|  |
| --- |
|  |
| Integración GIS-PGE |
| Informe de Proyecto de Grado  Facultad de Ingeniería de la Universidad de la República Oriental del Uruguay  **Abril – 2015** |

|  |
| --- |
| Integrantes:  Luciana Canales  Maximiliano Felix  Alejandro Remiro  Tutora: Raquel Sosa. |

**Resumen**

El siguiente proyecto tiene como objetivo desarrollar la solución que se propone en la tesis de maestría de Raquel Sosa, Agosto 2011 [7]. Dicha solución pretende resolver la problemática que se plantea en la integración de las plataformas de gobierno electrónico y los servicios GIS tomando como referencia la arquitectura de Uruguay.

**Contenido:**

[1 Introducción 7](#_Toc412832154)

[1.1 Motivación y Contexto 7](#_Toc412832155)

[1.2 Objetivos 7](#_Toc412832156)

[1.3 Resultados esperados 8](#_Toc412832157)

[1.4 Resultados alcanzados 8](#_Toc412832158)

[1.5 Gestión del proyecto 8](#_Toc412832159)

[1.6 Organización del documento 8](#_Toc412832160)

[2 Marco de trabajo. 9](#_Toc412832161)

[2.1 Web Services Geográficos 9](#_Toc412832162)

[2.1.1 WMS 9](#_Toc412832163)

[2.1.2 WFS 9](#_Toc412832164)

[2.2 Estándares SOAP 10](#_Toc412832165)

[2.2.1 SOAP 10](#_Toc412832166)

[2.2.2 REST 10](#_Toc412832167)

[2.2.3 WS-SECURITY 11](#_Toc412832168)

[2.2.4 WS-ADDRESSING 12](#_Toc412832169)

[2.2.5 MTOM 13](#_Toc412832170)

[2.3 Tecnologías y productos estudiados. 13](#_Toc412832171)

[2.3.1 ESB 14](#_Toc412832172)

[2.3.2 Servidores de mapas 15](#_Toc412832173)

[2.4 Plataforma de eGob de Uruguay 17](#_Toc412832174)

[2.4.1 Plataforma de Interoperabilidad 17](#_Toc412832175)

[3 Análisis y arquitectura. 23](#_Toc412832176)

[3.1 Escenarios propuestos en la tesis 23](#_Toc412832177)

[3.1.1 Público general accediendo a información geográfica 23](#_Toc412832178)

[3.1.2 Público especializado consultando información 23](#_Toc412832179)

[3.1.3 Instituciones colaborando en la creación de información geográfica 23](#_Toc412832180)

[3.1.4 Instituciones colaborando en trámites 23](#_Toc412832181)

[3.1.5 Público generando información geográfica 23](#_Toc412832182)

[3.1.6 Resumen análisis de escenarios 24](#_Toc412832183)

[3.2 Solución propuesta en la tesis 24](#_Toc412832184)

[3.3 Casos de uso 25](#_Toc412832185)

[3.3.1 Actores 25](#_Toc412832186)

[3.3.2 Público general consultando información. 25](#_Toc412832187)

[3.3.3 Instituciones colaborando en trámites. 25](#_Toc412832188)

[3.4 Problemas técnicos encontrados. 25](#_Toc412832189)

[3.5 Decisiones de arquitectura 25](#_Toc412832190)

[3.5.1 Mapeo de las direcciones del servicio 26](#_Toc412832191)

[3.5.2 Opciones de despliegue 26](#_Toc412832192)

[4 Diseño e implementación 29](#_Toc412832193)

[4.1 Arquitectura 29](#_Toc412832194)

[4.1.1 Diagramas de actividad 29](#_Toc412832195)

[4.2 Implementación, diagrama de componentes, productos utilizados 30](#_Toc412832196)

[4.2.2 Configuración de los ctps y pge 32](#_Toc412832197)

[5 Caso de Estudio 36](#_Toc412832198)

[6 Conclusiones y trabajo a futuro 36](#_Toc412832199)

[7 Referencias 37](#_Toc412832200)

# Introducción

## Motivación y Contexto

Con el paso del tiempo los avances en la tecnología han logrado que las mismas sean implantadas prácticamente en todos los ámbitos de la sociedad y en particular en las gestiones del gobierno. Es por esto que surge la necesidad de crear lo que se denomina gobierno electrónico.

Varias son las definiciones que se han propuesto a lo largo del tiempo para describir el concepto "gobierno electrónico", como ser la OCDE (Organización para la Cooperación y el Desarrollo Económicos) que lo define como *"El uso de las tecnologías de la información y comunicación (TIC’s), particularmente la Internet, como una herramienta para alcanzar un mejor gobierno"* [14] .

Hoy en día existen varios países que han incorporado este concepto en las actividades del gobierno, como es el caso de Uruguay. Con el objetivo de mejorar los servicios brindados a los ciudadanos uruguayos se crea el organismo AGESIC (Agencia para el Desarrollo del Gobierno de Gestión Electrónica y la Sociedad de la Información y del Conocimiento) [13].

Para lograr los objetivos propuesto AGESIC cuenta con la Plataforma de Gobierno Electrónico (PGE) [1] que facilita la integración de los servicios ofrecidos por los organismos, brindándolos a los ciudadanos a través de internet. Las plataformas de gobierno electrónico generalmente se basan en una Arquitectura Orientada a Servicios (SOA).

Los servicios de información geográfica de a poco se han ido integrando a las operaciones del Estado, lo cual implica que las administraciones para responder a estas necesidades se planteen la integración de servicios geográficos en las plataformas de gobierno electrónico.

A pesar de los avances en las TIC's la integración entre los servicios de información geográfica y las plataformas de gobierno electrónico presentan algunas dificultades debido a que las plataformas se basan generalmente en Web Services y SOAP lo cual presenta incompatibilidades en los GIS y sus servicios vía internet [7]. En particular en Uruguay, la PGE brinda soporte a servicios basados en el estándar SOAP [11] y los servicios geográficos provistos por los organismos se basan en el estándar REST [8].

Para resolver dicha problemática en la tesis de maestría de Raquel Sosa [7] se identifican cinco escenarios de integración de información geográfica en el marco de gobierno electrónico tomando como referencia la arquitectura de Uruguay y se propone la inserción de Componentes de Transformación de Protocolos (CTP). Dichos escenarios van desde consultas de información por parte de público general y público especializado hasta modificación y generación de información geográfica.

## Objetivos

El objetivo principal que plantea el proyecto es diseñar e implementar la arquitectura propuesta para la integración de Web Services Geográficos en Plataformas de Gobierno Electrónico, tomando en cuenta los escenarios de aplicación propuestos en la tesis de maestría de Raquel Sosa.

Para lograr dicho objetivo se plantean los siguientes puntos:

* Estudiar los escenarios de aplicación para la integración de Web Services Geográficos en Plataformas de Gobierno Electrónico, la arquitectura propuesta y sus mecanismos de transformaciones.
* Diseñar e implementar un prototipo de los mecanismos de transformaciones para cubrir la mayor parte de los escenarios analizados.
* Desarrollar un caso de ejemplo que muestre los mecanismos implementados.

## Resultados esperados

Se espera que al finalizar el proyecto, se tenga un prototipo de la arquitectura para Integración de Web Services Geográficos en Plataformas de Gobierno Electrónico. El mismo contará con los mecanismos de transformaciones implementados para cubrir la mayor cantidad de escenarios de aplicación posibles.

Por otro lado, se espera poder contar con un caso de estudio que permita validar el prototipo implementado.

## Resultados alcanzados

## Gestión del proyecto

## Organización del documento

El presente documento se estructura en seis secciones más la sección de referencias.

En la sección dos se hace una breve reseña de los estándares y tecnologías que fueron necesarios conocer y aprender previo al desarrollo del proyecto. En las secciones tres y cuatro se detallan las decisiones propuestas de arquitectura y diseño respectivamente. En la sección cinco se desarrolla un caso de estudio que tiene como objetivo validar las decisiones de arquitectura y diseño planteadas en la sección anterior. Y por último, en la sección seis se concluye sobre el todo el proceso de desarrollo del proyecto y se proponen trabajos a futuros.

# Marco de trabajo.

A continuación se presenta una breve descripción de los principales estándares mencionados en el documento y que son necesarios para la resolución del problema planteado.

## Web Services Geográficos

Open Geospatial Consortium (OGC) [3] es la organización que propone los estándares de Web Services Geográficos, los cuales están agrupados bajo el nombre OpenGIS Web Services (OWS).

Las siguientes descripciones de los protocolos Web Map Service (WMS) y Web Feature Service (WFS) se arman a partir de la conjunción de [4], [5], [6], y [7].

### WMS

WMS [4] es un estándar que define un protocolo para obtener mapas dinámicos, de datos referenciados espacialmente a partir de información geográfica distribuida. El mapa que se obtiene para representar la información geográfica es un archivo de imagen en formato PNG, GIF o JPEG, o en ocasiones como elementos gráficos basados en las especificaciones Scalable Vector Graphics (SVG) o Web Computer Graphics Metafile (WebCGM).

WMS define tres operaciones para publicar datos geográficos:

**getCapabilities:** Devuelve los metadatos del servicio, tales como, formatos que soporta, datos que posee, información de los valores admitidos de los parámetros, etc.

**getMap:** Devuelve un mapa geográfico a partir de ciertas capas.

**getFeatureInfo:** Operación opcional que devuelve información acerca de las características particulares mostradas en el mapa.

### WFS

WFS [5] es un estándar que define un protocolo para consultar y modificar información geográfica codificada en Geography Markup Language (GML).

WFS define seis operaciones para consultar y modificar datos geográficos:

**GetCapabilities:** Devuelve información del servicio.

**DescribeFeatureType:** Describe la estructura de cualquier tipo de entidad soportado por el servicio.

**GetFeature:** Devuelve los datos geográficos resultado de la consulta realizada por el cliente (analogía con SELECT).

**GetGmlObject:** Devuelve cualquier elemento GML a través de su identificador único.

**Transaction:** Operación que permite modificar las características de los datos geográficos, tales como, darlos de alta, modificarlos y darlos de baja.

**LockFeature:** Operación opcional que permite bloquear una entidad al momento de realizar una transacción.

En base a las operaciones ofrecidas los servicios WFS se clasifican de la siguiente manera:

**WFS Básico:** Estos servicios son únicamente de lectura y ofrecen las operaciones GetCapabilities, DescribeFeatureType y GetFeature.

**WFS con XLink:** Ofrecen las operaciones de un WFS básico más la operación GetGmlObject. Con XLink significa que el servidor puede solicitar datos extra a otros servidores.

**WFS Transaccional:** Ofrecen las operaciones de un WFS básico más la operación Transaction.

**WFS Transaccional con XLink:** Ofrecen las operaciones de un servicio WFS Transaccional más la operación GetGmlObject.

## Estándares SOAP

### SOAP

SOAP(Simple Object Access Protocol) es un protocolo para la comunicación entre aplicaciones por medio de internet. Es independiente de la plataforma y el lenguaje y define un formato para envío y recepción de mensajes basado en XML [11].

Se publican servicios como RPC (remote call procedure) y la comunicación funciona por medio de un request enviado por el cliente a lo que se devuelve una respuesta del servidor.

Se utiliza el lenguaje WSDL (Web Service Description Language) para describir los servicios que se ofrecen, así como el formato de la entrada y la salida de cada servicio. Un web service SOAP  tiene una especificación asociada escrita en este lenguaje que está basado en XML [12].

### REST

Una API REST es un tipo de arquitectura de desarrollo web basada en el estándar HTTP y que permite interconectar sistemas hypermedia distribuidos.

A diferencia de SOAP no está orientado a servicios (RPC) sino a recursos. Un recurso es cualquier información que pueda ser almacenada, una entidad que representa un concepto de negocio que a su vez puede ser accedido por otro sistema. Cada recurso posee un identificador único que lo distingue de otros recursos.  Algunos recursos son estáticos, o sea, en cualquier momento en que sean examinados luego de su creación siempre les va a corresponder el mismo resultado. Otros cambian a medida que pasa el tiempo [8].

La comunicación es stateless, cada recurso posee un estado interno, este estado no puede ser accedido desde el exterior. Cada pedido hecho del cliente al servidor debe tener toda la información necesaria para que el request pueda ser comprendido. Y no se puede sacar ventaja de tener un contexto guardado para cada sesión. La información de las sesiones son almacenadas enteramente en el cliente. Esto también mejora la visibilidad, confiabilidad y escalabilidad del servicio[8].

Un servicio REST permite las siguientes 4 operaciones sobre un recurso :

**Get:** Para obtener información sobre un recurso.

**Post:** Para crear una nueva instancia del recurso.

**Put:** Para modificar un recurso existente.

**Delete:** Para eliminar un recurso.

Se puede entender como una aplicación CRUD (Create, Read, Update and Delete), pero a diferencia de estas, un servicio REST le puede asignar otro comportamiento a una de estas operaciones estándar.

### WS-SECURITY

Es una extensión de los estándares de Web Services SOAP, publicado y mantenido por OASIS, encargado de brindar seguridad extremo a extremo a nivel del mensaje SOAP. Características similares pueden obtenerse utilizando TLS, aunque tiene el problema de que el mensaje no puede utilizarse para ruteo intermedio, debido que TLS encripta todo el mensaje http.

WSS brinda dos servicios básicos, confidencialidad e integridad. Los cuales pueden alcanzarse utilizando diferentes métodos, XML Signature y XML Encription. Estos se basan e integran con diferentes tecnologías de seguridad, como tokens de seguridad, certificados X.509 y Kerberos, entre otras.

Provee ciertas ventajas sobre la opción de TLS, como por ejemplo se puede cifrar todo o parte del mensaje, lo cual permite utilizar las partes no criticas para ruteo o controles de acceso.

Se integra con otros estándares de seguridad como WS-SecureConversation y WS-Trust.

También soporta la integración con SAML utilizado para proveer SingleSign on a lo largo de internet, permitiendo la interacción entre Proveedores de identidad y proveedores de servicio que confían en los mismos.

#### WS-TRUST

WS-Trust define extensiones a WS-Security que proveen lo siguiente : [18]

* Métodos para emisión, renovación y validación de tokens de seguridad.
* Gestión de relaciones de confianza entre los agentes.[18]

Esta especificación define el modelo Web Services Trust Model. Esto define un proceso que permite restringir a que las peticiones que llegan a un servicio cumplan con una serie de especificaciones. Si quien hace la solicitud no cumple con todos los requerimientos necesarios, entonces debe pedir el acceso a un servidor STS (Security Token Service).  Esto último se considera que es una relación de confianza que existe entre el STS y el servicio al que se quiere acceder.[19]

Web Services Trust Model define los siguientes mecanismos para verificar relaciones de confianza : [18]

* Fixed trust roots: El destinatario tiene definido un set fijo de relaciones de confianza. Cuando lleguen pedidos va a evaluar si contiene tokens emitidos por alguna de esas relaciones.
* Trust hierarchies:  Existe una jerarquía de confianza y el servicio destinatario solo tiene el root de esa jerarquía. Por lo tanto cuando llegue una petición, este deberá verificar que pertenece a la jerarquía de confianza.
* Authentication service: Existe un servicio de autenticación con el cual el destinatario mantiene una relación de confianza. Cuando llega un token, el destinatario lo enviará a este servicio de autenticación, este último determinará si la autenticación es válida o no.

[18]

### WS-ADDRESSING

WS-Addressing es un estándar para agregar información de direccionamiento en mensajes SOAP. Específicamente define elementos XML para identificar endpoints y segurizar la identificación punto a punto de los mismos en mensajes SOAP. [16]

Básicamente se definen dos tipos de elementos que se incluyen en mensajes SOAP :

Endpoint References : Identifican el punto donde son dirigidos los mensajes.

Message Information Headers: Cabezales que contienen información sobre la identificación del mensaje

**Endpoint references** [16]

Especifica las siguientes propiedades:

wsa:Address : Una dirección (URI) que identifica el endpoint. Puede ser una dirección de red o una dirección lógica.

wsa:ReferenceProperties : Propiedades que identifican al recurso transportado.

wsa:ReferenceParameters : Parámetros asociados al endpoint para facilitar la interacción.

wsa:PortType : Identificador (en formato QName) para el portType  del recurso transportado.

wsa:ServiceName : Es opcional e identifica la descripción del servicio que contiene un WSDL con la descripción del endpoint que se hace referencia.

wsa:Policiy : Especifica una política (policy) que es relevante para la interacción con el endpoint.

**Message Information Headers** [16]

Definen las siguientes propiedades:

wsa:to   : destino del mensaje

wsa:from  : origen del mensaje (emisor)

wsa:replyTo : contiene el endpoint al cual dirigir una respuesta.

wsa:faultTo : especifica el endpoint al cual se dirigen los fallos.

wsa:action  :  Contiene un identificador para las semánticas implicadas en el mensaje. (una URI indicando una entrada, salida o mensaje de error)

wsa:messageID : Contiene una URI que identifica el mensaje en espacio y tiempo. No debe haber dos mensajes con el mismo identificador.

wsa:relatesTo : Contiene un par de valores que indican como este mensaje se relaciona con otro mensaje.

### MTOM

Es un estándar de la W3C para la transmisión de datos binarios de forma óptima, como lo indica su nombre: Message Transmission Optimization Mechanism [13].

Debido a que XML no provee una forma de incluir datos binarios, previo al surgimiento de MTIOM la única forma de enviar o recibir este tipo de datos era transformarlo a Base64.

Base64 es una técnica que transforma datos binarios en una tira de caracteres ASCII de forma reversible. Esta solución que todavía se utiliza para algunos contextos. El problema de base64 es que hace crecer el tamaño de los datos, porque para la transformación convierte 6 bits de información en su correspondiente carácter ASCII de 8 bits, agregando un total de 25% extra de información. A priori puede parecer poco, pero para archivos grandes, de Gigabytes, un 25% es mucho.

MTOM soluciona este problema haciendo uso de los pedidos multi-parte de http. Funciona agregando en el body del mensaje SOAP “punteros” hacia las partes del mensaje http donde vienen los datos binarios sin codificar. De esta manera no agrega el overhead de Base64.

## Tecnologías y productos estudiados.

### ESB

Enterprise service bus es un modelo de arquitectura de software usado para implementar comunicación entre sistemas orientados a servicios. Generalmente se basa en plataformas que implementan y potencian sistemas de mensajería, definiendo puntos de entrada, principalmente web services SOAP. Además agregan servicios dentro del bus como seguridad y confidencialidad, así como también ruteo de mensajes según condiciones del mensaje o del contexto. También proveen servicios de transformación de datos, para adecuar los mensajes a diferentes sistemas interactuando los cuales agregan o quitan información.



**Figura X – Esquema de ESB[[1]](#footnote-2)wiki**

#### JBOSS ESB

Es un producto de Red Hat y su comunidad, una implementación de la arquitectura ESB en Java, basada también en el servidor de aplicaciones JBoss de la misma compañía. Se basa en estándares, tanto de Java como de la Web para brindar conectividad y comunicación con muchos otros tipos de tecnologías. En la siguiente figura vemos un esquema de todas las tecnologías de comunicación soportadas por este producto.

**Figura X – Esquema de ESB[[2]](#footnote-3)**

Básicamente en Jboss ESB un servicio define un conjunto de Listeners de los cuales quiere estar al tanto cuando llegan mensajes, luego el servicio es notificado de que un mensaje ha llegado y define para éste, filtros o procesadores que trabajarán con la información contenida en él. Por último el servicio ruteará el mensaje hacia algún conector de salida para invocar otro servicio fuera del ESB o dará una respuesta a quien inició la comunicación en primera instancia.

### Servidores de mapas

#### GEOSERVER

GeoServer es un servidor de mapas de código abierto que permite a los usuarios compartir y editar datos geoespaciales. Está diseñado para la interoperabilidad, publica los datos de cualquier fuente importante de datos espaciales usando estándares abiertos[9].

Es la implementación de referencia de los estándares Open Geospatial Consortium (OGC) Web Feature Service (WFS) y Web Coverage Service (WCS). Soporta WMS, WFS y WFS-Transaccional [9].

Posee las siguientes características [10]:

* Es el más adecuado para los desarrolladores java ya que ofrece facilidad para interoperar con aplicaciones escritas en esa plataforma.
* Soporta WFS y WFS-T.
* Como servidor soporta una variedad de formatos de almacenamiento (PostGis, Oracle spatial, Mysql,  GeoTiff) tanto vectorial como raster.
* Reproyección al vuelo.
* WMS Tiling cache (Usa GeoWebCache como cliente de tiles WMS).

#### MAPSERVER

Es una plataforma de código abierto para publicar información geográfica y aplicaciones interactivas de mapas en la web. [17]

El proyecto es administrado y gestionado por el MapServer Project Steering Committee (PSC), el cual está autorizado por OSGeo. Mientras tanto las mejoras y mantenimientos del proyecto son realizadas por usuarios y programadores en todo el mundo. [17]

Básicamente MapServer es una aplicación CGI alojada en un servidor web. Cuando llega un pedido a MapServer, se usa información que fue enviada en la URL y el archivo de configuración (Map file, ver abajo) para crear una imagen del mapa solicitado. [17]

Una aplicación de MapServer consiste en los siguientes elementos. [17]

**Map file**: Es un archivo de texto que sirve como configuración de una aplicación MapServer. Aquí se define el área del mapa, donde están los datos, las capas del mapa incluyendo datasources, proyecciones y símbolos.

**Información geográfica**: MapServer puede utilizar varios tipos de datasource. Por defecto se utiliza ESRI Shape Format pero muchos otros formatos son soportados por la herramienta.

**Páginas HTML:** Son la interface entre el usuario y la aplicación de mapas. MapServer puede ser invocado para colocar un mapa estático en un html. Para que sea interactivo la imagen se debe colocar en un formulario para poder hacer varias invocaciones MapServer y que este devuelva las distintas imágenes.

**MapServer CGI:** Un archivo ejecutable que recibe pedidos y devuelve imágenes o datos.

**Web/HTTP server:** Puede ser cualquier servidor web que aloje las paginas html con mapas que son solicitadas por el usuario.

## Plataforma de eGob de Uruguay

El gobierno electrónico o gobierno en red radica en la idea de la construcción de una Administración Pública enfocada en el ciudadano, siempre accesible y más cercana y que hace uso intensivo de las Tecnologías de la Información y las Comunicaciones (TIC) [1].

La plataforma de gobierno electrónico es un facilitador para el desarrollo de servicios y tramites en línea que provee servicios transversales y herramientas comunes a los Organismos del Estado así como también servicios a personas, empresas y organismos. Es un medio que facilita la interoperabilidad y el intercambio de información entre Organismos. Es el contexto tecnológico y legal que permite asegurar que la información intercambiada cumpla con los requisitos legales y tecnológicos predefinidos [1].

**Figura X - Plataforma de Gobierno Electrónico y Otros Servicios[[3]](#footnote-4)**

### Plataforma de Interoperabilidad

*"La Plataforma de Interoperabilidad (PDI)  forma parte de la Plataforma de Gobierno Electrónico (PGE) de AGESIC y tiene como objetivo general facilitar y promover la implementación de servicios de Gobierno Electrónico en Uruguay. Para esto, la PDI brinda mecanismos que apuntan a simplificar la integración entre los organismos del Estado y a posibilitar un mejor aprovechamiento de sus activos." [[4]](#footnote-5)*



**Figura X - Plataforma de Interoperabilidad [[5]](#footnote-6)**

La PDI está basada en una arquitectura orientada a servicios (SOA) y compuesta por un sistema de control de acceso, un sistema de gestión de metadatos y una plataforma de middleware [2].

**Sistema de control de acceso:** Es el punto de entrada a la plataforma y provee mecanismos de autenticación y autorización para el consumo de servicios basados en XML [2].

**Sistema de gestión de metadatos:** Provee especificación de alto nivel de los conceptos relativos a de los servicios públicos [2].

**Plataforma de middleware:** Cuenta con mecanismos para facilitar el desarrollo, despliegue e integración de servicios y aplicaciones. Está integrado por dos ESB, uno de tecnología Microsoft y otro de tecnología Java, con el fin de obtener lo mejor de ambos y ampliar el espectro de posibilidades en cuanto a los métodos de conexión.  Los organismos pueden utilizar esta plataforma para publicar y descubrir servicios, así como utilizar las diferentes capacidades de mediación, las cuales permiten desacoplar clientes y servicios [2].

#### Plataforma de Middleware

Uno de los principales componentes de la Plataforma de Interoperabilidad de la PGE es la Plataforma de Middleware [20].

A continuación se describe la plataforma de middleware mediante el siguiente ejemplo:



**Figura X - Ejemplo de funcionamiento de la plataforma de middleware[[6]](#footnote-7)**

Los mensajes enviados por los usuarios solicitando el Servicio de Cédula (indicado en la figura) cuando llegan a la PGE primeramente pasan por los controles de seguridad y luego son enviados a la plataforma de middleware, la cual realiza las siguientes acciones:

1) Verificación sintáctica: Validaciones de integridad tales como verificación de nulos, estructuras de datos incompletas o errores en tipos de datos. En caso de error, el mensaje es rechazado y se notifica al cliente los motivos.

2) Verificación de políticas de seguridad: Validaciones para constatar que se cumple con las restricciones de seguridad definidas en la Ley 18.331 de Protección de Datos Personales y acción de Habeas Actas así como también las políticas definidas por la PGE. En caso de error, el mensaje es rechazado y se notifica al cliente los motivos.

3) Elección del destino del mensaje: En base a la política de direccionamiento de mensajes de la PGE y según ejemplo indicado en la figura se define que *"Siempre se enviará el mensaje al servicio del organismo B si hay menos de 100 pedidos concurrentes pendientes. En caso contrario, se redirigirán los pedidos al servicio del organismo C."* [[7]](#footnote-8)

4) Transformación de datos: Según especificaciones del servicio puede que sea necesario transformar el pedido. En el ejemplo, el servicio del organismo C requiere que la cédula de identidad contenga puntos y digito verificador.

5) Envío del mensaje al servicio: Se envía el mensaje al servicio destino.

##### Componentes de la Plataforma de Middleware



**Figura X - Plataforma de Middleware[[8]](#footnote-9)**

**Entornos de ejecución:** Generalmente las aplicaciones y servicios de la PGE se alojan en los propios organismos, pero para el caso en que los organismos no cuenten con la infraestructura de hardware o software necesaria para sus servicios la plataforma de middleware ofrece entornos de ejecución basados en tecnologías de middleware tales como servidores de aplicaciones, entre otros.

Por otro lado, los entornos de ejecución también se utilizan para servicios, componentes o aplicaciones que brindan funcionalidades comunes o utilitarias. Actualmente en la PGE existe un servicio "Timestamp" provisto por AGESIC, el cual provee la fecha y hora actual.

Las principales plataformas para el desarrollo de aplicaciones empresariales proporcionadas por la plataforma de middleware son: la plataforma .NET de Microsoft y la plataforma Java Enterprise Edition (Java EE). Esta última se provee a través del JBoss Enterprise SOA Platform.

**Registros de servicios:** Provee funcionalidades para que los organismos publiquen, describan, busquen y descubran servicios en la PGE.

****

**Figura X - Directorio de Servicios[[9]](#footnote-10)**

Actualmente el registro de servicios se maneja de forma interna a AGESIC, pero se planea brindar un registro UDDI para que los organismos puedan buscar y descubrir servicios de acuerdo a distintos criterios.

**Productos Enterprise Service Bus:** Proveen mecanismos que pueden ser utilizados por los organismos para el consumo y provisión de servicios. La plataforma de middleware cuenta con los siguientes productos de tipo ESB: JBoss ESB y Microsoft Biztalk Server complementado con Biztalk ESB Toolkit.

#### Sistema de Seguridad

El sistema de seguridad es el segundo componente más importante de la Plataforma de Interoperabilidad [xx].

A continuación se describe el sistema de seguridad mediante el siguiente ejemplo:



**Figura X - Ejemplo de funcionamiento del Sistema de Seguridad[[10]](#footnote-11)**

Cuando un organismo publica un servicio, este debe especificar quien tiene acceso a cada método del mismo ya que el control de acceso en la plataforma se realiza a nivel de métodos.

En el ejemplo indicando en la figura el organismo B provee el servicio de Cédula con los métodos getNombre y getFechaNacimiento y brinda a los funcionarios del organismo A acceso únicamente al método getNombre. De esta forma si el organismo A intenta acceder al método getFechaNacimiento, este será rechazado por la PGE. Esto permite que el organismo B se olvide de la seguridad delegando el control de acceso al Sistema de Seguridad de la PGE.

##### Componentes del Sistema de Seguridad



**Figura X - Sistema de Seguridad[[11]](#footnote-12)**

**TERMINAR.............................**

# Análisis y arquitectura.

## Escenarios propuestos en la tesis

En la tesis de Raquel Sosa se presentan y analizan cinco escenarios, donde una tecnología como la desarrollada sería útil. A continuación se describen brevemente dichos escenarios.

### Público general accediendo a información geográfica

Este escenario está pensado acceder a información geográfica que sea publicada solamente para consultas ya que es de interés general. Dicha información podrá ser accedida por tanto por organismos como por los ciudadanos o público en general.

Como se trata de información pública, este escenario no requiere mecanismos de seguridad pero si debe asegurar la disponibilidad del servicio.

### Público especializado consultando información

En este escenario un organismo brinda un servicio específico para que la información pueda ser consultada por otro organismo. En este caso la información solo está disponible para usuarios especiales pertenecientes del organismo cliente y no para el público en general.

Este escenario tiene como principal requerimiento la disponibilidad del servicio. Ya que esta misma se debe asegurar en fin de que el organismo que consume la información pueda cumplir con sus funciones.

### Instituciones colaborando en la creación de información geográfica

Este escenario contempla el caso en que un organismo modifica o genera la información geográfica que es mantenida por otro organismo. Esto se da en organismos que trabajan con un conjunto de datos pero no tienen la responsabilidad de mantener esos datos.

Este escenario requiere un fuerte control de seguridad ya que se debe permitir el acceso solo a funcionarios específicos del organismo colaborador. También es necesario validar la información recibida por parte del organismo que mantiene la misma.

### Instituciones colaborando en trámites

En este escenario dos o más organismos colaboran para la generación de trámites públicos. Los organismos participantes deben compartir datos para poder brindar el servicio a los usuarios. Este proceso le evita a un usuario tener que presentarse en los distintos organismos para recoger la información que necesita para hacer el trámite.

Entre los problemas a enfrentar en este escenario está generar una buena coordinación institucional. A su vez el proceso se hace más complejo cuando en un trámite participan varios organismos.

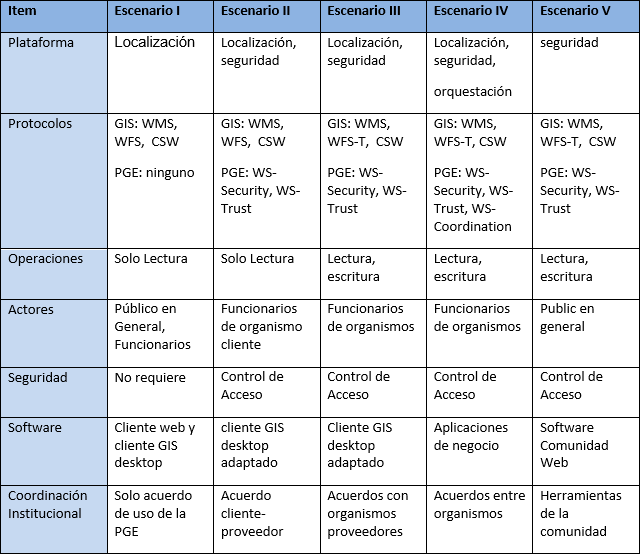
### Público generando información geográfica

En este escenario un usuario público puede modificar la información geográfica de un organismo.

Entre los desafíos que plantea este escenario está el de brindar mecanismos amigables para que los usuarios puedan enviar la información fácilmente. También es importante auditar los datos que son enviados de manera de filtrar solo los datos confiables.

### Resumen análisis de escenarios

En el siguiente cuadro se resumen las características de los escenarios descriptos [7].



**Figura X – Cuadro resumen de características de los escenarios.[[12]](#footnote-13)**

## Solución propuesta en la tesis

E

## Casos de uso

En esta sección se da una descripción de los casos de uso más críticos, elegidos especialmente porque obligan a tomar definiciones arquitectónicas específicas para poder soportar dichas interacciones en la plataforma. También se presenta una definición de los actores que interactúan en los mencionados casos de uso.

### Actores

#### Organismo proveedor

Es una entidad que contiene datos geográficos y ofrece servicios tanto de consulta como de actualización.

#### Organismo cliente

Es una entidad que consulta y/o actualiza datos geográficos de un organismo proveedor.

#### Público general

Es una persona con un perfil público que quiere actualizar o consultar información geográfica.

#### STS

Encargado  de la autenticación. Provee tokens de seguridad que luego la PGE se encarga de verificar.

#### PGE

Es el componente que se encarga de procesar todos los pedidos y controlar que sean correctos así como también maneja los mecanismos de seguridad y de balanceo de carga.

### Público general consultando información.

##### En este escenario se debe permitir a un usuario público consultar la información geográfica ofrecida por otro organismo.

### Instituciones colaborando en trámites.

Este caso se da cuando un organismo solicita información de otro organismo para realizar trámites solicitados por un usuario público.

## Problemas técnicos encontrados.

Describir problema con la versión de jboss-esb

## Decisiones de arquitectura

Esta sección describe algunas de las situaciones y problemas que fomentaron ciertas decisiones arquitectónicas para resolverlos Durante el desarrollo del proyecto.

### Mapeo de las direcciones del servicio

Uno de los grandes problemas enfrentados en el proyecto, el cual obligó a tomar ciertas decisiones arquitectónicas, fue resolver el mapeo de direcciones lógicas y físicas relacionadas a cualquier servicio expuesto en la PGE.

Por defecto existen tres direcciones para cualquier servicio en la PGE las cuales se detallan a continuación.

* **Dirección lógica del servicio**: es una cadena de caracteres en formato URL que identifica al servicio en la PGE, o sea es único.
* **Dirección física del servicio:** está dado por la URL, incluyendo IP y puerto del servidor donde está alojado físicamente el servicio. Esta dirección es por lo general interna a la REDuy y solo conocida por la PGE.
* **Dirección física del proxy del servicio**: URL del proxy por donde los clientes invocan el servicio en la PGE, esta dirección es la publicada.

A las direcciones ya mencionadas se agregan dos más:

* **Dirección física del CTP de entrada**: URL donde el cliente GIS hace la invocación REST para pedir la información geográfica.
* **Dirección física del CTP de salida**: URL donde la PGE reenvía el pedido SOAP, el cual es transformado a REST y luego enviado al servidor de mapas. Esta dirección en este caso viene a sustituir para la PGE la dirección física del servicio, pero esta no deja de existir porque es la dirección del servidor de mapas y debe ser conocida por el CTP de salida.

Dadas las pautas el problema es el siguiente. El cliente GIS debe conocer entonces solo la dirección del CTP de entrada, hasta acá no cambia nada en cuanto a la arquitectura. El CTP de entrada debe conocer la dirección física del proxy del servicio y la dirección lógica también, para poder hacer las peticiones a la PGE, esto implica configuración en el CTP. Pero luego se tiene para la salida de la PGE dos direcciones físicas, la del CTP salida y la del servidor de mapas. Esto no es soportado por la PGE, cuya relación entre dirección lógica y física es uno a uno para cualquier servicio.

Hay varias formas de solucionar este problema, una de ellas es modificar la PGE, lo cual fue descartado por las premisas del proyecto. Las otras opciones son diseñar una arquitectura acorde, esto último es explicado a continuación con una breve descripción de los pasos dados en el modelado para llegar a la solución final, la cual es descrita en profundidad en la sección 4.

### Opciones de despliegue

En la etapa de diseño se fueron manejando diferentes opciones para el despliegue de los componentes. Varias de estas opciones fueron descartadas por los motivos expuestos más adelante. Otras fueron cambiando hasta llegar a la opción elegida, la cual parece ser la que más se adapta a soportar la solución propuesta y tiene buenos niveles de adaptabilidad y baja complejidad de implantación.

#### CTP monolítico

En cierto momento se manejó la posibilidad de construir un CTP único totalmente externo a la PGE, que se encargara de recibir los pedidos de los clientes, encaminarlos a la PGE y esta luego de procesarlos los devolvería al CTP que haría el pedido al servidor de mapas correspondiente.

Como ventaja tiene que es un componente centralizado y manejable por un solo ente público, además acepta toda la responsabilidad de disponibilidad de los servicios deslindando de esta a los proveedores con infraestructuras débiles.

Las grandes contras por las que fue descartado es que se convierte en un único punto de falla susceptible a ataques a todos los servicios geográficos. Además necesita una infraestructura importante para no convertirse en un cuello de botella para la performance. Todo esto sumado a que la complejidad que alcanza la configuración de la plataforma la hace costosa de extender y mantener.

#### CTPs externos

Surge la necesidad de separar las tareas de los traductores, naciendo así el CTP RestConnector, el cual recibe los pedidos de los clientes y los encamina a la PGE. Y el CTP SoapConnector que enmascara el servidor de mapas recibiendo los pedidos SOAP provenientes de la PGE y los transforma a un pedido REST nuevamente.

No hay restricciones donde podría estar alojado el CTP RestConnector (RC), lo que elimina el único punto de falla, ya que podría haber varias instancias ejecutando al mismo tiempo, especialmente en los organismos clientes. Aún así puede convertirse en un cuello de botella y la carga de configuración de todos los servicios GIS sigue siendo grande.

Los CTP SoapConnector (SC) se alojan en los proveedores al igual que cualquier otro servicio ya

expuesto a través de la PGE. Acá surge un problema interesante con el manejo de las direcciones lógicas y físicas de los servicios en cuestión que es explicado anteriormente en el punto 3.4.1.

#### CTPs dentro de la PGE

Por motivo de dotar a los CTP de una infraestructura poderosa, eliminando así los problemas de cuello de botella y problemas de seguridad, también se exploró la opción que los CTP estuvieran dentro de la PGE. Esto implicaría modificar la PGE para incluir un punto de entrada del tipo REST. Lo cual es descartado porque la premisa del proyecto es no modificar y adaptarse a las soluciones existentes, para facilitar la rápida adopción. Eso incluye los clientes GIS, los servidores de mapas y la PGE.

#### CTPs en distribución ampliada

La arquitectura finalmente propuesta toma los criterios antes planteados, minimiza los problemas encontrados y reduce los riesgos ya mencionados. Hace uso extensivo de los recursos disponibles en la PGE.

Primero se plantea instanciar un servidor CTP SC por cada servidor de mapas a proteger en la PGE. Esto elimina uno de los mapeos de direcciones del servicio, lo que evita modificar la PGE. Además estos CTP SC estarán hosteados en los Execution Environment provistos por la PGE, lo cual elimina el problema de infraestructuras pobres en los entes.

Segundo, cada organismo que quiera consumir servicios geográficos deberá hostear su propio CTP RC, configurando solo los servicios que requiera para sus tareas. Permite reducir la configuración a su mínima expresión y responsabiliza al organismo cliente de los permisos que da sobre los servicios.

Estos CTP RC no deben estar accesibles por el público general, solo personal autorizado dentro del organismo.

Por último para aquellos organismos proveedores, que deseen exponer información pública, deberán hostear su propio CTP RC, el cual contendrá configurado solo los servicios propios.

Esta solución permite además que si un CTP RC de información pública es atacado, el acceso a esa información sigue estando disponible para los organismos clientes ya que acceden desde sus propios CTP RC.

# Diseño e implementación

## Arquitectura

Tomando en cuenta la solución propuesta por Raquel Sosa en su tesis de maestría [4] se tiene un conjunto de 5 subsistemas independientes que interactúan para resolver los casos de uso. Con respecto a la figura 6, de izquierda a derecha se tiene: El cliente GIS, el CTP de entrada llamado RestConnector, la PGE propiamente dicha, el CTP de salida llamado SoapConnector y el servidor de mapas que se quiere enmascarar.



Figura 6

Cliente GIS: Cualquier programa utilizado para consumir datos geográficos utilizando protocolos WMS y WFS. Puede ser desde una aplicación web usando OpenLayers hasta Gvsig.

CTP RestConnector: es un sub sistema encargado de recibir pedidos REST especificados según los protocolos WMS y WFS. Transforma estos pedidos a formato SOAP para que atraviesen la PGE. Y los encamina hacia ella. También según la configuración agregará información a los pedidos que requieran seguridad y otros datos exigidos por la PGE.

PGE: Es la plataforma de gobierno electrónico del estado uruguayo [1].

CTP SoapConnector: Este sistema estará registrado en la PGE como un proveedor de servicios, al cual se encaminarán los pedidos WMS y WFS previamente transformados por el CTP RestConnector. Su función es volver a componer el pedido REST para enviarlo al servidor de mapas configurado, y transformar la respuesta del mismo en un mensaje SOAP para que recorra el camino inverso. La idea es que exista un SoapConnector por cada servicio GIS expuesto a través de la PGE.

Servidor de mapas: Es un servidor que soporte los protocolos WMS y WFS. En principio habrá soporte para GeoServer [5] y MapServer [6].

### Diagramas de actividad

Los siguientes diagramas de actividad ilustran el flujo de datos de los



El flujo de datos a la inversa es automático debido a las tecnologías utilizadas que una vez terminada la invocación arman la respuesta para devolver al cliente.

## Implementación, diagrama de componentes, productos utilizados

En esta sección se describe cada uno de los subsistemas, debido a que son el objeto de ese proyecto se describen los CTPs y como se utiliza una versión reducida y minimalista de la PGE implementada específicamente para simular la PGE real en el contexto de este proyecto también se incluirán sus detalles en el siguiente apartado.

#### CTP RestConnector.

##### Diagrama de Componentes.



##### Descripción.

El RestConnector está compuesto de tres filtros con responsabilidades bien separadas que van agregando información al mensaje ESB para finalmente transformar el pedido de REST en un mensaje SOAP y un cliente de web service SOAP generado en algún framework de web services[7] que lo envía hacia la PGE.

El punto de entrada es un Getway HTTP cuya implementación es provista por la plataforma ESB. Este recibe las conexiones HTTP en una URI particular del servidor y lo transforma en un mensaje ESB que se transmite al primer filtro y se va pasando entre estos procesándolo como sea debido.

El primer filtro es el GisRestService, este se encarga de parsear el pedido WMS o WFS para averiguar que servicio geográfico expuesto a través de la PGE se quiere invocar. Para esto toma datos del pedido y consulta una base de conocimiento. Carga toda la información necesaria y delega al siguiente filtro.

El SecurityAggregator es el encargado de resolver las necesidades de autenticación y autorización que el servicio de la PGE requiera. Con la información proporcionada por el filtro anterior conformará el token de seguridad SAML que se requiere para atravesar la PGE.

Finalmente el SOAPClient utiliza un cliente de webservice generado a partir del webservice SOAP que provee el CTP SOAPConnector, para invocar el servicio geográfico con los datos necesarios incluyendo los parámetros del pedido original y los agregados por la PGE. Con la diferencia de que el endpoint invocado no es el propio SoapConnnector sino el proxy del servicio expuesto por la PGE.

Al recibir la respuesta se coloca en el mensaje ESB lo enviado por el proveedor y el servidor ESB se encarga automáticamente de convertir eso en una respuesta HTTP adecuada, ya que se ha terminado de invocar los filtros configurados para ese servicio.

Por otro lado se proyecta una aplicación web llamada ctp-config que será implementada en Grails. La funcionalidad de esta aplicación es brindar una interfaz de usuario agradable para configurar los diferentes servicios que pueden ser accedidos desde esa instancia del CTP y la seguridad que cada uno requiera. La información se guarda en una base de datos que será accedida tanto por el CTP como por la aplicación de configuración.

#### CTP SoapConnector.

##### Diagrama de Componentes.



##### Descripción.

El SoapConnector también está implementado en la tecnología ESB pero es bastante diferente al anterior. En este caso se configura un servicio cuyo punto de acceso en un procesador SOAP provisto por el ESB, el mismo recibe pedidos HTTP que contengan un mensaje SOAP, lo procesa e invoca a la clase que lo implementa, en este caso el componente GisSoapService, el cual tiene declarados como métodos todos los definidos por los estándares WMS y WFS. Este componente recibe toda la información y utiliza la configuración para armar la URL del servidor de mapas configurado. Dado que se utilizará una instancia de este sub sistema por cada servicio geográfico, es decir por cada servidor de mapas que se quiera exponer, solo se configura un servidor y su URL, IP, puerto y la URI. Esta URI puede variar según el protocolo y según la implementación del servidor, por ejemplo existen variantes entre GeoServer y MapServer.

El componente RestInvoker es el encargado de armar el pedido HTTP correspondiente, basado en los parámetros del pedido original y la URL provista por GisSoapService.

Para la respuesta, simplemente se retorna lo enviado por el servidor de mapas, los componentes provistos por ESB que implementan el webservice se encargan de convertirlo en un mensaje SOAP Response. Solo se debe tener en cuenta que para el método GetMap la respuesta es una imagen en formato binario y no un XML como en todos los demás métodos.

### Configuración de los ctps y pge

Visto toda la información que se debe manejar en los CTP, esta sección describe el modelo de datos utilizado para el CTP RC, para soportar la configuración de diferentes servicios y sus propiedades de seguridad. También se describe un modelo básico que se utilizó para el simulador de la PGE creado en el proyecto.

#### CTP RestConnector

El modelo de datos a continuación permite configurar servicios y sus propiedades de seguridad para que las peticiones puedan atravesar la plataforma en la forma debida.



Este modelo es guardado en una base de datos relacional estándar y se ha construido una aplicación web sencilla para configurarlo, la cual sería una interfaz de usuario que utilizaría el administrador del CTP, para cualquiera de sus modalidades, dentro de un organismo o expuesto en internet. Tanto la base de datos como la aplicación son locales a cada instancia del CTP RC. Esto quiere decir que no son centralizados.

#### CTP SoapConnector

Este caso es mucho más sencillo, ya que cada instancia del CTP SC invoca un único servidor de mapas, existe un archivo de configuración donde se definirán los datos de conexión.

|  |  |
| --- | --- |
| **Nombre de la propiedad** | **Valor de ejemplo** |
| url | <http://localhost:8081/geoserver> |
| Propwms | /integraciongispge/wms? |
| Propows | /integraciongispge/ows? |
| Propwfs | /integraciongispge/wfs? |

La propiedad “url” indica la ubicación del servidor, y luego es concatenada con una de las otras propiedades para completar la URL a donde se enviará el pedido.

Nótese que el ejemplo es para Geoserver, para Mapserver, las propiedades son las mismas, pero cambia el valor debido a la forma de las URLs en este servidor.

#### PGE

Para el simulador se eligió un modelo sencillo que soporte lo básico de la configuración de servicios. La idea es simular el comportamiento en un caso de éxito o casos de falta de configuración o invocaciones con datos incorrectos.



También es guardado en una base relacional, pero en este caso es solo por simplicidad. No se pretende usabilidad a futuro como en el CTP ya que es solo un simulador.

# Caso de Estudio

# Conclusiones y trabajo a futuro

# Referencias

1. Plataforma de Gobierno Electrónico. *AGSIC.* [En línea] [Citado el: 05 de 05 de 2014.] http://www.agesic.gub.uy/innovaportal/v/452/1/agesic/plataforma\_de\_gobierno\_electronico.html?menuderecho=5.

2. Plataforma de Interoperabilidad. *AGSIC.* [En línea] [Citado el: 05 de 05 de 2014.] http://agesic.gub.uy/innovaportal/v/1710/1/agesic/plataforma\_de\_interoperabilidad.html?menuderecho=5.

3. Open Geospatial Consortium. *OGC.* [En línea] [Citado el: 02 de 05 de 2014.] http://www.opengeospatial.org/.

4. Estándar WMS. *OGC.* [En línea] [Citado el: 02 de 05 de 2014.] http://www.opengeospatial.org/standards/wms.

5. Estándar WFS. *OGC.* [En línea] [Citado el: 02 de 05 de 2014.] http://www.opengeospatial.org/standards/wfs.

6. Notas del curso Taller de Sistemas de Información Geográficos Empresariales. *Facultad de Ingeniería de la Universidad de la República Oriental del Uruguay.* [En línea] [Citado el: 02 de 05 de 2014.] http://www.fing.edu.uy/inco/cursos/tsi/TSIG/clases2012/WebServicesGeograficos2012.pdf.

7. **Sosa, Raquel.** Integración de Servicios Geográficos en Plataformas de Gobierno Electrónico. *Facultad de Ingeniería de la Universidad de la República Oriental del Uruguay.* [En línea] [Citado el: 02 de 05 de 2014.] http://www.fing.edu.uy/~raquels/TesisRaquelSosa\_vf\_1.2.pdf.

8. **Fielding, Roy Thomas.** DISSERTATION. [En línea] [Citado el: 09 de 05 de 2014.] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\_arch\_style.htm.

9. GeoServer. [En línea] [Citado el: 09 de 05 de 2014.] http://geoserver.org/display/GEOS/Welcome.

10. OSGeo Live. [En línea] [Citado el: 09 de 05 de 2014.] http://live.osgeo.org/es/overview/geoserver\_overview.html.

11. SOAP Introduction. *W3Schools.* [En línea] [Citado el: 09 de 05 de 2014.] http://www.w3schools.com/webservices/ws\_soap\_intro.asp.

12. Introduction to WSDL. *W3Schools.* [En línea] [Citado el: 09 de 05 de 2014.] http://www.w3schools.com/webservices/ws\_wsdl\_intro.asp.

13. AGESIC. *Agencia de Gobierno Electrónico y Sociedad de la Información.* [En línea] [Citado el: 05 de 05 de 2014.] http://www.agesic.gub.uy/.

14. **Naser, Alejandra.** Gobierno Electrónico y Gestión Pública. *CEPAL (Comisión Económica para América Latina y el Caribe).* [En línea] [Citado el: 28 de 09 de 2014.] http://www.cepal.org/ilpes/noticias/paginas/5/39255/gobierno\_electronico\_anaser.pdf..

15. **POLICY BRIEF, OECD.** The e-government imperative: main findings. Marzo 2003. [En línea] [Citado el: 20 de 10 de 2014.] http://unpan1.un.org/intradoc/groups/public/documents/APCITY/UNPAN015120.pdf.

16. W3Schools. *Web Services Addressing.* [En línea] 10 de 08 de 2004. [Citado el: 23 de 10 de 2014.] http://www.w3.org/Submission/ws-addressing.

17. MapServer. [En línea] [Citado el: 15 de 08 de 2014.] http://mapserver.org/.

18. OASIS. *WS-Trust 1.4.* [En línea] http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html.

19. Marco de Desarrollo de la Junta de Andalucía. *Conceptos de seguridad en los servicios WEB.* [En línea] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/211#WS-Trust.

20. **PGE.** Capítulo III. *Descripción Técnica de la Plataforma de Gobierno Electrónico.* [En línea] [Citado el: 27 de 02 de 2015.] http://www.agesic.gub.uy/innovaportal/file/1454/1/capitulo\_3.pdf.

1. Imagen tomada de [1]. [↑](#footnote-ref-2)
2. Imagen tomada de [1]. [↑](#footnote-ref-3)
3. Imagen tomada de [1]. [↑](#footnote-ref-4)
4. Cita tomada de [2]. [↑](#footnote-ref-5)
5. Imagen tomada de [2]. [↑](#footnote-ref-6)
6. Imagen tomada de [20]. [↑](#footnote-ref-7)
7. Cita tomada de [20]. [↑](#footnote-ref-8)
8. Imagen tomada de [20]. [↑](#footnote-ref-9)
9. Imagen tomada de [20]. [↑](#footnote-ref-10)
10. Imagen tomada de [20]. [↑](#footnote-ref-11)
11. Imagen tomada de [20]. [↑](#footnote-ref-12)
12. Imagen tomada de [7]. [↑](#footnote-ref-13)