UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 3

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Componente para la gestión de la seguridad de los servicios web en el marco de trabajo Sauxe.

Autor: Luis Angel Santana Garriga.

Tutor: Dr. Oiner Gómez Baryolo

La Habana, 2013

"Año 55 de la Revolución "

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de este trabajo y autorizo al Departamento de Tecnología de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____
Luis Angel Santana Garriga

_____
Oiner Gómez Baryolo

Datos de Contacto

Tutor: Dr. Oiner Gómez Baryolo

Graduado de Ingeniero en Ciencias Informática en la Universidad de las Ciencias Informáticas, La Habana, Cuba, en el 2007. En la cual obtuvo el título de máster en Informática Aplicada en el 2010 y el grado científico de Doctor en Ciencias Técnicas en el 2012. Actualmente está al frente de un departamento de desarrollo tecnológico para sistemas de gestión empresarial y sus investigaciones se concentran en el área de la seguridad para este tipo de soluciones informáticas.

oiner@uci.cu

DEDICATORIA

Dedico esta tesis, en especial a mamá, por darme su apoyo en los momentos más difíciles y confiar siempre en mí, por brindarme su amor, cariño y comprensión. A mi papá por ser mi guía, mi amigo, el ejemplo a seguir, espero que estén orgullosos de mí y poder devolverles todo lo que han hecho por mí. A mi hermano, que espero poder verlo haciendo su tesis muy pronto, a mis abuelos, a mis tíos y tías, a mis primos y mis amigos de la UCI y de Manzanillo en especial a Julio y Alain.

AGRADECIMIENTOS

Quiero agradecer a todos los que de una forma u otra ayudaron en la realización de este trabajo, muchas gracias. A mi familia en especial a mi mamá y a mi papá, a mi hermano un abrazo grande y un beso para ellos.

A mi tío Delio y mi tío Tito por siempre darme consejos y ser como unos padres para mí, a mis tías Ana, Paula, Mary y Mayke por su ayuda y consejos, a mi abuelo Angel y a mis abuelas Blanca y Argelia que siempre confiaron en mí y me brindaron su amor, a todos mis primos Erne, Isma, Tony, Fico, Emil, Jesús, Eric y en especial Robertico y a Camila a todos muchas gracias por su ayuda.

Agradecer de forma muy especial a Cuca, a Victoria, a Daniel y a Hugo por su ayuda y abrirme las puertas de su casa durante estos 5 años de universidad. A Anita, Rafael, Arle y Jasmina, por su apoyo incondicional mil gracias, a Chicho, Mary y a Kare.

A la gente del proyecto ACAXIA, a Mile, Rene, Andrés, a todos los que me ayudaron y con los cuales compartí buenos momentos, a Janier, José, Quiroga, Racielis, Héctor, los Carlos, Celso, Eidel, Katia. A la gente del edificio a Yandy, David, Aroldo.

Agradecimientos a mi tutor Oiner por su ayuda en la realización de este trabajo, sus consejos, por ayudarme a superarme y a prepararme para el futuro.

Agradecerle a Mayte por ayudarme en todo lo que me hizo falta, a Evelio por sus conocimientos deportivos y musicales, a Lisbet, Dailin, Male, Betty, Roider, Diogenes, Raidel, Camilo, Ale, Norge, Alfredo, a la China, a Yudi, a todos los del grupo por los años de convivencia juntos y los buenos ratos que hemos compartido, a los que ya no están en el grupo pero siguen siendo parte de él, a Maire, Yamila y en especial a Jany por todos sus conocimientos sobre cocina, lavado, limpieza y organización muchas gracias me han servido de mucho, a los que a no están en la universidad a Alain y al Pedri, y por último pero no menos importante sino todo lo contrario un agradecimiento muy, pero muy especial a Jenny por haberme ayudado en todo, no solo con las cosas de la tesis y de la docencia, sino que también por ayudarme a no perder el tiempo mil gracias.

RESUMEN

El marco de trabajo Sauxe publica varios de servicios web, los cuales pueden ser consumidos por sistemas externos sin que se realicen verificaciones y validaciones respecto a que si realmente el sistema que realiza la petición tiene los permisos necesarios para realizar esa petición. Esto afecta la disponibilidad, confidencialidad e integridad de la información que viaja a través de servicios web.

El presente trabajo tiene como objetivo desarrollar una solución que permita gestionar la seguridad de los servicios web en el marco de trabajo Sauxe, mediante el cual se aumente la confidencialidad, integridad y disponibilidad de los datos. A través de dicha solución se pretende asegurar, mediante escenarios de seguridad que se encuentren bajo el estándar WS-Security, el intercambio de la información que se realiza a través de los servicios web.

Durante la investigación se definieron los estilos arquitectónicos y patrones de diseño a utilizar para implementar la propuesta de solución, se definieron métricas de diseño para evaluar la calidad de los atributos internos del sistema. Para la validación se utilizó un sistema externo, realizando peticiones, primero, sin la utilización de la solución, y después añadiendo el componente y evaluando si existe un aumento de la seguridad en el intercambio de datos a través de servicios web.

Palabras claves: escenarios de seguridad, servicios web**,** WS-Security

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

El auge cada vez mayor de las nuevas Tecnologías de la Informática y las Comunicaciones (TICs) en las diferentes esferas de la sociedad a escala mundial ha alcanzado niveles nunca antes vistos. El impetuoso desarrollo de las ciencias y la tecnología ha llevado a la sociedad a entrar en un nuevo milenio inmerso en lo que se ha denominado “Era de la informatización”. Sin dudas, se está en presencia de una revolución tecnológica y cultural de alcances insospechados [1]. Entre las esferas en que su uso ha alcanzado mayor desarrollo están las industriales, comerciales, militares, investigativas y las de gestión empresarial y de servicios donde son usados los Sistemas de Información (SI).

Un SI es un conjunto de elementos interrelacionados con el propósito de prestar atención a las demandas de información de una organización, para elevar el nivel de conocimientos que permitan un mejor apoyo a la toma de decisiones y desarrollo de acciones [2]. Un SI está compuesto por un conjunto de procesos, datos, modelos, tecnologías y personas que responden a una estructura coherente en función del propósito de una organización [3]. El resultado de la aplicación de estos SI ha traído consigo un incremento en el manejo de la información lo cual ha obligado a las organizaciones a elevar el nivel de seguridad con el objetivo de protegerla.

En los últimos años se ha experimentado un crecimiento significativo de las pérdidas ocasionadas por Ataques Informáticos (AI). Estos son propiciados por aberturas en la seguridad de los SI, denominadas amenazas, las cuales se clasifican en cuatro tipos: interrupción, intercepción, modificación y fabricación. La existencia de estas aberturas ha traído consigo el desarrollo de herramientas y técnicas sofisticadas para ejecutar ataques con diversos objetivos, que han provocado una nueva forma de conflicto denominada ciberguerra [3].

Uno de los objetivos primordiales de los AI son los Sistemas Distribuidos (SD). Estos son definidos como una colección de computadores autónomos conectados por una red, y con el software distribuido adecuado para que el sistema sea visto por los usuarios como una única entidad capaz de proporcionar facilidades de computación [4].

Uno de los objetos de ataques hacia estos sistemas es la vía de comunicación a través de Servicios Web. Estos son componentes de software reutilizables y distribuidos que ofrecen una funcionalidad concreta, independientes tanto del lenguaje de programación en que están implementados como de la plataforma de ejecución. Se puede considerar como aplicaciones autocontenidas que pueden ser descritas, publicadas, localizadas e invocadas sobre la Internet (o cualquier otra red) y basados en estándares de el World Wide Web Consortium (W3C), especialmente eXtensible Markup Language (XML) [5].

Los servicios web ofrecen algunas ventajas significativas sobre los modelos de arquitecturas distribuidas tradicionales, tales como, interoperabilidad entre distintas tecnologías, comunicación de aplicaciones escritas en cualquier lenguaje, transmisión HTTP (atraviesa firewalls) y presenta un mecanismo estándar de localización de servicios [6].

Una de las razones por las cuales se aplican los servicios web es que pueden aportar gran independencia entre la aplicación que los usan y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada [6].

Cuando una organización debe decidir si adoptar o no una arquitectura basada en servicios web debe tener en cuenta los numerosos desafíos que esto implica. Desde el punto de vista más abstracto, el objetivo es conseguir definir un entorno en el que las transacciones de los mensajes y los procesos de negocio se puedan llevar a cabo de manera segura de extremo a extremo. Una necesidad clara es la de asegurar los mensajes durante su tránsito, con o sin presencia de intermediarios [7].

Los servicios web cumplen el rol de conectores entre diferentes sistemas de información; esta característica trae muchos beneficios, pero a su vez compromete la seguridad [6]. La seguridad en los servicios web requiere el uso de políticas corporativas que puedan ser integradas con políticas de otras empresas externas y con la resolución de la confianza [7].

Los servicios web son portables, fiables y soportan conexiones, tanto sincrónicas como asincrónicas. Mediante el uso de estos servicios se pueden intercambiar diferentes informaciones, esto hace posible que se puedan ofrecer diferentes funcionalidades mediante la web que puedan agilizar y optimizar las operaciones de diversos organismos. Por estas razones es necesario que estos cumplan con estándares de seguridad, dado que las comunicaciones pueden estar comprometidas en el envío de mensajes con información sensitiva [6].

Con el fin de aumentar la protección de las transacciones de los mensajes y proteger los servicios web varias compañías de gran relevancia encabezadas por Microsoft e IBM desarrollaron una serie de especificaciones relacionadas con la seguridad. Dentro de estas especificaciones se encuentra WS-Security, estandarizadas por la Organización para el Avance de los Estándares de Información Estructurada (OASIS), por sus siglas en inglés, la cual constituye la base para la futura implementación de otras especificaciones [7].

Con el objetivo de minimizar los ataques y brindar seguridad a las aplicaciones que se desarrollan en el país, en la Universidad de las Ciencias Informáticas (UCI) se desarrolla el marco de trabajo Sauxe.

Este cuenta con una arquitectura Cliente-Servidor y utiliza servicios Web para brindar las distintas funcionalidades con las que da soporte a sus clientes.

En el marco de trabajo Sauxe se publican varios servicios web, los cuales pueden ser consumidos libremente por sistemas externos, a pesar de que los usuarios que realizan las peticiones no tengan permisos para realizarlas, incluso pueden ser consumidos por usuarios que no estén registrados dentro del marco de trabajo Sauxe. Esto trae como consecuencia que se vea afectada la confiabilidad, integridad y disponibilidad de los datos intercambiados por esta vía y que además disminuya las posibilidades de integración con otros sistemas.

Teniendo en cuenta lo antes expuesto surge el **problema a resolver:** ¿Cómo preservar la confidencialidad, integridad y disponibilidad de los datos intercambiados a través de servicios web en el marco de trabajo Sauxe para fortalecer su seguridad?

El cual se enmarca en el **objeto de estudio:** Seguridad en la comunicación basada en servicios web, delimitado por el **campo de acción:** Estándares de seguridad para la comunicación basada en servicios web.

Se define como **objetivo general:** Desarrollar un componente basado en el estándar WS-Security integrado al marco de trabajo Sauxe para fortalecer la seguridad de los datos intercambiados a través de servicios web.

Para hacer cumplir este objetivo general se plantearon los siguientes **objetivos específicos:**

- Construir el marco teórico conceptual de la investigación, relacionado con los estándares más utilizados para lograr la seguridad en servicios web y las soluciones, componentes o framework que realizan esta función.
- Analizar y diseñar la solución propuesta.
- Implementar la solución propuesta.
- Validar la solución propuesta.

Concluido el marco teórico de la investigación se arribó a la siguiente **idea a defender**: El desarrollo de un componente basado en el estándar WS-Security integrado al marco de trabajo Sauxe, permitirá fortalecer la seguridad de los datos intercambiados a través de servicios web.

Para el desarrollo de la investigación se trazaron las siguientes tareas:

- Realizar un estudio de los principales Estándares de Servicios Web existentes.
- Realizar un estudio sobre los Estándares para brindar seguridad a los Servicios Web profundizando en el estándar WS-Security.
- Investigar los principales Framework existentes y realizar una investigación sobre sus características.

- Realizar el análisis y diseño de la solución propuesta.
- Implementar la solución propuesta.
- Validar la solución propuesta.

La utilización de los siguientes métodos teóricos permitieron estudiar las características del objeto de investigación que no son observables directamente, facilitaron la construcción de modelos e idea a defender en la investigación y crearon las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, contribuyendo al desarrollo de las teorías científicas y para su ejecución se apoyan en el proceso de análisis y síntesis [8]. Los métodos utilizados son:

- Analítico - Sintético: el análisis de los elementos de la situación problemática se realiza relacionando estos elementos entre sí y vinculándolos con la situación problema como un todo. A su vez, la síntesis se produce sobre la base de los resultados dados previamente por el análisis. De esta forma se realizó el estudio de los principales sistemas [9].
- Inductivo-Deductivo: para deducir nuevos presupuestos confirmados mediante razonamientos inductivos en diferentes momentos de la investigación, sobre todo en la determinación de tendencias y posiciones teóricas relacionadas con la seguridad en aplicaciones web, así como en la interpretación de los instrumentos aplicados y la elaboración de las conclusiones parciales y generales [9].
- Modelación: para reproducir y analizar los nexos y las relaciones de los elementos que están inmersos en el objeto de estudio [9].

Estructura del documento

El presente documento se encuentra estructurado en tres capítulos. El primero, Fundamentación teórica, tiene como objetivo analizar los principales aspectos teóricos relacionados con los servicios web y los estándares utilizados para brindar seguridad a los servicios web. Además se analizan los conceptos asociados a la investigación que ayudan al entendimiento de la temática, se abordará sobre las soluciones más importantes, se analizará el modelo de desarrollo y las tecnologías y herramientas definidas.

En el capítulo dos se realizará el modelo conceptual y se describirán los requisitos funcionales y no funcionales. Se elaborará el diagrama de clases con estereotipos web correspondientes a la solución, y se describen los patrones de diseño y estilos arquitectónicos definidos para el desarrollo de la solución.

En el tercer capítulo se presentan los resultados obtenidos como resultado de la aplicación del estándar WS-Security al marco de trabajo Sauxe.

Finalmente se presentan las conclusiones y recomendaciones de la investigación.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En este capítulo se abordará acerca de los servicios web, las ventajas de su utilización, los estándares tanto de servicios web, como los de seguridad para los servicios web. Se analizarán los principales conceptos que serán tratados en la investigación. Además se realizará un estudio de los principales componentes, framework o librerías para aplicar seguridad a los servicios web, centrando la atención en los que utilizan el estándar WS-Security para analizar las características comunes con Sauxe. Por último, se puntualizarán las tecnologías y herramientas para desarrollar la solución.

1.1 Conceptos asociados al dominio de la investigación

- **Sistema de Información:** Un conjunto de componentes interrelacionados que reúne (u obtiene), procesa, almacena y distribuye información para apoyar la toma de decisiones y el control en una organización [3].

  También se puede definir a los SI como un conjunto de elementos interrelacionados con el propósito de prestar atención a las demandas de información de una organización, para elevar el nivel de conocimientos que permitan un mejor apoyo a la toma de decisiones y desarrollo de acciones [2].
- **Sistemas distribuidos**: Un sistema distribuido es una colección independiente de ordenadores que aparecen a los usuarios como un único sistema coherente. Conjunto de computadores interconectados igual que un sistema en red que comparten un estado ofreciendo una visión de sistema único igual que un sistema centralizado [10].
- **Servicio Web:** Componentes que hacen posible interactuar sistemas distribuidos los cuales pueden realizar desde las tareas más sencillas hasta las más complejas. Estos son aplicaciones modulares auto contenidas, las cuales pueden ser descritas, ubicadas e invocadas utilizando el mismo estándar web de transporte; que necesitan un lenguaje común para intercambiar datos, como lo es XML [6].

  También se puede definir, según la W3C, como: Un Servicio Web (Web Service [WS]) es una aplicación software identificada por un URI (Uniform Resource Identifier), cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML. Los servicios web hacen posible la interacción entre agentes software (aplicaciones) utilizando mensajes XML intercambiados mediante protocolos de Internet [11].
- **Estándares de seguridad:** Las tecnologías que apoyan los servicios web y la forma en que se utilicen. Dado que los servicios web basan el intercambio de mensajes en SOAP (Simple Object Access Protocol), pueden usar diferentes aplicaciones de transporte o protocolos, que

pueden o no tener un modelo de seguridad. También los servicios web permiten que existan intermediarios, los cuales pueden modificar los mensajes, por lo que el modelo de seguridad debe incluir un manejo de estos intermediarios [6].

- **Confidencialidad:** Propiedad del mensaje que implica que su contenido o parte del mismo no se encuentre a entidades, usuarios o procesos no autorizados [12].
- **Disponibilidad:** Se garantiza que los usuarios autorizados tengan acceso a la información y a los recursos relacionados con la misma, siempre que lo requieran [3].
- **Integridad:** Propiedad del mensaje que implica que su contenido no fue alterado [12].
- **Interoperabilidad**: Distintas aplicaciones, en lenguajes de programación diferentes, ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos [13].
- **Claim:** Declaración hecha por una entidad. Nombre, identidad, clave, grupo, etc [12].
- **Token de seguridad:** Representa un conjunto de claims [12].
- **ID Reference:** Posibilidad de referenciar elementos de seguridad dentro de un mensaje sin tener la necesidad de obtener el esquema completo del mensaje [12].
- **Framework:** Estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta [14].
- **Un framework Web**: Por tanto, podemos definirlo como un conjunto de componentes (por ejemplo clases en java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web .[14]
- **SSL Secure Sockets Layer:** Es un protocolo diseñado para permitir que las aplicaciones envíen y reciban información segura. Las aplicaciones que utilizan el protocolo Secure Sockets Layer saben cómo dar y recibir claves de cifrado con otras aplicaciones, así como la manera de cifrar y descifrar los datos enviados [15].
- **Kerberos:** Es un sistema de autentificación de usuarios, que establece el intercambio de mensajes entre el cliente y las estaciones que forman parte de su infraestructura. Usa una criptografía de claves simétricas, lo que significa que se emplea la misma clave para cifrar y para descifrar los mensajes [3].
- **X.509:** Es un formato estándar de certificado digital que se utiliza para la gestión y distribución segura de los certificados de firma digital a través de redes seguras de Internet. Describe dos diferentes niveles de autenticación. Autenticación simple, que se basa en el uso de una

contraseña para verificar la identidad del usuario y la autenticación fuerte, que utiliza las credenciales que se crean por medios criptográficos [16].

1.2 Estándares de Servicios Web

Los Servicios Web están basados fundamentalmente en estándares y protocolos abiertos, tales como:

- **eXtensible Markup Language (XML):** Es un metalenguaje basado en marcas y etiquetas que se utiliza para crear otros lenguajes. Puede definirse además, como un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos [17].
- **Simple Object Access Protocol (SOAP):** Protocolo estándar basado en XML para el intercambio de información entre aplicaciones en entornos descentralizados y distribuidos. Los mensajes XML que se desean enviar desde un punto a otro se codifican dentro de los sobres SOAP. Los posibles mensajes que se pueden incorporar dentro de SOAP son: llamadas y parámetros, datos enviados como respuesta tras la ejecución del servicio, información de error y de estado, datos adjuntos con formato distinto a XML datos binarios (codificación Base64) o un enlace a los datos. SOAP ofrece ventajas, entre ellas, que no depende de un sistema en concreto, puede comunicar aplicaciones que estén escritas en cualquier lenguaje, atraviesa firewalls corporativos y permite la interoperabilidad entre distintas aplicaciones [17].
- **Universal Description Discovery and Integration (UDDI):** Publicación, localización y enlazado de los servicios web. El objetivo de UDDI es permitir a las entidades de negocio que se descubran unas a otras, utilizando distintos criterios de búsqueda [17].
- **Web Service Description Language (WSDL):** Lenguaje para la descripción estándar de un servicio web. Es una tecnología XML que estandariza la representación de los parámetros de entrada y de salida que provienen de una invocación externa y permite a los diferentes clientes entender automáticamente cómo interactuar con el servicio web, tiene características como independencia de lenguaje y plataforma [17].

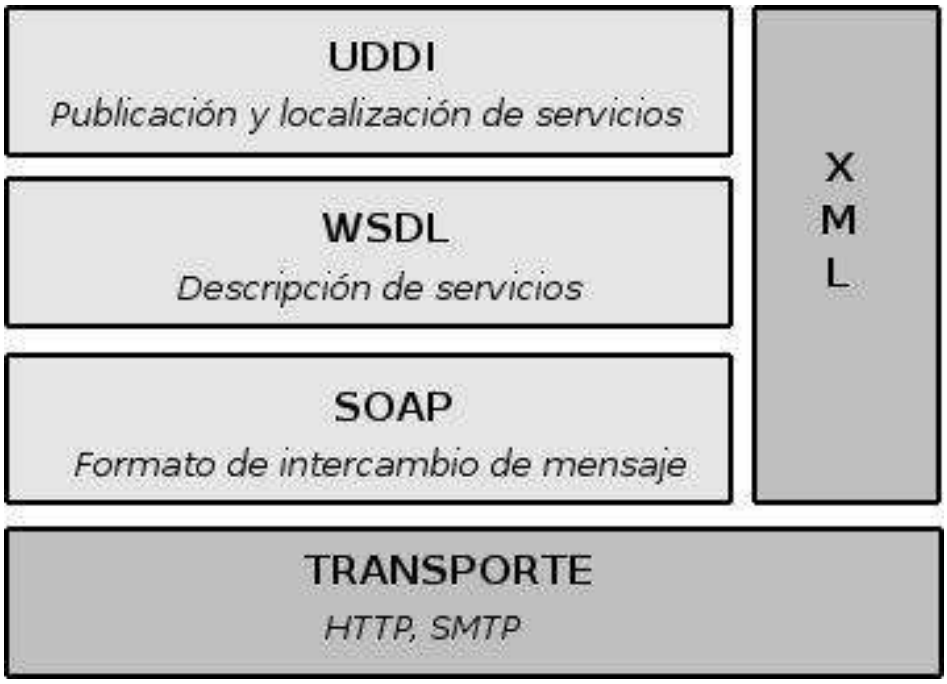


Ilustración 1: Estándares y protocolos de los servicios web

Fuente: [7]

1.3 Estándares de seguridad para Servicios Web

Algunos estándares desarrollados por la W3C y otras organizaciones basados en XML pueden garantizar la confidencialidad, integridad y disponibilidad de los servicios web, estos protocolos proveen mecanismos para cifrar, firmar digitalmente, autenticar y certificar documentos XML; algunos de los más importantes son:

- **WS-Policy:** Estándar definido por OASIS, que define un framework donde los servicios web, pueden expresar las limitaciones y requerimientos, los cuales pueden ser expresados como políticas que describen como los mensajes deben ser asegurados, incluyendo el nivel de seguridad de transporte. La intención es proveer suficiente información, a los participantes para tener un intercambio seguro de mensajes. Este framework provee la flexibilidad necesaria para definir estas políticas, con diferentes herramientas de seguridad [6].

  Para la aplicación de este estándar se debe tener un amplio dominio y conocimiento sobre el contrato del servicio, además de cualquier requisito, capacidad y preferencias adicionales. Intentar la integración con un servicio sin conocer todos estos detalles supone un altísimo gasto de esfuerzo del que será muy difícil obtener alguna recompensa [7].
- **WS-SecureConversation:** Define, describe y especifica como los contextos son establecidos. Por lo que determina extensiones para permitir, establecer y compartir contextos seguros,

mejorando la seguridad y el desempeño de los intercambios de mensajes posteriores [6]. WS-Security proporciona los mecanismos básicos sobre los cuales definir un marco de mensajería seguro, WS-SecureConversation introduce el concepto de seguridad contextual y describe cómo utilizarlo. Esto permite que se establezcan contextos y, potencialmente, que se realicen intercambios más eficientes de claves o de nuevo material de clave, consiguiendo el incremento global del rendimiento y de la seguridad en los intercambios posteriores [7].

Las metas principales de WS-SecureConversation son:

- Definir cómo se establecen los contextos de seguridad [7].
- Especificar cómo se calculan y distribuyen las claves derivadas [7].

- **WS-Trust:** Define extensiones que se construyen sobre WS-Security y que consisten en ampliar la capacidad de los mecanismos de seguridad definidos por ésta, permitiendo la solicitud, emisión e intercambio de tokens de seguridad y la gestión de las relaciones de confianza. El objetivo principal de esta especificación es, por lo tanto, habilitar a los sistemas para que puedan crear patrones de intercambio confiados de mensajes. Esta confianza se representa mediante el intercambio e intermediación de los tokens de seguridad. La especificación define un protocolo agnóstico que permite emitir, renovar e intercambiar tokens de seguridad [7].
- **WS-Federation:** Define los mecanismos necesarios para conseguir la federación de dominios de seguridad distintos o similares y lo consigue permitiendo e intermediando la confianza de las identidades, atributos, y autenticación de los servicios web participantes. La especificación WS-Federation define un modelo y un marco de trabajo general para conseguir la federación. Tal y como se expresa en la especificación, se irán generando documentos, denominados profiles (perfiles) los cuales detallarán cómo los distintos solicitantes encajan en este modelo. Un objetivo prioritario de esta especificación habilitar la federación de la información de las identidades, atributos, autenticación y autorización [7].
- **WS-Privacy**: Esta especificación utiliza los mecanismos definidos por la especificación WS-Security, WS-Policy y WS-Trust para permitir la transmisión de las políticas de privacidad. Estas políticas de privacidad serán definidas por la organización propietaria del servicio web y los mensajes SOAP recibidos deberán ser conformes con dichas políticas de privacidad. WS-Privacy señalará cómo incluir afirmaciones de política de privacidad mediante WS-Policy y utilizará WS-Trust para evaluar las sentencias de privacidad encapsuladas en los mensajes SOAP contra las preferencias de los usuarios y las políticas de la organización [7].

- **WS-Security:** Estándar definido por OASIS el cual provee las funciones básicas, definiendo mecanismos para asegurar: autenticidad, integridad y confidencialidad de los mensajes. La especificación propone un conjunto de estándares, para la extensión de SOAP 1.1 y SOAP 1.2 que pueden ser utilizados cuando se construyen servicios web seguros. Adicionalmente, describe como codificar tokens seguros y agregarlos a los mensajes SOAP. Estos tokens pueden ser ampliados y utilizados con otros protocolos de servicios web para orientarlos en una variedad de requerimientos de seguridad [6].

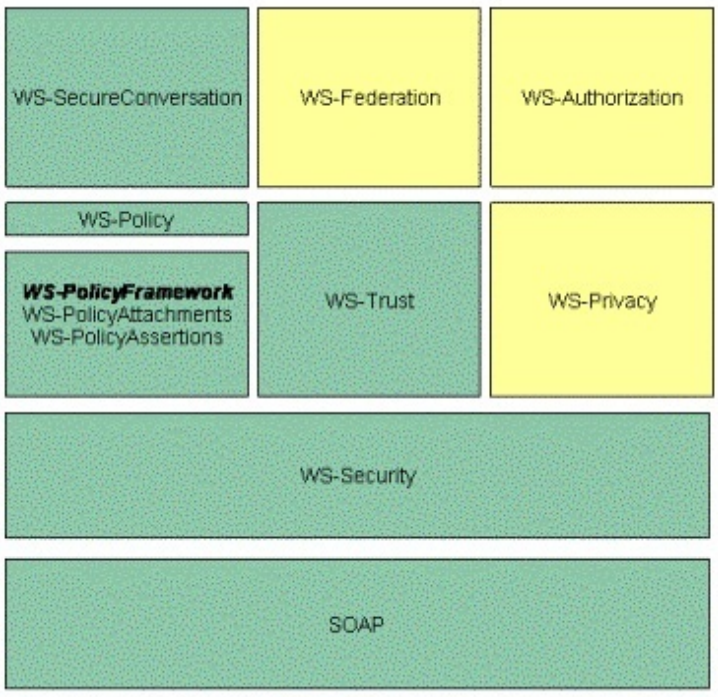


Ilustración 2: Estándares para la seguridad a los servicios web

Fuente: [7]

1.4 Características generales de WS-Security

WS-Security especifica cómo firmar y cifrar mensajes SOAP. Uno de los puntos de partida de este estándar es que los mecanismos de seguridad basados en http que se usan en muchos casos para intercambiar mensajes sobre SOAP, no son suficientes. Es claro que los principios de autenticación, integridad y confidencialidad deben potencializarse en muchos puntos y saltos de la comunicación, y no solamente extremo a extremo tal como lo hacen protocolos como SSL, TLS. WS-Security retoma los elementos de seguridad pero usando las especificaciones y estándares tradicionales, es decir, aprovecha las fortalezas de estándares como X.509, Kerberos, XML Encryption y XML Signature,

pero WS-Security adiciona mecanismos a estos estándares para que puedan ser incluidos en mensajes SOAP. El estándar define elementos para los encabezados SOAP y puede utilizar XML Signature para definir la manera como se firmará el mensaje (las claves usadas y la firma), otra información puede ser cifrada con XML Encryption e incluirse en su encabezado. WS-Security transfiere las credenciales de los clientes a partir del *UserNameToken, o* cuando se utilizan datos binarios se usa BinarySecurityToken [17].

WS-Security es flexible y su diseño constituye la base para la creación de modelos de seguridad más complejos incluyendo PKI, Kerberos y SSL. En particular, WS-Security proporciona soporte para múltiples tokens de seguridad, múltiples dominios de confianza, múltiples formatos de firma y múltiples tecnologías de cifrado. Esta especificación proporciona 3 elementos principales:

1. Propagación de tokens de seguridad.
2. Confidencialidad de los mensajes.
3. Integridad de los mensajes [7].

La meta de WS-Security es permitir que las aplicaciones construyan intercambios seguros de mensajes SOAP. Esta especificación está destinada a proporcionar un conjunto flexible de mecanismos que puedan ser utilizados para construir una amplia gama de protocolos de seguridad; en otras palabras, esta especificación intencionadamente no describe explícitamente protocolos de seguridad sino que define las primitivas para poder hacerlo [7].

Dada las características y potencialidades que brinda WS-Security y la importancia de asegurar el envío de los mensajes utilizando servicios Web, además de constituir la base para la implementación futura de otros estándares de seguridad se decide aplicar el mismo al marco de trabajo Sauxe.

1.5 Soluciones o componentes para aplicar seguridad a los Servicios Web utilizando WS-Security

- **AXIS2:** Axis2 es la versión mejorada del contenedor de Web Services AXIS. El proyecto ha evolucionado independientemente de la primera versión debido a que implementa especificaciones diferentes. AXIS originalmente se convirtió en uno de los contenedores más extendidos para implantar soluciones basadas en Web Services o que proveían acceso a sistemas preexistentes mediante Web Services SOAP. AXIS2 implementa la especificación JAX-WS del Java Comunity Process, WS-Messging y WS-Security, dispone de una estructura modular que permite ampliar la funcionalidad básica del sistema en el futuro gracias al desarrollo de módulos adicionales (al estilo de los plugins de Eclipse). Soporta protocolos de Web Services más actuales que SOAP como por ejemplo REST [18].

Axis2, entre las especificaciones que implementa está WS-Security la cual constituye el objetivo de esta investigación pero está desarrollada en el lenguaje de programación Java y no presenta compatibilidad con el marco de trabajo Sauxe.

- **WSO2 Web Services Framework para C++ (WSO2 FSM / C + +):** Es una extensión del framework de la WSO2, Web Services Framework para C, compatible con los estándares de grado empresarial, de código abierto, biblioteca de C++ para proporcionar y consumir servicios Web en C++. WSO2 FSM / C++ agrega las características necesarias a WSO2 FSM / C, por medio de una envoltura de C++ para la biblioteca C. FSM / C++ es una solución completa para crear y desplegar servicios web, y es el Envoltorio C++ para la biblioteca de C con la más amplia gama de especificaciones para brindar seguridad a los servicios web implementadas, incluyendo WS-Addressing, WS-Policy , WS-Security, WS-SecurityPolicy, WS y WS-ReliableMessaging [19].

  WSO2 Web Services Framework para C++, al igual que Axis2 cumple con la implementación de WS-Security y propicia la seguridad necesaria los servicios web, pero el lenguaje de desarrollo es distinto al utilizado en Sauxe y no es compatible con el del marco de trabajo.

- **WSO2 Web Services Framework para PHP (WSO2 FSM / PHP):** Es una fuente abierta, de nivel empresarial, la extensión de PHP para proporcionar y consumir servicios web en PHP. WSO2 FSM / PHP es una solución completa para crear y desplegar servicios Web, y es la única extensión de PHP con la más amplia gama de especificaciones para brindar seguridad a los servicios web implementaciones. Las características claves incluyen, servicios de seguros y clientes con WS-Security apoyo, archivos adjuntos binarios con masa máxima de despegue automático de generación de WSDL (modelo del primer código), modo WSDL para los servicios y los clientes (modelo de contrato en primer lugar) y la interoperabilidad con .NET y J2EE. WSO2 FSM / PHP se libera con Apache License, Version 2.0 y se basa en WSO2 FSM / C [20].

  WSO2 Web Services Framework para PHP, este framework al igual que los anteriores implementa el estándar objetivo de la investigación, el lenguaje de programación utilizado para su desarrollo es compatible con el utilizado en el desarrollo del marco de trabajo Sauxe, pero no gestiona los procesos de los servicios web de la misma forma que Sauxe.

- **Valoración de las soluciones y componentes estudiados:**

Los sistemas analizados presentan las características necesarias para brindar seguridad a los servicios web. Independientemente de que cubren escenarios comunes a los de la solución que se

persigue, no presentan compatibilidad con el marco de trabajo Sauxe y no brindan solución a la problemática planteada.

1.6 Modelo de desarrollo

El modelo de desarrollo de software varía en cada proyecto atendiendo a las características de cada uno tales como: tipo de proyecto, calidad y experiencia del personal, las especificaciones del software entre otros. Por política del Centro de Informatización de la Gestión de Entidades (CEIGE), en el Departamento de Tecnología se utiliza el Modelo de Desarrollo Orientado a Componentes del proyecto ERP-Cuba. Presenta características como:

- **Orientado a Componentes:** el software se desarrolla en varios componentes independientes que cuando se unen forman el sistema completo [21].
- **Iterativo e Incremental:** el sistema tendrá tantas iteraciones como sea necesario y en cada iteración habrá un incremento y mejoramiento de las funcionalidades del sistema [21].
- **Responde al nivel 2 CMMI:** cumple con los requisitos propuestos para el nivel 2 de CMMI para las áreas de: Medición y análisis, Monitoreo y control de proyecto, Planeación de proyecto, Aseguramiento de la calidad del proceso y el producto, Administración de requisitos, Administración de la configuración, Administración de acuerdo con proveedores [21].

1.7 Tecnologías para el desarrollo

Para la investigación y desarrollo se utilizaron las siguientes tecnologías:

Sauxe 2.2:

Sauxe es un Marco de Trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo [22].

Zend Framework 1.11:

Es un framework open source para PHP desarrollado por Zend, empresa encargada de la mayor parte de las mejoras hechas a PHP, por lo que se podría decir que es el framework “oficial”. Implementa el patrón MVC, es 100% orientado a objetos y sus componentes tienen un bajo acoplamiento por lo que pueden ser usados en forma independiente. Un punto importante es que brinda un estándar de codificación [23].

Doctrine 2.4:

Doctrine es un mapeador de objeto relacional (ORM) para PHP 5.2.3+ que se encuentra en la parte superior de una capa poderosa de abstracción de base de datos (DBAL por sus siglas en inglés). Una de sus principales características es la opción de escribir las consultas de base de datos en un

dialecto orientado a objetos de propiedad SQL llamado Doctrine QueryLanguage (DQL), lo que proporciona a los desarrolladores una alternativa a SQL, que mantiene la flexibilidad sin necesidad de duplicar el código innecesario [22].

ExtJS 2.2:

ExtJS es una librería JavaScript ligera y de alto rendimiento, construida para el desarrollo veloz de aplicaciones Web, compatibles con la mayoría de navegadores. Utiliza técnicas como Ajax, DHTML y manipulación del DOM. ExtJS incluye un conjunto de controles para el desarrollo de interfaces en los desarrollos web [22].

Unified Modeling Language (UML) 8.0:

UML es un lenguaje que permite modelar, construir y documentar los elementos que forman un software orientado a objetos; es la sucesión de una serie de métodos de análisis y diseño. Se ha convertido en el estándar de facto de la industria, ya que, a pesar de no tener un respaldo formal de una autoridad institucional o un organismo de normalización, es ampliamente usado [22].

PHP 5.2.6:

Lenguaje de programación interpretado, diseñado originalmente para la creación de Página web dinámicas. Es usado principalmente en la 333interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+ [24].

1.8 Herramientas para el desarrollo

Servidor web Apache 2.0:

El proyecto Apache HTTP Server es un esfuerzo por desarrollar y mantener un servidor HTTP de código abierto para sistemas operativos modernos, incluyendo UNIX y Windows NT. El objetivo de este proyecto es proporcionar un servidor seguro, eficiente y extensible que proporcione servicios HTTP en sincronización con los estándares HTTP actuales [25].

PostgreSQL 9.1:

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando [26].

Visual Paradigm 5.0:

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos [27].

RapidSVN 012.0:

RapidSVN es un cliente gráfico para Subversion, un programa de control de versiones sustituto de CVS. Permite acceder a direcciones SVN, subir y descargar contenido y sincronizarlo con el servidor original, comprobar su estado, crear y fusionar direcciones, etc. Además tiene la ventaja de funcionar en varias plataformas y de disponer de un completo manual en línea [15].

NetBeans 7.0:

NetBeans IDE es una aplicación de código abierto ("open source") diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java. NetBeans IDE dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y sus funcionalidades son ampliables mediante la instalación de packs [28].

Conclusiones parciales

Mediante el estudio realizado en este capítulo se arribó las siguientes conclusiones:

- El estudio de las especificaciones para brindar seguridad a los servicios web demostró que todos los estándares definidos en la misma proporcionan la seguridad requerida para los Servicios Web, destacando al estándar WS-Security como la base para la futura implementación del resto de las especificaciones.
- Se realizó un estudio del estado del arte sobre soluciones que implementan el estándar WS-Security, mostrando que ninguna de las soluciones estudiadas es compatible con el marco de trabajo Sauxe.
- Mediante la definición de las tecnologías y herramientas a emplear para el desarrollo de la solución fue posible determinar los elementos que serán necesarios para responder a las necesidades del marco de trabajo Sauxe.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

En este capítulo se profundizará en las fases de Modelación del Negocio, Requisitos y Análisis y Diseño. Durante las mismas ser generarán los artefactos correspondientes a cada una de ellas, definiendo los principales conceptos asociados a la investigación, se identificarán y definirán los requisitos funcionales y no funcionales, y se realizarán el diagrama de clases del diseño y diagrama de secuencia y modelo de datos.

2.1 Modelo Conceptual

Ilustra conceptos en el dominio del problema, explica cuales son y cómo se relacionan estos conceptos en la descripción del problema [12]. A continuación se definen los conceptos fundamentales tratados en la propuesta de solución:

- **WS-Security:** define un conjunto de extensiones a SOAP para mantener la integridad y confidencialidad de los mensajes [12].
- **Escenario de Seguridad:** tecnologías que apoyan los servicios web y la forma en que se utilizan [6].
- **Componente Gestionar Servicio Web:** componente encargado de adicionar y eliminar paquetes de servicios web dentro del marco de trabajo Sauxe
- **Servicio Web:** aplicaciones modulares auto contenidas, las cuales pueden ser descritas, ubicadas e invocadas utilizando el mismo estándar web de transporte; que necesitan un lenguaje común para intercambiar datos, como lo es XML [6].
- **Mensaje SOAP:** es un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto [29].
- **ProxiServer:** fichero php genérico que se encarga de brindar los servicios web.
- **Sistema externo:** cliente o usuario que trata de consumir los servicios web.
- **Escenario_serv:** fichero utilizado para guardar la configuración de cada servicio web.

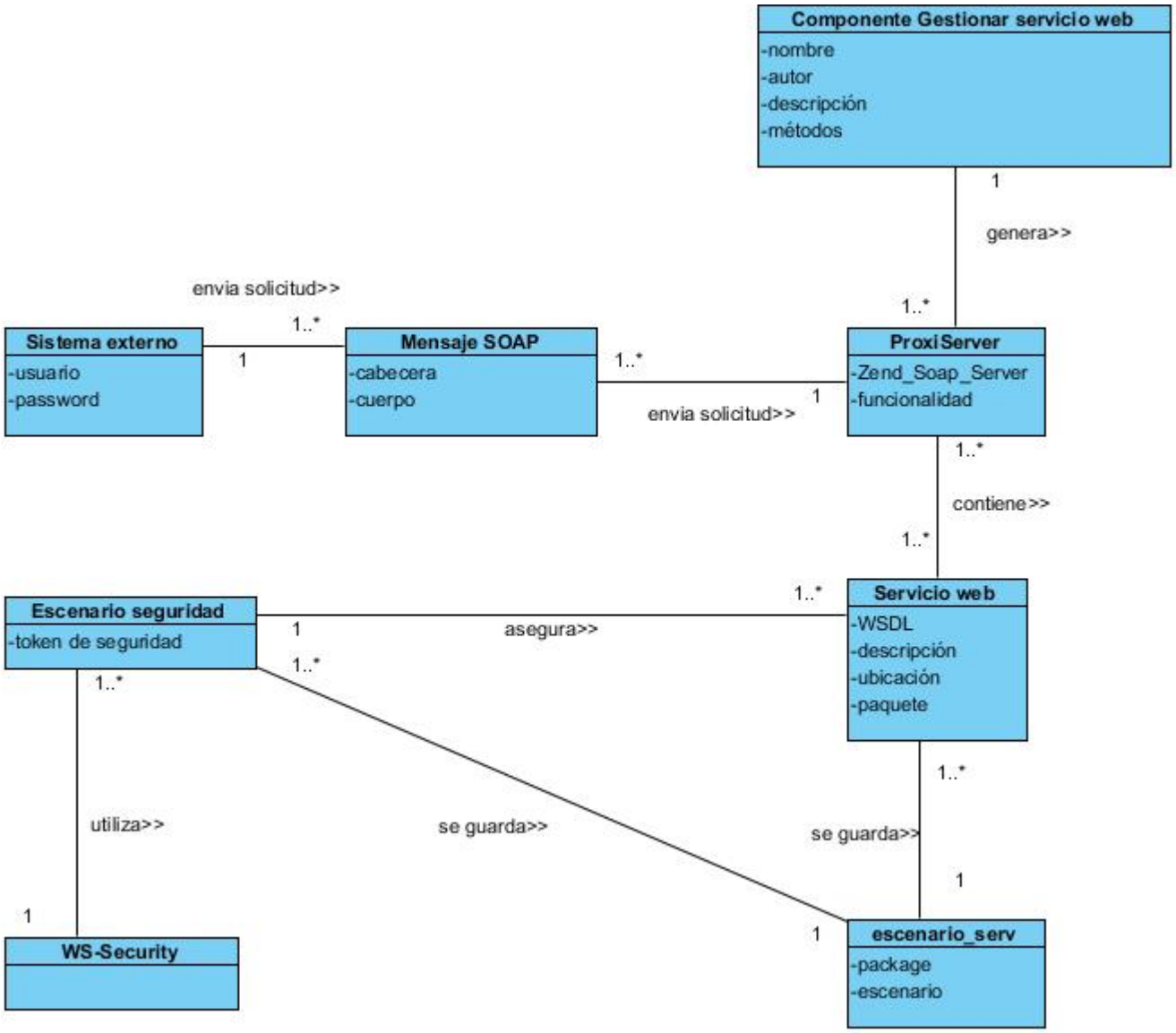


Ilustración 3: Modelo Conceptual

Fuente: Elaboración propia

2.2 Especificación de requisitos de software.

Los requisitos de software determinan lo que hará el sistema, restricciones sobre su operación e implementación. También se aplica a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, una norma o una especificación [30].

2.2.1. Técnicas de captura de requisitos

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema. El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa [31]. A continuación se describen las técnicas de captura de requisitos utilizadas:

- **Entrevistas:** Las entrevistas son la técnica de captura más utilizada, y de hecho son prácticamente inevitables en cualquier desarrollo ya que son una de las formas de comunicación más naturales entre personas [32]. Para la realización de dicha técnica se realizaron reuniones con el cliente donde quedaron definidos los elementos a tener en cuenta para el desarrollo del sistema.
- **Brainstorming o tormenta de ideas:** Es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios [32]. Para la realización de dicha técnica se realizaron reuniones donde participaron varios especialistas del departamento de Tecnología definiendo los requisitos funcionales sobre la base de los elementos definidos por el cliente.

2.2.2. Técnicas para la definición de requisitos

Para la actividad de definición de requisitos en el proceso de ingeniería de requisitos hay un gran número de técnicas propuestas. A continuación se describen las técnicas utilizadas.

- **Plantillas o patrones:** Esta técnica tiene por objetivo describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta, menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado estructurado, el trabajo de rellenar las plantillas y mantenerlas, puede ser demasiado tedioso [31].

2.2.3. Técnicas de validación de requisitos

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos

de la etapa de definición de requisitos son correctos [31]. Las técnicas usadas para validar los requisitos definidos son:

- **Prototipo:** Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final. Para la validación del requisito funcional Regular escenario de seguridad a cada servicio web se realizó un prototipo de interfaz el cual fue aprobado en reuniones con el cliente [31].
- **Revisiones técnicas formales:** En reuniones con especialistas del departamento de Tecnología, se revisaron las especificaciones de cada requisito, aprobando así los requisitos funcionales y no funcionales obtenidos.

2.2.4. Requisitos funcionales del sistema

Definición de los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo se debe comportar ante situaciones particulares, deben describir los servicios que hay que proporcionar con todo detalle [30].

Listado de requisitos funcionales para el Componente para la gestión de la seguridad de los servicios web en el marco de trabajo Sauxe.

1 Gestionar escenarios de seguridad de los servicios web que brinda un sistema.

1.1. Asignar escenarios de seguridad a cada servicio web.

1.2. Listar servicios web.

1.3. Identificar el escenario correspondiente al servicio web.

1.4. Relacionar escenario de seguridad a un servicio web.

2 Gestionar consumo servicios web.

2.1. Validar los datos.

2.2. Consumir el servicio web.

Especificación de los requisitos funcionales

- **RF 1: Gestionar escenarios de seguridad de los servicios web que brinda un sistema.**

- **RF 1.1: Regular escenarios de seguridad a cada servicio web.**

Tabla 1: RF Regular escenario de seguridad a cada servicio web.

Fuente: Elaboración propia.

Precondiciones	Se ha registrado al menos un servicio web. Existe al menos un escenario de seguridad que asignar.
Flujo de eventos	
Flujo básico 1 Regular escenarios de seguridad a cada servicio web.	
1	El usuario selecciona un servicio web de la lista de los mismos.
2	Se habilita el botón Asignar Escenario
3	El usuario presiona dicho botón
4	El sistema muestra la interfaz Asignar escenario.
5	El usuario selecciona el escenario de seguridad a asignar
6	Se oprime el botón Aceptar.
7	El sistema valida los campos.
8	El sistema asigna el escenario al servicio web.
9	Se salva la configuración de ese servicio web en el fichero escenario_serv.xml
10	El sistema muestra un mensaje “El escenario fue asignado correctamente”
11	Concluye el requisito.
Pos-condiciones	
1	Se asignó un nivel de seguridad a un servicio web.
Flujos alternativos	
Flujo alternativo 6.a No se selecciona ningún escenario	
1	El sistema muestra un mensaje de error “Debe seleccionar al menos un escenario de seguridad”.
2	Volver al paso 4 del flujo básico 1.

Pos-condiciones		
1	N/A	
Validaciones		
1	Debe seleccionarse al menos un escenario.	
Conceptos	**Escenario seguridad**	Visibles en la interfaz: nombre Utilizados internamente: Token seguridad.
	Escenario_serv	Visibles en la interfaz: Utilizados internamente: Package Escenario
	Servicio web	Visibles en la interfaz: Descripción Ubicación Paquete Utilizados internamente: WSDL
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

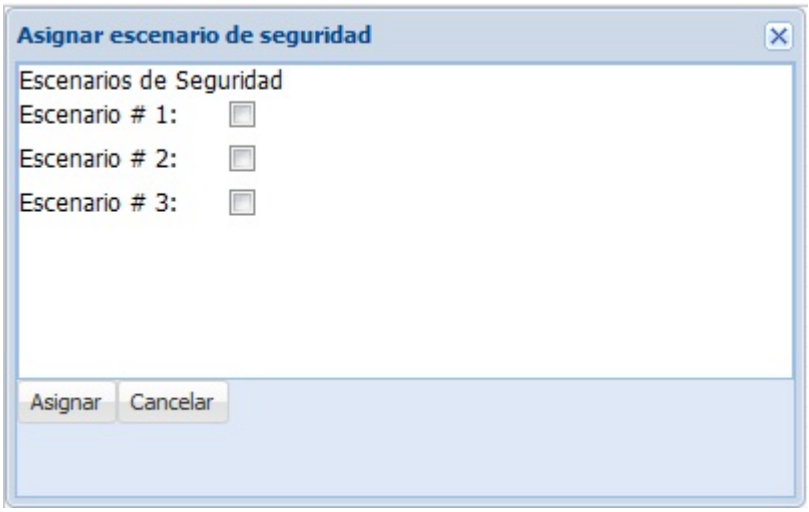


Ilustración 4 Prototipo de interfaz RF 1.1 Regular escenarios de seguridad a cada servicio web

Fuente: Elaboración propia.

- **RF 1.2: Listar servicios web**

Tabla 2: Listar servicio web.

Fuente: Elaboración propia.

Precondiciones	Se ha registrado al menos un servicio web.
Flujo de eventos	
Flujo básico Listar Servicios Web	
1	El usuario accede a la interfaz Gestionar servicio web.
2	El sistema muestra los servicios web que tiene registrado.
3	Concluye el requisito
Pos-condiciones	
1	Se muestra un listado con los servicios web registrados.
Flujos alternativos	
1 N/A	
Validaciones	
1 N/A	

Conceptos	**Servicio web**	Visibles en la interfaz: Ubicación Descripción Paquete Utilizados internamente: WSDL
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

N/A

- **RF 1.3: Identificar el escenario de seguridad correspondiente al servicio web.**

Tabla 3: Identificar el escenario de seguridad correspondiente al servicio web.

Fuente: Elaboración propia.

Precondiciones	Se ha registrado al menos un servicio web. Existe al menos un escenario de seguridad que asignar.
Flujo de eventos	
Flujo básico 1 Identificar el escenario de seguridad correspondiente al servicio web.	
1	Se define el paquete de servicios web que va a ser adjuntado.
2	El sistema accede al fichero escenario_serv.xml que cuenta con la configuración de los escenarios y los servicios web.
3	El sistema obtiene todas las configuraciones guardadas en el fichero escenario_serv.xml correspondiente al escenario de seguridad.
4	El sistema busca la configuración del escenario de seguridad que le corresponde al paquete de servicios web definido.
5	El sistema guarda temporalmente el resultado de la búsqueda realizada.
6	Concluye el requisito.

Pos-condiciones		
2	Se obtiene los parámetros del escenario correspondiente al servicio web seleccionado.	
Flujos alternativos		
Flujo alternativo 3.a No se encuentra la configuración correspondiente al escenario de seguridad en el fichero escenario_serv.xml.		
1	El sistema muestra un mensaje de error “El servicio web no tiene asignado ningún escenario de seguridad”.	
2	Volver a la descripción del requisito Regular escenario de seguridad de los servicios web que brinda un sistema.	
Pos-condiciones		
1	N/A	
Validaciones		
1	Debe seleccionarse al menos un escenario.	
Conceptos	**Escenario seguridad**	Visibles en la interfaz: N/A Utilizados internamente: Token seguridad.
	escenario_serv.xml	Visibles en la interfaz: N/A Utilizados internamente: Package Escenario
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario.

N/A

- **RF 1.4: Relacionar escenario de seguridad a un servicio web.**

Tabla 4: Relacionar escenario de seguridad a un servicio web.

Fuente: Elaboración propia.

Precondiciones		Debe existir al menos un escenario de seguridad implementado
Flujo de eventos		
Flujo básico Relacionar escenario de seguridad a un servicio web.		
1	El usuario define el nivel de seguridad que va a tener un servicio web (ver descripción del requisito 2.1).	
2	El sistema adjunta el código del escenario a la cabecera del mensaje SOAP del Servicio Web	
3	Concluye el requisito.	
Pos-condiciones		
1	El Servicio Web queda asegurado con el escenario de seguridad seleccionado.	
Flujos alternativos		
Flujo alternativo		
1	N/A	
Pos-condiciones		
1	N/A	
Validaciones		
1	N/A	
Conceptos	**Escenario de seguridad**	Visibles en la interfaz: N/A Utilizados internamente: Tokens de seguridad
	Servicio web	Visibles en la interfaz: Descripción

	Ubicación Paquete Utilizados internamente: WSDL
Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo elemental de interfaz gráfica de usuario.

N/A

- **RF 2: Gestionar consumo servicios web.**
  - **RF 2.1: Validar datos.**

Tabla 5: Validar datos
Fuente: Elaboración propia.

Precondiciones	Existe un sistema externo conectado al marco de trabajo Sauxe.
Flujo de eventos	
Flujo básico Validar los datos.	
1	El sistema externo solicita el consumo de un servicio web
2	A través de un mensaje SOAP el sistema externo envía los datos correspondientes al escenario de seguridad definido para el servicio web que trata de consumir,
3	El sistema obtiene los datos enviados por el sistema externo.
4	El sistema compara que los datos recibidos se correspondan con los datos definidos por el escenario de seguridad.
5	El sistema valida los datos.
6	Concluye el requisito
Pos-condiciones	
2	Se validan los datos enviados.

Flujos alternativos		
Flujo alternativo 5.a Los datos recibidos no se corresponden con los definidos por el escenario de seguridad.		
1 El sistema muestra un mensaje de error" Los datos enviados no corresponden al servicio web".		
2 Volver al paso 1 del flujo básico.		
Flujo alternativo 6.b Los datos validados no tienen permisos sobre el servicio web		
1 El sistema muestra un mensaje de error" No tiene permiso sobre este servicio web"		
2 Volver al paso 1 del flujo básico.		
Validaciones		
1 N/A		
Conceptos	**Sistema externo**	Visibles en la interfaz: N/A Utilizados internamente: Usuario Password
	Escenario seguridad	Visibles en la interfaz: N/A Utilizados internamente: Token seguridad.
	Mensaje SOAP	Visibles en la interfaz: N/A Utilizados internamente: Código de estándar.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	
Prototipo elemental de interfaz gráfica de usuario.		
N/A		

- **RF 2.2: Consumir servicio web.**

Tabla 6: Consumir servicio web.

Fuente: Elaboración propia.

Precondiciones		Debe haber una solicitud para consumir un servicio web
Flujo de eventos		
Flujo básico Consumir el servicio web.		
4	El sistema valida los datos (ver descripción del requisito 2.1).	
5	A través de mensajes SOAP el sistema permite el consumir del servicio web.	
6	Concluye el requisito.	
Pos-condiciones		
2	Se consume el servicio web.	
Flujos alternativos		
Flujo alternativo		
2	N/A	
Pos-condiciones		
2	N/A	
Validaciones		
2	N/A	
Conceptos	**Mensaje SOAP**	Visibles en la interfaz: N/A Utilizados internamente: Código del estándar Funcionalidades
	Servicio web	Visibles en la interfaz: N/A Utilizados internamente: WSDL
Requisitos especiales	N/A	
Asuntos pendientes	N/A	
Prototipo elemental de interfaz gráfica de usuario.		

N/A

2.2.5. Requisitos no funcionales

Definen cualidades o atributos globales del sistema, o Establecen restricciones sobre el producto desarrollado, el proceso de desarrollo o externas. No están generalmente relacionados con la funcionalidad del sistema [33].

Durante el proceso de captura de los requisitos se detectaron Requisitos no Funcionales (RnF) y fueron complementados con los RnF detectados en el proyecto Sistema de Gestión Integral de Seguridad (ACAXIA) descritos en el documento *"Especificación de Requisitos de Software"*, el cual se encuentra en la documentación de dicho proyecto.

Eficiencia

- RnF 1 Los tiempos de respuesta y velocidad de procesamiento de la información no serán mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

Soporte.

- RnF 2 La aplicación contará antes de su puesta en marcha con un período de pruebas, se le dará mantenimiento, configuración y se brindará el servicio de instalación.
- RnF 3 Los desarrolladores deben utilizar para el desarrollo de la solución el estándar de codificación elaborado en el Departamento de Tecnología.

Restricciones de diseño.

- RnF 4 El lenguaje de programación a utilizar para desarrollar el sistema debe ser php para la lógica del negocio y ExtJS para la capa de presentación.
- RnF 5 Las herramientas a utilizar para desarrollar el sistema deben ser PostgreSql 8.3 como gestor de base de datos y Pgadmin III como cliente, Doctrine para la capa de acceso a datos y ZendExt Framework para la lógica del negocio.
- RnF 6 El sistema debe ser multiplataforma, específicamente en Linux y Windows.

Estándares Aplicables

- RnF 8 El sistema debe utilizar el estándar Lenguaje de Enmarcado de Aserciones de Seguridad (SAML) para la estandarización del proceso de autenticación.

- RnF 9 El sistema debe utilizar el estándar Control de Acceso Basado en Roles (RBAC) para la estandarización del proceso de autorización.
- RnF 10 El sistema debe utilizar el estándar WS-Security para garantizar la seguridad de los servicios web.

Ambiente

Para que el sistema funcione es necesario garantizar los siguientes requisitos de software:

Para el cliente:

- RnF 1 Navegador Mozilla Firefox.
- RnF 2 Sistema operativo Linux y Windows 98 o superior o.

Para el servidor:

- RnF 3 Sistema operativo Windows Advancer Server (2000 o superior) o Linux en cualquiera de sus distribuciones.
- RnF 4 Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible, este debe estar configurado con la extensión “pgsql” incluida.
- RnF 5 Un servidor de base de datos PostgreSQL 8.0 o superior.

Para que el sistema funcione es necesario garantizar los siguientes requisitos de hardware:

Para el servidor:

- RnF 6 Requerimientos mínimos: Procesador Pentium III a 1GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- RnF 7 Al menos 40Gb de espacio libre en disco duro.
- RnF 8 Debe tener una tarjeta de red.

Para el cliente:

- RnF 9 Requerimientos mínimos: Procesador Pentium II a 133Mhz con 128 Mb de memoria RAM.
- RnF 10 Debe tener una tarjeta de red.

2.3 Modelo Diseño

2.3.1. Estilos arquitectónicos

El estilo arquitectónico definido por el Departamento de Tecnología donde se desarrolla el marco de trabajo Sauxe es el estilo de Arquitectura en Capas que constituye una organización jerárquica tal que cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior, al dividir un sistema en capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. La división de un sistema en capas facilita el diseño modular, en la que cada capa encapsula un aspecto concreto del sistema y permite además la construcción de sistemas débilmente acoplados, lo que significa que si se minimizan las dependencias entre capas, resulta más fácil sustituir la implementación de una capa sin afectar al resto del sistema [34]. Donde se han definido 3 capas fundamentales:

- **Capa de Funcionamiento:** en esta capa se emplea el patrón de arquitectura Modelo Vista Controlador (MVC), así como el elemento de Acceso de Respuesta Rápida el cual brinda informaciones de forma vertiginosa violando las restricciones que impone el estilo en capas propuesto. De forma vertical al modelo descrito hasta este momento, estará el módulo de cacheo del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación Web [22].

2.3.2. Patrones arquitectónicos

El patrón arquitectónico utilizado es el Modelo-Vista-Controlador (MVC), separa los datos de la aplicación, la interfaz de usuario y la lógica de negocio en tres componentes:

Modelo: Agrupa los estados de la aplicación, responde a los requerimientos, muestra la funcionalidad de la aplicación, notifica los cambios a la Vista. Está compuesto por datos, funcionalidades y reglas del negocio que sirven para la comunicación con el framework Doctrine [35].

Vista: Interpreta el modelo, solicita actualizaciones del modelo, envía las acciones del usuario al Controlador, permite al Controlador seleccionar las Vistas [35].

Controlador: Define el comportamiento de la aplicación, mapea las acciones del Usuario a actualizaciones del Modelo, selecciona la Vista de respuesta y muestra la información del Modelo al Usuario [35].

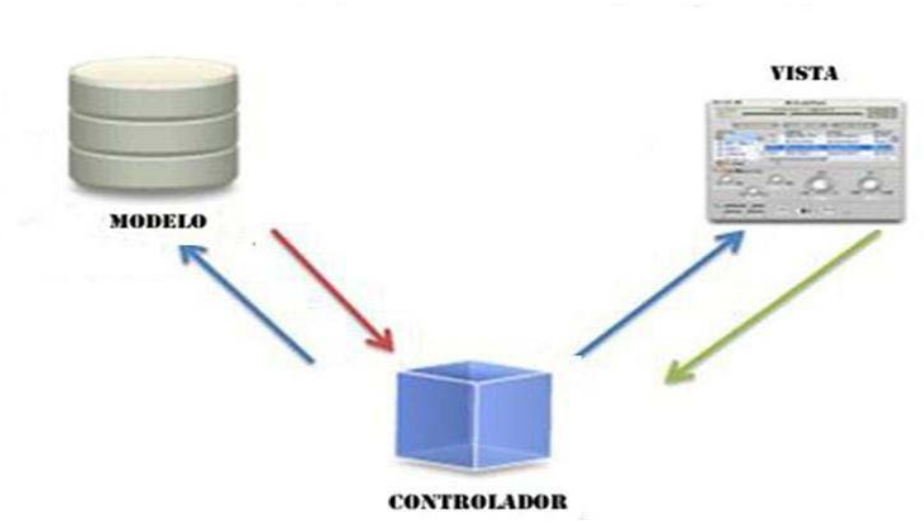


Ilustración 5: Modelo-Vista-Controlador.

Fuente: [22]

2.3.3. Patrones de diseño

GRASP (General Responsibility Assignment Software Patterns):

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada [36]. Se puede ver el uso de este patrón en la clase ServiceModel.php.

Creador: El creador guía la asignación de responsabilidades relacionadas a la creación de objetos, una tarea muy común en sistemas orientados a objetos. El intento básico del patrón creador es encontrar un creador que necesite estar conectado al objeto creado en un evento en particular [36]. Este patrón se pone de manifiesto entre la clase GestwebserviceController.php pues en ella se crean objetos de las clases ServiceModel.php y GenerateModel.php.

Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión [36]. Este patrón queda evidenciado en la clase GestwebserviceController que se delega a la clase ServiceModel.php la función de gestionar los datos en el fichero escenario_serv.xml.

Bajo acoplamiento: El uso de éste patrón se evidencia en la poca relación existente entre las clases que conforman el componente [36].

Alta cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable [36]. Queda evidenciado este patrón

en la clase GestwebserviceController.php pues los datos manejados en esta clase son coherentes y la información que en ella se maneja es única de esa clase.

GOF: Gang of Four (“pandilla de los cuatro”).

Fachada: Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar. Sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. En el componente de seguridad en servicios web este patrón se manifiesta en la clase pgPackage.js pues es la clase donde se construye la interfaz pricipal juntando varias clases de interfaz visual [37].

2.3.4. Diagrama de clases del diseño

El diagrama de clases del diseño se realiza para que sirva de guía durante el proceso de implementación del sistema. A continuación se muestran los diagramas de clases del diseño para los requisitos funcionales Gestionar escenarios de seguridad de los servicios web que brinda un sistema y Gestionar consumo de servicio web:

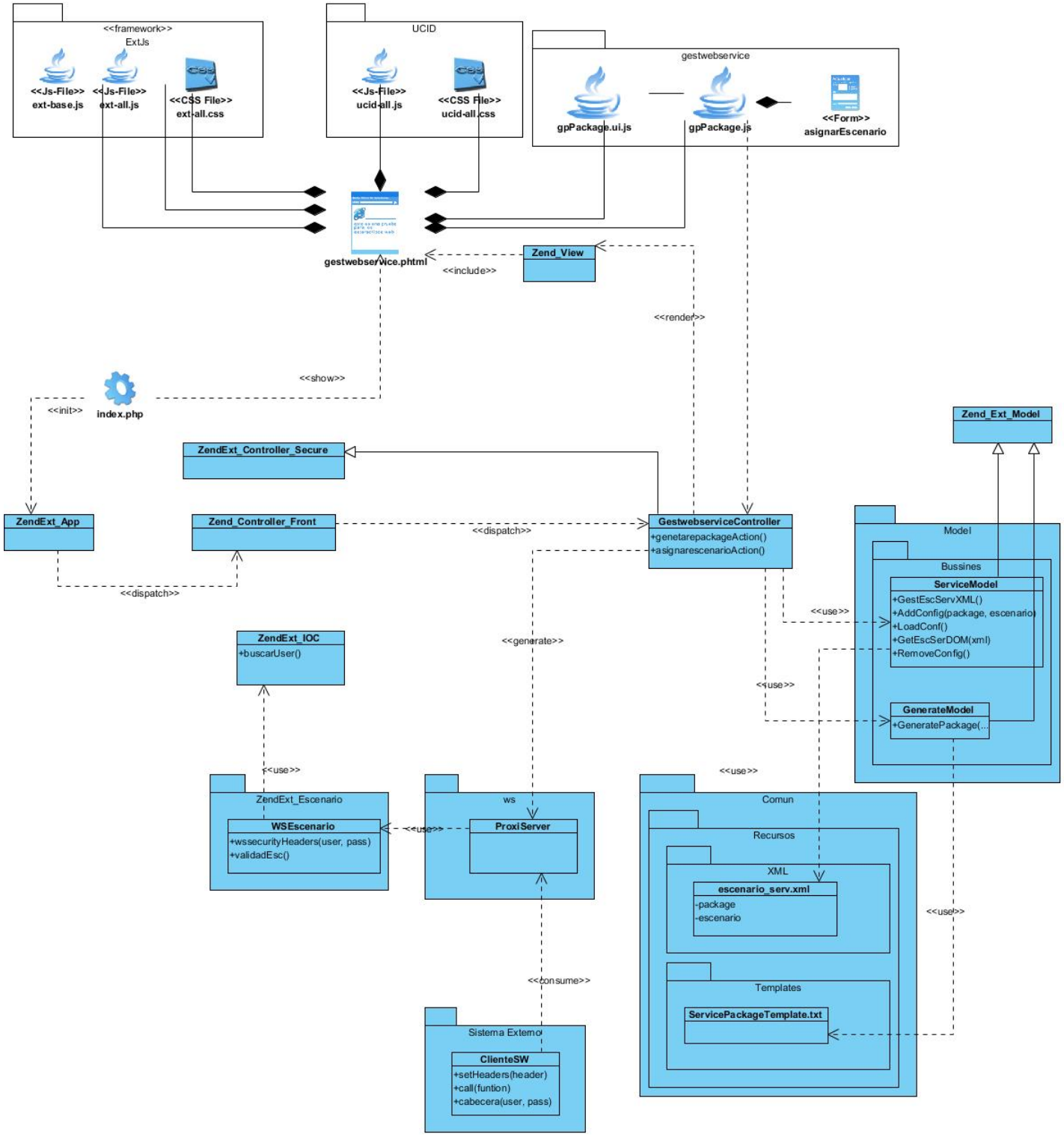


Ilustración 6: Diagrama de clases Componente para la gestión de la seguridad de los servicio web en el marco de trabajo Sauxe.

Fuente: Elaboración propia.

Descripción de las clases del diseño.

A continuación se muestra la descripción de cada clase que componen el diagrama de clases del diseño tomados del artefacto Modelo de Diseño el cual se encuentra en repositorio documental de proyecto Sistema de gestión integral de seguridad (ACAXIA).

Tabla 7: Descripción de clases del diseño

Fuente: Elaboración propia.

Clase	Descripción
gpPackage.js	Clase donde se selecciona el escenario que va ser asignado al servicio web seleccionado.
gpPackage.ui.js	Clase donde se crean el botón Asignar Escenario y se selecciona el servicio web al que se le aplicará el escenario de seguridad.
Ext-base.js	Biblioteca JavaScript utilizada para el desarrollo de las interfaces de usuario.
Ext-all.js	Biblioteca JavaScript utilizada para el desarrollo de las interfaces de usuario.
Ext-all.css	Biblioteca CSS utilizada para el desarrollo de las interfaces de usuario.
Ucid-all.js	Clases utilizadas para la configuración de los mensajes.
Ucid-all.css	Clases utilizadas para la configuración de los mensajes
<<Form>>asignarEscenario	Formulario utilizado para la selección del escenario de seguridad.
Gestservice.phtml	Es el HTML donde se incluyen todos los java script para la representación del componente.

Index.php	Es la clase servidora que se encarga de toda la configuración de la aplicación, re direccionar y dar seguridad.
Zend_View	Provee una plantilla basada en PHP, proporciona un ayudante que permite la creación de códigos de visualización.
ZendExt_Controller_Secure	Clase de donde heredan todas las clases controladoras.
Zend_Controller_Front	Inicializa el entorno de la solicitud, re direcciona la solicitud entrante, y luego envía las acciones descubiertas; le agrega las respuestas y los devuelve cuando el proceso se haya completado.
ZendExt_App	Clase que proporciona ayuda en la gestión del Modelo-Vista-Controlador.
Zend_Ext_Model	Obtiene las conexiones activas en el sistema cuando se trata de acceder a una clase determinada del modelo.
GestserviceController.php	Es la clase que gestiona la lógica de negocio y se encarga de capturar los parámetros que le son enviados desde la interfaz.
GeneratorModel.php	Es la clase encargada de generar los clientes para el consumo de los servicios web.
ServiceModel.php	Es la clase encargada de salvar los datos al fichero xml.
ServicePackegeTemplate.txt	Fichero utilizado por la clase GeneratorModel.php para generar la clase cliente.

Escenario_serv.xml	Fichero XML donde se salvan los datos de la asignación del escenario al servicio web.
ProxiServer.php	Clase donde se crea el servicio web.
ClienteWS.php	Es la clase utilizada para hacer interactuar al usuario con el servicio web.
WSEscenario.php	Es la clase que contiene el código de cada escenario de seguridad.
ZendExt_IoC	Clase que contiene el servicio web utilizado para buscar usuarios.

2.4 Diagrama de Secuencia:

En el diagrama de secuencia se indican los módulos o clases que forman parte del programa y las llamadas que se hacen en cada uno de ellos para realizar una tarea determinada. Define acciones que se pueden realizar en la aplicación en cuestión [38]. A continuación se muestra el diagrama correspondiente al diagrama de clases:

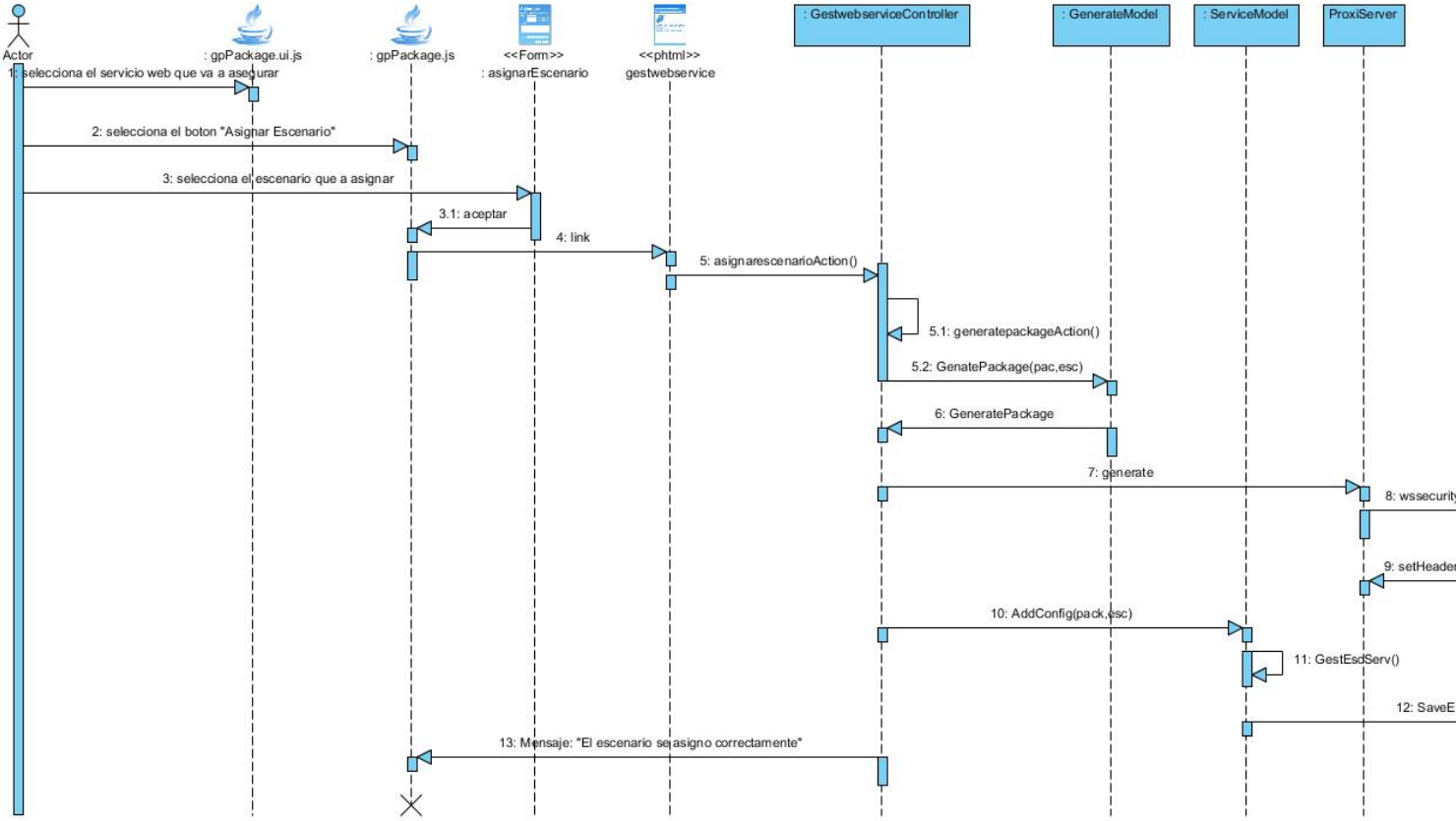


Ilustración 7: Diagrama de secuencia Gestionar escenarios de seguridad de los servicios web que brinda un sistema

Fuente: Elaboración propia.

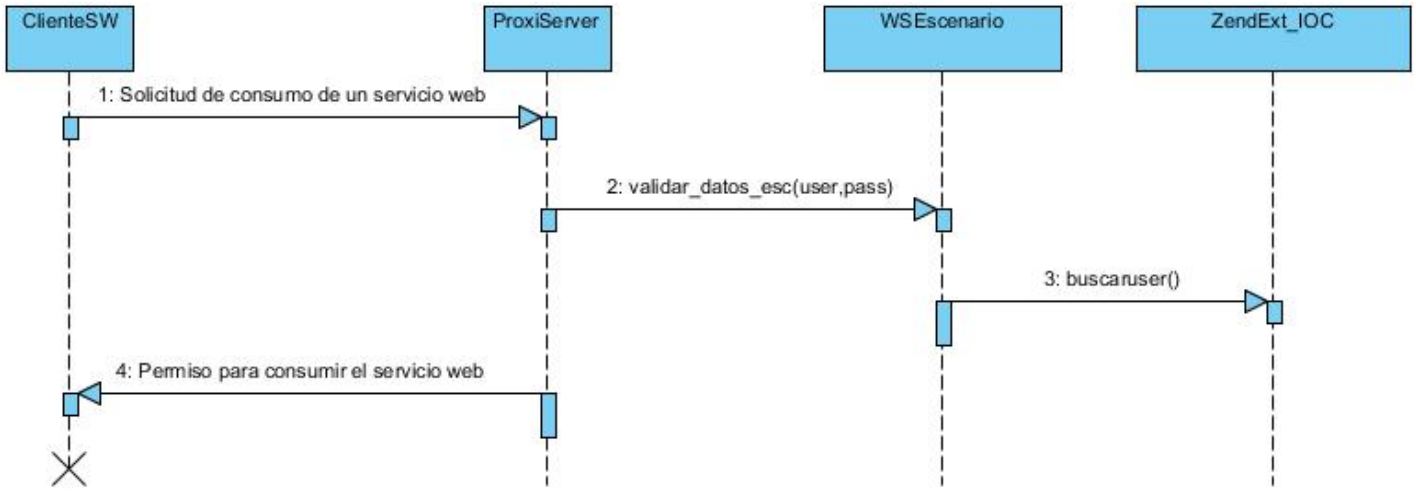


Ilustración 8: Diagrama de Secuencia Gestionar consumo de servicio web.

Fuente: Elaboración propia.

2.4 Modelo de datos

Para el desarrollo del Componente para seguridad en servicios web se definió un modelo de datos para salvar la asignación de los escenarios de seguridad a cada servicio web que consta de un fichero XML.

saveXML: Fichero XML donde se salva la configuración del servicio web respecto al escenario de seguridad asignado.

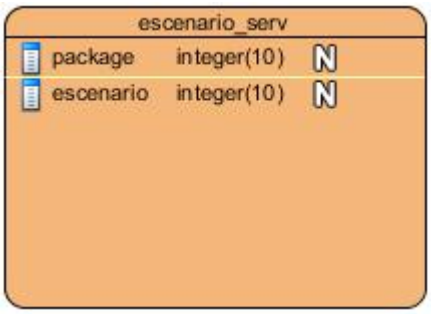


Ilustración 9: Modelo de datos.

Fuente: Elaboración propia.

Conclusiones parciales

Entre los principales resultados alcanzados en el desarrollo del capítulo están:

- Se analizaron los principales conceptos asociados a la investigación obteniéndose el modelo conceptual del sistema el cual permitió tener una mejor visión acerca de la solución que se desea alcanzar.
- La utilización de técnicas de captura, definición y validación de requisitos permitió la definición de los requisitos funcionales y no funcionales los cuales facilitarán la realización del componente.
- Se realizó el análisis y diseño de la solución permitiendo una mejor comprensión de la solución a implementar.
- Se utilizaron patrones arquitectónicos y de diseño, los cuales permitieron la elaboración de diagramas de clase del diseño y de secuencias más robustos y fáciles de comprender.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

En el presente capítulo se presentarán los diagramas de componentes y de despliegue correspondientes al Modelo de Implementación. Se describirán los estándares de codificación utilizados para garantizar el entendimiento y legibilidad del código. Además se mostrarán los resultados de la solución a través de las pruebas realizadas.

3.1- Modelo de Implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros [39].

3.1.1. Diagrama de componentes

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Permiten visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Muestran las dependencias lógicas entre componentes software, sean éstos componentes fuentes, binarios o ejecutables.

Para la realización del Componente para la gestión de los servicios web en el marco de trabajo Sauxe fue necesario la interacción entre diferentes componentes entre los que se encuentran WSEscenario, gestwebservices, proxiServer y Sistema Externo utilizando IoC para hacer uso de los servicios internos del marco de trabajo. A continuación se muestra el diagrama de componentes.

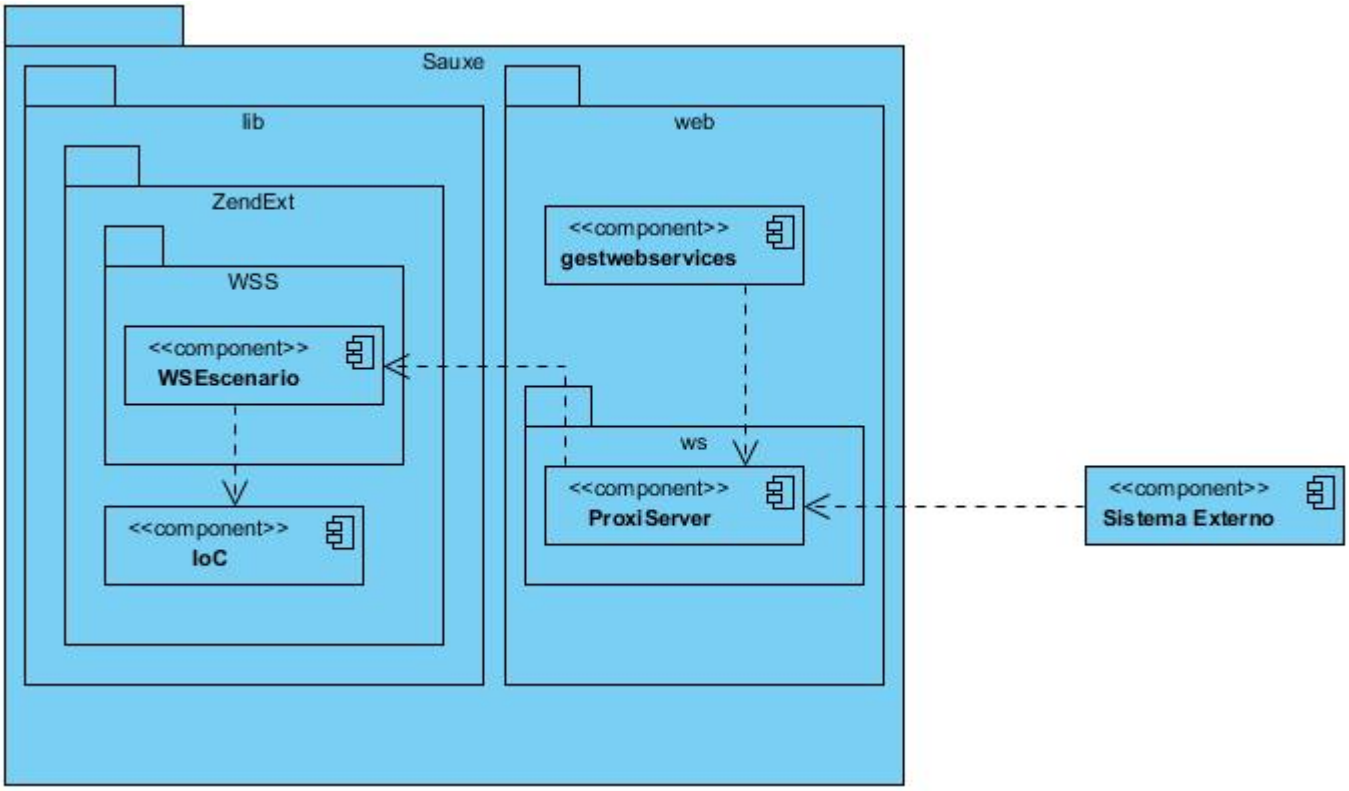


Ilustración 10: Diagrama de Componentes.

Fuente: Elaboración propia.

3.1.2. Modelo de despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema además es un Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes [40].

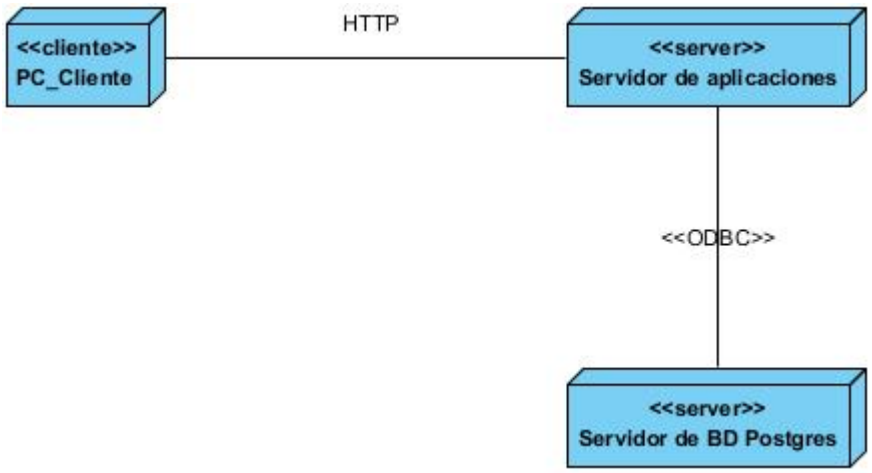


Ilustración 11: Diagrama de despliegue.

Fuente: Elaboración propia.

3.1.3. Estándares de codificación.

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código [41]. Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código pueda ser mantenido [41].

Estándares de nomenclatura.

- **Nomenclatura de clases.**

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing.

Ejemplo:

- **Nomenclatura según el tipo de clases.**
  - **Clase controladora.**

Las clases controladoras después del nombre llevan la palabra: “Controller”.

Ejemplo: GestwebservicesController.

- **Nomenclatura de las funciones.**

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing.

Ejemplo: setHeaders.

- **Nomenclatura de variables.**

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing.

Ejemplo: escSeguridad.

Normas de comentarios.

Es una necesidad comentar todo lo que se haga dentro del desarrollo, es decir, establecer las pautas que conlleven a lograr un código más legible y reutilizable, de manera que se pueda aumentar la posibilidad de mantenimiento a lo largo del tiempo. Los comentarios deben añadir claridad al código. Deben contar el por qué y no el cómo (calleja).

- **En las clases.**

Antes la declaración de cada clase se realiza una breve descripción de la misma donde se explique su propósito.

Ejemplo:

```
/**
* Nombre de la clase *
* Descripcion *
* @author *
* @package *(módulo)
* @subpackage *(sub módulo)
* @copyright *
* @version (versión - parche) */
```

- **En las funciones.**

Antes la declaración de cada función se realiza una breve descripción de la misma donde se explique su propósito.

Ejemplo:

```
/**
* Nombre de la función *
* Descripcion *
* @author * (en caso de que no sea el autor de la clase)
* @param *(los parámetros que se le pasan a la función con su descripción)
* @throws *(en caso de que dispare una excepción)
* @return *(se pone lo que devuelve la función y un comentario)
*/
```

Estilo de código.

En la implementación cuando se escriba una sentencia en PHP la forma de utilizar los tab del mismo es la siguiente:

```php
<?php
//código
?>
```

Ilustración 12: Estilo de código.

Fuente: [22]

- **Sangría o indexado.**

La política de sangría a utilizar en la implementación es por tab.

```php
<?php
/**
 * Indentation
 */
class Example {
    var $theInt = 1;
    function foo($a, $b) {
        switch ( $a) {
            case 0 :
                $Other->doFoo ();
            break;
            default :
                $Other->doBaz ();
        }
    }
    function bar($v) {
        for($i = 0; $i < 10; $i ++) {
            $v->add ( $i );
        }
    }
}
?>
```

Ilustración 13: Sangría o indexado.

Fuente: [22]

- **Brazas o llaves.**

En la declaración de clases o interfaces, métodos, bloques y switch, la apertura de llaves se hace en la misma línea.

```php
<?php
/**
 * Braces
 */
interface EmptyInterface {
}

class Example {
    function bar($p) {
        for($i = 0; $i < 10; $i ++) {
        }
        switch ( $p) {
            case 0 :
                $fField->set ( 0 );
            break;
            case 1 :
                {
                    break;
                }
            default :
                $fField->reset ();
        }
    }
}
?>
```

Ilustración 14: Brazas o llaves.

Fuente: [22]

3.2- Métrica de diseño

Permiten medir de forma cuantitativa la calidad de los atributos internos del software, lo cual permite al ingeniero evaluar la calidad durante el desarrollo del sistema. Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo [42].

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta [42].

Complejidad de implementación: Consiste en el grado de dificultad que tiene implementado un diseño de clases determinado [42].

Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software [42].

Acoplamiento: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización [42].

Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto [42].

Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado [42].

Las métricas seleccionadas para evaluar la calidad del diseño del componente para la gestión de la seguridad de los servicios web en el marco de trabajo Sauxe son:

3.2.1- Tamaño Operacional de Clases (TOC)

Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Responsabilidad: Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.

Complejidad de implementación: Un aumento del TOC implica un aumento de la complejidad de implementación de la clase**.**

Reutilización: Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Para la evaluación de dichos atributos si definieron los siguientes criterios y categorías de evaluación.

Tabla 8: Criterios de evaluación de la métrica TOC.
Fuente: Elaboración propia.

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
	Baja	<=Promedio

Complejidad de implementación	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Resultados obtenidos

Evaluando la cantidad de operaciones de cada una de las clases con que cuenta el Componente para la gestión de la seguridad de los servicios web en el marco de trabajo Sauxe según los criterios establecidos en la Tabla 9, se obtuvo que un 67% de las clases cuentan con Responsabilidad y Complejidad Baja, y un 67% de las clases tienen una Reutilización Alta.

Tabla 9: Instrumento de evaluación de la métrica TOC.

Fuente: Elaboración propia.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GestwebserviceController	2	Baja	Baja	Alta
ServiceModel	5	Alta	Alta	Baja
GenerateModel	1	Baja	Baja	Alta
WSEscenario	2	Baja	Baja	Alta
ServerSW	0	Baja	Baja	Alta
ClienteSW	3	Media	Media	Madia

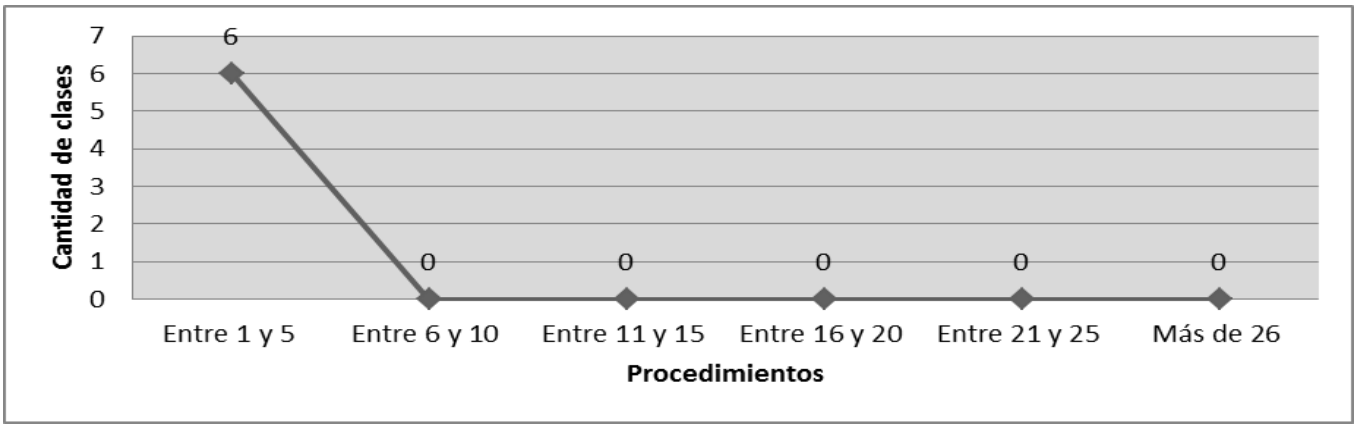


Ilustración 15: Representación de las clases según la cantidad de operaciones.

Fuente: Elaboración propia.

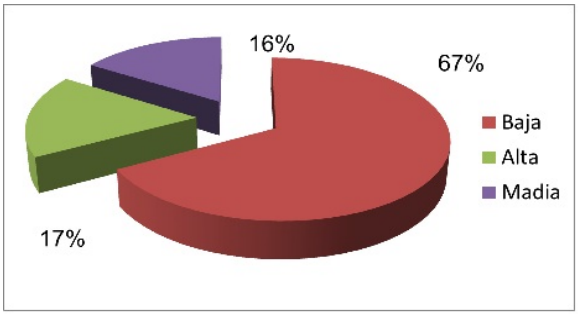


Ilustración 16: Resultados de la evaluación de la métrica TOC para el atributo de calidad Responsabilidad.

Fuente: Elaboración propia.

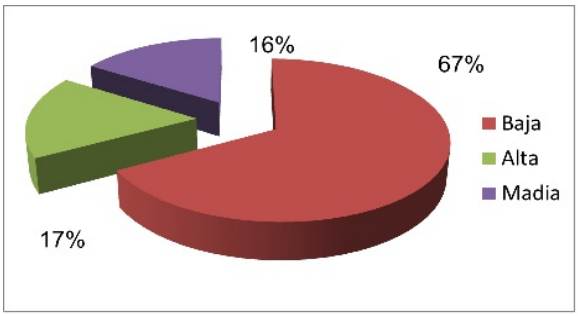


Ilustración 17: Resultados de la evaluación de la métrica TOC para el atributo de calidad Complejidad.

Fuente: Elaboración propia.

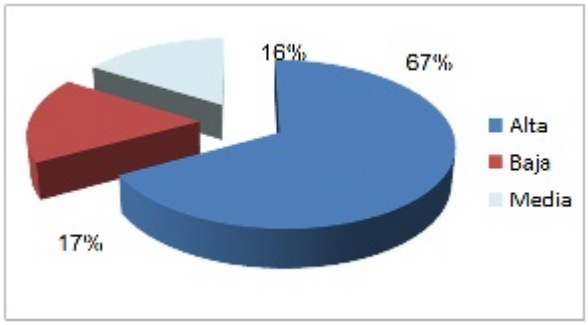


Ilustración 18: Resultados de la evaluación de la métrica TOC para el atributo de calidad Reutilización.

Fuente: Elaboración propia.

3.2.2- Relación entre Clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Acoplamiento: Un aumento del RC implica un aumento del Acoplamiento de la clase.

Complejidad de mantenimiento: Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.

Reutilización: Un aumento del RC implica una disminución en el grado de reutilización de la clase.

Cantidad de pruebas: Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para la evaluación de dichos atributos si definieron los siguientes criterios y categorías de evaluación.

Tabla 10: Criterios de evaluación de la métrica RC.

Fuente: Elaboración propia.

Atributos	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de	Baja	<=Prom
	Media	Entre Prom y 2*Prom

mantenimiento	Alta	>2*Prom
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom
	Alta	<= Prom.
Cantidad de pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Resultados obtenidos

Evaluando la cantidad de relaciones entre las clases con que cuenta el Componente para la gestión de la seguridad de los servicios web en el marco de trabajo Sauxe, mostrados en la Tabla 11, se obtuvo que el 60% de las clases tienen Bajo Acoplamiento y un 40% Acoplamiento Medio, un 83% de las clases tienen una Complejidad de mantenimiento Baja, el 83% de las clases cuentan con Alta Reutilización, así como el 83% de las clases cuentan con una Baja Cantidad de pruebas.

Tabla 11: Instrumento de evaluación de la métrica RC.

Fuente: Elaboración propia.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad de Mantenimiento	Reutilización	Cantidad de Pruebas
GestwebserviceController	5	Alta	Alta	Baja	Alta
ServiceModel	2	Medio	Baja	Alta	Baja
GenerateModel	2	Medio	Baja	Alta	Baja
WSEscenario	1	Bajo	Baja	Alta	Baja
ServerSW	1	Bajo	Baja	Alta	Baja

ClienteSW	1	Bajo	Baja	Alta	Baja

Teniendo en cuenta la cantidad de dependencias entre las clases mostradas, está establecido que una cantidad de relaciones Buena será igual o menor que 1, una cantidad Regular sería entre 1 y 3, mientras que una cantidad Mala será igual o mayor que 3. Como se puede observar en la siguiente tabla el 50% de las clases poseen cantidad de relaciones Buena.

Tabla 12: Evaluación según cantidad de relaciones.

Fuente: Elaboración propia.

Umbral	Relaciones	Cantidad de clases	Promedio
Buena	<=1	3	50
Regular	>1 <3	2	33.33333
Malo	=>3	1	16.66666

Estos resultados son positivos según los atributos de Acoplamiento, Complejidad del Mantenimiento, Reutilización y Cantidad de Pruebas.

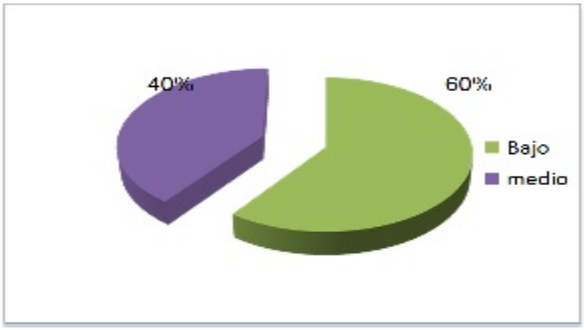


Ilustración 19: Resultados de la evaluación de la métrica RC para el atributo de calidad Acoplamiento.

Fuente: Elaboración propia.

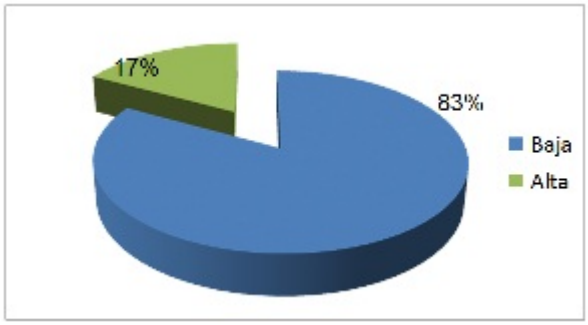


Ilustración 20: Resultados de la evaluación de la métrica RC para el atributo de calidad Complejidad de Mantenimiento.

Fuente: Elaboración propia.

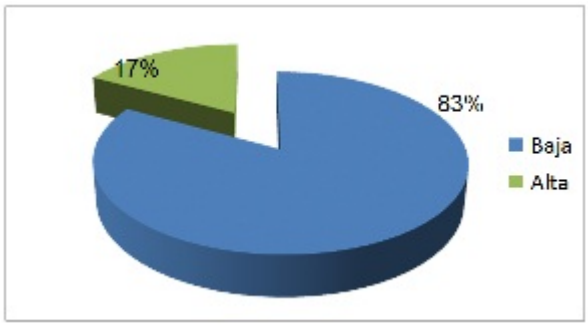


Ilustración 21: Resultados de la evaluación de la métrica RC para el atributo de calidad Cantidad de Pruebas.

Fuente: Elaboración propia.

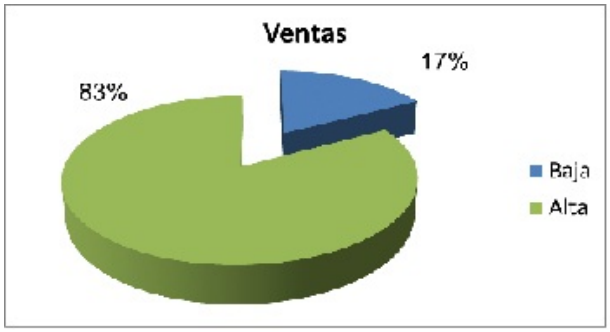


Ilustración 22: Resultados de la evaluación de la métrica RC para el atributo de calidad Reutilización.

Fuente: Elaboración propia.

3.3- Pruebas de Software

3.3.1. Pruebas de caja negra

Las pruebas de caja negra permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos [43].

Para la realización de las pruebas de caja negra del Componente para la gestión de seguridad de los servicios web en el marco de trabajo Sauxe, se elaboraron casos de pruebas específicos para los requisitos funcionales, a los cuales se les puede aplicar este tipo de prueba, descritos en la sección **2.2.1** de este documento. Dichos casos de pruebas fueron aprobados por el equipo de trabajo del Departamento de Tecnología encargado de revisar y aprobar el componente y pueden ser encontrados en el repositorio documental del proyecto Sistema de Gestión Integral de Seguridad (ACAXIA) en el documento Diseño de casos de prueba, se puede observar el acta de liberación del componente en el Anexo 6.

3.4- Prueba del Componente para la gestión de la seguridad de los servicios web en el marco de trabajo Sauxe como solución al problema científico de la investigación basándose en el método de Cuasiexperimento.

Se definió un conjunto de cuasiexperimentos para validar la efectividad de la solución propuesta.

Diseño experimental del cuasiexperimento

Para la realización del cuasiexperimento se definieron tres escenarios de pruebas y dos momentos. Estos representan posibles situaciones en las que se puede apreciar la variación de la variable seguridad en servicios web [44]. La tabla 13 muestra el diseño experimental realizado y la simbología utilizada es la siguiente.

- G: Grupo de participantes. En este caso son los servicios web que son tratados de consumir por el sistema externo asociado a tres momentos.
- X: Tratamiento o estimulo. En este caso X_1 constituye el escenario de seguridad número uno y X_2 corresponde al escenario número dos.
- O: Observaciones.

Tabla 13: Diseño experimental propuesto.

Fuente: Elaboración propia.

	Original		Aplicación del escenario #1		Aplicación del escenario #2
G_1	O_1	X_1	O_2	X_2	O_3
G_2	O_4	X_1	O_5	X_2	O_6
G_3	O_7	X_1	O_8	X_2	O_9

En el primer momento (Original) las observaciones O_1, O_4, O_7, representan la evaluación de la seguridad en los servicios web en su estado original. En el segundo momento (Aplicación del escenario #1) las observaciones O_2, O_5, O_8, representan la evaluación de la seguridad en los servicios web después de la aplicación del escenario de seguridad número uno. En el último momento (Aplicación del escenario #2) las observaciones O_3, O_6, O_9, representan la evaluación de la seguridad en los servicios web luego de aplicar el escenario de seguridad número dos.

Para la realización de las evaluaciones se utilizaron indicadores, los cuales fueron definidos por especialistas del departamento de Tecnología del Centro de Informatización de Gestión de Entidades (CIEGE) dichos indicadores se pueden apreciar en el Anexo 1. La evaluación de los indicadores puede tomar valores entre cero y cinco en función del nivel de cumplimiento que tenga el escenario en el indicador evaluado dicha evaluación se puede apreciar en el Anexo 2. Con este objetivo se aplican los indicadores definidos para cada uno de los procesos. Concluida esta actividad, se realizan los cálculos establecidos en el cuasiexperimento para obtener las evaluaciones finales de los escenarios por cada uno de los indicadores.

La segunda prueba consiste en verificar la existencia de diferencias significativas entre las evaluaciones finales obtenidas. Con este objetivo se aplica el test de Kruskal-Wallis para hacer un análisis no paramétrico de *k* muestras independientes.

La tercera prueba toma como base los resultados anteriores, de ellos se seleccionan los dos momentos con mayor rango medio para hacer comparación por pares, con este objetivo se aplica test de Mann-Whitney. Este análisis es para detectar diferencias significativas entre el escenario original existente y el escenario con mayor cumplimiento de los indicadores.

En ambos test, para el nivel de significación, se aplica el método de Monte Carlo con intervalos de confianza del 99%, considerando que existen diferencias significativas cuando la significación sea

menor a 0.05. En el procesamiento de los resultados de los experimentos se usó el programa estadístico SPSS (siglas de Statistical Package for the Social Sciences) versión 13.0. Con este diseño experimental se procede a la aplicación del pre-experimento y al análisis de los resultados, lo que se presenta a continuación.

Aplicación del cuasiexperimento y análisis de los resultados

La aplicación cuasiexperimento tiene como objetivo valorar el nivel de fortaleza que presentan los escenarios de seguridad implementados en comparación con escenario existente antes de la aplicación de componente para la gestión de la seguridad de los servicios web. Para ello fue necesario evaluar la seguridad en los servicios web en cada momento a partir del nivel de cumplimiento de los indicadores seleccionados.

Para realizar estas evaluaciones, se realizaron a través de un sistema externo y la realización de un conjunto de peticiones para consumir los servicios web utilizados como muestra para este cuasiexperimento. Dichas peticiones se describen a continuación.

Para el escenario original:

Ilustración 23: Sistema externo.

Fuente: Elaboración propia.

El sistema externo consume el servicio de Autenticar usuario publicado por el marco de trabajo Sauxe, sin que se realicen verificaciones sobre el usuario que está realizando la petición, el flujo de mensajes SOAP puede verse en el Anexo 5.

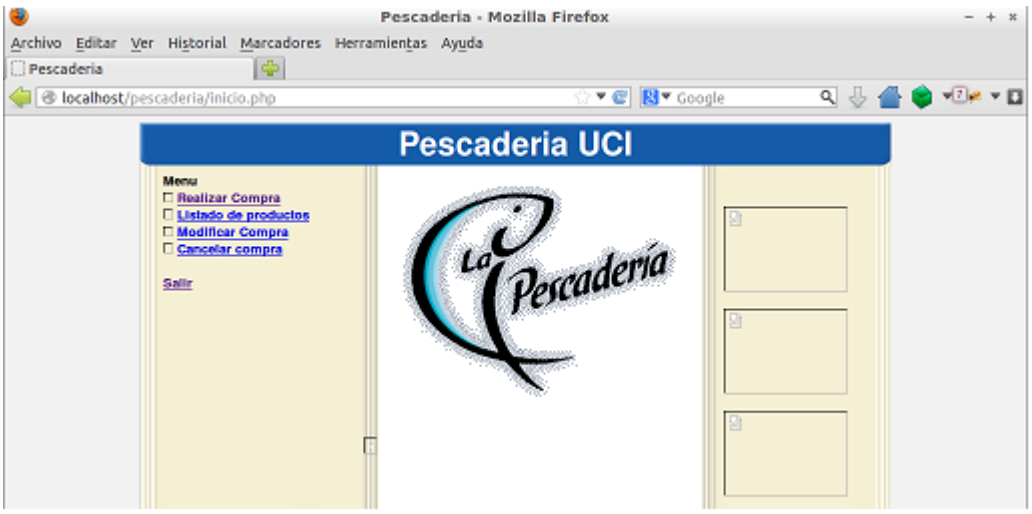


Ilustración 24: Consumo libre del servicio web por parte del sistema externo.

Fuente: Elaboración propia.

Para el escenario uno se realizó dos tipos de peticiones una cuando los parámetros enviados son inválidos y otra con parámetros válidos, los mensajes SOAP para los dos ejemplos son mostrados en el Anexo 5. Para las peticiones inválidas, el componente desarrollado realiza las verificaciones y validaciones correspondientes a este escenario denegando la petición.

Ilustración 25: Petición denegada al sistema externo una vez verificados los parámetros correspondientes al escenario número uno.

Fuente: Elaboración propia.

Para las peticiones validas se verificaron y validaron de los parámetros por el componente desarrollado, dando acceso a la información solicitada correspondiente al servicio web Autenticar usuario.

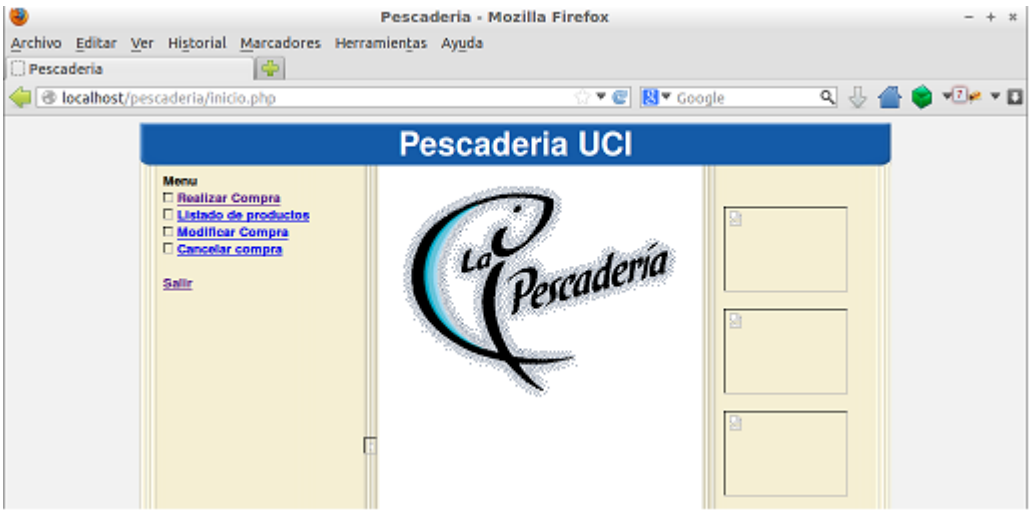


Ilustración 26: Consumo del servicio web por parte del sistema externo una vez verificados los parámetros correspondiente al escenario número uno.

Fuente: Elaboración propia.

Para el escenario dos se realizaron también dos tipos de peticiones una con parámetros válidos y otra con parámetros inválidos, los mensajes SOAP enviados se pueden apreciar en el Anexo 5. Para los parámetros inválidos el componente realiza las verificaciones y validaciones correspondientes a este escenario denegando la petición.

Ilustración 27: Petición denegada al sistema externo una vez verificados los parámetros correspondientes al escenario número dos.

Fuente: Elaboración propia.

Para los datos validos el componente realiza las verificaciones y validaciones necesarias y da acceso a la información correspondiente al servicio web Autenticar usuario.

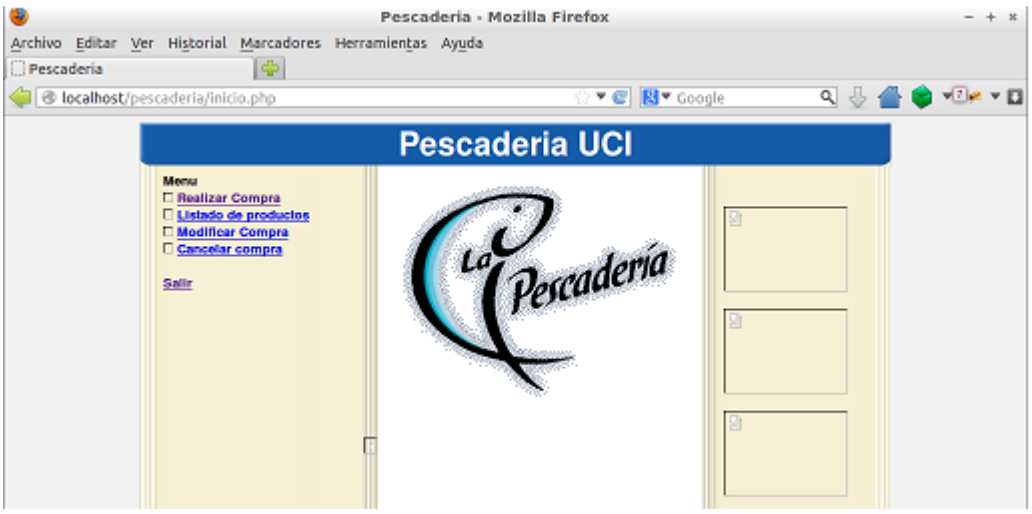


Ilustración 28: Consumo del servicio web por parte del sistema externo una vez verificados los parámetros correspondientes al escenario número dos.

Fuente: Elaboración propia.

A partir de estas pruebas se obtuvieron las evaluaciones de los indicadores asociados a los tres momentos seleccionados, arrojando los siguientes resultados:

- El escenario original presenta deficiencias en la verificación de la existencia de los parámetros definidos para el consumo de servicios web.
- El escenario original presenta deficiencias en la validación de los parámetros recibidos en el mensaje SOAP.
- El escenario original y el escenario uno presentan deficiencias en la implementación de algoritmos de codificación para el envío de los parámetros.
- El escenario original presenta limitaciones en el momento de dar respuesta de forma segura a las peticiones aceptadas.

La Figura 22 muestra los resultados de las evaluaciones realizadas en función del porciento de cumplimiento de los indicadores.

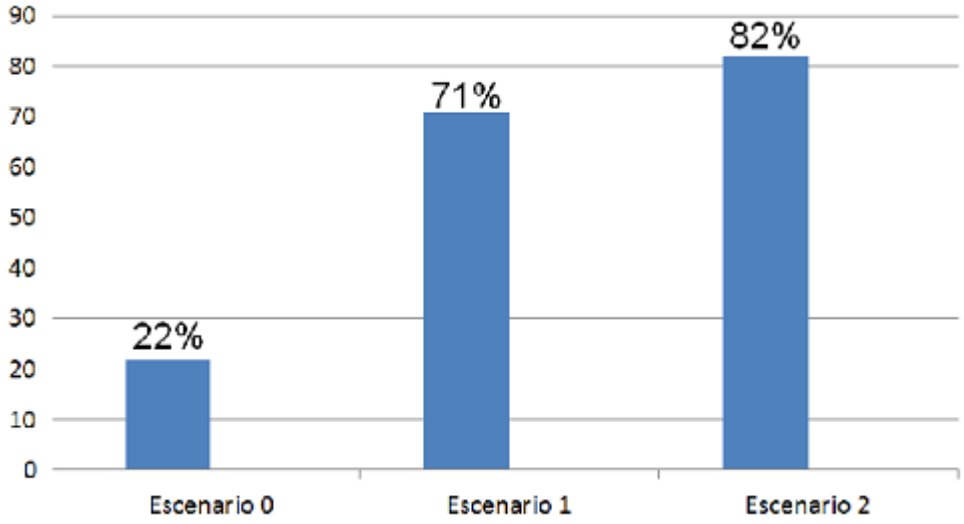


Ilustración 29: Cumplimiento de los indicadores.

Fuente: Elaboración propia.

En el análisis anterior se pudo evidenciar que los escenarios uno y dos correspondientes al componente desarrollado presentan mayores niveles de fortaleza que el escenario original y que el escenario dos presenta mayor fortaleza que el escenario uno. A pesar de esto, es necesario comprobar si existen diferencias significativas entre los valores obtenidos a partir de las evaluaciones realizadas. Para realizar esta verificación se demuestra si existen diferencias entre las evaluaciones de los sistemas en los indicadores (indicadores del momento original al momento del escenario). Con este objetivo se aplicó el test de Kruskal-Wallis. A partir de los resultados de esta prueba, evidenciados en el Anexo 3, se pudo evidenciar que existen diferencias significativas (significación menor a 0.05) entre las evaluaciones de los sistemas.

Partiendo de los resultados anteriores, para realizar las comparaciones por pares se seleccionó el escenario original y el escenarios dos. Para dicha comparación se aplicó el test de Mann-Whitney, se observa que la comparación entre el escenario dos y el escenario original son significativas (significación menor a 0.05) entre las evaluaciones de los indicadores asociados a cada uno de ellos, observándose los resultados en el Anexo 4.

Las pruebas realizadas permiten afirmar que en los escenarios uno y dos la variable seguridad en los servicios web aumenta considerablemente con respecto al escenario original, demostrando que el componente para la gestión de la seguridad de los servicios web en el marco de trabajo Sauxe fortalece en mayor medida la confidencialidad, integridad y disponibilidad de los datos intercambiados a través de los servicios web.

Conclusiones parciales

Entre los resultados alcanzados en este capítulo están:

- Mediante la realización del modelo de implementación fue posible profundizar en el desarrollo de la solución obteniendo los diagramas de Componentes y de Despliegue, además de quedar determinados los estándares de codificación empleados en el mismo.
- Fueron realizadas las validaciones correspondientes al diseño mediante el empleo de las métricas TOC y RC, las cuales permitieron señalar la veracidad del mismo, arrojando como resultados que el diseño realizado presenta robustez y cumple con los atributos planteados por las métricas.
- Fueron realizadas dos iteraciones de pruebas de caja negra por el grupo de trabajo del departamento de Tecnología obteniéndose cero no conformidades en la última iteración demostrando que la solución implementada responde a los requisitos planteados.
- La aplicación del Cuasiexperimento permitió constatar que los escenarios uno y dos presentan mayor fortaleza que el escenario original, esto demuestra que el componente desarrollado aumenta la seguridad en los servicios web.

CONCLUSIONES

- Se realizó un estudio tanto de las especificaciones utilizadas para brindar seguridad a los servicios web centrando la atención en el estándar WS-Security, como de las soluciones que implementan estas especificaciones mostrando que a pesar de brindar seguridad a los servicios web ninguna presenta compatibilidad con el marco de trabajo Sauxe.
- El modelo conceptual propuesto por el autor integra el estándar WS-Security para incorporar elementos de seguridad a los servicios web logrando preservar la confidencialidad, integridad y disponibilidad de los datos.
- La implementación de los dos escenarios de seguridad UsernameToken y BinarySecurityToken permitieron aumentar la seguridad en las peticiones y respuestas de servicios web en el marco de trabajo Sauxe.
- La solución modelada fue implementada en el componente para la gestión de la seguridad en los servicios web el cual permite fortalecer la confidencialidad, integridad y disponibilidad de la información que se intercambia por esta vía y dando solución al problema planteado.
- Como parte de la validación de la solución se realizaron pruebas incluyendo las de caja negra y la aplicación de un cuasiexperimento donde se pudo constatar que la solución implementada cumple con los requisitos funcionales planteados y que el componente desarrollado permite fortalecer la seguridad en los servicios web en el marco de trabajo Sauxe.

RECOMENDACIONES

- Implementar otros escenarios de seguridad con el objetivo de aumentar la protección de los servicios web.
- Actualizar los servicios web del marco de trabajo Sauxe para facilitar la aplicación de esta solución.

REFERENCIAS BIBLIOGRÁFICAS

1. Rodríguez, T.M.d.l.T., *Las tecnologias de la informatica y las comunicaciones en la educacion y la salud. Su repercusion en nuestro desempeño.* 2007: p. 2.
2. Pechuán, I.G., *Sistemas y Tecnologías de la Información para La Gestión.* 2006: p. 37.
3. Baryolo, M.O.G., *MODELO DE CONTROL DE ACCESO PARA SISTEMAS DE INFORMACIÓN EN ENTORNOS MULTIDOMINIOS*. 2012, Universidad de las Ciencias Informáticas: Habana, Cuba.
4. Rojo, J.O., *Introducción a los sistemas distribuidos.* 2003.
5. *Sistemas distribuidos panorama.* p. 17.
6. Hurtado, S., *Seguridad en comunicaciones de servicios Web.* 2008.
7. C. Gutiérrez, E.F.-M., M. Piattini, *Seguridad en Servicios Web.* 2005.
8. *metodos de investigacion.*
9. Echarri, P.A.M.V.R., *La metodología de la Investigación Educativa para la Formación del Profesional.* 2002.
10. Alberto Lafuente, M.L., *Sistemas Distribuidos, Introducción.*
11. *Servicios WEB.* 2008.
12. Fabricio Alvarez, L.G., Guzmán Llambías, *Especificaciones WS-*.* 2007.
13. *Servicio Web.* 2008.
14. Gutiérrez., J.J., *¿Qué es un framework web?* : p. 4.
15. *SSL Secure Sockets Layer*. Available from: http://www.digicert.com/ssl.htm.
16. Martz, C. *X.509 - Digital Certificate Standard*. Available from: http://www.birds-eye.net/definition/x/x.509-digital_certification_standard.shtml.
17. Isaza E., G.A., *Estándares de seguridad basados en XML para servicios web y web semántica.*
18. *Axis y Axis2, contenedores de Web Services*. Available from: http://www.consultoriajava.com/tools/axis.shtml.
19. *WSO2 Web Services Framework for C++*. Available from: http://wso2.com/products/web-services-framework/cpp/.
20. *WSO2 Web Services Framework for PHP*. Available from: http://wso2.com/products/web-services-framework/php/.
21. ENTIDADES, C.D.I.D.L.G.D., *MODELO DE DESARROLLO DE SOFTWARE.*
22. Thais Figueras Garcia, M.L.L.I., *Solución informática para la Compartimentación de la información en los sistemas que utilizan el Sistema de Gestión Integral de Seguridad ACAXIA*. 2012, Universidad de las Ciencias Informaticas: Habana, Cuba.
23. *Guía Zend: Introducción y primera aplicación*. Available from: http://www.maestrosdelweb.com/editorial/guia-zend/.
24. *PHP*. Available from: http://www.php.net.
25. *Apache*. Available from: http://httpd.apache.org/.
26. *Sobre PostgreSQL*. 2010; Available from: http://www.postgresql.org.es/sobre_postgresql.
27. *Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) 6.0*. Available from: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.
28. *NetBeans IDE*. Available from: http://netbeans-ide.softonic.com/.
29. C., B.G. *Descubriremos todo lo concerniente al protocolo SOAP*. 2004; Available from: http://www.desarrolloweb.com/articulos/1557.php.
30. Laguna, M.A., *Ingeniería del Software I 3º I.T.I.Gestión.* p. 16.
31. Escalona, M.J., *Ingeniería de Requisitos en Aplicaciones para la Web – Un estudio comparativo.* 2002.

32. Amador Durán Toro, B.B.J., *Metodología para la Elicitación de Requisitos de Sistemas Software.* 2000.
33. Ruiz, F., *INGENIERÍA DEL SOFTWARE I tema 4 Diseño de Software.* p. 44.
34. Peña, P.O., *DISEÑO ARQUITECTÓNICO PARA LA BÚSQUEDA SEMÁNTICA EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS*. 2011, Universidad de las Ciencias Informáticas: La Habana.
35. Quintero, J.B., *Arquitectura de Software Patrones en la arquitectura.*
36. Mora, R.C., *Patrones de GRASP.*
37. Mühlrad, D., *Patrones de diseño.* 2008.
38. Tello, J.C., *Diagrama de secuencia.*
39. Acuña, K.B., *SELECCIÓN DE METODOLOGÍAS DE DESARROLLO PARA APLICACIONES WEB EN LA FACULTAD DE INFORMÁTICA DE LA UNIVERSIDAD DE CIENFUEGOS.*
40. Zurita, A.M., *diagrama de despliegue.* 2013.
41. Calleja, M.A., *Estándares de codificación.*
42. IEEE, *Standard Glossary of Software Engineering Terms*.
43. S. Pressman, R., *Ingeniería de Software un enfoque práctico*. 2001, Madrid.
44. Herrera, M.R.Y., *Modelo para la estimación de información ausente en las trazas usadas en la minería de proceso.* 2012, Universidad de la Ciencias Informáticas.

ANEXOS

Anexo 1. Indicadores definidos para realizar las evaluaciones a cada momento.

1. Aceptación de las peticiones de consumo de los servicio web publicados por el marco de trabajo Sauxe. 1__,2__,3__,4__,5__
2. Verificación de que la petición SOAP realizada por el sistema externo no cuenta con los parámetros definidos para el consumo de los servicios web. 1__,2__,3__,4__,5__
3. Verificación de que la petición SOAP realizada por el sistema externo cuente con los parámetros usuario y contraseña en texto plano. 1__,2__,3__,4__,5__
4. Verificación de que la petición SOAP realizada por el sistema externo cuente con los parámetros usuario y contraseña en codificada. 1__,2__,3__,4__,5__
5. Verificación cuando la petición SOAP realizada por el sistema externo cuente con los parámetros usuario y contraseña en texto plano y el marco de trabajo Sauxe tiene asignado a ese servicio web los parámetros codificados. 1__,2__,3__,4__,5__
6. Verificación cuando la petición SOAP realizada por el sistema externo cuente con los parámetros usuario y contraseña codificada y el marco de trabajo Sauxe tiene asignado a ese servicio web los parámetros en texto plano. 1__,2__,3__,4__,5__
7. Validaciones hechas por el marco de trabajo Sauxe con el objetivo de verificar que el usuario tenga permisos para consumir el servicio web. 1__,2__,3__,4__,5__
8. Respuesta del sistema a una petición de consumo de un servicio web que cumple con las validaciones realizadas utilizando parámetros en texto plano. 1__,2__,3__,4__,5__
9. Respuesta del sistema a una petición de consumo de un servicio web que cumple con las validaciones realizadas utilizando parámetros en codificados. 1__,2__,3__,4__,5__
10. Respuesta del sistema a una petición de consumo de un servicio web que no cumple con las validaciones realizadas. 1__,2__,3__,4__,5__
11. Respuesta del sistema a una petición de consumo de un servicio web. 1__,2__,3__,4__,5__

Anexo 2. Resultados de las evaluaciones de cada indicador.

Indicadores	Escenario Original	Escenario 1	Escenario 2
1	5	5	5
2	0	3	5
3	0	5	0

4	0	0	5
5	0	5	5
6	0	5	5
7	2	3	5
8	0	5	0
9	0	0	5
10	0	3	5
11	5	5	5

Anexo 3. Resultados de la aplicación del test de Kruskal-Wallis

Ranks

	Indicadores	N	Mean Rank
Escenarios	1.00	11	10.45
	2.00	11	18.91
	3.00	11	21.64
	Total	33	

Test Statistics[b,c]

			Escenarios
Chi-Square			9.811
df			2
Asymp. Sig.			.007
Monte Carlo Sig.	Sig.		.005[a]
	99% Confidence Interval	Lower Bound	.003
		Upper Bound	.007

a. Based on 10000 sampled tables with starting seed 1314643744.

b. Kruskal Wallis Test

c. Grouping Variable: Indicadores

Anexo 4. Resultados de la aplicación del test de Mann-Whitney

Ranks

	Indicadores	N	Mean Rank	Sum of Ranks
Escenarios	1.00	11	8.09	89.00
	3.00	11	14.91	164.00
	Total	22		

Test Statistics[c]

			Escenarios
Mann-Whitney U			23.000
Wilcoxon W			89.000
Z			-2.784
Asymp. Sig. (2-tailed)			.005
Exact Sig. [2*(1-tailed Sig.)]			.013[a]
Monte Carlo Sig. (2-tailed)	Sig.		.010[b]
	99% Confidence Interval	Lower Bound	.008
		Upper Bound	.013
Monte Carlo Sig. (1-tailed)	Sig.		.005[b]
	99% Confidence Interval	Lower Bound	.003
		Upper Bound	.007

a. Not corrected for ties.

b. Based on 10000 sampled tables with starting seed 624387341.

c. Grouping Variable: Indicadores

Anexo 5. Mensajes SOAP.

Petición SOAP correspondiente al escenario original para el servicio web Obtener rol.

```
−<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  −<SOAP-ENV:Body>
      <ns1:obtenerRol/>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Respuesta SOAP correspondiente al escenario original para el servicio web Obtener rol.

```
−<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  −<SOAP-ENV:Body>
    −<ns1:obtenerRolResponse>
      −<return SOAP-ENC:arrayType="ns2:Map[6]" xsi:type="SOAP-ENC:Array">
        +<item xsi:type="ns2:Map"></item>
        +<item xsi:type="ns2:Map"></item>
        +<item xsi:type="ns2:Map"></item>
        +<item xsi:type="ns2:Map"></item>
        +<item xsi:type="ns2:Map"></item>
        +<item xsi:type="ns2:Map"></item>
        </return>
      </ns1:obtenerRolResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Petición SOAP correspondiente al escenario uno para el servicio web Obtener usuario.

```
−<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  −<SOAP-ENV:Header>
    −<ns2:Security>
      −<ns2:UsernameToken>
          <ns2:Username>instalacion</ns2:Username>
          <ns2:Password>instalacion</ns2:Password>
        </ns2:UsernameToken>
      </ns2:Security>
    </SOAP-ENV:Header>
  −<SOAP-ENV:Body>
    −<ns1:obtenerUsers>
      −<security xsi:type="SOAP-ENC:Struct">
        −<namespace xsi:type="xsd:string">
            http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
          </namespace>
          <name xsi:type="xsd:string">Security</name>
        −<data xsi:type="SOAP-ENC:Struct">
          −<ns2:UsernameToken xsi:type="SOAP-ENC:Struct">
              <ns2:Username xsi:type="xsd:string">instalacion</ns2:Username>
              <ns2:Password xsi:type="xsd:string">instalacion</ns2:Password>
            </ns2:UsernameToken>
          </data>
          <mustUnderstand xsi:type="xsd:boolean">false</mustUnderstand>
        </security>
      </ns1:obtenerUsers>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Respuesta SOAP correspondiente al escenario uno para el servicio web Obtener usuario.

```
−<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  −<SOAP-ENV:Body>
    −<ns1:obtenerUsersResponse>
      −<return SOAP-ENC:arrayType="xsd:ur-type[2]" xsi:type="SOAP-ENC:Array">
        −<item SOAP-ENC:arrayType="ns2:Map[7]" xsi:type="SOAP-ENC:Array">
          +<item xsi:type="ns2:Map"></item>
          +<item xsi:type="ns2:Map"></item>
          +<item xsi:type="ns2:Map"></item>
          +<item xsi:type="ns2:Map"></item>
          +<item xsi:type="ns2:Map"></item>
          +<item xsi:type="ns2:Map"></item>
          +<item xsi:type="ns2:Map"></item>
          </item>
        −<item xsi:type="SOAP-ENC:Struct">
          −<namespace xsi:type="xsd:string">
              http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
            </namespace>
            <name xsi:type="xsd:string">Security</name>
          −<data xsi:type="SOAP-ENC:Struct">
            −<ns3:UsernameToken xsi:type="SOAP-ENC:Struct">
                <ns3:Username xsi:type="xsd:string">instalacion</ns3:Username>
                <ns3:Password xsi:type="xsd:string">cb58ceb169fa69a98b7d60a820b0b37c</ns3:Password>
              </ns3:UsernameToken>
            </data>
            <mustUnderstand xsi:type="xsd:boolean">false</mustUnderstand>
          </item>
        </return>
      </ns1:obtenerUsersResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

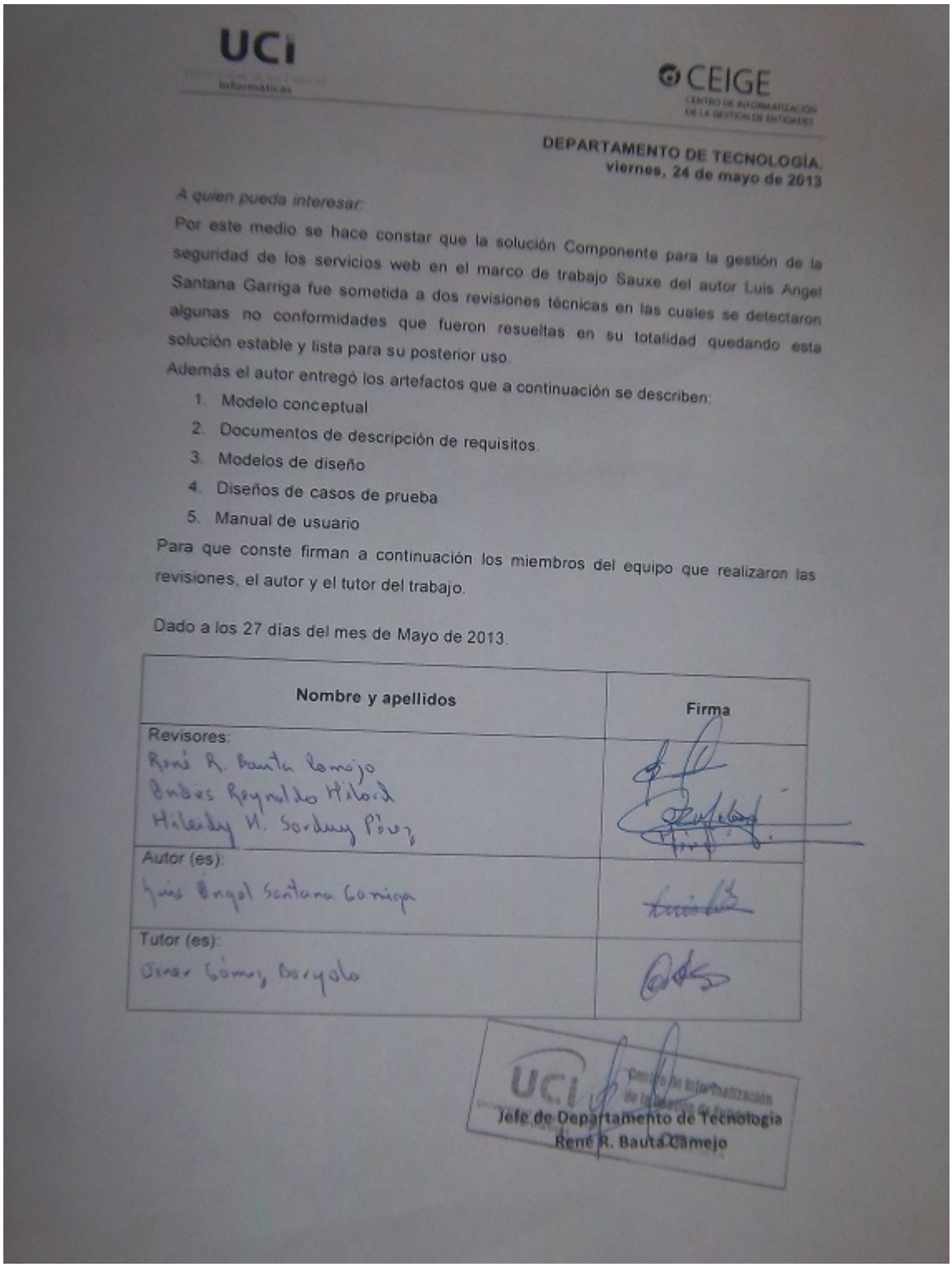

Petición SOAP correspondiente al escenario dos para el servicio web Autenticar usuario.

```
−<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  −<SOAP-ENV:Header>
    −<ns2:Security>
      −<ns2:UsernameToken>
          <ns2:Username>aW5zdGFsYWNpb24=</ns2:Username>
          <ns2:Password>aW5zdGFsYWNpb24=</ns2:Password>
        </ns2:UsernameToken>
      −<ns3:Timestamp>
          <ns3:Created>2013-05-26T17:23:03Z</ns3:Created>
        </ns3:Timestamp>
      </ns2:Security>
    </SOAP-ENV:Header>
  −<SOAP-ENV:Body>
    −<ns1:AutenticarUsuario>
        <user xsi:type="xsd:string">instalacion</user>
        <password xsi:type="xsd:string">instalacion</password>
      −<sec xsi:type="SOAP-ENC:Struct">
        −<namespace xsi:type="xsd:string">
            http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
          </namespace>
          <name xsi:type="xsd:string">Security</name>
        −<data xsi:type="SOAP-ENC:Struct">
          −<ns2:UsernameToken xsi:type="SOAP-ENC:Struct">
              <ns2:Username xsi:type="xsd:string">aW5zdGFsYWNpb24=</ns2:Username>
              <ns2:Password xsi:type="xsd:string">aW5zdGFsYWNpb24=</ns2:Password>
            </ns2:UsernameToken>
          −<ns3:Timestamp xsi:type="SOAP-ENC:Struct">
              <ns3:Created xsi:type="xsd:string">2013-05-26T17:23:03Z</ns3:Created>
            </ns3:Timestamp>
          </data>
          <mustUnderstand xsi:type="xsd:boolean">false</mustUnderstand>
        </sec>
      </ns1:AutenticarUsuario>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Respuesta SOAP correspondiente al escenario dos para el servicio web Autenticar usuario.

```
−<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  −<SOAP-ENV:Body>
    −<ns1:AutenticarUsuarioResponse>
      −<return SOAP-ENC:arrayType="xsd:ur-type[4]" xsi:type="SOAP-ENC:Array">
          <item xsi:type="xsd:string">MTEwMDAwMDAwMDE=</item>
        −<item xsi:type="SOAP-ENC:Struct">
          −<namespace xsi:type="xsd:string">
              http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
            </namespace>
            <name xsi:type="xsd:string">Security</name>
          −<data xsi:type="SOAP-ENC:Struct">
            −<ns2:UsernameToken xsi:type="SOAP-ENC:Struct">
                <ns2:Username xsi:type="xsd:string">aW5zdGFsYWNpb24=</ns2:Username>
                <ns2:Password xsi:type="xsd:string">Y2I1OGNlYjE2OWZhNjlhOThiN2Q2MGE4MjBiMGIzN2M=
                </ns2:Password>
              </ns2:UsernameToken>
            −<ns3:Timestamp xsi:type="SOAP-ENC:Struct">
                <ns3:Created xsi:type="xsd:string">2013-05-26T17:20:50Z</ns3:Created>
              </ns3:Timestamp>
            </data>
            <mustUnderstand xsi:type="xsd:boolean">false</mustUnderstand>
          </item>
          <item xsi:type="xsd:string">trace</item>
          <item xsi:nil="true"/>
        </return>
      </ns1:AutenticarUsuarioResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Anexo 6. Acta de liberación del componente realizada por el grupo de trabajo del departamento de Tecnología encargado de la revisión del mismo.



UCI
Facultad 3
Informáticas

CEIGE
CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES

DEPARTAMENTO DE TECNOLOGÍA.
viernes, 24 de mayo de 2013

A quien pueda interesar:

Por este medio se hace constar que la solución Componente para la gestión de la seguridad de los servicios web en el marco de trabajo Sauxe del autor Luis Angel Santana Garriga fue sometida a dos revisiones técnicas en las cuales se detectaron algunas no conformidades que fueron resueltas en su totalidad quedando esta solución estable y lista para su posterior uso.

Además el autor entregó los artefactos que a continuación se describen:

1. Modelo conceptual
2. Documentos de descripción de requisitos.
3. Modelos de diseño
4. Diseños de casos de prueba
5. Manual de usuario

Para que conste firman a continuación los miembros del equipo que realizaron las revisiones, el autor y el tutor del trabajo.

Dado a los 27 días del mes de Mayo de 2013.

Nombre y apellidos	Firma
Revisores: René R. Bauta Camejo Andres Reynaldo Hilera Hilerday H. Sardany Pérez	
Autor (es): Luis Angel Santana Garriga	
Tutor (es): Omar Gómez Berzobe	

UCI
Centro de Informatización de la Gestión de Entidades
Jefe de Departamento de Tecnología
René R. Bauta Camejo