UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
DIRECCIÓN DE FORMACIÓN POSTGRADUADA

PROPUESTA DE PROCESOS DE AUTENTICACIÓN, AUTORIZACIÓN Y AUDITORÍA PARA APLICACIONES BASADAS EN SERVICIOS WEB XML

Tesis presentada en opción al título de
Máster en Informática Aplicada

Autor:
Ing. Karel Gómez Velázquez

Tutores:
MSc. Antonio Rey Roque
MSc. Héctor Raúl González Diez

Ciudad de La Habana, noviembre de 2010

DECLARACIÓN JURADA DE AUTORÍA Y AGRADECIMIENTOS

Yo, *Karel Gómez Velázquez* con carné de identidad *83081008985*, declaro que soy el autor principal del resultado que expongo en la presente tesis titulada *Propuesta de procesos de Autenticación, Autorización y Auditoría para aplicaciones basadas en servicios Web XML*, para optar por el título académico de Máster en Informática Aplicada.

Este trabajo fue desarrollado durante el periodo comprendido entre 2008 y 2009, en colaboración con los colegas de equipo *Ing. Annia Arencibia Morales, Ing. Leonardo González González, Ing. Danisbel Rojas Ríos e Ing. Héctor Manuel Solís Mulet*, quienes reconocen a nombre de este autor la responsabilidad principal del resultado expuesto.

A ellos, el agradecimiento más profundo. En especial, agradecer al *MSc. Antonio Rey Roque* y al *MSc. Héctor Raúl González Diez*, quienes fungieron como tutores y a los profesores *MSc. María Francisca Velázquez Rodríguez, MSc. Ricardo Caridad Gómez León, MSc. Mirna Cabrera Hernández, MSc. Idelsis Martínez Ungo, MSc. César Nicolás Richard Martínez, MSc. Iván Juan Valido González y MSc. María Caridad Valdés Rodríguez* quienes contribuyeron a mi crecimiento profesional y humano en general. A todos ellos, así como a otros colegas y amigos que no he mencionado por razones de espacio, les doy las más sinceras gracias.

Finalmente, declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de La Habana, a los **24** días del mes de **noviembre** del año **2010**.

_____

Ing. Karel Gómez Velázquez

Autor

SÍNTESIS

La investigación está encaminada a obtener una implementación de requerimientos de seguridad para la homogenización de los procesos de Autenticación, Autorización y Auditoría (AAA) en aplicaciones basadas en servicios Web XML. Esto permite una gestión eficiente de usuarios, asignación de roles y privilegios de acceso para todos los sistemas que consuman los servicios implementados, incluye además un proceso de trazabilidad y auditoría, para llevar a cabo un control estricto de las operaciones en que se involucran los usuarios. De esta forma, se obtiene una gestión eficiente de todos los requerimientos de seguridad para aquellos sistemas externos que consuman sus servicios, lográndose de esta forma, la reutilización del código y se evita que se realicen acciones innecesarias fuera de cada negocio en particular; los servicios implementados se encuentran debidamente descritos utilizando el Lenguaje de Descripción de Servicios Web (WSDL).

Su desarrollo está basado en tecnologías libres, multiplataformas y sobre una arquitectura en capas, utilizando PHP 5 como lenguaje de programación e implementando el patrón de arquitectura Modelo Vista Controlador, PostgreSQL 8.2 como sistema de gestión de bases de datos, metodología AJAX, para realizar eficientemente las peticiones al servidor y la librería YUI para obtener una interfaz visual moderna. Utiliza estándares abiertos como XML, que permite la interoperabilidad entre aplicaciones desarrolladas sobre diferentes plataformas y además implementa los patrones “proxy” y “observer”, para el control de acceso centralizado a las peticiones realizadas entre los servicios y para el mantenimiento de la integridad referencial sobre el flujo de información distribuida respectivamente.

Palabras Claves: *Seguridad, Servicios, Autenticación, Autorización, Auditoría.*

ÍNDICE

INTRODUCCIÓN

La estrategia general para la Informatización del Sistema Nacional de Salud (SNS) cubano, propone el desarrollo y despliegue de una solución informática integral que proporcione un mayor grado de acceso a información unificada y confiable en tiempo real, de modo que aporte la rapidez y fiabilidad necesaria para las modernas técnicas de administración, la toma de decisiones en los diferentes niveles, así como que esta información llegue con la precisión y rapidez necesaria a todos los niveles requeridos. *[Delgado, 2009]*

El Ministerio de Salud Pública (MINSAP) y el Ministerio de Informática y las Comunicaciones (MIC), proyectan desde el año 2003, la ejecución de un grupo de acciones para garantizar definitivamente la informatización del SNS, como un paso más en el desarrollo estratégico en cuanto a las alianzas externas y la máxima expresión de los cambios que se introducen en el sector. De este modo, se ha logrado una integración política, estratégica y de acción que asegura una adecuada introducción de tecnologías de la información y las comunicaciones.

En sentido general se puede decir que la estrategia planteada, no siempre ha contado con una solución armónica de problemas claves identificados por el sector; la infraestructura tecnológica necesaria para el desarrollo y la generalización de las soluciones, así como, los servicios de mantenimiento, actualización y de sostenibilidad, no han estado garantizados. Se ha transitado por varias etapas en consonancia con los avances de las Tecnologías de la Información en el mundo, incorporando estas en el desarrollo, desde las aplicaciones locales en una estación de trabajo, pasando por la arquitectura cliente servidor y el trabajo en redes, hasta el empleo de sistemas distribuidos basados en tecnologías Web.

En todos los casos el objetivo de dicha estrategia ha sido**,** el de proveer al SNS de información confiable, consistente y oportuna, para la toma de decisiones y el mejoramiento de los procesos médico asistenciales, garantizando de esta manera el incremento en la calidad y seguridad de la atención médica de la población. A partir de estos aspectos, de la importancia y de la atención diferenciada que recibe el desarrollo de la informatización y automatización de los servicios de salud, el Ministerio de Salud Pública (MINSAP) ha definido como una de sus prioridades de trabajo, la informatización de todos sus procesos asistenciales o no asistenciales, es por ello que dicho organismo ha convocando a un grupo de instituciones propias y del MIC para, de una manera conjunta, definir los proyectos a desarrollar, tomando en algunos casos como punto de partida, sistemas ya elaborados en el país.

Para la concreción del objetivo planteado, es necesario crear una infraestructura o plataforma informática en la que se integren todos los productos o servicios durante la ejecución de la estrategia de informatización. Los proyectos deben acometerse por etapas y para su desarrollo se tendrá en cuenta como primer punto, la integración de todas las aplicaciones, aspecto que garantizará la consistencia, no duplicidad, oportunidad y precisión de la información. Por otro lado, un requerimiento deseado es que no se convierta cada sistema creado, en una isla de información con utilidad y beneficios muy limitados ya que por el contrario, al contar con la integración de los datos generados en los distintos niveles de salud donde puede ser atendido un paciente, se optimiza la calidad asistencial ofrecida, facilitando las funciones del personal de la salud y colaborando con la gestión administrativa,

asistencial, docente y de investigación. Es esta integración, la que permite hablar de informatización en el sector de la salud pública, no de proyectos aislados. *[Delgado, 2009]*

En la nueva etapa, iniciada en el año 2003, se ha definido por el MINSAP, un grupo de premisas y requisitos incorporando los últimos adelantos en el área de las tecnologías de la información y las comunicaciones, que garantizan la compatibilidad y sostenibilidad de los productos a desarrollar, tales como: empleo de tecnologías basadas en internet (servicios Web XML), software libre, documentación de todo el proceso productivo, requerimientos de seguridad de software, independencia de la base de datos, desarrollo multiplataforma y empleo de estándares internacionales para los productos relacionados con la salud. *[Delgado, 2006]*

Actualmente, el escenario típico de cualquier sistema informático para la salud es insertarse en un panorama donde coexisten muchos otros con características heterogéneas, tanto desde el punto de vista estructural, como que se encuentran desplegados o distribuidos en diferentes ubicaciones físicas. La integración de sus datos es un hecho ineludible, lo cual consiste en la combinación de la información que se encuentra almacenada en diferentes componentes, proporcionándole al usuario final una vista unificada de la información. Tal integración tiene el propósito de facilitar a los usuarios, ya sea personal médico, administrativo, investigativo o al propio paciente, la obtención de un conglomerado de información, a partir de la ejecución de los diferentes procesos que se desarrollan en las instituciones de salud, de modo que se hace necesario considerar la existencia de una estricta política de compatibilidad, estándares y escalabilidad.

Los servicios Web XML *[Wolter, 2005],* se han convertido en una una opción factible para describir datos interoperables e interfaces de negocio, donde la plataforma no es un elementos a considerar, facilitando el desarrollo de transacciones más abiertas. Una de las posibles formas de concebir sistemas de software en la actualidad, consiste en diseñar aplicaciones en las cuales puedan acoplarse componentes que pertenezcan a organizaciones o dominios diferentes, para crear una aplicación más completa. Con esto se espera que se reduzcan los costos de interoperabilidad, mejore la productividad, aumente la flexibilidad en las aplicaciones, mejore la integración entre los sistemas y se reutilicen mejor los componentes de software. Considerando lo anterior se puede expresar que los servicios Web XML constituyen una de las variantes por la que se ha apostado, para garantizar la comunicación entre los componentes y soluciones desarrollados para el sector de la salud en el país.

Durante el proceso de construcción de cualquier sistema informático se debe considerar, desde el primer momento, la implementación de mecanismos que permitan gestionar los requerimientos de seguridad previamente especificados para el mismo. Uno de los principios de la seguridad informática, consiste en garantizar que los recursos de software de una organización, empresa o institución, se usen únicamente para los propósitos que fueron creados y dentro del marco previsto, además, que el acceso o modificación de la información manipulada por los mismos, sólo sea posible por las personas que se encuentren acreditadas y dentro de los límites de la autorización *[Redwine, 2008]*. La fortaleza de estos mecanismos, depende de la sensibilidad e importancia de la información gestionada, para evitar que la misma se encuentre en estados corruptos o inconsistentes.

Actualmente una de las principales dificultades en el ámbito del desarrollo de aplicaciones Web seguras, donde se hace uso de los servicios Web XML como mecanismo de interoperabilidad, lo constituye la gestión de identidades, que no es más que el conjunto integrado de políticas,

procedimientos y tecnologías que permite a las organizaciones facilitar y controlar debidamente el acceso de los usuarios internos y externos a las aplicaciones y sus funcionalidades, así como a recursos comunes. La gestión de identidades abarca uno de los aspectos fundamentales de la seguridad informática y de los sistemas: la política de protección conocida como Autenticación, Autorización y Auditoría (AAA). También incluye aspectos tales como el aprovisionamiento, la definición de roles o los flujos de autorización, cuestiones que se pueden ver completadas con la posibilidad de llevar a cabo auditorías. *[Pérez, 2007]*

Un sistema o componente de software, con el objetivo de gestionar los requerimientos de seguridad en aplicaciones Web, debe apoyarse en algunos elementos esenciales con el fin de lograr el propósito de su implementación. Un elemento primordial de este tipo de sistemas lo constituye el control de acceso a las aplicaciones *[Rodríguez, 2010]*, que es una poderosa herramienta para proteger el uso de un sistema completo o sólo a ciertas funcionalidades públicas; este control consta generalmente de dos pasos: la *autenticación*, que es el proceso de verificación de la identidad digital de un usuario, que realiza una petición para utilizar a un determinado recurso y la *autorización*, que es el proceso donde se comprueba que los usuarios con identidad válida, solo tengan acceso a aquellos recursos sobre los cuales se les asignaron privilegios. *[Microsoft, 2010]*

Actualmente cada sistema desarrollado para la informatización del SNS, por el Centro de Informática Médica (CESIM) de la Universidad de las Ciencias Informáticas (UCI), posee la gestión de los requerimientos de seguridad de forma aislada y descentralizada, implicando que en todos no definan de manera similar las políticas de control de acceso, autenticación y autorización, inclusive, en algunos casos, presentan implementaciones muy básicas, lo cual atenta considerablemente contra la seguridad e integridad de la información clínica que manipulan. Asimismo, al estar descentralizada la gestión de estos requerimientos, los usuarios de los sistemas implementados requieren diferentes credenciales para acceder a cada uno de ellos, no pudiendo obtener todos sus privilegios mediante un proceso único de autenticación. Del mismo modo, estos privilegios en los distintos niveles de acceso, se asignan individualmente a cada usuario, pues no se pueden crear roles o grupos de usuarios a los que se le pudiera asignar el derecho de ejecución a las funcionalidades que son comunes para todos ellos.

Lo anteriormente planteado implica que los usuarios para identificarse ante varias aplicaciones, destinadas al mismo sector, son forzados a recordar numerosos nombres de usuario y contraseñas, lo que promueve a que en la mayoría de los casos se elijan contraseñas sencillas, poniendo potencialmente en riesgo la seguridad del sistema. La utilización de una Arquitectura Orientada a Servicios (SOA) *[Hurtwitz, et al., 2007]* tiene entre sus objetivos la posibilidad de proveer acceso a los usuarios de múltiples servicios y aplicaciones con un mismo conjunto de credenciales. En la mayoría de los casos se encuentra que cada uno de los servicios o aplicaciones cuenta con su propia política o mecanismo de seguridad, lo que puede comprometer la seguridad del sistema en su conjunto, el cual se construye a partir de la reutilización de varios componentes previamente implementados y probados. El nivel de seguridad de todo un sistema es igual al nivel de seguridad del componente más inseguro que lo integre.

Prácticamente en todos los sistemas actualmente desarrollados o en fase de construcción, se ha obviado una fuerte estrategia de trazabilidad y auditoría de la información manipulada por los usuarios, como consecuencia de ello, no se lleva un control de los accesos, información modificada o eliminada

por los usuarios, implicando que en un momento determinado, no se pueda realizar la auditoría informática necesaria de esta información de carácter histórico. Es por ello importante considerar, la existencia de adecuados requerimientos para llevar a cabo la trazabilidad en los sistemas del ámbito sanitario, considerando la integridad y privacidad que debe poseer la información clínico administrativa que los mismos gestionan. Es importante considerar también, que las tablas de la base de datos, que se utilizan para almacenar las trazas tienden a crecer muy rápidamente, lo cual podría impactar negativamente en el rendimiento de las aplicaciones necesitando entonces un monitoreo constante de este parámetro desde la aplicación, con el fin de identificar el momento adecuado para realizar acciones de mantenimiento o salvas.

Si bien los beneficios aportados por los servicios Web XML en el ámbito empresarial son significativos, tales como la accesibilidad a la información, el intercambio dinámico y la autonomía, estos no están acordes con los modelos y controles tradicionales de seguridad; lo cual genera desafíos con el fin de obtener soluciones que permitan modificar los mecanismos de confidencialidad e integridad asociados con la autorización, para ello se deben considerar los siguientes elementos: *[Corredera, 2005]*

- Asegurar la autenticación entre el cliente que consume y el proveedor que expone el servicio Web. Ténganse en cuenta que la autonomía de los servicios, implica que los mismos puedan ser accesibles de forma pública.
- No permitir la autorización al consumo de servicios o procesos a aquellos usuarios que no tengan los privilegios para ello, es decir, los permisos deben poder ser administrados y controlados de forma centralizada para todos los componentes y servicios que interactúen entre si.
- Garantizar que la identificación y autenticación sea única para todos los usuarios de modo que puedan acceder a diversos sistemas, sin necesidad de autenticación en cada uno de ellos.
- Garantizar la integridad de los datos para protegerlos de alteraciones ya sean accidentales, o intencionadas.

Una debilidad funcional que poseen los sistemas de seguridad actuales, es que no permiten registrar para cada componente desplegado, los servicios Web XML que los mismos publican. Esto provoca que se asuma cierto nivel de confiabilidad entre los componentes y servicios que pudieran ser consumidos por estos, imponiéndose así un riesgo considerable sobre la información gestionada y compartida. Asociado también con el no registro de componentes, existe incapacidad para almacenar centralizadamente las direcciones Uniform Resource Locator (URL), donde los mismos poseen desplegados sus servicios, de modo que si en un momento determinado se modifica el nodo físico donde éste se encuentra hospedado, no se afecte el correcto funcionamiento del resto de los componentes que lo reutilizan, actualizando centralizadamente la modificación ocurrida; asimismo, no se puede realizar una distribución de servicios, de modo que un mismo servicio pudiese estar desplegado en varios nodos proveedores.

Al no atender centralizadamente las peticiones entre los componentes es necesario además, establecer para cualesquiera dos componentes que interactúen, una conexión para el consumo de sus servicios, de modo que si se pudiera establecer un nodo en el cual converjan todas las peticiones, estas podrían ser analizadas desde el punto de vista de disponibilidad, accesibilidad y posibilidad de

autorización en función de los privilegios del usuario autenticado. Otro aspecto adverso a considerar al no poseer este nodo de convergencia, es que no se pueden aprovechar las posibilidades asociadas a los procesos de sindicación de servicios y la implementación de algoritmos mucho más eficientes para atender todas las peticiones externas realizadas a un componente determinado, pudiéndose de este modo, poseer direcciones alternativas donde se encuentren hospedados los mismos servicios, rompiéndose con el temor de que en un momento determinado un servicio no esté disponible y colapse así el sistema que lo requiera.

La utilización del modelo arquitectónico orientado a servicios y basado en componentes, presupone como otra posible debilidad la falta de integridad en la información que ha sido compartida o utilizada con anterioridad entre los componentes, mediante el consumo servicios expuestos o publicados. Por ejemplo, si en un momento determinado se referencia cierta información obtenida de un componente proveedor en la base de datos de un componente consumidor, y esta se intenta eliminar o modificar en el primero de ellos, debe existir algún mecanismo que analice la posibilidad o no, de ejecución de la operación involucrada en dependencia de que esta información esté referenciada o no.

Entiéndase lo descrito como que en cualquier sistema gestor de base de datos entre sus requisitos deseados se encuentra, el mantenimiento de la integridad referencial, si se extiende este concepto el problema consiste en el mantenimiento de la integridad referencial en información que se encuentra distribuida y referenciada en diferentes componentes que se encuentran interoperando entre si.

Algunas experiencias para la implementación del mantenimiento de la integridad referencial del flujo de información basada en servicios Web XML, lo constituye la utilización de un conjunto de tablas denominadas caché en las bases de datos de los componentes, contenedoras de información no relacionada con su negocio de utilización frecuente, las que tienen que ser objeto de constantes actualizaciones en forma manual. Mediante esta solución parcial y poco elegante, no es posible configurar las restricciones de integridad que se establecen entre los componentes y servicios expuestos, así como especificar el tipo de dependencia que se establece entre ellos ya sea restrictiva, en cascada u otra que se desee definir.

Según la situación problémica anteriormente descrita, el ***Problema científico*** que se identifica para la presente investigación consiste en: *¿Cómo homogenizar la gestión de los requerimientos de Autenticación, Autorización y Auditoría de las aplicaciones basadas en servicios Web XML, desarrolladas por el Centro de Informática Médica?*

Se define como ***Objeto de estudio*** al *Proceso de gestión de requerimientos de seguridad.*

Teniendo como ***Campo de acción*** al *Proceso de gestión de requerimientos de Autenticación, Autorización y Auditoría.*

Para resolver el problema identificado se propone el siguiente ***Objetivo general:*** *Implementar requerimientos de seguridad para la homogenización de la gestión de los procesos de Autenticación, Autorización y Auditoría, de las aplicaciones basadas en servicios Web XML desarrolladas por el Centro de Informática Médica.*

Objetivos específicos:

1. Analizar el funcionamiento de las aplicaciones basadas en servicios Web XML y de sus principales requerimientos de seguridad.
2. Diseñar un modelo de requerimientos de seguridad para la homogenización de la gestión de los procesos de Autenticación, Autorización y Auditoría, en aplicaciones basadas en servicios Web XML.
3. Implementar el modelo obtenido para la homogenización de la gestión de los procesos de Autenticación, Autorización y Auditoría en aplicaciones basadas en servicios Web XML.
4. Validar los resultados de implementación obtenidos a partir de los indicadores de escalabilidad, rendimiento, reutilización de componentes y capacidad de generalización de la solución.

Para guiar la investigación y comprobar los resultados se propone la siguiente ***Hipótesis:*** *Si se implementan los requerimientos de seguridad propuestos, en las aplicaciones basadas en servicios Web XML entonces se homogenizan los procesos de Autenticación, Autorización y Auditoría, de los productos desarrollados por el Centro de Informática Médica.*

Entre los métodos de la investigación científica utilizados se encuentran los siguientes:

- **Métodos teóricos:** El *método analítico-sintético,* para descomponer el problema de investigación en el estudio por separado de los procesos de Autenticación, Autorización y Auditoría, así como del mantenimiento y control de las condiciones restrictivas de integridad, para luego sintetizarlos en la solución general del problema planteado.
- **Métodos lógicos**: El método *hipotético – deductivo,* para la definición de la hipótesis de la investigación y para proponer nuevas líneas de trabajo a partir de los resultados parciales obtenidos, los cuales serán verificados posteriormente mediante métodos empíricos. El método de la *modelación*, con el fin de obtener modelos, mediante abstracciones, para explicar e interpretar la realidad y el método *sistémico,* para lograr que todos los elementos de la estructura y concepción de la solución funcionen como un todo, lo que representa la expresión de su comportamiento.
- **Métodos empíricos**: El método de la *medición,* como procedimiento para obtener información numérica acerca de algunas características de la solución obtenida, como por ejemplo su rendimiento y el método *experimental*, para realizar las pruebas unitarias y de integración de la solución, mediante el desarrollo de casos de prueba experimentales previamente diseñados.

La novedad y el aporte práctico de la investigación se concretan en la implementación de un modelo que describe cómo deben gestionarse los procesos de Autenticación, Autorización y Auditoría para aplicaciones basadas en servicios Web XML. El funcionamiento de la solución obtenida no está limitado sólo a productos desarrollados para el ámbito de la salud, sino que es totalmente escalable a cualquier otro tipo de sistema cuyo mecanismo de interoperabilidad sean los servicios, es decir, que el mismo posibilita una amplia posibilidad de reutilización y generalización, uno de los principios más deseados en el desarrollo de software actual.

Otro elemento importante a considerar, es que los diseñadores de aplicaciones sólo deberán centrarse en aquellos requerimientos que son propios de su negocio e incorporar posteriormente, el uso de los

requerimientos de seguridad descritos en la solución propuesta. Ello disminuye el tiempo de desarrollo de las aplicaciones y aumenta los niveles de seguridad de las mismas, pues se reutiliza un software ya probado y especializado en los procesos antes descritos.

La solución obtenida posee una amplia relevancia en el contexto sobre el cual se desarrolla, debido a que el país viene realizando importantes esfuerzos en pos del desarrollo de la informática en todas las actividades que se realizan en el ámbito de la salud. El desarrollo de estos procesos de seguridad, garantiza una homogénea y estandarizada forma de gestionar los mismos, basada en modelos internacionales que describen y documentan satisfactoriamente cuáles son los principios necesarios para garantizar la seguridad en las aplicaciones Web.

El desarrollo del presente documento se encuentra estructurado en tres capítulos. El primero de ellos, ***"FUNDAMENTOS TEÓRICOS PARA LA GESTIÓN DE PROCESOS DE AUTENTICACIÓN, AUTORIZACIÓN Y AUDITORÍA"***, tiene como objetivo abordar los diferentes aspectos teóricos y conceptuales que constituyen la base para el desarrollo de la propuesta de solución. Seguidamente en el capítulo denominado ***"PROPUESTA DE PROCESOS DE AUTENTICACIÓN, AUTORIZACIÓN Y AUDITORÍA PARA APLICACIONES BASADAS EN SERVICIOS WEB XML"***, se presenta un marco conceptual asociado al campo de acción, llegándose a un acuerdo sobre las funcionalidades, requerimientos deseados y el objeto de automatización, así como que se describen los servicicios implementados para la gestión de la autenticación, autorizacuión, auditoría así como para el control centralizado de las peticiones de consumo de un determinado servicio.

El tercer capítulo titulado ***"VALIDACIÓN DE LA PROPUESTA DE PROCESOS DE AUTENTICACIÓN, AUTORIZACIÓN Y AUDITORÍA PARA APLICACIONES BASADAS EN SERVICIOS WEB XML"*** aborda una valoración crítica tanto desde un enfoque cuantitativo como cualitativo de los resultados obtenidos, a partir del desarrollo de un conjunto de pruebas experimentales realizadas a los resultados de implementación obtenidos.

CAPÍTULO I: FUNDAMENTOS TEÓRICOS PARA LA GESTIÓN DE PROCESOS DE AUTENTICACIÓN, AUTORIZACIÓN Y AUDITORÍA

El presente capítulo tiene como objetivo abordar los diferentes elementos que brindan la base teórico conceptual para el desarrollo de la solución propuesta. Valorándose de forma crítica las tendencias y tecnologías actuales, así como los antecedentes asociados con la concepción de procesos para la gestión de requerimientos de autenticación, autorización y auditoría. De este modo, se podrá realizar una correcta interpretación de la situación problémica y del problema a resolver.

La complejidad en la administración de los servicios de red en las organizaciones actuales está demostrando que los métodos tradicionales de control de acceso a los recursos, basados en mecanismos como nombre de usuario y contraseña o certificados de identidad, son incapaces de adaptarse a los requerimientos impuestos por la propia infraestructura interna de la organización y el tipo de servicio ofrecido, más aún, cuando se presentan en escenarios con varias organizaciones involucradas en la compartición de estos servicios. Estas organizaciones están optando por implantar mecanismos de control de acceso en base a información adicional a la identidad del usuario, como atributos, roles o pertenencia a grupos. De este modo, una vez autenticado el usuario, el sistema autoriza o no, el acceso a determinados recursos en función de dicha información adicional. *[López, 2006]*

1.1 Modelo de Autenticación, Autorización y Auditoría (AAA)

El modelo AAA se definió como un marco de funcionamiento común para gestionar las operaciones de Autenticación, Autorización y Auditoría en un escenario de control de acceso a la red, de modo que fuese compatible con las distintas tecnologías existentes, extensible, y que permitiera su gestión entre diferentes dominios. Es importante comprender qué se entiende en el modelo AAA por Autenticación, Autorización y Auditoría, para identificar correctamente los requerimientos que debe cumplir un buen sistema de este tipo: *[López, 2006]*

- **Autenticación:** Cuando un usuario o cliente requiere acceso a la red, debe presentar información personal que permita establecer de manera unívoca su identidad dentro del entorno. El usuario podrá presentar cualquier tipo de credencial de identidad (nombre de usuario y contraseña, certificado de identidad secreto o compartido), usando para ello algunos de los mecanismos de control de acceso. El servidor AAA y concrétamente su servicio de autenticación, deberá obtener dichas credenciales y de forma interna (mediante bases de datos, políticas de autenticación o servicios auxiliares), determinar si ese usuario es realmente quien dice ser.
- **Autorización:** El proceso de autorización deberá determinar si el usuario, una vez autenticado (si es necesario), tiene o no derecho a hacer uso del servicio solicitado. Para ello, la decisión de autorización puede tomarse en función de diferentes credenciales del usuario. Por ejemplo, la autorización puede otorgarse simplemente comprobando que el usuario ha sido autenticado correctamente, aunque en entornos más especializados puede otorgarse en base a credenciales de autorización, como pueden ser sentencias de atributos que identifican el rol o

papel que juega el usuario dentro de la organización. En este último caso, la manera en que el servicio AAA recupera está información de autorización, depende del escenario concreto de acceso, ya que puede ser presentada directamente por el usuario o recuperada desde repositorios bien conocidos.

- **Auditoría:** Existen entornos donde se requiere llevar un control detallado de las operaciones que realizan los usuarios de los sistemas, por ejemplo, en un entorno de alta seguridad donde toda la información operacional debe ser almacenada, como por ejemplo los entornos sanitarios. En este tipo de entornos, cualquier información relativa a una sesión de usuario (acceso al servicio, tipo de servicio obtenido, cantidad de tráfico enviado o recibido, tiempo de conexión), debe almacenarse para que después pueda ser usada apropiadamente durante el proceso de auditoría.

1.1.1 Estructura básica de un sistema de Autenticación, Autorización y Auditoría (AAA)

El modelo incorpora en su estructura básica un servidor genérico AAA que provee un conjunto de interfaces específicas para los procesos que este describe, de modo que puedan ser utilizadas por un conjunto de aplicaciones, componentes o servicios que necesiten los requerimientos que se especifican en el modelo. La arquitectura básica de un escenario AAA está compuesta principalmente por tres elementos: el cliente; el servidor AAA, que es el responsable de gestionar los procesos de autenticación y autorización de los usuarios; y los proveedores de servicios, que son las entidades que ofrecen los servicios finales y que pueden o no, formar parte de la infraestructura AAA. *[López, 2006]*

Este modelo es un elemento clave en la solución propuesta por la presente investigación, pues constituye la base para su correcto diseño e implementación. En sentido general, se puede decir que el modelo simplemente describe cómo deben llevarse a cabo los procesos descritos, a modo de guía, no especificando en ningún momento, los elementos funcionales que deben formar parte de la solución que lo tendrá como base teórico conceptual.

En un modelo arquitectónico de seguridad no intrusivo, basado en AAA, se requiere además un componente adicional que actúa como proxy o pasarela (gateway). Este componente, se encarga de gestionar tanto las peticiones de autenticación como las de autorización. En la siguiente figura se muestra este esquema de funcionamiento:

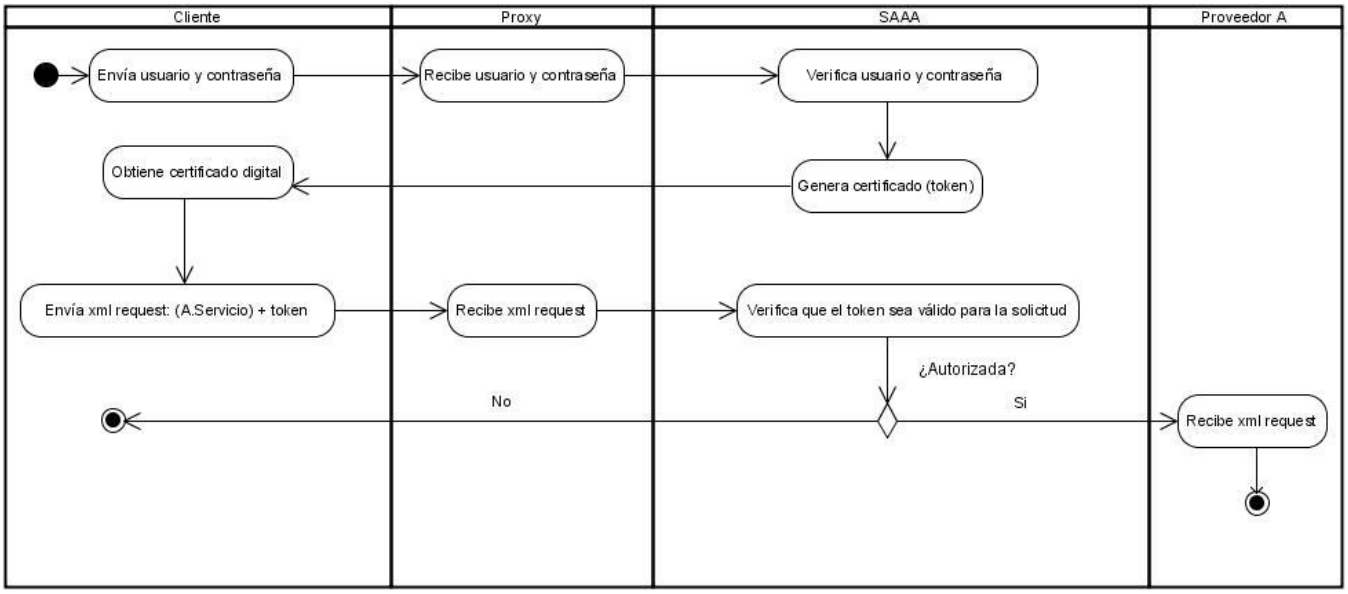


Figura 1.1: Estructura básica de un Sistema de Autenticación, Autorización y Auditoría

Como se muestra en la **Figura 1.1**, durante el proceso de Autenticación/Autorización se involucran cuatro actores fundamentales: la aplicación cliente; un componente de servicios intermedio proxy o pasarela; el Sistema de Autenticación, Autorización y Auditoría (SAAA) y el proveedor de servicios al que se le realiza determinada petición. El *Cliente,* solicita la ejecución de un servicio determinado publicado por el *Proveedor A*, para ello primeramente debe presentar sus credenciales de autenticación, a partir de lo cual, se genera su certificado digital *(token)* el cual será utilizado, en lo adelante, como mecanismo para comprobar la autenticidad del cliente que solicita la petición.

Una vez obtenido este certificado, se construye la solicitud del servicio y se envía al *Proxy*, este posteriormente mediente algún servicio de autorización que provea el SAAA, verifica si el cliente, con el certificado generado, tiene los permisos o privilegios necesarios para consumir el servicio solicitado. Si la respuesta es satisfactoria, se redirecciona la petición hacia el proveedor finalizando así el proceso, en el caso contrario, se notifica al cliente que no se tiene autorización para realizar la petición solicitada, culminando así también el proceso descrito. En todo este flujo de actividades, hay una actividad subyacente, el registro de las trazas (logs), mediante un servicio para este fin, que debe proveer el SAAA. En el proceso descrito en la figura anterior se registran trazas en los subprocesos de autenticación y autorización, ya sean estos satisfactorios o no.

1.2 Seguridad en aplicaciones Web

En el mundo de hoy, debido al alto grado de interconectividad existente entre estaciones de trabajo y otros dispositivos electrónicos; la Web se ha convertido en un importante medio de comunicación e intercambio de información, que ofrece un acceso abierto a un conjunto de datos que explícitamente se hacen públicos. Por tanto, es necesario limitar el acceso a informaciones reservadas o útiles para un conjunto restringido de personas. Otro aspecto que cobra especial importancia es la seguridad de la información que se intercambia en la Web, que cada vez está más expuesta, a personas que incurren en delitos informáticos.

La principal vulnerabilidad de la información en una aplicación Web *[PCMAG, 2010]*, se presenta en que la información viaja a través de la red, lo que implica que puede ser sustraída por cualquier persona, no obstante, existe una serie de vulnerabilidades como la falsificación de identidad, el robo de sesiones autenticadas, la falsificación o modificación de los mensajes, entre otras. Una aplicación Web que se considere segura, debe contemplar mecanismos que garanticen que efectivamente, ninguna persona ajena al sistema pueda modificar, obtener o falsificar datos de la misma.

El proyecto OWASP (The Open Web Application Security Project, por sus siglas en inglés) *[OWASP, 2010]* tiene como objetivo ofrecer una metodología, de libre acceso y utilización, que puede ser utilizada como material de referencia por parte de los arquitectos de software, desarrolladores, fabricantes y profesionales de la seguridad, involucrados en el diseño, desarrollo, despliegue y verificación de la seguridad de las aplicaciones y servicios Web.

Según esta guía, los principios básicos de seguridad que debe cumplir cualquier aplicación o servicio Web son: *[OWASP, 2005]*

- **Validación de la entrada y salida de información:** La entrada y salida de información es el principal mecanismo que dispone un atacante para enviar código malicioso contra el sistema. Por tanto, siempre debe verificarse que cualquier dato entrante o saliente, es apropiado y en el formato que se espera. Las características de estos datos deben estar predefinidas y deben verificarse en todas las ocasiones.
- **Diseños simples:** Los mecanismos de seguridad deben diseñarse para que sean lo más sencillo posible, huyendo de elementos que compliquen excesivamente el trabajo a los usuarios. Si los pasos necesarios para proteger de forma adecuada una función o módulo son muy complejos, la probabilidad de que estos no se ejecuten de la forma esperada será muy elevada.
- **Utilización y reutilización de componentes de confianza:** Debe potenciarse el principio de la reutilización en todos los casos en que sea posible. Por tanto, cuando exista un componente que resuelva un problema determinado correctamente, lo más adecuado es reutilizarlo.
- **Defensa en profundidad:** Nunca confiar en que un componente realizará su función de forma permanente y ante cualquier situación. Es necesario disponer de los mecanismos de seguridad suficientes para que, cuando un componente del sistema falle ante un determinado evento, otros sean capaces de detectarlo.
- **Proveer servicios de autenticación segura a las aplicaciones Web:** Las aplicaciones deberán poseer reglas o mecanismos de autenticación donde los certificados de seguridad establecidos puedan poseer variables que se modifiquen durante el tiempo en que la sesión se encuentre activa.
- **Verificación de privilegios:** Los sistemas deben diseñarse para que funcionen con los menos privilegios posibles. Igualmente, es importante que los procesos únicamente dispongan de los privilegios necesarios para desarrollar su función, de forma que queden compartimentados.
- **Ofrecer la mínima información:** Ante una situación de error o una validación negativa, los mecanismos de seguridad deben diseñarse para que faciliten la mínima información posible. De

la misma forma, estos mecanismos deben estar diseñados para que una vez denegada una operación, cualquier operación posterior sea igualmente denegada.

- **Manejo de errores y auditoría:** Las soluciones implementadas deben poseer un fuerte proceso de trazabilidad y control de las transacciones en las que se involucren los usuarios, para saber en todo momento, quién realizó cada una de estas y qué información sufrió cualquier tipo de modificación.

Considerando estos elementos se hace necesario cada día más, el desarrollo de aplicaciones o componentes de software especializados en el control de acceso a las aplicaciones Web y la gestión de sus requerimientos de seguridad. Se debe considerar además, que el desarrollo de aplicaciones Web ha alcanzado un auge sin precedentes pues las soluciones se orientan a fines y propósitos cada vez más especializados. Esto a su vez, implica que las mismas manipulen información y datos de elevada sensibilidad, la cual podría encontrase sin la debida protección, en estados corruptos e inconsistentes, perdiendo así los principios básicos de la información, es decir, la confidencialidad, integridad y disponibilidad. *[ISO 27000, 2008]*

1.3 Modelo de interoperabilidad basado en Servicios Web XML

Para comenzar es preciso definir qué se entiende por Arquitectura de Software. Según Roger Pressman, esta representa, ***"(…) la descripción de los subsistemas o componentes de un sistema informático y las relaciones entre ellos (…)".*** Es decir, el conjunto de decisiones significativas sobre la organización del sistema, la selección de los elementos estructurales y las interfaces por las que está compuesto. Describe los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. *[Pressman, 2005]*

En la actualidad nadie duda que cuando se desea obtener mejoras significativas, en cuanto a productividad y calidad, en el proceso de desarrollo de software, una de las piezas imprescindibles es la reutilización sistemática de componentes de software. Los entornos de producción responden cada vez más a las características del desarrollo de software actual en el que se impone la distribución de los recursos técnicos y humanos, la coexistencia de tecnologías heterogéneas, entre otras, lo que introduce una complejidad a manejar a la hora de dotar de un soporte operativo a las organizaciones que opten por implantar la reutilización sistemática en sus procesos de desarrollo. *[Crespo et al., 2005]*

Las instituciones exigen aplicaciones cada vez más complejas, con menos tiempo y presupuesto. Para crear estas aplicaciones, se requiere en muchos casos de funcionalidades ya implementadas como parte de otros sistemas. Exponer procesos de negocio como servicios, es una de las posibles vías para un exitoso proceso de reutilización. Esto permite que otras piezas de software puedan hacer uso de servicios de manera natural, sin importar su ubicación física. La Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés) y Basada en Componentes (CBA, por sus siglas en inglés); define los servicios por lo que estará compuesto el nuevo sistema a desarrollar y sus interacciones, donde no importará en lo absoluto las tecnologías que se utilicen en su proceso de construcción.

1.3.1 Desarrollo de aplicaciones orientadas a servicios y basadas en componentes

La Arquitectura Orientada a Servicios nace como una estrategia de integración y constituye una tendencia creciente que intenta reconciliar la visión técnica y de negocios, basándose en estándares abiertos y promoviendo la interoperabilidad entre diversas organizaciones y plataformas de manera

eficiente y flexible a los cambios. Actualmente todos los proveedores de tecnologías se han orientado hacia el soporte de este tipo de arquitecturas tanto en empresas pequeñas o en crecimiento, como en grandes corporaciones, debido a que SOA no es una moda pasajera, sino que es una alternativa que permite enfrentarse a los retos de las organizaciones, que necesitan una respuesta flexible y rápida en entornos competitivos y de alta complejidad.

Esta define diferentes capas de software, estas son: aplicativa básica, que son sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad. La segunda es la de exposición de funcionalidades, donde las mismas, son expuestas en forma de servicios. Otra es, la de integración de servicios, que facilita el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración. Además, la de composición de procesos, que los definen en términos del negocio y sus necesidades, y varía en función del negocio. Por último, la de entrega, donde los servicios son desplegados a los usuarios finales.

Entre los beneficios que proporciona SOA se encuentran: *[SOFTEL, 2006]*

- La reducción de costos y tiempo en el desarrollo de aplicaciones: reutilización de los componentes de aplicaciones existentes para resolver problemas comunes a otras aplicaciones. Como consecuencia, también se reducen los costos de mantenimiento y se logra aumentar la robustez del nuevo sistema, al utilizarse software ya probado.
- Facilita el proceso de integración: se interactúa con elementos que se abstraen de la ubicación de los servicios y tecnología con la que fueron creados los mismos.

Existen varias definiciones de componente, aunque todas giran entorno a que es una pieza de software que posee una interfaz y función bien definidas. El desarrollo de software basado en componentes, se centra en la construcción de aplicaciones complejas mediante ensamblado de módulos o componentes, que han sido previamente diseñados por otras personas a fin de ser rehusados en múltiples aplicaciones. Entre las características más relevantes de esta arquitectura se pueden citar la modularidad y la reusabilidad, rasgos en los que coincide con la arquitectura orientada a servicios. *[Aruquipa et al., 2007]*

1.3.2 Servicios Web XML como mecanismo de interoperabilidad

Los servicios Web *[Cabrera et al., 2004]* tienen gran importancia en la tendencia de la computación distribuida en internet. Los estándares abiertos y el enfoque en la comunicación y la colaboración entre personas y aplicaciones, han creado un entorno donde los servicios Web XML (eXtensible Markup Language), están favoreciendo una plataforma para la integración de aplicaciones. Los servicios Web funcionan como aplicaciones "*independientes*" que ofrecen interfaces de acceso basadas en mensajes conformes a estándares para el desarrollo Web. Estos cuentan con todos los beneficios propios de las tecnologías distribuidas en capas y constituyen un paso adelante en las aplicaciones basadas en componentes.

Un *servicio Web* es un software diseñado para soportar la interacción entre máquinas de una red que contiene una interfaz bien definida descrita en formato WSDL (Web Service Description Language) y permite la interacción con otros sistemas a través de mensajes SOAP (Simple Object Access Protocol), que cumplen con los elementos descritos en el WSDL. *[W3C, 2004].* Sun Microsystems en su

publicación “Building Web Services” *[Sun, 2002]* ha identificado a los servicios Web, como componentes de aplicaciones distribuidas que solucionan ciertos problemas y que siguen un estándar que los hace accesibles externamente.

Aunque las definiciones consultadas muestran diferencias, se puede concluir que en todos los casos se identifican, como base de la arquitectura, los estándares XML, WSDL y SOAP.

En la **Tabla 1.1** se muestran algunas características de los servicios Web atendiendo a varios indicadores de construcción y desempeño. *[Ronda, 2004]*

Indicador	**Desempeño de los Servicios Web**
Acoplamiento	Débilmente acoplados, se basan en protocolos estándares y trabajan conjuntamente con independencia de dónde residen y de cómo han sido implementados.
Acceso	Exponen su funcionalidad mediante el estándar WSDL. Accesibles a través de mensajes que se basan en protocolos abiertos e independientes de plataforma, como por ejemplo SOAP.
Control	Son controlados y administrados de forma centralizada e independiente de las aplicaciones clientes.
Ubicación	Son transparentes a su localización en la red. A través del estándar WSDL se pueden conocer las particularidades de los servicios y ubicación.
Funcionamiento	Habilitan direcciones URL que mediante el paso de una serie de parámetros, ejecutan algún proceso del negocio y devuelven información en distintos formatos (generalmente documentos XML).
Rendimiento	Su rendimiento sobre la red se ve afectado, en ocasiones, debido a que todo el flujo de información es en formato de texto.
Políticas de seguridad, tolerancia a fallos y seguridad	Son aplicaciones independientes que definen sus propias políticas de seguridad, configuración y escalabilidad. Ofrece un conjunto de tecnologías adicionales para el manejo transaccional, la seguridad, entre otras.

Tabla 1.1: Comportamiento de algunos indicadores de desempeño de los servicios Web

Una de las estrategias para el desarrollo de aplicaciones empresariales actuales, es facilitar su diseño e implementación en pos de llevar a cabo procesos de integración; además de que se debe alentar y motivar la construcción de aplicaciones, que provean interfaces bien definidas mediante servicios a futuras aplicaciones que las requieran. De esta manera, una aplicación final simplemente consume u orquesta la ejecución de un conjunto de servicios, añade su lógica de negocio particular y le presenta una interfaz al usuario final, donde este último se abstrae de la forma en que se encuentra construida la aplicación. **(Ver Anexo 1)**

Un servicio debe ser una aplicación completamente autónoma e independiente. A pesar de esto, no es una isla de información, porque expone una interfaz de llamado basada en mensajes, capaz de ser accedida a través de la red. Generalmemente, los servicios incluyen tanto lógica de negocio como manejo de estados (datos), relevantes a la solución del problema para el cual fueron diseñados. La manipulación de los estados se define en función de las reglas de negocio asociadas al servicio involucrado. *[Guinea et al., 2007]* **(Ver Anexo 2)**

En la **Figura 1.2** se describe un esquema que muestra la integración existente entre los diferentes estándares subyacentes al funcionamiento de los servicios Web XML:

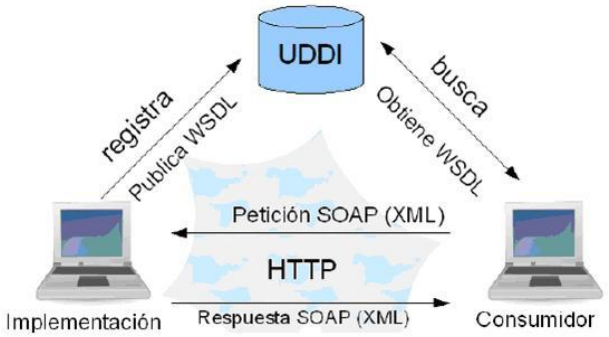


Figura 1.2: Representación general de funcionamiento de los servicios Web

1.3.3 Requerimientos de calidad en servicios Web (QoS)

A continuación se enuncian los requerimientos de calidad en servicios Web: *[Ortega, 2006]*

Disponibilidad: La disponibilidad es el aspecto de calidad que indica si el servicio Web está presente o listo para su uso inmediato. Representa la probabilidad en que un servicio esté disponible. A este término se asocia el tiempo de reparación, el cual representa el tiempo que le toma al servicio recuperarse de fallas.

Accesibilidad: La accesibilidad es el aspecto de calidad de un servicio que representa el grado en el que se capaz de servir una solicitud a un servicio Web. Puede ser expresada como la medida de probabilidad que denota la tasa de éxito, o la posibilidad de desempeño de un servicio exitosamente en la medida que pase el tiempo. Puede ocurrir que un servicio Web esté disponible y no esté accesible. La alta accesibilidad de un servicio Web puede alcanzarse por medio de la construcción de sistemas altamente escalables. La escalabilidad se refiere a la capacidad de servir consistentemente las solicitudes independientemente de las variaciones en el volumen de solicitudes.

Integridad: La integridad es el aspecto de calidad relacionado con la manera en que se mantiene el buen funcionamiento del servicio Web con respecto a su código. La apropiada ejecución de las transacciones de los servicios Web es lo que garantiza el buen funcionamiento. Una transacción se refiere, a la secuencia de actividades que tienen lugar en una unidad de trabajo. Todas las actividades tienen que ser terminadas para que la transacción sea exitosa.

Funcionamiento: Es el aspecto que se mide en términos de rendimiento y latencia. Alto rendimiento y baja latencia muestran un buen funcionamiento de un servicio Web. El rendimiento está dado por el número de servicios Web que solicitan un servicio en un período de tiempo dado. La latencia es el tiempo de viaje de ida y vuelta, entre enviar una solicitud y recibir la respuesta.

Fiabilidad: Representa el grado en el cual es capaz de mantenerse el servicio y la calidad del mismo. El número de fallas mensuales o anuales representan una medida de fiabilidad de un servicio Web. En otras palabras, la fiabilidad se refiere a las entregas ordenadas y seguras para mensajes enviados y recibidos por servicios proveedores y consumidores.

Regulación: Este aspecto de seguridad tiene que ver con las reglas, las leyes, conformidad con los estándares, y el acuerdo de la capa de servicio establecido.

Seguridad: Es el aspecto de calidad que provee confidencialidad y no repudio por medio de la autentificación de las partes involucradas, los mensajes encriptados, y control de acceso. Se le ha concedido a la seguridad, un nivel de importancia mayor dado que los servicios Web son invocados desde la red pública. EL proveedor de servicios puede tener diferentes niveles o capas para proveer seguridad en dependencia de los servicios solicitantes.

1.3.4 Seguridad en la arquitectura de referencia de servicios Web XML

El hecho de publicar una serie de servicios Web accesibles puede suponer una ventaja comercial, pero en ocasiones, la seguridad en determinados servicios puede ser de vital importancia y es necesario evitar acceso de terceros a la información que viaja en los mensajes SOAP. Por ello es necesario tener en cuenta los siguientes factores para asegurar la seguridad del servicio, descritos en la norma ISO 7498-2: *[Atkinson, 2002]*

- Autenticación: Asegurar que la procedencia del mensaje sea quien realmente dice ser.
- Autorización: Para determinar si se tienen privilegios para realizar una determinada acción.
- No repudio: Para poder demostrar que un determinado cliente realizó una determinada acción.
- Confidencialidad: Los mensajes no puedan ser leídos por terceros.
- Integridad: Asegurar a los receptores que el mensaje no ha sido modificado.

Debido a los estándares que son utilizados por los servicios Web y en concreto, al uso de SOAP, las técnicas de seguridad convencionales que se han venido usando en internet, ya no son suficientes. Con SOAP, cada mensaje simple que se intercambia, realiza múltiples saltos y es enrutado a través de numerosos puntos antes de que alcance su destino final. Es por ello, que los servicios Web necesitan tecnologías que protejan los mensajes desde el principio hasta el final. Por estas causas, existen un conjunto de estándares que se pueden usar para garantizar la seguridad a nivel de mensaje *[Paz, 2007]*. Estos son: *[W3C, 2010]*

- *XML Encryptation*: Evita que los datos se vean expuestos a lo largo de su recorrido. *XML Encryptation* es una recomendación cuya función principal es asegurar la confidencialidad de partes de documentos XML a través de la encriptación parcial o total del documento transportado. Además de describir la estructura y la sintaxis de los elementos XML necesarios para presentar información cifrada, especifica los pasos que se deben seguir a la hora de cifrar y descifrar documentos XML (o partes de ellos). Este elemento será utilizado posteriormente en la solución propuesta para el envío del token de autenticidad del usuario registrado, en la cabecera de los documentos SOAP.
- *XML Digital Signature*: Asocia los datos del mensaje al usuario que emite la firma, de modo que este usuario, es el único que puede modificar dichos datos. En esta versión de la solución propuesta aún no incorpora la utilización del estándar XML Digital Signature.
- *SAML (Security Assertion Markup Language)*: SAML, está desarrollado por OASIS (Organization for the Advancement of Structured Information Standards) hace posible que los servicios Web intercambien información de autentificación y autorización entre ellos, de modo que un servicio Web confíe en un usuario autentificado por otro servicio Web.

Al aplicar seguridad en el nivel de aplicación, se puede modificar el propio mensaje SOAP para que incorpore las credenciales del usuario que quiere consumir el servicio. Este enfoque requiere que tanto el servicio como el cliente, incorporen el mecanismo de autenticación para poder ofrecer el acceso al mismo. La aplicación de este método haría viajar los credenciales en texto plano, a no ser que se utilice el protocolo Secure Socket Layer (SSL, por sus siglas en inglés) *[VeriSing, 2010]* para el consumo del servicio Web. En cualquier caso, este enfoque es análogo a la autenticación basada en formularios, admitiendo varias maneras de ser llevado a cabo:

- Enviar los credenciales en cada mensaje SOAP transmitido, de modo que el servidor pueda validarlos para cada petición.
- Enviar los credenciales en el primer mensaje SOAP transmitido, de modo que el servidor pueda validarlos y establecer internamente una sesión para el usuario. El servidor devuelve al usuario un identificador de sesión que el cliente podrá usar en cada petición posterior, y el servidor simplemente comprobar, si la sesión fue establecida previamente, y si el acceso al recurso solicitado, es concedido al usuario propietario de la sesión indicada. Con la finalidad de aumentar la seguridad en este enfoque, es recomendable establecer periodos de caducidad para las sesiones.

Asimismo, la seguridad se ha conseguido, en ocasiones, mediante el uso de métodos externos a los servicios Web, por el ejemplo mediante el uso de SSL sobre HTTP de lo cual se obtiene HTTPS (Hypertext Transfer Protocol Security), pero esta solución garantiza la seguridad solamente en la capa de transporte. Para solucionar el problema en la capa de aplicación, se crea la especificación WS-Security (Web Service Security) que define una serie de extensiones para el protocolo SOAP, que permite el intercambio entre el cliente y el servidor de tokens de seguridad. El intercambio de tokens junto con mecanismos de encriptación y firmado, permiten conseguir los factores anteriormente nombrados.

WS-Security permite tratar la autenticación y autorización apropiada de los servicios así como la protección de los mensajes intercambiados de modo que estos sean protegidos de accesos y modificaciones no autorizadas. Esta especificación provee un modelo unificado que utiliza las tecnologías existentes y que permite que las aplicaciones intercambien mensajes SOAP de forma segura. Aprovechando las potencialidades de WS-Security un conjunto de aplicaciones pueden crear un dominio de seguridad externalizando sus procesos de autenticación y autorización de manera que comparten un elemento *UsernameToken* de seguridad generado por un proveedor de identidad y definido en el perfil *Username Token Profile,* los que a menudo pueden ser firmados mediante *XML Digital Signature* para alcanzar mayor nivel de seguridad.

Adicionalmente, WS-Security describe cómo codificar estos tokens de seguridad, específicamente describe cómo codificar certificados *X.509* y *tickets de Kerberos*. También incluye mecanismos de extensibilidad que pueden ser utilizados para describir las características de las credenciales que se incluyen en un mensaje. De esta forma, se extiende el marco de trabajo de mensajería distribuida SOAP mediante un mensaje que incluye una cabecera de seguridad con el nombre de usuario y el token generado por el servicio de autenticación. Todo esto implica que la administración global de las identidades digitales y sus derechos de acceso se simplifique y resulte más fiable. Adicionalmente, los

usuarios pueden llegar a acceder a todas las aplicaciones con un único proceso de autenticación (Single Sign-on).

A pesar de esos beneficios, WS-Security también tiene algunas desventajas. Entre ellas se pueden mencionar, que la configuración de WS-Security puede ser compleja y que a veces añade mucho volumen a las cabeceras los mensajes que se intercambian. Ese volumen añadido, puede tener un impacto significativo en el rendimiento cuando se envían los datos entre el cliente y el servidor en una red. La calidad de la conexión de red entre el cliente y el servidor, determina el nivel del impacto que ese volumen añadido tiene en el rendimiento, pero queda claro que, mientras mayores son los mensajes, más lento es su intercambio. *[Sosnoski, 2009]*

En la mayoría de las aplicaciones, WS-Security y XML Encryption son una exageración cuando se trata de preservar la confidencialidad. Si el servicio es contactado directamente por los clientes (y no indirectamente, a través de otros servidores) y realiza directamente todo el procesamiento necesario, es posible usar sólo conexiones SSL (HTTPS) para el acceso, para proporcionar garantías excelentes de confidencialidad a un costo de rendimiento mucho más bajo que el de *WS-Security*. Sin embargo, ese enfoque sólo funciona en conexiones directas con los clientes. *[Sosnoski, 2009]*

Antes de pasar a la situación problémica es importante resaltar que el problema asociado al campo de acción y que ocupa a la presente investigación, es garantizar una autenticación y autorización mediante políticas homogéneas de acceso, para las aplicaciones y los servicios expuestos por estas, desarrolladas por el Centro de Informática Médica de la Universidad de las Ciencias Informáticas (UCI), aunque se podrá analizar posteriormente que en la propuesta de solución, muchas de las ideas expuestas con anterioridad son tomadas como punto de partida para su diseño e implementación.

1.3.5 Situación problémica y objeto de automatización

Actualmente cada sistema destinado a la informatización del SNS cubano desarrollado por el CESIM, posee la gestión de los requerimientos de seguridad de forma aislada, descentralizada e independiente, implicando que no se defina de manera similar las políticas de control de acceso, autenticación y autorización. Las soluciones actuales en algunos casos presentan implementaciones básicas, orientadas fundamentalmente hacia la autenticación, a partir del uso de simples mecanismos basados en usuario/contraseña, lo cual atenta considerablemente contra la seguridad e integridad, de la información clínica que se gestiona.

Asimismo, al estar descentralizada la gestión de estos requerimientos, los usuarios de los sistemas implementados requieren diferente conjunto de credenciales para acceder a cada uno de ellos, no pudiendo obtener todos sus privilegios mediante un proceso único de autenticación. Del mismo modo, estos privilegios, en los distintos niveles de acceso, se asignan individualmente a cada usuario, pues no se pueden crear roles o grupos, a los que se le pudiera asignar el derecho de ejecución de las funcionalidades o servicios que son comunes para todos ellos, aunque después se pudiese restringir estas funcionalidades comunes, a un usuario en específico.

Lo anterior implica que los usuarios para identificarse ante varias aplicaciones, son forzados a recordar numerosos nombres de usuario y contraseñas, así como que estas últimas, son elegidas con una complejidad relativamente débil, poniendo potencialmente en riesgo la seguridad del sistema. La utilización de una Arquitectura Orientada a Servicios (SOA), tiene entre sus objetivos la posibilidad de proveer acceso a los usuarios a múltiples servicios y aplicaciones con un mismo conjunto de

credenciales. En la mayoría de los casos se encuentra, que cada uno de los componentes y servicios cuentan con su propia política o mecanismo de seguridad, lo que puede comprometer la seguridad del sistema en su conjunto, el cual se construye a partir de la reutilización de varios componentes previamente implementados y probados.

Ejemplificando al nivel de la interacción entre diferentes componentes, cada servicio Web participante en una interacción, podría requerir autenticación de la parte que solicita su ejecución. Es decir, si cierto servicio *A* dirige una petición al servicio *B*, éste puede requerirle unas credenciales junto con una demostración de que le pertenecen, tal como un par nombre de usuario (credencial)/contraseña (demostración) o un certificado de autenticidad, este es uno de los principales problemas de la autenticación, cuyo origen radica en la naturaleza heterogénea de los servicios Web. Otro punto a resolver es, poder definir un modelo de autenticación Single Sign-On de forma que, un servicio *A* que necesita interactuar con otros 6 servicios, para completar un proceso de negocio *P*, no necesite autenticarse más que una vez, frente al primero de ellos, para poder completar la operación bajo un tiempo de respuesta aceptable.

Asimismo, en prácticamente todos los sistemas actualmente desarrollados, se ha obviado una fuerte estrategia de trazabilidad y auditoría de la información manipulada, como consecuencia de ello no se lleva un control de los accesos así como información modificada o eliminada, implicando que en un momento determinado no se pueda realizar la auditoría informática de esta información de carácter histórico. A partir de este proceso de auditoría es necesario que queden registrados todos los accesos y peticiones realizadas por los clientes o servicios. Es importante considerar también, que las tablas de las bases de datos para almacenar las trazas tienden a crecer muy rápidamente lo cual podría impactar negativamente en el rendimiento de las aplicaciones, implicando un monitoreo constante de este parámetro desde la aplicación, con el fin de identificar el momento oportuno para realizar acciones de mantenimiento o salvas.

Otra debilidad funcional existente, es que no es posible registrar para cada componente desplegado, los servicios que estos publican, lo que provoca que se asuma cierto nivel de confiabilidad entre los mismos, ya que los servicios pudieran ser consumidos determinados clientes sin la debida autorización, imponiéndose así un riesgo considerable sobre la información gestionada y compartida. Asociado también al no registro de componentes, existe incapacidad para almacenar centralizadamente las direcciones URL donde los mismos poseen la descripción WSDL de sus servicios, de modo que, si en un momento determinado se modifica esta URL, no se afecte el correcto funcionamiento del resto de los componentes que utilizan el mismo, pues bastaría con actualizar centralizadamente el cambio de dirección ocurrido, tampoco se puede realizar una distribución de servicios, de modo que un mismo servicio, pudiese estar hospedado o publicado en varios nodos que lo provean.

Al no atender centralizadamente las peticiones entre los componentes es necesario además, establecer para cualesquiera dos componentes que interactúen, una conexión para el consumo de sus servicios, de modo que si se pudiera establecer un nodo en el cual converjan todas las peticiones, estas podrían ser analizadas desde el punto de vista de disponibilidad, accesibilidad y posibilidad de autorización en función de los privilegios del usuario autenticado. Otro aspecto adverso a considerar al no poseer este nodo de convergencia, es que no se pueden aprovechar las posibilidades asociadas a los procesos de sindicación de servicios y la implementación de algoritmos mucho más eficientes para

atender todas las peticiones externas realizadas a un componente determinado, pudiéndose de este modo inclusive poseer direcciones alternativas donde se encuentren hospedados los mismos servicios, rompiendo con el temor a que en un momento determinado un servicio no esté disponible y colapse así, el sistema que lo requiera.

La utilización del modelo arquitectónico orientado a servicios y basado en componentes, presupone como otra posible debilidad la falta de integridad en la información que haya sido compartida o utilizada con anterioridad entre los componentes, mediante el consumo de servicios expuestos o publicados. Por ejemplo, si en un momento determinado se referencia cierta información obtenida de un componente proveedor en la base de datos de un componente consumidor y esta se intenta eliminar o modificar en el primero de estos, debe existir algún mecanismo que analice la posibilidad o no de ejecución de la operación involucrada en dependencia de su referencia o no. Entiéndase esto, como que en cualquier sistema gestor de base de datos entre los requerimientos deseados se encuentra, el mantenimiento de la integridad referencial, si se extiende este concepto el problema consiste en el mantenimiento de la integridad referencial en la información que se encuentra distribuida y referenciada en diferentes componentes que se encuentran interoperando entre si.

A continuación se presenta un ejemplo, para facilitar la correcta comprensión de la problemática existente. Suponiendo que el *Registro de Ciudadanos* es un componente independiente donde se encuentran almacenados los datos sociodemográficos de todas las personas y este expone el servicio *BuscarCiudadano*, el cual dado un conjunto de parámetros o argumentos de búsqueda, devuelve los ciudadanos que cumplen con los mismos. Considerando además, que a partir de esta búsqueda se inicia el flujo de actividades de los procesos de negocio que se llevan a cabo en los componentes *componente1* y *componente2*.

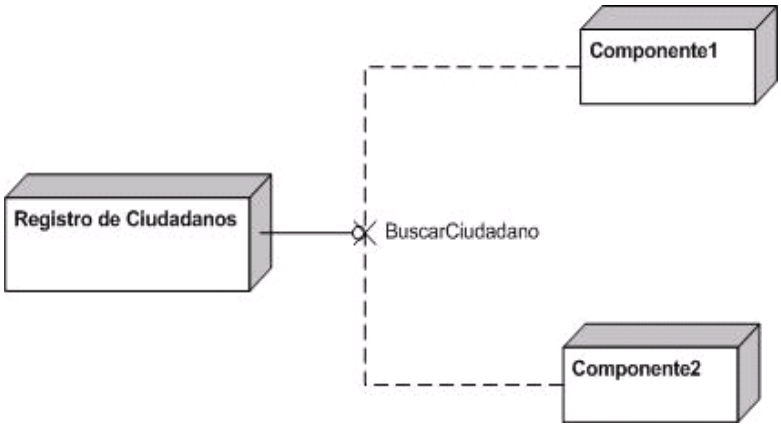


Figura 1.3: Ejemplo de interacción entre componentes distribuidos

Entonces, si se deseara eliminar a la persona cuyo identificador es *83071268985* y este posee información referenciada en cualquiera de las bases de datos de los otros dos componentes, es necesario incluir algún mecanismo o regla de negocio, que permita notificar al componente que referencia los datos que se intentan eliminar, para que el mismo ejecute ciertas funcionalidades que indiquen si es posible o no llevar a cabo la acción solicitada, es decir, es necesario configurar las restricciones de integridad que se aplicarán sobre la información que se comparte entre estos componentes.

Algunas experiencias para la implementación del mantenimiento de la integridad referencial del flujo de información basada en servicios Web XML, lo constituye la utilización de un conjunto de tablas denominadas caché en las bases de datos de los componentes, contenedoras de información no relacionada con su negocio cuyo uso es frecuente, las que tienen que ser objeto de constantes actualizaciones en forma manual. Mediante esta solución parcial y poco elegante no es posible configurar las restricciones de integridad que se establecen entre los componentes y servicios expuestos, así como especificar el tipo de dependencia que se establece entre estos ya sea restrictiva o en cascada.

1.4 Sistemas automatizados existentes asociados al campo de acción

1.4.1 TrustedX Web Services

TrustedX Web Services es una plataforma de servicios Web, de la empresa española Safelayer Secure Communications S.A., fabricante de soluciones de software de seguridad para gestión de identidad digital, firma electrónica y protección de datos, cuyo precio aproximado se encuentra sobre los 300 000 euros. Esta plataforma, cuya primera versión fue lanzada en el 2004 se encuentra ya en la versión 3.0 y aporta mecanismos de seguridad y confianza en arquitecturas SOA. TrustedX ofrece una solución para integrar funciones de seguridad en arquitectura SOA y XML. De este modo, se identifica con la práctica predominante en los sistemas de información corporativos y cierra una etapa de predominio de arquitecturas de software poco flexibles. *[Safelayer, 2009]*

Las funciones de TrustedX permiten gestionar la seguridad y la confianza de una forma estándar y orientada a servicios, por ejemplo: *[Safelayer, 2009]*

- A través de los servicios de autenticación y autorización, las aplicaciones corporativas y los dominios de seguridad externos intercambian información de autenticación y autorización. De este modo, se dispone de un control de acceso único a través de Single Sing - On mediante los estándares definidos por OASIS.
- Gracias al servicio de validación de certificados digitales se reconocen múltiples prestadores de servicios de certificación y se homogeniza la información asociada a los certificados. También se soportan mecanismos estándares de validación de certificados y cualquier otro mecanismo personalizado.
- El servicio de firma electrónica soporta la mayoría de formatos de firma de documentos electrónicos, correo electrónico y mensajería Web. En concreto, los formatos soportados incluyen firmas múltiples, firmas con sello de tiempo y firmas longevas (para validar la firma una vez transcurrido el período de vigencia de los certificados digitales).
- Mediante la pasarela de integración se definen y conectan sucesivas transformaciones de datos XML (en interacción con los diferentes servicios de la plataforma). De este modo, la plataforma actúa como pasarela de confianza (entre procesos, aplicaciones o redes), e integra aplicaciones con cero intrusiones.
- El servicio de auditoría y registro centraliza de manera uniforme y segura la información de trazabilidad (generada tanto por los componentes de servicio de la plataforma, como por el uso o consumo de los mismos).

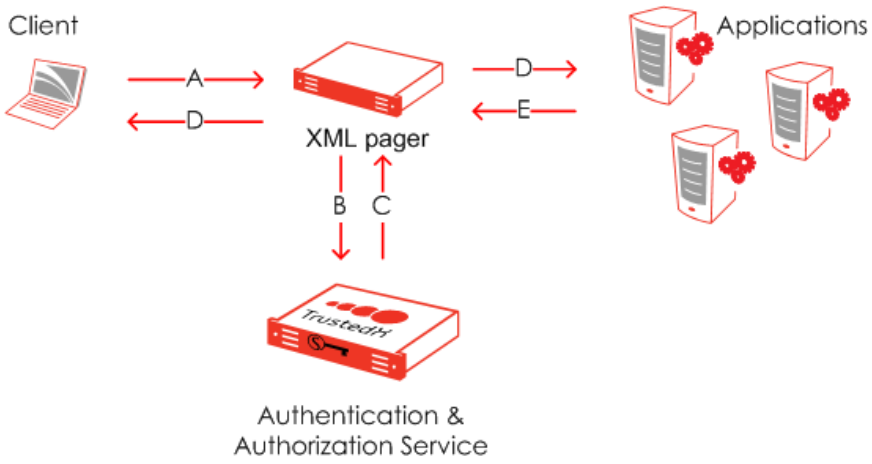


Figura 1.4: Vista general del funcionamiento de TrustedX Web Services

1.4.2 Zain

Zain es una plataforma de servicios de confianza que incluye un conjunto de estos, para lograr mecanismos de seguridad globales y estandarizados (autenticación, autorización, firma electrónica y protección de datos) como servicios Web. Desarrollado por la Izenpe S.A, Empresa de certificación y servicios**,** sociedad anónima constituida en 2002. Esta plataforma, soluciona la complejidad de dotar de los mecanismos de seguridad a las aplicaciones, mediante el uso de las clásicas herramientas de integración, su precio se encuentra alrededor de los 800 000 euros. La SOA de la plataforma y su completo sistema de gestión de la información, simplifican la integración de los mecanismos de confianza en los procesos de negocio, independizándolos unos de otros, y ofreciendo la capacidad de gestión de las políticas de seguridad y auditoría de forma centralizada. Su completa modularidad, garantiza, además el crecimiento futuro tanto en funcionalidad como en prestaciones, caracterizándose por su escalabilidad y su capacidad para adecuarse a los requerimientos más exigentes de alta disponibilidad. Los servicios fundamentales que ofrece Zain son los siguientes: *[Izenpe, 2010]*

Autenticación y autorización: Intercambio de información de autenticación y autorización entre las aplicaciones y dominios de seguridad externos, haciendo posible el control de acceso único a través de SSO, mediante los estándares definidos por OASIS. Se permite la autenticación, autorización y control del acceso de las entidades registradas haciendo posible el control de acceso único y federación en toda la plataforma (entre usuarios, servicios Web y aplicaciones).

Validación de certificados digitales: Reconocimiento de múltiples prestadores de servicios de certificación, homogenizando la información asociada a los certificados. Soporta los mecanismos de validación de certificados estándares y admite la integración de cualquier otro mecanismo personalizado.

Cifrado de datos: Protección de la información mediante mecanismos de cifrado; ya sea documentos electrónicos, correo electrónico o mensajería Web. En futuras versiones se incluirá el servicio de custodia de las claves de cifrado, controlando el acceso a los datos para los grupos de personas o sistemas de confianza.

Pasarela de integración: Permite definir y conectar entre sí un conjunto de transformaciones sucesivas de datos XML, en interacción con los diferentes servicios de la plataforma, actuando de pasarela de confianza entre los procesos y aplicaciones o redes.

Auditoría y registro: Servicio que gestiona de forma centralizada, uniforme y segura toda la información de traza generada por todos los componentes de servicio de la plataforma, así como la información de uso o consumo de los mismos.

Estas soluciones garantizan la seguridad a nivel de los mensajes XML intercambiados de una forma satisfactoria, e inclusive cumpliendo con los estándares internacionales asociados con la seguridad para servicios Web. Además, implementan eficientes procesos de autenticación, autorización y auditoría mediante la publicación de servicios, los cuales pueden ser utilizados por las aplicaciones que se integren con las plataformas descritas.

A pesar de que estas, pueden tomarse como referencia para la implementación de la propuesta de solución que ocupa la presente investigación, tienen algunos inconvenientes, por ejemplo que las mismas poseen elevados precios para su adquisición, no cumplen con todos los requerimientos deseados para el registro de los componentes y sus servicios de manera centralizada, configurar las restricciones de integridad entre los mismos y proveer a los administradores de las aplicaciones, las interfaces visuales necesarias para realizar configuraciones pertinentes para el correcto funcionamiento de las mismas.

1.4.3 Acaxia

Acaxia, es un sistema integral de seguridad, desarrollado por el Centro de Gestión de Entidades (CEIGE) de la UCI, que provee de una administración centralizada a todos los sistemas que utilizan sus servicios, disminuyendo así el tiempo de desarrollo de las aplicaciones, el costo total de los proyectos y los riesgos de seguridad. Para ello, se controla la información de cada uno de estos, los cuales serán provistos de seguridad y auditados en un momento determinado.

De cada uno de estos sistemas, se controlan las funcionalidades que brindan y de cada una de ellas las acciones que se realizan. Una vez registrada la estructura de los sistemas, se procede a la creación de roles, donde se le asignarán los permisos a los diferentes niveles de la misma, estos roles serán asignados a los usuarios dentro de una o varias entidades, de esta forma, si la seguridad se maneja de forma centralizada, el usuario podrá acceder a dichas entidades desde cualquier punto.

El control de acceso a los servicios Web que consumen o brindan los sistemas, es otra de las funcionalidades que brinda Acaxia para asegurar la seguridad en la comunicación entre componentes. Un aspecto importante que incorpora el sistema, es la administración de conexiones donde se restringe el acceso de los sistemas y usuarios a los servidores, bases de datos, esquemas y otras estructuras de datos, con esto, no se tendrá que almacenar las configuraciones de conexión en ficheros físicos que pueden ser objetos de ataques. La auditoría es un punto muy importante dentro de la seguridad informática, para prevenir e identificar violaciones; para esto, el sistema cuenta con un componente capaz de registrar todas las acciones realizadas en un sistema.

Los tres tipos de integración con otros sistemas, para reutilizar todas las ventajas que brinda Acaxia son:

1. Integración a nivel de interfaz.

2. Integración a nivel de servicios Web.
3. Integración a nivel de servicios internos.

A pesar de ser una solución bien interesante, asociada al campo de acción de la investigación, se puede decir que no permite un control centralizado de las peticiones, mediante una especie de *proxy* para analizar la disponibilidad previa del servicio, asimismo, no permite el proceso para control de las condiciones restrictivas de integridad que se pueden establecer entre los servicios. Del mismo modo, se puede decir que el control de acceso a nivel de bases de datos, es una condición muy compleja para manipular, asociado a la propia forma en que se desarrollan los sistemas distribuidos. Se considera que esto sólo es posible, cuando se tiene el control de todas las bases de datos de las aplicaciones registradas, lo que en la práctica, no sucede en los sistemas que interoperan entre si a nivel de servicios, pues los desarrolladores se abstraen de la ubicación física donde se encuentran desplegados los componentes.

1.5 Patrones de arquitectura y diseño utilizados

Un patrón, en sentido general, define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones. Estos pretenden: proporcionar catálogos de elementos reusables en el diseño de sistemas de software y evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Existen varios tipos de patrones, dependiendo del contexto particular en el cual se aplican o de la etapa en el proceso de desarrollo. Algunos de estos tipos son: patrones de diseño, de arquitectura, para ambientes distribuidos, entre otros. *[Larman, 1999]*

En el caso que ocupa a la presente investigación y por su utilización en la propuesta de solución, se analizan los patrones de diseño *Modelo Vista Controlador (MVC)*, el patrón estructural *Proxy* y el patrón *Observer* u *Observador/Observado* como también se le conoce.

1.5.1 Modelo Vista Controlador (MVC)

Este patrón de arquitectura de software, permite separar los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos, esto proporciona múltiples vistas sobre un mismo modelo de datos. El patrón MVC se usa frecuentemente en aplicaciones Web donde se utilicen diferentes interfaces de usuario y el código que provee los datos a la página es dinámico. Soporte de vistas múltiples: dado que la vista se encuentra separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Los tres elementos esenciales de este patrón son los siguientes: *[Kicillof, 2004]*

Modelo: Administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado usualmente formulados desde la vista, respondiendo a instrucciones de cambio para modificar el estado de estos datos, habitualmente desde el controlador.

Vista: Esta presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.

Controlador: Este responde a eventos, usualmente, acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

1.5.2 Patrón estructural proxy

Este patrón proporciona una interfaz intermedia, sustituto o representante para controlar los accesos a un objeto o componente determinado. Su funcionamiento consiste en que un componente proxy recibe las peticiones que se realizan a otros objetos, de modo que el proxy provee una interfaz a través de la cual se puede consumir el servicio del componente que realmente lo provee. En sentido general se puede decir, que el patrón obliga a que las llamadas a métodos o el consumo de servicios ocurran indirectamente a través de un componente u objeto proxy.

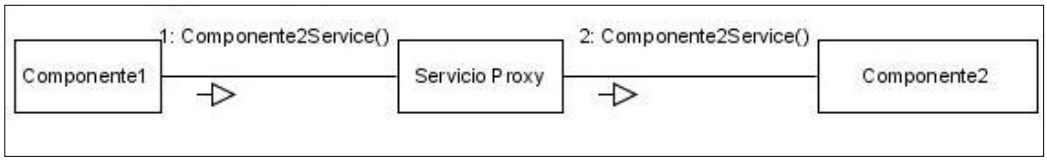


Figura 1.5: Descripción gráfica del funcionamiento del patrón de diseño estructural Proxy

Asimismo, un proxy puede ocultar el hecho de que un objeto o componente, reside en un espacio de direcciones diferente, permitiendo como consecuencia una posible distribución de servicios e inclusive un particionamiento horizontal o vertical de los servicios que se encuentran involucrados en la solución de un problema determinado. Asimismo, es posible implementar de forma más eficiente el mecanismo en el que el proxy accede o redirecciona las peticiones hacia los servicios de los componentes, analizando para ello variables como por ejemplo la cantidad de accesos por unidad de tiempo y el número de proveedores disponibles que poseen el servicio Web involucrado en la petición, entre otras; de modo que se pueda mejorar otra de las debilidades de las arquitecturas SOA que es la disponibilidad y el bajo acoplamiento que debe existir entre los servicios encapsulados en los diferentes componentes coexistentes.

1.5.3 Patrón observer u Observador/Observado

Este patrón es muy utilizado en la implementación de una arquitectura MVC. El modelo es un subtipo de *Observado* y la vista un subtipo de *Observador*. Normalmente se implementan como dos clases que manejan adecuadamente la función de notificación de cambios que necesita MVC, pues proporciona el mecanismo por el cual, las vistas pueden ser notificadas automáticamente de los cambios producidos en el modelo. Permite gestionar la relación entre un componente observado y sus observadores, pudiendo un componente determinado comportarse de ambas formas, es decir, cada componente observado posee una referencia a todos sus observadores, así como los servicios que están siendo consumidos por estos. El patrón de diseño *Observer* es aplicable cuando: *[Kicillof, 2004]*

- Existe una relación fuerte entre datos y vistas, de manera que conviene separar el control de los datos de su representación final.
- Un cambio en el estado de un determinado componente afecta a muchos otros.
- Se necesita notificar a otros componentes sin hacer presunciones sobre la identidad y ubicación física, evitando un fuerte acoplamiento lo cual ayuda a diseñar sistemas con diferentes capas de abstracción claramente separadas.

- Se necesitan sincronizar las acciones de varios componentes sobre un estado común y la coherencia de ellos.

Este patrón es utilizado en la solución propuesta para el control de las restricciones de integridad que se establecen entre los servicios que proveen los componentes, de modo que haya un proceso de notificación entre los mismos, cuando se desee modificar o eliminar información que se encuentre referenciada en uno o varios componentes observadores.

Conclusiones

En este capítulo se han expuesto los fundamentos teóricos conceptuales que constituyen la base para la correcta concepción, diseño e implementación de la solución propuesta. Como se pudo observar existen numerosos estándares, que describen los requerimientos que deben cumplir las aplicaciones que utilicen los servicios Web XML como medio de interoperabilidad, lo cual constituye un marco de referencia importante a considerar. Se describe además, el modelo de Autenticación, Autorización y Auditoría, que guía los desarrolladores en qué elementos deben considerar para aquellos sistemas que necesiten este tipo de requerimientos de seguridad.

Asimismo, en la construcción de la solución se utiliza una implementación del patrón "*proxy"* para centralizar las peticiones que se realizan a los servicios publicados por los componentes y el patrón "*observer"* para el control de las restricciones de integridad que se establecerán entre estos.

CAPÍTULO II: PROPUESTA DE PROCESOS DE AUTENTICACIÓN, AUTORIZACIÓN Y AUDITORÍA PARA APLICACIONES BASADAS EN SERVICIOS WEB XML

En este capítulo se realiza una descripción de la propuesta de solución considerando para ello, en primer lugar, la descripción de un modelo conceptual que contiene las principales definiciones que se utilizan en la propuesta, así como la relación que se establecen entre ellas. Asimismo se especifican los requerimientos funcionales que forman parte de la solución. Se realiza una descripción de su arquitectura y finalmente, se describen en términos de implementación, los principales algoritmos, servicios Web y patrones de diseño utilizados, a partir del problema identificado en el diseño de la investigación.

2.1 Descripción de la propuesta de solución

2.1.1 Modelo conceptual asociado a la propuesta de solución

Un Modelo Conceptual o Modelo de Dominio, constituye una representación visual para el usuario de los conceptos u objetos significativos del mundo real para un problema o área de interés. Representa conceptos del mundo real, no de los componentes de software, mediante clases conceptuales del dominio del problema, encargándose de capturar los tipos más importantes de objetos y eventos que suceden en el entorno. *[Pressman, 2005]*

La propuesta de solución cuenta con dos actores fundamentales: un *Administrador General*, el cual, en dependencia de su nivel de acceso, tiene los permisos necesarios para gestionar la información para usuarios que en la estructura jerárquica de niveles de seguridad que defina un organismo (dominio) determinado. Esta gestión incluye crear usuarios, modificar sus datos y privilegios, eliminarlos y realizar búsquedas a partir de diferentes parámetros. También es posible realizar una auditoría estricta a través de las trazas almacenadas por el sistema, conociendo qué usuario ha participado en cada transacción. Por otro lado, existe un *Administrador de Configuración,* que es el único encargado de la gestión de organismos (dominios), niveles de seguridad, ubicaciones, componentes, servicios y roles. Para cada uno de los usuarios creados se le define además un *Nivel de actividad,* que puede ser activo o pasivo, en diferentes *Periodos de actividad,* los que representan diferentes rangos de fechas que indican la posibilidad de acceder de los usuarios.

Todo *Usuario* que se autentique en el sistema, recibirá un *Certificado* (token) generado en forma aleatoria, de 32 caracteres, que lo identifica de forma unívoca en un periodo de tiempo previamente configurado. Mediante este certificado, el *Usuario* obtiene los datos y privilegios necesarios que serán verificados posteriormente mediante la autorización.

Un *Componente* debe entenderse como una unidad de software ejecutable que representa el núcleo de una aplicación, la cual puede ser implementada en forma independiente y ser a la vez, sujeto a la composición con otras partes. Es decir, se puede tomar el componente e integrarlo a otros componentes, mediante el consumo de determinados servicios que estos publiquen a través de interfaces WSDL bien definidas. A cada uno de estos componentes se le debe registrar las

funcionalidades que posee, los servicios Web que publican y los roles que tienen acceso a él, describiendo para cada uno, qué funcionalidades pueden ser utilizadas.

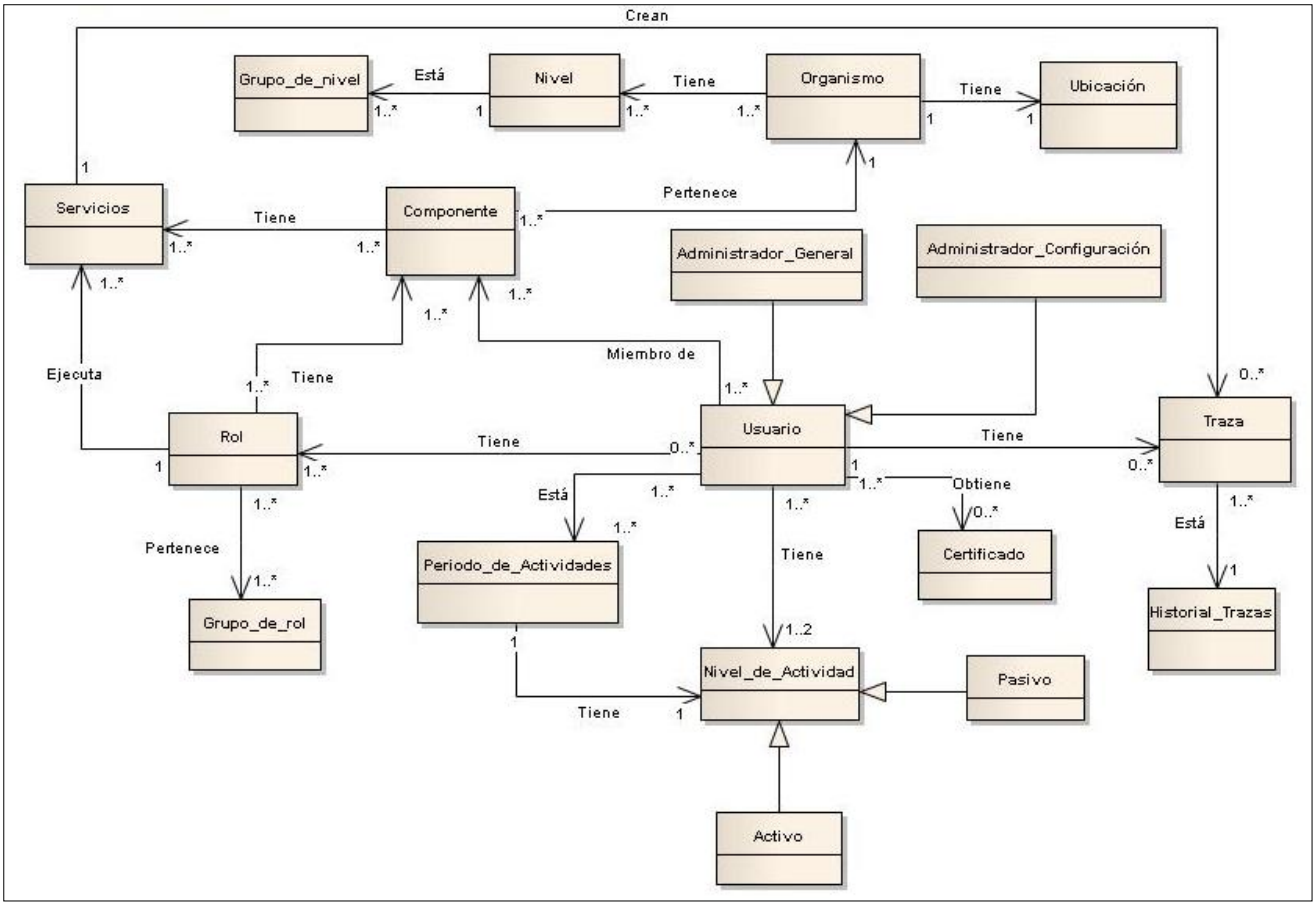


Figura 2.1: Modelo de dominio de la propuesta de solución

2.1.2 Especificación de los requerimientos de software

El IEEE Standard Glossary of Software Engineering Terminology, define los requerimientos de software como condiciones o capacidades que deben estar presentes en un sistema o componentes de este, para satisfacer un contrato, estándar, especificación u otro documento formal. Es necesario hacer énfasis en la precisión con que se debe realizar esta tarea por cumplir un papel primordial en el proceso de producción de software, pues se enfoca en un área fundamental: la definición de lo que se desea producir, permitiendo describir con mayor claridad el comportamiento del sistema, minimizando los problemas derivados de su desarrollo. A continuación se describen los principales requerimientos funcionales que posee la solución, agrupados por paquetes para a su correcta comprensión. La totalidad de los requerimientos se encuentran especificados en el **Anexo 5**.

LISTADO DE PRINCIPALES REQUERIMIENTOS FUNCIONALES	
Paquete Autenticar:	*Paquete Traza:*
RF1: Autenticar	RF5: Registrar traza
RF2: Autorizar	RF25: Buscar traza
RF3: Crear certificado digital	RF26: Buscar traza histórica

RF4: Crear lista de privilegios	RF27: Eliminar fichero de traza RF28: Crear fichero de traza RF52: Verificar cantidad de trazas
Paquete Usuario: RF11: Adicionar usuario RF14: Buscar usuario RF15: Buscar usuario histórico	*Paquete Consumir servicio:* RF51: Consumir servicio RF54: Verificar disponibilidad de servicio RF55: Verificar WSDL opcional RF58: Verificar componente RF59: Verificar petición RF65: Obtener petición RF66: Verificar servicio
Paquete Componente: RF6: Adicionar componente RF10: Asignar componente	*Paquete Servicio:* RF41: Adicionar servicio RF63: Asignar servicio

Tabla 2.1: Listado de principales requerimientos funcionales

Para la solución se han identificado cinco actores fundamentales en función de los requerimientos funcionales especificados en el **Anexo 5.** El primero de ellos, el *Administrador General* es quien podrá registrar nuevos usuarios al SAAA, a partir de este registro podrá configurar reportes personalizados de las trazas existentes en función de parámetros seleccionados previamente por este. A su vez el *Administrador de Configuración,* es quien interactuando con el SAAA puede crear países, organismos, ubicaciones, niveles de seguridad, roles y componentes. El registro de un nuevo componente implica la adición de las funcionalidades que este posee, los servicios Web que publica, los roles que tienen acceso al componente, así como la relación de cuáles roles tienen acceso a cada uno de los servicios y funcionalidades publicados. Asimismo, cuando se registra un nuevo componente es necesario configurar las restricciones de integridad que se establecen entre este, con los que él está observando.

El actor *Componente* representa el sistema externo que utilizará los requerimientos de seguridad que provee el SAAA así como que el actor *Proxy,* es un componente de servicios que actuará como intermediario entre las peticiones realizadas entre un *Componente* y otro, estando estos previamente registrados en el SAAA. El *Proxy* es quien se encarga de recibir la petición realizada por el componente, verificando que la misma sea válida en función del componente y el servicio al cual va dirigida. Asimismo controla que el certificado digital recibido, asociado a un usuario determinado, tenga los privilegios necesarios para llevar a cabo la acción solicitada, antes de redireccionar la misma, analiza también las restricciones de integridad que existen sobre la petición solicitada.

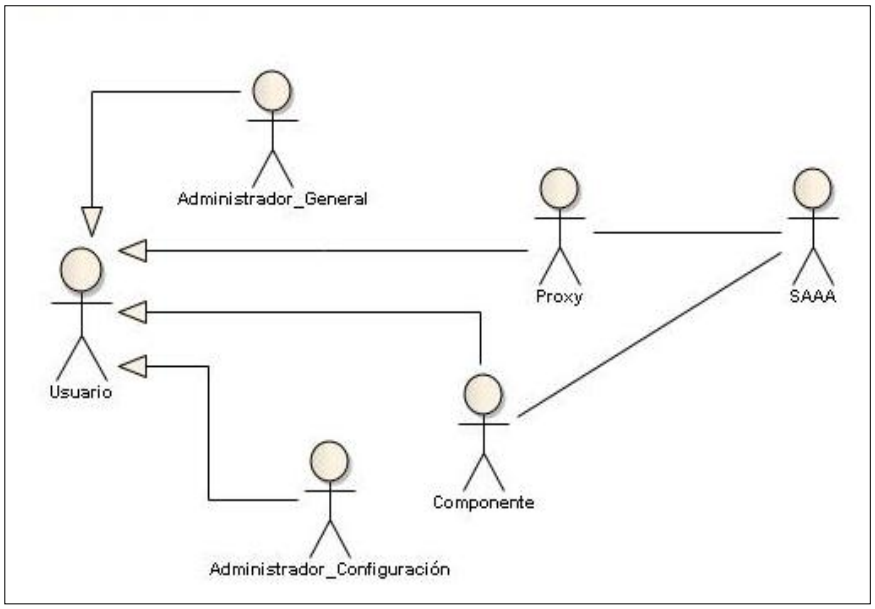


Figura 2.2: Vista global de actores

2.2 Descripción de la arquitectura del sistema

Para el desarrollo de la solución se utilizó una arquitectura en tres capas, basada en el patrón Modelo Vista Controlador (MVC). La capa de datos fue implementada utilizando PostgreSQL 8.2, mediante lo cual se pueden crear las crear funciones y vistas necesarias. La capa de lógica del negocio se implementó utilizando el lenguaje de script PHP 5, considerando la librería SOAP que este posee para el desarrollo de aplicaciones basadas en servicios Web XML.

Para el desarrollo de la capa de presentación se utilizaron XHTML, Smarty, Ajax y YUI. Smarty, se consideró como motor de plantillas que se encarga de separar el código PHP, como la lógica de negocios; del código XHTML como lógica de presentación y genera contenidos Web mediante la colocación de etiquetas Smarty en un documento XHTML. La interfaz de usuario fue desarrollada utilizando Yahoo User Interface (YUI), lo que constituye un conjunto de librerías escritas en JavaScript para la construcción de aplicaciones. Estas librerías son utilizadas para el desarrollo Web, específicamente, en la programación de aplicaciones Web, con componentes vistosos y personalizables y con una amplia incorporación de AJAX. Provee una gran cantidad de utilidades en JavaScript y controles de interfaz de usuario, lo que permite la homogeneidad de las interfaces visuales de los productos de software. Como servidor Web se utiliza el Apache.

En sentido general se puede plantear que todas las tecnologías seleccionadas para integrarlas en la solución, están basadas en software libre y de código abierto, así como que forman parte de la estrategia general para la informatización del SNS cubano, donde se describe que las aplicaciones deben cumplir con este lineamiento, estar desarrolladas mediante el lenguaje PHP y que deben interoperar mediante la publicación y consumo de servicios Web XML. *[Delgado, 2009]*

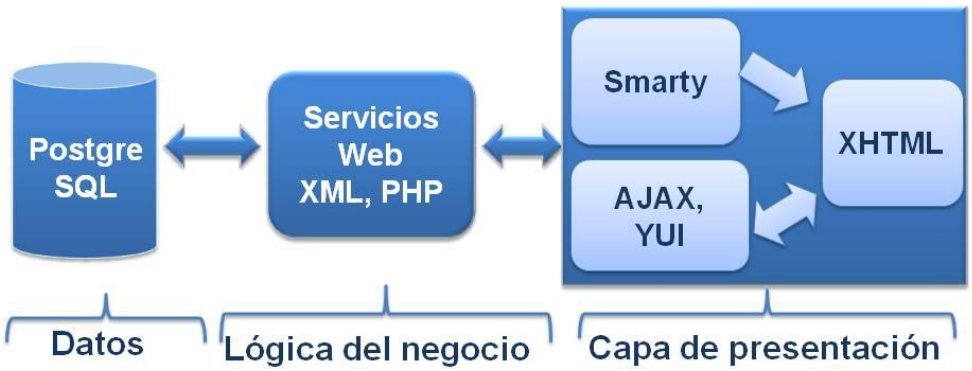


Figura 2.3: Descripción de la arquitectura del sistema

2.3 Implementación de la propuesta de solución

La implementación es el centro de las iteraciones durante la fase de construcción, aunque también se lleva a cabo la implementación durante la fase de elaboración. El Modelo de Implementación describe cómo los elementos del Modelo de Diseño son implementados en términos de componentes, donde se detalla además, su organización de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y el lenguaje o lenguajes de programación utilizados, así como la dependencia que se establece entre estos componentes. *[Pressman, 2005]*

A continuación se describe en términos de implementación, los procesos de creación de componentes y usuarios, considerándose que son los más importantes desde el punto de vista del funcionamiento del SAAA. Seguidamente se exponen los elementos referidos con la implementación de los patrones de diseño *proxy* y *observer*, para el control centralizado de las peticiones realizadas entre los componentes y el mantenimiento de la integridad referencial en el flujo de información distribuida respectivamente. Finalmente, se describen los servicios Web implementados y que publica el SAAA.

2.3.1 Descripción de los procesos para la adición de componentes y usuarios

2.3.1.1 Proceso *Adicionar componente*

El proceso para adicionar un determinado componente que se muestra en la **Figura 2.4**, se inicia seleccionando el país al que este pertenece, en caso de que el mismo no exista, se procede a crear el país asociado al componente, en caso contrario, se selecciona su organismo o dominio, por ejemplo MINSAP, el cual ya tiene previamente asociado los niveles de seguridad que este posee y las ubicaciones físicas asociadas con sus niveles. Para la creación de las ubicaciones físicas, se construye un árbol en la base de datos donde cada vez que se registra una nueva ubicación se le debe indicar cuál es su padre, de modo que aquellas ubicaciones que no tengan hijas, serán consideradas las hojas en el árbol de jerarquía.

A este organismo adicionalmente se le puede configurar un directorio LDAP (Lightweight Directory Access Protocol), lo cual permitirá entre otros elementos una autenticación única o Single Sign - On (SSO) para todas las aplicaciones, así como contar perfiles de usuarios y autenticación/autorización centralizados *[LDAP, 2004]*. La utilización alternativa de LDAP es aprovechada posteriormente para agilizar el registro de usuarios al sistema, debido a que todos sus datos generales ya se encontrarán previamente registrados en el LDAP, teniendo que introducirse sólo aquella información de configuración adicional.

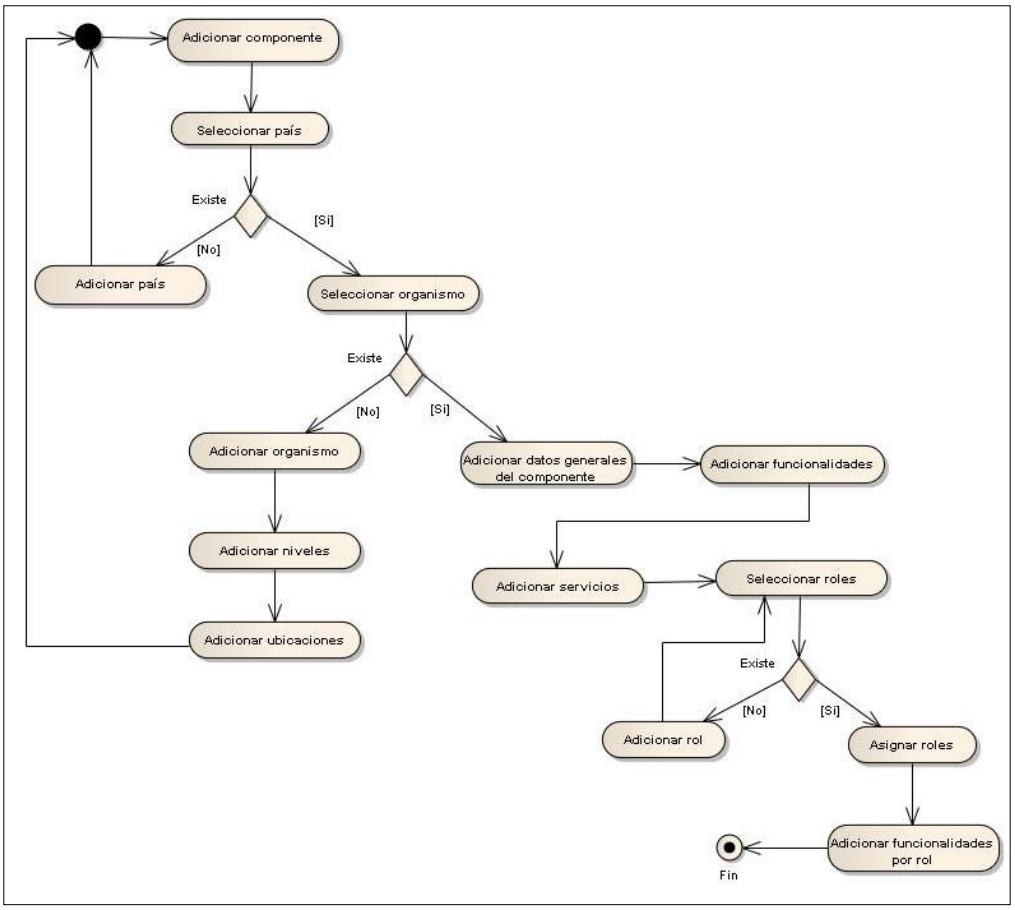


Figura 2.4: Diagrama de actividades del proceso *Adicionar componente*

Una vez seleccionado el país y el organismo respectivamente, se adicionan los datos generales del componente siendo estos su nombre, dirección URL donde se encuentra el WSDL y la dirección URL de la página principal del componente, en caso de que este posea interfaz de usuario.

Seguidamente, se registran todas sus funcionalidades, a partir de su nombre y una breve descripción, como algunas de ellas pueden ser publicadas como servicio, se registra en estos casos el nombre del servicio Web XML y hasta tres direcciones URL donde se encuentra el WSDL que describe al servicio, para ser utilizadas alternativamente en caso de que el servicio involucrado en una petición no se encuentre disponible. Cuando se adiciona un nuevo componente es necesario registrar además, los componentes para los cuales, el que esta siendo registrado, se comporta como observador porque tiene información referenciada, la cual obtiene mediante el consumo de un servicio publicado por el componente observado, se debe registrar además el tipo de restricción, ya sea esta restrictiva o en cascada; este proceso será utilizado en la implementación del patrón *observer.*

Seguidamente, se registran los roles que tienen acceso al componente. Primero se hace una búsqueda en un codificador de roles existente y si los deseados se encuentran en este se le asignan al componente, en caso contrario, es necesario registrar los nuevos roles para posteriormente asignarlos. Finalmente, por cada rol se seleccionan las funcionalidades y servicios a los que tiene privilegios o derechos de ejecución, esta información será indispensable para la autorización, una vez que un usuario determinado desee consumir un servicio Web registrado en el SAAA. En el **Anexo 6** se muestra la interfaz de usuario mediante la cual es posible adicionar un nuevo componente.

2.3.1.2 Proceso *Adicionar usuario*. Jerarquía de usuarios

La creación de usuarios en el sistema SAAA, comienza con el registro de los datos generales del usuario tales como nombre, primer apellido, correo electrónico, nombre de usuario y contraseña. Posteriormente, se registra para cada uno de los organismos los componentes a los que tiene acceso el usuario que se está creando, especificando para cada componente los roles que podrá jugar el usuario, en qué nivel jerárquico lo puede ejercer, así como dentro de cada nivel, qué servicios pueden ser consumidos por este. Del mismo modo, se puede seleccionar para cada componente si el usuario es administrador o no, pues este rol, es el que tendrá privilegios de crear nuevos usuarios en el SAAA.

Asimismo, por cada componente se le debe especificar un *Nivel de actividad* el cual puede ser *Activo* o *Pasivo*, en un *Periodo de actividad* determinado, el cual requiere una fecha de inicio y una fecha de fin, de modo que, si esta última no se especifica se asume que posee permisos ilimitados en el tiempo. Así es como un usuario puede poseer diferentes *Niveles de actividad*, en diferentes *Periodos de actividad*, lo que garantiza niveles de seguridad mucho más flexibles y personalizables para los usuarios y componentes registrados en el SAAA. La interfaz de usuario que permite adicionar nuevos usuarios se muestra en el **Anexo 7.**

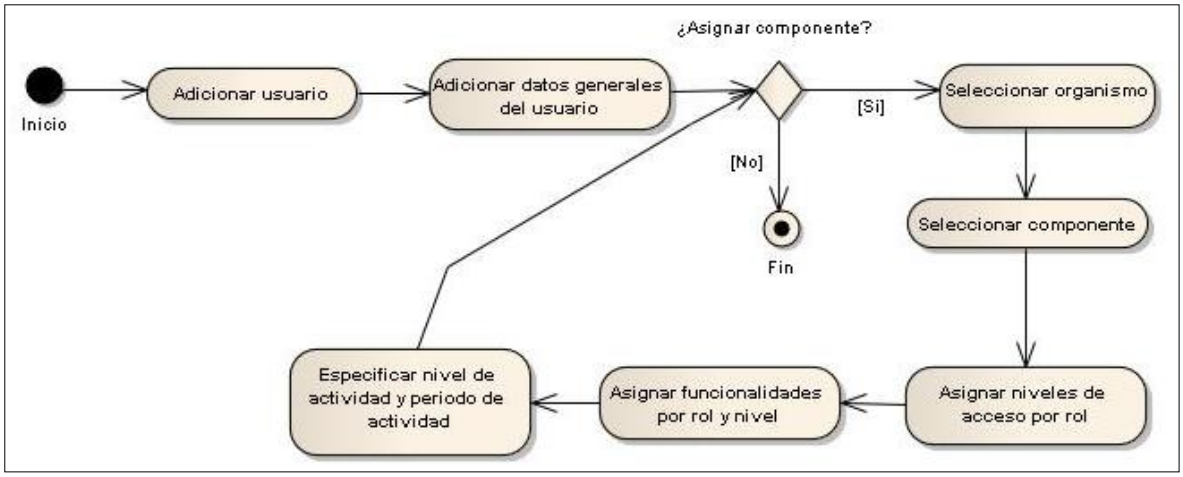


Figura 2.5: Diagrama de actividades del proceso *Adicionar usuario*

La creación de usuarios se ha concebido para que se desarrolle en forma jerárquica. Es decir, si a un usuario determinado se le dan privilegios de administrador de un organismo determinado, tiene la posibilidad de crear nuevos usuarios no administradores en los niveles de seguridad a los que este pertenece y así, crear otros administradores en el nivel inmediato inferior al que este pertenece. En el **Anexo 8** se presenta gráficamente un ejemplo para facilitar la comprensión de lo anteriormente descrito.

Suponiendo que se está trabajando en el organismo MINSAP, el que tiene los niveles de seguridad nacional, provincial, municipal y de unidades de salud y este a su vez, tiene registrado un determinado componente que posee los roles de editor y visualizador, entonces se puede observar cómo los administradores de los diferentes niveles sólo pueden crear editores y visualizadores en su mismo nivel así como los administradores del nivel inmediato inferior. Por ejemplo, el administrador nacional de Cuba puede crear los administradores de cualquiera de las provincias del país y así sucesivamente, hasta el nivel más bajo en la jerarquía, donde no se podrán crear nuevos administradores. De este modo, se adiciona mayor seguridad en el proceso de administración centralizada de los usuarios, pues

los usuarios administradores, a pesar de poseer estos privilegios, se les impondrán restricciones en función de los niveles que posee y de las ubicaciones a las que pertenece.

2.3.2 Definición de servicios Web implementados

Antes de comenzar a describir los servicios implementados, es necesario exponer algunos elementos del Lenguaje de Descripción de Servicios Web (WSDL). En sentido general, se puede decir que son documentos XML asociados a los servicios Web que describen: las funciones que estos ofrecen, cómo se realiza el intercambio de mensajes, especifican el contenido de una petición y el aspecto de la respuesta en una notación inequívoca. Además de describir el contenido de los mensajes, WSDL define dónde está disponible el servicio y qué protocolo de comunicaciones utilizar para comunicarse con el mismo. Esto significa que el archivo WSDL define todo lo necesario para desarrollar una aplicación que interactúe con un servicio Web. *[W3C, 2010]*

El empleo de XML como notación de los archivos WSDL los convierte en documentos neutrales respecto al lenguaje de programación y accesibles desde una amplia variedad de plataformas. Se muestra en el diagrama del **Anexo 4** las etiquetas fundamentales que contiene un documento WSDL. La etiqueta <*definitions*>, constituye el nodo raíz de la jerarquía del documento e incluye la definición del servicio, generalmente, se asocia un documento WSDL a cada servicio. Dentro de esta etiqueta se encuentran las secciones <*types*>, <*message*>, <*portType*>, <*binding*> y <*services*>. *[Ronda, 2004]*

La etiqueta <*types*>, agrupa las definiciones de los tipos de datos propios empleados. Esta sección puede ser omitida si no se definirán nuevos tipos. La sección <*message*>, incluye la descripción de los parámetros de entrada y salida. Los elementos de la sección <*portType*>, definen el conjunto de operaciones que se describen el WSDL. Cada elemento de la sección <*operation*>, describe los parámetros de entrada y salida de la operación. La sección <*binding*>, puede incluir cero, uno o más elementos. Su propósito es establecer el protocolo y reglas de codificación para un <*portType*>. La sección <*service*> también puede incluir desde cero hasta varios elementos <*port*>, que definen el protocolo y dirección de red para garantizar la conexión de un cliente con un servicio. *[Ronda, 2004]*

Asimismo, para estos elementos se han establecido algunas pautas para la descripción de los servicios mediante el WSDL, de modo que sea posible cierta estandarización de los mismos y los usuarios finales del SAAA puedan familiarizarse con ellos, los mismos se describen a continuación:

1. Los nombres de los WSDL deben seguir el siguiente formato: *SAAA+Servicio.wsdl*. Ejemplo: *SAAAAutenticar.wsdl*.
2. El atributo *targetNamespace* debe tener un valor que corresponda con su verdadero espacio de nombres, se debe poner ese mismo valor en el valor del atributo *xmlns:typens*. Se sigue el siguiente formato *SAAA+Servicio.*
3. Como consecuencia de que algún método en PHP se le pase como parámetro un arreglo o devuelva un arreglo, este se representa en WSDL como un tipo de dato complejo. En estos tipos de datos complejos el nombre debe comenzar con la palabra *ArrayResultado* y posteriormente, un nombre que identifique el arreglo. Este nombre debe coincidir con el nombre del método PHP para el cual se está realizando el tipo de dato compuesto. Ejemplo: *ArrayResultadoAutenticar*.

4. Para construir la estructura interna de un arreglo, se escogerá el modelo “sequence” antes de definir los elementos del arreglo y se le pondrá como nombre “*Struct+nombre de la estructura*”, donde el nombre de la estructura coincidirá con el nombre que identifica el arreglo. Cada elemento del arreglo tiene un nombre y un tipo de dato (que tienen que coincidir con el nombre de ese elemento y el tipo de dato esperado en el arreglo de salida del método php). Ejemplo: *StructComponente.*
5. Los WSDL cuentan con un solo puerto. Este puerto debe llamarse *SAAA+Servicio+ServicePort*. Ejemplo: *SAAAAutenticarServicesPort.*
6. Sólo se creará un binding que se llamará *SAAA+Servicio+ServiceBinding* y se enlazará con el puerto creado, además el tipo del binding debe ser SOAP, el estilo RPC y el protocolo de transporte sea HTTP. Ejemplo: *SAAAAutenticarServiceBinding.*
7. Se debe crear un solo servicio que se debe nombrar como *SOAP_ SAAA*. El nombre del puerto del servicio debe ser *SAAA+Servicio+ServicePort* y el binding, debe coincidir con el nombre definido para este, en la sección enlaces (binding) pero con el prefijo “*typens:*”.

La implementación de servicios Web desempeña un papel protagónico en el desarrollo del SAAA, debido a que brinda la posibilidad a componentes o sistemas externos, de utilizar los procesos de Autenticación, Autorización y Auditoría que proporciona el modelo propuesto, independientemente de la plataforma o lenguaje en el que hayan sido desarrollados los mismos. A continuación se describen estos servicios, haciéndose énfasis en los parámetros de entrada y las estructuras de salida que los mismos utilizan.

2.3.2.1 Autenticar

El SAAA brinda este servicio mediante el cual los usuarios de las aplicaciones podrán realizar el proceso de autenticación, obteniendo así la lista de privilegios que posee y un certificado digital que será incluido posteriormente en todas las cabeceras de los mensajes SOAP asociados a las peticiones para el consumo de los servicios registrados.

Nombre del servicio: *Autenticar*	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
usuario	string
contrasena	string
PARÁMETROS DE SALIDA	
Descripción	**Tipo**
idusuario	string
usuario	string
nombre	string
correoelectronico	string
certificado	string
ListaDerechos	array

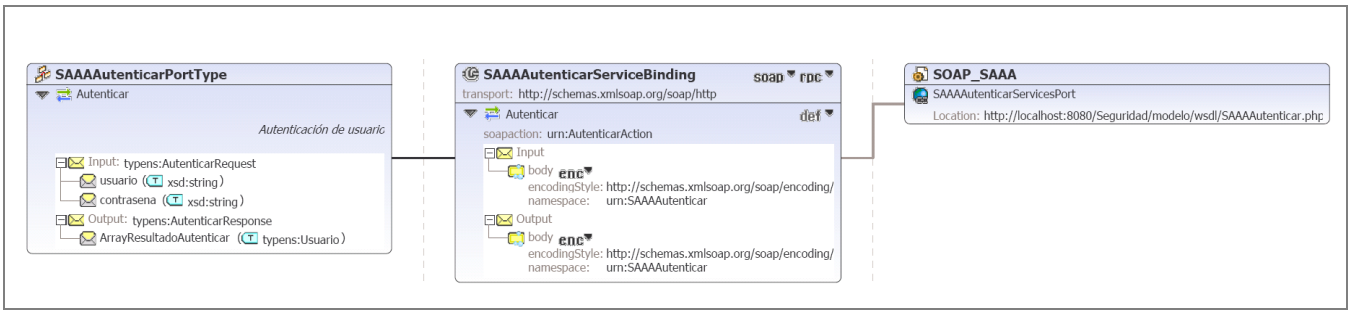


Tabla 2.2: Descripción de los parámetros de entrada y salida del servicio *Autenticar*

En la **Figura 2.6** se muestra el XML schema asociado a la estructura de datos construida para la salida de este servicio, donde a partir de los parámetros de usuario y contraseña, el servicio devuelve un arreglo asociativo que contiene los parámetros simples descritos en la tabla anterior para un arreglo, con la lista de derechos que posee el usuario autenticado. Este contiene por cada *componente* a los que tiene acceso el identificador del componente, el nombre del componente, la dirección URL de su página principal, en caso de tener interfaz de usuario y una lista de los roles que tienen acceso al componente. Esta lista de roles, posee por cada *rol* el identificador del rol, el nombre del rol y una lista de los niveles en los que este rol tiene privilegios de acceso.

Por su parte la lista de niveles describe, por cada nivel, el identificador del nivel, el nombre del nivel y la lista de los *servicios* que pueden ser accedidos en ese *componente*, por ese *rol* y en ese *nivel*. Cada servicio se describe a su vez mediante el identificador del servicio y su nombre.

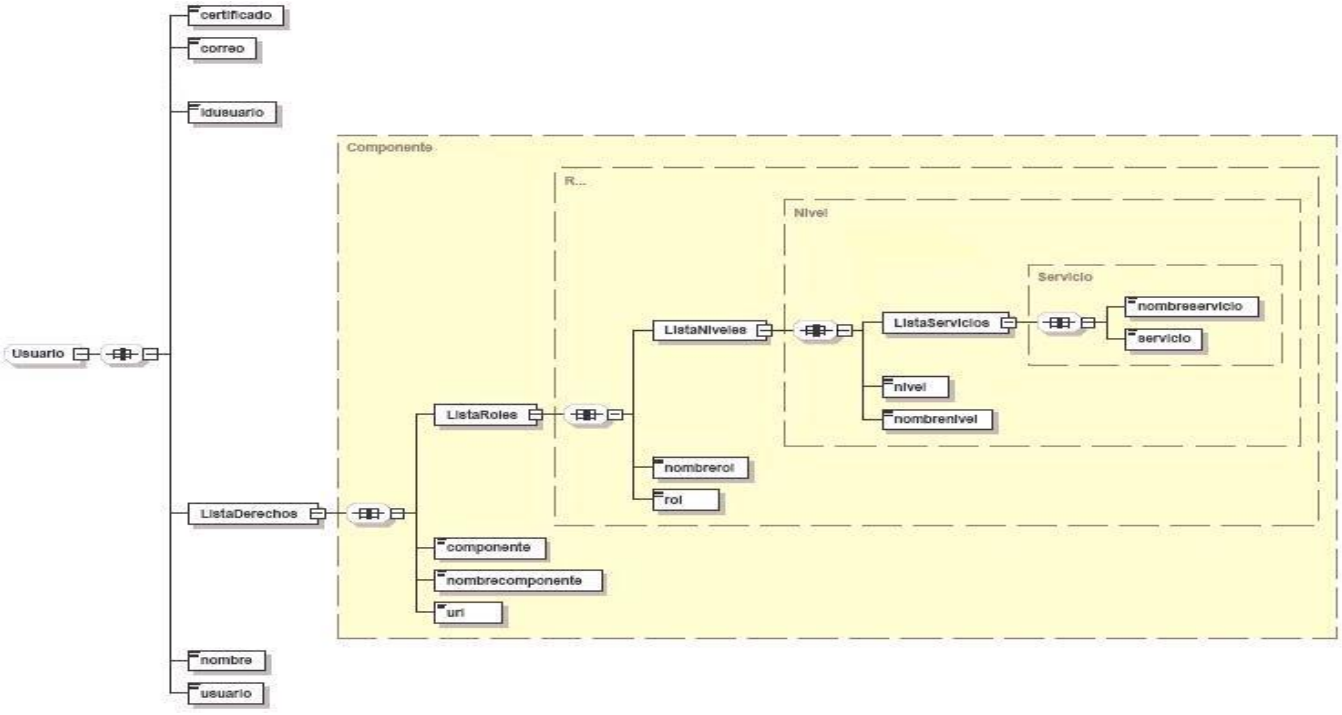


Figura 2.6: XML schema utilizado por el servicio *Autenticar* para la estructura de salida

A continuación se muestra un ejemplo de mensaje SOAP que se construye al utilizar el WSDL *SAAAAutenticar.wsdl* **(Ver Anexo 9)**. En este se puede observar claramente los parámetros de entrada necesarios para el servicio descrito:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <SOAP-ENV:Body>
            <m:Autenticar xmlns:m="urn:SAAAAutenticar" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
                  <usuario xsi:type="xsd:string">Usuario</usuario>
                  <contrasena xsi:type="xsd:string">Contraseña</contrasena>
            </m:Autenticar>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 2.7: Ejemplo de mensaje SOAP utilizando el servicio *Autenticar*

Para la generación del certificado digital se utiliza la siguiente instrucción:

```
$sCertificado = md5(uniqid(mt_rand(200000000), true));
```

La función *int mt_rand ([int $min,] int $max),* es un generador de números aleatorios que utiliza el algoritmo Mersenne Twister (Tornado de Mersenne) *[Motsumoto et al., 2002]*. Se dice que un número es de Mersenne si es una unidad menor que una potencia de 2, es decir, $M_n = 2^n - 1$. La función *mt_rand* utiliza el tornado MT19937 de Mersenne. Posteriormente se utiliza la función *string uniqid ([string $prefijo [,bool $mas_entropia]]),* la cual obtiene un identificador con prefijo único basado en la hora actual en microsegundos, con un *prefijo* vacío, la cadena devuelta tiene una longitud de 13 caracteres. Si *mas_entropia* es *true*, tendrá 23 caracteres. Finalmente, a la cadena que se obtiene de la función *uniqid,* se le aplica el algoritmo *MD5 (Message-Digest Algorithm 5),* obteniendo una cadena de 32 caracteres que representa el certificado generado.

Una vez que se genera el certificado, este es registrado en la tabla de la base de datos *tb_usuario_certificado*. Esta tiene como atributos el identificador del usuario, un identificador del certificado, formado por la concatenación del identificador del usuario con el certificado generado, el propio certificado, así como la fecha y hora de inicio en que este fue generado. Además se registra cuando se cierre la sesión la fecha y hora de fin, la que se mantiene en *null* mientras tanto no ocurra esta acción, lo cual significa que el usuario está activo y no se puede autenticar otro con el mismo conjunto de credenciales, para lo cual, el servicio emitirá el siguiente mensaje de error.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
      <SOAP-ENV:Body>
            <SOAP-ENV:Fault>
                  <faultcode>3</faultcode>
                  <faultstring>El usuario y contraseña especificado está siendo
usado en este momento.</faultstring>
            </SOAP-ENV:Fault>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 2.8: Respuesta del servidor con error de autenticación

Una vez que se realiza el proceso de autenticación y se obtiene el certificado digital, de ahí en lo adelante a las peticiones, se les debe incluir en la cabecera SOAP el certificado obtenido, el cual se

utilizará como medio para verificar los privilegios del usuario durante la autorización de una petición determinada. A continuación se muestra el *elemento* que debe ser incluido en el WSDL de modo que sea requerido el certificado:

```
<element name="Certificado">
   <complexType>
      <attribute name="Certificado" use="required" type="xsd:string"/>
   </complexType>
</element>
```

Figura 2.9: Elemento *Certificado* definido requerido en el mensaje SOAP

```
<message name="Header">
  <wsdl:part name="Certificado" element="typens:Certificado"/>
</message>
```

Figura 2.10: Mensaje *Header* que incluye al elemento *Certificado*

```
<soap:header message="typens:Header" part="Certificado" use="literal"/>
```

Figura 2.11: *SOAP: HEADER* que incluye el mensaje *Header*

2.3.2.2 Autorizar

Este es el servicio que provee el SAAA que permite la autorización o no de ejecución de un servicio o funcionalidad, perteneciente a un componente por un rol en un nivel determinado. Este servicio Web requeire que se incluya en la cabecera del mensaje SOAP de la petición el certificado o token, para verificar si el usuario asociado tiene los privilegios para poder llevar a cabo la petición solicitada.

Nombre del servicio: *Autorizar*	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
sComponente	string
sServicio	string
sRol	string
sNivel	string
listaparametros	array
PARÁMETROS DE SALIDA	
Descripción	**Tipo**
ResultadoAutorizar	bool

Tabla 2.3: Descripción de los parámetros de entrada y salida del servicio *Autorizar*

En la **Figura 2.12** se muestra un mensaje SOAP asociado a una petición, donde se puede observar claramente en la cabecera del mismo, el certificado del usuario al que se le analizará la petición por el servicio de autorización. En el WSDL de este servicio se puede observar cómo se incluyeron los elementos mostrados en las **Figuras 2.9**, **2.10** y **2.11**, necesarios para que en la cabecera del mensaje SOAP se incluya el certificado digital generado. **(Ver Anexo 10)**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:ns1="urn:SAAAAutorizar"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns2="http://xml.apache.org/xml-soap"
xmlns:ns3="kgomez.uci.cu"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Header>
  <ns3:Certificado SOAP-ENV:mustUnderstand="1" SOAP-
ENV:actor="SAAA">b1277e8065fe7ee572dcc56b78ed0970</ns3:Certificado>
</SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns1:Autorizar>
      <sComponente xsi:type="xsd:string">SAAA</sComponente>
      <sServicio xsi:type="xsd:string">EliminarRol</sServicio>
      <sRol xsi:type="xsd:string">Configurador</sRol>
      <sNivel xsi:type="xsd:string">Administracion</sNivel>
      <listaparametros xsi:type="ns2:Map">
        <item>
          <key xsi:type="xsd:string">rol</key>
          <value xsi:type="xsd:string">Administrador</value>
        </item>
      </listaparametros>
    </ns1:Autorizar>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 2.12: Ejemplo de mensaje SOAP utilizando el servicio *RegistrarTraza*

Este servicio primeramente verifica que el componente al que se le está haciendo la petición exista, así como que posea el servicio especificado. Seguidamente, se analiza si este servicio puede ser consumido por un usuario que sea del tipo *sRol* en el nivel *sNivel.* Luego, se analiza si el certificado es válido, es decir, que exista un usuario al que le pertenezca y que el mismo aún no haya expirado,

comparando la diferencia entre el momento en que se creó el certificado y el instante de tiempo en que está realizando la petición con el valor una variable de configuración global *$GlobalTiempoExpiracion* previamente asignada. El servicio analiza si el usuario tiene privilegios en el componente *sComponente,* pertenece al rol *sRol* de este componente, así como que tiene privilegios sobre el servicio *sServicio* en el nivel *sNivel*.

Si todas las condiciones descritas se cumplen satisfactoriamente, entonces la petición es autorizada, si al menos una de estas verificaciones se incumple, entonce el servicio retorna un mensaje de error (fault) describiendo el por qué no fue autorizada la petición en cuestión.

Este servicio también analiza las condiciones restrictivas de integridad que han sido previamente configuradas entre los componentes y sus servicios, pues mediante una implementación del patrón de diseño *observer*, se lleva a cabo el control de este conjunto de reglas de negocio. A continuación se describe la implementación del referido patrón.

Para garantizar el mantenimiento eficiente de la integridad referencial en el flujo de información distribuida, se identificó que este requerimiento debía ser configurado en el proceso de gestión de componentes. En tal sentido, fueron configuradas las restricciones de integridad que se establecen entre los componentes desplegados, teniendo en cuenta que estas pueden ser restrictivas o en cascada. Esta configuración es utilizada para comprobar las dependencias de integridad cada vez que sea realizada una petición desde los consumidores de servicio.

Para realizar estas configuraciones se cuenta en el SAAA con la siguiente tabla:

Figura 2.13: Descripción de la tabla tb_observer

Donde *id_componente_observador* e *id_componente_observado,* son llaves foráneas de la tabla *tb_codificador_componente*. Del mismo modo, sucede con el atributo *id_funcionalidad_observada,* el cual tiene una referencia de la *tabla tb_codificador_funcionalidades*. El atributo *tipo_restriccion,* es de tipo *boolean*, siendo verdadero si la restricción es en cascada y falso, si es restrictiva. El tipo de restricción restrictiva significa que si se desea modificar o eliminar información en el componente observado pero que se encuentra referenciada en algún componente observador, entonces no se permita llevar a cabo esta acción; mientras que si se especifica que la misma es en cascada, entonces se realiza la acción y los datos modificados o eliminados, son actualizados en los componentes observadores.

Una vez analizadas las condiciones iniciales de autorización descritas anteriormente, en caso de ser todas satisfactorias, entonces el servicio de autorización recorre la lista de componentes observadores de la petición involucrada y que la observan en forma restrictiva, a partir de esto se consume de cada uno de los componentes observadores un servicio booleano previamente implementado, que tiene la siguiente definición: *bool check($sComponente, $sServicio, $listaparametros).* El servicio *check* devuelve verdadero, si el componente observador no tiene los datos involucrados referenciados en su base de datos y falso en caso contrario. Este servicio es implementado en dependencia de los

negocios de los componentes observadores y es responsabilidad de sus desarrolladores definir las reglas de negocio para cada uno de los casos.

Si de al menos uno de los observadores involucrados, su servicio *check* devuelve falso, la petición no es autorizada lo cual es notificado mediante un mensaje de error (fault), se le indica al usuario que la petición no fue autorizada especificando el componente que denegó la misma por contener los datos a gestionar. En caso de que la respuesta de todos los servicios *check* sea verdadera, entonces finalmente se autoriza la petición.

En la **Figura 2.14** se muestra un ejemplo del método *check* implementado en un componente cualquiera. Como se puede observar, este es un observador restrictivo del componente *Ciudadanos* y servicio *EliminarCiudadano*, lo que implica que si posee una referencia al ciudadano que se desea eliminar, entonces el método retornará falso, para que el servicio de autorización deniegue la petición. Por cada dúo componente/servicio se debe adicionar una condición en el método *check*.

```
public function check($sComponente,$sServicio,$listaparametros)
{
  $con = new Conexion();
  if($sComponente == "Ciudadanos" && $sServicio == "EliminarCiudadano")
  {
   $id_ciudadano=$listaparametros['id_usuario'];
   $query="SELECT * FROM TABLA WHERE id=$id_ciudadano";
   $resultado = $con->queryAsArray($query);
   if(sizeof($resultado)>0)
    return false;
   else
    return false;
  }
}
```

Figura 2.14: Ejemplo de implementación del método *check*

Más adelante se explicará cómo se actualiza la información referenciada en los componentes observadores, si la restricción fuese en cascada. En este caso se utilizará la definición de otro servicio, también previamente implementado, al que se le denominará *update.*

2.3.2.3 RegistrarTraza. Exportar e importar trazas mediante ficheros XML

Este servicio permite a los componentes externos y al propio SAAA, el registro de *trazas* o *logs* asociados con las operaciones en las que se ven involucrados los servicios registrados, para posteriormente poder realizar búsquedas, con el fin de llevar a cabo acciones de auditoría sobre las trazas almacenadas. Asimismo, se le evita a todos los componentes el control de su trazabilidad, pues simplemente utilizando este servicio quedan almacenadas todas las acciones que se deseen registrar, en el SAAA.

Nombre del servicio: *RegistrarTraza*	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
sComponente	string
sFuncionalidad	string
sTipoTraza	string
txDescripción	string
PARÁMETROS DE SALIDA	
Descripción	**Tipo**
ResultadoRegistrarTraza	bool
	

Tabla 2.4: Descripción de los parámetros de entrada y salida del servicio *RegistrarTraza*

Como se observa, el servicio recibe como parámetros el nombre del componente, el nombre del servicio, el tipo de traza o error que se desea almacenar, así como una descripción de la misma. Primeramente, el servicio verifica que el componente *sComponente* se encuentre registrado; posteriormente verifica que este posea un servicio *sFuncionalidad.* Este servicio también incluye el certificado en la cabecera del mensaje, en este caso, se utiliza para definir el usuario que está registrando la traza.

Considerando el rápido crecimiento de las tablas de la base de datos que almacenen las trazas, se ha implementado un mecanismo de balanceo de las mismas entre dos tablas, *tb_traza* y *tb_traza_historica*. La primera de ellas es donde se encuentran las trazas de los usuarios que se encuentran activos en el sistema, de modo que si un usuario es eliminado, sus trazas pasan a la otra de las tablas, la cual no posee referencia alguna con los identificadores de la información que tenía referenciada la tabla *tb_traza*. Asimismo, el SAAA brinda la posibilidad de eliminar trazas de la primera de las tablas sin necesidad de que el usuario se encuentre eliminado. Finalmente, se pueden realizar eliminaciones de las trazas de la tabla *tb_traza_historica,* las cuales son almacenadas en el servidor mediante ficheros XML. A estas trazas también se le puede realizar auditoría, debido a que se permite realizar un almacenamiento temporal de las mismas en una tabla *tb_trazas_historica_recuperadas*.

Durante la configuración del SAAA se debe registrar los valores de cuatro variables de configuración, *$limite_t, $limite_th, $cant_mover_t* y *$cant_mover_th*, las dos primeras representan, la cantidad de trazas límite que pueden poseer las tablas *tb_traza* y *tb_traza_historica*, mientras que *$cant_mover_t* representa la cantidad de trazas de la tabla *tb_traza* que se moverán a la tabla *tb_traza_historica* en caso de que se alcance el límite *$limite_t.* De la misma forma, la variable *$cant_mover_th* representa la cantidad de trazas históricas que se almacenarán mediante ficheros XML en el servidor cuando se alcance el límite *$limite_th*. Es importante destacar que, a la hora de realizar estas eliminaciones sucesivas siempre se seleccionan las trazas más antiguas. En la siguiente figura se describe visualmente la solución:

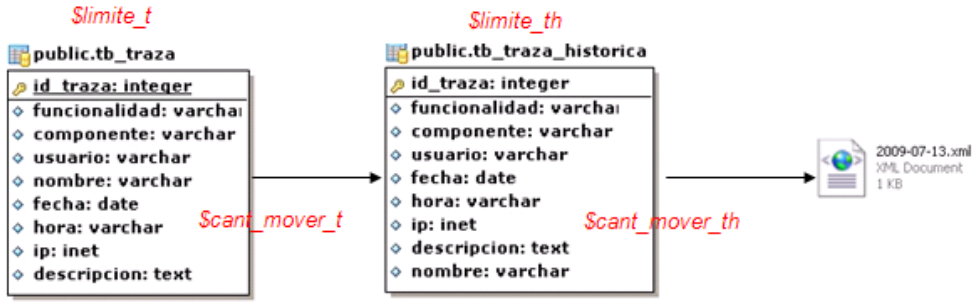


Figura 2.15: Descripción del mecanismo de balanceo de trazas implementado

Para ello se implementaron un conjunto de funciones en la base de datos que le evitan realizar este negocio al servicio *RegistrarTraza;* de modo que cada vez que se registra una nueva traza se chequeen las variables descritas anteriormente, ejecutándose de ser necesario este proceso. Cabe destacar que el mismo se puede realizar mediante interfaces visuales provistas por el SAAA, flexibilizando el proceso a que se realice cuando el *Administrador* desee llevar a cabo tareas de mantenimiento, inclusive, sin haber sido alcanzados los límites permitidos. Las funciones implementadas a nivel de base de datos se describen a continuación:

- **pa_trazas_to_xml(cantidad:integer):** Esta función recibe la cantidad de trazas que se desean eliminar además construye la raíz del documento XML, obtiene el identificador de la última de las trazas a incluir en el fichero, debido a que el identificador de estas en la tabla de trazas históricas es un autonumérico, entonces a medida que este aumenta disminuye su antigüedad. También obtiene los nombres de usuario involucrados en estas trazas. Finalmente, invoca a la función *pa_usuarios_to_xml(uss:varchar, id_max:integer)* para cada uno de los usuarios identificados. Esta es la función que permite construir la estructura completa del contenido del fichero de trazas que se desea exportar, la cual describirá finalmente por cada usuario, componente y funcionalidad, las trazas que ha generado. **(Ver Anexo 11)**

- **pa_usuarios_to_xml(uss:varchar, id_max:integer):** Esta función recibe como parámetros el nombre de usuario así como el identificador de la última traza a extraer de la tabla de históricas. Con estos parámetros se seleccionan para ese usuario aquellos componentes involucrados y para cada uno de ellos, se procede a invocar la *función pa_componente_to_xml(id:varchar, id_max: integer, uss:varchar).*

- **pa_componente_to_xml(id:varcahar, id_max: integer, uss varchar):** Esta función recibe como parámetros el nombre del componente, el identificador de la última traza a extraer de la tabla de históricas y el nombre de usuario. Con estos parámetros se seleccionan para cada componente las funcionalidades involucradas y para cada una de ellas se procede a invocar la función *pa_funcionalidad_to_xml(id:varchar, id_max:integer, comp:varchar, uss:varchar).*

- **pa_funcionalidad_to_xml(id:varchar, id_max:integer, comp:varchar, uss:varchar):** Esta función recibe como parámetros la funcionalidad y el identificador de la última traza a extraer de la tabla de históricas, el nombre del componente y el nombre de usuario. Con estos parámetros se seleccionan para cada funcionalidad las trazas involucradas y para cada una de ellas se procede a invocar la función *pa_traza_to_xml(id: integer).*

- **pa_traza_to_xml(id:varchar):** Esta función es de vital importancia en el proceso debido a que la misma para un identificador de traza determinado construye el tag asociado con los datos propios de esta. En la **Figura 2.16** se muestra un ejemplo de traza mediante un tag <*traza*>, creado por esta función:

```
<traza>
  <tipo>Acceso</tipo>
  <fecha>2010-10-06</fecha>
  <hora>17:38:12</hora>
  <ip>10.8.40.32/32</ip>
  <descripcion>Acceso Denegado</descripcion>
</traza>
```

Figura 2.16: Estructura XML para representar una traza

2.3.3 Implementación del patrón proxy

Para la implementación del patrón proxy, el SAAA brinda un servicio que permite, una vez realizada la autenticación y utilizando los servicios *Autorizar* y *RegistrarTraza,* llevar a cabo un análisis de seguridad estricto de todas las peticiones que se realizan entre los componentes. La definición del servicio *Peticion* es la que sigue:

Nombre del servicio: *Peticion*	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
sComponente	string
sServicio	string
sRol	string
sNivel	string
sCertificado	string
listaparametros	array
PARÁMETROS DE SALIDA	
Descripción	**Tipo**
ResultadoPeticion	array

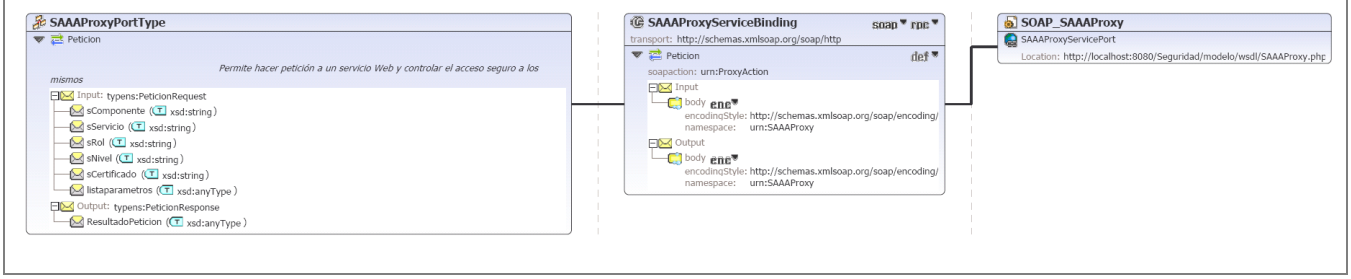


Tabla 2.5: Descripción de los parámetros de entrada y salida del servicio *Peticion*

La clase *Proxy* implementada posee dos métodos *Peticion* y *Consumir*, el primero de ellos es público y se expone mediante el servicio anterior, mientras que el segundo es privado y es utilizado por *Peticion* para redireccionar la petición hacia el proveedor *sComponente* que posee el servicio *sServicio*.

El método *Peticion* recibe los parámetros descritos anteriormente en la tabla anterior, lo primero que este comprueba es que el usuario con el certificado *sCertificado* posea los privilegios para consumir el servicio solicitado, para ello se utiliza el servicio *Autorizar,* en caso de que la autorización sea no

satisfactoria, se retorna al cliente el mismo mensaje de error que este último devolvió; este es el primer paso lo que implica que la petición al servicio deseado, no se construye hasta tanto no se consiga la autorización requerida.

Si el proceso de autorización fue satisfactorio entonces se procede a seleccionar la dirección del WSDL que describe al servicio teniendo en cuenta su disponibilidad. Téngase en cuenta que por cada servicio se registran varias direcciones donde el mismo se encuentra disponible, pudiendo aprovechar de esta forma la distribución de servicios e inclusive, tener el mismo servicio disponible en varios nodos o host. De este forma, cuando se tiene una dirección disponible entonces se invoca la función *Consumir($url,$servicio, $listaparametros),* donde sus parámetros significan la dirección del WSDL, el servicio que se desea consumir y la lista de parámetros necesarios respectivamente. Una vez obtenida la respuesta de este servicio, entonces se convierte en la respuesta del servicio *Peticion,* en caso de que durante la petición se lance una excepción (fault), esta se captura y se le envía al cliente.

Una vez realizada la petición y su respuesta es satisfactoria, y antes de devolvérsela al cliente, se debe completar la implementación del proceso de mantenimiento de la integridad referencial mediante el patrón *observer*. Recordando lo que sucede en el servicio *Autorizar* y lo que ocurre después que la autorización sea satisfactoria en el servicio *Peticion*, se tiene que, si la respuesta de todos los servicios *check*, de los componentes observadores es verdadera, entonces es redireccionada la petición hacia el componente observado y antes de retornar la respuesta del servicio involucrado en la petición, se notifica a los observadores cuya restricción es en cascada, para que actualicen su información en caso de tener registrada la que sufrió modificaciones en el componente observado. Para ello se implementa un servicio similar al *check* cuya definición es *bool update($componente,$servicio,$listaparametros)*, el que retorna verdadero, si se realizó alguna modificación y falso, en caso contrario.

De esta manera, son comprobadas todas las dependencias de integridad previamente configuradas al registrar un nuevo componente en el SAAA. Esta solución implicará no necesitar tablas caché en los futuros componentes a desarrollar, así como no tener que actualizar manualmente aquellas que se encuentran en los componentes actuales, proporcionando así el modelo propuesto un control estricto y total de las interdependencias entre los componentes distribuidos, protegiendo la información clínica contra estados corruptos e inconsistentes.

Resumiendo lo anterior descrito, el proceso para realizar una petición posee los siguientes pasos:

P1: Verificar si la petición es autorizada o no, a partir del certificado proporcionado.

P2: Seleccionar un proveedor de servicios disponible.

P3: Redireccionar la petición, mediante el método *Consumir*.

P4: Notificar los cambios a los casos en que las restricciones de integridad sean en cascada.

P5: Devolver al cliente el resultado de la petición

Para seleccionar un proveedor de servicios disponible, a continuación se describe un sencillo algoritmo de planificación, que permite seleccionar a cuál de las URL redireccionar la petición atendida por el *proxy*.

P2.1: Seleccionar del componente y servicio involucrado en la petición, aquellas URL, de las que tienen registradas para el mismo, que estén disponibles.

P2.2: Seleccionar de las URL que estén disponibles, la que permitió construir el cliente SOAP en el menor tiempo.

Seguidamente el servicio continúa en el paso tres, de esta forma, se evita que en un momento determinado se realice una petición a un servicio no disponible, así como que, de las direcciones en la que el mismo se encuentra disponible, se seleccione aquella que tiene la mayor posibilidad de devolver la respuesta en el menor tiempo.

Conclusiones

En el presente capítulo se realizó una descripción completa de la solución propuesta, haciéndose énfasis en los principales servicios Web implementados y que son publicados a los sistemas externos, para llevar a cabo los procesos de Autenticación, Autorización y Auditoría. Se describe la forma en que fueron implementados los patrones de diseño *proxy* y *observer*, para la centralización de las peticiones y el control de la integridad referencial en la información distribuida respectivamente, donde estas soluciones se integran satisfactoriamente con los servicios que publica el SAAA y los reutilizan.

CAPÍTULO III: VALIDACIÓN DE LA PROPUESTA DE PROCESOS DE AUTENTICACIÓN, AUTORIZACIÓN Y AUDITORÍA PARA APLICACIONES BASADAS EN SERVICIOS WEB XML

En el presente capítulo se realiza un análisis mediante un conjunto de criterios de medida que permiten evaluar los resultados obtenidos. Entre los indicadores que se tendrán en cuenta están la capacidad de reutilización de los servicios implementados y de generalización de la solución propuesta, así como, el rendimiento y eficiencia de los servicios implementados teniendo en cuenta variables como el tiempo mínimo, tiempo máximo y tiempo medio de respuesta del servidor para diferentes cantidades de usuarios concurrentes.

A partir de los resultados experimentales y los valores alcanzados por las variables antes descritas se realiza un ajuste de curvas, que permite obtener un modelo que represente aproximadamente a estos datos. La calidad del ajuste obtenido se mide en función de un conjunto de indicadores que representan la calidad del modelo, entre estos se encuentran:

- **SSE:** Es la suma de los cuadrados debido al error de la aproximación, si es cercano a cero indica que es un buen ajuste.
- **R-square:** Es el cuadrado del coeficiente de correlación múltiple o R^2, que para ser un buen ajuste deberá estar cercano a 1.
- **Adjusted R-square:** El cual se calcula con base en R^2, deberá tener un valor cercano a 1 para considerarse un ajuste confiable.
- **RMSE:** Mientras más cercano a cero sea, indicará un buen ajuste.

3.1 Sistema de Autenticación, Autorización y Auditoría

Los resultados de la investigación en sentido general forman parte de un Sistema de Autenticación, Autorización y Auditoría, el cual permite, mediante los procesos de administración proporcionados, una gestión eficiente de todos los requerimientos de seguridad para aquellos sistemas externos que consuman sus servicios, lográndose de esta forma la reutilización del código y evitándose que se realicen acciones innecesarias fuera de cada negocio. Este sistema integra todos los resultados de implementación obtenidos y permite mediante interfaces de usuario, realizar todos los requerimientos de software descritos con anterioridad, para el correcto funcionamiento de los servicios. Constituye el sistema que utilizarán los administradores para configurar todos los elementos de seguridad, que se les aplicará a los sistemas, que sean registrados en el SAAA y que consuman los servicios Web publicados por este.

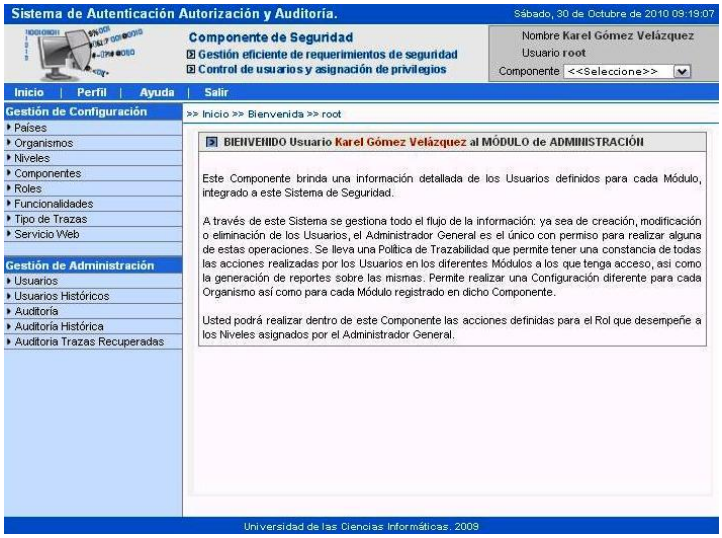


Figura 3.1: Página principal del Sistema de Autenticación, Autorización y Auditoría obtenido

3.1.1 Visión integrada de los resultados de implementación

A continuación se describe mediante un diagrama de actividades, el resumen general de los resultados de implementación obtenidos y que constituyen los principales aportes tanto desde el punto de vista informático como algorítmico de la presente investigación. De esta forma se ha completado la **Figura 1.1** *Estructura básica de un sistema de Autenticación, Autorización y Auditoría*, demostrándose cómo se puede extender el Modelo de AAA en forma natural a las aplicaciones basadas en servicios Web XML, utilizando en forma integrada los servicios implementados. De esta forma, se demuestra cómo se integran coherentemente cada uno de los resultados unitarios obtenidos. La secuencia general de pasos para la interacción, de un usuario perteneciente a un determinado sistema externo, con el SAAA es la que sigue:

Paso 1: Autenticación mediante el servicio *Autenticar.* Este retorna al cliente el certificado y su lista de derechos.

Paso 2: Realizar la petición al *proxy* mediante el servicio *Peticion*. Este a su vez, utiliza el servicio *Autorizar* para analizar que el usuario correspondiente tenga los privilegios para ejecutar la misma, así como que se comprueban las condiciones restrictivas de integridad existentes. Seguidamente, si la petición fue satisfactoria, realiza el consumo del servicio mediante el método *Consumir,* y obtiene así la respectiva respuesta. Se actualiza la información modificada en los observadores mediante su servicio *update*. Finalmente, se le envía la respuesta de la petición al cliente.

La secuencia de pasos descrita anteriormente se puede observar mediante el diagrama de actividades que se presenta en la **Figura 3.2**.

Los servicios implementados almacenan mediante el servicio *RegistrarTraza,* todas las acciones que impliquen interacción de los clientes o usuarios con los servicios, sean estas acciones satisfactorias o no, lo cual se hace de forma transparente para el usuario final, pero queda toda la correspondiente evidencia para su posterior auditoría.

Algunos consejos para el correcto diseño de los componentes observadores son los siguientes:

1. Analizar las necesidades de integración del componente observador que se esté desarrollando, con los componentes ya desarrollados o en fase de desarrollo.
2. Identificar la información que será reutilizada de estos componentes, es decir, aquellos datos que el componente en desarrollo almacenará en su base de datos, pero que la obtiene consumiendo servicios de los componentes observados.
3. Realizar un levantamiento de los métodos mediante los cuales será modificada o eliminada esta información reutilizada, teniendo en cuenta además el tipo de dependencia o restricción que se puede establecer.
4. Implementar los métodos *check* y *update*, en dependencia de las necesidades de los componentes. Para ello se tienen que realizar las consultas necesarias a la base de datos en dependencia de los argumentos que reciban los mismos. Si un componente sólo tiene restricciones en cascada entonces nada más se tendrá que implementar el servicio *update,* mientras que si únicamente posee dependencias restrictivas se implementará el servicio *check*. En el caso de identificarse ambos tipos de dependencias se deberán implementar ambos servicios.

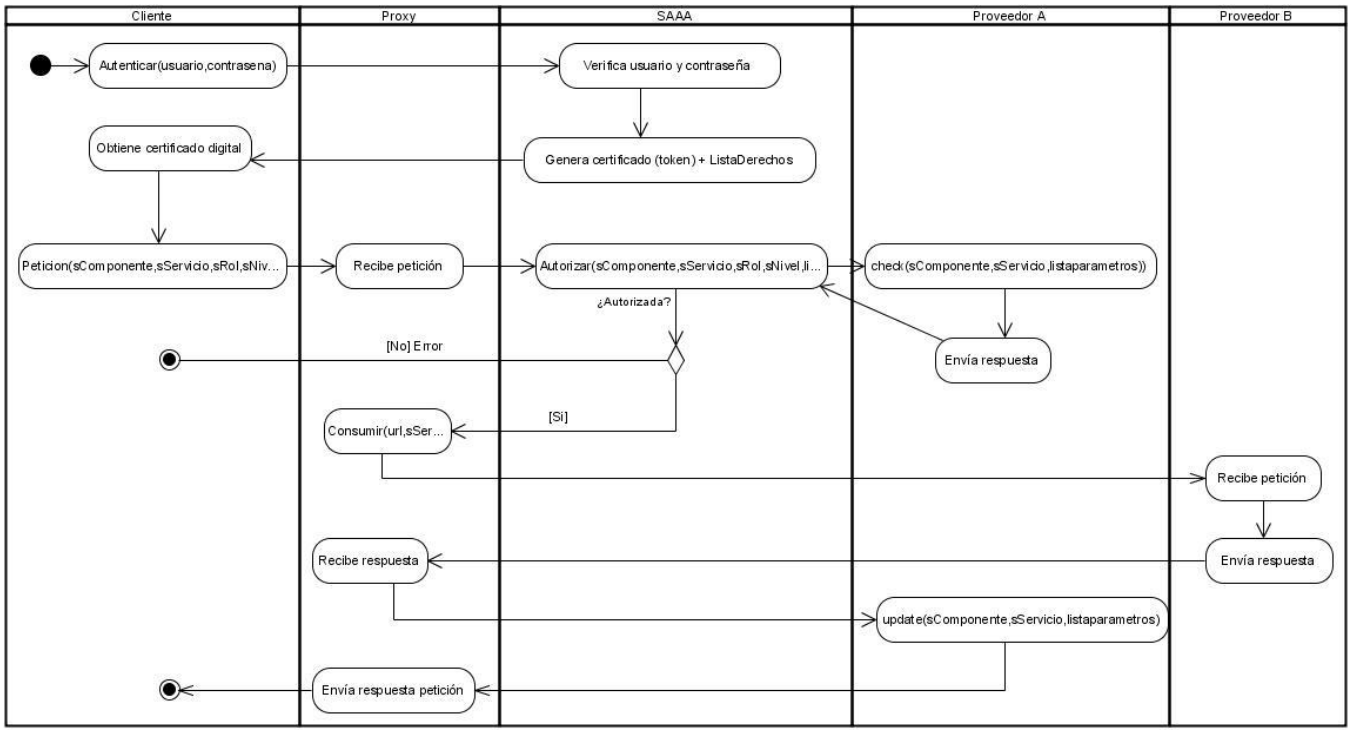


Figura 3.2: Representación integrada de los resultados de implementación

Se puede concluir que el Modelo AAA es completamente extensible a las aplicaciones basadas en servicios, debido a que es totalmente posible su implementación mediante estos. Queda explícitamente descrito también, cómo se integran todos los servicios implementados para proveer una visión única de seguridad a los usuarios finales, así como que, se dan algunos consejos para los desarrolladores que consideren el uso de la presente infraestructura.

3.2 Validación de los resultados de implementación

A continuación se muestran los resultados de la realización de un conjunto de pruebas funcionales a los cuatro servicios implementados, haciendo énfasis, en el análisis de los niveles de eficiencia y tiempo de respuesta de los mismos. Para el desarrollo de los casos de prueba se consideraron diferentes cantidades de usuarios concurrentes, a partir de asumir que los servicios implementados, se encontrarán disponibles en forma centralizada para todas las aplicaciones que deseen utilizarlos. Se debe tener en cuenta inicialmente que los WSDL de estos servicios fueron desarrollados mediante la suite de soluciones para el trabajo con documentos XML, Altova XML Spy 2010. Esta suite brinda la posibilidad de analizar si los WSDL están correctamente implementados y bien formados, así como construir peticiones SOAP al servidor, prefijando inclusive los tiempos máximos de respuesta.

Para el desarrollo de las pruebas se utilizó una computadora personal Hier, con 1 GB de memoria RAM y un procesador Core Duo a 1.87 Ghz de frecuencia, donde fue desplegado el SAAA completamente, incluyendo su base de datos.

Para el posterior análisis de los resultados obtenidos en las pruebas realizadas se medirán los tiempos de respuesta mínimos (*Tmín*) y máximos (*Tmáx*) en cada iteración de pruebas, así como el tiempo medio (*Tmed*) de respuesta a todas las peticiones, las que irán aumentando su nivel de concurrencia descrito mediante la variable (*Pcon*). Todos los tiempos serán medidos en segundos.

3.2.1 Pruebas unitarias

A continuación se muestra una tabla, con los resultados de las mediciones para los servicios *Autorizar* y *RegistrarTraza* respectivamente*:*

	Servcio *Autorizar*				Servcio *RegistrarTraza*			
Pconc	Tmín(s)	Tmáx(s)	Tmed(s)	Media (s)	Tmín(s)	Tmáx(s)	Tmed(s)	Media(s)
10	0.1251	0.2453	0.1744	0.5439	0.1539	0.1813	0.1667	0.4676
20	0.2623	0.3632	0.284		0.2385	0.2796	0.2591	
30	0.3698	0.4019	0.3716		0.3178	0.3425	0.3302	
40	0.424	0.4635	0.4473		0.3711	0.4042	0.3877	
50	0.4981	0.5184	0.5027		0.4102	0.4389	0.4246	
60	0.5252	0.5618	0.5485		0.4693	0.4932	0.4813	
70	0.5761	0.6332	0.6053		0.5187	0.5543	0.5365	
80	0.6528	0.726	0.6902		0.5993	0.6303	0.6148	
90	0.8345	0.8671	0.8412		0.6767	0.7044	0.6906	
100	0.9122	1.112	0.9739		0.7522	0.8167	0.7845	

Tabla 3.1: Resultados de las pruebas de rendimiento a los servicios *Autorizar* y *RegristrarTraza*

Se puede observar en la **Tabla 3.1** como los tiempos medios de respuesta para el servicio *Autorizar,* son mayores que para el servicio *RegistrarTraza* lo cual es algo lógico, considerando que el primero de estos, lleva mayor lógica de negocio que el segundo. A pesar de ello, en ambos casos el tiempo de respuesta del servidor fue muy satisfactorio si tiene en cuenta, que las pruebas no se están realizando en el escenario de hardware real donde debe ser desplegada esta solución en forma centralizada.

En las **Figuras 3.3** y **3.4** se muestran gráficos que representan la tendencia de los tiempos de respuesta en función de la cantidad de usuarios concurrentes utilizada en los casos de prueba anteriormente ejecutados, ajustados aproximadamente a las curvas:

$f(x) = 0.00000001346x^4 - 0.0000005881x^3 - 0.0001043x^2 + 0.01255x + 0.1372$, con los siguientes valores para los indicadores: $\text{SSE} = 0.0007899$, $\text{R-square} = 0.9987$, $\text{Adjusted R-square} = 0.9976$, $\text{RSME} = 0.01257$, lo que indica que se está en presencia de un buen modelo.

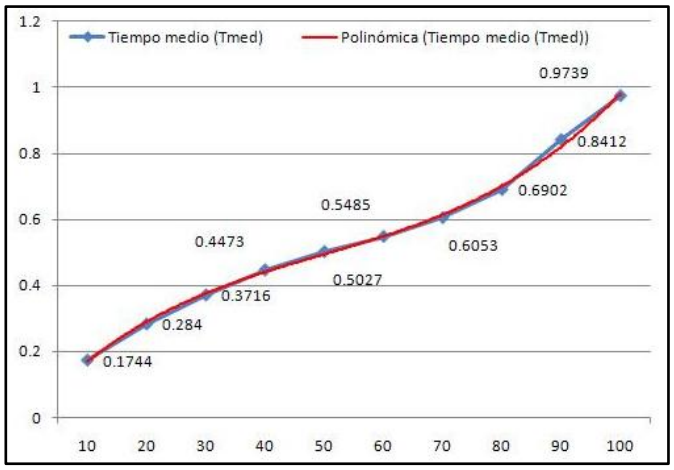


Figura 3.3: Gráfico del tiempo medio de respuesta del servicio *Autorizar*

$f(x) = -0.00000001104 + 0.000003333x^3 - 0.0003169x^2 + 0.017x + 0.02405$, con los siguientes valores para los indicadores: $\text{SSE} = 0.0001012$, $\text{R-square} = 0.9997$, $\text{Adjusted R-square} = 0.9995$, $\text{RSME} = 0.004499$, lo que indica que se está en presencia de un buen modelo.

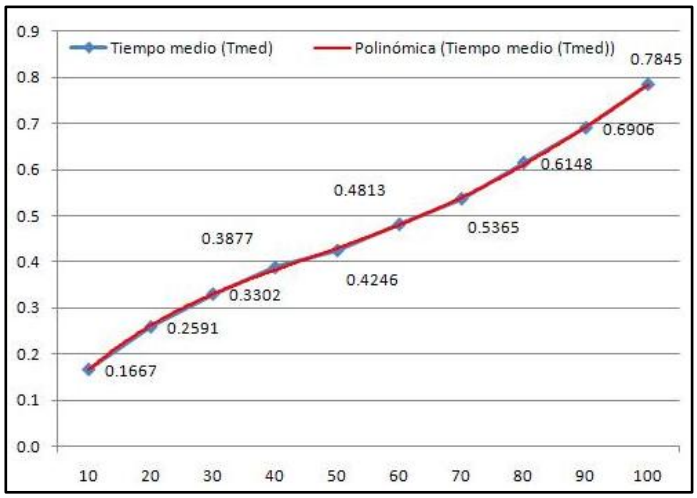


Figura 3.4: Gráfico del tiempo medio de respuesta del servicio *RegistrarTraza*

A partir del diseño experimental de casos de prueba para medir parámetros de eficiencia de los servicios *Autorizar* y *RegistrarTraza*, así como con el análisis de los resultados experimentales obtenidos, se puede arribar a la conclusión de que ambos poseen buenos niveles de eficiencia, dados por la media de sus tiempos de respuesta de aproximadamente 0.5439 y 0.4676 segundos respectivamente, con un grado de certidumbre del 95%.

3.2.2 Pruebas de integración

A continuación se muestra un conjunto de resultados, asociados al tiempo de respuesta del servidor al procesar las peticiones, teniendo en cuenta el uso del servicio *Peticion*, considerándose que el usuario está previamente autenticado. Para ello, se comparará además la diferencia en el tiempo de respuesta al tener el mismo servicio distribuido en varios nodos de cómputo con el caso en el que se encuentre disponible en solo uno de ellos. También se tendrá en cuenta, en los casos de prueba, el uso o no del algoritmo de planificación descrito en el capítulo anterior, para la atención más eficiente de las peticiones realizadas a un servicio de un determinado componente.

Para el desarrollo de las pruebas se utilizaron tres computadoras de 1 GB de memoria RAM y un procesador Intel Pentium a 3.00 Ghz de modo, que el hardware utilizado es similar para el caso de los nodos donde se desplegó el mismo servicio. El SAAA se mantuvo desplegado en la misma computadora personal descrita en el epígrafe anterior: *Proxy:* 10.36.10.12, *Nodo1:* 10.36.10.161, *Nodo2:* 10.36.10.32 y *Nodo3:* 10.36.10.111.

La siguiente tabla muestra los resultados de las pruebas realizadas, registrándose la cantidad de peticiones concurrentes (*Pconc*), el tiempo mínimo (*Tmín*), el tiempo máximo (*Tmáx*) y el tiempo medio (*Tmed*) de respuesta del servidor que atendió la petición, en todos los casos el mismo fue medido en segundos.

Planificación	Pconc	Tmín(s)	Tmáx(s)	Tmed(s)
No	10	0.1732	0.2087	0.1910
Si	10	0.1116	0.1285	0.1201
No	20	0.2311	0.2768	0.2540
Si	20	0.1523	0.1734	0.1629
No	30	0.3123	0.3582	0.3353
Si	30	0.2358	0.2573	0.2466
No	40	0.3865	0.4199	0.4032
Si	40	0.2789	0.2983	0.2886
No	50	0.4433	0.5494	0.4964
Si	50	0.3398	0.3751	0.3575
No	60	0.6021	0.6592	0.6307
Si	60	0.3963	0.4298	0.4131
No	70	0.6999	0.7629	0.7314
Si	70	0.4574	0.4830	0.4702
No	80	0.8423	0.8945	0.8684
Si	80	0.5327	0.5712	0.5520
No	90	0.9857	1.1254	1.0556
Si	90	0.6186	0.6373	0.6280
No	100	1.2919	1.3276	1.3098
Si	100	0.6515	0.7180	0.6848

Tabla 3.2: Resultados de las pruebas de integración mediante el servicio *Peticion*

En los resultados experimentales registrados en la tabla anterior, se puede observar como en todos los casos, a medida en que iba aumentando la concurrencia, los tiempos mínimos, máximos y medios, en

el caso en que se seleccionó el servidor en función del posible mejor tiempo de respuesta, se encontró por debajo, del caso en el que se prefijó un solo servidor. El tiempo medio de respuesta para el caso en que no se utilizó el algoritmo de planificación fue de 0.6276 segundos, mientras que para el caso en que si se tuvo en cuenta fue de aproximadamente 0.3923 segundos.

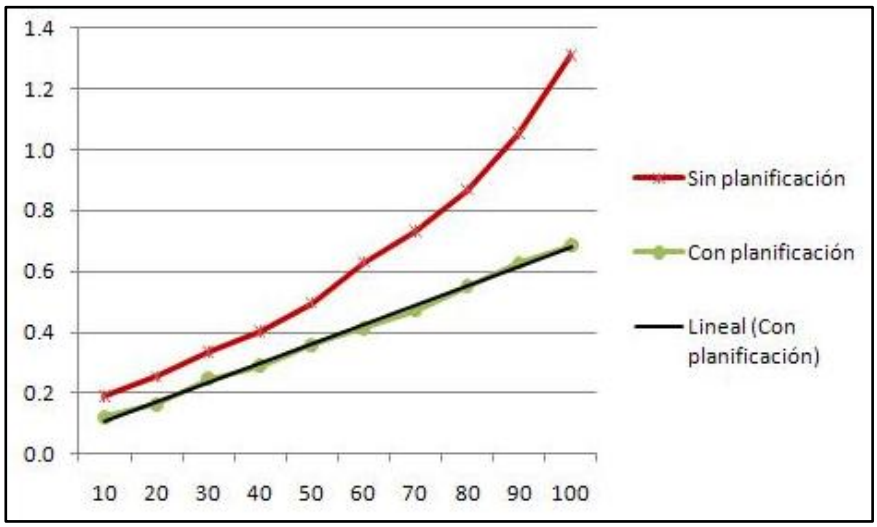


Figura 3.5: Gráfico del tiempo medio de respuesta del servicio *Peticion*

Asimismo se puede observar, que la tendencia al crecimiento en el caso en el que no se usa el algoritmo de planificación propuesto es exponencial, mientras que en el caso en el que si se aplica, la tendencia es aproximadamente lineal. Mediante la línea negra representada en el gráfico se puede corroborar lo anteriormente expresado, la cual se obtuvo mediante un ajuste de curvas utilizando un modelo lineal.

Teniendo en cuenta la teoría de análisis computacional, se puede decir que un algoritmo de planificación está satisfactoriamente implementado, pues en la medida que el modelo que ajusta los datos experimentales obtenidos se aproxime cada vez más a un modelo lineal, representará favorablemente cómo el algoritmo de planificación realizó la asignación distribuida de servicios correctamente.

Finalmente, se puede concluir parcialmente que a partir del desarrollo experimental de los casos de prueba aplicados y a partir de los tiempos medios de respuesta obtenidos en forma aproximada, que con el uso del algoritmo de planificación implementado, se puede reducir entre un 50 y 60 por ciento el tiempo medio de respuesta del servicio *Peticion*.

3.3 Guía de uso de la propuesta de solución

Para el uso efectivo por parte de los sistemas externos de los servicios que de Autenticación, Autorización y Auditoría que son provistos como parte de la solución, se presenta una plantilla que permite recopilar la información necesaria para el registro de componentes en el SAAA. Las secciones II, III y IV destinadas a las funcionalidades, servicios y roles del componente respectivamente, serán repetidas en función de la cantidad de la cantidad de funcionalidades, servicios y roles que posee el componente respectivamente.

FORMULARIO PARA EL REGISTRO DE COMPONENTES		
I. Datos generales		
Nombre (*):		
Acrónimo o Nombre corto (*): *<No puede poseer espacios>*		
URL WSDL:	URL Pág. Princ.:	
Descripción:		
II. Funcionalidades		
Nombre (*):		
Descripción (*):		
III. Servicios		
Nombre: *<No puede poseer espacios, tiene que ser exactamente igual como está descrito en el WSDL>*		
Descripción:		
URL WSDL1:		
URL WSDL2:		
URL WSDL3:		
IV. Roles		
Nombre (*):		
Descripción (*):		
Listado de las funcionalidades y servicios a los que tiene acceso (*). *<Tienen que ser de las relacionadas en la seccion de funcionales y servicios>*		
V. Configuración de las restricciones de integridad		
Componente:	Servicio	Tipo: R: Restrictiva o C: Cascada
<De los ya registrados en el SAAA >		

Tabla 3.3: Formulario para el registro de componentes

Seguidamente se muestra el formulario que obtiene la información necesaria para el registro de un nuevo usuario en el sistema:

FORMULARIO PARA EL REGISTRO DE USUARIOS
1. **Datos generales**
Nombre y apellidos (*):
Dirección correo electrónico:
Nombre de usuario:
¿Es administrador? ____Sí [Nivel: Ubicación:] ____No
II. Componentes o sistemas a los que tiene acceso
Nombre (*):
Organismo (*):
II.1 Niveles por roles (*):
Rol:

Listado de niveles a los que tiene acceso este rol de los que posee el componente en los niveles que posee el organismo.
II.2 Servicios por nivel y rol:
Rol: Nivel:
Listado de servicios a los que tiene acceso cada rol en los niveles de acceso que este posee.
II. 3 Otros datos del componente o sistema:
¿Está activo? ____Sí ____No
Periodo de actividad (Sólo si está activo) F.Inicio: _ /__ /__ F.Fin: __ /__ /__

Tabla 3.4: Formulario para el registro de componentes

Entre la información general que se recoge de cada usuario, se encuentra saber si el mismo es administrador o no, en el caso de que este lo sea, entonces por defecto se le asignarán estos privilegios en el SAAA, pero respondiendo a la estructura jerárquica de creación de usuarios definida. Es decir, un determinado administrador sólo puede crear otros en su nivel inmediato inferior, de esta forma, aumentan los niveles de seguridad durante la asignación de este tipo de privilegios. Las secciones II.1 y II.2 se repetirán en función de la cantidad de roles que se le vayan a asignar al usuario, de los previamente configurados para el componente, mientras que por cada configuración rol/nivel definida en la sección II.1, se debe especificar posteriormente la lista de servicios que pueden ser consumidos.

Brindar estos formularios, agiliza el proceso de puesta a punto de un determinado componente que desee integrarse con los servicios resultantes de la presente investigación, pues los desarrolladores de los mismos desde un primer momento, conocen la información que podrán obtener a partir de su uso, así como la que deben proveer para su correspondiente registro.

3.4 Valoración de la innovación y aporte práctico de los resultados. Beneficios

Una vez concluida la investigación es necesario evaluar la innovación, aporte práctico y capacidad de generalización de la misma, así como analizar si la hipótesis definida para guiarla fue cumplida satisfactoriamente, la cual plantea que *"Si se implementan los requerimientos de seguridad propuestos, en las aplicaciones basadas en servicios Web XML entonces se homogenizan los procesos de Autenticación, Autorización y Auditoría de los productos desarrollados por el Centro de Informática Médica (CESIM)"*. Entre los elementos que se pueden destacar que permiten concluir que la hipótesis ha sido validada satisfactoriamente, se encuentran los siguientes:

1. Se provee un diseño homogéneo y estandarizado para la implementación de un modelo que describe cómo deben gestionarse los procesos de Autenticación, Autorización y Auditoría para aplicaciones basadas en servicios Web XML.
2. Se ha implementado el diseño propuesto, utilizando estándares internacionales para la interoperabilidad de aplicaciones Web así como considerando el uso patrones de diseño que le imponen mayor robustez y escalabilidad a la solución.
3. Se cuenta con una completa y centralizada gestión de los requerimientos de seguridad para todos los productos desarrollados por el Centro de Informática Médica basados en servicios,

pues se reutilizan los requerimientos que provee el SAAA, trayendo consigo una disminución del tiempo de desarrollo de las aplicaciones y un considerable aumento de sus niveles de seguridad y confiabilidad de las mismas, al reutilizar software ya probado y especializado en los procesos antes descritos.

4. Aumento de los niveles de integración entre las diferentes aplicaciones del Centro de Informática Médica, evitando que cada uno posea de manera aislada la administración de sus usuarios.
5. Protección a la información gestionada y compartida evitando que la misma se encuentre en estados inconsistentes o corruptos, pues se proporcionan chequeos de integridad sobre el flujo de información intercambiada entre los sistemas mediante la implementación del patrón *observer*.
6. Se proporcionan mejoras para el rendimiento de las aplicaciones, debido a que la información asociada a sus usuarios, privilegios y trazas no es gestionada por las mismas. Así, como por la consideración de mecanismos que actúan cuando el volumen de datos relacionados con la información auditable crece, mediante la implementación de una política de balanceo de trazas para las tablas donde se almacena la información de esta naturaleza y para el control centralizado de las peticiones mediante una implementación del patrón *proxy*.
7. Constituye un valor agregado a los productos desarrollados por el Centro de Informática Médica, debido a que además de resolver su negocio tendrán integrado su propio sistema de seguridad.
8. Gestión eficiente del proceso de Auditoría para los productos, permitiendo hacer reportes de las trazas históricas de acuerdo con diferentes parámetros y controlando la acumulación de información de esta naturaleza en las bases de datos. La eliminación de los usuarios y sus respectivas trazas es lógica y no física, fortaleciéndose el proceso de auditoría.
9. Permite la integración con servidores LDAP, posibilitando la reutilización de los datos de usuarios de un organismo determinado, evitando de esta forma la duplicación de la información en la base de datos.
10. La solución obtenida no sólo se limita a las aplicaciones del ámbito de la salud, sino que puede ser totalmente extensible y escalable a cualquier aplicación cuyo mecanismo de interoperabilidad sean los servicios Web XML y se desee controlar los accesos al consumo de los mismos, implicando por tanto una amplia posibilidad de reutilización y generalización.
11. Concepción de una “Guía” que permite emplear los resultados de la presente investigación por parte de los diseñadores y desarrolladores de sistemas.

Actualmente la propuesta de solución gestiona los requerimientos de seguridad de un conjunto de aplicaciones desarrolladas por el CESIM. A continuación se relacionan las mismas y se da una breve descripción de los procesos que gestionan:

- *Sistema de Rehabilitación Integral (alas SRI):* Gestiona la información generada en los procesos de rehabilitación por los que transita un paciente en las salas de rehabilitación, de una institución hospitalaria. La solución constituye una aplicación Web, para ofrecer servicios al personal en línea que faciliten la ejecución de sus tareas.

- *Red Nacional de Nefrología (alas NefroRed):* Concebida para la gestión de la información de los pacientes en el programa de diálisis que necesitan de algún método sustitutivo para poder vivir. Contempla la hemodiálisis y la diálisis peritoneal. Permite la gestión de los estudios complementarios que permiten el seguimiento evolutivo del paciente. Apoya el proceso de control, tratamiento y seguimiento de pacientes que padecen de enfermedades renales crónicas. Está regida por las normas y estándares establecidos por la comisión médica nacional para el tratamiento de las enfermedades renales crónicas de Cuba y está orientada a mejorar la calidad de vida de estos pacientes.
- *Control Sanitario Internacional (alas SCI):* Es un sistema integrado de gestión de información de salud del Programa para el Control Sanitario Internacional que contempla tres subprogramas, permitiendo prevenir, detectar la introducción y evitar la propagación en el país de enfermedades exóticas, emergentes y reemergentes y adoptar las medidas necesarias con la retroalimentación adecuada a los distintos niveles del Sistema Nacional de Salud.
- *Bloque Quirúrgico Oftalmológico (alas BQO):* Es un sistema para la gestión de la información correspondiente a los procesos clínicos y quirúrgicos que se llevan a cabo en las áreas oftalmológicas de los hospitales o instituciones especializadas en el tema.
- *Sistema de Balance y Planificación de insumos médicos (alas BAP):* Es un sistema diseñado para gestionar en una unidad de salud toda la información relacionada con la planificación y almacén, permitiendo esto la integración entre ambos procesos. El sistema permite realizar y consultar las planificaciones de la unidad de salud en un período de tiempo planificado, logrando que estas sean lo más exacta, según las necesidades de la unidad de salud.
- *Sistema de Docencia Médica (alas DOC):* Sistema que gestiona la información generada en los procesos docentes por los que transita un recurso humano en formación de pregrado o posgrado, en una institución. El sistema permite el registro y actualización de los datos de matrícula de los recursos humanos en formación ya sea de pregrado o posgrado, profesores o personal involucrado en los procesos docentes, así como datos de caracterización de los mismos. Permitirá además el registro de todos los datos referentes a los movimientos que pueden realizar en cualquier etapa de su formación.
- *Sistema para el control de recetas médicas:* Sistema que permite el control de la asignación de recetas médicas en los diferentes niveles de la atención de salud, así como finalmente conocer a qué personal médico fue al que se le entregó un determinado conjunto de recetas médicas, para llevar a cabo la prescripción de los medicamentos.

De esta forma, se puede plantear que la solución propuesta constituye la base para la gestión de los requerimientos de seguridad de las aplicaciones desarrolladas por el CESIM de la UCI; lo que implica, que se garantice una homogeneidad en los procesos de Autenticación, Autorización y Auditoría, objetivo principal de la presente investigación, el que se considera haber cumplido satisfactoriamente.

Conclusiones

La solución obtenida posee una amplia relevancia en el contexto sobre el cual se desarrolla, debido a que el país viene realizando importantes esfuerzos en pos del progreso de la informática en todas las actividades que se realizan en el ámbito de la salud. La obtención de estos procesos de seguridad,

garantiza una homogénea y estandarizada forma de gestionar los mismos, basada en modelos internacionales que describen y documentan satisfactoriamente, cuáles son los requerimientos necesarios para garantizar la seguridad en las aplicaciones Web.

El desarrollo de las pruebas experimentales realizadas a los servicios Web implementados demuestra la eficiencia de los mismos, pues se obtuvieron tiempos de respuesta medios de 0.5439 y 0.4676 segundos, para los servicios *Autorizar* y *RegistrarTraza* respectivamente, con un grado de certidumbre del 95%. Asimismo, se puede concluir que con el uso del algoritmo de planificación implementado, se puede reducir entre un 50 y 60 por ciento, el tiempo medio de respuesta del servicio *Peticion*.

CONCLUSIONES

El desarrollo de la presente investigación permitió arribar a las siguientes conclusiones:

- El proceso de integración, es un hecho inevitable cuando se utilizan Arquitecturas Orientadas a Servicios (SOA) y Basadas en Componentes (CBA), elemento que debe ser considerado desde el proceso de conceptualización de un nuevo sistema de software.
- El estudio del estado del arte permitió identificar los principales componentes o elementos estructurales presentes en un modelo de Autenticación, Autorización y Auditoría (AAA), así como los requerimientos de seguridad indispensables para los servicios Web XML y las aplicaciones que los utilizan como mecanismo de interoperabilidad.
- Se obtuvo un modelo de requerimientos de seguridad con su respectiva implementación, que permite la gestión homogénea de los procesos de Autenticación, Autorización y Auditoría (AAA), para las aplicaciones basadas en servicios Web XML, desarrolladas por el Centro de Informática Médica (CESIM).
- Los resultados obtenidos tienen un alto nivel de escalabilidad, reutilización de componentes y generalización del modelo implementado.
- Los servicios implementados, poseen un satisfactorio nivel de eficiencia validado a partir del desarrollo experimental de casos de prueba y la valoración de los resultados obtenidos mediante la teoría de análisis computacional.

RECOMENDACIONES

Para el desarrollo de trabajos futuros asociados con los resultados obtenidos a partir del desarrollo de la presente investigación se recomienda:

- Generalizar los requerimientos de seguridad propuestos para la gestión homogénea de los procesos de Autenticación, Autorización y Auditoría (AAA) en otras soluciones no sólo del ámbito de la informática médica, sino en otro tipo de aplicaciones especializadas que los requieran.
- Incorporar nuevos requerimientos funcionales que permitan garantizar seguridad no sólo en el acceso a los servicios, sino también, en el contenido de los mensajes SOAP que se intercambian a partir del consumo de estos, mediante encriptación y firma digital XML; donde se utilicen las especificaciones de OASIS *XML Encryptation* y *XML Digital Signature* respectivamente.

REFERENCIAS BIBLIOGRÁFICAS

[Aruquipa et al., 2007] Aruquipa Chambi Marcelo G., Márquez Granado Edwin P. *Desarrollo de Software Basado en Componentes.* Universidad Mayor de San Andrés. La Paz Bolivia, 2007.Disponible en: http://pgi.umsa.bo/enlaces/investigacion/pdf/INGSW3_23.pdf

[Atkinson et al., 2002] Atkinson, B. et al. *Web Services Security (WSSecurity)*. IBM. 2002. Disponible en: http://www-128.ibm.com/developerworks/library/wssecure/#majorhead3

[Cabrera, et al.,2004] Cabrera, L., Kart, C., Box, D. *An Introduction to the Web Services Architecture and Its Specifications***.** Microsoft Corporation. 2004. Disponible en: http://msdn.microsoft.com/webservices/webservices/understanding/advancedwebservices/default.aspx?pull=/library/en-us/dnwebsrv/html/introwsa.asp#introwsa-upd_topic6a

[Chelsea, 2000] Chelsea Valentine; Chris Minnick. *XHTML serie práctica.* EUA. 2000 ISBN: 8420530115

[Corredera, 2005] Corredera de Cola, Luis Enrique. *Seguridad en XML.* Universidad Pontificia de Salamanca. Madrid. España. 2005

[Crespo et al., 2005] Crespo, Yania. *Integrando un modelo de reutilización en la producción de software: entorno distribuido para el desarrollo basado en reutilización.* Departamento de Informática. Universidad de Valladolid. España. Disponible en: http://giro.infor.uva.es/Publications/2003/CLP03/CLP03.pdf

[Delgado, 2006] Delgado Ramos, Ariel. *Presentación Informatización del Sistema Nacional de Salud*. Dirección Nacional de Registros Médicos y Estadísticas de Salud. Ministerio de Salud Pública. La Habana. Cuba. 2006.

[Delgado, 2009] Delgado Ramos, Ariel; Rodríguez Díaz, Alfredo; Cabrera Hernández, Mirna. *Estrategia de Informatización del Sistema Nacional de Salud.* VII Congreso Internacional de Informática en Salud. Informática 2009. La Habana. Cuba. 2009. Disponible en: http://www.informatica2009.sld.cu/Members/mirnacabrera/estrategia-de-informatizacion-del-sistema-nacional-de-salud/at_download/trabajo

[Guinea et al., 2007] Guinea de Salas, Alejandro; Jorrin Abellán, Sergio. *Arquitectura SOA para la integración entre software libre y software propietario en entornos mixtos.* I Jornada de Software de Información Geográfica Libres. Girona. España. Disponible en: http://www.sigte.udg.edu/jornadassiglibre2007/comun/1pdf/13.pdf

[Hurtwitz, et al., 2007] Hurtwitz, Judith, et al. *Services Oriented Architectures for Dummies.* Wiley Publishing, Inc. Indianapolis. EUA. 2007

[ISO 27000, 2008] ISO. Organización Internacional de Estandarización. *Sistemas de gestión de seguridad de la información. 2008.* Disponible en: http://www.27000.org/

[Izenpe, 2010] Izenpe S.A, Empresa de certificación y servicios. *Zain, Plataforma de*

servicios de firma. 2010. Disponible en: http://www.izenpe.com/s15-12020/es/contenidos/informacion/firma_zain/es_firma/firma_zain.html#01

[Kicillof, 2004] Kicillof, Nicolás. *Estilos y Patrones en la estrategia de arquitectura de Microsoft.*Universidad de Buenos Aires. Argentina. 2004. Disponible en: http://www.willydev.net/descargas/prev/Estiloypatron.pdf.

[LDAP, 2004] LDAP.es. *¿Qué es LDAP?*. Disponible en: http://www.ldap-es.org/

[López, 2006] López Millán Gabriel. *Definición de una infraestructura de control de acceso basada en la arquitectura AAA y el uso de credenciales de autorización.* Tesis Doctoral. Departamento de Ingeniería de la Información y las Comunicaciones. Universidad de Murcia. España. 2006. Disponible en: http://ants.dif.um.es/staff/gabilm/tesis/G.Lopez.PhD.pdf

[Microsoft, 2010] Microsoft Developer Network (MSDN). *Conceptos básicos de seguridad.* Microsoft Corporation. Disponible en: http://msdn.microsoft.com/es-es/library/z164t8hs.aspx

[Motsumoto et al., 2002] Motsomoto, Makoto et al. *What is Mersenne Twister (MT)?* Disponible en: http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/ewhat-is-mt.html

[Ortega, 2006] Ortega Martorell, Sandra. *Administración de la comunicación de los sistemas del Registros de Estado Civil basada en contratos.* Tesis para optar por el título de Máster en Informática Aplicada. CUJAE, Cuba. 2006

[OWASP, 2005] OWASP. *Una guía para contruir aplicaciones y servicios web seguros.* 2005. Disponible en: https://www.owasp.org/images/b/b2/OWASP_Development_Guide_2.0.1_Spanish.pdf

[OWASP, 2010] The Open Web Application Security Project. Disponible en: http://www.owasp.org/

[Paz, 2007] Paz Madrid Gorelov, Vadim. *Servicios web. Artículo asociado a Doctorado en Informática y Automática.* Universidad de Salamanca. España. 2007. Disponible en: http://zarza.usal.es/~fgarcia/doctorado/iweb/05-07/Trabajos/ServiciosWeb.pdf

[PCMAG, 2010] PCMAG Encyclopedia. Disponible en: http://www.pcmag.com/encyclopedia_term/0,2542,t=Web+application&i=54272,00.asp

[Pérez, 2007] Pérez, Carlos. *Garantía de confidencialidad del flujo de información, control de acceso y gestión de identidades.* Revista Red Seguridad. No. 48. Madrid. España. Disponible en: http://www.borrmart.es/articulo_redseguridad.php?id=1138

[Pressman, 2005] Pressman, Roger S. *Ingeniería de Software, un enfoque práctico. Parte 1*. La Habana, Cuba. Editorial Félix Varela. 2005.

[Redwine, 2008] Redwine, Samuel T. Jr. *Toward an Organization for Software System Security Principles and Guidelines*. Institute for Infrastructure and Information Assurance, James Madison University, IIIA Technical Paper 08-01. February 2008. Disponible en: http://www.jmu.edu/iiia/webdocs/Reports/SwA_Principles_Organization-

sm.pdf

[Rodríguez, 2010] Rodríguez, J. *¿Qué es el control de acceso en sistemas informáticos?* 2011. Disponible en: http://www.subinet.es/guias-y-tips/guias-tips-internet/%C2%BFque-es-el-control-de-acceso-en-sistemas-informaticos/

[Ronda, 2004] Ronda Amador, Yoel. *Desarrollo de aplicaciones Web multicapas sobre plataforma "Open Source": experiencias de diseño e implementación.* Tesis para optar por el título de Máster en Informática Aplicada. CUJAE, Cuba. 2004

[Safelayer, 2009] Safelayer Secure Communications S.A. *TrustedX, Plataforma de servicios de confianza.* España. 2009. Disponible en: http://www.safelayer.com/pdf/TrustedX_es.pdf

[SOFTEL, 2006] SOFTEL. *Documento sobre la Arquitectura de Software a emplear en los componentes del Sistema de Información para la Salud.* La Habana. Cuba. 2006.

[Sosnoski, 2009] Sosnoski, Dennis. *Servicios web. El alto costo de WS-Security*. IBM. 2009. Disponible en: http://www.ibm.com/developerworks/ssa/library/j-jws6/index.html

[Sun, 2002] Sun Microsystem. *Building Web Services*. 2002. Disponible en: http://www.sun.com.

[VeriSing, 2010] VeriSing. Inc. *Preguntas frecuentes: Conceptos básicos sobre SSL.* España, 2010. Disponible en: http://www.verisign.es/ssl/ssl-information-center/ssl-basics/index.html

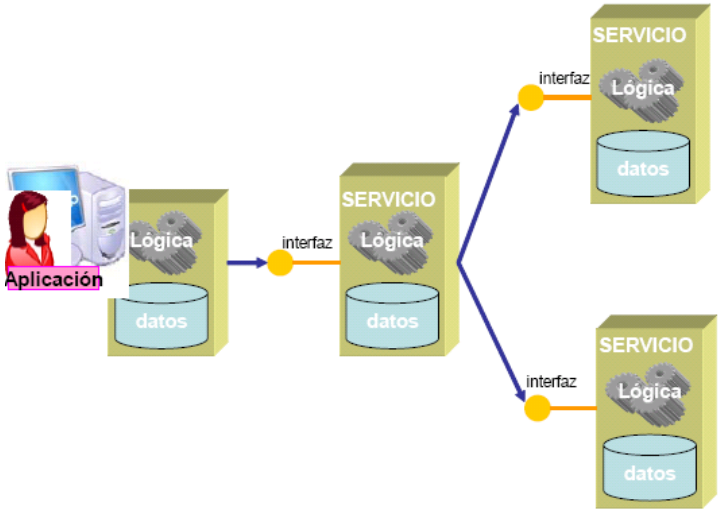
[Wolter, 2005] Wolter, Roger. *XML Web Services Basic.* Microsoft Developer Network (MSDN). Microsoft Corporation. EUA. Diciembre 2005. Disponible en: http://msdn.microsoft.com/en-us/library/ms996507.aspx

[W3C, 2004] World Wide Web Consortium (W3C). *Web Services Architecture*. 2004 Disponible en: http://www.w3.org/TR/ws-arch/

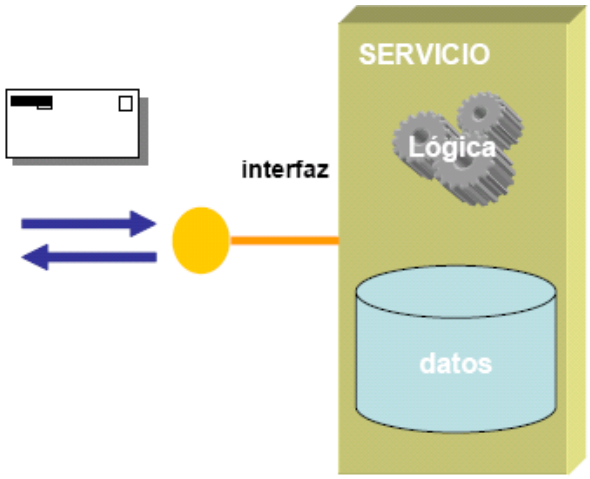
[W3C, 2010] World Wide Web Consortium (W3C). *Guía breve de Servicios Web.* Disponible en: http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb

ANEXOS

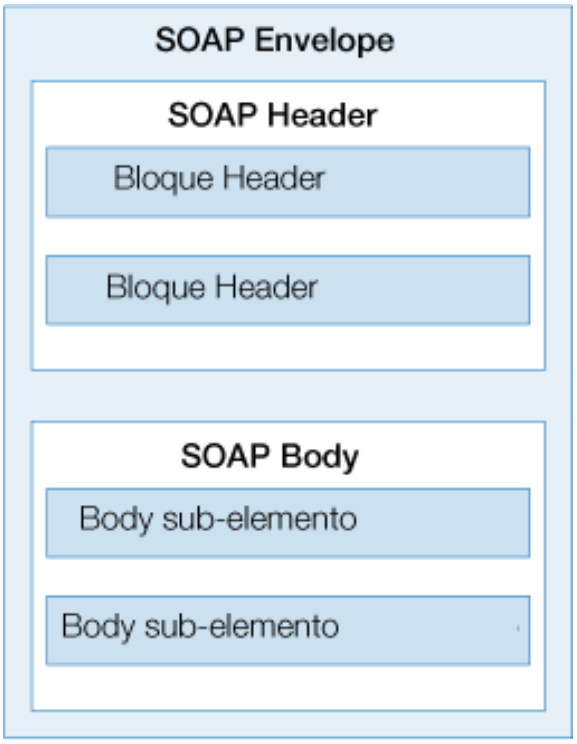
Anexo 1: *Ejemplo sencillo de sistema basado en servicios.*



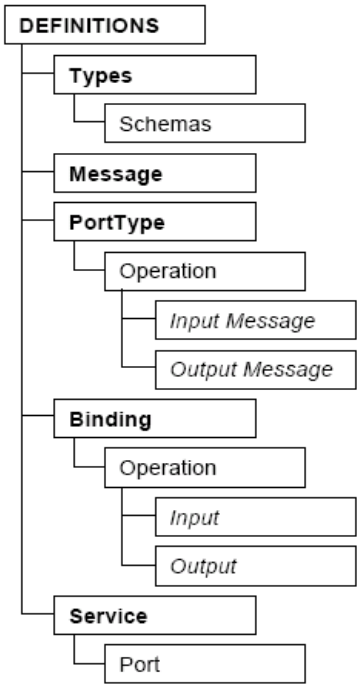
Anexo 2: *Visión interna de un servicio.*



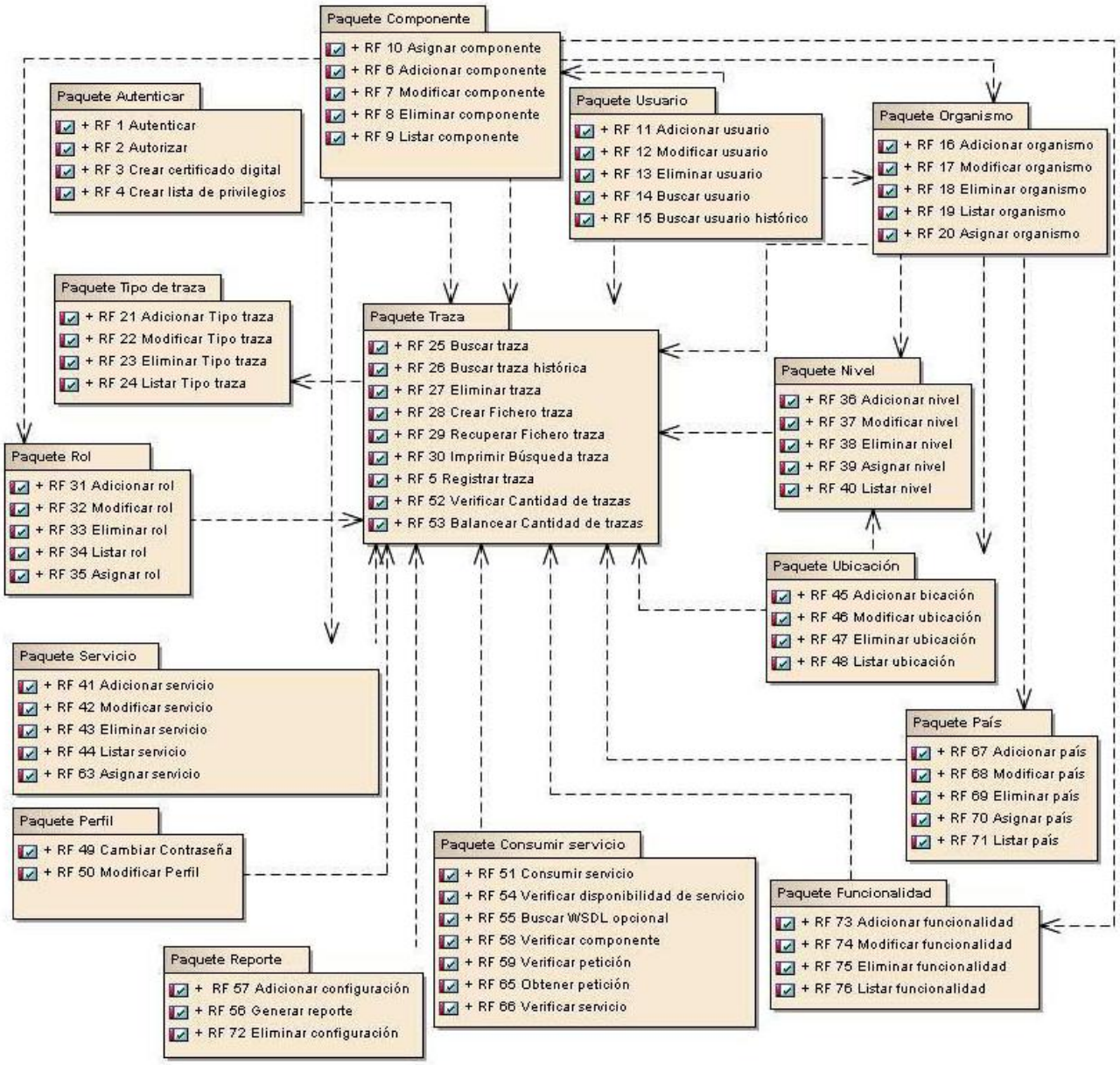
Anexo 3: *Estructura de los mensajes SOAP.*



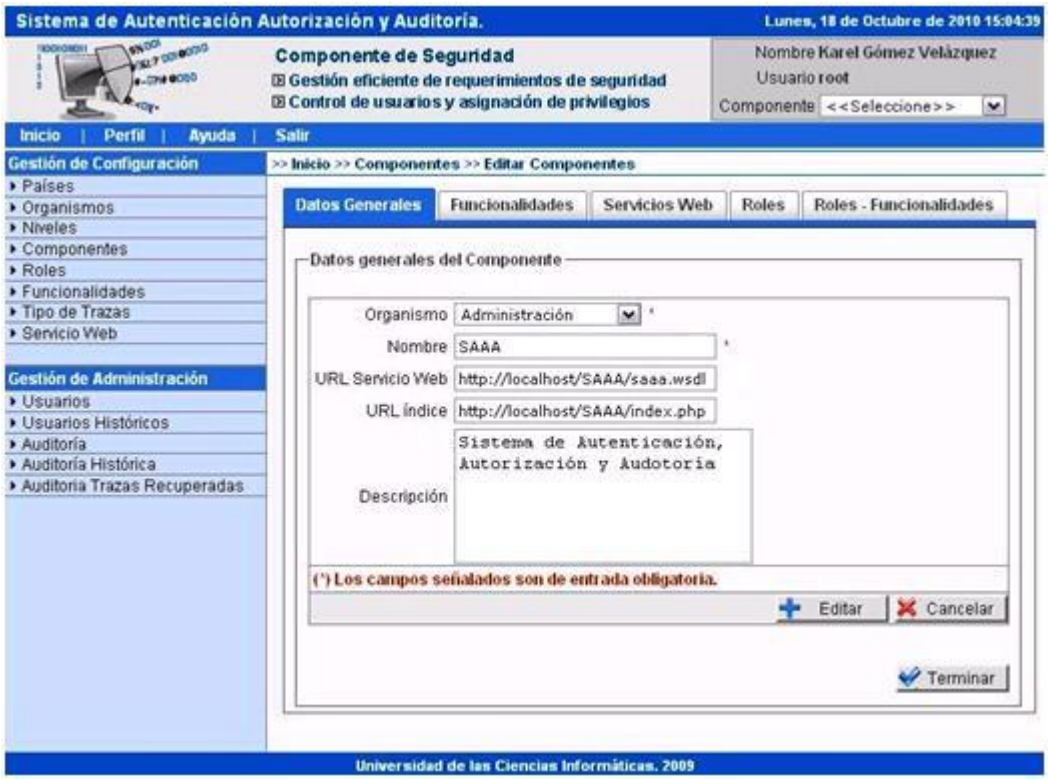
Anexo 4: *Esquema de un documento WSDL.*



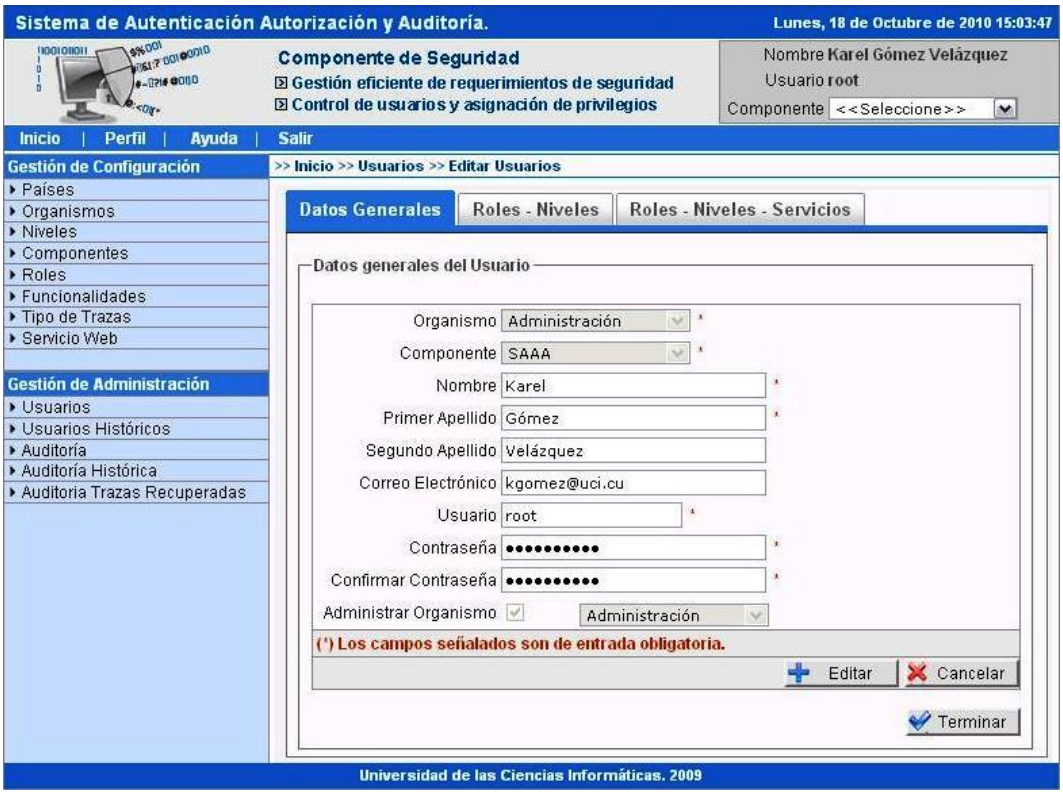
Anexo 5: *Requerimientos funcionales agrupados por paquetes*



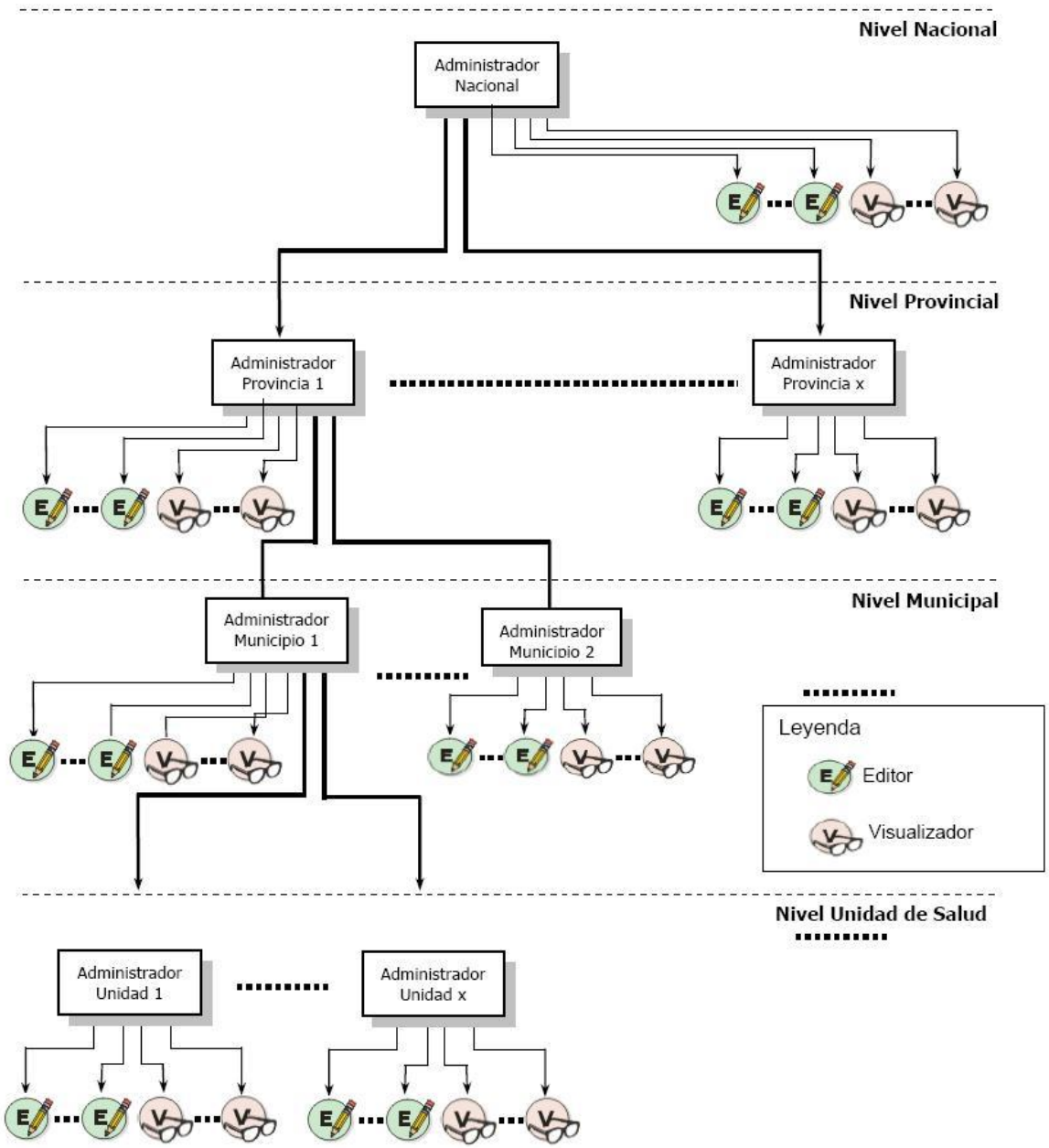
Anexo 6: *Interfaz de usuario para Adicionar componentes*



Anexo 7: *Interfaz de usuario para Adicionar usuarios*



Anexo 8: *Ejemplo de administración jerárquica de usuarios*



Anexo 9: *WSDL del servicio Autenticar*

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 (http://www.altova.com) by KAREL (UCI) -->
<definitions xmlns:typens="urn:SAAAAutenticar"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/" name="SAAAAutenticar"
targetNamespace="urn:SAAAAutenticar">
      <wsdl:types>
      <schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:SAAAAutenticar">
                  <complexType name="Usuario">
                        <all>
                         <element name="certificado" type="xsd:string"/>
                         <element name="correo" type="xsd:string"/>
                         <element name="error" type="xsd:string"/>
                         <element name="idusuario" type="xsd:string"/>
                         <element name="ListaDerechos"
type="typens:StructComponente"/>
                              <element name="nombre" type="xsd:string"/>
                              <element name="usuario" type="xsd:string"/>
                        </all>
                  </complexType>
                  <complexType name="ArrayResultadoAutenticar">
                        <complexContent>
                              <restriction base="soapenc:Array">
                                    <attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:Usuario[]"/>
                              </restriction>
                        </complexContent>
                  </complexType>
                  <complexType name="StructComponente">
                        <complexContent>
                              <restriction base="soapenc:Array">
                                    <attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:Componente[]"/>
                              </restriction>
                        </complexContent>
                  </complexType>
                  <complexType name="Componente">
                        <all>
                              <element name="ListaRoles" type="typens:StructRol"/>
                              <element name="componente" type="xsd:integer"/>
                              <element name="nombrecomponente" type="xsd:string"/>
```

```
                              <element name="url" type="xsd:string"/>
                        </all>
                  </complexType>
                  <complexType name="StructRol">
                        <complexContent>
                              <restriction base="soapenc:Array">
                                    <attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:Rol[]"/>
                              </restriction>
                        </complexContent>
                  </complexType>
                  <complexType name="Rol">
                        <all>
                              <element name="ListaNiveles"
type="typens:StructNivel"/>
                              <element name="nombrerol" type="xsd:string"/>
                              <element name="rol" type="xsd:integer"/>
                        </all>
                  </complexType>
                  <complexType name="StructNivel">
                        <complexContent>
                              <restriction base="soapenc:Array">
                                    <attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:Nivel[]"/>
                              </restriction>
                        </complexContent>
                  </complexType>
                  <complexType name="Nivel">
                        <all>
                              <element name="ListaServicios"
type="typens:StructServicio"/>
                              <element name="nivel" type="xsd:integer"/>
                              <element name="nombrenivel" type="xsd:string"/>
                        </all>
                  </complexType>
                  <complexType name="StructServicio">
                        <complexContent>
                              <restriction base="soapenc:Array">
                                    <attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:Servicio[]"/>
                              </restriction>
                        </complexContent>
                  </complexType>
                  <complexType name="Servicio">
                        <all>
                              <element name="nombreservicio" type="xsd:string"/>
                              <element name="servicio" type="xsd:integer"/>
```

```
                        </all>
                  </complexType>
            </schema>
      </wsdl:types>
      <message name="autenticarRequest">
            <part name="usuario" type="xsd:string"/>
            <part name="contrasena" type="xsd:string"/>
      </message>
      <message name="autenticarResponse">
            <part name="autenticarReturn" type="typens:Usuario"/>
      </message>
      <portType name="SAAAAutenticarPortType">
            <operation name="Autenticar">
                  <documentation>
                        Autenticación de usuario
                  </documentation>
                  <input message="typens:autenticarRequest"/>
                  <output message="typens:autenticarResponse"/>
            </operation>
      </portType>
      <binding name="SAAAAutenticarServiceBinding"
type="typens:SAAAAutenticarPortType">
            <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="Autenticar">
                  <soap:operation soapAction="urn:UsuarioAction"/>
                  <input>
                        <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:SAAAAutenticar"/>
                  </input>
                  <output>
                        <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:SAAAAutenticar"/>
                  </output>
            </operation>
      </binding>
      <service name="SOAP_SAAA">
            <port name="SAAAAutenticarServicesPort"
binding="typens:SAAAAutenticarServiceBinding">
                  <soap:address
location="http://localhost:8080/Seguridad/modelo/wsdl/SAAAAutenticar.php"/>
            </port>
      </service>
</definitions>
```

Anexo 10: *WSDL del servicio Autorizar*

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 (http://www.altova.com) by KAREL (UCI) -->
<definitions xmlns:typens="urn:SAAAAutorizar"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/" name="SAAARegistrarTraza"
targetNamespace="urn:SAAAAutorizar">
      <wsdl:types>
            <schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:SAAAAutorizar">
                  <element name="Certificado">
                        <complexType>
                              <attribute name="Certificado" use="required"
type="xsd:string"/>
                        </complexType>
                  </element>
            </schema>
      </wsdl:types>
      <message name="AutorizarRequest">
            <part name="sComponente" type="xsd:string"/>
            <part name="sServicio" type="xsd:string"/>
            <wsdl:part name="sRol" type="xsd:string"/>
            <wsdl:part name="sNivel" type="xsd:string"/>
      </message>
      <message name="AutorizarResponse">
            <part name="ResultadoAutorizar" type="xsd:boolean"/>
      </message>
      <message name="Header">
            <wsdl:part name="Certificado" element="typens:Certificado"/>
      </message>
      <portType name="SAAAAutorizarPortType">
            <operation name="Autorizar">
                  <documentation>
                        Permite la autorización de peticiones
                  </documentation>
                  <input message="typens:AutorizarRequest"/>
                  <output message="typens:AutorizarResponse"/>
            </operation>
      </portType>
      <binding name="SAAAAuBinding" type="typens:SAAAAutorizarPortType">
            <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="Autorizar">
```

```
                  <soap:operation soapAction="urn:AutorizarAction" style="rpc"/>
                  <input>
                        <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:SAAAAutorizar"/>
                        <soap:header message="typens:Header" part="Certificado"
use="literal"/>
                  </input>
                  <output>
                        <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:SAAAAutorizar"/>
                  </output>
            </operation>
      </binding>
      <service name="SOAP_SAAA">
            <port name="SAAAAAutorizarPort" binding="typens:SAAAAuBinding">
                  <soap:address
location="http://localhost:8080/Seguridad/modelo/wsdl/SAAAAutorizar.php"/>
            </port>
      </service>
</definitions>
```

Anexo 11: *Ejemplo de un fichero XML de trazas históricas*

```
<trazashistoricas>
 <usuarios>
    <usuario>
       <nombre>root</nombre>
       <componentes>
         <componente>
            <nombre> SAAA</nombre>
            <funcionalidades>
              <funcionalidad>
                <nombre>Eliminar roles</nombre>
                <trazas>
                   <traza>
                     <tipo>Acceso</tipo>
                      <fecha>2010-10-25</fecha>
                      <hora>19:23:17</hora>
                      <ip>10.8.40.20/32</ip>
                      <descripcion>Acceso Denegado</descripcion>
                </traza>
              </trazas>
            </funcionalidad>
            </funcionalidades>
         </componente>
       </componentes>
```

```
    </usuario>
  </usuarios>
</trazashistoricas>
```