Pontificia Universidad JAVERIANA Bogotá

Ingeniería de Sistemas

PROPUESTA PARA PROYECTO DE GRADO

TÍTULO

WS-Guardian Client's Agent

MODALIDAD

Proyecto de pregrado

OBJETIVO GENERAL

Diseñar un agente que se ejecute en el lado del consumidor de servicios web, el cual proporcionará principios de seguridad como *autorización, confidencialidad, integridad y no repudio* sobre los mensajes emitidos hacia *WS-Guardian.*

ESTUDIANTE(S)

Christian Giovanny Rojas Diaz _____

Documento	Celular	Correo Javeriano
cc. 1085315457	314-753-0222	christian-rojas@javeriana.edu.co

Juan David Gama Peña _____

Documento	Celular	Correo Javeriano
cc. 1018497844	312-330-2066	gamaj@javeriana.edu.co

Michael Andres Vargas Buitrago _____

Documento	Celular	Correo Javeriano
cc. 1056412777	321-297-3192	michael.vargas@javeriana.edu.co

Rie Kaneko Bojaca _____

Documento	Celular	Correo Javeriano
cc. 1032477898	321-356-0042	rkaneko@javeriana.edu.co

William Alexander Moreno Prieto _____

Documento	Celular	Correo Javeriano
cc. 1010236308	305-792-7706	williammoreno@javeriana.edu.co

DIRECTOR

Ing. Javier Humberto Galindo _____

Documento	Celular	Teléfono fijo	Correo Javeriano	Empresa donde trabaja y cargo
cc. 79793227	310-2679516	7444822	javier.galindo@itac.co	ITAC CEO ITAC

Historial de Cambios

Fecha	Autor	Descripción
07/03/2019	Todos	Se realizó el capítulo 1, visión global.
19/03/2019	Todos	Se realizó la corrección del primer capítulo según el comentario del profesor y Javier.
28/03/2019	Todos	Se realizó el capítulo 5, marco teórico.
30/03/2019	Todos	Se realizó los otros capítulos que faltan en el documento.
19/05/2019	Christian	Realizó las correcciones.

Contenido

1 Visión global

1.1 Antecedentes, problema y solución propuesta

1.1.1 Descripción de la problemática u oportunidad

Hoy en día existen múltiples aplicaciones desarrolladas en diferentes lenguajes de programación, muchas de ellas requieren intercambiar información con otros sistemas. De aquí surgen las *Arquitecturas Orientadas a Servicios* (SOA) que aportan interoperabilidad entre los aplicativos con el uso de los estándares abiertos [1].

La *Arquitectura Orientada a Servicios* permite la flexibilidad de un sistema unificado, que posibilita la rápida respuesta a los cambios en las necesidades de los negocios para así lograr las comunicaciones inherentes entre los diferentes lenguajes de programación. Los programas que trabajan con esta arquitectura envían datos a través de la Internet, exponiéndose a amenazas de seguridad presentes en su periodo de transporte. Esto puede ser mitigado con *Virtual Private Networks* (VPN) o protocolos seguros como *https*. Pero, cuando se habla a nivel de mensaje, la información tiene mayor riesgo de ser vulnerada y los principios de seguridad informática (autenticación, autorización, confidencialidad e integridad) se pueden ver afectados [2].

La empresa *ITAC* (Innovación para la Transformación Digital) ofrece el producto *WS-Guardian* (WSG) que es una herramienta que potencia la arquitectura *SOA* protegiendo el tráfico de entrada y salida del lado del servidor de *Servicios Web* (SW) bajo criterios de confianza y privacidad a nivel de mensajes por medio de técnicas como *encriptación*, *username-token*, *timestamp*, *firmas* y *certificados digitales*. WSG se caracteriza por brindar gobernabilidad y administración de políticas de seguridad sobre los SW y que esté asegura la transferencia de archivos e información sin la necesidad de alterar las aplicaciones utilizadas por el receptor implicado [3].

Como se mencionó anteriormente, WSG aplica controles de acceso y seguridad semántica a nivel del mensaje para asegurar que estos sean protegidos completamente, y no solo a nivel de tráfico como lo hace el protocolo https. Para que la información viaje protegida desde el origen hasta el destino, ya sea en comunicaciones cliente/WSG o WSG/servidor, el emisor debe tener un alto conocimiento de ciberseguridad para implementar técnicas de XML signature, XML encryption, timestamp y username-tokens [4]. Esto afecta negativamente la mantenibilidad y compatibilidad del sistema, puesto que cualquier cambio que se haga en WSG implica realizar cambios en las aplicaciones del consumidor de servicios y da un motivo más para que una compañía no implante medidas de aseguramiento a sus datos.

1.1.2 Formulación del problema

Actualmente, los clientes de *WS- Guardian* son los responsables de asegurar la *autenticación, confidencialidad, no repudio e integridad* por medio de *tokens de usuario, certificados y/o firmas digitales.* Muchos clientes ya cuentan con un despliegue de sus aplicaciones. Por ende, hacer una modificación e implementación de estos sistemas es costoso en materia de tiempo, dinero y esfuerzo. Pero, *WS-Guardian Client's Agent* siendo la solución propuesta de este proyecto, necesita una menor inversión de recursos y además aumenta la seguridad de la información.

1.1.3 Propuesta de solución

Para dar solución a la problemática se propone implantar un Agente llamado *WS-Guardian Client's Agent* en la máquina del cliente, para poder aplicar políticas que aseguren *confidencialidad*, *autorización*, *integridad* y *no repudio* en los mensajes emitidos hacia WSG. El agente brinda ciertas funcionalidades como ser capaz de solicitar, descargar e integrar las funcionalidades que requiera (Introspection and Reflection) [5] para cumplir con las restricciones de seguridad en el momento de consumir SW. Esta solución se enmarca en el área de desarrollo de software, debido a que un entregable será un prototipo que implemente ciertos mecanismos de seguridad para los mensajes dirigidos a WSG. Adicionalmente, este agente se despliega en la máquina del cliente de manera separada de sus aplicaciones.

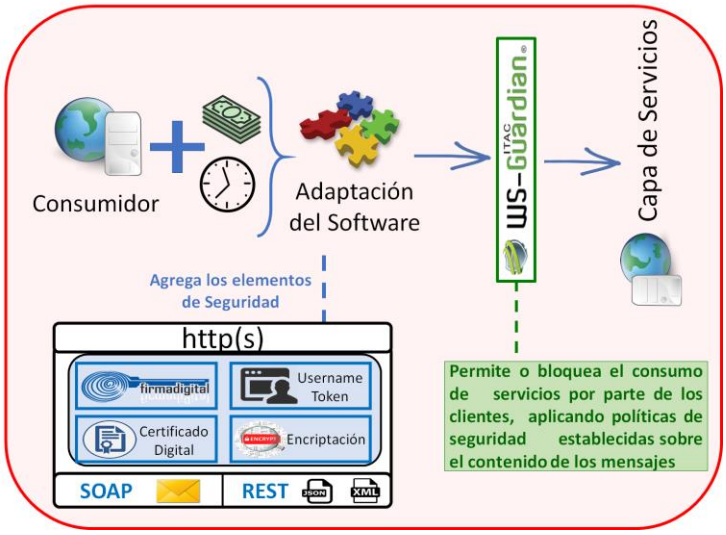

Figura 1 - Infraestructura actual

La *Figura 1 - Infraestructura actual* ilustra la arquitectura actual de comunicación entre el consumidor y la capa de *SW* mediante mensajes *http* o *https*. *WSG* aparece como una capa intermedia para la protección de los *Servicios sensibles* (Servicios del negocio del cliente que requieren seguridad para ser consumidos) expuestos por una organización. La figura también muestra la inversión de tiempo y dinero que debe hacer el cliente al adaptar su sistema con políticas de seguridad antes de comunicarse con WSG.

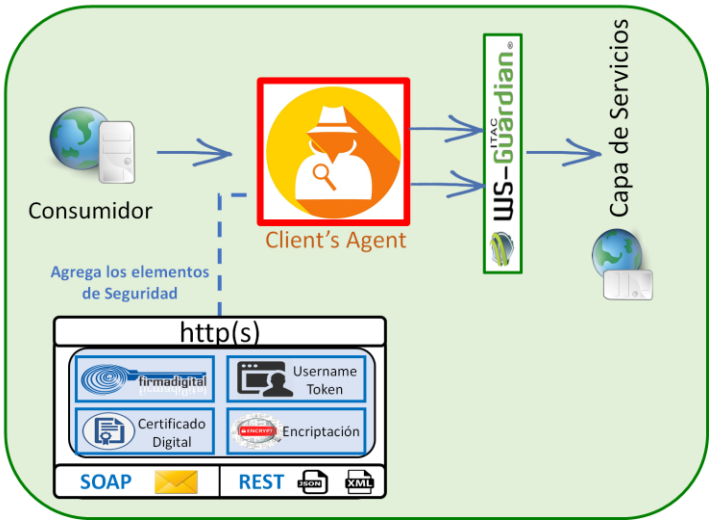

Figura 2 - Arquitectura general de la solución

En la *Figura 2 - Arquitectura general de la solución* se expone la solución a partir de la arquitectura de comunicación entre el consumidor, la capa de *Servicios Web*, *WS-Guardian* y el nuevo elemento *WS*-Guardian *Client's Agent*. Este último reemplaza la adaptación del sistema que debe hacer el cliente en su aplicativo para el cumplimiento de las políticas de seguridad.

1.1.4 Justificación de la solución

Existen clientes que no tienen experiencia en ciberseguridad y sus peticiones viajan por la red sin estar aseguradas. Para fomentar la seguridad en la red con apoyo del producto WSG, surge el agente *WS-Guardian Client's Agent,* que le ahorra al cliente tener que implementar técnicas de aseguramiento en su sistema. Dicho agente trabajará con los protocolos http y https, al ser los soportados por SW. La solución se plantea en la máquina del cliente para que la petición del servicio pueda ser asegurada desde su origen hasta su destino, evitando que viaje de forma vulnerable.

Este trabajo permitirá mejorar la adopción de la seguridad y evitar modificaciones que se puedan requerir en el cliente, ya que se reciben las peticiones, se evalúan y se transmiten de forma directa hacia WSG, quien validará la autorización de consumo.

En la *Tabla 1 - Fortalezas y Debilidades* se evidencia el análisis de Fortalezas y Debilidades de la solución con respecto a otras soluciones enfocadas a resolver la misma problemática.

Soluciones	Fortalezas	Debilidades
WS-Guardian Client's Agent	● Seguridad a nivel de mensaje ● Ligero y Portable ● Despliegue en multiplataformas ● Administración de políticas de seguridad	● Comunicación única con WS-Guardian
VPN	● Seguridad a nivel de transporte ● Independiente de la plataforma	● Necesidad de pago extra monetario para acceder al servicio ● Los mecanismos de seguridad no son manipulables
Proxy	● Bloqueo de peticiones no autorizados como un firewall. ● Administración de criterios para el bloqueo de mensajes	● Limitaciones de operatividad (no modifica ni retransmite las peticiones) ● Limitaciones de despliegue ● Es necesario configurar la aplicación que usará este recurso
TLS	● Seguridad a nivel de transporte ● Proporciona Interoperabilidad ● Adopción de nuevos algoritmos de cifrado	● Necesidad de pago monetario para acceder al servicio ● Las políticas de seguridad son establecidas y no cambiantes

Tabla 1 - Fortalezas y Debilidades

1.2 Descripción general del proyecto

El agente a desarrollar se encuentra en el marco de la seguridad informática. Este actúa como mecanismo de seguridad e implanta en los mensajes de petición políticas de seguridad para finalmente enviar la solicitud a WSG. De esta manera, se cumple con los parámetros normativos de confidencialidad, autorización, integridad y no repudio que exige *WSG* para consumir cada *SW*.

1.2.1 Objetivo general

Diseñar un agente que se ejecute en el lado del consumidor de servicios, el cual proporcionará principios de seguridad como *autorización, confidencialidad, integridad y no repudio* sobre los mensajes emitidos hacia WS-Guardian.

1.2.2 Objetivos Específicos

- Diseñar el agente para que pueda ser ejecutado en dos o más Plataformas
- Implementar un prototipo del agente para dos o más Plataformas
- Implementar dos técnicas de seguridad dentro del agente
- Determinar la validez de los requerimientos funcionales y no funcionales por medio del prototipo
- Demostrar la calidad de software mediante una metodología de pruebas

1.3 Entregables, estándares utilizados y justificación

Al finalizar el proyecto, se reflejarán los entregables presentes en la *Tabla 2 - Entregables y Estándares.* En esta tabla se muestran los estándares asociados a cada entregable, junto a su respectiva descripción.

Entregable	Estándares asociados	Justificación
PMP	ISO/IEC/IEEE 16326-2009 UML BPMN	Para el Plan de Administración de Proyecto de Software se tomó como base el estándar IEEE 1058.1-1987, adaptando su contenido para ese proyecto. En este documento se describe una vista general, contexto y administración del proyecto. De la misma manera se definen herramientas, estándares y metodología de trabajo para su desarrollo [6]. Se utiliza estándar UML (Unified Modeling Language y BPMN (Business Process Model and Notation) para representación de diagramas de secuencia y procesos que se llevarán a cabo en el proyecto [7], [8].
SRS versión inicial y final	IEEE 830-1998 UML BPMN	La plantilla para la Especificación de Requerimientos de Software es tomada del estándar IEEE 830-1998. Aquí se especifican todas las funcionalidades y atributos pertenecientes al sistema a desarrollar, junto con los lineamientos acordados con el cliente (ITAC) [9]. Se utiliza estándar UML (Unified Modeling Language y BPMN (Business Process Model and Notation) para representación de diagramas de secuencia y procesos que se llevarán a cabo en el proyecto [7], [8]. Se genera una versión final del documento con el fin de complementar requerimientos que no se contemplaron en las fases iniciales.
SDD versión inicial y final	IEEE 1016-2009 UML BPMN	Se adoptó el estándar IEEE 1016-2009 para el Documento de Diseño del Software con el fin de establecer la arquitectura y las interfaces lógicas de usuario que serán diseñadas e implementadas [10]. Se utiliza estándar UML (Unified Modeling Language y

		BPMN (Business Process Model and Notation) para representación de diagramas de secuencia y procesos que se llevarán a cabo en el proyecto [7], [8]. Se genera una versión final del documento con el fin de complementar aspectos de overview y casos de uso que no se tuvieron en cuenta en la primera versión.
Informe y Documento de Pruebas	ISO/IEC/IEEE 29119 OWASP ASVS V2	En este documento, se pretenden mostrar las pruebas y resultados del testing realizados sobre el prototipo funcional con el fin validar el correcto funcionamiento de los módulos implementados y el cumplimiento de los requerimientos. Se utiliza el estándar ISO/IEC/IEEE 29119 dado que incluye pruebas dinámicas y permite el uso de metodologías ágiles [11]. Se incluirá el estándar OWASP ASVS V2 ya que en este se verifican controles de seguridad para aplicaciones que manejan transacciones y activos sensibles [12].
Prototipo Funcional / Código Fuente	Java Code Conventions ISO/IEC/IEEE 12207-2017 Versionado Semántico 2.0.0-rc.2	Al finalizar el proyecto, el equipo de trabajo entregará un prototipo funcional acorde a los requerimientos especificados en el documento SRS. Se entregará un código que cumpla con el estándar para que sea mantenible y legible, con el fin de permitir que cualquier ingeniero pueda continuar con el desarrollo del software [13]. Se utiliza el estándar ISO/IEC/IEEE 12207- 2017 dado que proporciona procesos que se pueden emplear para definir, controlar y mejorar el ciclo de vida del software [14]. El estándar de versionamiento semántico define como asignar y aumentar los números de versión a lo largo del desarrollo del software [15].
Manuales de Usuario	IEEE - 1063 - 2001	Junto al prototipo funcional se entregarán manuales de usuario útiles para la administración del aplicativo, los cuales deben cumplir con el estándar para que sean entendibles [16].

Tabla 2 - Entregables y Estándares

2 Análisis de impacto

En caso de una realización satisfactoria del trabajo de grado, se impactará de manera directa a las empresas involucradas (ITAC S.A y Clientes), indirectamente se impactará en la sociedad al generar adopción de seguridad de forma automática.

- **Corto plazo**

En un periodo de entre 6 meses y 1 año, la empresa ITAC S.A se beneficiará con el nuevo prototipo de agente que representa un avance frente al producto WS-Guardian, disminuirá costos, tiempo y desarrollo; además, este nuevo producto se complementará con el producto ya existente. El proyecto brinda respaldo a la misión de la compañía que es *"Generar e implementar soluciones informáticas efectivas para satisfacer las necesidades de nuestros clientes"* [17], dado que al ofrecer el agente a sus clientes, se resuelve la problemática de facilitar la implementación de la seguridad en arquitecturas SOA.

- **Mediano plazo**

En un periodo de entre 1 y 3 años, el impacto será significativo, beneficiará a la empresa ITAC S.A y a sus Clientes. Una vez completado el desarrollo y su integración con WS-Guardian, la empresa podrá hacer un despliegue del producto (*WS-Guardian Client's Agent*), WSG contará con mayor aceptación por parte de sus compradores, ya que con el uso del agente no deberán preocuparse por el desarrollo de la seguridad a la hora de acceder a los servicios protegidos por WS-Guardian.

- **Largo plazo**

En un periodo de 3 años en adelante, el agente será utilizado por grandes empresas que usen WS-Guardian. También se espera que, como los clientes no van a tener que realizar cambios en sus servicios al comprar WSG, este producto lo utilicen miles de compañías, ya que *WS-Guardian Client's Agent* implementará seguridad sobre el consumo de SW de forma genérica, traerá consigo nuevas ideas de software por parte de compañías pertenecientes al campo de la seguridad y hará de los SW un tema en el cual la seguridad se garantice por su fácil implementación.

3 Proceso

El presente trabajo de grado utiliza la metodología RUP (Rational Unified Process), que es un producto de proceso desarrollado y mantenido por Rational® Software, Propiedad de IBM [18].

Se toma las mejores prácticas de RUP y se combina con la metodología ágil SCRUM, un marco de trabajo para abordar problemas complejos y reaccionar al cambio. SCRUM se define por los equipos y sus roles, eventos, artefactos y reglas asociadas para un producto exitoso [19].

Basados en RUP, a continuación, se describen cada una de las fases.

3.1 Fase inicial

En esta fase se establece el caso de negocio para el sistema y se delimita el alcance del proyecto [5].

3.1.1 Método

Se utiliza la primera fase de la metodología RUP en esta se establece objetivos y alcance del proyecto, de la metodología SCRUM se establece los *Sprint* correspondientes y el *Product Backlog* [6].

3.1.2 Actividades

Lista de actividades a realizar en esta fase:

- Planteamiento del problema
- Conceptualización
- Definición de la propuesta de trabajo
- Plan de proyecto (PMP)
- Establecer Product Backlog inicial
- Un glosario inicial del proyecto

3.1.3 Resultados esperados

Al final de esta fase, se espera tener concordancia entre los interesados en la definición del alcance y cronograma, comprensión de los requisitos como lo demuestra la fidelidad de los casos de uso principales, comprensión de los requisitos y funcionalidades principales, junto con la credibilidad de las estimaciones de cronograma [18].

El entregable de esta fase es el documento PMP presente en la sección 1.3.

3.2 Fase elaboración

El propósito de la fase de elaboración es analizar el dominio del problema, establecer una base arquitectónica sólida, desarrollar el plan del proyecto y eliminar los elementos de mayor riesgo del proyecto [18].

3.2.1 Método

Se combinan las iteraciones de RUP con los *Sprint* de SCRUM y se utiliza el *Product Backlog* para listar todas las características, funciones, requisitos, mejoras y correcciones. A partir de ello, se realiza una planeación del *Sprint*,

definiendo el *Sprint Backlog*, con elementos seleccionados del *Product Backlog*, más un plan para entregar el incremento del producto y lograr el objetivo del *Sprint* [6].

3.2.2 Actividades

Lista de actividades a realizar en esta fase:

- Establecer *Product Backlog* con funcionalidades bien definidas
- Especificación de requisitos del sistema (SRS)
- Diseño de la arquitectura (SDD)
- Un prototipo arquitectónico ejecutable

3.2.3 Resultados esperados

Una vez terminada esta fase la visión y la arquitectura del agente debe tener claridad y contar con un plan lo suficientemente detallado; a su vez las partes interesadas deben estar de acuerdo[18].

El entregable de esta fase es una versión parcial de los documentos SRS, SDD mencionados en la sección 1.3.

3.3 Fase construcción

Todos los componentes restantes y las características de la aplicación se desarrollan e integran en el producto, y todas las características se prueban exhaustivamente. Se llevará a cabo el desarrollo del código del proyecto, debido a que se está utilizando la metodología SCRUM por cada una de las funcionalidades que se vayan completando se llevará a cabo el proceso para la realización de las pruebas, este proceso establece tres partes Especificaciones de pruebas (Objetivos, Alcance, Procesos, Responsables, Productos, Técnica, Herramientas), Gestión de Pruebas (plan de pruebas, monitoreo y control, terminación) y Pruebas dinámicas (diseño e implementación, establecer entorno, ejecutar, reporte) [18], [11].

3.3.1 Método

Se combinan las iteraciones de RUP con los sprint de SCRUM, se utiliza el product backlog para listar todas las características, funciones, requisitos, mejoras y correcciones. A partir de ello se realiza una planeación del *sprint*, definiendo el *Sprint Backlog*, con elementos seleccionados del *Product Backlog*, más un plan para entregar el incremento del producto y lograr el objetivo del *Sprint* [6]. Para el desarrollo del agente se define un repositorio para el grupo, y se definen ramas específicas para desarrollo y pruebas .

3.3.2 Actividades

Lista de actividades a realizar en esta fase:

- Documentación de pruebas.
- SRS.
- SDD.
- El prototipo de software integrado en las plataformas adecuadas.
- Una descripción de la versión actual.
- Código fuente.

3.3.3 Resultados esperados

Al final de esta fase se espera una versión del producto estable, lo suficientemente madura y con dos políticas de seguridad [18].

El entregable de esta fase está compuesto por el prototipo funcional, código fuente y la versión final de SRS y SDD, mencionados en la sección 1.3.

3.4 Fase validación y cierre

El propósito de la fase es la transición del producto de software a la comunidad de usuarios [18]. En esta se llevará a cabo el uso del estándar de seguridad OWASP ASVS V2 para verificar que el producto entregado cumpla con un nivel de seguridad alto debido a los usuarios que lo usarán [12].

3.4.1 Método

Se combinan las iteraciones de RUP con los *sprint* de SCRUM, se utiliza el *product backlog* para listar todas las características, funciones, requisitos, mejoras y correcciones. A partir de ello se realiza una planeación del *sprint*, definiendo el *Sprint Backlog*, con elementos seleccionados del *Product Backlog*, más un plan para entregar el incremento del producto y lograr el objetivo del *Sprint*, se establecen *Sprint* basados en el proceso de pruebas Especificaciones de pruebas, Gestión de Pruebas y Pruebas dinámicas [6],[11].

3.4.2 Actividades

Lista de actividades a realizar en esta fase:

- Validación del prototipo.
- Los manuales de usuario.
- Documentación de pruebas (incluido OWASP).
- Postmortem.
- Cierre del proceso.

3.4.3 Resultados esperados

Al terminar esta fase se espera lograr la concordancia de las partes interesadas junto a la implementación completa y consistente de los criterios de evaluación [18].

El entregable de esta fase es informe y documento de pruebas presente en la sección 1.3 .

4 Aspectos generales del proyecto

4.1 Derechos patrimoniales

4.1.1. Propiedad intelectual

Con la suscripción del presente documento, LAS PARTES convienen y aceptan que:

i) La titularidad de la propiedad intelectual sobre los resultados que se obtengan o se pudieran obtener en el desarrollo del presente proyecto / programa estará a cargo de ambas partes.

ii) Ambas partes tendrán los derechos patrimoniales sobre todos y cada uno de los entregables generados.

iii) De igual manera cualquiera de las partes podrá iniciar los mecanismos de protección correspondientes garantizando el respeto que a cada una de las partes corresponden sobre los resultados.

iv) La custodia y cuidado de los productos tecnológicos, prototipos, que se materialicen estarán a cargo de ITAC durante la duración del proyecto.

v) Los derechos morales de autor que le correspondan a estudiantes, profesores o investigadores de las partes, que por sus aportes significativos en una determinada obra le corresponden como autor(es) o coautor(es), serán a estos siempre reconocidos.

vi) Las partes tendrán el derecho a perpetuidad para el uso del producto desarrollado por el proyecto.

vii) Los componentes generados dentro del marco del Proyecto, serán de propiedad de las partes, y cada uno de ellos podrá hacer uso de éstos exclusivamente para la ejecución de su objeto social.

viii) Sin perjuicio de lo anterior las partes podrán efectuar modificaciones al presente documento por mutuo acuerdo de acuerdo con las condiciones de desarrollo del proyecto.

4.1.2. Confidencialidad

Las partes se obligan recíprocamente a no divulgar y mantener bajo reserva, en calidad de información confidencial, la información que sea suministrada o que conozca directamente, en desarrollo del objeto del presente proyecto de grado, a no darle destinación diferente a la requerida para la ejecución del mismo, y en general, a los términos establecidos en el *Anexo 1 - Acuerdo de Confidencialidad*.

4.1.3. Carta de compromiso

La empresa ITAC se compromete a proporcionar a los estudiantes los recursos mencionados en el *Anexo 2 - carta de compromiso con ITAC*, documento que el product owner y los integrantes del grupo han leído, aceptado y firmado previamente.

5 Marco teórico

5.1 Fundamentos y conceptos relevantes para el proyecto

5.1.1 ITAC

La compañía ITAC IT APPLICATIONS CONSULTING S.A es una empresa fundada en el año 2004, líder en productos y servicios usando tecnología Java, dedicada a desarrollar soluciones que permiten intercambio seguro de información, brinda asesoría a sus clientes en la adopción de estrategias de seguridad y tecnologías, cuenta con más de 60 certificaciones de industria sobre la tecnología Java y es reconocida como fábrica de software seguro valorada en CMMI Nivel 5 [17], [20].

En el año 2009 la compañía lanza el producto de seguridad WS-Guardian, plataforma de Seguridad y Gobernabilidad SOA que permite administrar el ciclo de vida de los servicios y habilitar mecanismos de seguridad para la exposición y consumo de estos, es allí donde surge la problemática a tratar [3]**.**

5.1.2 WS-Guardian

Es un servidor de capa de aplicación diseñado para arquitecturas SOA, funciona sobre protocolos http y https, este software se encuentra en un punto central, a él llegan mensajes de consumidores y prestadores de servicios, se establecen políticas de seguridad las cuales son acordadas con los consumidores y prestadores, cada vez que se envía un mensaje éste debe pasar por WS-Guardian para verificar el cumplimiento de las políticas establecidas, en caso de que algunas políticas requieran manejo de llaves, el almacenamiento se hace dentro del software, el paso por WS-Guardian es transparente al cliente, debido a que el consumidor de servicios crea el mensaje con las políticas de seguridad y lo envía a WS-Guardian como si él fuera quien ofrece el servicio, la respuesta funciona de manera similar el consumidor recibe el mensaje como si el prestador del servicio fuera WS-Guardian [3]**.**

5.1.3 Seguridad Informática

Según la Real Academia Española (RAE), el concepto "seguro" se define como *"libre, y exento de riesgo"*. La seguridad informática se enfoca en la infraestructura computacional y todo lo relacionado con esta, se encarga de diseñar las normas, procedimientos, métodos y técnicas destinadas a conseguir un sistema de información seguro y confiable [25].

Existen dos tipos de seguridad informática, la seguridad activa y la seguridad pasiva. La seguridad activa tiene como objetivo evitar o reducir el riesgo que amenaza al sistema comprendiendo conjuntos de defensas. La seguridad pasiva tiene como objetivo de minimizar y facilitar la recuperación del sistema, como por ejemplo la administración de copias de seguridad de los datos [25].

5.1.3.1 Principios de Seguridad

Existen principios básicos que se deben satisfacer para garantizar la seguridad del software y su comunicación. A continuación, se mencionan aquellos que son clave para *WS-Guardian Client's Agent*.

Confidencialidad

Según la Organización para la Cooperación y el Desarrollo Económicos (OCDE), la confidencialidad en términos de seguridad informática es *"el hecho de que los datos o información estén únicamente al alcance del*

conocimiento de las personas, entidades o mecanismos autorizados, en los momentos autorizados y de una manera autorizada" [25].

Para satisfacer este principio, se debe diseñar el sistema de tal manera que se realice un control de acceso teniendo en cuenta el servicio implicado, quién lo solicita y cuándo puede ser accedido.

Integridad

La integridad garantiza la autenticidad y precisión de la información sin importar el momento en que se solicita. Esto significa que se debe asegurar que los datos no sean alterados ni destruidos de manera no autorizada [25].

No repudio

Se garantiza una comunicación en donde la persona que envía un mensaje es quien dice ser (no repudio en el origen), y el receptor también es quien dice ser (no repudio en el destino). De esta forma, cualquiera de las partes no puede retractarse de haber enviado o recibido el mensaje respectivamente [26].

Autenticación

Asegura que la persona o máquina que intenta acceder a algún servicio o recurso del sistema es quien dice ser [26].

5.1.3.2 Políticas de Seguridad

En la seguridad de la información existen múltiples políticas que se aplican para poder garantizar los principios de seguridad (algunos mencionados anteriormente). A continuación, se nombran 4 políticas que se manejan dentro del software *WS-Guardian* y se relacionan con *WS-Guardian Client's Agent*.

Timestamp

Es un mecanismo que permite demostrar que los datos transmitidos existieron en un momento determinado y que no han sido alterados. Cumple con la normativa de la ISO 8601 para estandarizar las fechas y horas evitando ambigüedades según la ubicación geográfica o configuración regional de las máquinas [27].

El timestamp se define como una marca temporal, esto significa que cuando se genera uno de ellos, se inicia un conteo de tiempo hasta su caducidad. Esto permite que los atacantes tengan un tiempo limitado para intentar acceder a la información de la cual no tienen permisos [28].

Username token

Permite generar un nombre de usuario y contraseña (según el caso), para ser utilizado al momento de acceder a algún servicio del sistema. Contiene un timestamp para que una vez se cumpla su tiempo de vida, el token no sea válido como credencial de acceso [29].

A continuación, se observa el flujo básico para la generación de un usernametoken [30].

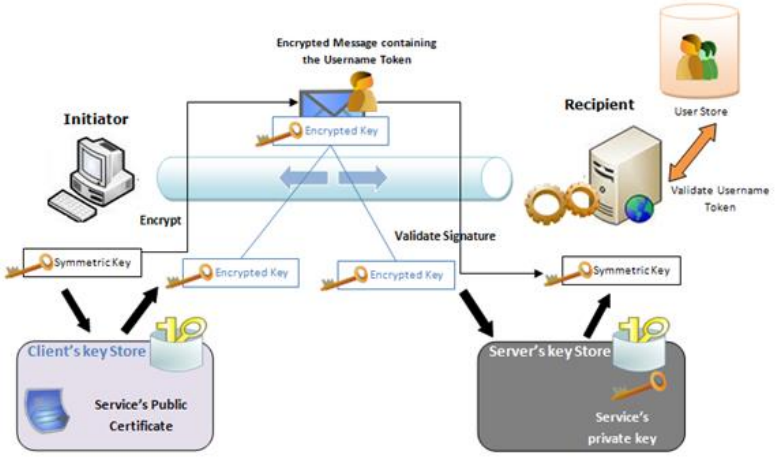

Figura 3: Flujo para utilizar username token [30]

1. El cliente genera una clave simétrica y cifra el mensaje con esa clave.
2. Se cifra la clave simétrica con la clave pública del servicio y se envía dentro del mensaje SOAP.
3. El servidor descifra la clave simétrica utilizando su clave privada.
4. Descifra el mensaje con la clave simétrica obtenida en el paso 3.
5. El usuario envía el username token dentro del mensaje.
6. Se valida lo anterior teniendo en cuenta las condiciones como el timestamp.

Firma digital

Es un método criptográfico generalmente asimétrico obtenido mediante un algoritmo matemático, asocia la identidad del emisor con un mensaje enviado a través de la red, para así evitar la falsificación o suplantación de identidad. Esto asegura especialmente el principio de no repudio [25].

En la siguiente imagen (*Figura 8: Flujo para enviar firma digital*) se puede observar el flujo para enviar un mensaje junto con una firma digital [31].

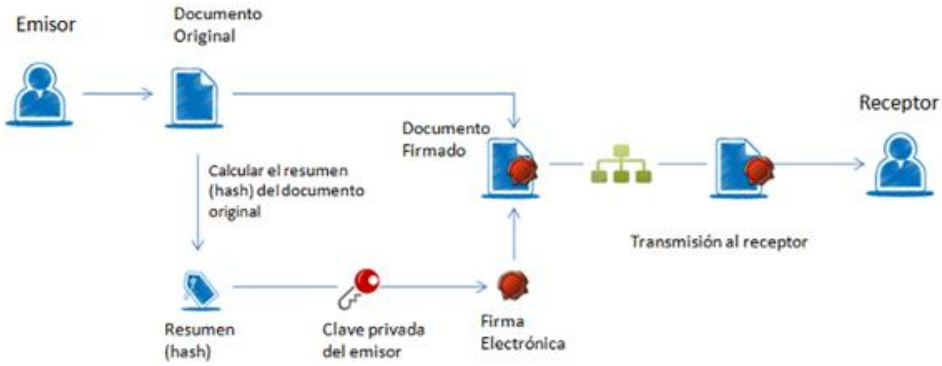

Figura 4: Flujo para enviar firma digital [31]

1. El emisor cuando crea el mensaje calcula el hash de este.
2. Utiliza la clave privada del emisor para firmar el hash.
3. Se envía el mensaje junto con la firma digital al receptor.
4. El receptor descifra el hash del mensaje con la llave pública del emisor.
5. El receptor calcula y compara el hash del mensaje con el hash descifrado en el paso 4.

Certificado digital

Son documentos digitales mediante los cuales una entidad autorizada garantiza la técnica y que el emisor es quien dice ser, respaldado por la verificación de su clave pública. Protege la integridad y la confidencialidad de la información [25].

Existen dos tipos fundamentales de certificados, el certificado electrónico en donde el documento es firmado electrónicamente por un prestador de servicios de certificación que vincula datos de verificación de firma al documento. El segundo es el certificado reconocido, es un certificado electrónico que cumple con los requisitos de la Ley de Firma Electrónica en cuanto a su contenido, como en condiciones para el prestador de servicios de certificación [32].

Se debe tener en cuenta que algunos certificados son válidos dependiendo el país o región en que se encuentre.

Se han definido varios formatos para los certificados digitales como [33]:

- X.509: Estándar UIT-T de infraestructura de claves públicas, contiene formatos para certificados de claves públicas y algoritmos de validación.
- PGP: Programa que tiene como objetivo proteger la información distribuida a través de internet mediante el uso de criptografía pública.
- SAML: Estándar abierto de esquema XML para el intercambio de datos de autenticación y autorización.

5.1.4 Web Services

Un servicio web es un módulo de software autodescriptivo, autocontenido disponible en la red. El cual sirve como ayuda para completar tareas, resolver problemas o comportamientos de transacciones en nombre de un usuario o aplicación [23].

Un servicio web provee funciones lógicas que son utilizadas independiente del dispositivo, de la red, del acceso del usuario, cuando y donde sea es necesario. El objetivo a largo plazo de la tecnología web service es permitir aplicaciones distribuidas que puedan ser dinámicamente ensambladas de acuerdo con las necesidades del negocio [23].

Al construir software sobre las arquitecturas tradicionales, la reutilización del servicio es restringido a la misma tecnología, un servicio web se centra en esto para facilitar la integración con otras aplicaciónes, motivado por el deseo de nuevas formas de compartir servicios, debido a diferentes escenarios de interacciones entre departamentos, o entre socios de negocios. [23]

Características de un servicio web: tipos de servicios web, propiedades funcionales y no funcionales, pérdida de acoplamiento, granularidad, sincronización, buena definición, contexto de uso y estado [23].

5.1.4.1 Arquitectura orienta a servicios

Mejor conocido como SOA siglas en inglés de *Service Oriented Architecture*, es una forma lógica de diseñar un sistema de software para proveer servicios ya sea para aplicaciones finales de usuario o para otros servicios distribuidos en una red. Para lograr esto se reorganiza un portafolio de aplicaciones independientes que deben interactuar entre sí, y con apoyo de la infraestructura de una organización crea una colección interconectada de servicios fáciles de usar. Cada uno de estos servicios es encontrado y accesible mediante un estándar de interfaz y protocolo del mensaje. [23]

SOA es un estilo arquitectónico para dar interoperabilidad extensible orientado a servicios, cuando múltiples aplicaciones corren en una variedad de tecnologías y plataformas, convirtiendo los sistemas monolíticos y estáticos en componentes modulares y flexibles, que pueden ser solicitados mediante un protocolo estándar. [23]

Roles de interacción en SOA: Proveedor del servicio web, consumidor del servicio web, y un registrador de servicio web. [23]

Operaciones SOA: Publicar, encontrar y enlazar. [23]

5.1.4.2 Pila de tecnología

Una meta de la tecnología es permitir a las aplicaciones trabajar juntas sobre los protocolos estándares de internet, sin una intervención humana directa. La mínima infraestructura necesaria para el paradigma de servicios web es la existente para una red de computadoras, con esto se puede asegurar que el servicio web pueda ser implementado, y accedido, desde cualquier plataforma usando cualquier tecnología y un lenguaje de programación. [23]

La *Figura 5 The Web service technology stack* muestra la definición general aceptada de un web service que se apoya en estándares complementarios, resultado de la proliferación de un vertiginoso número de estándares y siglas.

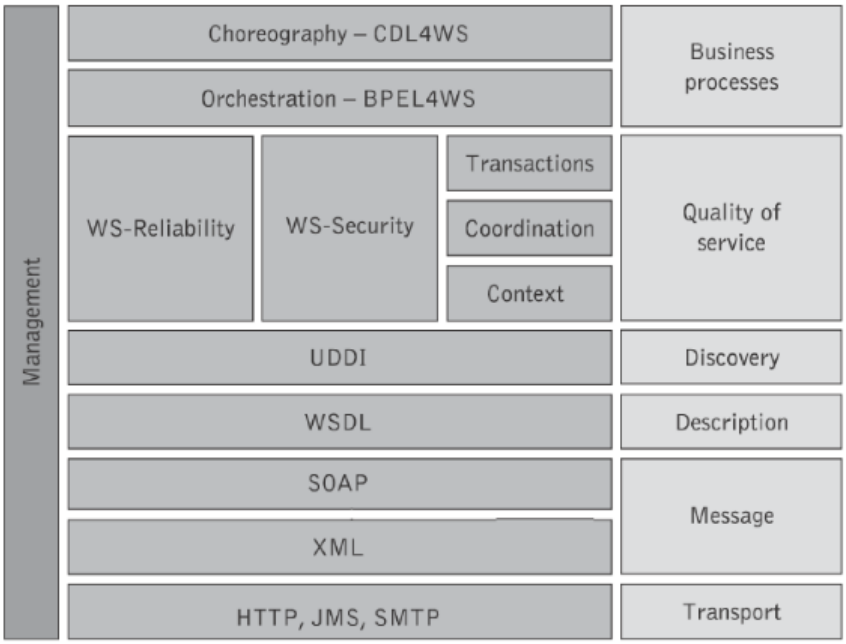

Figura 5. The Web service technology stack [23]

Los principales estándares de servicios web [23] son:

- **Protocolos de comunicación:** El protocolo SOAP implementa un modelo de petición/respuesta para la comunicación entre servicios web, y usa HTTP para su transporte.
- **Descripción del servicio:** Para ser usado correctamente un servicio web y su cliente, se deben conocer los datos y operaciones, indicando un contrato de un servicio web que representa una funcionalidad que el servicio web provee. Para lograr esto se describen las características de un servicio web usando un lenguaje de descripción de servicio web, conocido como WSDL. Este define la gramática en XML para describir los servicios como colección de puntos finales capaces de intercambiar mensajes.
- **Publicación del servicio:** La publicación de un servicio web se logra usando el registro *Universal Description, Discovery and Integrati*on, UDDI, es un directorio público que provee la publicación de servicios en línea y facilita su descubrimiento.
- **Composición del servicio:** Esta familia de estándares describe la ejecución lógica de un servicio web basado en aplicaciones, controla el flujo y prescribe las reglas para la administración de los datos del negocio.
- **Colaboración del servicio:** Este estándar describe la colaboración de servicios web entre compañías.
- **Estándares de colaboración y transacción:** *WS-Coordination* y *WS-Transaction* proveen mecanismos para definir protocolos estándares para ser usados en sistemas de procesamiento de transacciones, sistemas de flujo de trabajo u otras aplicaciones que deben coordinar múltiples servicios web.
- **Estándares para añadir valores:** Elementos adicionales que apoyan la interacción de negocios complejos, deben ser implementados antes de ocurrir un proceso de negocio crítico. Los estándares para añadir valores incluyen mecanismos de seguridad y autenticación, confiabilidad, privacidad, conversaciones seguras, etc. Entre estos estándares se encuentran *WS-Security, WS-Policy* y *WS-Management*.

RESTful services

Representational State Transfer (REST) es un estilo de arquitectura para sistemas de hipermedia tal como *World Wide Web*. En este estilo arquitectónico los servicios web son vistos como recursos y pueden ser identificados únicamente por sus URLs. La característica principal de un servicio web *RESTful* es el explícito uso de los métodos de HTTP para marcar la invocación de diferentes operaciones. [23]

La aparición del estilo *RESTful* para servicios web fue una reacción por la sobrecarga de los estándares basados en SOAP. Los servicios web *RESTful* son simples, claros y proveen un estilo arquitectónico, el cual no requiere de un conjunto complejo de estándares, se enfatiza en una comunicación punto a punto sobre HTTP usando XML plano fácil de consumir. [23]

5.1.4.3 Estándares de seguridad en Servicios Web

Web service Interoperability

La Organización de Interoperabilidad de Servicios Web (WS-I) es una organización de industria abierta creada para establecer las mejores prácticas para la interoperabilidad de servicios web, para grupos seleccionados de estándares de servicios web, en todas las plataformas, sistemas operativos y lenguajes de programación. [24]

Versiones de las marcas de WSI de las especificaciones de servicios web se definen como perfiles interoperables. Los perfiles interoperables identifican un objetivo de las tecnologías web service y proveen aclaraciones sobre el uso individual y en conjunto. Los *WS-I Profiles* contienen una lista de nombres y versiones de especificaciones de servicios web, junto con un conjunto de implementaciones y guías para desarrollar servicios web interoperables.

WS-I Basic Profile 1.0 incluye guías de implementación usadas en el núcleo de servicios web, son convenciones alrededor del mensaje, descripción y descubrimiento. Estas especificaciones incluyen SOAP 1.1, WSDL 1.1, UDDU 2.0, XML 1.0, y xml schema. La versión 1.0 de este perfil intenta proveer un marco de trabajo para soluciones interoperables dando un punto de referencia a los ingenieros. Mientras tanto WS-I en noviembre del 2010 publicó la v 2.0 del Basic Profile usando SOAP 1.2, UDDI 3 y WSAddressing . [23]

5.1.4.3.1 Seguridad en Servicios Web

Surge una preocupación de la seguridad en la comunicación de extremo a extremo y las bases de seguridad para el servicio web, IBM y Microsoft tienen que unirse para escribir el mapa de la seguridad de los servicios web, como un significado para desarrollar un conjunto de especificaciones, tecnologías y estándares de seguridad para servicios web. Estos estándares [23] se explican brevemente a continuación.

- **WS-security:** Conjunto de extensiones SOAP enfocado en la integridad y confidencialidad del contenido del mensaje, Los mecanismos especificados por este estándar pueden ser usados para el acomodo de la variedad de modelos de seguridad y tecnologías de seguridad.
- **WS-SecurityPolicy:** Es una parte de *WS-Security* e indica las políticas de defensa para *WS-Policy* que se aplican a WS-Security. Especifican las políticas de defensa de seguridad como requisitos de un servicio web. Estos requisitos incluyen algoritmos de soporte para encriptación y firmas digitales, atributos privados, y cómo la información debe ser transmitida para un servicio web
- **WS-Trust:** Define una serie de primitivas basadas en XML para solicitar y emitir tokens de seguridad, así como para gestionar relaciones de confianza.
- **WS-SecureConversation:** Define la extensión que se basa en *ws-security* para proporcionar una comunicación segura.
- **WS-Federation:** Define mecanismos que son usados para habilitar la verificación de identidad, atributo, autenticación, y autorización a través de diferentes ámbitos de confianza. Este estándar también incluye políticas de privacidad que son establecidos por organizaciones que despliegan servicios web y requieren indicarle al consumidor las restricciones del servicio para usar y distribuir información sensible, contenida dentro del token.

5.1.4.3.2 Estándares de seguridad XML

XML y servicios web basados en SOA facilitan la integración de negocios dentro y a través de los límites organizacionales. Sin embargo, estos beneficios tienen un precio, la seguridad, los sistemas de seguridad también deben ser integrados [23].

Las tecnologías de servicios web usan mensajes basados en XML sobre protocolos basados en internet para interactuar con otras aplicaciones. Para lograr esto, se confía en *XML Trust services,* lo cual es una suite de especificaciones abierta de XML para desarrolladores de aplicaciones, para hacer más fácil integrar un amplio rango de servicios de seguridad en XML dentro aplicaciones de decisión integrados sobre la web [23].

Las principales tecnologías para XML Trust Services [23] incluyen:

- XML *Signature* para autenticación criptográfica de datos.
- XML *Encryption* para encriptación datos.
- XML *Key Management specification* (XKMS) estándar para facilitar el uso de una llave pública.

- Security *Assertions Markup Language* (SAML) especifica un estándar que define un esquema XML para el intercambio de datos de autenticación y autorización
- XML *Access Control Markup Language* (XACML) describe cómo evaluar peticiones de acceso de acuerdo a las políticas establecidas en un servicio web.

A continuación, se describen los estándares para firmas digitales y encriptación en XML.

XML Signature

El objetivo de *XML Signature* es asegurar la integridad de los datos, autenticación del mensaje y no repudio de servicios. El estándar XML *signature* define un esquema para capturar el resultado de una operación de firma digital aplicada a contenidos digitales arbitrarios a través de un direccionamiento indirecto. Los datos son aceptados y firmados criptográficamente. En sí mismo XML *signature* generalmente indicará la localización original del objeto firmado. XML *signature* puede firmar más de un tipo de recurso como datos codificados en caracteres (HTML), binario (JPG), XML [23] .

Existen tres tipos de firmas [23]:

- Firmas envolventes, donde la firma envuelve todo el documento.
- Firmas envueltas, donde el XML *signature* se utiliza para firmar parte del documento.
- Firmas separadas, donde el documento XML y la firma residen independiente y el documento es usualmente referenciando por un URI externa. Una firma separada significa que la firma de los datos no es la firma del elemento.

XML encryption

XML *signature* no define ningún mecanismo estándar para la inscripción de entidades XML, el cual es otra característica importante de seguridad que promueve la confianza del uso de aplicaciones web. Esta funcionalidad es provista por las especificaciones de XML *Encryption*, un esfuerzo de W3C, el cual soporta la encriptación de todo o una parte del documento XML. XML *encryption* no es bloquear un esquema específico de encriptación, sino que requiere información adicional para proveer un contenido encriptado y una llave, a través de elementos que contienen o hacen referencia a los datos cifrados [23].

Los pasos [23] para incluir XML *Encryption* son:

- Seleccionar el documento XML para ser encriptado.
- Convertir el documento XML para ser encriptado para una forma canónica, si es necesario.
- Encriptar el resultado de la forma canónica, usando llave pública de encriptación.
- Enviar el documento XML encriptado al destinatario deseado.

5.1.5 Sistemas Distribuidos

Los sistemas distribuidos surgen del uso de distintas redes de comunicación como: Local Area Network (LAN), Metropolitan Area Network (MAN), y Wide Area Network (WAN) que son usadas actualmente para acceder a internet. Por ende, el acceso a las aplicaciones desplegadas por internet aumenta haciendo que se desplieguen arquitecturas de cómputo donde se realice el procesamiento distribuido de tareas. Esto, en busca de lograr una mayor eficiencia a la hora de comunicarse con otros sistemas y procesar información [21].

Un sistema distribuido se entiende como una colección de computadoras independientes que se muestran ante un usuario como una única computadora [21].

Los sistemas distribuidos al momento de diseñarse se ha de centrar en 9 aspectos importantes y característicos de este [21] como lo son:

- Transparencia: Hace referencia a la capacidad de ocultar al usuario la manera en que el sistema funciona o está construido. Se busca lograr transparencia en aspectos como: *localización* (donde están los datos), *réplica* (copias existentes), *concurrencia* (acceso concurrente a un mismo recurso), *paralelismo* (procesamiento paralelo), *fallas, desempeño* (funcionamiento y velocidad de las máquinas) *y escalabilidad* (incorporación de máquinas).
- Flexibilidad: No debe tener una estructura rígida y debe facilitar la modificación del diseño inicial[21].
- Confiabilidad: En caso de que una máquina falle el servicio puede continuar por medio de la sustitución realizada por otra máquina.
- Desempeño: Tiempos de respuesta de la aplicación.
- Escalabilidad: Adición de más máquinas para aumentar el poder de cómputo.
- Repartición de carga: Análisis de los equipos pertenecientes al sistema (Procesamiento, RAM, ROM ...) para realizar asignación de tareas y no someter a *overload*.
- Mantenimiento de consistencia: Mantener consistencia en el sistema de: modificaciones, fallas, replicación, relojes entre otros.
- Funcionalidad: Que el sistema sea capaz de cumplir con su propósito y lo haga de una manera más eficiente frente a un sistema centralizado.
- Seguridad: Aplicar las medidas de seguridad para proteger la información que se expone por la red y en las máquinas del ambiente distribuido.

Los sistemas distribuidos se pueden llegar a clasificar en 3 tipos de taxonomías[21], como lo son:

- Sistemas con software débilmente acoplado en hardware débilmente acoplado
- Sistemas con software fuertemente acoplado en hardware fuertemente acoplado
- Sistemas con software fuertemente acoplado en hardware débilmente acoplado

5.1.5.1 Agente

Se entiende como agente al software que se sitúa en un entorno y realiza acciones flexibles de manera autónoma buscando la realización de un objetivo. Por lo cual, el agente sólo actúa por estímulos del entorno externo y realiza una acción, comunicándose con otros sistemas y a su vez contando con criterios de decisión, comunicación, adquisición (Adquirir conocimiento) para lograr cumplir su objetivo[22]. Por lo cual, un agente debe contar con características como:

- Autonomía: Cuando el agente se cree con la información necesaria, él debe actuar autónomamente en el *background*, y a su vez saber cómo reaccionar ante eventos guardando un estado.
- Reactividad: Dar respuesta a eventos.

- Habilidad "Social": Comunicarse con otros sistemas, agentes o usuarios.

5.1.5.1.1 Taxonomía de Agentes

Los agentes se pueden clasificar según su forma de actuar y cómo están compuestos. Existen tres tipos: agente reactivo, agente cognitivo o deliberado y agente híbrido.

Agente reactivo

Actúan de manera estimulo - respuesta, esta arquitectura cuenta con una serie de capas que se especializan en los diferentes eventos que surgen en el entorno[22]. Existen dos tipos de estructuras de capas:

Arquitectura horizontal: Todas las capas tienen acceso a los sensores.

Arquitectura vertical: La salida de una capa es la entrada de otra.

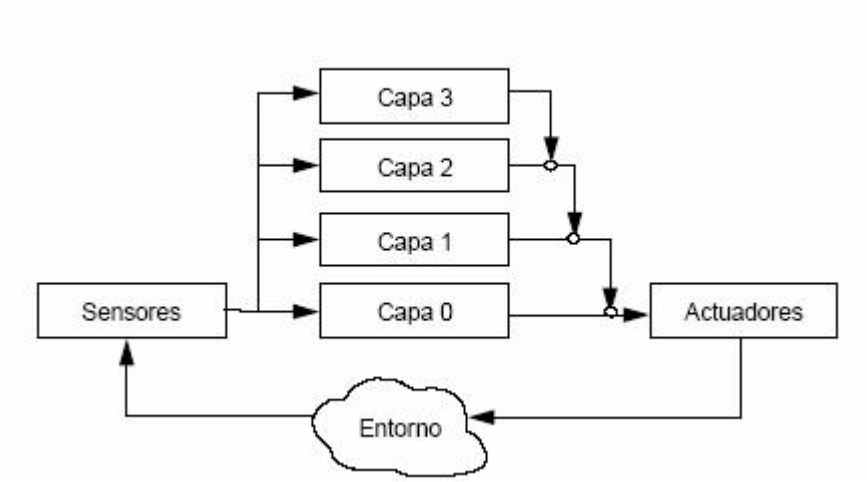

Figura 6. Agentes Reactivos[22]

Agente cognitivo o deliberado

Manejan un enfoque intencional, con base a un modelo BDI (*Believes, Desires, Intentions*) actuando con base a este [22].

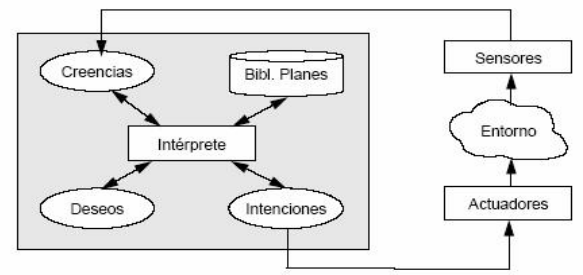

Figura 7. Agentes Cognitivos[22]

Agente híbrido

Poseen características de los anteriores tipos, contando con subsistemas de percepción y acción, con tres capas de control[22], estas son:

- Capa Reactiva: Capta los eventos del entorno que han de ser procesados y se los transmite a las otras capas.
- Capa de planificación: Planifica las acciones que se realizarán.
- Capa de modelado: Contiene información sobre sobre el estado de otras entidades

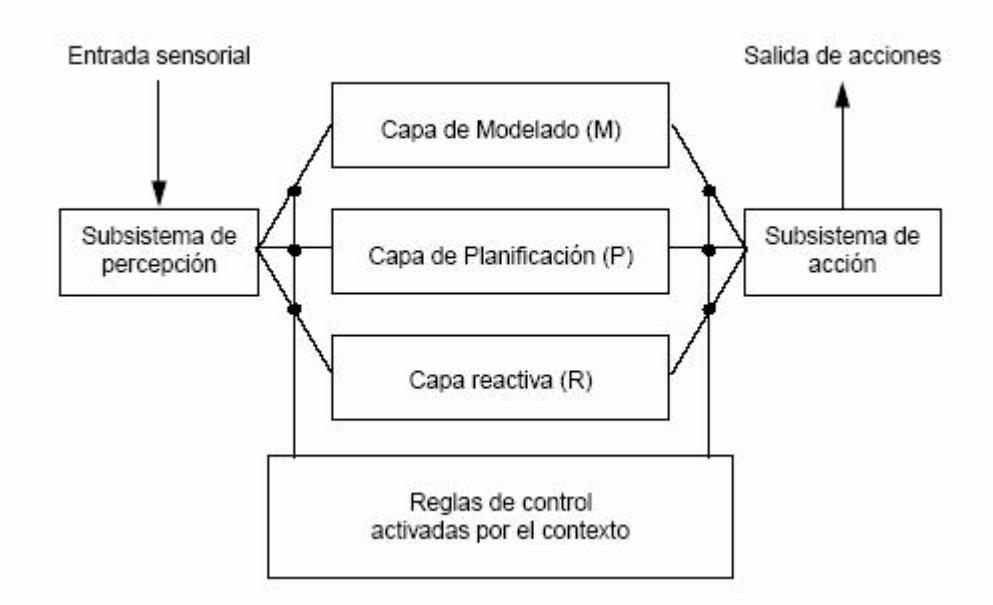

Figura 8. Agentes Híbridos [22]

5.2 Trabajos importantes en el área

5.2.1. Tabla de Similitud

En la *Tabla 3 Similitudes de los trabajos importantes*, se muestra la relación que existe entre WS-Guardian Client's Agent y otros proyectos, con respecto a la problemática o solución planteada.

	Problema distinto	**Problema similar**	**El mismo problema**
Solución diferente			• WebSphere Application Server • WebLogic • Encryption Wizard
Solución similar	• HostwapAgent • Byte Buddy	• MyDiamo	• Oracle Web Services Manager • Digital Guardian Data Cloud Protection
La misma solución			

Tabla 3: Similitudes de los trabajos importantes

5.2.2. Trabajos Relacionados

HostwapAgent

HostwapAgent es un proyecto enfocado a evitar el tedioso proceso de implementar cambios sobre el código fuente, compilarlo y volver a desplegar el aplicativo, para modificar la lógica de este. *HostwapAgent* permite realizar el CRUD (*Create*, *Read*, *Update* and *Delete*) sobre los campos, métodos, anotaciones y miembros estáticos de las clases, junto a la carga de las mismas en tiempo de ejecución. Cabe aclarar que esta herramienta no admite el cambio de jerarquías (variaciones en superclases o eliminar una interfaz). Dichas ediciones se hacen a nivel de marcos de Java y contenedores de Servlet, funcionando como un sistema de complementos preconfigurados para la edición [34].

Los *plugin* de *HostwapAgent* se limitan a versiones JDK 7 y superiores. Adicionalmente, es necesario combinarlo con DCEVM (*Dynamic Code Evolution Virtual Machine*) [35] para realizar cambios a nivel de la máquina virtual de Java (JVM) [34].

Byte Buddy

Es una biblioteca de generación y manipulación de código, para crear y modificar clases de Java durante el tiempo de ejecución de una aplicación Java, sin la ayuda de un compilador. Permite la inserción de clases

arbitrarias, sin limitación en implementar interfaces para la creación de *proxies runtime*. Este recurso está escrito en JDK 5 pero admite la generación de clases en cualquier versión de Java [36].

La solución cambia las clases durante la compilación o usa un Agente Java en tiempo de ejecución. Este producto es de código abierto, pero siendo desarrollado sobre Java solo se puede usar en esta plataforma. *Byte Buddy* es un proyecto de código abierto distribuido bajo la licencia liberal y apache 2.0. Su código fuente está disponible gratuitamente en *GitHub*. Tenga en cuenta que *Byte Buddy* depende de la biblioteca ASM que se distribuye bajo una licencia BSD [36].

WebLogic

WebLogic es un servidor de aplicación Java EE desarrollado por Oracle para alojar sistemas desarrollados en Java. Esta solución aplica políticas de seguridad a nivel de mensaje sobre el tráfico SOAP. Esto es hecho después de que un cliente de servicios web establece una conexión con un servicio de proxy de Bus de servicio de Oracle o un servicio de negocios, y antes de que el servicio de proxy o de negocios procese el mensaje [37].

La seguridad se aplica a los mensajes entre los servicios de proxy de Oracle Service Bus y los servicios empresariales SOAP-HTTP o SOAP-JMS, aplicándola tanto a la solicitud como a la respuesta. Los procesos de aseguramiento son ejecutados por el servidor *WebLogic* antes de la transmisión. Este recurso es adquirido automáticamente al desplegar una aplicación dentro del servidor *WebLogic*, y crear el proxy para el consumo de servicios web. Dicho proxy es el encargado de procesar el sobre, para descifrar el mensaje y verificar que este cumpla con los requerimientos de seguridad como firmas digitales, tokens de seguridad y otras construcciones [37].

Oracle Web Services Manager

Esta solución proporciona un marco de políticas para administrar y asegurar los servicios web de manera consistente en toda una organización. Proporciona capacidades para crear, hacer cumplir, ejecutar y monitorear políticas de servicios web, tales como: seguridad, mensajería confiable, MTOM y políticas de direccionamiento.

OWSM proporciona la seguridad a través de agentes en el cliente para asegurar a los consumidores del servicio por medio de un marco de políticas para administrar y asegurar los servicios web de manera consistente en toda la aplicación. El agente puede ser usado como un componente de la aplicación contenida en *WebLogic Server*, o puede ser desplegado como otra aplicación junto al consumidor en el caso de entornos móviles (Android y IOS). Para el uso de este recurso es necesario usar un mismo OWSM *Agent* en cada intermediario del camino, obligando a usar solamente tecnología Oracle [38].

WebSphere Application Server

WebSphere Application elaborado por IBM, garantiza que los datos privados y confidenciales están fuertemente protegidos y que cumple con las regulaciones y los actos legislativos. El software realiza protección de: datos sólidos para el sistema BD2 (base de datos de IBM), archivos en vivo, configurado de archivos, registro y backup de archivos. También es capaz de realizar filtros según las políticas de seguridad, cumpliendo con los requisitos de rendimiento siendo transparente para aplicaciones, usuarios, bases de datos y entornos de almacenamiento [39].

Este es un producto de software que cumple la función de un servidor de aplicaciones web, siendo un marco de software y middleware que aloja aplicativos desarrollados en Java. WebSphere Application Server aplica a los mensajes de servicios web autenticación, firmas digitales y cifrados XML, para asegurar la información compartida. El proceso se lleva a cabo en el proxy creado para el consumo de un servicio y sobre el proveedor del mismo. La funcionalidad y las políticas de seguridad definidas son agregadas al proxy, en el momento de

establecer la conexión con el servidor. El acceso a este recurso se obtiene desplegando servicios web y consumidores sobre WebSphere Application Server, sin olvidar que solo pueden ser aplicativos Java [40].

Digital Guardan Cloud Data Protection

Producto desarrollado por la empresa Digital Guardian, una compañía que se enfoca a desarrollar software para protección de datos de redes corporativas, servidores, bases de datos y cloud.

Esta aplicación permite cifrar, procesar y firmar digitalmente los archivos confidenciales almacenados en la nube, como Office 365, antes de transferirlos de manera automática. Además, permite analizar y auditar los datos que se almacenan en la nube en cualquier momento [41].

Los productos de Digital Guardian están relacionados entre ellos, lo que significa que Cloud Data Protection tiene conexión a otros productos como Digital Guardian Network DLP que permite aplicar políticas de seguridad, monitorear y controlar el flujo de datos [42].

MyDiamo

Es un software que realiza el cifrado de datos con el objetivo de cubrir las vulnerabilidades de confidencialidad existentes en las bases de datos de código abierto, especialmente con MySQL, MariaDB, Percona y PostgreSQL.

MyDiamo permite cifrar selectivamente segmentos de datos dentro de las bases de datos, cifrar columnas específicas apoyando el rendimiento del sistema y la flexibilidad para que el usuario pueda elegir qué desea cifrar. Además, para hacer uso del producto, no se necesita realizar modificaciones en el código de bases de datos [43].

Encryption Wizard

Es un software portátil desarrollado por TENS que encripta todo tipo de archivos y carpetas basado en Java para la protección de información confidencial a nivel de transporte. Este software puede ser utilizado en diferentes sistemas operativos como: Microsoft Windows, Mac OS X, Linux, Solaris y otros [44].

El software no es transparente para el usuario debido a que se debe configurar los parámetros de políticas y en el momento de utilizarlo, es necesario “arrastrar y soltar” el archivo para encriptar y desencriptar. Como ventaja sobresale su diseño sencillo y fácil de utilizar [45].

5.2.3. Tabla de Comparación

En la *Tabla 4 comparación de los trabajos importantes*, se contrasta WS-Guardian Client’s Agent con otros proyectos respecto a los criterios de seguridad, rendimiento, movilidad, portabilidad y disponibilidad, aspectos importantes a tener en cuenta en el desarrollo de *WS-Guardian Client's Agent*. Para cada criterio se evalúa lo siguiente:

- **Seguridad**: *¿Es seguro?* teniendo en cuenta si el programa realiza encriptación o alguna forma para proteger la información.
- **Rendimiento**: *¿Es rápido?* dependiendo de cuánto tiempo en segundos que tarda realizando una tarea el sistema.
- **Flexible**: *¿Es modificable?* que es flexible, si los cambios en el software deben ser fácil de hacer
- **Portabilidad**: *¿Es movible?* se dice que es movible, si el software funciona con diferentes tipos de hardware y software.

- **Disponibilidad**: *¿Es continuo?* se dice que cuando el software tiene un grado de continuidad operacional alto, tiene una disponibilidad alta.

El cumplimiento de criterios se clasifica de la siguiente manera: (+) Soporta, (-) No soporta, (+/-) Soporta parcialmente y (?) No se puede determinar con la información disponible.

	Seguridad	**Rendimiento**	**Flexible**	**Portabilidad**	**Disponibilidad**
HostwapAgent	?	+	+	+	+
Byte Buddy	?	+	+	+	+
WebSphere Application	+	+/-	-	?	+
WebLogic	+	+/-	-	?	+
Oracle Web Services Manager	+	+/-	-	+/-	+
Digital Guardian Data Cloud Protection	+	+	-	-	+
MyDiamo	+	+	+	?	+
Encryption Wizard	+	+	+	+	+
WS-Guardian Client's Agent	**+**	**+**	**+**	**+/-**	**+**

Tabla 4: Comparación de los trabajos importantes

6 Referencias

[1] «SOA aporta a la Administración Pública un nuevo paradigma para mejorar la interoperabilidad entre sus aplicaciones. Su implantación requiere la aceptación de estándares y normas comunes», *ComputerWorld*, 16-feb-2007. [En línea]. Disponible en: https://www.computerworld.es/archive/soa-aporta-a-la-administracion-publica-un-nuevo-paradigma-para-mejorar-la-interoperabilidad-entre-sus-aplicaciones-su-implantacion-requiere-la-aceptacion-de-estandares-y-normas-comunes. [Accedido: 30-mar-2019].

[2] Packt, «SOA: Implementing Message-Level Security», *Packt Hub*, 30-abr-2010. [En línea]. Disponible en: https://hub.packtpub.com/soa-implementing-message-level-security/. [Accedido: 30-mar-2019].

[3] «Home - wsguardian.co». [En línea]. Disponible en: http://wsguardian.co/home. [Accedido: 30-mar-2019].

[4] «Mecanismos de WS-Security», 24-oct-2014. [En línea]. Disponible en: undefined. [Accedido: 31-mar-2019].

[5] «Reflection & Introspection: Objects Exposed». [En línea]. Disponible en: http://www2.sys-con.com/itsg/virtualcd/Java/archives/0305/sagar2/index.html. [Accedido: 21-abr-2019].

[6] «IEEE 1058.1-1987 - Estándar IEEE para planes de gestión de proyectos de software». [En línea]. Disponible en: https://standards.ieee.org/standard/1058_1-1987.html#Additional. [Accedido: 06-abr-2019].

[7] «BPMN Specification - Business Process Model and Notation». [En línea]. Disponible en: http://www.bpmn.org/. [Accedido: 15-abr-2019].

[8] «Acerca de la especificación del lenguaje de modelado unificado versión 2.5.1». [En línea]. Disponible en: https://www.omg.org/spec/UML/About-UML/. [Accedido: 15-abr-2019].

[9] «IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications». [En línea]. Disponible en: https://standards.ieee.org/standard/830-1998.html. [Accedido: 06-abr-2019].

[10] «IEEE 1016-2009 - IEEE Standard for Information Technology--Systems Design--Software Design Descriptions». [En línea]. Disponible en: https://standards.ieee.org/standard/1016-2009.html. [Accedido: 06-abr-2019].

[11] «ISO/IEC/IEEE 29119 Software Testing». [En línea]. Disponible en: http://softwaretestingstandard.org/. [Accedido: 06-abr-2019].

[12] «Estándar de Verificación de Seguridad en Aplicaciones 3.0.1», p. 75.

[13] «codeconventions-150003.pdf». .

[14] 14:00-17:00, «ISO/IEC/IEEE 12207:2017», *ISO*. [En línea]. Disponible en: http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/37/63712.html. [Accedido: 17-abr-2019].

[15] T. Preston-Werner, «Versionado Semántico 2.0.0-rc.2», *Semantic Versioning*. [En línea]. Disponible en: https://semver.org/lang/es/. [Accedido: 17-abr-2019].

[16] «IEEE Standard for Software User Documentation», *IEEE Std 1063-2001*, pp. 1-24, dic. 2001.

[17] ITAC, «DevOps e Innovación para la Transformación Digital», *ITAC*. [En línea]. Disponible en: https://itac.co/mision%2Fvision. [Accedido: 01-abr-2019].

[18] Rational Software, «Rational Unified Process Best Practices for Software Development Teams». 2011.

[19] K. Schahwaber y J. Sutherland, «The Scrum Guide», *Scrum.org*. [En línea]. Disponible en: https://www.scrum.org/index.php/resources/scrum-guide. [Accedido: 31-mar-2019].

[20] ITAC, «DevOps e Innovación para la Transformación Digital», *ITAC*. [En línea]. Disponible en: https://itac.co/compa%C3%B1%C3%ADa. [Accedido: 01-abr-2019].

[21] Francisco de Asís López Fuentes, «Sistemas distribuidos», p. 201.

[22] José Manuel Gómez Álvarez, «Doctorado en Ciencias de la Computación e Inteligencia Artificial, 2002-2003», *Sistemas de Información Multiagente*, 09-oct-2007. [En línea]. Disponible en: https://web.archive.org/web/20071009182341/http://www.irisel.com/~jmgomez/IT/doctorate/taller_2.htm#_To

c129439877. [Accedido: 10-abr-2019].
[23] M. P. Papazoglou, *Web services. principles and technology*. Harlow, Essex ; New York Pearson/Prentice Hall 2008., 2008.
[24] «About WS-I - Overview». [En línea]. Disponible en: http://www.ws-i.org/about/Default.aspx. [Accedido: 21-abr-2019].
[25] Purificación Aguilera López, *Seguridad Informática*. EDITEX, 2010.
[26] J. F. Roa Buendía, *Seguridad informática*. Madrid: McGraw-Hill España, 2013.
[27] MikeRayMSFT, «rowversion (Transact-SQL) - SQL Server». [En línea]. Disponible en: https://docs.microsoft.com/es-es/sql/t-sql/data-types/rowversion-transact-sql. [Accedido: 31-mar-2019].
[28] H. A. Lopez, A. Aristizabal, F. D. Valencia, y C. Rueda, «Tiempo, Listas negras y Seguridad en SPL», p. 12.
[29] «Señal de nombre de usuario», 24-oct-2014. [En línea]. Disponible en: undefined. [Accedido: 31-mar-2019].
[30] «Escenario 7: solo cifrado - autenticación UsernameToken - Guía del desarrollador de WSO2 [Libro]». [En línea]. Disponible en: https://www.oreilly.com/library/view/wso2-developers-guide/9781787288317/092a1b2c-9fcc-4caf-9b99-d16a71ad19ce.xhtml. [Accedido: 31-mar-2019].
[31] «Firma digital - ¿Qué es y para qué sirve?», *Emprender Fácil*, 26-nov-2014. [En línea]. Disponible en: https://www.emprender-facil.com/es/firma-digital-que-es-y-para-que-sirve/. [Accedido: 31-mar-2019].
[32] T. Vivas, M. Huerta, A. Zambrano, R. Clotet, y C. Satizábal, «Aplicación de Mecanismos de Seguridad en una Red de Telemedicina Basados en Certificados Digitales», en *IV Latin American Congress on Biomedical Engineering 2007, Bioengineering Solutions for Latin America Health*, 2008, pp. 971-974.
[33] F. A. T. Jr y C. W. Jennings, «Digital Certificate», US6189097B1, 13-feb-2001.
[34] «| HotswapAgent». [En línea]. Disponible en: http://hotswapagent.org/index.html. [Accedido: 21-abr-2019].
[35] «Dynamic Code Evolution VM | A modification of the Java HotSpot(TM) VM that allows unlimited class redefinition at runtime.» [En línea]. Disponible en: http://ssw.jku.at/dcevm/. [Accedido: 19-abr-2019].
[36] «Byte Buddy - runtime code generation for the Java virtual machine». [En línea]. Disponible en: https://bytebuddy.net/#/. [Accedido: 21-abr-2019].
[37] «Fusion Middleware Developer's Guide for Oracle Service Bus». [En línea]. Disponible en: https://docs.oracle.com/cd/E21764_01/dev.1111/e15866/message_level.htm#OSBDV1624. [Accedido: 21-abr-2019].
[38] «Oracle® Fusion Middleware Understanding Oracle Web Services Manager». [En línea]. Disponible en: https://docs.oracle.com/middleware/1212/owsm/OWSMC/owsm-intro.htm#OWSMC110. [Accedido: 21-abr-2019].
[39] «IBM InfoSphere Guardium Data Encryption for encryption of data at rest». [En línea]. Disponible en: https://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.5.0/com.ibm.db2.luw.admin.sec.doc/doc/c0053466.html. [Accedido: 16-abr-2019].
[40] «Seguridad de servicios web, Parte 1», 06-ene-2012. [En línea]. Disponible en: http://www.ibm.com/developerworks/ssa/library/ws-best11/index.html. [Accedido: 21-abr-2019].
[41] «Cloud Data Protection Solution | Digital Guardian». [En línea]. Disponible en: https://digitalguardian.com/products/cloud-data-protection. [Accedido: 16-abr-2019].
[42] «Network Data Loss Prevention | Digital Guardian». [En línea]. Disponible en: https://digitalguardian.com/products/network-dlp. [Accedido: 16-abr-2019].
[43] «MyDiamo Product Features». [En línea]. Disponible en: https://mydiamo.com/product/features/. [Accedido: 16-abr-2019].
[44] «Trusted End Node Security - Encryption Wizard». [En línea]. Disponible en: https://www.spi.dod.mil/ewizard.htm. [Accedido: 16-abr-2019].
[45] «Download Encryption Wizard 3.4.11». [En línea]. Disponible en: https://www.softpedia.com/get/Security/Encrypting/Encryption-Wizard.shtml. [Accedido: 16-abr-2019].