoFeatureEnricher" />
      </listeners>
      <actions mep="OneWay">
```

```
        <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeAction">
           <property name="destinations">
              <route-to service-category="EnricherSource"
                        service-name="WFSEnricherSource" />
           </property>
        </action>
     </actions>
  </service>
  <service category="EnricherSource" name="WFSEnricherSource">
     <listeners>
        <jms-listener busidref="route_to_specificGeoFeatureEnricher"
                      name="route_to_specificGeoFeatureEnricher" />
     </listeners>
     <actions mep="OneWay">
        <action name="GeoFeatureInfoEnricher"
       class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoFeatureInfoEnricher">
           <property name="sourceName" value="WFSTest" />
           <property name="wsdl"
              value="http://localhost:8080/BPSEnterpriseDataA/EspaciosLibresWS?wsdl" />
           <property name="SOAPAction" value="infoEspaciosLibres" />
           <property name="ws_query_params_xpath"
                     value="ws_query_params_xpath_InfoEspacioLibre.properties" />
           <property name="get-payload-location" value="ws_parameters_in_message" />
        </action>
        <action class="org.jboss.soa.esb.actions.StaticRouter"name="routeToAggregator">
           <property name="destinations">
              <route-to destination-name="aggregatorService"
                service-category="Aggregation" service-name="WFSEnrichmentAggregrator"/>
           </property>
        </action>
     </actions>
  </service>
  <service category="Aggregation" name="WFSEnrichmentAggregrator">
     <listeners>
       <jms-listener busidref="wfs_AggregatorListenQueue"name="AggregatorListenQueue"/>
     </listeners>
     <actions mep="OneWay">
        <action class="org.jboss.soa.esb.actions.Aggregator" name="Aggregator">
           <property name="timeoutInMillis" value="600000" />
        </action>
        <action name="WFSTest_assemble"
      class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoFeatureInfoAgregator">
           <property name="sourceName" value="WFSTest" />
        </action>
        <action class="org.jboss.soa.esb.smooks.SmooksAction" name="WFSTest_transform">
```

```
            <property name="smooksConfig" value="smooks-res_WFSTest.xml" />
            <property name="get-payload-location" value="ims_response" />
            <property name="set-payload-location" value="ims_response" />
            <property name="mappedContextObjects" value="java.lang.String" />
         </action>
         <action class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.RestoreMessage"
                 name="RestoreMessage" />
         <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeAction">
            <property name="destinations">
               <route-to service-category="Response" service-name="ESBResponse" />
            </property>
         </action>
      </actions>
   </service>

   <service category="GeoFeatureTypeAdapter" name="GetFeatureTypeAdapter">
      <listeners>
         <jms-listener busidref="route_to_geoFeatureTypeAdapter"
                       name="route_to_geoFeatureTypeAdapter" />
      </listeners>
      <actions mep="OneWay">
         <action name="GeoFeatureTypeAdapter"
         class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoFeatureTypeAdapter">
            <property name="featureTypeAdapterSchema" value="InfoEspacioLibre.xsd" />
            <property name="featureTypeAdapterXpaths"
                      value="InfoEspacioLibre.properties" />
         </action>
         <action class="org.jboss.soa.esb.actions.StaticWiretap" name="routeAction">
            <property name="destinations">
               <route-to service-category="Response" service-name="ESBResponse" />
            </property>
         </action>
      </actions>
   </service>
   <!-- ############################## FIN ENRICHER ################################## -->

   <!-- ############################## RESPONSE ################################## -->
   <service name="ResponseEntry" description="" category="Response" invmScope="GLOBAL">
      <actions mep="OneWay">
         <action class="org.jboss.soa.esb.actions.ContentBasedRouter" name="GeoRouter">
            <property name="cbrAlias" value="Regex" />
            <property name="ruleSet" value="/regex-authorization-rules.properties" />
            <property name="ruleLanguage" />
            <property name="ruleReload" value="true" />
            <property name="destinations">
```

```
            <route-to destination-name="geoserver_response"
                  service-category="Response" service-name="GeoServerResponse" />
            <route-to destination-name="blank_response"
                  service-category="Response" service-name="BlankResponse" />
         </property>
         <property name="object-paths">
            <object-path esb="properties.is_authorized" />
         </property>
      </action>
   </actions>
</service>

<service name="BlankResponse" description="" category="Response" invmScope="GLOBAL">
   <actions mep="OneWay">
      <action class="uy.edu.fing.seggeoesb.mecanismos.basicos.GeoBlankResponse"
            name="Blank" />
      <action class="org.jboss.soa.esb.actions.StaticWiretap" name="esbresponse">
         <property name="destinations">
            <route-to service-category="Response" service-name="ESBResponse" />
         </property>
      </action>
   </actions>
</service>

<service name="GeoServerResponse" description="" category="Response" invmScope="GLOBAL">
   <actions mep="OneWay">
      <action class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.GeoProxy"
            name="GeoProxy">
         <property name="imsName" value="http://localhost:8081/geoserver/" />
      </action>
      <action class="org.jboss.soa.esb.actions.ContentBasedRouter" name="Router">
         <property name="cbrAlias" value="Regex" />
         <property name="ruleSet" value="/regex-enrichment-rules.properties" />
         <property name="ruleLanguage" />
         <property name="ruleReload" value="true" />
         <property name="destinations">
            <route-to destination-name="enricher" service-category="Enricher"
                  service-name="EntryPoint" />
            <route-to destination-name="response" service-category="Response"
                  service-name="ESBResponse" />
         </property>
         <property name="object-paths">
            <object-path esb="properties.enableGeoEnrichment" />
         </property>
      </action>
```

```
   </actions>
</service>

<service name="ESBResponse" description="Encargado de enviar la respuesta al cliente"
         category="Response" invmScope="GLOBAL">
   <actions mep="OneWay">
      <action class="org.jboss.soa.esb.actions.ContentBasedRouter" name="WSTypeRouter">
         <property name="cbrAlias" value="Regex" />
         <property name="ruleSet" value="/regex-wstype-rules.properties" />
         <property name="ruleLanguage" />
         <property name="ruleReload" value="true" />
         <property name="destinations">
            <route-to destination-name="soapws" service-category="Response"
                      service-name="SOAPResponse" />
            <route-to destination-name="restws" service-category="Response"
                      service-name="RestResponse" />
         </property>
         <property name="object-paths">
             <object-path esb="properties.ws_type" />
         </property>
      </action>
   </actions>
</service>

<service name="RestResponse" description="Respuesta a pedido REST" category="Response"
invmScope="GLOBAL">
                     <actions mep="RequestResponse">
                            <action
class="uy.edu.fing.seggeoesb.mecanismos.basicos.RestoreRESTMessage"
name="RestoreRESTMessage" />
                     </actions>
               </service>

   <service name="SOAPResponse" description="" category="Response" invmScope="GLOBAL">
      <actions mep="OneWay">
         <action class="org.jboss.soa.esb.actions.ContentBasedRouter"
                 name="ContentTypeRouter">
            <property name="cbrAlias" value="Regex" />
            <property name="ruleSet" value="/regex-contenttype-rules.properties" />
            <property name="ruleLanguage" />
            <property name="ruleReload" value="true" />
            <property name="destinations">
               <route-to destination-name="binary" service-category="EEIP"
                                                   service-name="WMS_2_SOAP_Binary" />
               <route-to destination-name="text" service-category="EEIP"
```

```
                                                    service-name="WMS_2_SOAP_Text" />
            <route-to destination-name="application" service-category="EEIP"
                                                    service-name="WMS_2_SOAP_Text" />
         </property>
         <property name="object-paths">
            <object-path esb="properties.content_type" />
         </property>
      </action>
   </actions>
</service>

<service category="EEIP" description="WMS_2_SOAP Binary" invmScope="GLOBAL"
      name="WMS_2_SOAP_Binary">
   <actions faultXsd="/lib/fault.xsd" inXsd="/lib/wms_schemas.xsd"
         mep="RequestResponse" outXsd="/lib/wms_schemas.xsd" webservice="true">
      <action name="WMS2SOAPBinaryTranslator"
  class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.WMS2SOAPBinaryTranslator"/>
   </actions>
</service>

<service category="EEIP" invmScope="GLOBAL" name="WMS_2_SOAP_Text">
   <actions faultXsd="/lib/fault.xsd" inXsd="/lib/wms_schemas.xsd"
         mep="RequestResponse" outXsd="/lib/wms_schemas.xsd" webservice="true">
      <action name="WMS2SOAPTextTranslator"
   class="uy.edu.fing.seggeoesb.mecanismos.basicos.geoeeip.WMS2SOAPTextTranslator"/>
   </actions>
</service>
<!-- ############################## FIN RESPONSE ################################## -->

 </services>
</jbossesb>
```

1.4 WMS wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="http://www.fing.edu.uy/eeip/wms"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:wms="http://www.fing.edu.uy/eeip/wms"
           xmlns:wms_defs="http://www.fing.edu.uy/eeip/wms_definitions"
           xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
           xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
```

```
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
          xmlns:ows="http://www.opengis.net/ows"
          xmlns:eeip="http://www.fing.edu.uy/eeip">
<import namespace="http://www.fing.edu.uy/eeip/wms_definitions"
        location="wms_definitions.wsdl"/>
<binding name="WmsServiceSoapBinding" type="wms_defs:WmsPortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetMap">
    <soap:operation soapAction="http://www.fing.edu.uy/eeip/GetMap" style="document"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="GetFeatureInfo">
    <soap:operation soapAction="http://www.fing.edu.uy/eeip/GetFeatureInfo"
                    style="document"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="GetCapabilities">
    <soap:operation soapAction="http://www.fing.edu.uy/eeip/GetCapabilities"
                    style="document"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
<binding name="WmsServiceHTTPPostBinding" type="wms_defs:WmsPortType">
  <http:binding verb="POST"/>
  <operation name="GetMap">
    <http:operation location="/GetMap"/>
    <input>
      <mime:content type="application/x-www-form-urlencoded"/>
```

```
      </input>
      <output>
        <mime:mimeXml part="body"/>
      </output>
    </operation>
    <operation name="GetFeatureInfo">
      <http:operation location="/GetFeatureInfo"/>
      <input>
        <mime:content type="application/x-www-form-urlencoded"/>
      </input>
      <output>
        <mime:mimeXml part="body"/>
      </output>
    </operation>
    <operation name="GetCapabilities">
      <http:operation location="/GetCapabilities"/>
      <input>
        <mime:content type="application/x-www-form-urlencoded"/>
      </input>
      <output>
        <mime:mimeXml part="body"/>
      </output>
    </operation>
  </binding>
  <binding name="WmsServiceHTTPGetBinding" type="wms_defs:WmsPortTypeHTTPGet">
    <http:binding verb="GET"/>
    <operation name="GetMap">
      <http:operation location="/GetMap"/>
      <input>
        <http:urlEncoded/>
      </input>
      <output>
        <mime:mimeXml part="Body"/>
      </output>
    </operation>
    <operation name="GetCapabilities">
      <http:operation location="/GetCapabilities"/>
      <input>
        <http:urlEncoded/>
      </input>
      <output>
        <mime:mimeXml part="Body"/>
      </output>
    </operation>
  </binding>
```

```
  <service name="WmsService">
    <!--SOAP binding-->
    <port name="WmsServiceSoap" binding="wms:WmsServiceSoapBinding">
      <soap:address location="http://localhost/sias/cgi-bin/siscgi.exe/wms/soap"/>
    </port>
    <!--HTTP Post binding-->
    <port name="WmsServiceHTTPPost" binding="wms:WmsServiceHTTPPostBinding">
      <http:address location="http://localhost/sias/cgi-bin/siscgi.exe/wms/post"/>
    </port>
    <!--HTTP Get binding-->
    <port name="WmsServiceHTTPGet" binding="wms:WmsServiceHTTPGetBinding">
      <http:address location="http://localhost/sias/cgi-bin/siscgi.exe/wms"/>
    </port>
  </service>
</definitions>
```

1.4.1 wms_definitions.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="http://www.fing.edu.uy/eeip/wms_definitions"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wms_defs="http://www.fing.edu.uy/eeip/wms_definitions"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:HTTP="http://schemas.xmlsoap.org/wsdl/HTTP/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ows="http://www.opengis.net/ows" xmlns:eeip="http://www.fing.edu.uy/eeip">
  <import namespace="http://www.fing.edu.uy/eeip" location="wms_schemas.xsd"/>
  <message name="GetMapIn">
    <part name="parameters" element="eeip:GetMap"/>
  </message>
  <message name="GetMapOut">
    <part name="body" element="eeip:GetMapResponse"/>
  </message>
  <message name="GetFeatureInfoIn">
    <part name="parameters" element="eeip:GetFeatureInfo"/>
  </message>
  <message name="GetFeatureInfoOut">
    <part name="body" element="eeip:GetFeatureInfoResponse"/>
  </message>
  <message name="GetCapabilitiesIn">
    <part name="parameters" element="eeip:GetCapabilities"/>
```

```
</message>
<message name="GetCapabilitiesOut">
  <part name="body" element="eeip:GetCapabilitiesResponse"/>
</message>
<message name="GetMapHTTPGetIn">
  <part name="request" type="xs:string"/>
  <part name="version" type="eeip:WmsVersionType"/>
  <part name="service" type="eeip:ServiceType"/>
  <part name="exceptions" type="eeip:ExceptionType"/>
  <part name="format" type="xs:string"/>
  <part name="srs" type="xs:string"/>
  <part name="bbox" type="xs:string"/>
  <part name="transparent" type="xs:boolean"/>
  <part name="height" type="xs:integer"/>
  <part name="width" type="xs:integer"/>
  <part name="bgcolor" type="eeip:ColorType"/>
  <part name="layers" type="xs:string"/>
  <part name="styles" type="xs:string"/>
  <part name="time" type="xs:string"/>
  <part name="elevation" type="xs:integer"/>
</message>
<message name="GetMapHTTPGetOut">
  <part name="Body" element="eeip:base64Binary"/>
</message>
<message name="GetCapabilitiesHTTPGetIn">
  <part name="request" type="xs:string"/>
  <part name="version" type="eeip:WmsVersionType"/>
  <part name="service" type="eeip:ServiceType"/>
  <part name="exceptions" type="eeip:ExceptionType"/>
  <part name="Section" type="eeip:CapabilitiesSectionType"/>
  <part name="updatesequence" type="xs:string"/>
</message>
<message name="GetCapabilitiesHTTPGetOut">
  <part name="Body" type="xs:string"/>
</message>
<portType name="WmsPortType">
  <operation name="GetMap">
    <input message="wms_defs:GetMapIn"/>
    <output message="wms_defs:GetMapOut"/>
  </operation>
  <operation name="GetFeatureInfo">
    <input message="wms_defs:GetFeatureInfoIn"/>
    <output message="wms_defs:GetFeatureInfoOut"/>
  </operation>
  <operation name="GetCapabilities">
```

```
    <input message="wms_defs:GetCapabilitiesIn"/>
    <output message="wms_defs:GetCapabilitiesOut"/>
  </operation>
</portType>
<portType name="WmsPortTypeHTTPGet">
  <operation name="GetMap">
    <input message="wms_defs:GetMapHTTPGetIn"/>
    <output message="wms_defs:GetMapHTTPGetOut"/>
  </operation>
  <operation name="GetCapabilities">
    <input message="wms_defs:GetCapabilitiesHTTPGetIn"/>
    <output message="wms_defs:GetCapabilitiesHTTPGetOut"/>
  </operation>
</portType>
</definitions>
```

2 Configuración de BPSEnterpriseData

2.1 BPSLocalesPagoWS.wsdl

```
<wsdl:definitions name="BPSLocalesPagoWS" targetNamespace="http://bps.gub.uy">
   <wsdl:types>
      <xs:schema elementFormDefault="unqualified" targetNamespace="http://bps.gub.uy"
                     version="1.0">
         <xs:element name="InfoLocalPago" type="tns:infoLocalPago" />
         <xs:element name="InfoMedioTransporte" type="tns:infoMedioTransporte" />
         <xs:element name="infoLocal" type="tns:infoLocal" />
         <xs:element name="infoLocalResponse" type="tns:infoLocalResponse" />
         <xs:element name="infoTransporte" type="tns:infoTransporte" />
         <xs:element name="infoTransporteResponse" type="tns:infoTransporteResponse" />
         <xs:complexType name="infoLocal">
            <xs:sequence>
               <xs:element minOccurs="0" name="local_id" type="xs:string" />
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="infoLocalResponse">
            <xs:sequence>
              <xs:element minOccurs="0" name="InfoLocalPago" type="tns:infoLocalPago" />
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="infoLocalPago">
            <xs:sequence>
              <xs:element minOccurs="0" name="horario" type="tns:horario" />
            </xs:sequence>
            <xs:attribute name="id" type="xs:string" />
            <xs:attribute name="servicios">
               <xs:simpleType>
                  <xs:list itemType="xs:string" />
               </xs:simpleType>
            </xs:attribute>
         </xs:complexType>
         <xs:complexType name="horario">
            <xs:sequence />
            <xs:attribute name="dia" type="xs:string" />
            <xs:attribute name="hora" type="xs:string" />
         </xs:complexType>
         <xs:complexType name="infoTransporte">
            <xs:sequence>
               <xs:element minOccurs="0" name="transporte_id" type="xs:string" />
```

```
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="infoTransporteResponse">
            <xs:sequence>
               <xs:element minOccurs="0" name="InfoTransporte"
                           type="tns:infoMedioTransporte" />
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="infoMedioTransporte">
            <xs:sequence>
               <xs:element name="estacionamiento" type="xs:boolean" />
               <xs:element minOccurs="0" name="id" type="xs:string" />
               <xs:element name="paraDeTaxi" type="xs:boolean" />
               <xs:element maxOccurs="unbounded" minOccurs="0" name="paradas"
                           nillable="true" type="tns:parada" />
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="parada">
            <xs:sequence>
               <xs:element name="numero" type="xs:int" />
               <xs:element minOccurs="0" name="omnibus" type="xs:string" />
               <xs:element minOccurs="0" name="parada" type="xs:string" />
            </xs:sequence>
         </xs:complexType>
      </xs:schema>
   </wsdl:types>
   <wsdl:message name="infoTransporte">
      <wsdl:part element="tns:infoTransporte" name="parameters" />
   </wsdl:message>
   <wsdl:message name="infoLocal">
      <wsdl:part element="tns:infoLocal" name="parameters" />
   </wsdl:message>
   <wsdl:message name="infoTransporteResponse">
      <wsdl:part element="tns:infoTransporteResponse" name="parameters" />
   </wsdl:message>
   <wsdl:message name="infoLocalResponse">
      <wsdl:part element="tns:infoLocalResponse" name="parameters" />
   </wsdl:message>
   <wsdl:portType name="BPSLocalesPagoWS">
      <wsdl:operation name="infoLocal">
         <wsdl:input message="tns:infoLocal" name="infoLocal" />
         <wsdl:output message="tns:infoLocalResponse" name="infoLocalResponse" />
      </wsdl:operation>
      <wsdl:operation name="infoTransporte">
         <wsdl:input message="tns:infoTransporte" name="infoTransporte" />
```

```
        <wsdl:output message="tns:infoTransporteResponse" name="infoTransporteResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="BPSLocalesPagoWSSoapBinding" type="tns:BPSLocalesPagoWS">
      <soap:binding style="document"
                    transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="infoLocal">
         <soap:operation soapAction="" style="document" />
         <wsdl:input name="infoLocal">
         <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output name="infoLocalResponse">
            <soap:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="infoTransporte">
         <soap:operation soapAction="" style="document" />
         <wsdl:input name="infoTransporte">
         <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output name="infoTransporteResponse">
            <soap:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="BPSLocalesPagoWS">
      <wsdl:port binding="tns:BPSLocalesPagoWSSoapBinding" name="BPSLocalesPagoWSPort">
         <soap:address
                 location="http://localhost:8080/BPSEnterpriseDataA/BPSLocalesPagoWS" />
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

2.2 EspaciosLibresWS.wsdl

```
<wsdl:definitions name="EspaciosLibres" targetNamespace="http://www.montevideo.gub.uy">
   <wsdl:types>
      <xs:schema elementFormDefault="unqualified" version="1.0"
                 targetNamespace="http://www.montevideo.gub.uy" >
         <xs:element name="InfoMedioTransporte" type="tns:infoMedioTransporte" />
         <xs:element name="infoEspaciosLibres" type="tns:infoEspaciosLibres" />
         <xs:element name="infoEspaciosLibresResponse"
```

```
                type="tns:infoEspaciosLibresResponse" />
      <xs:complexType name="infoEspaciosLibres">
         <xs:sequence />
      </xs:complexType>
      <xs:complexType name="infoEspaciosLibresResponse">
         <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
                        name="InfoEspacioLibre" type="tns:infoEspacioLibre" />
         </xs:sequence>
      </xs:complexType>
      <xs:complexType name="infoEspacioLibre">
         <xs:sequence>
            <xs:element name="bancos" type="xs:boolean" />
            <xs:element name="banio" type="xs:boolean" />
            <xs:element name="basurero" type="xs:boolean" />
            <xs:element name="habPerros" type="xs:boolean" />
            <xs:element name="hamaca" type="xs:boolean" />
            <xs:element name="id" type="xs:int" />
         </xs:sequence>
      </xs:complexType>
      <xs:complexType name="infoMedioTransporte">
         <xs:sequence>
            <xs:element name="estacionamiento" type="xs:boolean" />
            <xs:element minOccurs="0" name="id" type="xs:string" />
            <xs:element name="paraDeTaxi" type="xs:boolean" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="paradas"
                        nillable="true" type="tns:parada" />
         </xs:sequence>
      </xs:complexType>
      <xs:complexType name="parada">
         <xs:sequence>
            <xs:element name="numero" type="xs:int" />
            <xs:element minOccurs="0" name="omnibus" type="xs:string" />
            <xs:element minOccurs="0" name="parada" type="xs:string" />
         </xs:sequence>
      </xs:complexType>
      <xs:complexType name="horario">
         <xs:sequence />
         <xs:attribute name="dia" type="xs:string" />
         <xs:attribute name="hora" type="xs:string" />
      </xs:complexType>
   </xs:schema>
</wsdl:types>
<wsdl:message name="infoEspaciosLibresResponse">
   <wsdl:part element="tns:infoEspaciosLibresResponse" name="parameters" />
```

```
   </wsdl:message>
   <wsdl:message name="infoEspaciosLibres">
      <wsdl:part element="tns:infoEspaciosLibres" name="parameters" />
   </wsdl:message>
   <wsdl:portType name="EspaciosLibresWS">
      <wsdl:operation name="infoEspaciosLibres">
         <wsdl:input message="tns:infoEspaciosLibres" name="infoEspaciosLibres" />
         <wsdl:output message="tns:infoEspaciosLibresResponse"
                      name="infoEspaciosLibresResponse" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="EspaciosLibresSoapBinding" type="tns:EspaciosLibresWS">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="infoEspaciosLibres">
         <soap:operation soapAction="" style="document" />
         <wsdl:input name="infoEspaciosLibres">
            <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output name="infoEspaciosLibresResponse">
            <soap:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="EspaciosLibres">
      <wsdl:port binding="tns:EspaciosLibresSoapBinding" name="EspaciosLibresWSPort">
         <soap:address
                 location="http://localhost:8080/BPSEnterpriseDataA/EspaciosLibresWS"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

3 Configuración del PDP

3.1 WSPDP.wsdl

```
<definitions name="WSPDPService" targetNamespace="http://pdp.com/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://pdp.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <types>
  <xs:schema targetNamespace="http://pdp.com/" version="1.0" xmlns:tns="http://pdp.com/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="ExcepcionWSPDP" type="tns:ExcepcionWSPDP"/>
   <xs:element name="evaluate" type="tns:evaluate"/>
   <xs:element name="evaluateResponse" type="tns:evaluateResponse"/>
   <xs:element name="refreshPolicy" type="tns:refreshPolicy"/>
   <xs:element name="refreshPolicyResponse" type="tns:refreshPolicyResponse"/>
   <xs:complexType name="evaluate">
    <xs:sequence>
     <xs:element minOccurs="0" name="request" type="xs:string"/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name="evaluateResponse">
    <xs:sequence>
     <xs:element minOccurs="0" name="return" type="xs:string"/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name="ExcepcionWSPDP">
    <xs:sequence>
     <xs:element minOccurs="0" name="message" type="xs:string"/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name="refreshPolicy">
    <xs:sequence/>
   </xs:complexType>
   <xs:complexType name="refreshPolicyResponse">
    <xs:sequence/>
   </xs:complexType>
  </xs:schema>
 </types>
 <message name="ExcepcionWSPDP">
  <part element="tns:ExcepcionWSPDP" name="ExcepcionWSPDP"/>
 </message>
 <message name="WSPDP_refreshPolicy">
  <part element="tns:refreshPolicy" name="refreshPolicy"/>
```

```
</message>
<message name="WSPDP_evaluate">
 <part element="tns:evaluate" name="evaluate"/>
</message>
<message name="WSPDP_evaluateResponse">
 <part element="tns:evaluateResponse" name="evaluateResponse"/>
</message>
<message name="WSPDP_refreshPolicyResponse">
 <part element="tns:refreshPolicyResponse" name="refreshPolicyResponse"/>
</message>
<portType name="WSPDP">
 <operation name="evaluate" parameterOrder="evaluate">
  <input message="tns:WSPDP_evaluate"/>
  <output message="tns:WSPDP_evaluateResponse"/>
  <fault message="tns:ExcepcionWSPDP" name="ExcepcionWSPDP"/>
 </operation>
 <operation name="refreshPolicy" parameterOrder="refreshPolicy">
  <input message="tns:WSPDP_refreshPolicy"/>
  <output message="tns:WSPDP_refreshPolicyResponse"/>
 </operation>
</portType>
<binding name="WSPDPBinding" type="tns:WSPDP">
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="evaluate">
  <soap:operation soapAction=""/>
  <input>
   <soap:body use="literal"/>
  </input>
  <output>
   <soap:body use="literal"/>
  </output>
  <fault name="ExcepcionWSPDP">
   <soap:fault name="ExcepcionWSPDP" use="literal"/>
  </fault>
 </operation>
 <operation name="refreshPolicy">
  <soap:operation soapAction=""/>
  <input>
   <soap:body use="literal"/>
  </input>
  <output>
   <soap:body use="literal"/>
  </output>
 </operation>
</binding>
```

```
 <service name="WSPDPService">
  <port binding="tns:WSPDPBinding" name="WSPDPPort">
   <soap:address location="http://127.0.0.1:8080/PDP/services/WSPDP"/>
  </port>
 </service>
</definitions>
```

3.2 Políticas

Se presentan las 3 políticas que fueron desarrolladas para el Caso de Estudio.

3.2.1 policy-admin.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" PolicyId="1507"
RuleCombiningAlgId="urn:oasis:name:tc:xacml:1.0:rule-combining-algorithm:deny-overides">
   <Description>Policy for admin user</Description>
   <Target>
     <Subjects>
       <Subject>
         <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
              admin</AttributeValue>
           <SubjectAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
         </SubjectMatch>
       </Subject>
     </Subjects>
   </Target>
   <Rule RuleId="1" Effect="Permit">
     <Target>
       <Actions>
         <Action>
           <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                GetCapabilities</AttributeValue>
             <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
           </ActionMatch>
         </Action>
         <Action>
```

```
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
               GetMap</AttributeValue>
            <ActionAttributeDesignator
               AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
               GetFeatureInfo</AttributeValue>
            <ActionAttributeDesignator
               AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
  </Target>
  <Condition>
      <Apply FunctionId="urn:ogc:def:function:geoxacml:1.0:geometry-Intersects">
        <ResourceAttributeDesignator AttributeId="bbox" DataType="gml:geometry"/>
        <AttributeValue DataType="gml:geometry">
          <gml:geometry xmlns:gml="http://www.opengis.net/gml">
            <gml:Polygon>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:coordinates cs="," decimal="." ts=" ">
                     286681.8446823484,5892788.716326414
                     286681.8446823484,6505879.6017696615
                     873206.6183040687,6505879.6017696615
                     873206.6183040687,5892788.716326414
                     286681.8446823484,5892788.716326414
                  </gml:coordinates>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:geometry>
        </AttributeValue>
      </Apply>
  </Condition>
</Rule>
<Rule RuleId="2" Effect="Permit">
  <Target>
     <Actions>
```

```
          <Action>
            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                 GetFeature</AttributeValue>
              <ActionAttributeDesignator
                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
     </Target>
   </Rule>
</Policy>
```

3.2.2 policy-serv_por_zona.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" PolicyId="1507"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
">
   <Description>Politica para un usuario, limitando el acceso por zona</Description>
   <Target>
     <Subjects>
       <Subject>
         <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
              usuario_zona</AttributeValue>
           <SubjectAttributeDesignator
              DataType="http://www.w3.org/2001/XMLSchema#string"
              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" />
         </SubjectMatch>
       </Subject>
     </Subjects>
     <Resources>
       <Resource>
         <ResourceMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
              .*ejes.*</AttributeValue>
           <ResourceAttributeDesignator AttributeId="layers"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
         </ResourceMatch>
       </Resource>
```

```
      <Resource>
        <ResourceMatch
             MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
             .*libres.*</AttributeValue>
          <ResourceAttributeDesignator AttributeId="layers"
             DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
      <Resource>
        <ResourceMatch
             MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
             .*manzanas.*</AttributeValue>
          <ResourceAttributeDesignator AttributeId="layers"
             DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Rule RuleId="1" Effect="Permit">
     <Target>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
               GetCapabilities</AttributeValue>
            <ActionAttributeDesignator
               AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
               GetMap</AttributeValue>
            <ActionAttributeDesignator
               AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
               GetFeatureInfo</AttributeValue>
```

```
            <ActionAttributeDesignator
               AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition>
      <Apply FunctionId="urn:ogc:def:function:geoxacml:1.0:geometry-Intersects">
        <ResourceAttributeDesignator AttributeId="bbox" DataType="gml:geometry"/>
        <AttributeValue DataType="gml:geometry">
          <gml:geometry xmlns:gml="http://www.opengis.net/gml">
            <gml:Polygon>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:coordinates cs="," decimal="." ts=" ">
                     286681.8446823484,5892788.716326414
                     286681.8446823484,6505879.6017696615
                     873206.6183040687,6505879.6017696615
                     873206.6183040687,5892788.716326414
                     286681.8446823484,5892788.716326414</gml:coordinates>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:geometry>
        </AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId="2" Effect="Permit">
    <Target>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
               GetFeature</AttributeValue>
            <ActionAttributeDesignator
               AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
</Policy>
```

3.2.3 policy-serv_comerciales.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" PolicyId="1507"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
   <Description>Politica para un usuario, limitando el acceso por zona</Description>
     <Target>
       <Subjects>
         <Subject>
           <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
             <AttributeValue  DataType="http://www.w3.org/2001/XMLSchema#string">
                usuario_zona</AttributeValue>
             <SubjectAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
           </SubjectMatch>
         </Subject>
       </Subjects>
       <Resources>
         <Resource>
           <ResourceMatch
                MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                .*comerciales.*</AttributeValue>
             <ResourceAttributeDesignator AttributeId="layers"
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
           </ResourceMatch>
         </Resource>
       </Resources>
     </Target>
     <Rule RuleId="1" Effect="Permit">
       <Target>
         <Actions>
           <Action>
             <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                  GetCapabilities</AttributeValue>
               <ActionAttributeDesignator
                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
             </ActionMatch>
```

```
        </Action>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
             GetMap</AttributeValue>
          <ActionAttributeDesignator
             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
             DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ActionMatch>
      </Action>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
             GetFeatureInfo</AttributeValue>
          <ActionAttributeDesignator
             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
             DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Condition>
    <Apply FunctionId="urn:ogc:def:function:geoxacml:1.0:geometry-Intersects">
      <ResourceAttributeDesignator AttributeId="bbox" DataType="gml:geometry"/>
        <AttributeValue DataType="gml:geometry">
          <gml:geometry xmlns:gml="http://www.opengis.net/gml">
            <gml:Polygon>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:coordinates cs="," decimal="." ts=" ">
                     576619.543581578531303,6134539.461004209704697
                     577692.092633236548863,6135405.983684362843633
                     577758.748224017559551,6136060.420393848791718
                     578322.290946075110696,6136672.439909202046692
                     576619.543581578531303,6137405.651407793164253
                     575268.252968472661451,6136969.360268135555089
                     575292.491365120280534,6136163.433579601347446
                     575843.914888854022138,6136096.777988820336759
                     576619.543581578531303,6134539.461004209704697
                  </gml:coordinates>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:geometry>
        </AttributeValue>
```

```
        </Apply>
      </Condition>
    </Rule>
    <Rule RuleId="2" Effect="Permit">
      <Target>
        <Actions>
          <Action>
            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue  DataType="http://www.w3.org/2001/XMLSchema#string">
                 GetFeature</AttributeValue>
              <ActionAttributeDesignator
                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
```

4 STS

4.1 Ejemplo de Token SAML

Ejemplo de token de seguridad SAML versión 2.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <RequestSecurityTokenResponse>
      <TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
      </TokenType>
      <Lifetime>
        <wsu:Created>2016-04-28T00:16:28.494Z</wsu:Created>
        <wsu:Expires>2016-04-28T01:16:28.494Z</wsu:Expires>
      </Lifetime>
      <RequestedSecurityToken>
        <saml2:Assertion IssueInstant="2016-04-28T03:16:26.052Z" Version="2.0">
          <saml2:Issuer>STS-SegGeoESB</saml2:Issuer>
          <ds:Signature>
            <ds:SignedInfo>
              <ds:CanonicalizationMethod
                       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n" />
              <ds:SignatureMethod
                       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
              <ds:Reference URI="">
                <ds:Transforms>
                  <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#envloped-
                                           signature" />
                  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                    <ec:InclusiveNamespaces PrefixList="xs" />
                  </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>7uxD4FaElwA+/DbQYBKvv+nvEVg=</ds:DigestValue>
              </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>{firma}</ds:SignatureValue>
            <ds:KeyInfo>
              <ds:X509Data>
                <ds:X509Certificate>{certificado}</ds:X509Certificate>
              </ds:X509Data>
              <ds:KeyValue>
                <ds:RSAKeyValue>
                  <ds:Modulus>{clave}</ds:Modulus>
                  <ds:Exponent>AQAB</ds:Exponent>
                </ds:RSAKeyValue>
              </ds:KeyValue>
            </ds:KeyInfo>
          </ds:Signature>
          <saml2:Subject xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
            <saml2:NameID xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"/>
          </saml2:Subject>
          <saml2:Conditions NotBefore="2016-04-28T03:01:26.052Z"
                   NotOnOrAfter="2016-04-28T03:31:26.052Z">
            <saml2:AudienceRestriction>
              <saml2:Audience>http://SegGeoESB/sts</saml2:Audience>
            </saml2:AudienceRestriction>
          </saml2:Conditions>
          <saml2:AttributeStatement>
            <saml2:Attribute Name="User">
```

```
            <saml2:AttributeValue xsi:type="xs:string">user</saml2:AttributeValue>
          </saml2:Attribute>
        </saml2:AttributeStatement>
      </saml2:Assertion>
    </RequestedSecurityToken>
  </RequestSecurityTokenResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```