INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCUPLINARIA DE INGENIERÍA Y CIENCIAS SOCIALES Y ADMINISTRATIVAS

SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN.

IMPLEMENTACIÓN DE CONTROLES DE SEGURIDAD EN ARQUITECTURAS ORIENTADAS A SERVICIOS (SOA) PARA SERVICIOS WEB

T E S I S

QUE PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS CON ESPECIALIDAD EN INFORMÁTICA
P R E S E N T A:

EMILIO ANAYA LOPEZ.

DIRECTOR
M. EN C. RAFAEL IBAÑEZ CASTAÑEDA

MÉXICO, D.F **AÑO 2011**

SIP-14

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00hrs. horas del día 21 del mes de junio del 2011 se reunieron los miembros de la Comisión Revisora de Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de UPIICSA para examinar la tesis titulada:

" IMPLEMENTACIÓN DE CONTROLES DE SEGURIDAD EN ARQUITECTURAS ORIENTADAS A SERVICIOS (SOA) PARA SERVICIOS WEB"

Presentada por el alumno:

ANAYA	LÓPEZ	EMILIO
Apellido paterno	Apellido materno	Nombre(s)

Con registro: B 0 8 1 8 7 2

aspirante de:

MAESTRO EN CIENCIAS EN INFORMÁTICA

Después de intercambiar opiniones, los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Director de tesis

M. en C. RAFAEL IBAÑEZ CASTAÑEDA

DR. MAURICIO JORGE PROCEL MORENO

M. en C. GUILLERMO PÉREZ VÁZQUEZ

DR. EDUARDO GUTIÉRREZ GONZÁLEZ

M. en C. ABEL BUENO MEZA

LA PRESIDENTA DEL COLEGIO

DRA. MARÍA ELENA TAVERA CORTÉS

I. P. N.
ESTADOS UNIDOS MEXICANOS
U. P. I. I. C. S. A
SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, Distrito Federal, el día 20 del mes de junio del año 2011, el que suscribe Emilio Anaya López alumno del Programa de Maestría en Ciencias en Informática con número de registro B081872, adscrito a la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del M. en C. Rafael Ibañez Castañeda y cede los derechos del trabajo intitulado IMPLEMENTACIÓN DE CONTROLES DE SEGURIDAD EN ARQUITECTURAS ORIENTADAS A SERVICIOS (SOA) PARA SERVICIOS WEB, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección emilioal1977@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Emilio Anaya López

A mis padres por su amor y entereza.

A mi hermano por ser un ejemplo a seguir.

AGRADECIMIENTOS

A mi director de tesis por su apoyo y orientación durante la maestría y la elaboración de esta tesis.

A mi comité tutorial por sus correcciones y aportaciones en el presente trabajo y su valioso apoyo.

A mis profesores por compartirme sus conocimientos y experiencia.

A mis compañeros de trabajo por concederme las facilidades para realizar la maestría.

Resumen

El uso de los servicios web como una herramienta para integrar de aplicaciones, intercambiar información y realizar transacciones electrónicas en internet, ha generado nuevos problemas y oportunidades. Esto se debe a las características de intercambio de información y los niveles de apertura que tienen los servicios web.

Esta tesis tiene como finalidad proporcionar una solución práctica a los problemas relacionados con la seguridad en los servicios web al intercambiar información entre dos o más aplicaciones. Se plantea un caso práctico en el que se desarrolla una propuesta de solución y el procedimiento detallado para resolver los problemas haciendo uso de los estándares WS-Security, XML Signature , XML Encryption y SAML.

Como resultado de este trabajo se concluye que el uso de estándares de seguridad basados en XML, es una alternativa segura y confiable a un bajo costo a los problemas de seguridad a los que se encuentran expuestos los servicios web.

Abstract

The use of Web services as a tool to integrate applications, share information and carry out electronic transactions on the Internet has created new problems and opportunities. This is due to the characteristics of information exchange and opening levels that have Web services.

This thesis aims to provide a practical solution to problems related to security in web services to exchange information between two or more applications. It presents a case study that develops a proposed solution and the detailed procedure to solve the problem by using the standards WS-Security, XML Signature, XML Encryption and SAML.

As a result of this work concludes that the use of security standards based on XML is a safe and reliable low cost alternative to the security problems that are exposed Web services.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Con la aparición de internet y su rápida expansión, se ha creado un medio ideal para el desarrollo de aplicaciones que realizan intercambio de información, transacciones electrónicas, venta de productos y servicios en línea, entre otras funcionalidades. Este factor ha impulsado a las organizaciones a aumentar su interacción con clientes y otras organizaciones utilizando este medio.

Una de las herramientas más utilizadas para facilitar la integración de este tipo de aplicaciones son los servicios web. Sin embargo con su llegada fueron revelados nuevos problemas y oportunidades que antes no existían en los entornos cerrados. Los niveles de apertura y sus características de intercambio de datos e interoperabilidad también han significado nuevos desafíos para asegurar datos e identidades[1]. Es por esto que los servicios web han tenido una gran aceptación solo en aquellas aplicaciones que no requieren seguridad en sus transacciones.

La solución que se pretende obtener al concluir esta investigación es una guía práctica que facilite al lector, la comprensión y resolución de algunos de los problemas de seguridad que se originan al emplear servicios web en la integración de aplicaciones.

El primer capítulo inicia con la definición de los servicios web, ventajas, desventajas y vulnerabilidades. Después se define y delimita el problema, el cual consiste en solucionar los problemas de seguridad como la autorización, autenticación, no repudio de la información, confidencialidad e integridad en los servicios web. Para esto se establece el objetivo, alcances y limitaciones de la solución.

El siguiente capítulo es el marco teórico que tiene como finalidad ayudar al lector a comprender los conceptos que se utilizaran durante el desarrollo de la solución. Las teorías que conforman este capítulo tratan acerca de arquitecturas orientadas a servicios, servicios web y estándares de seguridad aplicadas a servicios web como son: WS-Security, XML-Encription y XML-Signature.

[1] David Chappell, Java Web Services, O´Reilly, 2002, pág.227

El capítulo 3 está formado por la propuesta de solución, el caso práctico, el diseño conceptual y el diseño físico. El caso práctico consiste en el envió de transacciones entre dos aplicaciones de manera segura utilizando servicios web. En el diseño conceptual se documentan los requerimientos del caso práctico. Finalmente en el diseño físico se definen los componentes de hardware, software y la arquitectura de la solución, se desarrollaran los servicios web y clientes proxy, se crearan los certificados digitales y se configuran los mecanismos de seguridad.

El capítulo 4, describe detalladamente la implementación y configuración de los componente que integran la solución del caso práctico descrito en el capítulo anterior, además se especifican las configuraciones y los escenarios que se utilizaran en las pruebas de funcionalidad y vulnerabilidad, para concluir se presentan los resultados de las pruebas realizadas.

Capitulo 1: Servicios Web y Definición del Problema

Este capítulo inicia explicando la definición formal de los servicios web, se mencionan ejemplos del uso de los servicios web para la integración de aplicaciones; se indican también los riesgos de seguridad a los que se encuentran expuestos. Finalmente se define y justifica el problema basándose en los requisitos de seguridad que los servicios web requieren y se limita el alcance de la solución propuesta.

1.1. Servicios web

Existen varias definiciones acerca de los servicios web. ~~IBM los define como:~~ "Tanto un servicio web como los servicios web son auto contenidos, aplicaciones modulares que pueden ser descritas, publicadas, localizadas, e invocadas a través de una red, en general, la World Wide Web." [2]

Otra definición de servicio web explica: "Un servicio web se describe así mismo y a las aplicaciones empresariales modulares que exponen la lógica de negocio como servicios sobre Internet a través de interfaces programables y el uso de protocolos de Internet con el propósito de proporcionar formas de buscar, suscribirse e invocar esos servicios."[3]

De forma resumida un servicio web es una aplicación modular publicada en Internet, que permite conectarse con otras interfaces, de forma sencilla.

Algunos ejemplos del uso de los servicios web son:

- Validaciones de tarjetas de crédito y autorizaciones entre instituciones bancarias
- Consultas enviando mensajes de texto desde teléfonos celulares,
- Consultas a bases de datos como las que permiten realizan el Registro Nacional de Población (RENAPO) y la Secretaria de Administración Tributaria (SAT)
- Servicios de búsqueda en internet utilizando Google.

[2] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003, pág. 4

[3] Ramesh Nagappan, Robert Skoczylas, Rima Patel Sriganesh, Developing Java Web Services, Wiley Publishing Inc., 2003, pág. 22

1.1.1. Ventajas y Desventajas

Los enfoques de diversos autores enuncian una serie de ventajas y desventajas, con una perspectiva un tanto distinta, a continuación se mencionan algunas de las ventajas que ofrecen los servicios web:

- Totalmente independientes de la plataforma, no hay restricciones en cuanto a la plataforma en la que pueden ser desarrollados, las aplicaciones que utilizan los servicios web pueden ejecutarse en cualquier plataforma[4].
- Basados en estándares de XML, los servicios web pueden ser desarrollados como componentes de aplicación débilmente acoplados utilizando cualquier lenguaje de programación, cualquier protocolo o plataforma[5].
- Cualquier programa puede ser mapeado a un servicio web y cualquier servicio web a cualquier programa[6].
- Al utilizar protocolos de Internet estándar, la mayoría de las organizaciones ya cuentan con gran parte del software de comunicaciones y la infraestructura necesarios para la implementación de los servicios web[7].

Como resultado de la evaluación de diferentes enfoques, las desventajas que hay que tener presentes para cualquier implementación son:

- Los servicios web no son una tecnología probada; existen sospechas de que son una solución de moda y como muchas otras soluciones al problema de procesamiento distribuido en el pasado, no van a cumplir lo prometido.
- La dependencia de los servicios web con XML, el uso de XML incrementa el tamaño de los datos varias veces, el tamaño de un mensaje de SOAP se traduce en más almacenamiento y tiempo de transmisión. La flexibilidad de SOAP significa que más procesamiento es necesario para formatear y analizar los mensajes.[8]
- Cuando se liberan nuevas versiones de servicios web pueden ser no compatibles con versiones anteriores, no es claro como los estándares para los servicios web soportarán las versiones[9].
- Las capas del stack de los servicios web no considera la seguridad, autenticación, flujo de trabajo e identidad.

---

[4] Anura Gurugé, Web Services: Theory and Practice, Digital Press, 2004, pág. 9

5 Ramesh Nagappan, Robert Skoczylas, Rima Patel Sriganesh, Developing Java Web Services, Wiley Publishing Inc., 2003, pág. 22

6 Eric Newcomer, Understanding Web Services, Pearson Education Corporate Sales Division, 2002, pág. 3

7 Bret Hartman, Donald J. Flinn, Konstantin Beznosov, Shirley Kawamoto, Mastering Web Services Security, Wiley Publishing Inc., 2003, pág.29

8 Bret Hartman, Donald J. Flinn, Konstantin Beznosov, Shirley Kawamoto, Mastering Web Services Security, Wiley Publishing Inc., 2003, pág.30

9 David Chappell, Java Web Services, O´Reilly, 2002, pág.186

1.1.2. Vulnerabilidades

Al encontrarse en ambientes abiertos los servicios web se encuentran expuestos, a continuación se presentan los tipos de ataques más relevantes que pueden afectar a los servicios web:

- Ataques externos: Las aplicaciones de e-business intercambian información que es muy valiosa. Las e-business son empresas que intercambian miles de registros de pacientes y comercio de acciones que valen millones. Para los servicios web basados en Internet, los ataques a estos sistemas pueden ser montados en cualquier máquina de escritorio en el mundo utilizando herramientas de software muy simples.
- Ataques internos: Se ha sabido que la mayoría de las violaciones de seguridad son realizadas por empleados que se presumen son de confianza. Pueden establecer una trampa para acceder a datos corporativos después de dejar la empresa. Además de que es posible que puedan cometer fraudes creando clientes ficticios, a fin de negociar con acciones o fabricando mercancías[10].

- Los servicios web están diseñados para ser abiertos e interoperables. Desde que se establecen los cortafuegos para permitir pasar el tráfico HTTP, las solicitudes de los servicios web por HTTP pasan a través de los cortafuegos fácilmente, dejando a la red interna expuesta[11].

- Los datos envueltos en sobres SOAP proveen un camino para entender la estructura y el significado de los datos que son enviados y recibidos por los servicios web[12].

[10] Bret Hartman, Donald J. Flinn, Konstantin Beznosov, Shirley Kawamoto, Mastering Web Services Security, Wiley Publishing Inc., 2003, pág.350

[11] Bret Hartman, Donald J. Flinn, Konstantin Beznosov, Shirley Kawamoto, Mastering Web Services Security, Wiley Publishing Inc., 2003, pág.351

[12] David Chappell, Java Web Services, O´Reilly, 2002, pág.227

1.2. Delimitación y justificación del problema

En el punto anterior se mencionaron las ventajas, desventajas y vulnerabilidades de los servicios web, además se dieron varios ejemplos del uso de los servicios web para la integración de aplicaciones. Considerando todos estos elementos se procede a la definición y delimitación del problema a solucionar.

1.2.1. Objetivo

El objetivo de este trabajo es la implementación de controles de seguridad que permitan y garanticen la autorización, autenticación, no repudio de la información, confidencialidad e integridad en los servicios web, con la intención de generar una guía práctica que ofrezca una solución a los problemas de seguridad que se presentan cuando se utilizan servicios web para la integración de aplicaciones y servicios.

1.2.2. Justificación del problema

La necesidad de implementar controles de seguridad en los servicios web como un componente independiente, se origina por la falta de tecnologías en el stack de los servicios web que permitan implementar soluciones relacionados a la seguridad. La siguiente tabla contiene las tecnologías que forman el stack con el que los servicios web son implementados:

Nombre	Tecnología
Directorio	Universal Description, Discovery, and Integration (UDDI)
Descripción	Web Service Description Language (WSDL)
Empaquetado	Simple Object Access Protocol (SOAP) y Extensible Markup Language (XML)
Transporte	Hyper Text Transfer Protocol (HTTP)
Red	TCP/IP

Tabla 1: Tecnologías del stack de los servicios web

Como se puede apreciar en la tabla anterior, las capas del stack de los servicios web no proveen una solución completa a muchos problemas de negocios. Por ejemplo, no se

ocupa de la seguridad, autenticación, flujo de trabajo, identidad, y muchas otras preocupaciones del negocio[13].

1.2.3. Alcances

El presente trabajo se centrará en el análisis e implementación de elementos de autenticación, autorización, no repudio en la información, confidencialidad e integridad en servicios web, utilizando como guía las especificaciones de los estándares de seguridad WS-Security, XML Signature, XML Encryption y SAML, que serán tratados a detalle en el capítulo 2.

Como parte del alcance de este trabajo es necesario mencionar que la implementación de la solución, se realizara utilizando únicamente herramientas y productos de software publicados bajo una licencia libre. Lo anterior con la finalidad de reducir costos de licenciamiento de software en la implementación de la solución.

1.2.4. Limitaciones

Debido a que existen una gran cantidad de opciones para resolver los problemas de seguridad en los servicios web, este trabajo se centrara específicamente en la implementación de seguridad utilizando la capa de trasporte SOAP de los servicios web.

Es necesario considerar que los certificados digitales no serán generados por una entidad certificadora por lo que su uso es meramente demostrativo. En el caso de que se requiera llevar el aplicativo a un ambiente productivo se deben solicitar los certificados a una entidad certificadora.

Finalmente este trabajo no dará solución a problemas relacionados a Infraestructura de clave pública (PKI) y Administración de políticas de acceso en servicios web.

[13] Doug Tidwell, James Snell, Pavel Kulchenko, Programming Web Services with SOAP, O'Reilly, 2001, pág. 10

Capitulo 2: Plataforma SOA y Seguridad en Servicios Web

La finalidad de este capítulo es proveer la teoría, conceptos y definiciones necesarias para poder solucionar el problema planteado en el capítulo anterior. Se inicia con la definición y conceptos de diseño de SOA. Finalmente se presenta la teoría acerca de los requisitos y estándares de seguridad que los servicios web implementaran.

2.1. Arquitectura Orientada a Servicios

La definición de una Arquitectura Orientada a Servicios es una arquitectura de software que está basada en conceptos clave que pueden ser interfaces de aplicaciones, servicios, repositorios de servicios y servicios de bus. Un servicio consiste en un contrato, una o más interfaces y una implementación[14].

SOA establece un modelo de arquitectura que tiene como objetivo mejorar la eficiencia, agilidad y productividad de una empresa por servicios de posicionamiento como el principal medio, a través del cual, la lógica de la solución es representada en soporte de la realización de los objetivos estratégicos, asociados con la computación orientada a servicios. Como una forma de arquitectura de tecnología, una implementación de SOA puede consistir en una combinación de tecnologías, productos, APIs, soportando extensiones de infraestructura, y algunas otras partes[15].

2.1.1. Elementos

Son varios los elementos que integran SOA, Interfaces de aplicación, servicios de contrato, servicios de interface, servicios de implementación, servicios de lógica de negocio y servicios de datos, juntos permiten la implementación de una arquitectura orientada a servicios.

a) Interfaces de aplicación

Las interfaces de aplicación son los actores activos en la arquitectura, su función es Iniciar y controlar todas las actividades de los sistemas de la empresa. Las interfaces de aplicación más utilizadas son:

[14] Dirk Krafzig, Karl Banke, Dirk Slama, Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall PTR, 2004, pág.78
[15] [10] Thomas Erl., SOA: principles of service design, Prentice Hall, 2008, pág.38

- Interfaces gráficas de usuario: Éste tipo de interfaz permite a los usuarios finales interactuar directamente con la aplicación, las interfaces gráficas pueden ser aplicaciones web o clientes ricos.
- Programas de lotes o procesos: Los programas o procesos de larga vida invocan su funcionalidad de manera periódica o son el resultado de acontecimientos concretos

Sin embargo, es posible que una interfaz de aplicación delegue gran parte de la responsabilidad a servicios o procesos de negocio.

b) Servicios de Contrato

Los servicios de contrato proporcionan una especificación informal de la finalidad, funcionalidad, restricciones y el uso del servicio. La forma de esta especificación puede variar, dependiendo del tipo de servicio. Un elemento no obligatorio de los servicios de contrato es una definición de interfaz formal basada en lenguajes como son el lenguaje de definición de Interface (IDL) o el lenguaje de descripción del servicio web (WSDL). Estos elementos proporcionan abstracción e independencia de tecnología, incluyendo el lenguaje de programación, el protocolo de middleware de la red y su entorno de ejecución. El contrato puede imponer la semántica detallada en las funciones y parámetros que no están sujetos a las especificaciones IDL o WSDL. Es importante comprender que cada servicio requiere un servicio de contrato en particular si no hay una descripción formal basada en una norma como WSDL o IDL.1

c) Servicios de Interfaz

La funcionalidad de los servicios es expuesta a los clientes por el servicio de interfaz, los clientes deben estar conectados al servicio utilizando una red. Aunque la descripción de la interfaz es parte del servicio de contrato, la implementación física de la interfaz consta de esbozos del servicio, que están incorporados en los clientes de un servicio y un despachador.

d) Servicios de implementación

Los servicios de implementación proporcionan físicamente la lógica de negocios requerida y los datos que son apropiados. Esto es la realización técnica que cumple con servicio de contrato. El servicio de implementación consiste de uno o más artefactos como son programas, datos de configuración y bases de datos[16].

[16] Dirk Krafzig, Karl Banke, Dirk Slama, Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall PTR, 2004, pág.81

e) Servicios de Lógica de Negocio

Los servicios de lógica de negocio son los encargados de encapsular la lógica de negocio como parte de su implementación. Ésta se encuentra disponible a través de interfaces de servicios. Sin embargo, la programación en contra de las interfaces es deseable, si se aplica un planteamiento orientado al servicio[17].

f) Servicios de Datos

Un servicio también puede incluir datos. En particular, este es el propósito de los servicios de datos céntricos[18].

2.1.2. Diseño y Tecnología

Existen varios objetivos generales que deben estar presentes cuando se realiza un diseño orientado a servicios, estos objetivos nos ayudan a cumplir con un correcto diseño de nuestra arquitectura.

- Determinar el conjunto básico de extensiones de la arquitectura.
- Establecer los límites de la arquitectura.
- Identificar los estándares de diseño necesarios.
- Definir diseños de interfaces de servicios abstractos.
- Identificar las composiciones de servicios potenciales.
- Evaluar el apoyo para los principios de orientación de servicio.
- Explorar el apoyo a las características de la arquitectura SOA contemporánea.

El proceso de diseño orientado a servicios consiste en una serie de pasos iterativos que establecen el marco para la creación del diseño de diferentes tipos de servicios, y finalmente del diseño de de la solución global[19].

Paso 1: Composición SOA

Con Independencia de la forma o el tamaño de la arquitectura, se debe tener un número de componentes de tecnología que establecerán un entorno en el que los servicios funcionaran.

[17] Dirk Krafzig, Karl Banke, Dirk Slama, Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall PTR, 2004, pág.81
[18] Dirk Krafzig, Karl Banke, Dirk Slama, Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall PTR, 2004, pág.81
[19] Thomas Erl., SOA: principles of service design, Prentice Hall, 2008, pág.52

Los componentes fundamentales que generalmente comprenden una arquitectura deben incluir:

- Una representación de la arquitectura de datos XML
- Servicios web basados en estándares de la industria.
- Una plataforma capaz de recibir y procesar los datos XML y servicios web

Paso 1:1 Selección de Capas del Servicio

Componer una arquitectura requiere decidir una configuración de diseño para las capas de servicios que comprenderán y estandarizaran la representación de la lógica dentro de la arquitectura.

Paso 1.2: Normas Fundamentales de la Posición

Se deben valorar las normas fundamentales que debe incluir la arquitectura y cómo deben aplicarse para apoyar de la mejor manera las características y necesidades de la solución orientada a servicios.

Paso 1.3: Elección de Extensiones de SOA

Se requiere determinar qué características de SOA se quiere que la arquitectura orientada a servicios de soporte. Esto nos ayudará a decidir cuál de las especificaciones disponibles WS-* deben formar parte de nuestro entorno orientado a servicios.

Paso 2: Diseño de Servicios de Negocio de Entidades Centralizadas

Los servicios de negocio de entidades centralizadas representan una capa de servicio que es la menos influenciada por otras capas. Su propósito es representar con exactitud los datos correspondientes a entidades definidas en los modelos de negocio de una organización. Estos servicios son estrictamente soluciones y procesos de negocio-agnóstico, construidos para la reutilización de cualquier aplicación que necesite acceder o manejar la información asociada con una entidad en particular.

Paso 2.1: Revisión de los Servicios Existentes

El primer paso en el diseño de un nuevo servicio es confirmar si en realidad es necesario. Si existen otros servicios que puedan proporcionar alguna o todas de las funciones identificadas en la operación, servicios candidatos que tengan un marco adecuado en el cual se puedan implementar esas nuevas operaciones.

Paso 2.2: Definir los Tipos de Mensaje de Esquema

Para comenzar un diseño de interfaz de servicio con una definición formal de los mensajes que el servicio requiere para procesar. Es necesario formalizar las estructuras de mensaje que se definen dentro del área de los tipos del WSDL.

Los mensajes SOAP llevan los datos de carga dentro de la sección body de SOAP. Estos datos necesitan ser organizados y escritos, utilizando esquemas XSD. En el caso de un servicio de entidad centralizado, es de gran ayuda utilizar XSD para representar con exactitud la información asociada. Este esquema de entidad centralizada puede ser la base para la definición del servicio WSDL.

Paso 2.3: Obtenga una Interfaz de Servicio Resumen

A continuación, se deben analizar los servicios candidatos propuestos y seguir los siguientes pasos para definir la interfaz de servicio inicial:

1. Confirmar que cada candidato de operación es adecuado, genérico y reutilizable, asegurando que la granularidad de la lógica de encapsulado es la adecuada.

2. Crear el área de <portType> en el documento WSDL y rellenarla con la operación que corresponden a los candidatos operación.

3. Formalizar la lista de los valores de entrada y de salida necesaria para adaptarse a la transformación de la lógica de cada operación. Esto se logra definiendo el <message> apropiado que hace referencia a los tipos del esquema XSD de los elementos <part> hijos.

Paso 2.4: Implementación de los Principios de Orientación al Servicio

Se deben aplicar los 4 principios de orientación a servicios que son los siguientes:

- Reutilización de servicio
- Autonomía de servicio
- Independencia de servicio
- Identificación de servicio

Paso 2.5: Estandarización y Perfección de la Interfaz de Servicio

Se deben revisar de las normas de diseño y las directrices existentes y aplicar las que sean apropiadas.

Paso 2.6: Extender el Diseño del Servicio

El proceso de modelado de servicios tiende a centrarse en las necesidades evidentes del negocio. Aunque la reutilización siempre es una buena opción, frecuentemente se utiliza el proceso de diseño para garantizar que la mayoría de la funcionalidad reutilizable se construya en diferentes servicios.

Esto implica la realización de un análisis especulativo de lo que otras características en el contexto funcional predefinido del servicio deben ofrecer.

Existen dos formas para implementar nueva funcionalidad:

- Adición de nuevas operaciones
- Adición de nuevos parámetros para las operaciones existentes

Mientras que la última opción puede simplificar las interfaces de servicio, también puede ser contra-intuitivo ya que demasiados parámetros asociados con una operación pueden requerir que los solicitantes de servicios necesiten saber mucho acerca del servicio para utilizarlo de manera efectiva.

Paso 2.7: Identificación de Procesamiento Requerido

Una vez que se tiene un diseño real del nuevo servicio de negocios, se deben estudiar las condiciones de transformación de cada una de sus operaciones detalladamente. Lo anterior para determinar si los servicios de aplicaciones adicionales son necesarios para realizar cada parte de la funcionalidad expuesta. Si se determina la necesidad de nuevos servicios de aplicación, se tendrá que determinar si ya existen, o si deben añadirse a la lista de servicios que se ofrecen como parte de esta solución.

Paso 3: Proceso de Diseño de Aplicaciones de Servicio

Los servicios de aplicación representan la sub-capa inferior de la capa de servicio integrado, son los responsables de llevar a cabo las demandas de procesamiento dictados por el negocio y las capas de la orquestación.

El diseño de servicios de aplicación no requiere experiencia en análisis de negocios. La capa de servicios de aplicación es una abstracción de servicio orientados a ambientes técnicos de una organización, mejor definido por aquellos que entienden la mayoría de estos entornos.

Paso 3.1: Revisión de los Servicios Existentes

Es importante asegurar que la funcionalidad requerida como parte del servicio de aplicación candidato, de alguna manera, o forma, ya existen. Por lo que es necesario revisar el inventario de servicios de aplicación existente para buscar algo parecido a lo que está a punto de ser diseñado.

Paso 3.2: Confirmación del Contexto

Al realizar un análisis orientado a servicios, se centra en los requisitos de negocio inmediato. Como resultado, los servicios de aplicación candidatos producidos en esta etapa, no toman el contexto establecido por los servicios de aplicaciones existentes.

Por lo que es importante que el agrupamiento de candidatos de operación propuestos por los candidatos de servicio sean reevaluados y comparados con los diseños de servicio de aplicación existentes.

Paso 3.3: Obtención de Interfaz de Servicio Inicial

Para definir la interfaz de servicio inicial se debe analizar la propuesta de servicios de operación candidatos y seguir los siguientes pasos:

- Utilizar el servicio de aplicación candidato como entrada principal, se debe asegurar que la granularidad de las particiones lógicas representadas sean genéricas y reutilizables.
- Documentar la entrada y salida de los valores requeridos para procesar cada operación candidata, y definir estructuras de mensaje utilizando el esquema XSD
- Completar la definición del servicio abstracto, añadiendo el <portType> y el <message> necesario contenido en los elementos <part> que refieren los adecuados tipos de esquema.

Paso 3.4: Implementación de los Principios de Orientación al Servicio

Se deben aplicar los 4 principios de orientación a servicios que son los siguientes:

Reutilización de servicio

- Autonomía de servicio
- Independencia de servicio
- Identificación de servicio

Paso 3.5: Estandarización y Perfección de la Interfaz de Servicio

La siguiente lista son acciones recomendadas para lograr un diseño de servicios estandarizados y simplificados:

- Aplicar los estándares de diseño existentes en relación con la interfaz de servicio.
- Revisar cualquier característica de SOA que se haya elegido para los servicios de apoyo y evaluar si es posible incrementar el apoyo para estas características en el diseño de este servicio.

Paso 3.6: Identificación de Limitaciones Técnicas

Como siguiente paso se deben estudiar y documentar las demandas de procesamiento de cada operación de servicio más a detalle. Para cada operación, se debe escribir una lista de las funciones de procesamiento requeridas por la operación para realizar su procesamiento. Por cada entrada en la lista, encontrar exactamente cómo el procesamiento de la función necesitará ser ejecutada en el entorno técnico existente.

Los tipos de detalle que se deben buscar son los siguientes:

- El punto de conexión física de la función en particular.
- Los problemas de seguridad relacionados con cualquier parte de la transformación.
- El tiempo de respuesta para cada función de procesamiento.
- La disponibilidad de procesamiento del sistema subyacente para procesamiento de la función.
- Los factores ambientales relacionados con la ubicación del servicio de implementación.
- Las limitaciones técnicas de la lógica de la aplicación subyacente (Especialmente cuando se exponen sistemas heredados).
- Los requisitos de la Administración impuesta por el servicio.
- Los posibles requisitos de SLA.

El resultado de este estudio es típicamente una serie de restricciones y limitaciones impuestas por el entorno técnico en nuestra interfaz de servicio. En algunos casos, las restricciones serán tan graves que una operación puede requerir ser incrementada considerablemente.

Paso 4: Diseño de Servicios Empresariales Centrados en la Tarea.

El proceso para el diseño de servicios centrados en la tarea no considera la reutilización como algo primordial. Por lo tanto, sólo los servicios de operación identificados como parte del proceso de modelado de servicios son considerados.

Paso 4.1: Definición de la Lógica del Flujo de Trabajo

Los servicios centrados en la tarea normalmente contienen la lógica de flujo de trabajo integrado para coordinar una composición de servicios subyacentes. El primer paso, es definir esta lógica para cada escenario de interacción posible que se pueda imaginar.

Debido a que se diseña el servicio empresarial centrado en la tarea, después de que los diseños de entidad centrada y los servicios de aplicación se han completado, es necesario volver a examinar estos documentos del escenario y convertirlos en modelos de interacción de servicio.

Paso 4.2: Derivación de la Interfaz de Servicio

Se deben seguir los siguientes pasos para montar una interfaz de servicio:

- Utilizar los servicios de aplicación de operación para obtener un conjunto de operaciones correspondiente.
- Incluir en la interfaz de servicio diagramas de actividad y la lógica de flujo de trabajo que se documenta en el paso anterior.
- El documento de los valores de entrada y salida necesarios para procesamiento de cada operación y llenado de la sección de <types> del esquema XSD requerido para procesar las operaciones.
- Construir la definición WSDL mediante la creación del area portType, insertando el identificador <operation>. Luego se agregan los <message> necesarios que hacen referencia a los tipos de esquema apropiados.

Paso 4.3: Implementación de los Principios de Orientación al Servicio

Se deben aplicar los 4 principios de orientación a servicios que son los siguientes:

- Reutilización de servicio
- Autonomía de servicio
- Independencia de servicio
- Identificación de servicio

Paso 4.4: Estandarización y perfección de la interfaz de servicio

La siguiente lista presenta el estándar de las acciones recomendadas que se pueden tomar para lograr un diseño de servicios estandarizado y simplificado:

- Incorporación de normas de diseño y las directrices existentes.
- Asegurar que cualquier característica de arquitectura elegida es totalmente compatible con el diseño de la interfaz de servicio.

Paso 4.5: Identificación del Procesamiento Requerido

Debido a que éste es el último de los tres procesos de diseño de servicios, son necesarios todos los servicios de apoyo de la aplicación para ser identificados. El diseño de la lógica de proceso puede revelar la necesidad de servicios de aplicaciones adicionales que no han sido consideradas.

Paso 5: Diseño de Procesos de Negocio Orientados a Servicios

Para establecer un inventario de diseños de servicio, se debe crear la capa de orquestación que es la encargada de unir los servicios con la lógica de procesos de negocio. Este es el último paso y genera la definición ejecutable de la lógica de flujo de trabajo[20].

2.1.3. SOA y los servicios web

Es muy importante para ver la posición de SOA como un modelo de arquitectura que es escéptico a cualquier plataforma tecnológica. De esta forma, una empresa tiene la libertad de lograr continuamente los objetivos estratégicos relacionados con la computación orientada a servicios. En el mercado actual, la plataforma tecnológica que más se asocia con la realización de SOA son los servicios web.

Con la computación orientada a servicios viene un modelo de arquitectura distinta que ha sido posicionado por la comunidad de proveedores con la que pueden aprovechar completamente el potencial de interoperabilidad abierta de los servicios web, especialmente cuando los servicios individuales son constantemente formados por la orientación a servicios.

Adicionalmente, el hecho de que los servicios web proporcionan un marco de comunicaciones basadas en contratos físicamente desconectados que permiten que cada contrato de servicio sea normalizado totalmente independiente de la aplicación. Esto

[20] Thomas Erl., Service-Oriented Architecture: Concepts, Technology, and Design Prentice Hall, 2005, pag 449

facilita un alto nivel de potencial de abstracción del servicio, mientras que proporciona la oportunidad de desvincular totalmente el servicio de cualquier propiedad del detalle de implementación[21].

Las principales ventajas de implementar una arquitectura orientada a servicios utilizando servicios web es que son penetrantes, simples, y su plataforma es neutral.

La arquitectura de servicios web básica consiste en las especificaciones (SOAP, WSDL y UDDI) que apoyan la interacción de un solicitante del servicio web con un proveedor y el descubrimiento de la descripción del servicio web. El proveedor publica una descripción WSDL de sus servicios web, y el solicitante tiene acceso a la descripción mediante una UDDI u otro tipo de registro, y pide la ejecución del servicio del proveedor mediante el envío de un mensaje SOAP. La ~~figura 1~~ muestra a detalle la arquitectura de servicios web básica.

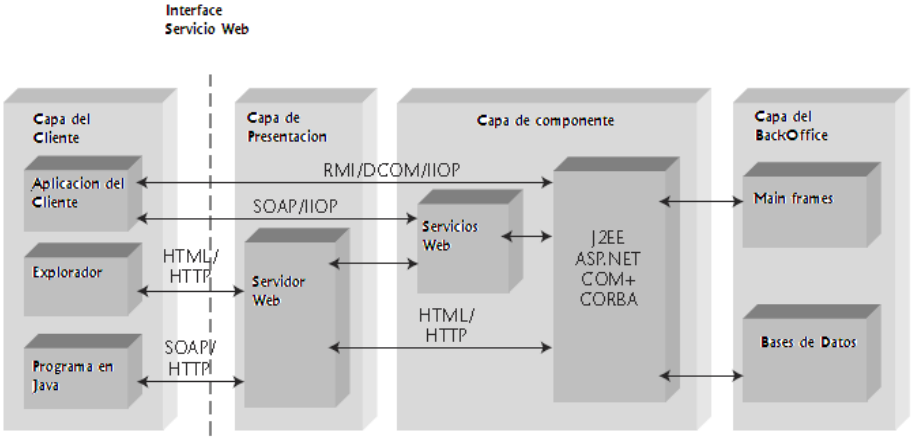


Figura 1: Arquitectura de servicios web básica[22].

La combinación de servicios web y SOA proporciona una solución de integración rápida, que fácil y rápidamente alinean las inversiones en TI y las estrategias empresariales, centrándose en los datos compartidos y servicios reutilizables, en lugar de integración de productos patentados.

---

[21] Thomas Erl., SOA: principles of service design, Prentice Hall, 2008, pág.50

[22] Bret Hartman, Donald J. Flinn, Konstantin Beznosov, Shirley Kawamoto, Mastering Web Services Security, Wiley Publishing Inc., 2003, pág.4

Las Interfaces de servicios web son buenas para permitir acceso multicanal, ya que son accesibles desde una amplia gama de clientes, incluyendo web, Java, C #, dispositivos móviles y otras arquitecturas con servicios web, esto hace que simplifiquen la tarea de las organizaciones para habilitar estos canales de acceso para sus servicios.

2.2. Implementación de Seguridad en SOAP

La seguridad en SOAP puede ser implementada en 3 capas diferentes:

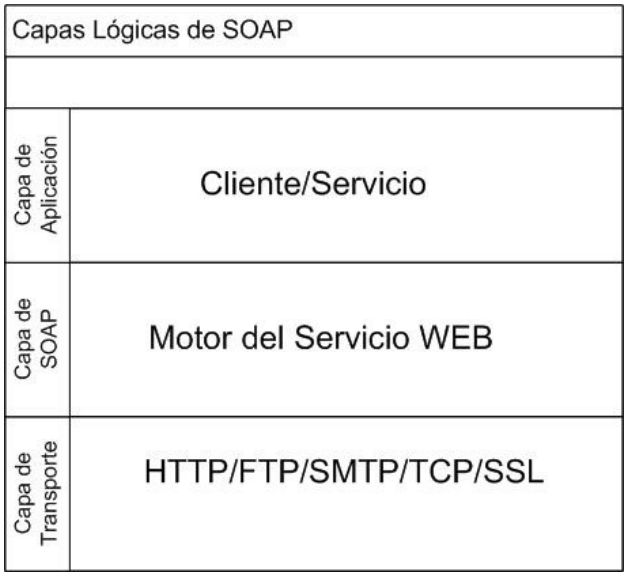


Figura 2: Detalle de las capas lógicas de SOAP [23].

2.2.1. Capa de Transporte

La capa de transporte se refiere a los protocolos TCP, SSL, HTTP, FTP, SMTP y JMS/MQ que permiten cuidar de la seguridad. Aunque se puede implementar seguridad en la capa de transporte tiene las siguientes desventajas:

1. La seguridad está limitada a interacciones punto a punto.
2. Las aplicaciones no pueden obtener el contexto de seguridad como nombre de usuario o rol desde esta capa.

[23] Ramarao Kanneganti, Prasad Chodavarapu, SOA Security, Manning Publications Co., 2008, pág.89.

2.2.2. Capa de Aplicación

La implementación de la seguridad en esta capa permite delegar la seguridad a la aplicación. Sin embargo elegir la implementación de la seguridad en esta capa genera los siguientes inconvenientes:

1. Cada aplicación desarrolla sus propios mecanismos de seguridad generando dificultad para la integración de aplicaciones.
2. La asignación de privilegios es complicada.

2.2.3. Capa de SOAP

Para realizar la implementación de la seguridad en la capa de SOAP, se utiliza un poderoso mecanismo de extensión que permite soportar la seguridad haciendo uso de extensiones.

Sin embargo si la seguridad se delega al uso de extensiones, existe el riesgo de que cada vendedor defina su propia extensión, dañando la interoperabilidad en los procesos. Esto se puede evitar utilizando un estándar para extensiones de seguridad como WS-Security, creado por el grupo de estandarización OASIS. Además de que si el estándar no satisface el escenario que se requiere, existe la posibilidad de crear una extensión de seguridad propia.

2.3. Requisitos de Seguridad de los servicios web

Cuando se pretende realizar el diseño un sistema seguro es necesario identificar los componentes lógicos de seguridad en la información, además de la función que realiza cada uno de ellos y cuando deben ser utilizados.

2.3.1. Confidencialidad

Algunas de las definiciones de este término nos describen que la confidencialidad es el atributo de la información que se utiliza para referirse a los requerimientos de datos en tránsito entre dos partes en comunicación, no siendo disponible a terceros que pueden tratar de espiar a la comunicación.

Los enfoques generales de la confidencialidad son los siguientes:

- Utilizar una conexión privada entre las dos partes, utilizando una VPN o una línea dedicada.

- Aplicar cifrado a los datos que viajan a través de una red insegura, como Internet[24].

2.3.2. Autenticación

La autenticación verifica que los usuarios humanos, las entidades del sistema registradas y los componentes sean quienes dicen ser. El resultado de la autenticación es un conjunto de credenciales, las cuales describen los atributos que pueden ser asociados con quien se ha autenticado[25].

2.3.3. Integridad

La definición de integridad plantea que: "la Integridad es una propiedad de seguridad para garantizar que la información es modificada solo por los sujetos autorizados."[26]

Integridad de la información no quiere decir que la información no puede ser alterada, significa que sí la información es alterada, esta alteración puede ser detectada. La integridad de los datos se basa en algoritmos matemáticos conocidos como algoritmos hash como lo es SHA-1[27].

De lo anterior podemos decir que la Autenticación, Confidencialidad y la integridad tienen la responsabilidad de, proteger los mensajes en tránsito entre los clientes que participan en el envío y recepción de información.

2.3.4. Autorización

La autorización es un requerimiento de seguridad de la información que a veces se vincula con la autenticación sin embargo, es muy importante ver la diferencia entre los dos. La autenticación trata acerca de "¿Quién es usted?" la autorización trata acerca de "¿Qué tiene permitido hacer?" En resumen si un usuario está autenticado, no quiere decir que también deba estar autorizado[28].

---

[24] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003, pág.23

[25] Dirk Krafzig, Karl Banke, Dirk Slama, Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall PTR, 2004, pág.23

[26] Bret Hartman, Donald J. Flinn, Konstantin Beznosov, Shirley Kawamoto, Mastering Web Services Security, Wiley Publishing Inc., 2003, pág.402

[27] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003, pág.27

28 Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003, pág.35

2.3.5. No repudio

Este término resulta un tanto complejo de entender, sin embargo, es muy simple ya que se refiere a la idea de impedir que un participante en una acción niegue de manera fehaciente su responsabilidad en ésta.

El no repudio significa que quien origine un mensaje no puede reclamar que no ha enviado el mensaje. Para lograr el no repudio se debe ligar una llave pública a la parte de la identidad que es la firma digital de los datos[29].

Los elementos a considerar para cumplir el no repudio son certificados digitales y llaves públicas.

2.3.6. Disponibilidad

Uno de los componentes de seguridad más importantes en un sistema de información es la disponibilidad que consiste en la capacidad de la ofrecer datos y servicios cuando se requieran[30].

Existen diferentes formas de denegación de servicio, a continuación se presentan las que considero son las de más incidencia:

- Denegación de servicio: Un ataque de denegación de servicio se propone utilizar todos los recursos de un servicio de forma que no estará disponible para los usuarios legítimos.
- Privacidad: El requisito de la privacidad se refiere a los derechos de privacidad del sujeto de los datos[31].

2.4. Estándares de Seguridad en Servicios Web

Gran parte del trabajo que actualmente está en investigación son una serie de tecnologías para asegurar servicios web basados en XML.

[29] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003, pág.29

[30] Bret Hartman, Donald J. Flinn, Konstantin Beznosov, Shirley Kawamoto, Mastering Web Services Security, Wiley Publishing Inc., 2003, pág.396

[31] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003, pág.36

Estas tecnologías están representadas por sus respectivas especificaciones, siendo desarrollados en varios organismos de normalización casi en paralelo. Sin embargo, todos estos estándares son esfuerzos centrados en obtener un conjunto de especificaciones que puedan ofrecer una solución completa en términos de asegurar los servicios web[32].

La siguiente es una lista de los estándares de seguridad que están basadas en XML, en los cuales se basa la presente investigación:

- WS-Security
- XML Encryption
- XML Signature (XML DSIG)
- Security Assertions Markup Language (SAML)
- XML Key Management Services (XKMS)

2.4.1. WS-Security

La especificación de WS-Security propone un conjunto estándar de extensiones SOAP que pueden ser utilizadas cuando se construyen servicios web seguros para implementar la integridad y la confidencialidad en el contenido del mensaje. WS-Security es flexible y está diseñado para ser utilizado en una amplia variedad de modelos de seguridad como es PKI, Kerberos y SSL, además provee soporte para múltiples formatos de tokens de seguridad, dominios de confianza, formatos de firma, y tecnologías de cifrado.

Ésta especificación provee 3 mecanismos:

1. Envío de tokens de seguridad como parte del mensaje
2. Integridad del mensaje
3. Confidencialidad del mensaje

Estos mecanismos no proveen una solución completa para los servicios web por sí mismos, ésta especificación es un bloque de construcción que puede ser utilizado con otras extensiones para servicios web y protocolos de aplicaciones especificas de alto nivel para acomodar una amplia variedad de modelos de seguridad y tecnologías de seguridad[33].

[32] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003, pag 621
[33] Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), pag 7

2.4.2. XML Encryption

La especificación XML Encryption es actualmente desarrollada por el W3C (http://www.w3.org/standards/techs/xmlenc#w3c_all), XML Encryption es la base de la seguridad de servicios web, tiene por objeto definir el proceso de codificación y decodificación del contenido digital. Su nombre es debido a que utiliza la sintaxis XML para la representación del contenido que se ha cifrado, así como para la representación de la información que permite al destinatario del contenido cifrado poder descifrarlo. XML Encryption no habla de otras cuestiones de seguridad, como autenticación, autorización, integridad, o la confianza. La norma está completamente centrada en la prestación de confidencialidad de la información que ha sido cifrada. Esta especificación define el proceso para cifrado de datos representando el resultado en XML, donde los datos pueden ser elementos XML, Documentos XML o cualquier contenido XML, y el resultado del cifrado de datos es una cifrado XML contenido en el elemento EncryptedData, este elemento tiene la siguiente estructura:

```
<EncryptedData Id? Type? MimeType? Encoding?>
    <EncryptionMethod/>?
    <ds:KeyInfo>
      <EncryptedKey>?
      <AgreementMethod>?
      <ds:KeyName>?
      <ds:RetrievalMethod>?
      <ds:*>?
    </ds:KeyInfo>?
    <CipherData>
      <CipherValue>?
      <CipherReference URI?>?
    </CipherData>
    <EncryptionProperties>?
  </EncryptedData>[34]
```

Los operadores de cardinalidad que se encuentran después de algunas etiquetas indican la ocurrencia que los elementos pueden tener en el documento, la siguiente tabla describe los operadores utilizados:

Operador	Descripción
*	Cero o más ocurrencias
+	Una o más ocurrencias
?	Cero o más ocurrencias

Tabla 2: Operadores de cardinalidad en XML Encryption

[34] http://www.w3.org/TR/xmlenc-core/

Sintaxis

A continuación de describe la sintaxis y características para implementar XML encription, las cuales deben ser implementadas salvo en los casos que se indique lo contrario.

Elemento <EncryptedType>

EncryptedType es un tipo abstracto del que derivan los elementos EncriptedData y EncriptedKey, este elemento está formado por los elementos EncryptionMethod, ds:KeyInfo, CipherData, EncryptionProperties, Id, Type y MimeType, la definición del esquema es la siguiente:

```
Schema Definition:

  <complexType name='EncryptedType' abstract='true'>
    <sequence>
      <element name='EncryptionMethod' type='xenc:EncryptionMethodType'
               minOccurs='0'/>
      <element ref='ds:KeyInfo' minOccurs='0'/>
      <element ref='xenc:CipherData'/>
      <element ref='xenc:EncryptionProperties' minOccurs='0'/>
    </sequence>
    <attribute name='Id' type='ID' use='optional'/>
    <attribute name='Type' type='anyURI' use='optional'/>
    <attribute name='MimeType' type='string' use='optional'/>
    <attribute name='Encoding' type='anyURI' use='optional'/>
   </complexType>
```

[35]

Elemento < EncryptionMethod >

Éste elemento es opcional y describe el algoritmo de cifrado que se aplicará al dato a cifrar, si el elemento no se encuentra el algoritmo de cifrado debe ser conocido por el receptor para poder realizar el decifrado, la definición del esquema es la siguiente:

```
  Schema Definition:

  <complexType name='EncryptionMethodType' mixed='true'>
    <sequence>
      <element name='KeySize' minOccurs='0' type='xenc:KeySizeType'/>
      <element name='OAEPparams' minOccurs='0' type='base64Binary'/>
      <any namespace='##other' minOccurs='0' maxOccurs='unbounded'/>
    </sequence>
    <attribute name='Algorithm' type='anyURI' use='required'/>
  </complexType>
```

[35] http://www.w3.org/TR/xmlenc-core/

[36]

Elemento <CipherData>

Éste elemento es mandatorio y provee el dato cifrado. Debe contener la secuencia de octeto cifrado como texto cifrado en base64 del elemento CipherValue, o proveer una referencia a una dirección externa que contenga la secuencia del octeto cifrado utilizando el elemento CipherReference, la definición del esquema es la siguiente:

```
  Schema Definition:

  <element name='CipherData' type='xenc:CipherDataType'/>
  <complexType name='CipherDataType'>
     <choice>
       <element name='CipherValue' type='base64Binary'/>
       <element ref='xenc:CipherReference'/>
     </choice>
   </complexType>
```

[37]

Elemento <CipherReference>

El elemento CipherReference identifica un origen que cuando es procesado obtiene la secuencia del octeto cifrado, la definición del esquema es la siguiente:

```
Schema Definition:

  <element name='CipherReference' type='xenc:CipherReferenceType'/>
   <complexType name='CipherReferenceType'>
       <sequence>
         <element name='Transforms' type='xenc:TransformsType'
minOccurs='0'/>
       </sequence>
       <attribute name='URI' type='anyURI' use='required'/>
   </complexType>

    <complexType name='TransformsType'>
       <sequence>
         <element ref='ds:Transform' maxOccurs='unbounded'/>
       </sequence>
     </complexType>
```

[38]

Elemento <EncryptedData>

[36] http://www.w3.org/TR/xmlenc-core/
[37] http://www.w3.org/TR/xmlenc-core/
[38] http://www.w3.org/TR/xmlenc-core/

Éste elemento es básico en la sintaxis, contiene elementos CipherData que contienen los datos cifrados y remplaza los elementos cifrados o sirve como elemento raíz para un nuevo documento, la definición del esquema es la siguiente:

```
Schema Definition:

<element name='EncryptedData' type='xenc:EncryptedDataType'/>
<complexType name='EncryptedDataType'>
  <complexContent>
   <extension base='xenc:EncryptedType'>
   </extension>
  </complexContent>
</complexType>
```

[39]

Extensiones del elemento <ds:KeyInfo>

Existen 3 formas que pueden proveer el material de las claves para descifrar el elemento CipherData, para ello los elementos EncryptedData o EncripteKey especifican la asociación del material de las claves utilizando el Elemento hijo ds:KeyInfo.

1. Utilizar ds:KeyInfo para transportar llaves públicas (No recomendado)
2. Utilizar ds:KeyInfo para referirse a elementos EncriptedKey CarriedKeyName (Recomendado)
3. Utilizar para el mismo documento ds:RetrievalMethod (Requerido)

Elemento <EncriptedKey>

El elemento EncryptedKey transporta las llaves de inscripción desde el creador del mensaje a los receptores. Puede aparecer dentro de un elemento EncriptedData como un elemento ds:KeyInfo. El valor de la clave siempre esta encriptado para el receptor, cuando EncriptedKey es descifrado los octetos resultantes se ponen a disposición del algoritmo EncriptyonMethod, la definición del esquema es la siguiente:

```
Schema Definition:

<element name='EncryptedKey' type='xenc:EncryptedKeyType'/>
<complexType name='EncryptedKeyType'>
  <complexContent>
    <extension base='xenc:EncryptedType'>
      <sequence>
        <element ref='xenc:ReferenceList' minOccurs='0'/>
        <element name='CarriedKeyName' type='string' minOccurs='0'/>
```

[39] http://www.w3.org/TR/xmlenc-core/

```
        </sequence>
        <attribute name='Recipient' type='string' use='optional'/>
      </extension>
    </complexContent>
  </complexType>
```
[40]

Elemento <ds:RetrievalMethod>

Éste elemento proporciona una forma de expresar una conexión al elemento EncryptedKey, que contiene la clave necesaria para descifrar el elemento CipherData asociado con el elemento EncryptedData o EncriptedKey, la definición del esquema es la siguiente:

```
  Schema Definition:

  <!--
      <attribute name='Type' type='anyURI' use='optional'
      fixed='http://www.w3.org/2001/04/xmlenc#EncryptedKey' />
  -->
```
[41]

Elemento <ReferenceList>

Éste elemento contiene referencias de un valor clave del EncriptedKey a elementos cifrados por ese valor clave, la definición del esquema es la siguiente:

```
Schema Definition:

  <element name='ReferenceList'>
    <complexType>
      <choice minOccurs='1' maxOccurs='unbounded'>
        <element name='DataReference' type='xenc:ReferenceType'/>
        <element name='KeyReference' type='xenc:ReferenceType'/>
      </choice>
    </complexType>
  </element>

  <complexType name='ReferenceType'>
    <sequence>
      <any namespace='##other' minOccurs='0' maxOccurs='unbounded'/>
    </sequence>
    <attribute name='URI' type='anyURI' use='required'/>
  </complexType>
```
[42]

[40] http://www.w3.org/TR/xmlenc-core/
[41] http://www.w3.org/TR/xmlenc-core/
[42] http://www.w3.org/TR/xmlenc-core/

Elemento <EncryptionProperties>

Éste elemento permite colocar elementos relacionados a la creación de los elementos EncryptedData o EncryptedKey como pueden ser Fechas, el número serial del hardware utilizado durante el cifrado. El elemento Target identifica la estructura EncryptedType que se describe. El elemento anyAttribute permite la inclusión de atributos para el namespace XML que se incluirá, la definición del esquema es la siguiente:

```
  Schema Definition:

  <element name='EncryptionProperties'
type='xenc:EncryptionPropertiesType'/>
  <complexType name='EncryptionPropertiesType'>
    <sequence>
      <element ref='xenc:EncryptionProperty' maxOccurs='unbounded'/>
    </sequence>
    <attribute name='Id' type='ID' use='optional'/>
  </complexType>

    <element name='EncryptionProperty'
type='xenc:EncryptionPropertyType'/>
    <complexType name='EncryptionPropertyType' mixed='true'>
      <choice maxOccurs='unbounded'>
        <any namespace='##other' processContents='lax'/>
      </choice>
      <attribute name='Target' type='anyURI' use='optional'/>
      <attribute name='Id' type='ID' use='optional'/>
      <anyAttribute namespace="http://www.w3.org/XML/1998/namespace"/>
    </complexType>
```

[43]

2.4.3. XML Signature (XML DSIG)

La especificación XML Signature, ofrece un mecanismo para la aplicación de firmas digitales a los documentos XML. El objetivo detrás de utilizar XML en la firma digital es proporcionar fuerte integridad para autenticación del mensaje, la autenticación de firma y el no repudio de los servicios de datos de cualquier tipo, no importa si estos datos se encuentran dentro del documento XML que lleva la firma digital o se encuentran en otro lugar. La especificación XML está finalizada y fue desarrollada en el W3C www.w3.org/Signature.

Elementos de la Estructura XML Signature

La estructura XML Signature es formada por elementos que son etiquetas XML, su estructura define la relación padre-hijo para cada elemento.

[43] http://www.w3.org/TR/xmlenc-core/

```
<Signature ID?>
     <SignedInfo>
       <CanonicalizationMethod/>
       <SignatureMethod/>
       (<Reference URI? >
         (<Transforms>)?
         <DigestMethod>
         <DigestValue>
       </Reference>)+
     </SignedInfo>
     <SignatureValue>
    (<KeyInfo>)?
    (<Object ID?>)*
   </Signature>
```

Los operadores de cardinalidad que se encuentran después de algunas etiquetas indican la ocurrencia que los elementos pueden tener en el documento, la siguiente tabla describe los operadores utilizados:

Operador	Descripción
*	Cero o más ocurrencias
+	Una o más ocurrencias
?	Cero o más ocurrencias

Tabla 3: Operadores de cardinalidad en XML Signature

Sintaxis

A continuación de describe la sintaxis y características para implementar XML Signature, las cuales deben ser implementadas salvo en los casos que se indique lo contrario.

Elemento <Signature>

El elemento <Signature> es el elemento raíz en una firma XML, este elemento es representado por la siguiente estructura:

```
<element name="Signature" type="ds:SignatureType"/>
   <complexType name="SignatureType">
     <sequence>
       <element ref="ds:SignedInfo"/>
       <element ref="ds:SignatureValue"/>
       <element ref="ds:KeyInfo" minOccurs="0"/>
       <element ref="ds:Object" minOccurs="0" maxOccurs="unbounded"/>
     </sequence>
     <attribute name="Id" type="ID" use="optional"/>
   </complexType>
```

44

[44] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003, pag 650

Elemento <SignedInfo>

La estructura de SignedInfo incluye el algoritmo de canonización, un algoritmo de firma y uno o más referencias. El elemento SignedInfo puede contener un atributo ID opcional que le permitirá hacer referencia a otras firmas y objetos.

```
   <element name="SignedInfo" type="ds:SignedInfoType"/>
   <complexType name="SignedInfoType">
     <sequence>
       <element ref="ds:CanonicalizationMethod"/>
       <element ref="ds:SignatureMethod"/>
       <element ref="ds:Reference" maxOccurs="unbounded"/>
     </sequence>
     <attribute name="Id" type="ID" use="optional"/>
   </complexType>
```

[45]

Elemento <SignatureValue>

El elemento SignatureValue contiene el valor real de la firma digital, la firma siempre se codifica utilizando base64.

```
<element name="SignatureValue" type="ds:SignatureValueType"/>
   <complexType name="SignatureValueType">
     <simpleContent>
       <extension base="base64Binary">
         <attribute name="Id" type="ID" use="optional"/>
       </extension>
     </simpleContent>
   </complexType>
```

[46]

Elemento <CanonicalizationMethod>

El elemento CanonicalizationMethod es requerido, específica el algoritmo de canonización que se aplicará al elemento SignedInfo antes de realizar los cálculos de la firma.

```
<element name="CanonicalizationMethod" type="ds:CanonicalizationMethodType"/>
   <complexType name="CanonicalizationMethodType" mixed="true">
     <sequence>
       <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
       <!-- (0,unbounded) elements from (1,1) namespace -->
     </sequence>
     <attribute name="Algorithm" type="anyURI" use="required"/>
   </complexType>
```

[47]

Elemento SignatureMethod

[45] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
[46] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
[47] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

El elemento SignatureMethod es requerido y especifica el algoritmo utilizado para generar y validar la firma. El algoritmo identifica todas las funciones criptográficas que participan en la operación de la firma.

```
<element name="SignatureMethod" type="ds:SignatureMethodType"/>
   <complexType name="SignatureMethodType" mixed="true">
     <sequence>
       <element name="HMACOutputLength" minOccurs="0"
type="ds:HMACOutputLengthType"/>
       <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
       <!-- (0,unbounded) elements from (1,1) external namespace -->
      </sequence>
    <attribute name="Algorithm" type="anyURI" use="required"/>
   </complexType>
```

[48]

Elemento <Reference>

El elemento Reference puede ocurrir una o más veces, especifica el algoritmo de resumen y el valor a procesar, opcionalmente un identificador del objeto que se firmo, el tipo de objeto y una lista de Trasnforms que deben aplicarse antes de procesar. El identificador URI y Transforms describen como el contenido procesado se ha creado.

```
<element name="Reference" type="ds:ReferenceType"/>
   <complexType name="ReferenceType">
     <sequence>
       <element ref="ds:Transforms" minOccurs="0"/>
       <element ref="ds:DigestMethod"/>
       <element ref="ds:DigestValue"/>
     </sequence>
     <attribute name="Id" type="ID" use="optional"/>
     <attribute name="URI" type="anyURI" use="optional"/>
     <attribute name="Type" type="anyURI" use="optional"/>
   </complexType>
```

[49]

Elemento <Transforms>

El elemento opcional Transforms contiene una lista ordenada de elementos Transforms, los cuales describen cómo el firmante obtuvo el objeto de datos al que se digirió. La salida de cada Transforms sirve como entrada al siguiente Transforms. La entrada al primer Transforms es el resultado de eliminar las referencias al atributo del elemento URI y. La salida de los últimos Transforms es la entrada para el algoritmo DigestMethod.

```
<element name="Transforms" type="ds:TransformsType"/>
   <complexType name="TransformsType">
```

[48] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

[49] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

```
      <sequence>
        <element ref="ds:Transform" maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <element name="Transform" type="ds:TransformType"/>
    <complexType name="TransformType" mixed="true">
      <choice minOccurs="0" maxOccurs="unbounded">
        <any namespace="##other" processContents="lax"/>
        <!-- (1,1) elements from (0,unbounded) namespaces -->
        <element name="XPath" type="string"/>
      </choice>
      <attribute name="Algorithm" type="anyURI" use="required"/>
    </complexType>
```
[50]

Elemento < DigestMethod>

El elemento DigestMethod es un elemento necesario que identifica el algoritmo de resumen aplicado al objeto firmado.

```
<element name="DigestMethod" type="ds:DigestMethodType"/>
   <complexType name="DigestMethodType" mixed="true">
     <sequence>
       <any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
     </sequence>
     <attribute name="Algorithm" type="anyURI" use="required"/>
   </complexType>
```
[51]

Elemento <DigestValue>

El elemento DigestValue contiene el valor codificado de la digestión. El resumen es siempre codificado usando base64.

```
<element name="DigestValue" type="ds:DigestValueType"/>
   <simpleType name="DigestValueType">
     <restriction base="base64Binary"/>
   </simpleType>
```
[52]

[50] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
[51] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
[52] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

Elemento <KeyInfo>

El elemento KeyInfo es opcional y permite al destinatario obtener la clave necesaria para validar la firma. El elemento KeyInfo puede contener claves, nombres, certificados, además de información pública de gestión de claves.

```
<element name="KeyInfo" type="ds:KeyInfoType"/>
   <complexType name="KeyInfoType" mixed="true">
     <choice maxOccurs="unbounded">
       <element ref="ds:KeyName"/>
       <element ref="ds:KeyValue"/>
       <element ref="ds:RetrievalMethod"/>
       <element ref="ds:X509Data"/>
       <element ref="ds:PGPData"/>
       <element ref="ds:SPKIData"/>
       <element ref="ds:MgmtData"/>
       <any processContents="lax" namespace="##other"/>
       <!-- (1,1) elements from (0,unbounded) namespaces -->
     </choice>
     <attribute name="Id" type="ID" use="optional"/>
   </complexType>
```
[53]

Elemento <KeyVAlue>

El elemento KeyValue contiene una llave pública que puede ser utilizada para la validación de la firma.

```
<element name="KeyValue" type="ds:KeyValueType"/>
   <complexType name="KeyValueType" mixed="true">
    <choice>
      <element ref="ds:DSAKeyValue"/>
      <element ref="ds:RSAKeyValue"/>
      <any namespace="##other" processContents="lax"/>
    </choice>
   </complexType>
```
[54]

Elemento <X509Data>

Un elemento X509Data en el elemento KeyInfo contiene uno o más identificadores de claves y certificados X509, el contenido del elemento X509Data debe ser al menos un elemento: X509IssuerSerial, X509SubjectName, X509SKI, X509Certificate o X509CRL.

```
<element name="X509Data" type="ds:X509DataType"/>
   <complexType name="X509DataType">
     <sequence maxOccurs="unbounded">
       <choice>
         <element name="X509IssuerSerial" type="ds:X509IssuerSerialType"/>
         <element name="X509SKI" type="base64Binary"/>
```

[53] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

[54] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

```
        <element name="X509SubjectName" type="string"/>
        <element name="X509Certificate" type="base64Binary"/>
        <element name="X509CRL" type="base64Binary"/>
        <any namespace="##other" processContents="lax"/>
      </choice>
    </sequence>
  </complexType>

  <complexType name="X509IssuerSerialType">
    <sequence>
      <element name="X509IssuerName" type="string"/>
      <element name="X509SerialNumber" type="integer"/>
    </sequence>
  </complexType>
```
55

Elemento <Object>

El elemento Object es un elemento opcional que se puede utilizar una o más veces, cuando se presenta este elemento puede contener cualquier dato.

```
<element name="Object" type="ds:ObjectType"/>
  <complexType name="ObjectType" mixed="true">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <any namespace="##any" processContents="lax"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
    <attribute name="MimeType" type="string" use="optional"/>
    <attribute name="Encoding" type="anyURI" use="optional"/>
  </complexType>
```
56

2.4.4. Security Assertions Markup Language (SAML)

Para lograr la especificación de SAML se fusionaron los estándares de seguridad Security Services Markup Language (S2ML) y Authentication Markup Language (AuthML), su finalidad era lograr que los sistemas de seguridad diferentes se comunicaran entre sí de manera significativa. Esta fusión tuvo lugar cuando ambas especificaciones se presentaron a Organization for the Advancement of Structured Information Standards (OASIS) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security[58].

SAML define un estándar para representar la autenticación, atributos y autorización de información que pueden ser usadas en un ambiente distribuido por aplicaciones diferentes, todos los servicios de seguridad que cumplan con la especificación SAML, son

[55] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
[56] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/
[57] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003 pag 685
[58] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003 pag 685

capaces de interpretar los datos enviados desde un servicio de seguridad a otro servicio de seguridad.

El funcionamiento de SAML consiste en prever que los servicios de terceros funcionaran como recursos de confianza para afirmar la corrección de una autenticación, autorización o atributo de actividad y regresar a la parte solicitante las aserciones SAML relacionadas con esa actividad

eXtensible Access Control Markup Language (XACML)

XACML trabaja en el marco del consorcio OASIS (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml), y tiene el privilegio para desarrollar especificaciones de control de acceso basadas en XML. XACML es un lenguaje de control de acceso de propósito general basado en XML, pero no limitado para funcionar sólo con SAML. Con este fin, el comité técnico XACML ha especificado su lenguaje de modo que este aislado del entorno de la aplicación por una representación canónica de las entradas y salidas. La especificación XACML llama a esta capa de aislamiento de contexto XACML. La especificación de los estados que la metodología actual para convertir un entorno específico de aplicación a XACML está fuera de alcance, pero sugiere que una forma automatizada es utilizar Extensible Stylesheet Language Transformations (XSLT).

Modelos Básicos de Control de Acceso

Algunos aspectos adicionales que pueden orientar a los interesados en el tema, son las siguientes definiciones de autenticación. Existen 2 modelos básicos de control de acceso, access control lists ACL y Role-Based Access Control (RBAC).

Access Control Lists (ACL)

En el modelo de acceso ACL cada usuario tiene permisos separados para un conjunto particular de registros, incluyendo lectura, escritura, ejecución y cambio de permisos. Son ejemplos de ACL los sistemas operativos Linux y Windows.

Role-Based Access Control

El modelo RBAC utiliza el concepto de usuarios en una cuenta, los usuarios pueden ser incluidos en grupos y asignados en roles, los roles pueden tener una relación con la estructura o también los roles pueden heredar las propiedades de otros roles.

Capitulo 3: Diseño Conceptual y Físico

Después de haber realizado una revisión teórica de las ventajas y desventajas de los servicios web, el proceso de diseño de SOA, los componentes lógicos de seguridad en la información y los estándares de seguridad en servicios web basados en XML, se procede a realizar la selección de alternativa, el diseño conceptual y físico para la implementación de controles de seguridad en los servicios web, misma que tomará como referencia la teoría descrita en el capítulo anterior.

3.1. Caso Práctico 1

Como caso práctico se evaluó la implementación de la solución de la presente tesis en la plantilla de captura de calificaciones del IPN.

Para evaluar la implementación se consideraron los siguientes componentes:

- Servidor de aplicaciones a utilizar: El servidor de aplicaciones utilizado es GlassFish.
- Sistema Operativo: Los sistemas operativos utilizados son Ubuntu y Windows Server 2007
- Motor de Base de Datos: El manejador de Base de datos utilizado es SQL Server
- Lenguaje de Programación: El lenguaje de programación para desarrollar la aplicación es JAVA 1.6, utilizando Netbeans 6.8 y Echo 2 para desarrollo en AJAX.

La aplicación en donde se considera utilizar la plantilla de captura de calificaciones maneja GlassFish como servidor de aplicaciones, utiliza interfaces graficas dinámicas para realizar consultas y capturas a usuarios registrados. Se detectó que esta aplicación no requiere el uso o publicación de servicios web como parte de sus requerimientos de uso.
Debido a que la implementación de la presente tesis se centra en el aseguramiento de los servicios web y esta aplicación no los utiliza, se determinó no se cumple con el escenario que se requiere para poder realizar la implementación.

3.2. Caso Práctico 2

Para el caso práctico 2 se plantea un caso hipotético por lo que es necesario mencionar que la finalidad es únicamente demostrar la implementación de controles de seguridad en servicios web, en consecuencia la solución no se implementara en un ambiente productivo.

Una empresa presta sus servicios a través de su portal en Internet, la empresa requiere incrementar las opciones para recibir los pagos de los servicios que presta a sus clientes, por lo que además de la opción de recepción de pagos en línea con tarjetas de crédito, requiere poder recibir el pago de los servicios utilizando las ventanillas de cobro de instituciones bancarias.

La empresa requiere que las instituciones bancarias le envíen para cada transacción realizada la siguiente información:

- Banco donde se realiza la transacción
- Sucursal donde se realizó el pago
- Cuenta de depósito
- Forma en la que se realizó el pago
- Referencia del movimiento
- Importe del pago
- Fecha de pago
- Fecha de aplicación del abono a la cuenta
- Número de autorización

Descripción del Procedimiento

Procedimiento: Cobro de servicios en ventanilla bancaria.

Objetivo

Incrementar las opciones de pago para los clientes, utilizando las ventanillas de cobro de instituciones bancarias.

Alcance

Aplica ventanillas de instituciones bancarias que tengan convenio con la empresa.

Responsabilidades

Del cliente:

- Imprimir la forma para pago de servicios en la página web de la empresa.
- Pagar el costo correspondiente al servicio en la ventanilla del banco.

De la empresa:

- Prestar el servicio al cliente de forma inmediata, una vez que el cliente haya realizado el pago correspondiente.
- Recibir la información de los pagos realizados en las ventanillas bancarias de forma inmediata.

Del Banco:

- Recibir el pago de los clientes que presenten la forma de pago de la empresa en sus ventanillas.
- Enviar a la empresa la información de los pagos recibidos en las ventanillas en el momento en el que se recibe cada pago.

Definiciones

Cliente: Persona o empresa que solicita un servicio.

Banco: Intermediario financiero encargado de realizar la recepción de pagos de servicios.

Insumos

Necesidad de realizar el pago de servicios utilizando ventanillas de instituciones bancarias.

Resultados

Prestación de servicios al cliente de forma inmediata de los pagos realizados en ventanilla bancaria.

Desarrollo

La siguiente tabla ilustra de forma detallada las actividades a realizar para el procedimiento:

No.	Responsable	Descripción de la actividad
1	Cliente	Requiere un servicio e Imprime la forma para pago en ventanilla con importe y referencia en el portal de Internet de la empresa.
2	Cliente	Realiza el pago en una ventanilla bancaria de los bancos afiliados.
3	Banco	Recibe el pago del cliente y envía la información de la transacción a la empresa de forma inmediata.
4	Empresa	Recibe la información del pago enviada por el banco y presta el servicio al cliente.

Tabla 4: Descripción de Actividades

Diagramación

EL siguiente diagrama ilustra el flujo detallado definido para el proceso de cobro de servicios en ventanillas bancarias:

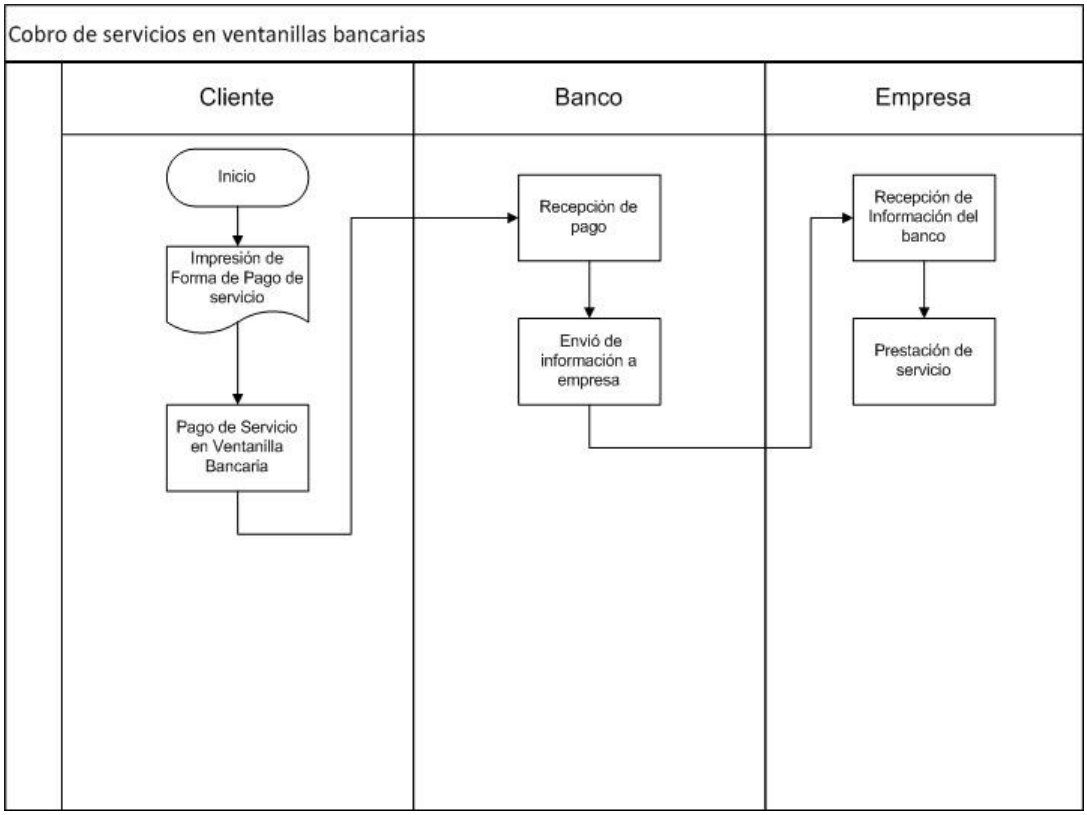


Figura 3: Diagrama del proceso de cobro

3.3. Propuesta de Solución

Para el problema descrito en el caso práctico 2, se propone hacer uso de servicios web y clientes proxy SOAP, para facilitar y reducir el tiempo de la implementación entre la empresa y los bancos.

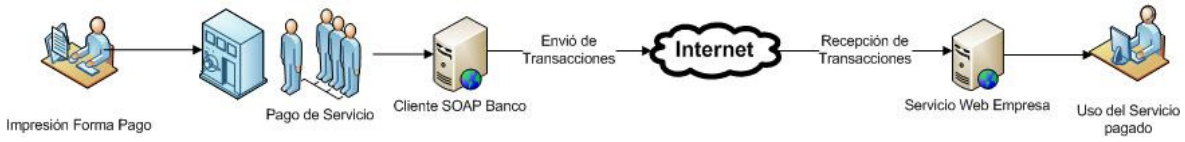


Figura 4: Propuesta de solución

Como se puede apreciar en la figura 4, el envío y recepción de la información de las transacciones hará uso del Internet como medio de transmisión dejando expuesta la

información, además de que existe el riesgo de que el servicio web sea utilizado por terceros.

Para dar solución a los problemas de seguridad identificados, se implementarán los siguientes componentes de seguridad:

- Confidencialidad
- Autenticación
- Integridad
- Autorización
- No repudio

Es necesario mencionar que la elección de los componentes de seguridad depende directamente del problema que se desea resolver y no siempre se requiere la implementación de todos los componentes.

Una vez definidos los componentes de seguridad a utilizar, se realizará su implementación utilizando el mecanismo de seguridad SAML Sender Vouches with Certificates, éste mecanismo cubre las necesidades de seguridad planteadas, ya que se utiliza para agregar seguridad en las comunicaciones entre sistemas, protege los mensajes para integridad y confidencialidad con certificados y usa tokens SAML para resolver el problema de la autorización y autenticación.

La siguiente tabla contiene los estándares XML a utilizar para la implementación del mecanismo SAML Sender Vouches with Certificates:

Componente	Estándar	Fase
Servicio web y cliente proxy	JAX-WS	Definición
Políticas de seguridad	WS-SecurityPolicy	Definición
Capa de SOAP	WS-Security	Funcionamiento
Integridad	XML Signature	Funcionamiento
No repudio	XML Signature	Funcionamiento
Confidencialidad	XML Encryption	Funcionamiento
Autenticación	SAML	Funcionamiento
Autorización	SAML	Funcionamiento

Tabla 5: Definición de estándares por componente y fase

3.4. Diseño Conceptual

Para realizar el diseño conceptual, se empleará el lenguaje de modelado UML (Lenguaje Unificado de Modelado), el cual se describe a continuación de manera muy breve.

El lenguaje UML utiliza varios diagramas para representar y organizar procesos de forma gráfica. UML utiliza reglas para combinar los elementos gráficos dependiendo del tipo de diagrama, a continuación se describen los tipos de diagramas que se utilizaran en el modelado del diseño conceptual.

Diagramas de Casos de Uso

Un diagrama de caso de uso describe las acciones de un sistema o un proceso mediante una representación grafica, los diagramas de caso de uso utilizan los siguientes elementos:

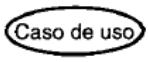
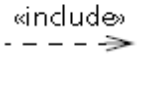
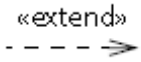
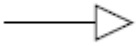
Elemento	Descripción	Símbolo
Caso de Uso	Un caso de uso describe un conjunto de secuencias en el cual cada secuencia representa la interacción de acciones.	
Actor	Un actor representa un conjunto de roles que juegan los usuarios del caso de uso, los actores pueden ser humanos o sistemas.	
Asociación	La asociación representa la comunicación entre el actor y el caso de uso.	
Inclusión	La inclusión indica que un caso de uso puede incluir a otro.	 b
Extensión	La extensión indica que la funcionalidad de un caso de uso puede ser extendida a otro caso de uso.	
Generalización	La generalización se utiliza para establecer una relación entre un caso de uso general a uno más específico que hereda y añade características	

Tabla 6: Elementos de diagramación casos de uso en UML

3.4.1. Definición del Servicio

El propósito del servicio web de pagos es incrementar las opciones de pago para los clientes, proporcionando un mecanismo que permita el registro inmediato de las transacciones bancarias que se originan por el pago de servicios de los clientes en las ventanillas bancarias.

El servicio web requiere recibir los siguientes datos:

Nombre	Tipo de dato	Descripción
Banco	Carácter (25)	Identificación del Banco (eje: Banamex, HSBC, Bancomer)
Sucursal	Carácter (30)	Número de sucursal donde se efectuó el pago
Cuenta	Carácter (30)	Número de cuenta donde se realizó el depósito
Forma de pago	Carácter (30)	Forma en la que el cliente realizo el pago (eje: Efectivo, Cheque, Tarjeta de crédito o debito)
Referencia	Carácter (30)	Identificador del servicios
Importe	Numérico	Importe pagado sin puntos ni comas con 2 decimales (eje: 198500)
Fecha de Pago	Carácter (8)	Fecha en la que se realizó el pago (Formato: ddmmaaaa)
Fecha de Aplicación	Carácter (8)	Fecha en la que el banco aplica el depósito (Formato: ddmmaaaa)
Autorización	Carácter (20)	Número que asigna el banco para autorización de la transacción.

Tabla 7: Datos requeridos para registro de pagos

Requerimientos Funcionales

Además del registro de datos el servicio web debe cumplir los siguientes requisitos de funcionalidad:

- Seguridad: Una parte indispensable para el servicio web es garantizar la seguridad de la información de sus transacciones y en el acceso al mismo. Por lo que debe implementar componentes de seguridad para cumplir este requisito.
- Estándares: Para facilitar la integración y reducir el tiempo de implementación del servicio web se debe hacer uso de estándares XML durante la construcción del servicio.

Criterios de Éxito

- La empresa debe recibir la información del pago realizado en la ventanilla bancaria en tiempo real.
- Se debe garantizar la seguridad, integridad y no repudio de la información de las transacciones.
- El acceso a los servicios web debe ser restringido y disponible solamente a sistemas autorizados y autenticados.

Actores

Nombre	Descripción
Banco	Envía la información de los pagos realizados en las ventanillas bancarias.
Empresa	Registra la información de los pagos enviada por los bancos.

Tabla 8: Actores identificados

Casos de uso

Identificador	Envió de transacción	
Descripción	El sistema deberá permitir al banco y a la empresa al momento de recibir el pago en la ventanilla del banco realizar el envió de la información de la transacción a la empresa para su registro según se describe en el siguiente caso de uso:	
Secuencia Normal	Paso	Acción
	1	Recepción del pago en la ventanilla bancaria
	2	Envió de información de la transacción de la empresa
	3	Registro de la transacción y confirmación de recepción al banco.
Excepciones	Paso	Acción
	1	Envió de mensaje en caso de registrarse cualquier error.
Comentarios	Ninguno	

Tabla 9: Caso de uso envió de transacción

El siguiente diagrama muestra la interacción de los actores para el caso de uso envió de transacción.

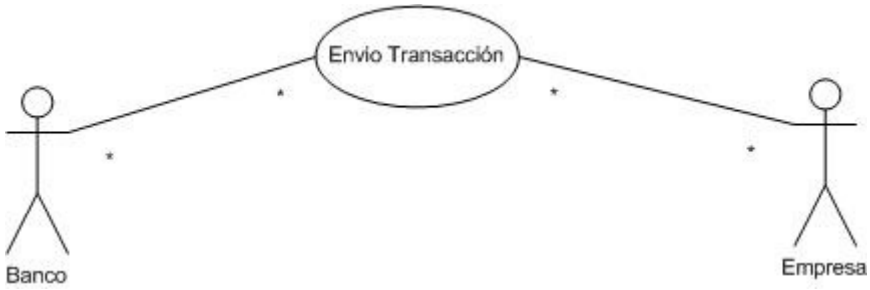


Figura 5: Diagrama de caso de uso envió de transacción

3.4.2. Servicios web

Después de analizar los requerimientos de funcionalidad se detectaron los siguientes servicios web:

Servicio de Pagos: Mediante este servicio se debe recibir la información de los pagos enviada por los bancos.

Clasificación de los Servicios

Para poder conocer los servicios web de la empresa a continuación se realiza el inventario de servicios, este inventario ayudará a identificar los servicios existentes y facilitar el reúso.

Servicio	Tipo	Descripción
WsPagos	Servicio de tareas	Este servicio interactuara con otros sistemas y servicios.

Tabla 10: Servicios web requeridos

Modelo de Funcionalidad Detallado

Las siguientes funciones están definidas para el servicio web WsPagos:

Nombre	Descripción
CrearTransaccionPago	Esta función recibe la información del pago realizado en la ventanilla bancaria y regresa un tipo de dato String.

Tabla 11: Funciones del servicio web WsPagos

Detalle de las funciones

Nombre:	CrearTransaccionPago
Descripción:	Esta función recibe la información del pago realizado en la ventanilla bancaria.
Pre-condiciones:	La datos de entrada deben de enviarse en el formato establecido
Datos de Entrada:	Banco, Sucursal, Cuenta, Forma de pago, Referencia, Importe, Fecha de pago, Fecha de aplicación, Autorización.
Datos de Salida:	La lista de los datos de la transacción registrada.
Post- condiciones:	La transacción del pago es registrada para poder prestar el servicio.
Condiciones de Excepción:	Formato de Importe inválido, Formato de fecha inválido.
Notas:	Ninguna

Tabla 12: Especificación de función CrearTransaccionPago

3.4.3. Clientes SOAP

Para poder hacer uso del servicio web de pagos se debe implementar un cliente proxy que permita al banco consumir el servicio web, se implementaran los siguientes clientes proxy:

Cliente proxy	Servicio que consume	Descripción
WsPagosService	WsPagos	Este cliente proxy envía los datos de los pagos al servicio pagos

Tabla 13: Clientes proxy requeridos

3.5. Diseño Físico

La definición de la infraestructura debe diseñarse considerando las necesidades y requerimientos del problema a resolver. Se deben considerar elementos como son; el requerimiento de procesamiento, licenciamiento, Telecomunicaciones (Red LAN) y Seguridad Lógica.

La infraestructura que se propone cumple con los requisitos básicos de hardware y software para demostrar la implementación de la seguridad en los servicios web. Por lo que si se desea realizar la implementación en un ambiente productivo, se deben analizar los requerimientos, capacidades y tiempos de respuesta esperados, para poder definir correctamente los elementos mencionados anteriormente.

3.5.1. Hardware

Los componentes de hardware que se utilizarán para implementar la solución son los siguientes:

Componente	Características del Hardware
Servidor 1	AMD Athlon X2 7750 Black Edition 2.7 GHz, 4Gb en memoria RAM, 250gb en disco, Tarjeta de Red inalámbrica, Tarjeta de Red alambrica
Servidor 2	Servidor Virtualizado 1 procesador AMD Athlon 7750 Black Edition 2.7 GHz, 1Gb en memoria RAM, 120Gb en disco
Switch	Wireless 11G Router 4-Puertos 10/100Mbps
Equipo Cliente	Intel Core 2 Duo 2.4 GHz, 2Gb en memoria RAM, 150Gb disco

Tabla 14: Componentes de Hardware

Es importante señalar que las características del hardware pueden variar dependiendo de los requerimientos de desempeño de la solución que se requiera, se debe recordar que los requerimientos que aquí se presentan son meramente demostrativos.

3.5.2. Software

Para el desarrollo e implementación de la solución, se utilizara software libre, con la finalidad de reducir los costos en licenciamiento de software, la tabla x contiene el software a utilizar, incluyendo su descripción y el sitio web de donde se puede descargar sin ningún costo.

Software	Descripción
GlassFish 3.1	Servidor de aplicaciones para implementar y ejecutar aplicaciones de las tecnologías definidas en la plataforma Java Enterprise Edition. Link de descarga: http://glassfish.java.net/downloads/3.1-final.html
Virtual box	Software de Vitalización para arquitecturas x86 Link de descarga: http://download.virtualbox.org/virtualbox/4.0.4/VirtualBox-4.0-4.0.4_70112_fedora14-1.i686.rpm
NetBeans 6.9.1	Entorno de Desarrollo Integrado para desarrollo de aplicaciones J2EE. Link de descarga: http://netbeans.org/downloads/
Metro web service stack 2.1	Herramienta para el desarrollo de servicios web que proporciona un stack que permite crear y desplegar servicios web y clientes seguros, confiables, transaccionales e interoperables. Link de descarga: http://download.java.net/maven/2/org/glassfish/metro/metro-standalone/2.1/metro-standalone-2.1.zip
Fedora 14	Sistema Operativo Linux para la plataforma x386 Link de descarga: http://mirrors.fedoraproject.org/publiclist/Fedora/14/
JAVA-JDK 1.6.24	Entorno de desarrollo para crear aplicaciones y componente en lenguaje java. Link de descarga: http://www.oracle.com/technetwork/java/javase/downloads/index.html
OpenSSL	Herramientas para la administración y bibliotecas relacionadas con la criptografía, que se utilizará para la creación de certificados.
Keytool	Herramienta criptográfica para crear y gestionar claves y certificados
Cywin	Colección de herramientas que proporciona una apariencia de Linux para ambientes Windows. Link de descarga: http://cygwin.com/setup.exe
JAVA-JRE	Entorno de ejecución para aplicaciones java. Link de descarga: http://www.oracle.com/technetwork/java/javase/downloads/index.html

Tabla 15: Componentes de Software

3.5.3. Configuración de Equipos

Antes de iniciar el desarrollo, se deben configurar los equipos e instalar el software, esto depende de las funciones que realizará cada equipo. A continuación se detallan los requerimientos software de los equipos a utilizar:

Equipo de Desarrollo

En el equipo que se utilizará para el desarrollo, se deben instalar y configurar los siguientes componentes de software:

- Sistema Operativo Windows XP Service Pack 3.
- NetBeans 6.9.1
- JAVA-JRE
- JAVA-JDK 1.6.24
- Metro web service stack 2.1
- Cywin con OpenSSL
- Keytool

Equipo Servidor 1

Para desplegar y publicar los servicios web es necesario un servidor de aplicaciones que ejecute aplicaciones web utilizando java. Para cubrir este requisito se deben instalar y configurar los siguientes productos de software:

- Sistema Operativo Fedora
- Java Development Kit (JDK)
- Java Runtime Environment (JRE)
- Servidor de aplicaciones Glassfish
- Metro web service stack
- Virtual Box

Nota: Se utilizará VirtualBox para crear una máquina virtual donde se instalará el sistema operativo y los productos de software del servidor 2, lo anterior con la finalidad de reducir gastos en hardware.

Servidor 2

Para alojamiento del Cliente proxy y el aplicativo de captura, es necesario un servidor de aplicaciones que ejecute aplicaciones web utilizando java, para cubrir este requisito se deben instalar y configurar los siguientes productos de software:

- Sistema Operativo Fedora
- Java Development Kit (JDK)
- Java Runtime Environment (JRE)
- Servidor de aplicaciones Glassfish
- Metro web service stack 2.1

Equipo Cliente

Los equipos que funcionarán como clientes, requieren tener instalado un explorador web para utilizar la aplicación del banco, se recomienda cualquiera de los siguientes:

- Internet Explorer 7 o superior
- Mozilla 3.0 o superior
- Safari
- Google Chrome.

3.5.4. Arquitectura de Hardware y Software

El diseño de la arquitectura integrará los elementos de hardware y software descritos anteriormente. La arquitectura que se propone considera que la conectividad de red puede ser en intranet o Internet, para el caso en estudio se realiza utilizando una intranet. La siguiente figura muestra la arquitectura propuesta:

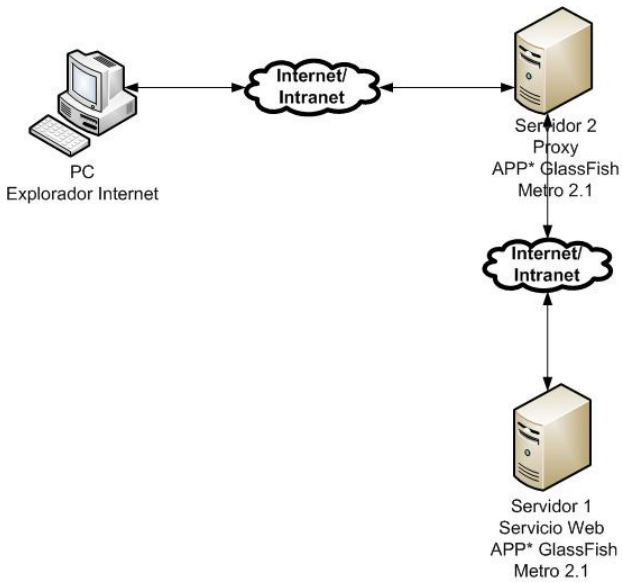


Figura 6: Diagrama de Arquitectura a implementar

3.5.5. Creación de Certificados

Antes de crear los certificados que se usaran para cifrar y firmar los datos, es necesario definir los parámetro a utilizar en el momento de generarlos. Para nuestro caso los valores que se utilizarán son los siguientes:

Estándar:	X509 V3
Algoritmo:	MD5withRSA
Días de validez:	3650
Extensiones:	SubjectKeyIdentifier

Tabla 16: Parámetros requeridos para creación de certificados

La creación de los certificados se realizará en Cywin utilizando Openssl. A continuación se describen los pasos a seguir para la generación de los certificados pem y p12:

Paso 1: Ingresar a la consola de Cywin

Paso 2: Dentro de la consola Cywin, se procede a crear los certificados pem con los siguientes comandos:

```
openssl req -x509 -days 3650 -md5 -newkey rsa:1024 -keyout empresakey.pem -out empresacert.pem -passout pass:changeit
openssl req -x509 -days 3650 -md5 -newkey rsa:1024 -keyout bancokey.pem -out bancocert.pem -passout pass:changeit
```

Paso 3: Después de generar los certificados pem, se crean los certificados p12 con los siguientes comandos:

```
openssl pkcs12 -export -inkey empresakey.pem -in empresacert.pem -out empresa.p12 -name empresa -passin pass:changeit -passout pass:changeit
openssl pkcs12 -export -inkey bancokey.pem -in bancocert.pem -out banco.p12 -name banco -passin pass:changeit -passout pass:changeit
```

3.6. Desarrollo de Componentes

Previo al desarrollo, es necesario instalar y configurar el software en el equipo de desarrollo. Para ver ha detalle del procedimiento de instalación y configuración de los productos de software, refiérase al apéndice 1 y 2 respectivamente.

Una vez que se cuenta con el ambiente de desarrollo en el equipo, se procede al desarrollo de los servicios web y clientes proxy.

Con el propósito de simplificar y agilizar el desarrollo de los servicios web, los clientes proxy y facilitar la administración de las políticas de seguridad, se utilizará el ambiente de desarrollo integrado de Netbeans.

3.6.1. Servicios web

El primer componente que se debe desarrollar es el servicio web de pagos, para la creación del servicio web se requieren seguir los siguientes pasos:

Paso 1: Crear el paquete pagos.

Paso 2: Crear el servicio web WsPagos en el paquete pagos.

Paso 3: Crear la Operación CrearTransaccionPago para el servicio web con los parámetros de entrada y salida definidos previamente.

El código fuente que se genera para el servicio web WsPagos es el siguiente:

```
package Pagos;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
@WebService()
public class WSPagos { @WebMethod(operationName = "CrearTransaccionPago")
    public String Recepcion(@WebParam(name = "Banco")
    String Banco, @WebParam(name = "Sucursal")
    String Sucursal, @WebParam(name = "Cuenta")
    String Cuenta, @WebParam(name = "Formapago")
    String Formapago, @WebParam(name = "Referencia")
    String Referencia, @WebParam(name = "Importe")
    double Importe, @WebParam(name = "FechaPago")
    String FechaPago, @WebParam(name = "FechaAplica")
    String FechaAplica, @WebParam(name = "Autorizacion")
    String Autorizacion) {
//registro de transaccion
    String Resultado=" Transacción Exitosa ";
    return Resultado;
    }
}
```

Configuración de Seguridad

Como se mencionó en la teoría descrita en el capítulo anterior, la ventaja de implementar la seguridad en la capa de SOAP es integrar la seguridad de forma independiente a la lógica de la aplicación. Para implementar la seguridad en los servicios web se usarán estándares XML, para la definición de políticas.

En el stack de servicios web Metro la configuración de la seguridad se almacena en el archivo wsit-Pagos.WSPagos.xml, este archivo contiene las políticas que se aplican para asegurar el servicio web. Las políticas dependen del mecanismo de seguridad que se esté implementando, para el servicio web Pagos se implementa el mecanismo SAML Sender Vouches with Certificates.

A continuación se describen algunas de las políticas definidas en el archivo de configuración wsit-Pagos.WSPagos.xml:

Definición de Elementos a Cifrar y Firmar

Para especificar la política de cifrado y firmado del cuerpo del mensaje, se utilizan los siguientes elementos:

```
<wsp:Policy wsu:Id=" WSPagosPortBinding_CrearTransaccionPago_Input_Policy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:EncryptedParts>
        <sp:Body/>
      </sp:EncryptedParts>
      <sp:SignedParts>
        <sp:Body/>
      </sp:SignedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
<wsp:Policy wsu:Id=" WSPagosPortBinding_CrearTransaccionPago_Output_Policy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:EncryptedParts>
        <sp:Body/>
      </sp:EncryptedParts>
      <sp:SignedParts>
        <sp:Body/>
      </sp:SignedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

- El elemento <sp:Body/> especifica que se firmará o cifrará el cuerpo del mensaje.
- El elemento <sp:EncryptedParts> especifica que se cifrará el cuerpo del mensaje.

- El elemento <sp:SignedParts> especifica que se firmará el cuerpo del mensaje.

Protección de Mensajes

El elemento <sp:AsymmetricBinding> implementa la protección de mensajes SOAP utilizando algoritmos de clave asimétrica en la capa SOAP, para ello utiliza los siguientes elementos para definir la política de seguridad de protección de mensajes:

```
<sp:AsymmetricBinding>
  <wsp:Policy>
    <sp:InitiatorToken>
      <wsp:Policy>
        <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
          <wsp:Policy>
            <sp:WssX509V3Token10/>
            <sp:RequireThumbprintReference/>
          </wsp:Policy>
        </sp:X509Token>
      </wsp:Policy>
    </sp:InitiatorToken>
    <sp:RecipientToken>
      <wsp:Policy>
        <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
          <wsp:Policy>
            <sp:WssX509V3Token10/>
            <sp:RequireThumbprintReference/>
          </wsp:Policy>
        </sp:X509Token>
      </wsp:Policy>
    </sp:RecipientToken>
    <sp:Layout>
      <wsp:Policy>
        <sp:Strict/>
      </wsp:Policy>
    </sp:Layout>
    <sp:IncludeTimestamp/>
    <sp:OnlySignEntireHeadersAndBody/>
    <sp:AlgorithmSuite>
      <wsp:Policy>
        <sp:Basic128/>
```

```
      </wsp:Policy>
    </sp:AlgorithmSuite>
  </wsp:Policy>
</sp:AsymmetricBinding>
```

- El elemento <sp:WssX509V3Token10/> especifica que se utilizará un token X509 V3.
- El elemento <sp:OnlySignEntireHeadersAndBody/> indica que las firmas solo se pueden aplicar al encabezado o al cuerpo del mensaje, no a elementos específicos.
- El elemento <sp:AlgorithmSuite> especifica el algoritmo a utilizar, en este caso <Basic128/>

Autenticación

La política de autenticación establecida utiliza tokens cifrados y encriptados para garantizar la integridad y confidencialidad, la definición la política para autenticación se define utilizando los siguientes elementos:

```
<sp:SignedEncryptedSupportingTokens>
  <wsp:Policy>
    <sp:SamlToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
      <wsp:Policy>
        <sp:WssSamlV11Token10/>
      </wsp:Policy>
    </sp:SamlToken>
  </wsp:Policy>
</sp:SignedEncryptedSupportingTokens>
```

- El elemento <sp:SignedEncryptedSupportingTokens> indica que se requiere el cifrado y encriptado de los tokens.
- El elemento <sp:SamlToken> indica que se utilizará un token SAML.
- El elemento <sp:WssSamlV11Token10/> indica que la versión del token SAML es la 1.1.

3.6.2. Clientes Proxy

Para desarrollar el cliente proxy del banco, es necesario el archivo de definición WSDL del servicio web WsPagos. Los pasos que se requieren seguir para crear el cliente proxy son los siguientes:

Paso 1: Crear un nuevo web service client utilizando el archivo WSDL pagos.

Con ayuda de Netbeans se crea el cliente proxy WSPagosService, con las especificaciones descritas en el Archivo WSDL. Una vez que se tiene el cliente proxy se debe crear la interfaz que permitirá la captura de los pagos en ventanilla. La interfaz utilizará el cliente proxy para enviar la información de las transacciones al servicio web. Para instanciar el cliente proxy se utilizará el siguiente código:

```
try {
        pagos.WSPagosService service = new pagos.WSPagosService();
        pagos.WSPagos port = service.getWSPagosPort();
        java.lang.String result = port.crearTransaccionPago(banco, sucursal, cuenta, formapago, referencia, importe,
                                                            fechaPago, fechaAplica, autorizacion);
        out.println("Result = "+result);
    } catch (Exception ex) {
        out.print(ex.getMessage());
    }
```

Configuración de Seguridad

Además de la definición de las políticas de seguridad, para implementar el mecanismo de seguridad SAML Sender Vouches with Certificates en el cliente proxy, se requiere implementar la clase Saml11SVCallbackHandler para el manejo de las aserciones SAML, esta clase requiere se incluyan al proyecto las librerías xmlsec-2.0.jar y xws-security-3.0.jar. El siguiente fragmento de código muestra el método que se encarga del manejo de las aserciones para SAML 1.1:

```
public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
    for (int i=0; i < callbacks.length; i++) {
        if (callbacks[i] instanceof SAMLCallback) {
            try{
                SAMLCallback samlCallback = (SAMLCallback)callbacks[i];
                samlCallback.setConfirmationMethod(samlCallback.SV_ASSERTION_TYPE);
                if (samlCallback.getConfirmationMethod().equals(samlCallback.SV_ASSERTION_TYPE)){
                    samlCallback.setAssertionElement(createSVSAMLAssertion());
                    svAssertion=samlCallback.getAssertionElement();
                }else{
```

```
                    throw new Exception("SAML Assertion Type is not matched.");
                }
            }catch(Exception ex){
                    ex.printStackTrace();
            }
        } else {
            throw unsupported;
        }
    }
}
```

La configuración de las políticas de seguridad del cliente proxy se guardan en el archivo WSPagosService.xml, este archivo contiene las políticas requeridas para integrar la seguridad del servicio web con el Cliente Proxy. La definición de las políticas de seguridad para el cliente Proxy son las siguientes:

```
<wsp:Policy wsu:Id="WSPagosPortBindingPolicy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sc:KeyStore wspp:visibility="private" location="glassfish/domains/domain1/config/keystore.jks" type="JKS"
            storepass="changeit" alias="banco" keypass="changeit"/>
      <sc:TrustStore wspp:visibility="private" location="glassfish/domains/domain1/config/cacerts.jks" type="JKS"
            storepass="changeit" peeralias="empresa"/>
      <sc:CallbackHandlerConfiguration wspp:visibility="private">
        <sc:CallbackHandler name="samlHandler" classname="samlcb.Saml11SVCallbackHandler"/>
      </sc:CallbackHandlerConfiguration>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

- El elemento sc:KeyStore especifica la ruta en la que se encuentra el almacén de keystore
- El elemento sc:TrustStore especifica la ruta en la que se encuentra el almacén de truststore
- El elemento sc:CallbackHandler especifica el nombre de la clase que se encargará del manejo de las aserciones SAML.

Capitulo 4: Implementación y pruebas

En este capítulo se realizara la instalación y configuración de software y hardware, dando como resultado la arquitectura funcional. Posteriormente se detalla el procedimiento de instalación de los componentes que conforman la solución. Finalmente se presentan los resultados de las pruebas de funcionalidad y seguridad de la aplicación.

4.1. Implementación

Una vez concluido el desarrollo de los servicios web y los clientes proxy, se deben instalar y configurar los productos de software de acuerdo a las especificaciones descritas en el capítulo anterior, para los siguientes componentes:

- Servidor 1
- Servidor 2
- Cliente

El procedimiento de instalación y configuración de los productos de software, se describen a detalle en el apéndice 1 y 2 respectivamente.

4.1.1. Arquitectura

Después de instalar y configurar los componentes de software y hardware con las especificaciones indicadas en el capítulo anterior, se obtiene como resultado la arquitectura a utilizar para implementar y probar los módulos desarrollados. La siguiente figura ilustra los componentes de la arquitectura resultante.

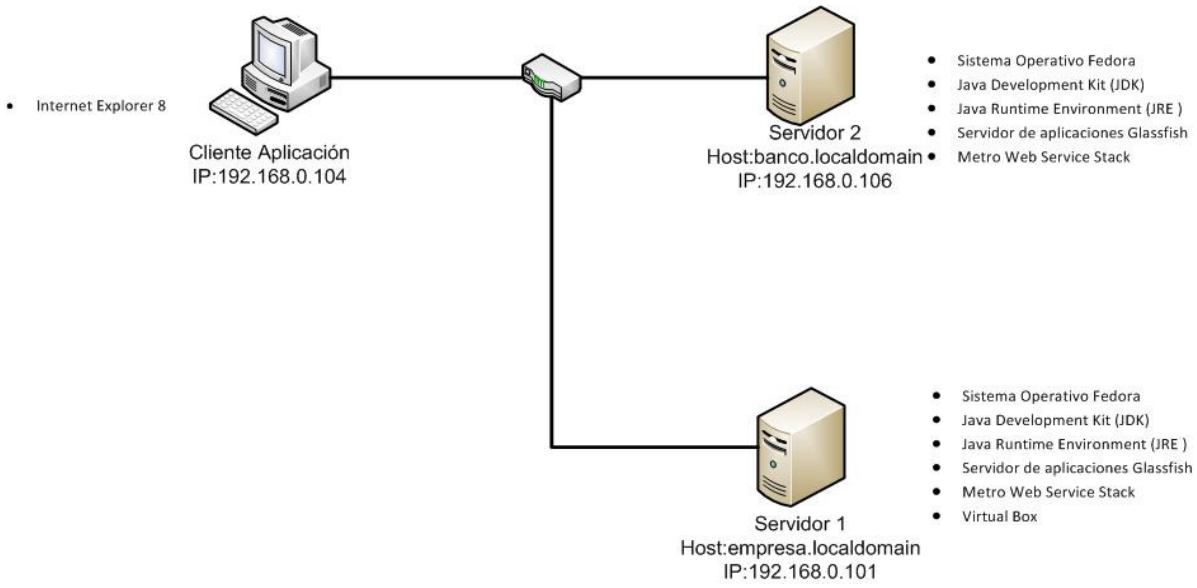


Figura 7: Diagrama de Arquitectura implementado

4.1.2. Instalación de certificados

Después de desplegar las aplicaciones, se deben importar los certificados creados en el capítulo anterior en el almacén de claves keystore y cacerts de los servidores de aplicaciones.

El procedimiento para importar los certificados utilizando la herramienta keytool desde la línea de comandos es el siguiente:

Paso 1: Ejecutar los siguientes comandos para importar los certificados p12 en el almacén de claves keystore:

keytool -importkeystore -destkeystore keystore.jks -deststorepass changeit -deststoretype jks -srckeystore empresa.p12 -srcstorepass changeit -srcstoretype pkcs12

keytool -importkeystore -destkeystore keystore.jks -deststorepass changeit -deststoretype jks -srckeystore banco.p12 -srcstorepass changeit -srcstoretype pkcs12

Paso 2: Ejecutar los siguientes comandos para exportar los certificados .cer del almacén de claves keystore.jks

keytool -exportcert -alias empresa -storepass changeit -keystore keystore.jks -file empresa.cer

keytool -exportcert -alias banco -storepass changeit -keystore keystore.jks -file banco.cer

Paso 3: Ejecutar el siguiente comando para importar los certificados .cer en el almacén de claves cacerts ejecutar el siguiente comando:

keytool -import -noprompt -trustcacerts -alias banco -file banco.cer -keystore cacerts.jks -storepass changeit

keytool -import -noprompt -trustcacerts -alias empresa -file empresa.cer -keystore cacerts.jks -storepass changeit

Para visualizar los certificados que se agregaron en los almacenes de claves se utiliza el siguiente comando:

keytool -list -keystore keystore.jks -alias empresa –v

Al ejecutar el comando se pedirá la contraseña del almacén de claves. Después de introducir la contraseña, se desplegará la información del certificado especificado como se muestra a continuación:

Nombre de alias: empresa
Fecha de creación: 27/03/2011
Tipo de entrada: PrivateKeyEntry
Longitud de la cadena de certificado: 1
Certificado[1]:
Propietario: O=Internet Widgits Pty Ltd, ST=Some-State, C=AU
Emisor: O=Internet Widgits Pty Ltd, ST=Some-State, C=AU
Número de serie: a4c5edb34ab27361
Válido desde: Sat Mar 26 23:57:34 CST 2011 hasta: Sun Mar 25 23:57:34 CST 2012
Huellas digitales del certificado:
MD5: 59:5C:5A:08:95:8D:6A:97:0E:48:A2:F7:D6:1B:EF:5D
SHA1: 12:0F:E9:6E:F7:35:BB:09:1D:B4:EE:A9:73:B3:EA:A0:90:B8:D9:F5
Nombre del algoritmo de firma: MD5withRSA
Versión: 3
Extensiones:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: FE FA CC 47 8A E3 79 34 DC CA B5 63 33 62 62 46 ...G..y4...c3bbF
0010: 0E B6 4E A8 ..N.
#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
CA:true
PathLen:2147483647
#3: ObjectId: 2.5.29.35 Criticality=false

```
AuthorityKeyIdentifier [
KeyIdentifier [
0000: FE FA CC 47 8A E3 79 34   DC CA B5 63 33 62 62 46  ...G..y4...c3bbF
0010: 0E B6 4E A8                                        ..N.
```

4.1.3. Instalación de aplicativos

Para efectuar la implementación de aplicaciones o módulos en el servidor de aplicaciones Glassfish, es necesario tener el archivo WAR de la aplicación o módulo que se desea implementar.

El primer paso para implementar una aplicación es ingresar a la consola de administración del servidor. La consola de administración se instala por defecto en el puerto 4848 y se puede acceder mediante el URL http://localhost:4848/.

Después de haber iniciado la consola de administración, se debe ingresar al menú Aplicaciones. A continuación se mostrará la interfaz de aplicaciones que permite implementar, eliminar, activar y desactivar aplicaciones. La siguiente figura ilustra la interfaz de aplicaciones.

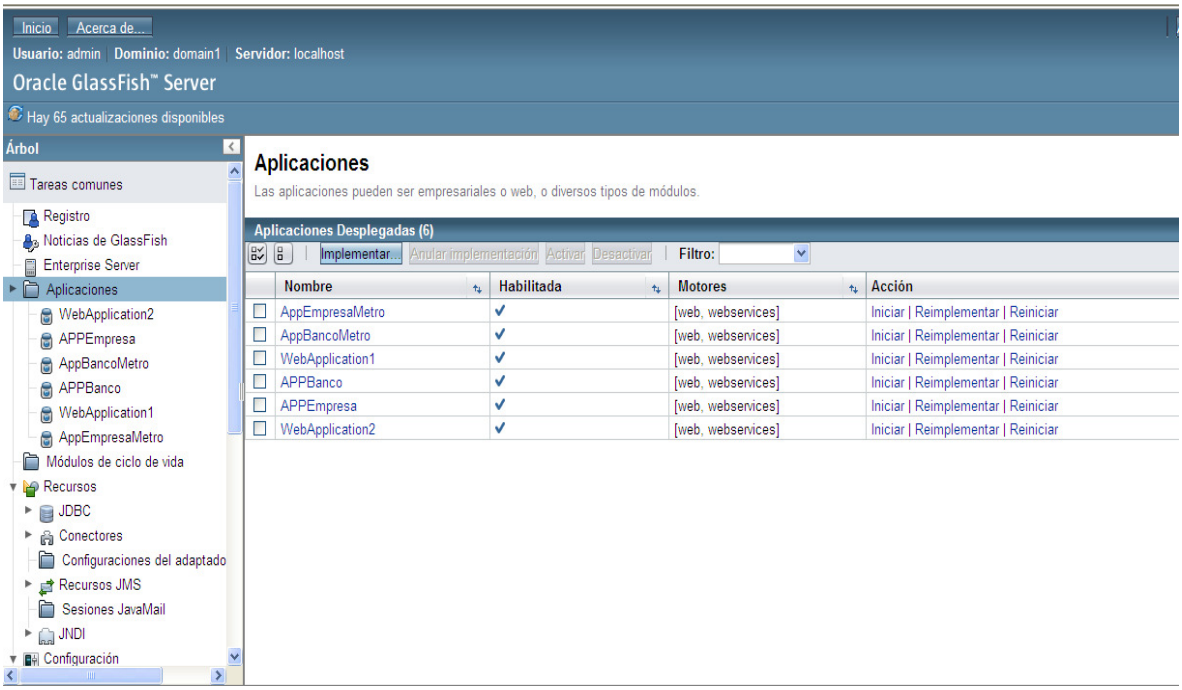


Figura 8: Interfaz de administración de aplicaciones.

Para desplegar una nueva aplicación se debe seleccionar el botón Implementar. Se desplegará la interfaz desplegar aplicaciones, en la que se debe definir la ubicación del

archivo de la aplicación (WAR), el tipo de aplicación, root de contexto, nombre de la aplicación como datos obligatorios. La siguiente figura muestra la interfaz para desplegar aplicaciones o módulos.

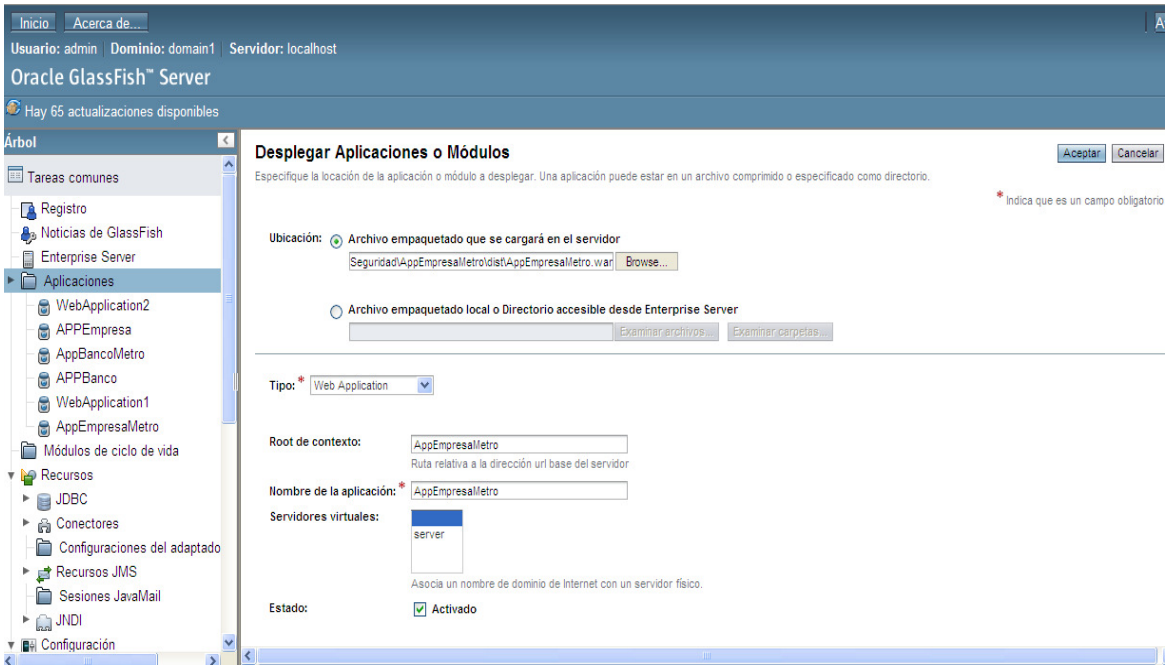


Figura 9: Interfaz para desplegar aplicaciones.

4.1.4. Implementación del servicio web

Para la implementación de la aplicación del servicio web se utilizarán los siguientes componentes:

- Servidor donde se instala: Servidor 1
- Archivo a desplegar: AppEmpresaMetro.war

Utilizando la consola de administración y siguiendo los pasos de instalación descritos anteriormente, se deben realizar las siguientes acciones:

- Ingresar a la interfaz para desplegar aplicaciones o módulos
- Seleccionar el archivo AppEmpresaMetro.war como el archivo a desplegar.

- Dejar los valores por defecto y seleccionar aceptar.

4.1.5. Puntos de Ejecución del Servicio Web

- Para poder hacer uso de las aplicaciones, es necesario conocer los puntos de ejecución disponibles.
- La siguiente tabla muestra los puntos de ejecución generados en el servidor de aplicaciones para el servicio web WsPagos:

Probador:	/AppEmpresaMetro/WsPagosService?Tester
WSDL:	/AppEmpresaMetro/WsPagosService?wsdl
Nombre de puerto:	WsPagosPort
Nombre de clase de implementación:	Pagos.WsPagos
URI de dirección de punto final:	/AppEmpresaMetro/WsPagosService
URL del servidor:	http://192.168.0.101:8080/

Tabla 17: Punto de ejecución del servicio web WsPagos

4.1.6. Implementación del Cliente Proxy

Para la implementación de la aplicación del Cliente proxy se utilizarán los siguientes componentes:

- Servidor donde se instala: Servidor 2
- Archivo a desplegar: AppBancoMetro.war

Utilizando la consola de administración y siguiendo los pasos de instalación descritos anteriormente, se deben realizar las siguientes acciones:

- Ingresar a la interfaz para desplegar aplicaciones o módulos
- Seleccionar el archivo AppBancoMetro.war como el archivo a desplegar.
- Dejar los valores por defecto y seleccionar aceptar.

4.1.7. Puntos de Ejecución del Cliente Proxy

La siguiente tabla muestra los puntos de ejecución generados en el servidor de aplicaciones para la aplicación del cliente proxy:

Inicio:	/AppBancoMetro/CapturaBanco.jsp
URL del servidor:	http:// 192.168.0.106:8080/

Tabla 18: Punto de ejecución del cliente proxy WsPagos

4.2. Configuración de Ambiente de Pruebas

Antes de iniciar la batería de pruebas es necesario explicar con claridad los componentes que se examinarán durante las pruebas. La siguiente tabla contiene los componentes, el tipo de componente, el equipo donde se encuentra instalado y finalmente el equipo desde el que se está utilizando.

Componente	Tipo componente	Instalado en	Utilizado en
WSPagosService	Servicio web	Servidor 1	Servidor 2
WSPagosService	Cliente Proxy	Servidor 2	Servidor 2
CapturBanco.jsp	JSP	Servidor 2	Cliente
EnviDatos.JSP	JSP	Servidor 2	Cliente

Tabla 19: Componentes de las pruebas

Para poder enviar, interceptar y analizar la información que viaja en la red, se requiere establecer previamente la configuración de los siguientes elementos:

4.2.2. Configuración de Nombres de Host

Debido a que los servidores no están publicados en un servidor de dominios, es necesario agregar a la configuración de red los hostname de los servidores a los que requieren acceder, la siguiente tabla contiene los hostname, IpS y el nombre de los equipos a configurar:

Hostname	IP	Equipos
empresa.localdomain	192.168.0.101	Servidor 2, Cliente
banco.localdomain	192.168.0.106	Servidor 1, Cliente

Tabla 20: Parámetros de nombres de host

4.2.3. Configuración de Cortafuegos

Para permitir a los equipos externos el acceso a los puertos que se utilizarán para las aplicaciones y las pruebas, es necesario establecer las políticas del cortafuegos de los servidores 1 y 2. Debido a que el servidor de aplicaciones publica las aplicaciones utilizando el puerto 8080 y la herramienta de monitoreo TcpMonitor escucha el tráfico en el puerto 8081, es necesario establecer para el servicio http estos puertos como de confianza.

4.2.4. Configuración de TcpMonitor

Para monitorear los mensajes que se envían y reciben con la aplicación se utilizará la herramienta TcpMonitor en el puerto 8081. La siguiente tabla contiene los valores de configuración de TcpMonitor que se requieren para realizar las pruebas de la solución.

Equipo	Target Hostname	Target port	Listen Port
Servidor 1	banco.localdomain	8080	8081
Servidor 2	empresa.localdomain	8080	8081

Tabla 21: Parámetros de TcpMonitor

Ver anexo 2 para detalles de la configuración de TcpMonitor.

4.3. Pruebas

Después de instalar y configurar los aplicativos en los equipos correspondientes, el siguiente paso es realizar pruebas a los aplicativos y a los mecanismos de seguridad, para verificar su correcto funcionamiento, lo anterior para garantizar que la implementación resuelva los problemas de seguridad planteados en esta tesis. Para realizar esta actividad se construyo la siguiente batería de pruebas:

Nombre de la Prueba	Descripción
Prueba de funcionalidad de la solución	Se debe verificar la funcionalidad de los aplicativos, el cliente proxy y el servicio web como una solución integral.
Prueba de autorización y autenticación	Se debe verificar que el servicio web no se encuentre disponible a terceros no autorizados y que solo pueden acceder al servicio web aquellos que están definidos en la política de seguridad de la empresa.
Prueba de integridad, confidencialidad y no repudio	Se debe verificar que la información que se envía y recibe entre el cliente proxy y el servicio web, ha sido firmada y cifrada utilizando los certificados digitales correspondientes.

Tabla 22: Batería de Pruebas

4.3.2. Prueba de Funcionalidad de la Solución

Las pruebas para demostrar la funcionalidad del aplicativo como una solución se realizan bajo el siguiente escenario:

Se captura la información del pago en el explorador web del cliente con la aplicación CapturaPago.jsp, se envían los datos al servicio web WSPagosService utilizando el cliente proxy WSPagosService. La siguiente figura ilustra la interacción de los componentes descritos en el escenario:

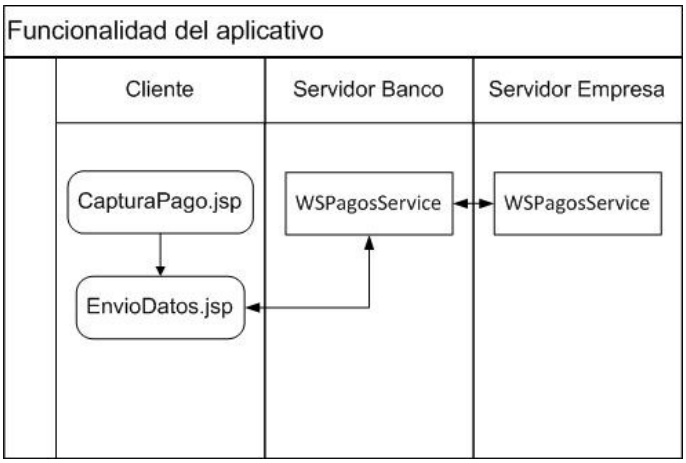


Figura 10: Diagrama de flujo prueba de funcionalidad

Para capturar los datos del pago de servicios se debe ingresar el URL http://banco.localdomain:8081/AppBancoMetro/CapturaBanco.jsp en el explorador del cliente. La siguiente figura muestra la pantalla de captura de pago de servicios:

Figura 11: Pantalla de captura de pago de servicios

Después de capturar la información del pago, se envían los datos de la transacción al servicio web, finalmente el servicio web regresa un mensaje indicando la transacción se realizó con éxito o si existió algún error.

Sí la transacción es exitosa el servicio web regresa el mensaje de transacción exitosa además de la información de la transacción recibida. La siguiente imagen muestra el resultado cuando la transacción es exitosa.

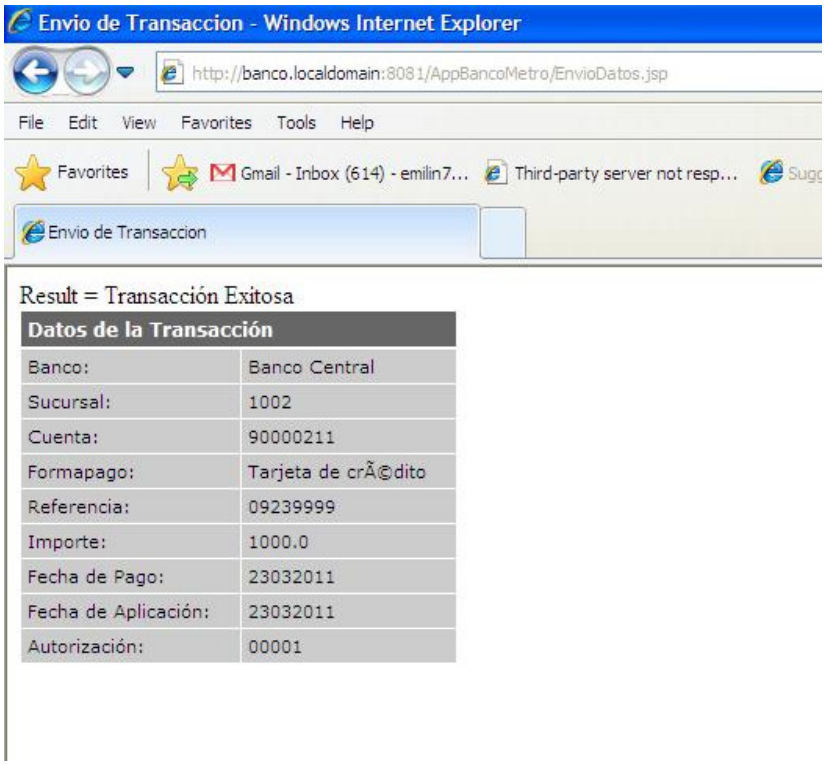


Figura 12: Pantalla de confirmación de transacción

En caso de existir algún error el servicio web regresa el mensaje con la descripción del error ocurrido. La siguiente imagen muestra el mensaje del servicio web cuando existe un error.

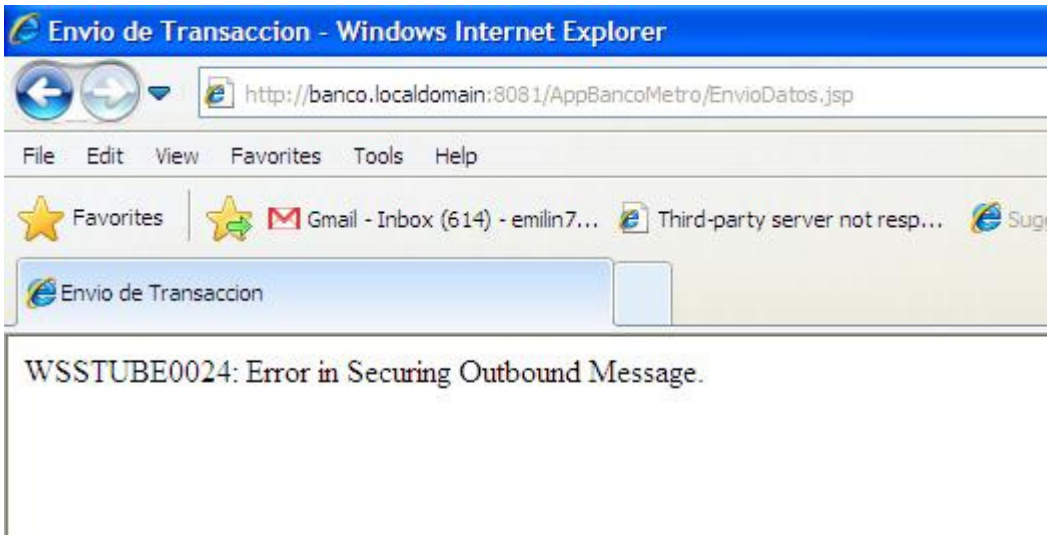


Figura 13: Pantalla de error en transacción

Con las evidencias obtenidas durante las pruebas, se puede verificar el correcto funcionamiento de la solución implementada, por lo que la prueba de funcionalidad de la solución es **satisfactoria.**

4.3.3. Prueba de Autorización y Autenticación

La prueba para verificar la autorización y autenticación del servicio web se realiza bajo los siguientes escenarios:

Escenario 1

Se utiliza la pantalla del probador del servicio web, en el explorador web del cliente para verificar si es posible acceder al servicio web sin autorización. La siguiente figura ilustra la interacción de los componentes descritos en el escenario:

Figura 14: Diagrama de flujo prueba de autorización y autenticación

Para acceder a la pantalla de prueba del servicio web, se debe ingresar el siguiente url http://empresa.localdomain:8081/AppEmpresaMetro/WsPagosService?Tester. en el explorador del cliente.

El resultado al intentar acceder a la pantalla de prueba se muestra en la siguiente imagen:

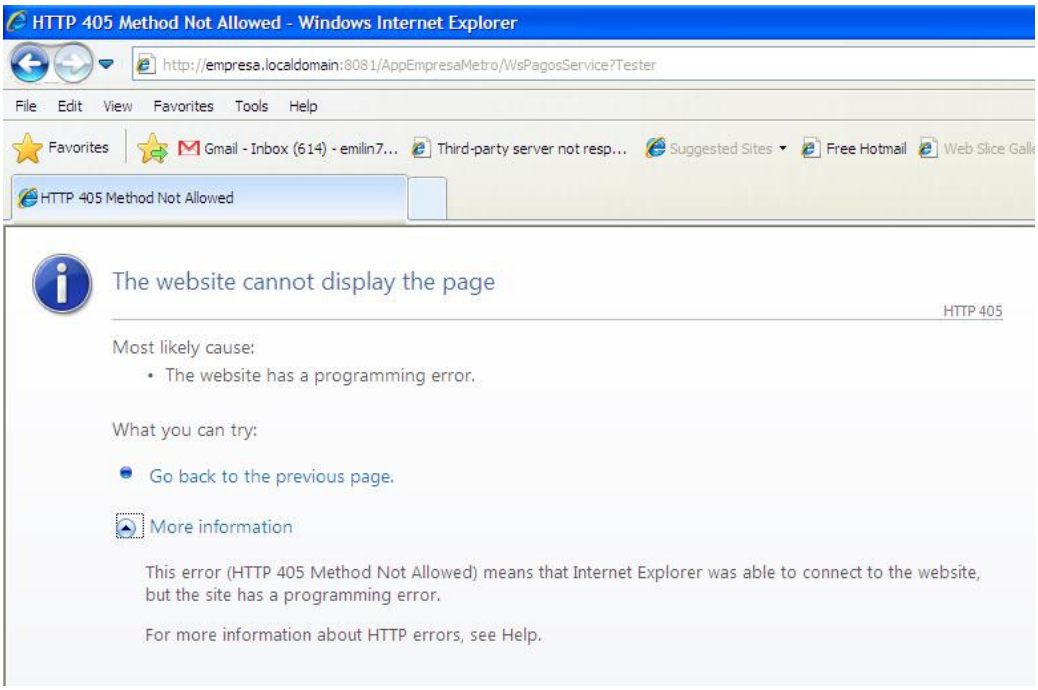


Figura 15: Pantalla de error punto de ejecución tester

Los datos obtenidos en la captura del tráfico en la red, al momento de enviar la petición del servicio web usando la pantalla de prueba son:

Captura de Tráfico Servidor 1

```
<html>
  <head>
    <title>Invalid Method Type</title>
  </head>
  <body>WsPagosService is a secured web service; Tester feature is not supported for secured services</body></html>
```

Escenario 2

Para demostrar la autorización y autenticación en este escenario, se utilizan en el servidor del banco almacenes de claves que no contienen los certificados de la empresa y el banco. Lo anterior para verificar si un servidor que no cuenta con las credenciales necesarias,

puede hacer uso del servicio web. Las pruebas de funcionalidad del aplicativo se realizan bajo el siguiente escenario:

Se captura la información del pago en el explorador web del cliente con la aplicación CapturaPago.jsp y se envían los datos al servicio web WSPagosService utilizando el cliente proxy WSPagosService. La siguiente figura ilustra la interacción de los componentes descritos en el escenario:

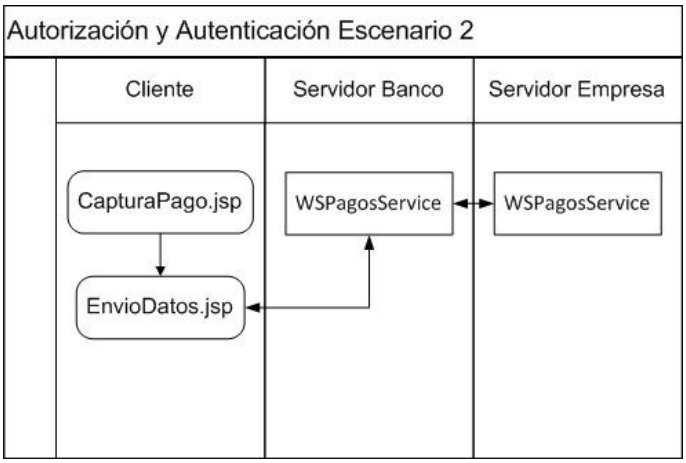


Figura 16: Diagrama de flujo prueba de autorización y autenticación

El resultado al intentar enviar la información capturada con la aplicación del banco se muestra en la siguiente imagen:

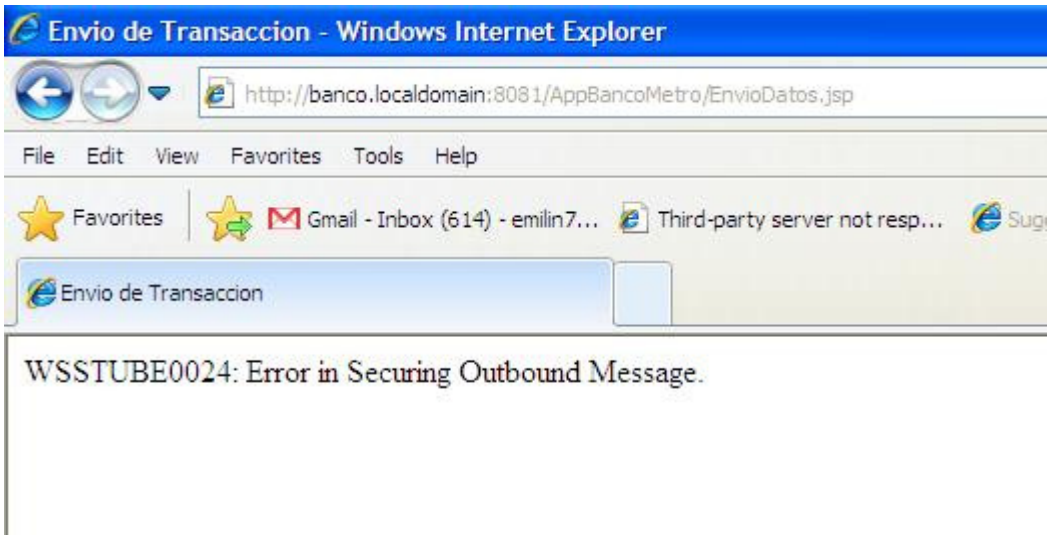


Figura 17: Pantalla de error prueba de autorización y autenticación

Los datos obtenidos en la captura del tráfico en la red del servidor 2, al momento de enviar la petición al servicio web usando el cliente proxy se muestran a continuación:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"   "http://www.w3.org/TR/html4/loose.dtd">
  <html>
    <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Envio de Transaccion</title>
        <link rel=stylesheet href="style.css" type="text/css">
    </head>
    <body>       WSSTUBE0024: Error in Securing Outbound Message.    </body>
  </html>
```

Como resultado de las pruebas de este escenario se obtiene el siguiente error:

WSSTUBE0024: Error in Securing Outbound Message.

Para este escenario los certificados empresa y banco fueron retirados de los almacenes de claves cacerts.jks y keystores.jks del servidor del banco. El error se genera debido a que no se encuentran los certificados especificados en las políticas del archivo de configuración WSPagosService.xml. A continuación se presenta la política que hace referencia a estos certificados:

```
<sc:KeyStore wspp:visibility="private" location="glassfish/domains/domain1/config/keystore.jks" type="JKS"
            storepass="changeit" alias="banco" keypass="changeit"/>
<sc:TrustStore wspp:visibility="private" location="glassfish/domains/domain1/config/cacerts.jks" type="JKS"
            storepass="changeit" peeralias="empresa"/>
```

Para corregir el error detectado se deben agregar los certificados empresa y banco a los almacenes de claves cacerts.jks y keystore.jks con los requisitos establecidos en la política, siguiendo los pasos descritos en la instalación de certificados.

Como se puede apreciar en las pruebas realizadas para el escenario 1 y 2, no es posible acceder al servicio web utilizando el explorador web o desde un aplicativo que no cuenta con los certificados y contraseñas requeridos. Con estas evidencias se concluye que las pruebas realizadas para demostrar los mecanismos de autorización y autenticación son **satisfactorias.**

4.3.4. Prueba de Integridad, Confidencialidad y No Repudio

Para verificar la integridad, confidencialidad y no repudio del servicio web, al igual que en la prueba anterior se captura la información del pago en el explorador web del cliente con la aplicación CapturaPago.jsp y se envían los datos al servicio web WSPagosService utilizando el cliente proxy WSPagosService. La siguiente figura ilustra la interacción de los componentes descritos en el escenario:

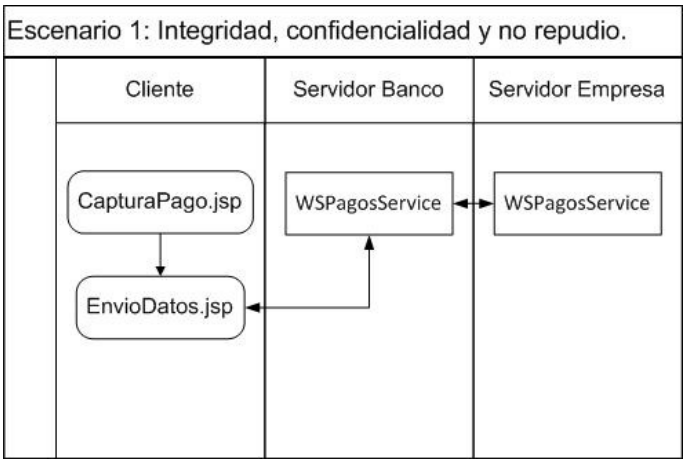


Figura 18: Diagrama de flujo Prueba de integridad, confidencialidad y no repudio

Para demostrar el correcto funcionamiento de los elementos de integridad, confidencialidad y no repudio, se analizarán los datos obtenidos en la captura del tráfico en la red, al momento de enviar la petición al servicio web usando el cliente proxy.

El análisis de los datos se basará en la identificación de los elementos definidos en los estándares. Los elementos XML identificados en los datos analizados que se utilizan para el cifrado y firmado de datos se describen en la siguiente tabla:

Nombre del elemento XML	Elemento de seguridad que implementa
<xenc:EncryptedData>	Confidencialidad
<wsse:BinarySecurityToken>	Integridad y no repudio
</ds:Signature>	Integridad y no repudio

Tabla 23: Relación de elementos XML con elementos de seguridad

A continuación se presenta el contenido de los elementos XML identificados el análisis de los datos capturados en el tráfico de la red:

En el elemento <xenc:EncryptedData> contiene la información cifrada del cuerpo de mensaje, en el elemento <xenc:CipherData> al cifrar la información, se evita que la información interceptada pueda ser leída, garantizando la privacidad de la información.

```
<S:Body wsu:Id="_5002">
  <xenc:EncryptedData xmlns:ns17="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
xmlns:ns16="http://www.w3.org/2003/05/soap-envelope" Id="_5004" Type="http://www.w3.org/2001/04/xmlenc#Content">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
    <xenc:CipherData>

<xenc:CipherValue>883PuFdSN7FHtxj5wr+1XxnmrRtCEHENPQj4cDkFQAZaXuJfdaqpJUbjCXv92iRD4iu68x1WoV60Sr2eNE
edja4Y2XdERRiA7CWAjysc0xnLstAUl5YizaxpIohxRc5lSzeDA2uExzV15lg+ABxUOeGuFJUusltZyqGUPoBIhw6FyjJeO0u0EZV/
1I2SOoCOzrG0byO5Y9zt4XvME+ex7b3fkED65PD1R43VDyvZaSwddiWM7R/X1IHVRzAICr2kp5N4BCBIhJ1pq6QqtqDeTiqjoq
oM0lFj3vXRKZ3knO3C1d3ZrN5yvNzfvw9qdxe1EEaGCsYx1jFKq+sDPdj+B460QNKNFMPc/fjODKDX5j4gpv2aYXk1e5ie3/LGZ
MUIVi80mAFSgUFEWaZuXNdcnxCDEPzK+YH+H6TJEmKPZ807P/ZV6gUFByp9TIlajSZIsCa98zvG08GL5uGikvBoE4v5LFjAm
MZfzQpvgpF//Dddvy0VEXjsyViX7q4Fxzci
</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</S:Body>
```

Dentro de los elementos XML se identificó el elemento <ds:Signature> utilizado para firmar los datos del mensaje. Este elemento contiene varios de los elementos XML definidos en la especificación de XML Signature. La información firmada del mensaje se localiza en el elemento <ds:SignatureValue>.

En el siguiente bloque se presenta el elemento <ds:Signature> y los elementos que lo conforman.

```
    <ds:Signature xmlns:ns17="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
xmlns:ns16="http://www.w3.org/2003/05/soap-envelope" Id="_1">
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <exc14n:InclusiveNamespaces PrefixList="wsse S" />
        </ds:CanonicalizationMethod>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
        <ds:Reference URI="#_5002">
```

<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
<exc14n:InclusiveNamespaces PrefixList="S" />
</ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<ds:DigestValue>g9h3By2t2GS3+fJ+CwdlN/BYOQI=</ds:DigestValue>
</ds:Reference>
<ds:Reference URI="#_3">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
<exc14n:InclusiveNamespaces PrefixList="wsu wsse S" />
</ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<ds:DigestValue>CWHr9LEN5FT2l6XNBnBEp+zI+rE=</ds:DigestValue>
</ds:Reference>
<ds:Reference URI="#uuid_cf2b316d-2e02-4a37-9723-18ea1ae5e7f6">
<ds:Transforms>
<ds:Transform Algorithm="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform">
<wsse:TransformationParameters>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
</wsse:TransformationParameters>
</ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<ds:DigestValue>eeAP7chSrI8oA/iTY3nuwKCEAe8=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>

<ds:SignatureValue>Nqj+K8b0ITxwoZX5b/Tr9wTNTLuMXUf+tBS55reBbigsgHOG+yEYBei/zV1v1fZ2hsfoPBpkI5dSwxRI9l2KIDFfTvKYWHLeOj6Br0jodyKaLwmQW5oa8GCG7eiyl8jzYsWTEY/dMxK3k1Z1CTCdNWwBrRJTTDLNP+HlBscmiUQ=</ds:SignatureValue>
<ds:KeyInfo>
<wsse:SecurityTokenReference>
<wsse:Reference URI="#uuid_c3fa3be8-5700-4701-ad20-d8eda29c2876" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" />
</wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>

Otro elemento identificado necesario para validar la información firmada y saber si ha sido alterada, es el elemento <wsse:BinarySecurityToken> que contiene el certificado digital con el que fue firmada la información, permitiendo al destinatario verificar la integridad de la información firmada utilizando el certificado, garantizando la integridad y el no repudio de la información.

En el siguiente bloque se presenta la conformación del elemento <wsse:BinarySecurityToken>.

<wsse:BinarySecurityToken xmlns:ns17="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512" xmlns:ns16="http://www.w3.org/2003/05/soap-envelope" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" wsu:Id="uuid_c3fa3be8-5700-4701-ad20-d8eda29c2876">MIICWDCCAcGgAwIBAgIJAMBhVKC+NwqAMA0GCSqGSIb3DQEBBAUAMEUxCzAJBgNVBAYTAkFVMRMwEQYDVQQIDApTb21lLVN0YXRlMSEwHwYDVQQKDBhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQwHhcNMTEwMzI3MDU1ODMwWhcNMTIwMzI2MDU1ODMwWjBFMQswCQYDVQQGEwJBVTETMBEGA1UECAwKU29tZS1TdGF0ZTEhMB8GA1UECgwYSW50ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC0G7TYZUx15uEI+BjqVhZbDRf2lC/um6faNu3hEU8xqWFYqORzKe8rHJtSd+tsHXOKp0/Zc3iVkVO+V2Ha/K9rgS7AAb8wX78IATDnc/NukdyNLl0/0tgxxWCXFa1aOIwVXAiojEcPWFR0oZRQ/Np46FPzg3nUukKYT8Tn9512YQIDAQABo1AwTjAdBgNVHQ4EFgQUoC+PMqEpXAPAxyS6YfOeEu33PPEwHwYDVR0jBBgwFoAUoC+PMqEpXAPAxyS6YfOeEu33PPEwDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0BAQQFAAOBgQBlpby2mkp0k1r0vW0sCvYdy1UPxco4Rf7W57qR1/+8jSTnMP2996CKUHOZQQZsldR5g6C+sH02rwrLTpfscWKNg6Aaco9zobxeQpcJTfjtnYagq8RblL4YUSfEQjxVLxWLo5my5fJb2BNVOHBQz3QnQjWPrq2UF6mhM4mhcA80/Q==</wsse:BinarySecurityToken>

Después de inspeccionar los elementos XML capturado en el tráfico de la red del servidor 1, se concluye que la prueba para los elementos de confidencialidad, integridad y no repudio es **satisfactoria.**

Conclusiones

En la actualidad el uso de los servicios web para integrar aplicaciones e intercambiar información en internet se ha incrementado considerablemente, sin embargo la carencia de elementos de seguridad como parte de los servicios web y los riesgos a los que se encuentran expuestos en internet, ha creado la necesidad de implementar elementos de seguridad para satisfacer las necesidades de seguridad que estos requieren.

El presente trabajo se centró en la implementación de elementos de seguridad en servicios web utilizando estándares XML, para proveer confidencialidad, integridad, no repudio, autorización y autenticación entre 2 aplicaciones que intercambian información. Es necesario mencionar que los elementos de seguridad, son determinados de acuerdo al problema a resolver y las necesidades de seguridad que esté requiera.

Para implementar los elementos de seguridad se crearon dos aplicaciones para intercambiar información utilizando servicios web, para ello fue necesario crear e instalar certificados digitales en los almacenes de claves, posteriormente se creó una extensión de la capa de SOAP utilizando WS-Security para configurar los elementos de seguridad.

Debido a que el stack de servicios web metro requiere la extensión SubjectKeyIdentifier en los certificados, se debe utilizar Openssl para la creación de los certificados y utilizar Keytool para importar los certificados, siguiendo los pasos descritos en los capítulos 3 y 4.

Finalmente se efectuaron pruebas para cada elemento de seguridad, los resultados obtenidos demostraron la confidencialidad, integridad, no repudio, autorización y autenticación de las aplicaciones del caso de estudio. Por lo que se puede concluir que la implementación de estándares XML en los servicios web proporcionan una solución integral a los problemas de seguridad propuestos.

Glosario de términos

API (Application Programming Interface): Conjunto de funciones de un sistema definidas de forma estricta para su uso desde un programa.

Certificado Digital: Una porción de información, generalmente un fichero de texto, empleado para establecer una conexión segura. Contienen información acerca del propietario, el emisor del certificado, fechas de emisión y caducidad, una identificación y una marca de seguridad cifrada para verificar el contenido del certificado.

Cortafuegos: Barrera de seguridad formada por uno o más ruteadores capaces de aceptar o rechazar la información transmitida. Se encuentra situada entre la red de la organización y la conexión a Internet, con el fin de impedir accesos no autorizados desde el exterior.

e-business: Aplicación informática dirigida a realizar transacciones comerciales como: distribución, compra, venta, marketing y suministro de información para productos o servicios a través de Internet o Intranet.

Host: Ordenador que, unido a una red, presta servicios a otros ordenadores, terminales o dispositivos.

HTTP (Hyper Text Transfer Protocol): El protocolo HTTP define las peticiones que un cliente puede enviar a un servidor y las respuestas que el servidor puede enviar como respuesta. Cada solicitud contiene una dirección URL, que es una cadena que identifica un componente web o un objeto estático.

IDL (Interface Definition Language): Lenguaje utilizado para describir una interfaz en lenguaje neutral, permitiendo la comunicación entre componentes de software que no comparten el mismo lenguaje de programación.

IP (Internet Protocol): Protocolo de red de nivel 3 (según el modelo OSI) que proporciona las funciones de encaminamiento de datagramas a través de una interconexión de redes.

IT (Information Technology): Es el estudio, diseño, desarrollo, implementación, soporte o dirección de los sistemas de información computarizados, en particular de software de aplicación y hardware de computadoras.

Kerberos: Sistema de seguridad del proyecto Athena del MIT. Está basado en criptografía de clave simétrica.

Llave pública: Clave criptográfica de un usuario que se hace de conocimiento público.

Middleware: Software situado entre sistemas como bases de datos, comunicaciones y programas de aplicación, que actúa como un intérprete.

OASIS (Organization for the Advancement of Structured Information Standards): Consorcio internacional no lucrativo que dirige el desarrollo, la convergencia, y la adopción de normas de comercio electrónico. Produce normas para la seguridad, el comercio electrónico, incidiendo en la estandarización en el sector público y para mercados de aplicación específicos.

PKI (Public Key Infrastructure): La infraestructura de clave pública es una combinación de hardware y software, políticas y procedimientos que permiten asegurar la identidad de los participantes en un intercambio de datos usando criptografía pública.

Servicio Web: Es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

SHA1 (Secure Hash Algorithm 1): SHA-1 es una función criptográfica hash diseñada por la National Security Agency. Se utiliza en diversas aplicaciones en seguridad como aplicaciones y protocolos.

SLA (Service Level Agreement): Acuerdo con una compañía externa o un departamento de la propia compañía, en el que se define qué nivel de servicio de operaciones se va a proporcionar para asegurar la disponibilidad, el rendimiento y la seguridad que se precisa para el trabajo a realizar.

SMTP (Simple Mail Transfer Protocol): El protocolo principal para enviar correo empleado en Internet. Se emplea entre el cliente de correo y el servidor y entre los servidores.

SOA (Service Object Architecture): Es un concepto de arquitectura de software que define la utilización de servicios que se comunican entre sí para dar soporte a los requisitos del negocio.

SOAP (Simple Object Access Protocol): El protocolo SOAP proporciona una estructura de empaquetado estándar, para el transporte de documentos XML a través de una variedad de tecnologías estándar de Internet, incluyendo SMTP, HTTP y FTP.

SSL (Secure Sockets Layer): Protocolo diseñado para permitir comunicaciones cifradas y autentificadas a través de Internet. Se emplea para la conexión segura entre dos nodos de la red.

TCP (Transfer Control Protocol): Protocolo orientado a la conexión desarrollado para la interconexión de redes. Este protocolo garantiza que la comunicación entre dos aplicaciones sea precisa.

UDDI (Universal Description, Discovery , and Integration): Es uno de los estándares básicos de los servicios web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL.

WSDL (Web Service Description Language): Es una tecnología XML que describe la interfaz de un servicio web de forma estandarizada. WSDL estandariza como un servicio web representa la entrada y salida de parámetros de una invocación externa.

W3C (World Wide Web Consortium): Conjunto de compañías comerciales e instituciones educativas que dirige la investigación y definición de normas en todas las áreas relacionadas con el WWW, con la intención de asegurar su estabilidad en su rápida evolución, desarrolla tecnologías inter-operativas.

XML (Extensible Markup Language): Es un formato de texto simple y muy flexible derivado de la SGML (ISO 8879). Originalmente diseñado para afrontar los retos de la publicación electrónica a gran escala, es utilizado en el intercambio de una amplia variedad de datos en la web y en otros lugares.

XSD (XML Schema Definition): Recomendación del World Wide Web Consortium, que especifica como describir formalmente los elementos en un documento XML.

Bibliografía

[1] Anura Gurugé, Web Services: Theory and Practice, Digital Press, 2004

[2] Bret Hartman, Donald J. Flinn, Konstantin Beznosov, Shirley Kawamoto, Mastering Web Services Security, Wiley Publishing Inc., 2003

[3] Datamonitor, The adoption of SOA among US and Western European Enterprises (Customer Focus), 2006.

[4] David Chappell, Java Web Services, O´Reilly, 2002

[5] Dirk Krafzig, Karl Banke, Dirk Slama, Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall PTR, 2004

[6] Doug Tidwell, James Snell, Pavel Kulchenko, Programming Web Services with SOAP, O'Reilly, 2001

[7] Eric Newcomer, Understanding Web Services, Pearson Education Corporate Sales Division, 2002

[8] Mark O'Neill, et al, Web Services Security, McGraw-Hill/Osborne, 2003

[9] Ramarao Kanneganti, Prasad Chodavarapu, SOA Security, Manning Publications Co., 2008.

[10] Ramesh Nagappan, Robert Skoczylas, Rima Patel Sriganesh, Developing Java Web Services, Wiley Publishing Inc., 2003

[11] Thomas Erl., Service-Oriented Architecture: Concepts, Technology, and Design Prentice Hall, 2005

[12] Thomas Erl., SOA: principles of service design, Prentice Hall, 2008

[13] Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)

[14] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[15] http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

[16] http://www.w3.org/TR/xmlenc-core/

Apéndice 1
Instalación de Software

Instalación Java Development Kit (JDK)

Para la instalación del Java Development Kit en Linux Fedora 14, se deben seguir los siguientes pasos:

Paso 1: Iniciar sesión con el usuario root, de no haberse iniciado la sesión con root, utilizar el siguiente comando para convertirse en super usuario:

```
[root@empresa Descargas]#su
```

Paso 2: Descargar el archivo jdk-6u24-linux-i586.bin del siguiente link:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Paso 3: Copiar archivo jdk-6u24-linux-i586.bin a la carpeta /opt

```
[root@empresa Descargas]#cp jdk-6u24-linux-i586.bin /opt
```

Paso 4: Establecer los permisos de ejecución del archivo:

```
[root@empresa opt]#chmod 777 jdk-6u24-linux-i586.bin
```

Paso 5: Ejecutar el archivo de instalación del JDK:

```
[root@empresa opt]#./jdk-6u24-linux-i586.bin
```

Paso 6: Establecer la variable de entorno $JAVA_HOME

Para establecer la variable de entorno $JAVA_HOME

```
[root@empresa opt]#cd /root

[root@empresa]#vi .bash_profile
```

Agregar los siguientes comandos en el archivo .bash_profile:

```
JAVA_HOME=/opt/jdk1.6.0_24
PATH=$PATH:$HOME/bin:/usr/java/jre1.6.0_24/bin:$JAVA_HOME/bin
export PATH
export JAVA_HOME
```

Paso 5: Finalmente para aplicar los cambios cierre la sesión e ingrese nuevamente.

Instalación Java Runtime Environment (JRE)

Para la instalación del Java Runtime Environment en Linux Fedora 14, se deben seguir los siguientes pasos:

Paso 1: Iniciar sesión con el usuario root, de no haberse iniciado la sesión con root, utilizar el siguiente comando para convertirse en super usuario:

```
[root@empresa Descargas]#su
```

Paso 2: Descargar el archivo jre-6u24-linux-i586-rpm.bin del siguiente link:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Paso 3: Establecer los permisos de ejecución del archivo:

```
[root@empresa Descargas]#chmod 777 jre-6u24-linux-i586-rpm.bin
```

Paso 4: Ejecutar el archivo de instalación del JRE:

```
[root@empresa Descargas]#./jre-6u24-linux-i586-rpm.bin
```

Instalación Servidor de Aplicaciones Glassfish

Para la instalación el Servidor de aplicaciones Glassfish en Linux Fedora 14, se deben seguir los siguientes pasos:

Paso 1: Iniciar sesión con el usuario root, de no haberse iniciado la sesión con root, utilizar el siguiente comando para convertirse en super usuario:

```
[root@empresa Descargas]#su
```

Paso 2: Descargar el archivo glassfish-3.1-unix-ml.sh del link:

http://glassfish.java.net/downloads/3.1-final.html

Paso 3: Establecer los permisos de ejecución del archivo:

```
[root@empresa Descargas]#chmod 777 glassfish-3.1-unix-ml.sh
```

Paso 4: Ejecutar el asistente de instalación de Glassfish:

```
[root@empresa Descargas]# ./glassfish-3.1-unix-ml.sh
```

Se iniciará el asistente de instalación, que nos guiará durante la instalación. El primer paso es elegir el Tipo de instalación a realizar, se seleccionará instalación típica.

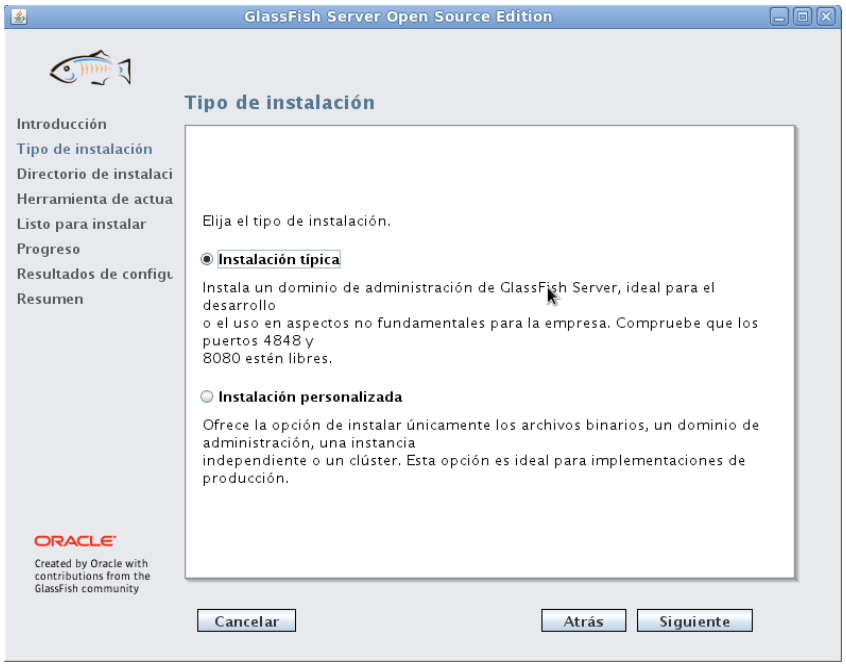


Figura 19: Tipo de instalación

El siguiente paso es seleccionar el directorio de instalación de Glassfish, en este caso se instalará en la siguiente ruta /root/glassfish3

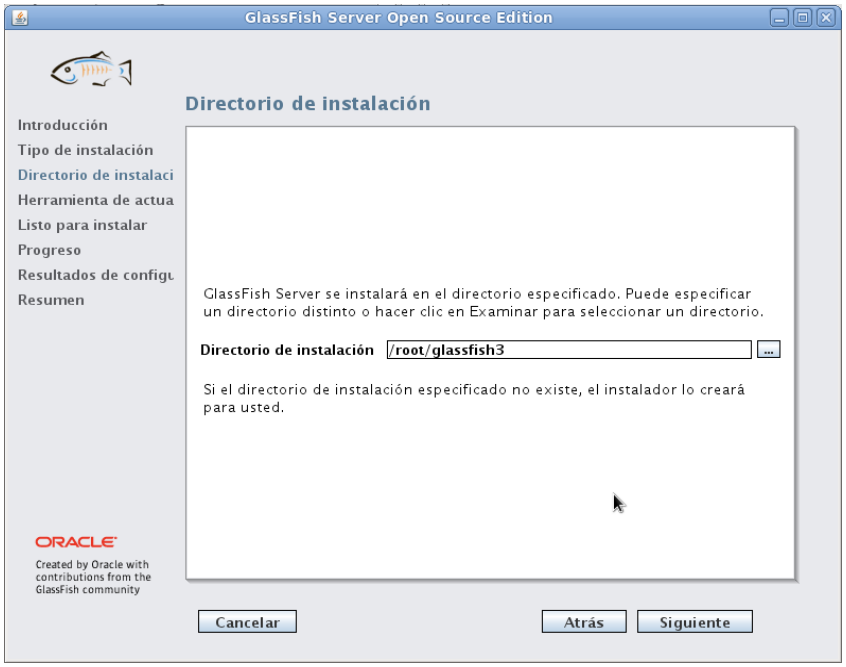


Figura 20: Ruta de instalación

Para mantener el servidor de aplicaciones actualizado y poder agregar y actualizar componentes, se debe seleccionar la opción de Instalar herramienta de actualización.

Figura 21: Herramienta de actualización

Finalmente se ejecuta el boton instalar para iniciar la instalacion de Glassfish.

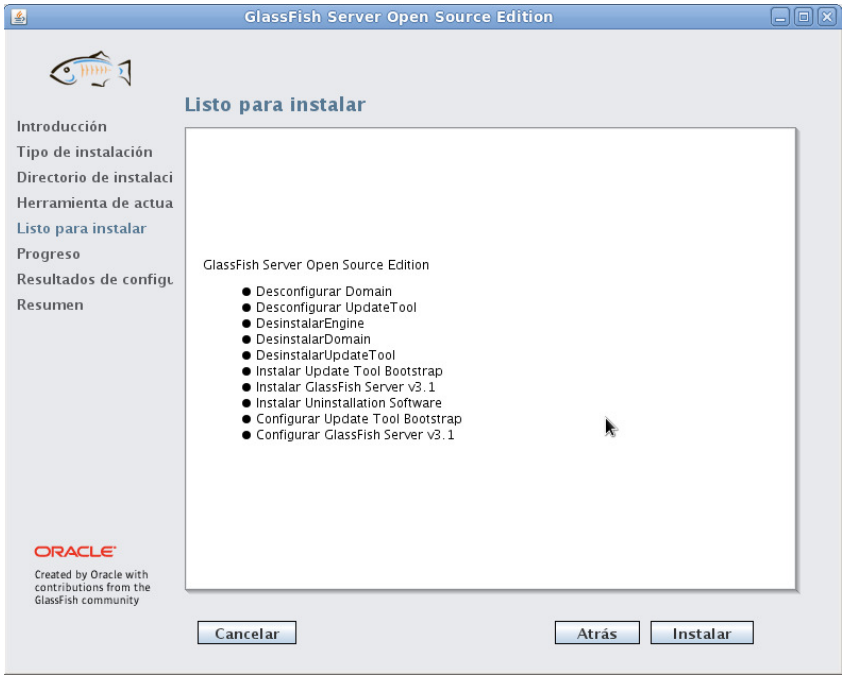


Figura 22: Inicio de Instalación

Instalación de Metro en Glassfish

Para instalar el stack de servicios web metro en el servidor de aplicaciones Glassfish, se utilizará la herramienta de actualización de Glassfish.

Para instalar esta herramienta sigua los siguientes pasos:

Paso 1: Instalar la herramienta de actualización con los siguientes comandos:

```
[root@empresa /]#cd /root/glassfish3/pkg/bin/
[root@empresa bin]#./pkg install updatetool
```

Paso 2: Ejecutar la herramienta de actualización con los siguientes comandos:

```
[root@empresa bin]#cd /root/glassfish3/updatetool/bin/
[root@empresa bin]#./updatetool
```

La siguiente pantalla muestra la herramienta de actualización, en la que se pueden instalar o actualizar los componentes de Glassfish. Se puede apreciar que el componente metro web service stack ya se encuentra instalado.

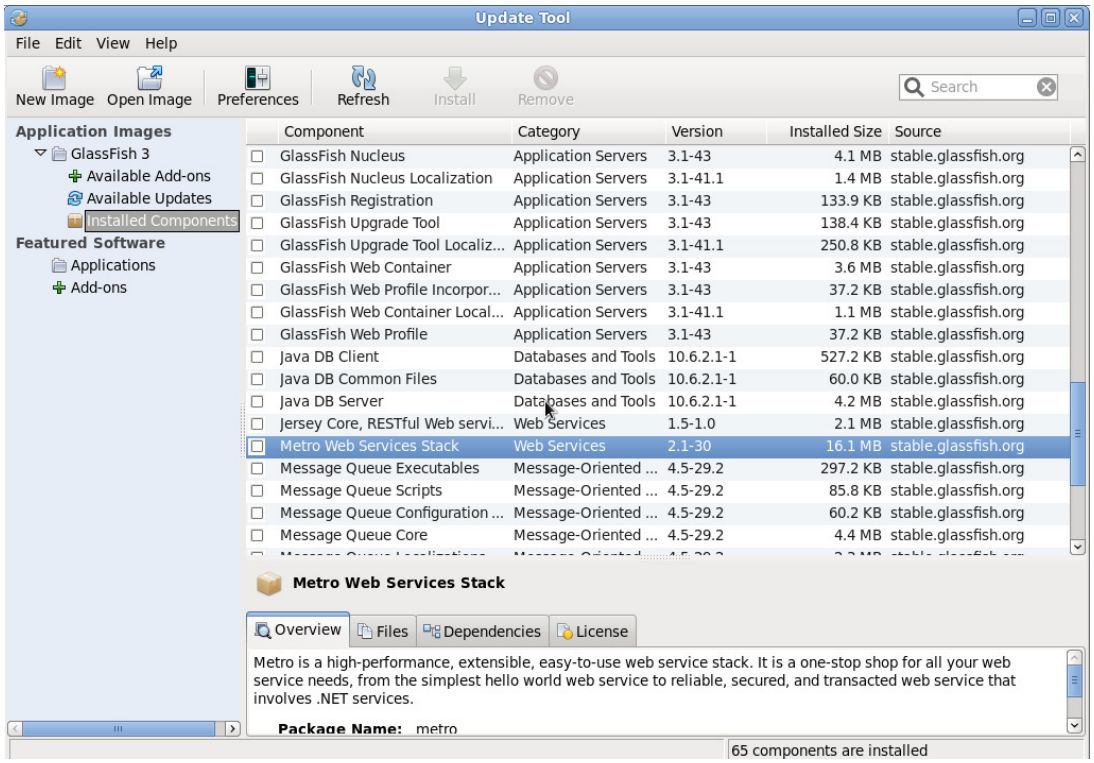


Figura 23: Herramienta Update tool Glassfish

Apéndice 2
Configuraciones

Actualización fedora

Para buscar e instalar en línea las actualizaciones del sistema operativo fedora se debe usar el siguiente comando:

```
[root@empresa /]#yum update
```

Habilitar usuario root en fedora 14

En el sistema operativo fedora el usuario root se tiene deshabilitado por defecto en el inicio de sesión de la interfaz gráfica. Por lo que es necesario habilitarlo de forma manual, el siguiente procedimiento habilita el inicio de sesión para el usuario root:

Paso 1: Comentar la linea auth required pam_succeed_if.so user != root quiet en el archivo pam.d ejecutando los siguientes comandos.

```
[root@empresa etc]$cd etc
[root@empresa etc]$cd pam.d
[root@empresa etc]$vi gdm
#auth       required     pam_succeed_if.so user != root quiet
```

Paso 2: Comentar la linea auth required pam_succeed_if.so user != root quiet en el archivo gdm-password

```
[root@empresa etc]$vi gdm-password
#auth       required       pam_succeed_if.so user != root quiet
```

Finalmente para ver los cambios realizados se debe cerrar la sesión e iniciar la sesión usando el usuario root.

Configuración TcpMonitor

Para configurar de la herramienta de monitoreo TcpMonitor se requieren especificar los parámetros Target Hostname, Target port y Listen Port. La siguiente pantalla muestra la configuración de TCP Monitor para la captura el tráfico del puerto 8080 en el host banco.localdomain a través del puerto 8081.

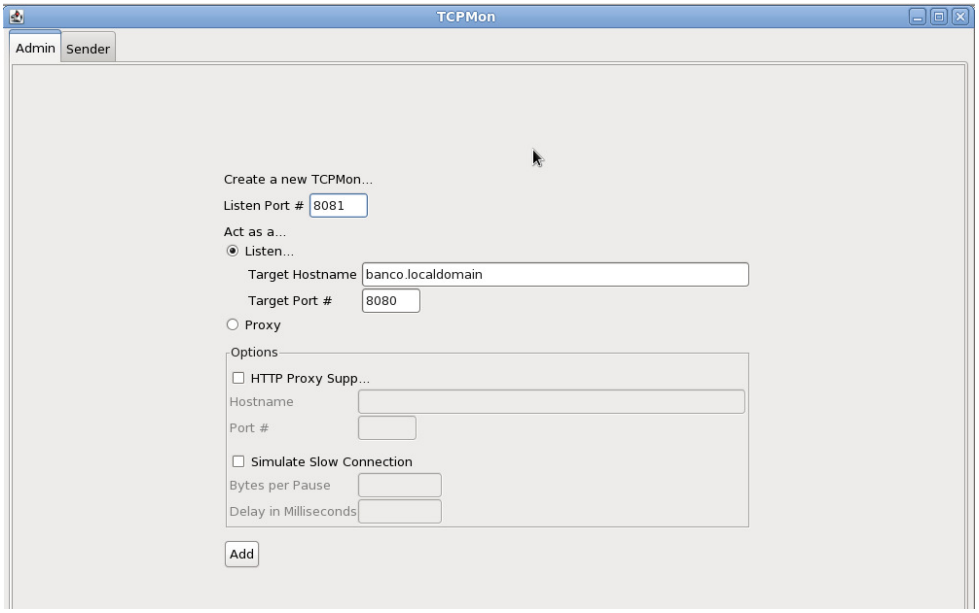


Figura 24: Pantalla de administración TcpMonitor