Universidad de las Ciencias Informáticas

Facultad 15

"Estandarización del componente autenticación del Sistema de Gestión Integral de Seguridad (ACAXIA)"

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

Autor(es): Daniel Enrique López Méndez

Tutor(es): Ing. Darién García Tejo

Ing. Noel Jesús Rivero Pino

Ciudad de la Habana

Junio de 2010

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al departamento de tecnología de CEIGE de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los 21 días del mes de junio del año 2010.

Daniel Enrique López Méndez — Firma del Autor

Ing. Noel Jesús Rivero Pino — Firma del Tutor

Dedicatoria

Dedico este trabajo a toda mi familia y mis amigos que me han apoyado siempre. A mis padres que son mi vida, mi orgullo, a mi tía Lile que también ha luchado conmigo y ha educado, a mis hermanitas que las quiero mucho, que estudien para que puedan graduarse, a mi tío Amílcar y mi primo Leandro, ojalá me estén viendo de algún lugar, a mis amigos, los de aquí y los de allá y en especial a mis abuelos que son la prueba más grande de amor que tengo.

Agradecimientos

Antes que nada a mis padres y mi sol que son lo más grande que tengo y lo que más quiero en el mundo. Gracias por siempre estar ahí.

A mi tutor Darién, a Oiner, a Borbón, a Noel que me ayudaron mucho en la solución de la tesis, muchas gracias hermanos por la paciencia y por el conocimiento.

A mis colegas, ... que colegas a mis hermanos de aquí de la UCI Edel, Yoandris, Misael, Jose, La Flaca, Nallelys, a Kamelia por aguantarme, gracias nn... en fin a todos que han apartado su grano de arena también y que son sin duda la adquisición más valiosa de la UCI.

Al tribunal y oponente por sus consejos y ayuda. Gracias

"Tu tiempo está limitado, así que no lo desaproveches viviendo la vida de algún otro. No te dejes arrastrar por los dogmas, que es lo mismo que vivir con los resultados del pensamiento de otras personas. No dejes que el ruido de las opiniones de otros ahoguen completamente tu voz interior. Y más importante, ten el valor de seguir a tu corazón y a tu intuición. Ellos, de algún modo, ya saben en lo que verdaderamente te quieres convertir. Todo lo demás es secundario."

Steve Jobs

Resumen

La seguridad es un factor fundamental en cualquier institución y los datos son una parte fundamental en las mismas, y cualquier pérdida, desvío o mala manipulación de la información ocasionaría daños económicos y sociales irreparables. Por tanto, se requiere de una buena administración y control de la información y medios materiales, así como una adecuada seguridad acorde a los intereses de un país.

En la actualidad, la tendencia que existe a nivel mundial para el control y administración de la información es el uso de sistemas de seguridad que implementen al menos tres de los procesos característicos de un sistema de este tipo (autenticación, autorización y auditoría).

Uno de los proyectos realizados en el Centro de Informatización para la Gestión de Entidades (CEIGE) de la Universidad de las Ciencias Informáticas (UCI) es el Sistema de Gestión Integral de Seguridad (ACAXIA). Dicho sistema en su primera versión, implementó los tres procesos mencionados anteriormente (autenticación, autorización y auditoría) y añadió dos más (administración de Perfiles y administración de Conexiones). Esta primera versión tenía los inconvenientes que dichos procesos no se ejecutaban de forma independiente y que su implementación no se regía por ningún estándar.

Es por esta razón que el objetivo del presente trabajo es realizar el análisis, diseño e implementación de un nuevo componente de autenticación de ACAXIA para estandarizar e independizar el proceso de autenticación. Para esto se realiza un estudio de las soluciones estandarizadas para la autenticación más usadas a nivel mundial, arrojando como solución el uso del estándar internacional para el intercambio de datos de autenticación y autorización SAML. La nueva versión del Sistema de Gestión Integral de Seguridad (ACAXIA), que surgirá a partir de la estandarización de este proceso, brindará nuevas funcionalidades, como son la inclusión de una firma y un certificado digital y la implementación de una arquitectura de Single Sign-On, funcionalidades que brindarán una mayor seguridad al sistema ACAXIA y en general hará de dicho sistema una solución más robusta.

La nueva versión brindará además, una mayor integración con otros sistemas a través de servicios web y del componente IoC[1]

[1] **IoC** del inglés Inversion of Control, es un concepto junto a unas técnicas de programación en las que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos (procedure calls) o funciones.

Índice

Introducción ..... - 1 -
CAPÍTULO 1: Fundamentación teórica ..... - 3 -
1.1 Introducción ..... - 3 -
1.2 Seguridad en aplicaciones web ..... - 4 -
1.3 Sistemas de Autenticación Centralizados ..... - 7 -
1.4 Sistemas de gestión de la autenticación utilizados actualmente a nivel mundial ..... - 7 -
1.5 Sistemas de gestión de la autenticación utilizados actualmente en la UCI ..... - 8 -
1.6 Estudio de las soluciones ..... - 9 -
1.6.1 Metodologías para el desarrollo de software ..... - 9 -
1.6.1.1 Metodologías tradicionales (no ágiles) ..... - 9 -
1.6.1.2 Metodologías ágiles ..... - 10 -
1.6.2 Lenguajes de programación ..... - 11 -
1.6.3 Tecnologías (Frameworks) ..... - 13 -
1.6.3.1 Symfony ..... - 13 -
1.6.3.2 CakePHP ..... **¡Error! Marcador no definido.**
1.6.3.3 Zend Framework ..... - 14 -
1.6.4 Herramientas ..... - 14 -
1.6.4.1 Zend Studio ..... - 14 -
1.6.4.2 Apache ..... - 15 -
1.6.4.3 Postgres SQL ..... - 16 -
1.6.4.4 Visual Paradigm ..... - 16 -
1.6.5 Soluciones para la autenticación ..... - 17 -
1.6.5.1 RADIUS ..... - 17 -
1.6.5.2 Portales cautivos ..... - 18 -
1.6.5.3 SAML ..... - 18 -
1.6.5.3.1 Arquitectura de SAML ..... - 19 -
1.6.5.3.2 Seguridad en SAML ..... - 22 -
1.6.5.3.3 Criptografía en SAML ..... - 26 -
1.6.5.3.4 Por qué usar SAML ..... - 28 -
1.7 Resultados esperados u objetivos de la solución propuesta ..... - 30 -
1.8 Conclusiones Parciales ..... - 30 -
Capítulo 2: Desarrollo de la solución ..... - 31 -
2.1 Introducción ..... - 31 -
2.2 Arquitectura de Software ..... - 31 -
2.3 Escenarios Arquitectónicos ..... - 31 -
2.3.1 Federación de Identidades ..... - 31 -

2.3.2 Varios drivers de autenticación ..... - 33 -
2.3.3 Single Sign-On ..... - 33 -
2.4 Propuesta de Solución ..... - 36 -
2.5 Requisitos de la solución propuesta ..... - 37 -
2.5.1 Requisitos Funcionales ..... - 37 -
2.5.2 Requisitos no Funcionales ..... - 42 -
2.6 Prototipo de interfaz de Usuario ..... - 43 -
2.7 Modelo de Análisis ..... - 44 -
2.7.1 Modelo Conceptual ..... - 44 -
2.8 Modelo de Diseño ..... - 45 -
2.8.1 Diagrama de clases con estereotipos web ..... - 45 -
2.8.2 Descripción de clases y atributos ..... - 46 -
2.9 Artefactos de implementación ..... - 49 -
2.9.1 Diagrama de componentes ..... - 49 -
2.9.2 Diagrama de despliegue ..... - 50 -
2.9.3 Estándares de Nomenclatura ..... - 50 -
2.10 Conclusiones parciales ..... - 52 -
Capítulo 3: Pruebas y validación ..... - 53 -
3.1 Introducción ..... - 53 -
3.2 Casos de Prueba ..... - 53 -
3.3 Validación de la solución ..... - 55 -
3.2.1 Valoración de la solución ..... - 55 -
3.2.1.1 Resultados del instrumento de evaluación de la métrica TOC ..... - 58 -
3.2.1.2 Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC)- 60 -
3.2.1.3 Matriz de cubrimiento ó matriz de inferencia de indicadores de calidad ..... - 63 -
3.3 Conclusiones Parciales ..... - 65 -
Conclusiones ..... - 66 -
Recomendaciones ..... - 67 -
Anexos ..... - 68 -
Anexo 1 ..... - 68 -
Anexo 2 ..... - 69 -
Referencias bibliográficas ..... - 71 -
Bibliografía ..... - 72 -

Introducción

En Cuba el desarrollo de la informática y las comunicaciones desde hace unos años hasta la fecha ha estado inmerso en un proceso de perfeccionamiento continuo con el objetivo de insertar a la isla en el mercado mundial del software, aprovechando la capacidad intelectual e inventiva de los cubanos y llegar así a convertirse en puntera del desarrollo tecnológico al igual países como Estados Unidos, China, India, entre otros.

Producto de dicho desarrollo surgió la Universidad de las Ciencias Informáticas (UCI), la cual cuenta con recursos tanto humanos como materiales para convertir en realidad las esperanzas de llegar a tener una de las sociedades mejor informatizadas.

El Sistema Integral de Gestión (Cedrux), es uno de los principales software que se encuentra desarrollando la UCI en conjunto con la Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa (UCID). Por la importancia de la información que manejará esta aplicación, se comenzó paralelamente el desarrollo de un sistema que permitiera administrar la seguridad de forma centralizada en un entorno de varias aplicaciones.

La primera versión desarrollada del Sistema de Gestión Integral de Seguridad permitía la autenticación, autorización, administración de perfiles, administración de conexiones y auditoría. Uno de los principales inconvenientes de esta versión era que los procesos antes mencionados no se ejecutaban de forma independiente, sino que estaban totalmente cohesionados, lo que imposibilitaba que los usuarios puedan utilizar por separado cada uno de los componentes. Por otra parte, para su construcción no se tuvieron en cuenta los estándares internacionales que rigen el desarrollo de un sistema de este tipo.

Por todo lo anteriormente descrito se propuso desarrollar una nueva versión del Sistema de Seguridad con el objetivo principal de estandarizar cada uno de sus componentes de forma independiente tomando en cuenta las nuevas funcionalidades, de forma tal que se obtenga un software mucho más flexible, configurable y de fácil utilización.

Teniendo en cuenta la situación planteada, el problema a resolver queda expresado en la siguiente interrogante: ¿Cómo lograr una mayor seguridad en la comunicación entre las aplicaciones, aumentando las posibilidades de integración y generalización del Sistema de Gestión Integral de Seguridad?

Para ello se tendrá como **Objeto de estudio**: Software de gestión.

Identificándose como **campo de acción**: La administración de la seguridad guiada por estándares en software de gestión.

Para resolver el problema planteado se ha propuesto como **objetivo general**: Desarrollar el componente de autenticación de ACAXIA guiado por estándares.

Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

- Definir y fundamentar los elementos teóricos para el análisis del problema a resolver.
- Seleccionar el estándar a utilizar para la autenticación.
- Definir escenarios arquitectónicos y agrupar los requisitos.
- Realizar el levantamiento de requisitos para ambos procesos.
- Diseñar e implementar el componente de autenticación.
- Realizar pruebas a la nueva versión del sistema de gestión integral de seguridad.

Se tiene como **idea a defender** en la presente investigación:

Si se lleva a cabo la estandarización del componente autenticación entonces se logrará una mayor seguridad en la comunicación entre las aplicaciones, aumentando las posibilidades de integración y generalización.

- V.I. Estandarización del componente de autenticación.
- V.D. Mayor seguridad en la comunicación entre las aplicaciones, aumentando las posibilidades de integración y generalización.

El trabajo consta de tres capítulos.

En el **Capítulo 1** se expone la fundamentación teórica del trabajo, el mismo incluye, estudio de soluciones (metodologías y modelo de desarrollo de software, soluciones para la autenticación, tecnologías y herramientas), elementos de reutilización y resultados esperados u objetivos de la solución propuesta.

En el **Capítulo 2** se exponen los requisitos, prototipos de interfaces, análisis y diseño de clases y los artefactos de implementación.

En el **Capítulo 3** se realizan los casos de prueba y se da una valoración de la solución.

CAPÍTULO 1: Fundamentación teórica

1.1 Introducción

El vertiginoso desarrollo alcanzado en las nuevas tecnologías de la informática y las comunicaciones ha llevado a la sociedad a entrar en lo que se ha denominado como "era de la información". La información puede existir en muchas formas, impresa o escrita en papel, almacenada digitalmente, transmitida por correo postal o utilizando medios digitales, presentada en imágenes o expuesta en una conversación. Cualquiera que sea la forma que adquiere la información es un recurso que, como el resto de los activos importantes tiene un gran valor, siendo a veces incalculable, por contener la "vida" de una organización; es por eso que debe ser debidamente protegida independientemente de los medios por los cuales se distribuye o almacena. En esta "sociedad de la información" persisten las razones y motivos para mantener mecanismos de control de acceso sobre las áreas (seguridad física) y la información (seguridad lógica) que se desea proteger.

A medida que crece el número de aplicaciones y usuarios, administrar las identidades y sus derechos de acceso se convierte en una tarea tediosa y proclive a errores. Así mismo, los usuarios se encuentran con la molestia de tener que re-autenticarse para cada aplicación usando diferentes credenciales e incluso diferentes métodos de autenticación.

Un conjunto de aplicaciones pueden crear un dominio de seguridad externalizando sus procesos de autenticación y autorización de manera que comparten un elemento (token[2]) de seguridad generado por un proveedor de identidad. De este modo la administración global de las identidades digitales y sus derechos de acceso se simplifica y resulta más fiable. Adicionalmente los usuarios pueden llegar a acceder a todas las aplicaciones con un único proceso de autenticación.

En el contexto de los servicios web, el estándar Lenguaje de Enmarcado de Aserciones de Seguridad (en inglés Security Assertion Markup Lenguaje cuyas siglas son SAML) estandariza el intercambio de los datos de autenticación y autorización generados por el proveedor de identidad e interpretados por las aplicaciones.

[2] Un ***token* de seguridad** (también ***token* de autenticación** o ***token* criptográfico**) es un dispositivo que se le da a un usuario autorizado de un servicio computarizado para facilitar el proceso de autenticación.

En este capítulo se hace un estudio de las soluciones que existen a nivel mundial para la autenticación y se propone una solución en cuanto a modelo de desarrollo, lenguaje, tecnologías, herramientas y estándar respecta.

1.2 Seguridad en aplicaciones web

Se expondrán elementos de la seguridad en aplicaciones web dando una definición de seguridad informática. Seguidamente se analizarán algunos posibles ataques y se analizará por su importancia el proceso de autenticación.

El término seguridad informática es una generalización para un conjunto de tecnologías que ejecutan ciertas tareas relativas a la seguridad de los datos (Microsoft Corporation, 2009).

ISO, en su norma 7498, define la seguridad informática como una serie de mecanismos que minimizan la vulnerabilidad de bienes y recursos, donde un bien se define como algo de valor y la vulnerabilidad como la debilidad que se puede explotar para violar un sistema o la información que contiene (Grupo de interés especial en datos de comunicación, 2010). El bien más preciado para cualquier institución es la información y de ahí que se han desarrollado protocolos y mecanismos adecuados, para preservar su seguridad. Se puede hablar en este sentido de cinco conceptos principales de la seguridad de los sistemas: autenticación o autentificación, autorización, auditoría, administración de perfiles y administración de conexiones. Basándose en estos conceptos a la hora de implementar la seguridad, se lograría cumplir con la confidencialidad[3], integridad[4] y el no-repudio[5] aspectos fundamentales para cualquier sistema que gestione información.

A la hora de desarrollar una aplicación web, generalmente se enfoca el desarrollo, más en la funcionalidad que en la seguridad. Lo que trae como consecuencia que los atacantes se aprovechen de esto y atenten contra cualquiera de estos tres aspectos. A continuación algunos ejemplos de ataques

[3] **Confidencialidad** es la propiedad de la información, por la que se garantiza que está accesible únicamente a personal autorizado a acceder a dicha información.

[4] Para la Seguridad de la Información, **la integridad** es la propiedad que busca mantener los datos libres de modificaciones no autorizadas.

[5] Proporciona protección contra la interrupción, por parte de alguna de las entidades implicadas en la comunicación, de haber participado en toda o parte de la comunicación. El servicio de Seguridad de **No repudio** está estandarizado en la ISO-7498-2

Ataques:

- **Inyección de SQL**: en algunas ocasiones existen compromisos por falta de la debida implementación del desarrollador, la causa de este tipo de problemas es la falta de verificación del código y colocación de validaciones de los datos en los puntos apropiados. La inyección de SQL consiste en colocar en los formularios de captura de información para los usuarios, instrucciones para la base de datos las que por falta de una adecuada validación pasan directo a la base de datos y se ejecutan como código legitimo, de esta forma, se puede obtener la lista de usuarios y todos los datos de las diferentes aplicaciones.
- **Manipulación de cookies**: la manipulación de cookies es una labor muy simple, solo se deben ubicar los archivos de textos que se almacenan en la máquina del usuario y modificarlas a su antojo, no importa que incluso la información esté codificada o encriptada, dado que la información está en la maquina del usuario, se puede acceder a ella e intentar un ataque por fuerza bruta.
- **Suplantación de sesión**: este tipo de ataque es más complejo que los anteriores, consiste en la captura inicial de información (usuario y contraseña) y luego la generación de un ataque de denegación del servicio (DoS)6 hacia el cliente, seguidamente se toman los datos de configuración del usuario y se continúa con la sesión del usuario.

En la seguridad de aplicaciones desempeñan un papel fundamental los procesos de autenticación y autorización, ya que permiten un mejor control en el acceso a la información.

Como se observa en los ejemplos anteriores la mayoría de los ataques tienen que ver de una forma u otra con el proceso de autenticación, de ahí su importancia para cualquier sistema de seguridad web.

A continuación, algunos aspectos de la **autenticación**:

Autenticación o autentificación es el acto de establecimiento o confirmación de algo (o alguien) como auténtico. La autenticación de un objeto puede significar la confirmación de su procedencia, mientras que la autenticación de una persona a menudo consiste en verificar su identidad (TechNet Microsoft Corporation, 2009). Existen cuatro tipos de técnicas que permiten realizar la autenticación de la

[6] En seguridad informática, un **ataque de denegación de servicio**, también llamado ataque **DoS** (de las siglas en inglés ***D**enial **o**f **S**ervice*), es un ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos. Normalmente provoca la pérdida de la conectividad de la red por el consumo del ancho de banda de la red de la víctima o sobrecarga de los recursos computacionales del sistema de la víctima.

identidad del usuario, las cuales pueden ser utilizadas individualmente o combinadas (autenticación de varios factores):

- Autenticación por algo que la persona posee: por ejemplo una tarjeta magnética.
- Autenticación por algo que el individuo es y que lo identifica unívocamente: por ejemplo las huellas digitales.
- Autenticación por algo que solo el individuo es capaz de hacer: por ejemplo los patrones de escritura.

La fortaleza de la autenticación es mayor mientras más factores se adicionen, generalmente solo se utilizan hasta 3 factores:

- 1-factor = contraseña.
- 2-factores = contraseña + token.
- 3-factores = contraseña + token + biometría[7].
- 4-factores = contraseña + token + biometría + localización geográfica (GPS).
- 5-factores = contraseña + token + biometría + localización geográfica + perfil de usuario.

Características de la autenticación

Cualquier sistema de identificación ha de poseer determinadas características para ser viable:

- Ha de ser fiable con una probabilidad muy elevada (se puede hablar de tasas de fallo de en los sistemas menos seguros).
- Económicamente factible para la organización (si su precio es superior al valor de lo que se intenta proteger, se tiene un sistema incorrecto).
- Soportar con éxito cierto tipo de ataques.
- Ser aceptable para los usuarios, que serán al fin y al cabo quienes lo utilicen.

Mecanismo general de la autenticación

[7] La **biometría** es el estudio de métodos automáticos para el reconocimiento único de humanos basados en uno o más rasgos conductuales o físicos intrínsecos. El término se deriva de las palabras griegas "bios" de vida y "metrón" de medida.

La mayor parte de los sistemas informáticos y redes mantienen de uno u otro modo una relación de identidades personales (usuarios) asociadas normalmente con un perfil de seguridad, roles y permisos. La autenticación de usuarios permite a estos sistemas asumir con una seguridad razonable que quien se está conectando es quien dice ser para que luego las acciones que se ejecuten en el sistema puedan ser referidas luego a esa identidad y aplicar los mecanismos de autorización y/o auditoría oportunos.

El primer elemento necesario para la autenticación, es la existencia de identidades que puedan ser identificadas con un identificador único (valga la redundancia). Los identificadores de usuarios pueden tener muchas formas siendo la más común una sucesión de caracteres conocida comúnmente como login o proceso de logueo.

El proceso general de autenticación consta de los siguientes pasos:

- El usuario solicita acceso a un sistema.
- El sistema solicita al usuario que se autentique.
- El usuario aporta las credenciales que le identifican y permiten verificar la autenticidad de la identificación.
- El sistema valida según sus reglas si las credenciales aportadas son suficientes para dar acceso al usuario o no.

1.3 Sistemas de Autenticación Centralizados

En el ecosistema de software de las empresas y corporaciones de hoy en día, la seguridad de acceso a las aplicaciones de la organización constituye un creciente desafío. Implementar y poner en marcha un modelo de seguridad de acceso implica navegar múltiples aplicaciones, en la mayoría de los casos, con muy pocas características comunes entre ellas.

La norma hasta el momento, ha sido que cada aplicación dentro de una empresa o corporación cuente con un adecuado sistema de control de accesos y su propio repositorio de datos. En la actualidad dentro del ambiente de tecnología de cualquier empresa o corporación existen diversos métodos de autenticación que generan una gran ineficiencia. Sin embargo, con la implementación de un agente de Inicio de Sesión Único, en inglés Single Sign-On (SSO)[8] el sistema se encarga de almacenar, en una

[8] **Single Sign-On** (SSO) es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación.

base datos o directorios protegidos, las credenciales que permiten al usuario acceder a cada una de las aplicaciones o servicios en el momento que lo desee, ya que el proceso de autenticación se realiza de manera transparente para el usuario, una vez que éste ha sido autenticado por medio de la arquitectura SSO. Se puede decir que con dicha implementación el sistema simplificaría y centralizaría el control de accesos a todas las aplicaciones de la empresa o corporación, reduciendo el costo en la administración de seguridad, lo que logra un mejor rendimiento y velocidad de los procesos de autenticación y acceso, que facilita a los usuarios la interacción con los sistemas de la empresa, simplificando el manejo de claves y aumentando así los niveles de seguridad, ya que contará con una plataforma central para el manejo de la seguridad en todos los procesos de autenticación y acceso a sus aplicaciones.

1.4 Sistemas de gestión de la autenticación utilizados actualmente a nivel mundial

Actualmente existen soluciones para el control de la autenticación de varias aplicaciones de manera centralizada, o sea, en un entorno de varias aplicaciones de manera que cada aplicación utilice un mismo sistema de autenticación. Se puede destacar que estas propuestas son incipientes todavía por lo que en muchas ocasiones son miradas con recelos por los clientes. Unas de las soluciones que proponen una autenticación centralizada utilizadas a nivel mundial son: Access Máster IAM (Gestión de Identidades y Acceso) & SSO (Single Sign-On).

1.5 Sistemas de gestión de la autenticación utilizados actualmente en la UCI

La investigación en la UCI sobre las aplicaciones de gestión de seguridad arrojó como resultado que la mayoría de las aplicaciones gestionan su propio sistema de autenticación, o sea, no utilizan un sistema de seguridad que implemente o gestione los procesos de autenticación. No obstante la mayoría de estos sistemas implementan un Sistema de Gestión de Sesiones basado en la arquitectura SSO cuyo objetivo es gestionar las sesiones de los usuarios en un único proceso de autenticación invisible a los ojos de los mismos, a través de un sistema con una fachada de servicios web que sea capaz de gestionar la apertura y cierre de sesiones por parte de las personas que trabajan en el dominio uci.cu.

1.6 Estudio de las soluciones

En la actualidad existen muchas metodologías de desarrollo de software, lenguajes de programación, tecnologías de desarrollo, herramientas y soluciones para la autenticación en general. A continuación se hace un estudio de las soluciones y se llega a una propuesta en cada aspecto de los antes mencionados.

1.6.1 Metodologías para el desarrollo de software

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc.

Patricio Letelier[9] (Letelier, 2003) define que la metodología en un proyecto de desarrollo de software define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

Según lo antes mencionado, una metodología puede definirse como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software ya que indican cómo hay que obtener los distintos productos parciales y finales.

Considerando su filosofía de desarrollo, estas se clasifican en metodologías tradicionales (no ágiles) y metodologías ágiles. Las metodologías tradicionales hacen mayor énfasis en la planificación y control del proyecto y en la especificación precisa de requisitos y modelado y las metodologías ágiles están más orientadas a la generación de código con ciclos muy cortos de desarrollo, haciendo especial hincapié en aspectos humanos asociados al trabajo en equipo e involucrando activamente al cliente en el proceso.

1.6.1.1 Metodologías tradicionales (no ágiles)

Estas se centran en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir y las herramientas y notaciones que se usarán. Dentro de estas metodologías se destaca RUP (Sánchez, 2007), llamada así por sus siglas en inglés Rational Unified Process, esta divide en 4 fases el desarrollo del software: Inicio, Elaboración, Construcción, Transición

[9] Patricio Letelier Torres Profesor titular de la universidad politécnica de Valencia.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es más adaptable para proyectos de largo plazo Es importante resaltar que RUP, por el especial énfasis que presenta en cuanto a su adaptación a las condiciones del proyecto (mediante su configuración previa a aplicarse), realizando una configuración adecuada, podría considerarse ágil. (Universidad politécnica de Valencia, 2006)

1.6.1.2 Metodologías ágiles

Las metodologías ágiles son efectivas en proyectos con requisitos muy cambiantes y en donde se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad de los mismos. Es una metodología sencilla, tanto en su aprendizaje como en su aplicación, lo que posibilita la reducción de los costos de implantación en un equipo de desarrollo. Están especialmente orientadas para proyectos pequeños y entre sus características más destacables está que dan un mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

Ejemplos de metodologías ágiles son:

- Extreme Programming.
- SCRUM.

Programación extrema (Extreme Programming, XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Es una de las metodologías de desarrollo de software más exitosas en la actualidad (Sánchez, 2007). XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al

extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP sin cubrir los detalles técnicos y de implantación de las prácticas. La Metodología XP, se recomienda para proyectos de corto plazo (Sánchez, 2007)

SCRUM

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos: El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. Las reuniones que se realizan a lo largo de la duración del proyecto, (entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (José H. Canós, 2003)) es otra característica importante.

Propuesta de modelo de desarrollo

Se propone como modelo de desarrollo la propuesta para software tecnológico desarrollado por la ingeniera en ciencias informáticas Mileydis Magalys Sarduy, que está orientado a la reutilización de componentes parametrizables, donde se simplifican las actividades y se eleva el nivel de especialización de los involucrados. Este modelo permite organizar el proceso de desarrollo de software tecnológico del Departamento de tecnología del Centro de Informatización para la Gestión de Entidades (CEIGE).

Dicho modelo está basado en principios y buenas prácticas de metodologías ágiles, fue elaborado teniendo en cuenta las características especiales que presenta la UCI y tiene un valor social representativo, ya que implica mejoría en aspectos importantes como la planeación, formación y satisfacción del equipo de trabajo (Pérez, 2009).

1.6.2 Lenguajes de programación

JAVA

Es un lenguaje de programación orientado a objetos. Este lenguaje de programación se ha convertido en uno de los lenguajes más demandados por los desarrolladores. Está enfocado fundamentalmente en que el entorno de desarrollo de un producto software sea independiente de la plataforma en la que se ejecute. Java ofrece un lenguaje altamente potente, similar al lenguaje C++ poniendo un poco más de énfasis en el tema de la seguridad. JAVA es un lenguaje de programación orientado a objetos y fue

creado con el objetivo de servir como nueva manera de manejar complejidad del software. JAVA se utiliza en una variedad de plataformas computacionales de los dispositivos. La programación de JAVA permite el desarrollo de programas del rendimiento seguro y alto en las plataformas múltiples. Entre las ventajas que presenta este lenguaje de programación están las siguientes:

- Es una fuente abierta, así que los usuarios no tienen que luchar con los impuestos sobre patente pesados cada año
- Independiente de la plataforma
- JAVA realiza la colección de basura de las ayudas, así que la gerencia de memoria es automática
- JAVA asigna siempre objetos en el apilado
- JAVA abrazó el concepto de especificaciones de la excepción
- Usando JAVA podemos desarrollar aplicaciones web dinámicas
- Permite que usted cree programas modulares y códigos reutilizables.

PHP

Sería idóneo utilizar PHP como lenguaje de programación puesto que es un lenguaje completamente orientado al desarrollo de aplicaciones Web. Se usa para la creación de contenido dinámico para sitios web, ofreciendo soluciones simples y universales para las paginaciones dinámicas web de fácil programación. Entre sus características fundamentales se encuentra que es gratuito, ya que al tratarse de software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre. Es de gran popularidad, pues existe una gran comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código. Presenta una sencilla integración con múltiples bases de datos. Actualmente se usa por su versatilidad de uso con diferentes sistemas operativos. Produce sensación al usuario de mayor rapidez y mayor usabilidad ya que es poco pesado. Dicho lenguaje de programación mejora el soporte para la programación orientada a objetos. Corre en cualquier plataforma usando el mismo código fuente. Puede interactuar con muchos motores de bases de datos. Su versión PHP 5.3.2 brinda mejoras significativas con respecto a versiones anteriores, con una orientación a objeto similar a la de Java. Utilizando PHP 5.3.2 las aplicaciones gozaran de nuevas capacidades, se obtendrá el beneficio de una mejor performance de ejecución (está comprobado experimentalmente que PHP5 corre un 25% más

rápido que versiones anteriores) y el código estará muy bien acondicionado en cuanto a la compatibilidad con la nueva versión que se asoma, PHP6.

Por estas razones y porque PHP fue el lenguaje seleccionado para implementar el Sistema de Gestión Integral de Seguridad, es el lenguaje que se propone para el desarrollo del componente.

1.6.3 Tecnologías (Frameworks)

Teniendo la propuesta de la metodología y el lenguaje se prosigue al estudio de las tecnologías desarrolladas sobre el lenguaje propuesto (PHP).

1.6.3.1 Symfony

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony cuenta con muchos complementos (plug-in)[10], un grupo de ellos es utilizado para gestionar la seguridad de las aplicaciones, dentro de este grupo el más usado, seguro y eficaz es: sfGuardPlug-in.

Sobre sfGuardPlug-in

En toda aplicación dinámica, siempre está presente la gestión de usuarios, permisos y roles. En Symfony esto puede resultar muy sencillo gracias a SfGuard. Este plug-in gestiona de manera muy sencilla usuarios, roles, permisos y logueos. Básicamente, la autenticación y la seguridad en una aplicación se basan en limitar el acceso a partes de la misma. Significa que los usuarios tendrán que iniciar una sesión (autenticación) para acceder a ciertas áreas (seguridad). Diferentes usuarios pueden tener diferentes privilegios (autorización). De esta manera, aseguras tu aplicación para diferentes tipos de usuarios. SfGuard Plug-in te brinda el modelo (usuario, grupo y objetos de permisos). Está compuesto por cuatro módulos:

- El módulo sfGuardAuth es el encargado del logueo y el acceso restringido.

[10] Un **complemento** es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API. También se lo conoce como plug-in (del inglés "enchufable").

- El módulo sfGuardGroup es el encargado de la gestión de grupos o roles.
- El módulo sfGuardPermission es el encargado de la gestión de permisos.
- El módulo sfGuardUser es el encargado de la gestión de usuarios.

1.6.3.2 Zend Framework

En su nivel más simple, Zend Framework es una librería de componentes escritos en PHP5, para facilitar el desarrollo de sitios web. Como está basada en PHP5 (5.1.4 es la versión mínima necesaria), eso significa que es completamente Orientada a Objetos. Zend está formado por muchos componentes, estos están divididos en grupos, uno de estos grupos se encarga de gestionar la seguridad, los componentes de este grupo son: Zend_Auth permite chequear y guardar credenciales de usuario de distintas maneras: utilizando la base de Datos, el método Diguest[11] de Apache, o autenticación http simple. Provee un Adapter[12] de Interfaz para personalizar los mecanismos de autenticación, además almacenamiento automático de identidad para una fácil personalización. Es simple y extensible. A su vez Zend_Session trabaja como un administrador de datos de sesión, al igual que en PHP, solo que ofrece algo de valor agregado.

Se propone Zend Framework para el desarrollo del componente porque nos provee de las funcionalidades requeridas y por ser el framework que se seleccionó para el desarrollo del Sistema de Gestión Integral de Seguridad.

1.6.4 Herramientas

Se proponen las siguientes herramientas de desarrollo seleccionadas por el departamento de tecnología del Centro de Informatización para la Gestión de Entidades (CEIGE).

1.6.4.1 Zend Studio

Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web en lenguaje PHP. El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos

[11] Este método tiene la gran ventaja de que el usuario y la clave van cifradas del cliente al servidor. Pero tiene el inconveniente de que podemos encontrarnos con versiones antiguas de navegadores, que no lo soporten.

[12] El patrón **Adapter** (Adaptador) se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda.

hasta la depuración de código. El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más.

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración.

Ventajas:

- Posee resaltado de sintaxis.
- Posee auto completado de código.
- Tiene identificación de errores en el código.
- Presenta depurador (debugger) de PHP.

1.6.4.2 Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA Http 1.3, pero más tarde fue reescrito por completo. Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA[13]. Era, en inglés, a patchy server (un servidor "parcheado").

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Ventajas:

[13] El NCSA HTTPd era un Servidor web desarrollado originalmente en el National Center for Supercomputing Applications por Robert McCool y una lista de colaboradores.

- Es modular.
- Es de código abierto.
- Es multi-plataforma.
- Es extensible.
- Es popular (fácil conseguir ayuda/soporte).

1.6.4.3 Postgres SQL

Este es un es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS), basado en Software libre. Está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo donde cada usuario puede tener una visión sobre lo último que se hizo. Ofrece nuevos conceptos como son: clases, herencia, tipos y funciones. Aporta en cierta forma potencia y flexibilidad adicional como son las restricciones, los disparadores, las reglas y la integridad transaccional (Lockhart, 1996). Postgres SQL es un gestor altamente extensible que soporta operadores y tipos de datos definidos por los distintos usuarios. Cuenta además con un mejor soporte para los procedimientos que se acumulan en el servidor. Este ofrece además menor tiempo de respuesta a la hora de ejecutar cualquier consulta.

Ventajas:

- Es estable.
- Es flexible.
- Se puede extender su funcionalidad.
- Tiene gran compatibilidad con sistemas operativos.

1.6.4.4 Visual Paradigm

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

Ventajas:

- Tiene ingeniería de ida y vuelta.
- Tiene ingeniería inversa (de código a modelo y de código a diagrama).
- Genera Base de Datos.
- Posee un modelado colaborativo con Subversion.
- Posee un editor de detalles de caso de uso.

1.6.5 Soluciones para la autenticación

Se estudiaron tres soluciones para la autenticación, dos soluciones como tal (RADIUS y Portales Cautivos) y un estándar (SAML) adaptable a cualquier solución.

1.6.5.1 RADIUS

RADIUS (acrónimo en inglés de Remote Authentication Dial-In User Server). Es un protocolo de autenticación y autorización para aplicaciones de acceso a la red o movilidad IP. Utiliza el puerto 1813 UDP para establecer sus conexiones.

Cuando se realiza la conexión mediante módem[14], DSL[15], cable módem, Ethernet o Wi-Fi[16], se envía una información que generalmente es un nombre de usuario y una contraseña. Esta información se transfiere a un dispositivo Servidor de Acceso a la Red, en inglés Network Access Server (NAS), quien redirige la petición a un servidor RADIUS sobre el protocolo RADIUS. El servidor RADIUS comprueba que la información es correcta. Si es aceptado, el servidor autorizará el acceso al sistema y le asigna los recursos de red como una dirección IP, etc.

Una de las características más importantes del protocolo RADIUS es su capacidad de manejar sesiones, notificando cuando comienza y termina una conexión, así que al usuario se le podrá

[14] Un **módem** es un dispositivo que sirve para enviar una señal llamada portadora mediante otra señal de entrada llamada moduladora.

[15] **DSL** (Digital Subscriber Line) es un protocolo de banda ancha de internet.

[16] **Wi-Fi** (pronunciado en español /wɪ fɪ / y en inglés /waɪ faɪ /) es un sistema de envío de datos sobre redes computacionales que utiliza ondas de radio en lugar de cables, además es una marca de la *Wi-Fi Alliance* (anteriormente la *WECA: Wireless Ethernet Compatibility Alliance*), la organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11.

determinar su consumo y facturar en consecuencia; los datos se pueden utilizar con propósitos estadísticos.

RADIUS es extensible; la mayoría de fabricantes de software y hardware RADIUS implementan sus propios dialectos.

1.6.5.2 Portales cautivos

Una herramienta común de autenticación utilizada en las redes inalámbricas es el portal cautivo. Este utiliza un navegador web estándar para darle al usuario la posibilidad de presentar sus credenciales de registro. También puede utilizarse para presentar información (como Política de Uso Aceptable) a los usuarios antes de permitir el acceso. Mediante el uso de un navegador web en lugar de un programa personalizado de autenticación, los portales cautivos funcionan en prácticamente todas las computadoras portátiles y sistemas operativos. Generalmente se utilizan en redes abiertas que no tienen otro método de autenticación (como WEP[17] o filtros MAC[18]).

Para comenzar, el usuario abre su computadora portátil y selecciona la red. Su computadora solicita una dirección mediante DHCP[19] y le es otorgada. Luego usa su navegador web para ir a cualquier sitio en Internet.

1.6.5.3 SAML

En el presente epígrafe se estudiarán los conceptos básicos de SAML y una serie de puntos de seguridad que SAML analiza con gran detalle. Se centrará el estudio en la autenticación o autentificación inicial y se detallarán dos modelos, uno de confianza y otro de amenazas.

¿Qué es SAML?

[17] **WEP**, acrónimo de ***Wired Equivalent Privacy*** o "Privacidad Equivalente Cableado", es el sistema de cifrado incluido en el estándar IEEE 802.11 como protocolo para redes Wireless que permite cifrar la información que se transmite.

[18] La **dirección MAC** (siglas en inglés de **Media Access Control** o *control de acceso al medio*) es un identificador de 48 bits (6 bloques hexadecimales) que corresponde de forma única a una ethernet de red. Se conoce también como la dirección física en cuanto identificar dispositivos de red.

[19] **DHCP** (sigla en inglés de Dynamic Host Configuration Protocol - Protocolo de configuración dinámica de host) es un protocolo de red que permite a los nodos de una red IP obtener sus parámetros de configuración automáticamente.

SAML en su versión 2.0. SAML es un estándar para intercambiar información de autenticación y autorización entre dominios. Está diseñado para ofrecer SSO para interacciones automáticas o manuales entre sistemas. Éste permite el intercambio de información de autenticación y autorización sobre usuarios, dispositivos o cualquier entidad identificable llamados sujetos. Usando sintaxis de XML, SAML define el protocolo petición-respuesta por el cual los sistemas aceptan o rechazan sujetos basados en aserciones (world, 2010).

¿Qué define SAML?

SAML define la sintaxis y la semántica de las aserciones (assertions) que se hacen acerca de un tema (en este caso la autenticación) por una entidad del sistema. SAML define también la estructura que van a tener sus afirmaciones y un conjunto de protocolos de comunicación que el estándar soporta sobre los que van a viajar estas afirmaciones por la red. SAML define además dos conceptos fundamentales que son el Proveedor de Servicios[20] y el Proveedor de Identidad[21] que van a ser las partes que intervienen en la comunicación SAML, garantizando así que cualquier aplicación pueda intercambiar información de autenticación abstrayéndose del cómo.

1.6.5.3.1 Arquitectura de SAML

SAML define aserciones, protocolos, enlaces y perfiles, basándose en el estándar XML. El término Núcleo de SAML, se refiere a la sintaxis y semántica general de las aserciones SAML, así como el protocolo usado para realizar peticiones y transmitir aquellas aserciones de un sistema a otro.

Un enlace SAML determina como las peticiones y respuestas SAML son mapeadas en protocolos de mensajes y comunicaciones estándares. Un enlace importante (sincrónico) es el enlace SAML Simple Object Access Protocol (SOAP)[22].

Un perfil SAML es un conjunto concreto de un caso de uso definido, usando una combinación particular de aserciones, protocolos y enlaces.

Aserciones SAML

[20] Un **proveedor de servicios** es un componente que ofrece servicios de acceso a recursos a sus subscriptores a través de la web.

[21] Un **proveedor de identidad** es un componente que provee un certificado digital o token de seguridad a sus subscriptores para la identificación. Se aplica a arquitecturas como SSO.

[22] **SOAP** (siglas de *Simple Object Access Protocol*) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Una aserción SAML contiene un paquete de información segura que provee una o más declaraciones hechas por una autoridad SAML. Usando un lenguaje más formal, una entidad de confianza interpretaría una aserción de la siguiente forma:

La aserción A fue expedida en el tiempo T por el emisor R con relación al sujeto S con condiciones C válidas.

Las aserciones SAML se transfieren entre proveedores de entidades y proveedores de servicios. Estas aserciones contienen elementos que el proveedor de servicios usa para tomar decisiones de control de acceso. Tres tipos de declaraciones son provistas por SAML:

En **las declaraciones de autenticación** se afirma al proveedor de servicios, que un sujeto efectivamente realizó una autenticación con el proveedor de identidades en cierto tiempo usando un método particular de autenticación. Otra información acerca del sujeto autenticado (también llamado contexto de autorización), puede ser divulgada en éste tipo de declaración.

En **las declaraciones de atributos** se afirma que el sujeto tiene asociados ciertos atributos. Un atributo es simplemente una pareja de valores. Las entidades de confianza usan estos atributos para realizar decisiones de control de acceso.

En **las declaraciones de autorización** se afirma que a un sujeto le es permitido realizar la acción A sobre el recurso R dado la evidencia E. La expresividad de las declaraciones de autorización en SAML es intencionalmente limitado.

Protocolos SAML

Un protocolo SAML, describe cómo ciertos elementos SAML (incluidas las aserciones) son empaquetadas como peticiones y respuestas SAML, además brinda las reglas que las entidades SAML deberían seguir cuando consumen o producen estos elementos. Un protocolo SAML es un simple protocolo petición-respuesta.

El tipo más importante de un protocolo-petición SAML es llamado consulta. Un proveedor de servicios realiza una consulta directamente a un proveedor de identidades sobre un canal seguro. Estos mensajes de consultas son típicamente ligados con SOAP. Existen tres tipos de consultas SAML correspondientes a los tres tipos de declaraciones: consulta de autenticación, consulta de atributos y consulta de autorización.

Los protocolos permiten al proveedor de servicios:

- Pedir una consulta a una aserción.

- Pedir la autenticación a un sujeto.
- Solicitar un logout simultáneo de una colección de sesiones (Single Log Out).

Enlaces SAML

Un enlace SAML es un mapeo de un mensaje en el protocolo SAML dentro de los formatos de mensajería y/o protocolos de comunicación estándares. Por ejemplo, el enlace SOAP SAML especifica cómo un mensaje SAML es encapsulado en un sobre SOAP, que a su vez es ligado a un mensaje HTTP.

Perfiles SAML

Generalmente, un perfil SAML define limitaciones o extensiones para el uso de SAML en una aplicación particular, describiendo en detalle cómo las aserciones, protocolos y enlaces SAML son combinados para soportar un caso de uso definido. El perfil más importante en SAML es el perfil Web Browser SSO.

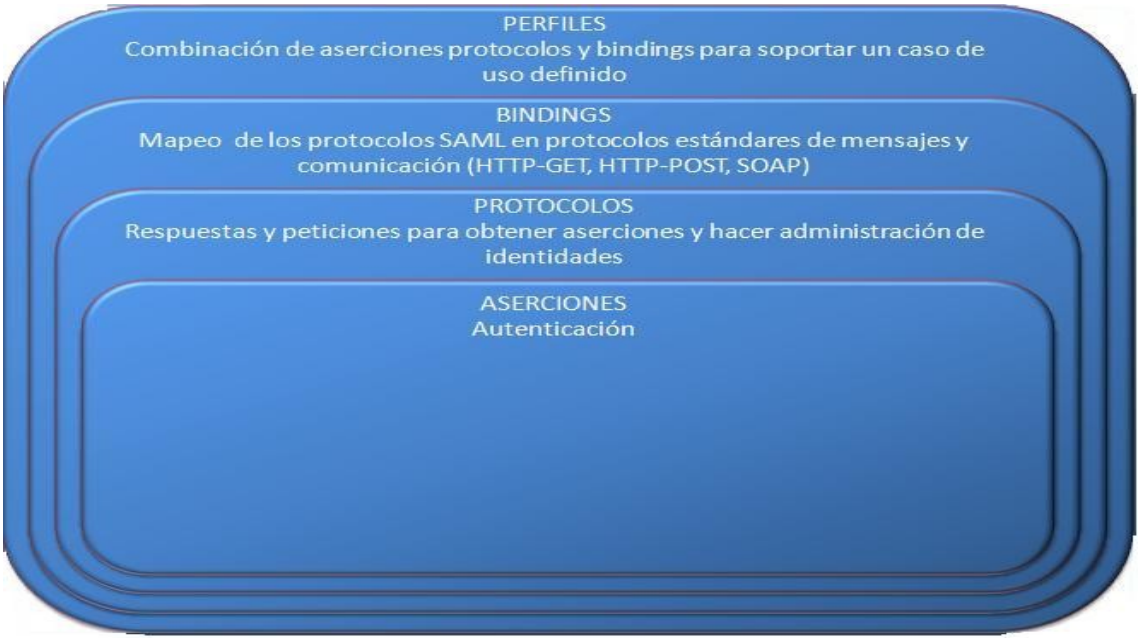


Fig. 1 Elementos de la Arquitectura del estándar SAML

SAML no especifica cuanta confidencialidad existe en una aserción. Los sistemas locales deciden si el nivel de seguridad y políticas de cierta aplicación son suficientes para proteger una organización, pues se pueden presentar resultados malignos de una decisión de autorización basada en una aserción errónea. Esta característica de SAML estimula relaciones de confianza y acuerdos operacionales entre

aplicaciones Web, en donde cada acuerdo para adherirse a un nivel de línea base de verificación debe realizarse antes de aceptar una aserción.

SAML puede ser integrado con múltiples protocolos de comunicación y transporte. De igual manera puede ser bastante útil al usarlo con SOAP sobre HTTP, como se mencionó anteriormente.

1.6.5.3.2 Seguridad en SAML

A continuación se definen unos conceptos involucrados en la seguridad, muy importantes. Se relacionan estos conceptos con la especificación SAML y se analizan que requisitos se deben de dar para que se cumplan.

Fig. 2 Elementos de seguridad

Privacidad

Por privacidad se entiende el control que los clientes tienen sobre la recolección, uso y distribución de su información personal. SAML incluye la capacidad de construir afirmaciones (aserciones) sobre los atributos (incluida la identidad) o sobre autorizaciones concedidas a entidades autenticadas o autentificadas. En muchas situaciones se podría dar el caso de que las entidades o sujetos sobre los que se realizan las afirmaciones desearan mantener parte de la información oculta, ya sea de atributos o de autorización. Es más, en función del consumidor de la aserción, el sujeto podría querer incluir unas u otras declaraciones. Todas las partes que juegan su papel en la creación de declaraciones y emisión, transmisión o consumo de afirmaciones SAML deben de ser conscientes de que pueden existir estas restricciones de privacidad y deberán intentar resolverlas en la implementación de sus sistemas SAML.

Confidencialidad

El aspecto más importante para asegurar la privacidad de las partes implicadas en una transacción SAML es garantizar la confidencialidad. En otras palabras ¿es posible transmitir la información en una aserción SAML desde el emisor hasta un cierto grupo de receptores, y sólo a este grupo, sin hacerla accesible a ninguna otra parte? Técnicamente es factible transmitir la información de manera confidencial mediante mecanismos criptográficos. Pero se observa que el hecho de hacer invisibles los contenidos de las aserciones puede no cubrir todos los requisitos de privacidad deseados. Existen muchos casos donde el hecho de que esté disponible la información de acceso a un servicio por parte de un cierto usuario (o una dirección IP) podría romper la privacidad.

Anonimato

Actualmente no existen definiciones sobre el anonimato que satisfagan todos los posibles escenarios en los que este concepto puede ser utilizado. Una definición muy común sobre el término en cuestión guarda relación con un emisor de un mensaje. Según esta definición, el emisor será anónimo si el receptor no es capaz de conocer su identidad a partir del mensaje recibido. Sin lugar a dudas esta definición es correcta para este caso en particular, aunque sería incompleta debido a la posibilidad del receptor de acumular información a lo largo del tiempo y tras sucesivas interacciones con el emisor, tener una sólida descripción, no quizás sobre su identidad, pero sí sobre su conducta.

Esta noción es muy relevante en SAML debido al uso de las autoridades. Aunque cierto sujeto fuera “anónimo”, aún sería identificable como un miembro del conjunto de posibles sujetos dentro del dominio de cierta autoridad SAML. En el caso de que el usuario disponga de una agregación de atributos, el conjunto puede llegar a ser más pequeño.

De esa manera se puede decir que los sistemas SAML están limitados a ser, en el mejor de los casos, “parcialmente anónimos” debido al uso de las autoridades. Es decir, se limita el conjunto de usuarios a los que se puede referir. Una entidad sobre la que se ha realizado una afirmación pasa a formar parte del conjunto de entidades identificables por pura relación con la autoridad SAML. Las autoridades ubicadas en el origen (como por ejemplo autoridades de autentificación) pueden proporcionar un grado de “anonimato parcial” empleando identificadores que sólo se puedan utilizar una vez o mediante claves. Este anonimato es a lo sumo parcial porque el sujeto SAML se encuentra necesariamente confinado a un conjunto de sujetos por su relación con la autoridad SAML. Este conjunto se podría reducir (reduciéndose así el anonimato) cuando se utiliza la agregación de atributos para acotar aún más el subconjunto de los posibles usuarios del sitio origen. Los usuarios que verdaderamente se preocupan sobre el anonimato deben tener la precaución de “disfrazar” o evitar patrones no usuales de conducta que pudieran servir para que los usuarios pierdan su anonimato con el paso del tiempo.

Autenticación en SAML

SAML permite crear declaraciones o afirmaciones sobre que cierto proceso de autenticación se ha llevado a cabo, pero no incluye como requisito ni especifica los procesos de autenticación en sí, tan sólo que se llevaron a cabo. Los consumidores de las aserciones de autenticación deberían ser cautos y no confiar ciegamente en estas afirmaciones a menos que conozcan los fundamentos sobre los que fueron hechas. La confianza en las aserciones nunca debe ser mayor que la confianza que llevó a crearla por parte de la entidad que la realizó.

Modelo de confianza

En muchos casos, la seguridad de una conversación SAML dependerá del modelo de confianza subyacente, el cual está típicamente basado en una infraestructura de gestión de claves (por ejemplo, PKI o clave secreta). Por ejemplo, los mensajes SOAP que incorporan firmas digitales utilizando los mecanismos descritos por la especificación W3C XML Digital Signature serán seguros en el grado en que las claves utilizadas para generar la firma sean de confianza. Si éstas estuvieran comprometidas la firma digital hecha sobre cierto conjunto de aserciones SAML no serían, desde el punto de vista de seguridad, de confianza. No detectar que las claves han sido comprometidas o que ciertos certificados han sido revocados podría provocar una brecha en la seguridad. Incluso un fallo por no requerir un certificado podría crear riesgos de ataques de suplantación de identidad.

Modelo de amenaza

A continuación se analiza el modelo de amenaza para los sistemas que utilizan SAML, tomando como ejemplo un modelo en el que se establece la comunicación entre las partes que conforman SAML (Proveedor de Servicios y Proveedor de Identidad) y el atacante interfiere sobre el canal de comunicaciones.

Fig. 3 Modelo de amenaza

Además, debido a la naturaleza de SAML como sistema de autentificación de múltiples partes y como protocolo intercambio de declaraciones de autorización, se deben considerar aquellos casos en los que uno o más de los interlocutores de una transacción SAML legítima, que operan legítimamente dentro de su rol asignado para la transacción, intentan utilizar de manera maliciosa en posteriores transacciones la información conseguida en transacciones previas.

Posibles ataques a SAML y contramedidas

- **Repetición de Mensajes**

Amenaza: Hay poca vulnerabilidad en el ataque de repetición de mensajes en el nivel del enlace SOAP. La preocupación principal a este ataque es la negación de servicio.

Contramedidas: En general, la mejor manera de prevenir el ataque de repetición es evitar la captura del mensaje. Algunos de los esquemas del nivel de transporte proporcionan confidencialidad en el tránsito del mensaje. Por ejemplo, si el intercambio petición/respuesta de SAML ocurre sobre SOAP en HTTP/TLS, se previene de capturar los mensajes. Se observa que no hace falta entender un mensaje para capturarlo por lo que esquemas de encriptación no sirven para proteger contra repeticiones. Si un atacante puede capturar una petición SAML que ha sido firmada por el solicitante y cifrada al respondedor, entonces, el atacante, puede repetir la petición de nuevo en cualquier momento sin necesidad de deshacer el cifrado. La petición SAML incluye información sobre el tiempo de emisión de dicha petición, que permite saber si es una repetición. Otra manera de saber si es una repetición es controlar el identificador de la petición.

- **Cancelación de Mensajes**

Amenaza: El ataque de cancelación de mensajes evitaría que una petición alcanzara a un respondedor, o evitaría que la respuesta alcanzara al solicitante.

Contramedidas: En cualquier caso, el enlace SOAP no trata esta amenaza. En general, la correlación de los mensajes de la petición y de respuesta puede disuadir tal ataque, por ejemplo usando el atributo “In Response To”.

1.6.5.3.3 Criptografía en SAML

Se propone el algoritmo criptográfico RSA por las características que a continuación se analizan y por ser el algoritmo seleccionado por la subdirección de tecnología para la realización de la primera versión de ACAXIA.

Criptografía de clave pública

Según la Real Academia Española la criptografía es el arte de escribir con clave secreta o de un modo enigmático. El objetivo de la criptografía es garantizar la seguridad en la comunicación entre dos entidades, además de asegurar que la información que se transmite es auténtica. La criptografía de clave pública también llamada asimétrica consiste en el uso de un par de claves diferente, privada y pública, que se le atribuye a una persona determinada y tiene como principales características:

- La clave privada debe permanecer secreta, es conocida sólo por la persona a quien se le ha atribuido el par de claves y se va a utilizar para cifrar los mensajes, mientras que la clave pública puede ser conocida por cualquier individuo.
- Ambas claves sirven para cifrar y descifrar mensajes.
- A partir de la clave pública no se puede obtener ni deducir la privada.

Algoritmo RSA

El algoritmo RSA fue creado en 1978 y debe su nombre a sus tres inventores Ronald Rivest, Adi Shamir y Leonard Adleman, estuvo bajo patente de los Laboratorios RSA hasta el 20 de septiembre del 2000. Es considerado uno de los algoritmos más seguros de la actualidad.

Este algoritmo puede ser usado para la encriptación de llave pública y firma digital. Se basa en la dificultad para factorizar grandes números. La clave privada y pública se obtiene a partir del producto de dos números primos grandes.

Si se desea generar el par de claves (Kp, Kp), se eligen aleatoriamente dos números primos grandes, p y q. Luego se calcula el producto n = pq. Después se elige un número primo *e* relativo con (p - 1) y (q – 1). (n, p) será la clave pública. Nótese que debe tener inversa mod (p -1) (q - 1), por lo que existirá un número d tal que de = 1 (mod (p - 1) (q - 1)) decir que d es la inversa de e. (d, n) será la clave privada. Se puede calcular esta inversa mediante el Algoritmo Extendido de Euclides. Para llevar a cabo el cifrado se emplea la expresión c = me mod(n) y para el descifrado se utilizará m = cd mod(n).

Hoy en día RSA es el algoritmo asimétrico de cifrado más usado, tanto en conexiones de Internet y protocolos seguros, como en cifrado de datos. Las longitudes de clave usadas hoy en día varían desde los 512 bits hasta los 4096 bits, aunque se suelen tomar de forma habitual claves de 1024 bits puesto que las de 512 bits no se consideran suficientemente seguras.

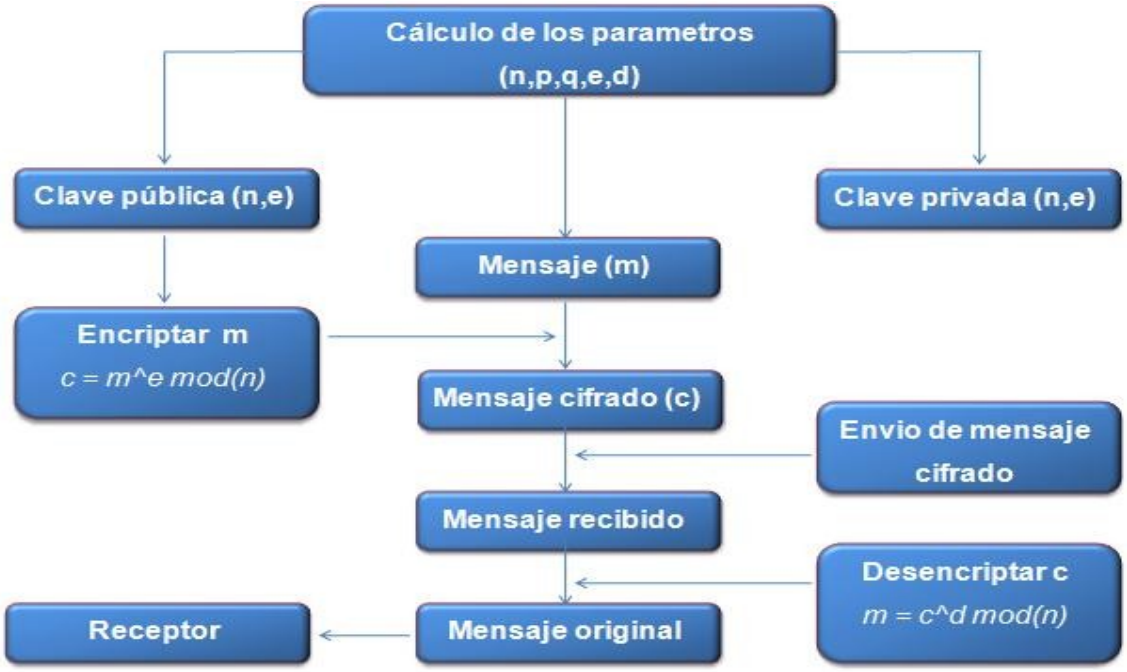


Fig. 4 Diagrama RSA

Ventajas

- Permite encriptar y firmar digitalmente.
- Utilizado conjuntamente con DES otorga una mayor velocidad de operación.
- La clave DES, empleada por RSA, es válida para un único mensaje, en el peor de los casos en que se logre 'quebrar' la clave, ésta no se puede aplicar para otro mensaje o documento.
- Permite además la detección de:
  1. Alteraciones en los documentos.
  2. Errores en la transmisión de documentos.
- Es un estándar internacional.

Desventajas

- **Claves demasiadas cortas**

Para que se considere segura una clave de RSA se recomienda que tenga una longitud de n no menor que 1024 bits. Esto se debe que en 1999 Adi Shamir desarrolló un dispositivo capaz de factorizar números con gran rapidez lo que ponía en peligro los mensajes encriptados con una clave con longitud menor o igual que 512 bits. La longitud de la clave está en función del tiempo que se quiera mantener la información en secreto, esto se debe al nivel de desarrollo que ha alcanzado la tecnología.

- **Firmar y codificar**

Nunca se debe firmar un mensaje después de ser codificado, sino todo lo contrario, porque existen ataques que dan la posibilidad de manipular mensajes primero codificados y luego firmados.

Detección de documentos alterados y/o errores de transmisión usando RSA

Una firma digital RSA es superior a una firma manuscrita ya que atestigua el contenido del mensaje y, al mismo tiempo, la identidad del firmante. Siempre que se utilice la función segura de "hash", no hay posibilidades de quitarle la firma a un documento y adjuntársela a otro o de alterar el mensaje firmado.

El más mínimo cambio en un documento firmado hará fracasar el proceso de verificación de la firma digital. El sistema de autenticación RSA permite verificar la integridad de un documento firmado. Por supuesto, si la verificación de la firma falla, no quedará claro si hubo un intento de fraguarla o si se trató de un mero error en la transmisión de la información.

Uso actual del sistema RSA

La utilización del sistema RSA está experimentando una rápida expansión y puede llegar a ser omnipresente en los próximos años. Hoy en día, se utiliza en una gran variedad de productos, plataformas e industrias del mundo entero. Se encuentra en productos comerciales de software y se planea una expansión mayor. RSA está incluido en sistemas Apple, Microsoft, Netscape, Novell, Sun, etc.

1.6.5.3.4 Por qué usar SAML

Ventajas:

- Independencia de Plataforma. SAML abstrae el framework de seguridad de arquitecturas de plataformas e implementaciones de soluciones SSO particulares. Haciendo que la seguridad sea más independiente de la lógica de las aplicaciones, esto es un principio importante de las Arquitecturas Orientadas a Servicios.

- Desacoplamiento de directorios. SAML no necesita información del usuario para mantenerse y sincronizarse entre directorios.
- Mejoramiento la experiencia en línea para los usuarios finales. SAML habilita SSO permitiéndole a los usuarios autenticarse ante el proveedor de identidades y posteriormente acceder a proveedores de servicios sin autenticación adicional. Adicionalmente, la federación de identidades (vinculación entre múltiples identidades) con SAML permite mejorar la experiencia al usuario de una manera personalizada en cada servicio promoviendo la privacidad del mismo.
- Reducción de costos administrativos para los proveedores de servicios. Usando SAML para reutilizar un único proceso de autenticación (como es el proceso de logueo con nombre de usuario y contraseña) varias veces entre varios servicios puede reducir el costo de mantener información de las cuentas. Esta carga es transferida al proveedor de identidades.
- Transferencia de riesgos. SAML puede actuar para transferir la responsabilidad de la administración adecuada de identidades al proveedor de identidades, que con más frecuencia es compatible con su modelo de negocio que en los proveedores de servicios

Por lo analizado en el presente epígrafe, porque la arquitectura de servicios web recomienda la evolución de los servicios de negocio a modulares y reutilizables y por los resultados que muestra la tabla a continuación arrojados luego de un estudio que se hizo de las ventajas que traía utilizar el estándar para la realización del componente de autenticación, se propone que SAML sea el estándar a utilizar.

Resultados del estudio realizado al aplicarle SAML al componente de autenticación del Sistema de Gestión Integral de Seguridad (ACAXIA).

Antes	Después
No hay un estándar implantado	Se rige por el estándar SAML
No hay control de las sesiones	Control de sesiones
Cifrado simple de datos	Se obtiene un token de seguridad cifrado de la autenticación
Solo maneja la solicitud por servicio web	Se utiliza además la solicitud por HTTP-Post
Poca integración en las aplicaciones	Se obtendría mayor integración
El servidor de aplicaciones estaba unido al de sesiones	Se tratarían de forma independiente

Fig. 5 Ventajas de usar SAML

1.7 Resultados esperados u objetivos de la solución propuesta

Como resultados esperados u objetivos de la solución propuesta, se tiene un sistema de seguridad capaz de gestionar la autenticación en un entorno de varias aplicaciones, guiado por el estándar internacional SAML basado en el intercambio de información en formato XML. Constituyendo este una facilidad para garantizar la seguridad de forma centralizada a cualquier sistema al que se le brinde servicios, contribuyendo en gran medida a la integración del sistema de seguridad y brindando nuevas funcionalidades.

1.8 Conclusiones Parciales

Como conclusiones al capítulo se tiene que se hizo un estudio en el que se definieron y fundamentaron los elementos teóricos para el análisis y diseño del problema a resolver y que arrojo la metodología, el lenguaje, la tecnología y herramientas a utilizar y las ventajas que conlleva desarrollar el componente de autenticación usando el estándar SAML. La nueva versión del Sistema de Gestión Integral de Seguridad (ACAXIA), tendrá una mayor integración con otros sistemas y su componente de autenticación estará totalmente desacoplado de los restantes procesos lo que quiere decir que podrá ejecutarse de forma ejecutarse de forma independiente.

Capítulo 2: Desarrollo de la solución

2.1 Introducción

En este capítulo se analiza la arquitectura que tendrá el componente de autenticación y se hace un levantamiento de los requisitos funcionales y no funcionales del componente desarrollado. Se desarrolla el modelo de análisis y diseño, con los artefactos correspondientes a cada uno y se describen los artefactos de implementación de la solución en cuestión

2.2 Arquitectura de Software

En los inicios de la informática, la programación se consideraba un arte y se desarrollaba como tal, debido a la dificultad que entrañaba para la mayoría de las personas, pero con el tiempo se han ido descubriendo y desarrollando formas y guías generales, en base a las cuales se puedan resolver los problemas. A estas, se les ha denominado Arquitectura de Software, porque a semejanza de los planos de un edificio o construcción, estas indican la estructura, funcionamiento e interacción entre las partes del software. En el libro "An introduction to Software Architecture", David Garlan y Mary Shaw[23] definen que la Arquitectura es un nivel de diseño que hace foco en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema".

2.3 Escenarios Arquitectónicos

- Federación de Identidades.
- Varios drivers de autenticación.
- Single Sign-On.

2.3.1 Federación de Identidades

La identidad federada es una de las soluciones para abordar la gestión de identidad en los sistemas de información. El valor añadido adicional respecto a otras soluciones es la gestión de identidad

[23] ***David Garlan y Mary Shaw:*** autores del libro An introduction to Software Architecture y son los que plantean la necesidad de la Arquitectura de Software como disciplina científica. En el libro se introduce por primera vez el término Estilos Arquitectónicos (tubería y filtros, repositorio, arquitectura en capas, etc.)

interdependiente entre empresas, lo que se denomina Administración de Identidad Federada (Federated Identity Management).

Como cualquier solución de Gestión de Identidad, su objetivo es obtener una gestión de usuarios eficiente, la sincronización de los datos identificativos, gestión de acceso, servicios de agrupación, servicios de directorio, auditoría e informes.

Mediante soluciones de Identidad Federada los individuos pueden emplear la misma identificación personal (típicamente usuario y contraseña) para identificarse en redes de diferentes departamentos o incluso empresas. De este modo las empresas comparten información sin compartir tecnologías de directorio, seguridad y autenticación, como requieren otras soluciones (meta directorio, Single Sign-On, etc.). Para su funcionamiento es necesaria la utilización de estándares que definan mecanismos que permiten a las empresas compartir información entre dominios. El modelo es aplicable a un grupo de empresas o a una gran empresa con numerosas delegaciones y se basa en el "círculo de confianza" de estas, un concepto que identifica que un determinado usuario es conocido en una comunidad determinada y tiene acceso a servicios específicos.

Arquitectura del escenario Federación de Identidades

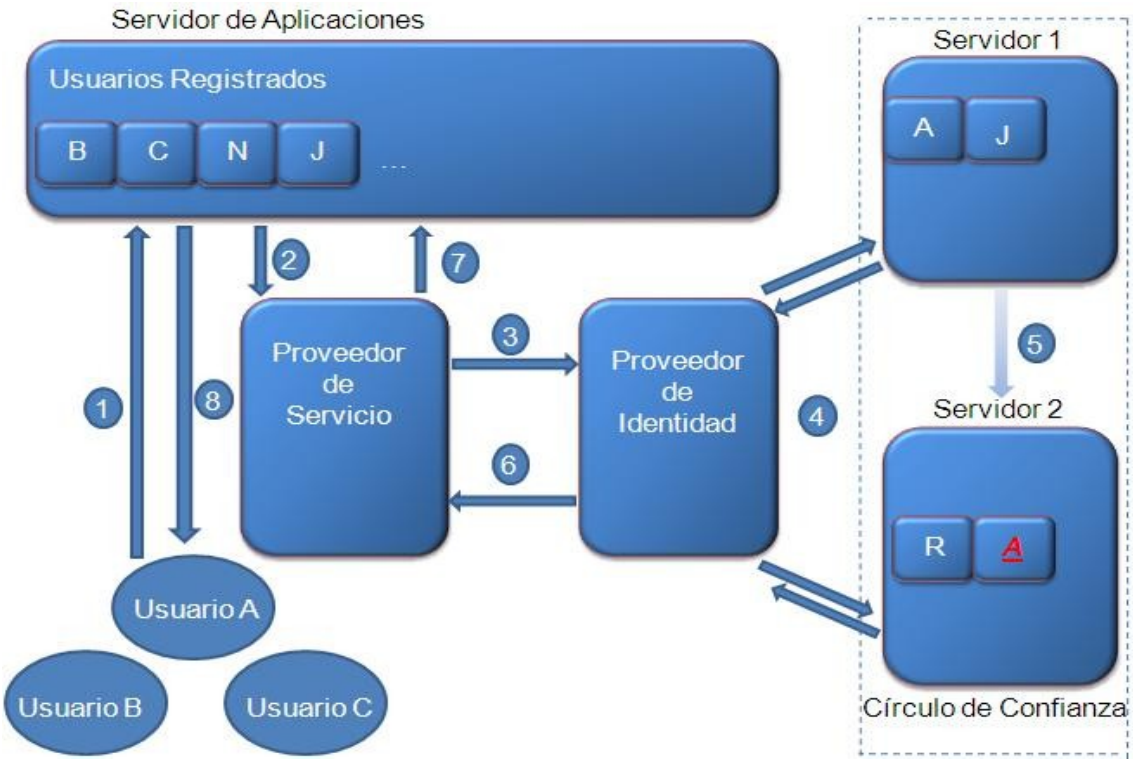


Fig. 7 Modelo de Federación

- (1)-Petición de Recurso.
- (2)-Petición de Autenticación.
- (3)-Comunicación SAML (Petición de Autenticación).
- (4)-Intercambio de Credenciales.
- (5)-Federación de Identidad (si es necesario).
- (6)-Comunicación SAML (Respuesta a Petición de Autenticación).
- (7)-Respuesta a Petición de Autenticación.
- (8)-Acceso o Rechazo.

2.3.2 Varios drivers de autenticación

Se propone la implementación de drivers para la autenticación contra varios tipos de servidores como se muestra a continuación en la figura.

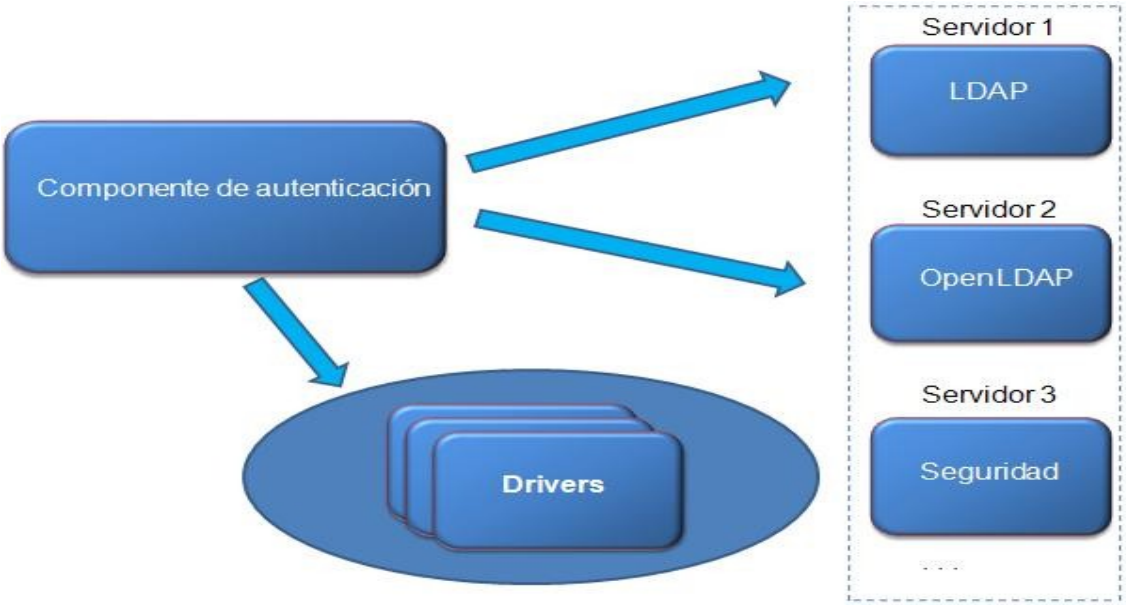


Fig. 8 Drivers de autenticación

2.3.3 Single Sign-On

Single Sign-On es un procedimiento de autenticación que se refiere al acceso a múltiples recursos por medio de un único proceso de ingreso. Gran cantidad de las arquitecturas implementadas en diferentes organizaciones, han sido diseñadas con el objetivo de dar acceso a los usuarios a múltiples

servicios Web y/o aplicaciones. En la mayoría de los casos, se encuentra que cada uno de los servicios o aplicaciones cuenta con su propio componente de seguridad, lo que generalmente compromete la seguridad de todo el sistema; dado que el nivel de seguridad de todo un sistema es igual al nivel de seguridad del componente más inseguro que lo compone. Una de las posibles soluciones a este problema es implementar la estrategia SSO.

El principal objetivo de una arquitectura que implemente Single Sign-On es transferir la funcionalidad y complejidad de todos los componentes de seguridad a un solo servicio de Single Sign-On (SSO). En una arquitectura SSO, todos los mecanismos de seguridad se encuentran concentrados en el SSO, siendo éste el único punto de autenticación y registro en el sistema. Otro beneficio de una arquitectura SSO, es que los usuarios deben hacer el proceso de ingreso una sola vez, aunque continúen

Una implementación real de SSO, debe contar con un agente SSO que normalmente se encarga de almacenar en una base de datos o directorio protegido las contraseñas que le permiten al usuario acceder a cada una de las aplicaciones o servicios, en el momento que lo desee. Esto se debe a que el proceso de logueo se realiza de manera transparente para el usuario, una vez que éste ha sido autenticado por medio de la arquitectura SSO.

Tipos de sistemas Single Sing-On

Existen cuatro tipos principales de sistemas SSO a nivel mundial, también conocidos como Sistemas de Autenticación Reducida (Reduced Sign-On Systems). Los cuales son:

- Enterprise Single Sign-On (E-SSO): también llamado Legacy Single Sign-On, el cual funciona luego de una autenticación primaria, interceptando los requerimientos de autenticación presentados por las aplicaciones secundarias para completarlos con el usuario y la contraseña. Los sistemas E-SSO permiten interactuar con sistemas que pueden deshabilitar la presentación de la pantalla de logueo.
- Web Single Sign-On (Web-SSO): conocido como Web Access Management (Web-AM), trabaja sólo con aplicaciones y recursos que se acceden vía Web. Los accesos son interceptados con la ayuda de un servidor Proxy o de un componente instalado en el servidor Web destino. Los usuarios no autenticados que tratan de acceder son redirigidos a un servidor de autenticación y regresan sólo después de haber logrado un acceso exitoso. Se utilizan cookies, para reconocer aquellos usuarios que acceden y su estado de autenticación.
- Kerberos: es un método popular de externalizar la autenticación de los usuarios. Los usuarios se registran en el servidor Kerberos y reciben un ticket, que luego utilizan para obtener acceso.

- Federation: es una nueva manera de concebir este tema, también para aplicaciones Web. Utiliza protocolos basados en estándares para habilitar que las aplicaciones puedan identificar los clientes sin necesidad de autenticación redundante.
- OpenID: es un proceso de SSO distribuido y descentralizado donde la identidad se compila en una URL de forma que cualquier aplicación o servidor pueda verificar.

Arquitectura Single Sing-On

Fig. 9 Arquitectura SSO

Clasificación de las arquitecturas Single Sing-On

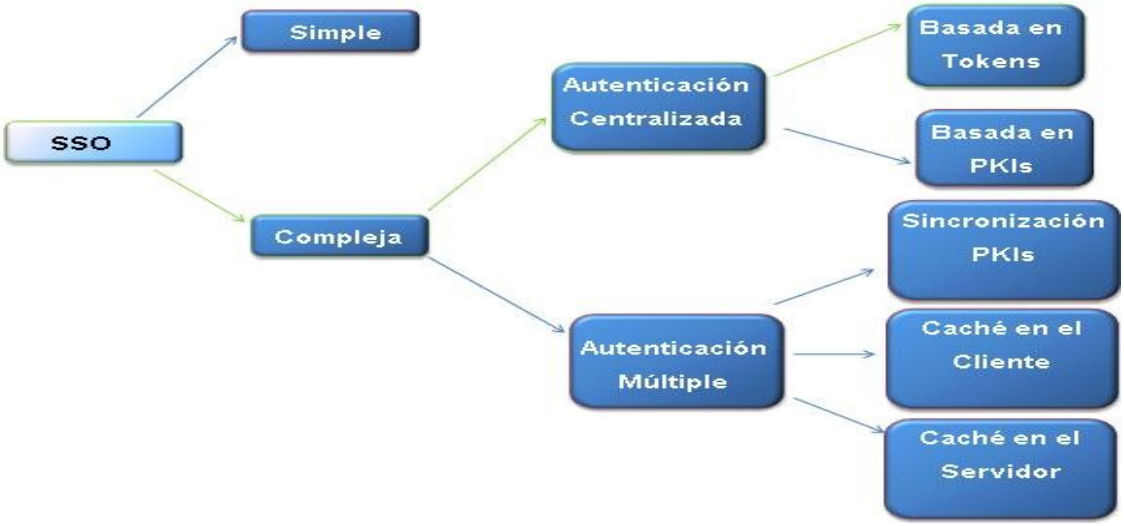


Fig. 10 Arquitecturas SSO

2.4 Propuesta de Solución

- Se propone para el desarrollo del componente de autenticación guiado por el estándar SAML:
- Utilización de PHP nativo para el Proveedor de Servicio, aumentando así la usabilidad del componente y asegurando la implementación de una arquitectura distribuida.
- Implementación de un formulario que servirá para asegurar el envío de datos por POST.
- Envío de datos a través del navegador asegurando así el escenario arquitectónico SSO.
- Implementación de una clase cliente (Service_Provider_Client) para cada aplicación.
- Uso de RSA como algoritmo de encriptación.
- Utilización de un token de seguridad como identificador del cliente.
- Se proponen dos soluciones (Arquitectura distribuida, Arquitectura centralizada).

Arquitectura distribuida del componente de autenticación

Fig. 11 Propuesta de Solución (Arquitectura distribuida)

Arquitectura centralizada del componente de autenticación

Fig. 12 Propuesta de Solución (Arquitectura centralizada)

2.5 Requisitos de la solución propuesta

En el documento Requisitos funcionales del Sistema de Gestión Integral de Seguridad (ACAXIA) se muestra una descripción detallada de los requisitos funcionales del sistema de autenticación.

2.5.1 Requisitos Funcionales

El componente de autenticación tiene un solo requisito funcional, pero se ha querido descomponer este requisito en un conjunto de funcionalidades que debe implementar este requisito. Estas funcionalidades están organizadas a su vez en dos componentes que son los conceptos que se manejan en el estándar SAML (Proveedor de Servicios y el Proveedor de Identidad), seguidamente se plantearan las funcionalidades correspondientes a cada componente.

- **R1 Gestionar la autenticación del usuario**

Requisitos Funcionales del Proveedor de Servicio

- R1.1 Crear XML Authentication Request (A.R.)[24].
- R1.2 Validar certificado.
- R1.3 Validar XML SAML Response (S.R.)[25].

[24] ***XML Authentication Request:*** Primer XML que se crea en la comunicación SAML.

Requisitos Funcionales del Proveedor de Identidad

- R1.4 Crear XML SAML Response (S.R.).
- R1.5 Crear sesión.
- R1.6 Crear certificado digital.
- R1.7 Gestionar el tipo de autenticación.
- R1.8 Validación de los datos de logueo.
- R1.9 Validar XML Authentication Request.

R1 Especificación del requisito gestionar la autenticación del usuario

Conceptos tratados	**Conceptos**	**Atributos**
	Autenticación o autentificación.	Usuario, contraseña.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Haberse establecido la comunicación SAML.	No procede.
Descripción	El sistema muestra la ventana de autenticación si el usuario no se ha autenticado previamente.	
Validaciones	Usuario y contraseña correctos.	
Post-condiciones	El usuario se ha autenticado satisfactoriamente en el sistema.	
Post-requisito	No procede.	

R1.1 Especificación del requisito crear XML Authentication Request

Conceptos tratados	**Conceptos**	**Atributos**
	XML Authentication Request.	Id (Token), nombre, url emisor, url destino, fecha y hora, url recurso solicitado, protocolo de comunicación.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	El sistema es el que inicia el requisito al usuario solicitar un recurso.	

[25] ***XML SAML Response:*** XML que se crea en respuesta al XML Authentication Request en la comunicación SAML.

Validaciones	No procede.
Post-condiciones	Se ha creado el primer XML (Authentication Request) en la comunicación SAML.
Post-requisito	No procede.

R1.2 Especificación del requisito validar certificado

Conceptos tratados	**Conceptos**	**Atributos**
	Certificado digital.	Token de seguridad encriptado.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	El sistema verifica si el cliente posee un certificado emitido por el proveedor de identidad, de ser así este verifica que sea un certificado válido.	
Validaciones	No procede.	
Post-condiciones	Se ha verificado la validez del certificado.	
Post-requisito	No procede.	

R1.3 Especificación del requisito validar XML SAML Response

Conceptos tratados	**Conceptos**	**Atributos**
	XML SAML Response.	Nombre, id, id del XML que inicia, fecha y hora, fecha y hora del XML que inicia, no antes de, no después de, url destino.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	El sistema verifica si la estructura del XML es correcta, de ser así verifica que el emisor y los datos sean válidos.	
Validaciones	No procede.	
Post-condiciones	Se ha verificado la validez del XML.	
Post-requisito	No procede.	

R1.4 Especificación del requisito crear XML SAML Response

Conceptos	**Conceptos**	**Atributos**

tratados	XML SAML Response.	Nombre, id, id del XML que inicia, fecha y hora, fecha y hora del XML que inicia, no antes de, no después de, url destino.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	El sistema luego de iniciar la comunicación SAML, y haber creado y enviado el XML Authentication Request, crea el segundo XML en la comunicación SAML (SAML Response).	
Validaciones	No procede.	
Post-condiciones	Se ha creado el SAML Response.	
Post-requisito	No procede.	

R1.5 Especificación del requisito crear sesión

Conceptos tratados	**Conceptos**	**Atributos**
	Sesión.	Id de sesión.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	El sistema crea la sesión en el servidor y la persiste en base de datos.	
Validaciones	No procede.	
Post-condiciones	Se ha creado la sesión.	
Post-requisito	No procede.	

R1.6 Especificación del requisito crear certificado digital

Conceptos tratados	**Conceptos**	**Atributos**
	Certificado digital.	Token de seguridad encriptado.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	El sistema luego de haber verificado los datos del usuario crea el certificado digital.	
Validaciones	No procede.	

Post-condiciones	Se ha creado el certificado digital.
Post-requisito	No procede.

R1.7 Especificación del requisito gestionar tipo de autenticación

Conceptos tratados	**Conceptos**	**Atributos**
	Tipo de autenticación.	Usuario y contraseña, huella dactilar, retina.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	El sistema en dependencia del recurso solicitado verifica su tipo de autenticación y muestra la interfaz pertinente.	
Validaciones	No procede.	
Post-condiciones	Se ha mostrado la interfaz pertinente.	
Post-requisito	No procede.	

R1.8 Especificación del requisito validación de los datos de logueo

Conceptos tratados	**Conceptos**	**Atributos**
	Datos de logueo.	Usuario y contraseña, huella dactilar, retina.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	El sistema en dependencia de la interfaz mostrada, valida y verifica los datos de logueo en base de datos.	
Validaciones	No procede.	
Post-condiciones	Se ha mostrado la interfaz pertinente.	
Post-requisito	No procede.	

Especificación del requisito validar XML Authentication Request

Conceptos tratados	**Conceptos**	**Atributos**
	XML Authentication Request.	Id (Token), nombre, url emisor, url destino, fecha y hora, url del recurso solicitado, protocolo de comunicación.

Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	El sistema confirma la validez del XML, así como la de su emisor y sus datos.	
Validaciones	No procede.	
Post-condiciones	Se ha validado el primer XML (Authentication Request) en la comunicación SAML.	
Post-requisito	No procede.	

2.5.2 Requisitos no Funcionales

Usabilidad (USB)

El componente podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.

Rendimiento (REN)

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 3 segundos.

Seguridad (SEG)

Autenticación (Contraseña de acceso). Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Portabilidad (POR)

El componente debe ser multiplataforma, haciendo énfasis en Linux y Windows.

Soporte (SOP)

La aplicación contará antes de su puesta en marcha con un período de pruebas, se le dará mantenimiento, configuración y se brindará el servicio de instalación.

Políticos Culturales (CUL)

El componente solo podrá ser utilizado en territorio cubano y por las entidades autorizadas por el Ministerio de las FAR. El producto no debe contener palabras en otros idiomas. El producto debe respetar los términos empleados normalmente por los especialistas en el tema de la esfera que se automatiza.

Legales (LEG)

El componente está avalado por los tres documentos rectores emitidos en el país para la certificación y validación de los sistemas contables:

Software (SFT)

- **Para el cliente:**

  1-Navegador Mozilla Firefox.

  2-Sistema operativo Windows 98 ó superior ó Linux.

- **Para el servidor:**

  1-Sistema operativo Windows Advancer Server (2000 o superior) o Linux en cualquiera de sus distribuciones.

  2-Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible.

  3-Un servidor de base de datos PostgreSQL 8.0 o superior, se recomienda usar 8.3.9.

Hardware (HDW)

- **Para el servidor:**

  1-Requerimientos mínimos: Procesador Pentium III a 1GHz de velocidad de procesamiento y 1Gb de memoria RAM.

  2-Al menos 40Gb de espacio libre en disco duro.

  3-Tarjeta de red.

- **Para el cliente:**

  1-Requerimientos mínimos: Procesador Pentium II a 133Mhz con 128 Mb de memoria RAM.

  2-Tarjeta de red.

2.6 Prototipo de interfaz de Usuario

En este epígrafe se relaciona el requisito funcional gestionar la autenticación del usuario con su interfaz de usuario.

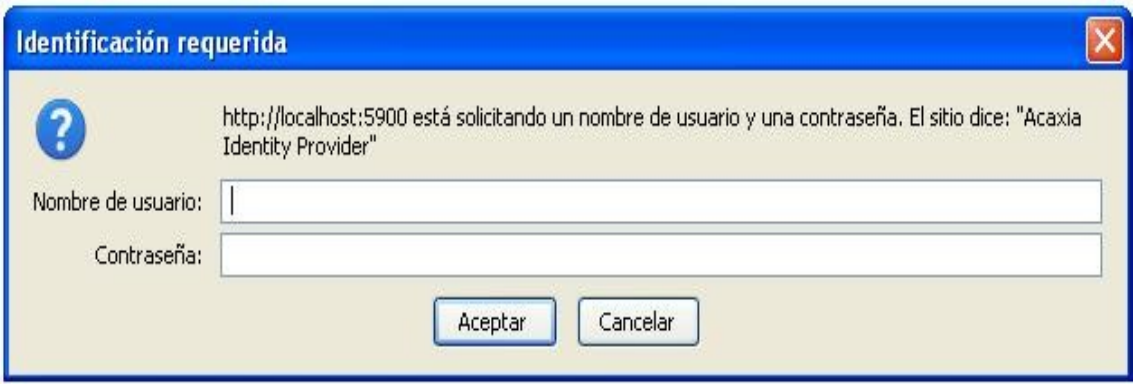


Fig. 13 Interfaz de Usuario

2.7 Modelo de Análisis

2.7.1 Modelo Conceptual

El Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. No representa términos relacionados con los componentes de software sino que se enfoca en clases conceptuales del dominio del problema y conceptos del mundo real.

Una clase conceptual puede ser una idea o un objeto físico (símbolo, definición y extensión).

En el Modelo de Dominio no se muestra comportamiento. Las clases conceptuales pueden tener atributos pero no métodos (Larman, 1999).

Estrategia para la identificación de conceptos

La meta es crear un modelo conceptual de conceptos interesantes o significativos del dominio en cuestión.

Una directriz interesante a la hora de identificar los conceptos es:

“Es mejor exagerar y especificar un modelo conceptual con muchos conceptos refinados que no especificarlo cabalmente”

Un modelo conceptual no es más adecuado si tiene menos conceptos; generalmente sucede lo contrario. Es frecuente omitir conceptos durante la fase inicial de identificación, muchos son descubiertos más tarde, cuando se examinan los atributos o asociaciones e incluso, durante la fase de diseño. Cuando se detecten, habrá que incorporarlos al modelo conceptual.

Representación del modelo conceptual

El modelo conceptual no es una descripción de los componentes del software; representa los conceptos en el dominio del problema en el mundo real .Entre los elementos que componen este diagrama pueden existir asociaciones, que no es más que una relación entre dos conceptos que indica alguna conexión significativa e interesante entre ellos (Sommerville, 2002).

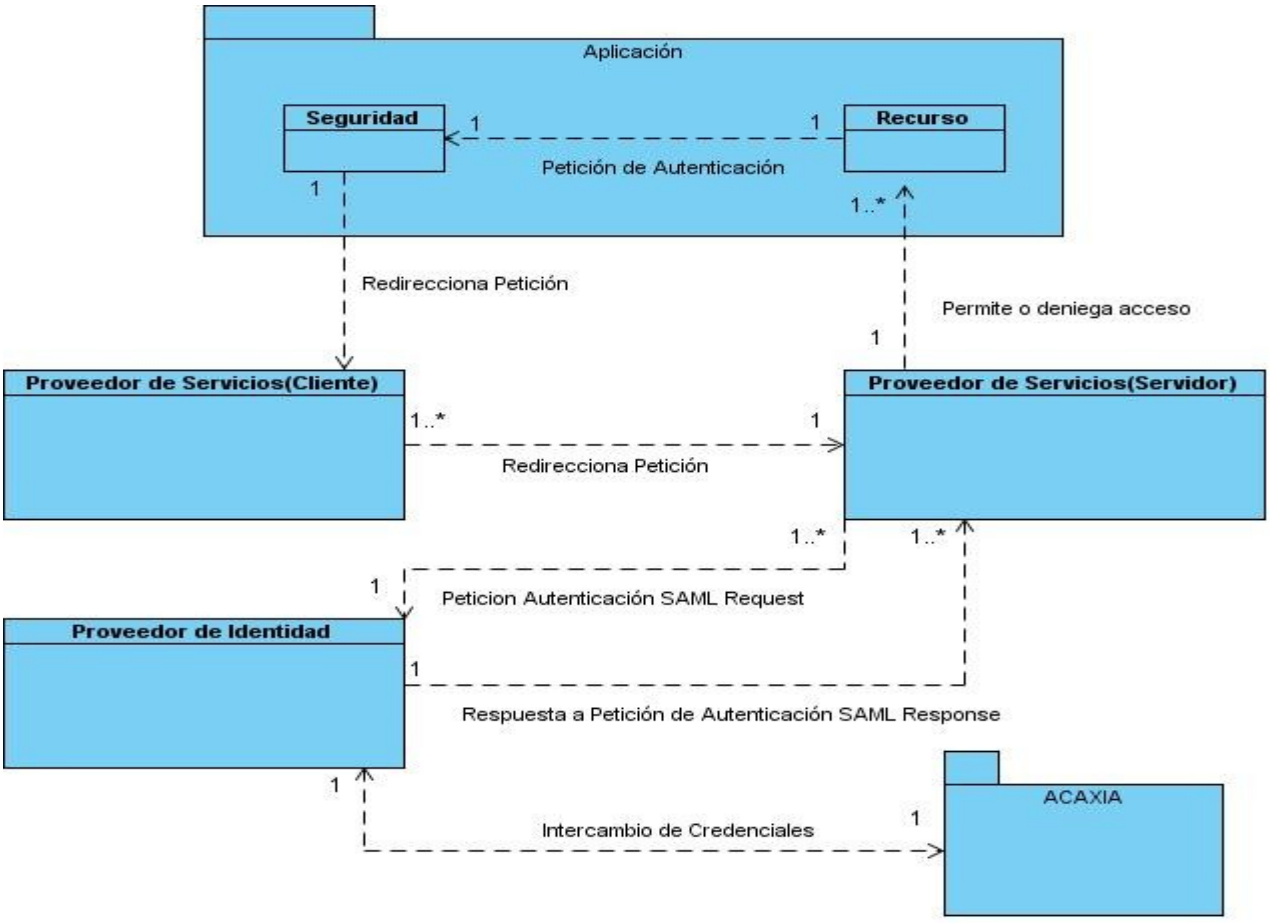


Fig. 14 Modelo conceptual.

2.8 Modelo de Diseño

2.8.1 Diagrama de clases con estereotipos web

El componente a desarrollar es Web, es decir, que tiene como elemento significativo de su arquitectura un navegador y un protocolo de comunicación HTTP. Para el modelado de este tipo de aplicaciones se aprovecha una de las características que le da flexibilidad a la notación de UML. La misma consiste en un conjunto de mecanismos de extensión. El uso de estereotipos permite

enriquecer el significado de los elementos clásicos de su notación. Las particularidades de esta, aplicadas a la web incluyen símbolos representativos de las clases en esta arquitectura así como las relaciones entre ellas.

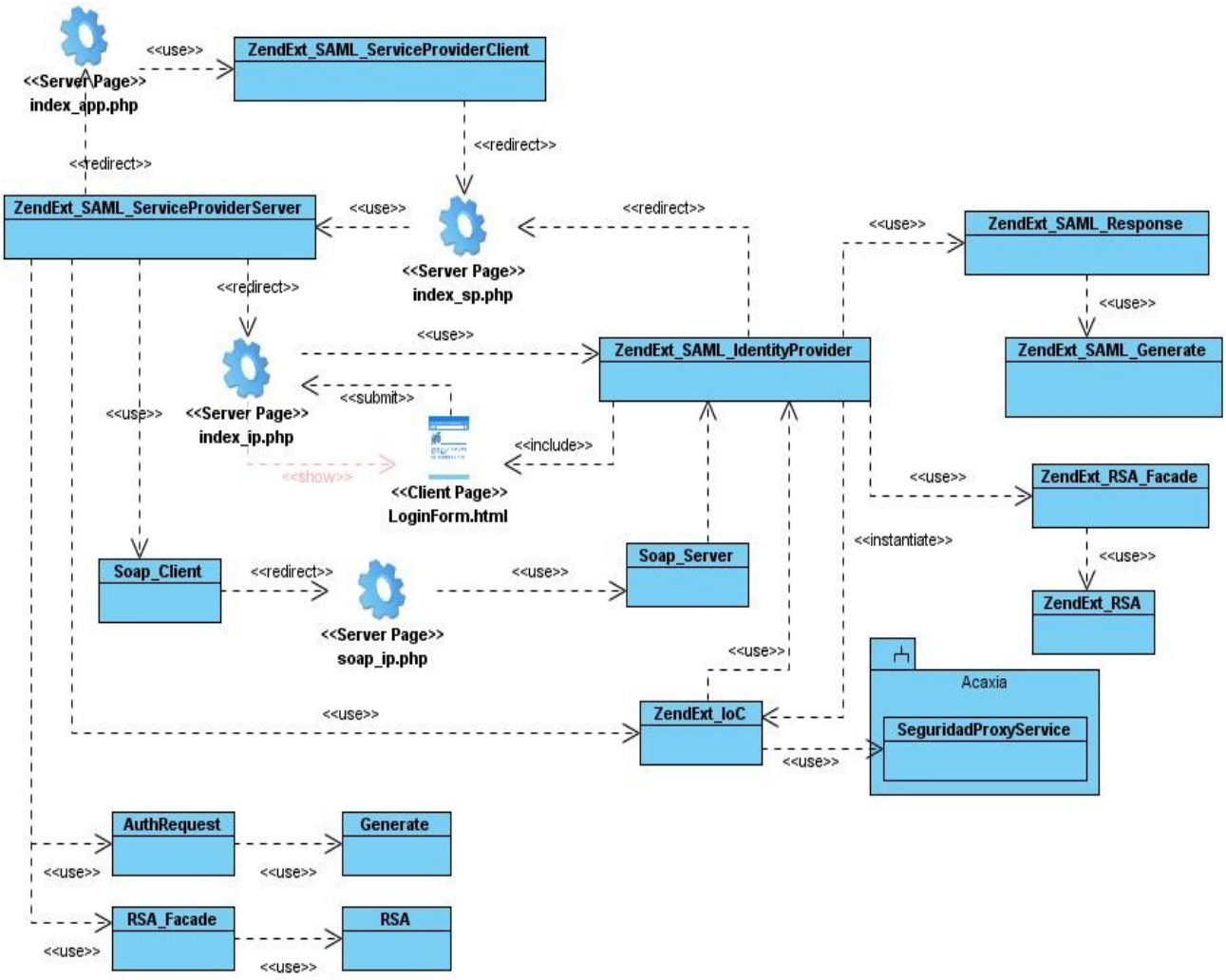


Fig. 15 Diagrama de clases

2.8.2 Descripción de clases y atributos

Nombre:	Service_Provider_Client.
Tipo de clase:	Servidora.
Para cada responsabilidad:	
Nombre:	**Descripción**
Init($request,$flag)	Método Interfaz.

loadXml()	Carga los datos del XML de configuración y los devuelve en un arreglo.
redirectToServiceProvider ($serversp,$serveridp,$app,$flag)	Construye un formulario y envía los datos por POST.

Nombre:	Service_Provider_Server.
Tipo de clase:	Servidora.
Para cada responsabilidad:	
Nombre:	**Descripción**
Init($request)	Método Interfaz.
create_AuthRequest($protocolo,$initdir,$url_self,$serveridp,$athreq,$app,$flag)	Crea el primer XML en la comunicación SAML.
check($certificado)	Chequea la validez del certificado a través de SOAP si se usa una arquitectura distribuida ó IoC si se usa una arquitectura centralizada.
redirectToIdentityProvider($request,$rsa,$athreq)	Crea un formulario y envía los datos por POST al Proveedor de Identidad.
redirectToResource($resource,$certificado)	Crea un formulario y envía los datos por POST al recurso solicitado.

Nombre:	AuthRequest.
Tipo de clase:	Entidad.
Para cada responsabilidad:	
Nombre:	**Descripción**
create_AuthRequest($protocolo, $initdir, $url_self, $serveridp,$app,$flag)	Crea el primer XML en la comunicación SAML.
data_extract($dec_xml)	Extrae los datos del XML Response y los devuelve en un arreglo.

Nombre:	Generate.
Tipo de clase:	Entidad.
Para cada responsabilidad:	
Nombre:	**Descripción**
samlCreateId()	Crea el id del primer XML en la comunicación SAML.
samlGetDateTime($timestamp)	Devuelve fecha y hora.

Nombre:	ZendExt_SAML_IdentityProvider_Server.
Tipo de clase:	Servidora.
Para cada responsabilidad:	
Nombre:	**Descripción**
init($request, $config)	Método Interfaz, trabaja con la sesión y con la configuración del sistema.
initRegister()	Se crea el registro de objetos, variables y arreglos. Se guarda el registro en el singleton de Registro.
initConfig($config)	Se guarda en el singleton la configuración del framework.
initSession()	Se inicia la sesión.
login()	Crea y devuelve el certificado digital (Método auxiliar).
createCertificate($usuario, $password)	Crea y devuelve el certificado digital.
Logout()	Desloguea al usuario.
clearOutSession ()	Limpia las variables de sesión.

Nombre:	ZendExt_SAML_Response.
Tipo de clase:	Entidad.
Para cada responsabilidad:	
Nombre:	**Descripción**
create_saml_response($id_auth,$auth_time,$url,$resource,$certificate,$type)	Crea el segundo XML en la comunicación SAML.
data_extract($dec_xml)	Extrae los datos del XML y los devuelve en un arreglo.
validSamlDateFormat($samlDate)	Valida la validez de los datos de fecha.

Nombre:	ZendExt_SAML_Generate.
Tipo de clase:	Entidad.
Para cada responsabilidad:	
Nombre:	**Descripción**
samlCreateId()	Crea el id del primer XML en la comunicación SAML.
samlGetDateTime($timestamp)	Devuelve fecha y hora.

Nombre:	ZendExt_RSA_Facade.

Tipo de clase:	Entidad.
Para cada responsabilidad:	
Nombre:	**Descripción**
encrypt($string)	Encripta los datos.
decrypt($encoded)	Desencripta los datos.

2.9 Artefactos de implementación

En este epígrafe se verá todo lo referente a los artefactos de implementación utilizados para llevar a cabo el desarrollo del componente. Dentro de estos artefactos se encuentran los diagramas de componente y despliegue y los estándares de codificación que no son más que pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Los estándares de codificación permiten una mejor integración en la producción y se establecen las pautas que conlleven a lograr un código más legible, reutilizable y fácil de dar mantenimiento en un futuro.

2.9.1 Diagrama de componentes

Este diagrama muestra las organizaciones y dependencias lógicas entre componentes de software, ya sean componentes de código fuente, binarios o ejecutables. Se emplea para mostrar una vista estática del sistema. A continuación se muestra el diagrama de componentes y su integración.

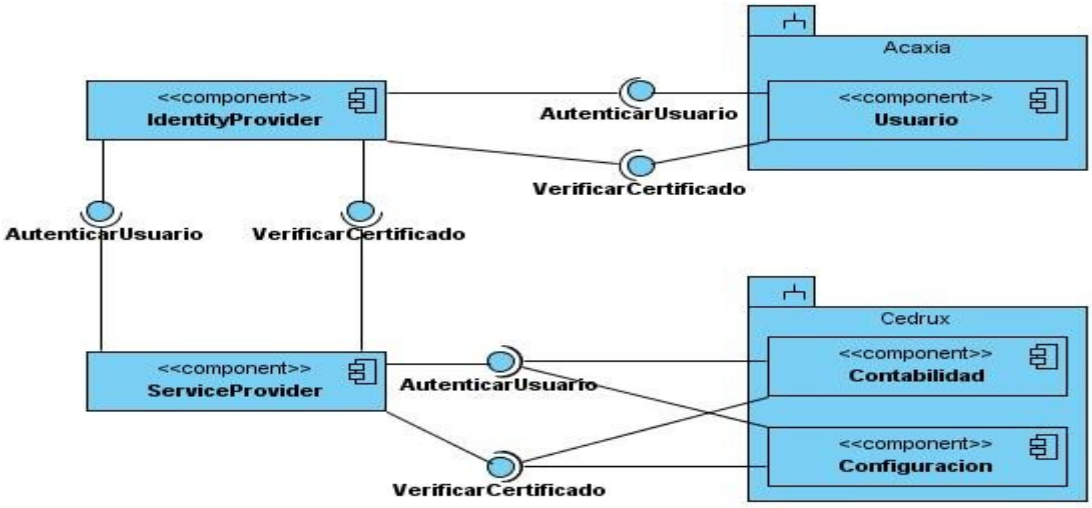


Fig. 16 Diagrama de componentes

2.9.2 Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes de hardware y software en el sistema final. Es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede reflejar componentes los cuales pueden estar conectados por relaciones de dependencia, posiblemente a interfaces. Teniendo en cuenta las características del componente los Diagramas de Componentes quedaron de la siguiente forma para las dos arquitecturas.

Diagrama de despliegue (arquitectura centralizada)

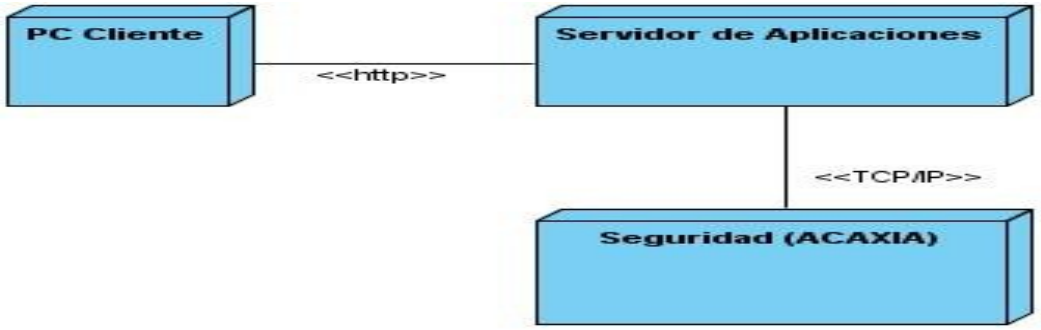


Fig. 17 Diagrama de despliegue (AC)

Diagrama de despliegue (arquitectura distribuida)

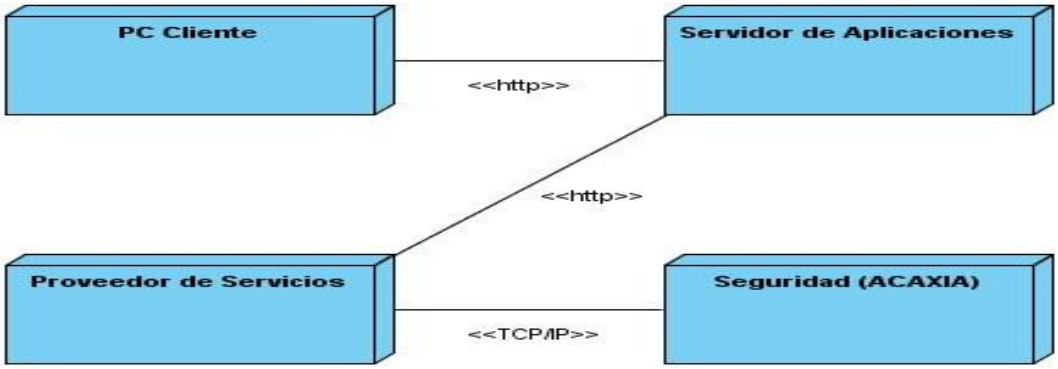


Fig. 18 Diagrama de despliegue (AD)

2.9.3 Estándares de Nomenclatura

Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará el guión bajo (under score (_)) y la primera letra seguida de este comienza con mayúscula. Con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: Encriptación_Rsa

- **Nomenclatura de las funciones**

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación Camel Casing[26], y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: function crearXML ().

Normas para comentar las clases, funciones y llaves

- **En las clases**

Antes de la declaración de una clase se escribe una breve descripción donde se explique el propósito de la misma. Quedando de la siguiente forma:

/**

* Nombre de la clase *

* Descripción *

* @author *

@package (módulo)

@subpackage (submódulo)

@copyright

@versión (versión-parche)/

- **En las funciones**

Antes de la declaración de la función se escribe una breve descripción donde se explique el propósito de la misma. Quedando de la siguiente forma:

/**

* Nombre de la función *

[26] ***Camel Case*** es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre Camel Case se podría traducir como Mayúsculas/Minúsculas Camello, aunque no es correcto en todos los contextos ya que la palabra inglesa Case no tiene traducción literal. El nombre se debe a que las mayúsculas a lo largo de una palabra en Camel Case se asemejan a las jorobas de un camello.

* Descripción *

* @author * (en caso de que no sea el autor de la clase) *

*@param *(los parámetros que se le pasan a la función con su descripción) *

*@throws *(en caso de que dispare una excepción) *

* @return *(se pone lo que devuelve la función y un comentario)*/

2.10 Conclusiones parciales

Durante el transcurso del capítulo se argumentaron los aspectos fundamentales que se llevan a cabo durante el proceso de análisis, comenzando por el levantamiento de los requisitos funcionales que contienen cada unos de los componentes que integran el componente de autenticación y de los requisitos no funcionales para garantizar una ejecución eficaz de la aplicación. También se reflejó el diseño de clases para una mejor comprensión de cómo están estructuradas las clases que conforman el componente. Se expusieron además los distintos artefactos de implementación generados, como son los diagramas de componentes y despliegue y estándar de codificación utilizado para el desarrollo.

Capítulo 3: Pruebas y validación

3.1 Introducción

En el presente capítulo se muestran los casos de prueba y se analizan las métricas que se aplican en la actualidad y cuáles de ellas se aplicaron al diseño de los componentes que integran la solución, asegurando así la calidad del producto.

3.2 Casos de Prueba

La realización de los casos de pruebas tiene como objetivo demostrar al cliente la reacción que corresponderá por parte del sistema luego de realizar alguna acción en el mismo. Para un mejor entendimiento de las respuestas o posibles funcionalidades que brindará el sistema, según la necesidad del usuario.

Caso de Prueba para R1 Gestionar la autenticación del usuario

Condiciones de ejecución

- El usuario debe haber pedido acceso a un recurso subscrito a ACAXIA.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Gestionar la autenticación del usuario.	El usuario al hacer la petición de un recurso se activa el mecanismo de autenticación a través del estándar SAML.	EP 1.1: Petición de autenticación sin certificado (token de seguridad).	– El cliente hace una petición del recurso. – Se activa el procedimiento de autenticación. – Se muestra la ventana de autenticación. – El usuario introduce sus credenciales de autenticación (usuario y contraseña). – Se crea la sesión y el certificado (token de seguridad). – Se verifica el certificado. – Si es correcto el certificado se da acceso al recurso si no se deniega el acceso.
		EP 1.2: Petición de autenticación con certificado (token de seguridad).	– El cliente hace una petición del recurso. – Se activa el procedimiento de autenticación. – Se verifica el certificado enviado por el cliente.

			– Si es válido para el recurso solicitado se da acceso al mismo de lo contrario se deniega.
		EP 1.3 Petición de autenticación de varios sistemas.	– El cliente hace una petición del recurso. – Se activa el procedimiento de autenticación. – Se hace otra petición a otro recurso y se envía el certificado creado en la primera petición, si el certificado es válido para el nuevo recurso solicitado se da acceso al sistema sin mostrar la ventana de autenticación de lo contrario se deniega el acceso.

Descripción de las variables

No	Nombre de campo	Tipo	Válido	Inválido
[1]	Usuario	Campo de texto	Nombre del usuario con roles en el sistema de seguridad ACAXIA	El usuario introducido no existe o no tiene roles asignados en el sistema de seguridad.
[2]	Contraseña	Campo de texto	Contraseña que tiene el usuario introducido anteriormente en el servidor.	La contraseña es incorrecta debido a que no coincide con la que tiene el usuario asignada en el servidor.

Juego de datos a probar

Id del escenario	Escenario	Servidor	Base de datos	Usuario	Contraseña	Respuesta del sistema	Resultado de la prueba
EP 1.1	Petición de autenticación sin certificado (token de seguridad).	V (localhost: 5900).	V (Seguridad).	V (instalación.	V (instalación.)	Se muestra la pantalla de autenticación.	Se muestra la pantalla de autenticación.
EP 1.2	Petición de autenticación con certificado (token de seguridad).	V (localhost: 5900).	V (Seguridad).	-	-	El sistema verifica el certificado y si es correcto muestra el recurso solicitado de lo contrario deniega el acceso.	El sistema verifica el certificado y si es correcto muestra el recurso solicitado de lo contrario deniega el acceso.

EP 1.3	Petición de autenticación de varios sistemas.	V (localhost: 5907.)	V (Seguridad).	-	-	El sistema verifica el certificado y si es correcto para el sistema en cuestión muestra el recurso solicitado de lo contrario deniega el acceso.	El sistema verifica el certificado y si es correcto para el sistema en cuestión muestra el recurso solicitado de lo contrario deniega el acceso.

[Las celdas de la tabla contienen V o (-). V indica válido, (-) indica vacio].

3.3 Validación de la solución

3.2.1 Valoración de la solución

Confirmación por inspección y provisión de evidencia objetiva de que los requerimientos particulares para un uso específico son alcanzados. En diseño y desarrollo, la validación está relacionada con el proceso de reexaminación de un producto para determinar la conformidad con las necesidades del usuario. La validación es realizada normalmente sobre el producto final bajo condiciones operacionales definidas. La valoración de la solución es el paso final en el proceso de evaluación del software. Donde una solución puede ser valorada de diversas maneras, debido a que existen distintos tipos de valoraciones como son:

- Valoración cualitativa.
- Valoración indirecta.
- Valoración directa.

Una valoración se realiza mediante una métrica para asignar valores de una escala (estos pueden ser un número o una categoría) al atributo de la entidad (sistema, subsistema o componente).

- Valoración cualitativa: Es una evaluación sistemática del grado o capacidad de una entidad para satisfacer necesidades o requerimientos específicos. Además se emplean categorías. Como algunos de los atributos más importantes de una entidad, se tiene: el lenguaje de desarrollo del programa (C, C ++, C #, PHP, JAVA).

- Valoración indirecta: Es la valoración de un atributo derivada del valor de uno o más atributos diferentes. Es la valoración externa de un atributo de un sistema, ejemplo: el tiempo de respuesta a la información alimentada por el usuario, es una valoración indirecta de los atributos del software, debido a que esta medida se verá influenciada por los atributos externos del sistema, así como los propios internos.
- Valoración directa: Es una valoración del producto, de forma indirecta o directa. Ejemplo: El número de líneas de código, las valoraciones de la complejidad, el número de fallas encontradas durante el proceso y el índice de señales o alertas, son todas las valoraciones internas propias del producto en sí.

Valoraciones aplicadas a la solución propuesta

Dentro de los distintos tipos de validación del software existentes se escogieron para la solución propuesta las de tipo cualitativa y directa.

Debido a los problemas actuales que existen con las licencias de software el sistema se realizó sobre plataforma libre implementándose en el lenguaje PHP utilizando el Zend Framework por decisión del departamento de tecnología del proyecto Cedrux perteneciente a CEIGE donde fue desarrollado el sistema. Se empleó como gestor de base de datos la herramienta PostgreSQL debido a que es una herramienta con características importantes como: presenta interfaz amigable y ágil de manipular. Para el acceso a datos se utilizó el framework de acceso a datos Doctrine. Todas estas herramientas fueron una decisión del departamento de tecnología del proyecto Cedrux perteneciente a CEIGE.

Métricas de software

Un aspecto importante a tener en cuenta en la fase de evaluación de la calidad del diseño ha sido la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto referenciadas por Pressman, teniendo en cuenta que este estudio brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software. Siendo esto la principal razón de la concepción de las métricas inspiradas en lo propuesto por él. Atributos de calidad que se abarcan:

- Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

- Reutilización: Consiste en el grado de reutilización que presente una clase o estructura de clase, dentro de un diseño de software.
- Acoplamiento: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a las características de Reutilización.
- Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño del componente de autenticación del Sistema de Gestión Integral de Seguridad (ACAXIA) y su relación con los atributos de calidad definidos en este trabajo son las siguientes:

- Tamaño operacional de clase (TOC): Número de métodos que tiene una clase.

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

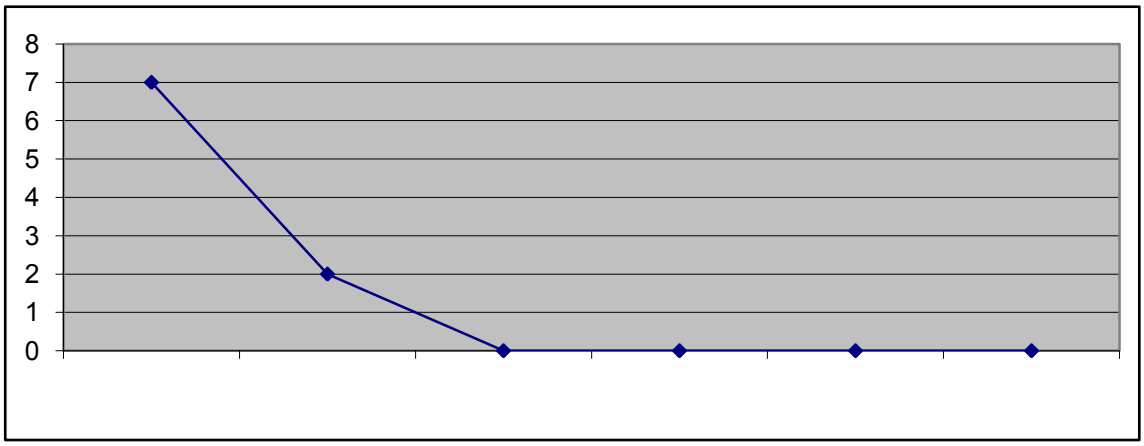
- Relaciones entre clases (RC): Esta dado por el número de relaciones de uso de una clase con otra.

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.

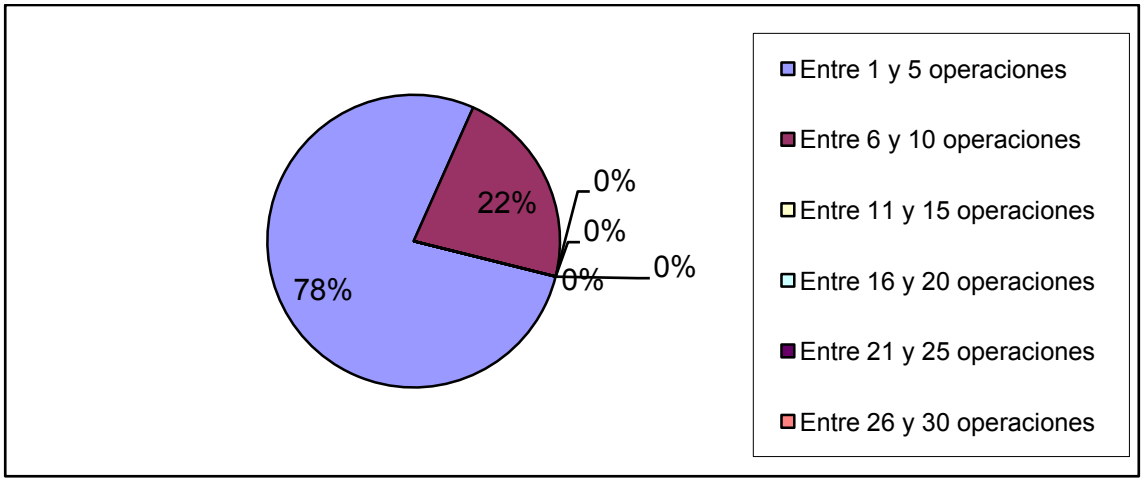
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

3.2.1.1 Resultados del instrumento de evaluación de la métrica TOC

Ver instrumentos y tablas de resultados en (Anexo 1: Instrumento de medición de la métrica Tamaño operacional de clase (TOC)).



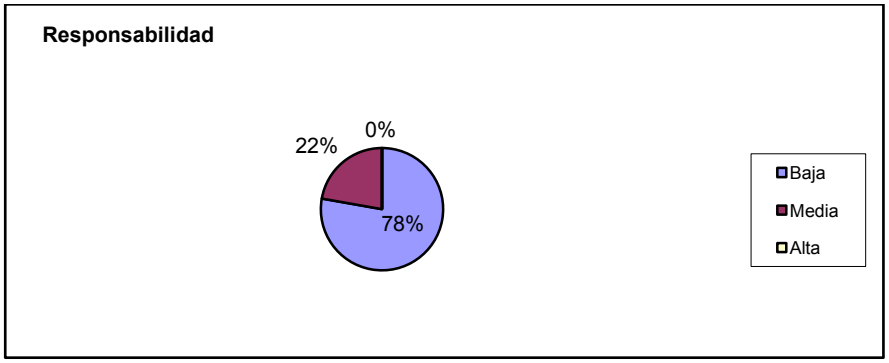
Gráfica 1. Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.



Gráfica 2. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Responsabilidad

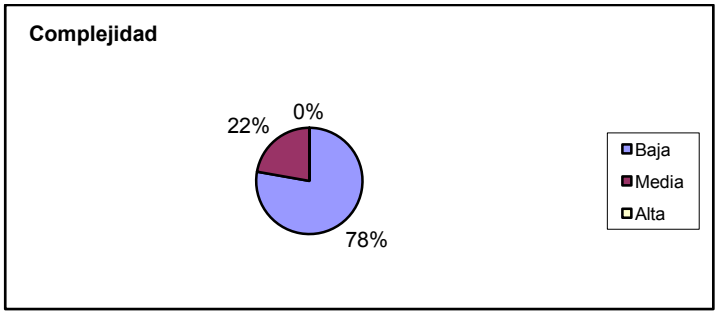
Responsabilidad	Cantidad de clases	Promedio
Baja	7	13,46153846
Media	2	3,846153846
Alta	0	0



Gráfica 3. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo responsabilidad.

Complejidad

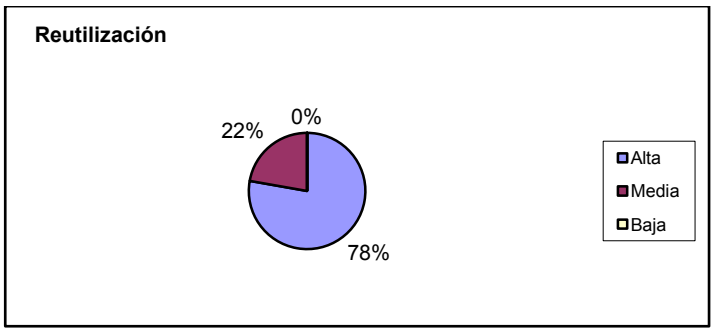
Complejidad	Cantidad de clases	Promedio
Baja	7	13,46153846
Media	2	3,846153846
Alta	0	0



Gráfica 4. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

Reutilización

Reutilización	Cantidad de clases	Promedio
Alta	7	13,46153846
Media	2	3,846153846
Baja	0	0

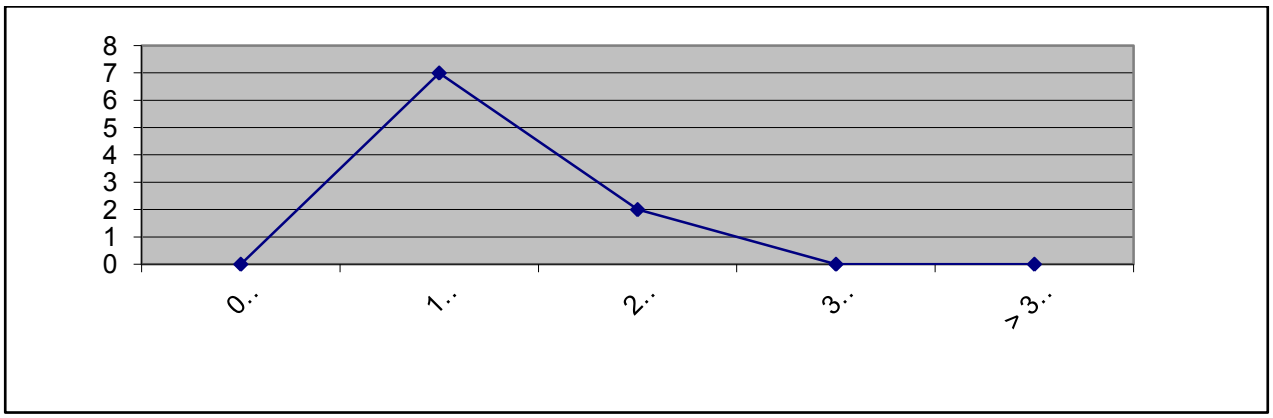


Gráfica 5. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

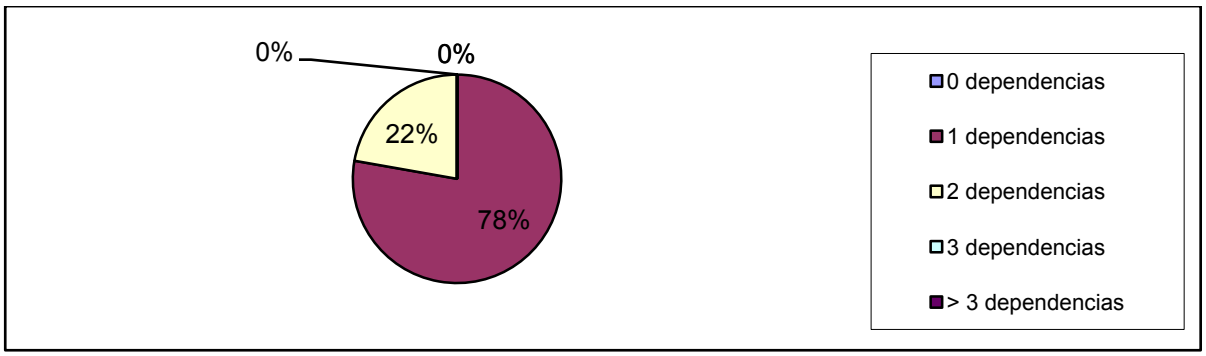
Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del componente de autenticación del Sistema de Gestión Integral de Seguridad tiene una calidad buena teniendo en cuenta que el 77,8 % de las clases incluidas en este componente posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones, en otras palabras existen 7 clases de las 9 que componen el componente de autenticación que poseen menos de 5 operaciones. Además el 100% de las clases poseen evaluaciones positivas (entiéndase calificaciones en 7 de 9 clases de baja y media en los atributos de calidad, responsabilidad y complejidad de implementación y calificaciones en 7 de 9 clases de alta y media en el atributo de calidad de reutilización).

3.2.1.2 Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC)

Ver instrumentos y tabla de resultados en (Anexo 2: Instrumento de medición de la métrica Relaciones entre clases (RC)).



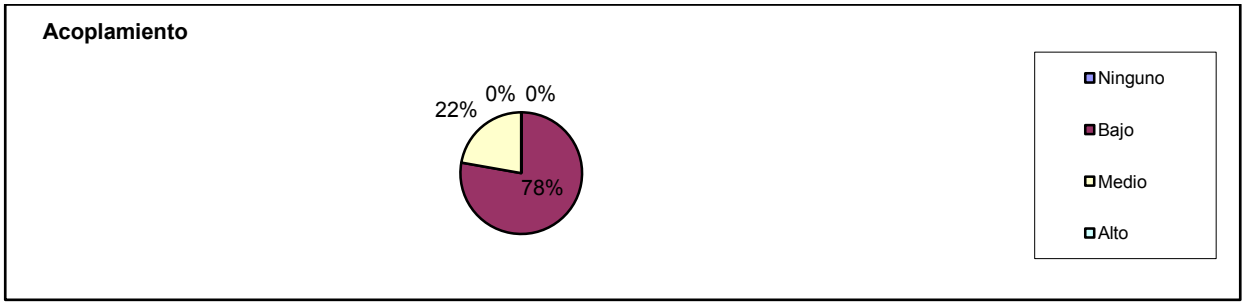
Gráfica 6. Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.



Gráfica 7. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Acoplamiento

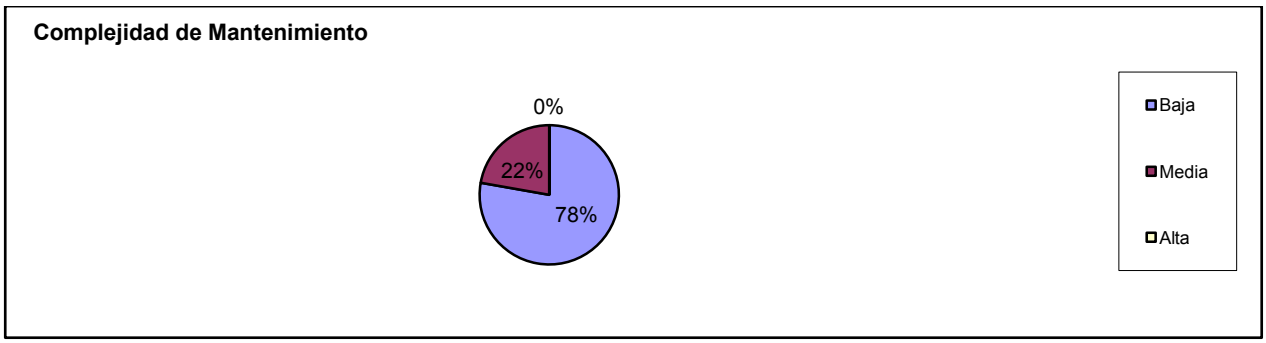
Acoplamiento	Cantidad de clases	Promedio
Ninguno	0	0
Bajo	7	8,974358974
Medio	2	2,564102564
Alto	0	0



Gráfica 8. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

Complejidad de Mantenimiento

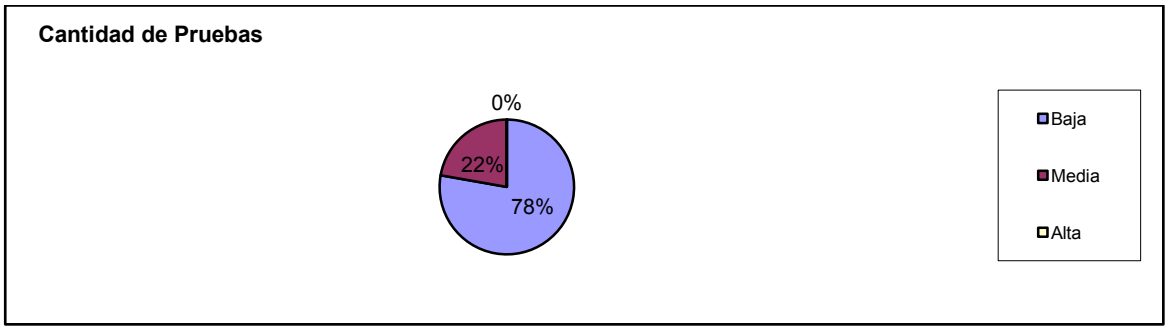
Complejidad de Mantenimiento	Cantidad de clases	Promedio
Baja	7	8,974358974
Media	2	2,564102564
Alta	0	0



Gráfica 9. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

Cantidad de Pruebas

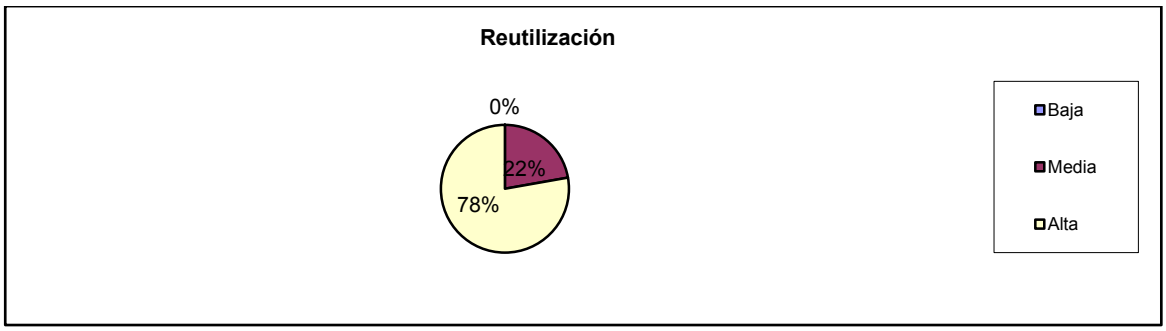
Cantidad de Pruebas	Cantidad de clases	Promedio
Baja	7	8,974358974
Media	2	2,564102564
Alta	0	0



Gráfica 10. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

Reutilización

Reutilización	Cantidad de clases	Promedio
Baja	0	0
Media	2	2,564102564
Alta	7	8,974358974



Gráfica 11. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del componente de autenticación del Sistema de Gestión Integral de Seguridad tiene una calidad buena teniendo en cuenta que el 100 % de las clases incluidas en estos subsistemas posee menos de 3 dependencias de otras clases. Además el 100% de las clases posee índices aceptables en cuanto a Acoplamiento (entiéndase calificaciones de bajo y medio para todas las clases). Así mismo los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización se comportan satisfactoriamente en un 100 % de las clases (entiéndase calificaciones de alta y media en el atributo de calidad y calificaciones de baja y media en los restantes).

3.2.1.3 Matriz de cubrimiento ó matriz de inferencia de indicadores de calidad

La matriz de cubrimiento o matriz inferencia de indicadores de calidad es una representación estructurada de los atributos de calidad y métricas utilizadas en los epígrafes anteriores para evaluar la calidad del diseño de los componentes que integran la solución propuesta. La misma permite conocer si el resultado obtenido de la relación atributo/métricas para cada componente es positivo o negativo. Llevando estos resultados a una escala numérica donde, si los resultados son positivos tendrá un valor de 1, si son negativos de 0 y si no existe relación alguna se tomará como nula (-). Una vez completado los datos de dicha relación se realiza un cálculo donde se promedia la sumatoria de los valores obtenidos de un atributo por cada métrica evaluada, y la división de dicha sumatoria por la cantidad de métricas evaluadas (solo se promedian las que arrojan un resultado, las nulas no). Este valor es el que va a tener el atributo dentro de una tabla que medirá si los atributos fueron buenos, regulares o malos. Se lograron los siguientes resultados:

Calificativo	Valor
Positivo	1
Negativo	0
Nulo	-1

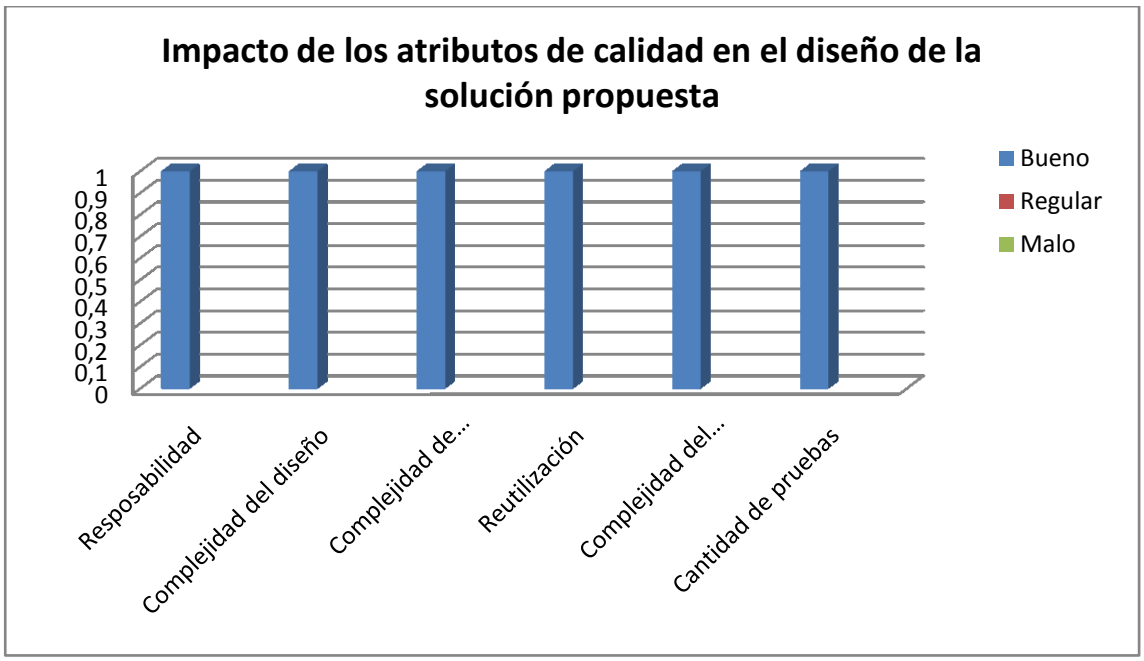
Atributos/Métricas	TOC	RC	Promedio
Responsabilidad	1	-1	1
Complejidad del diseño	1	-1	1
Complejidad de implementación	1	1	1
Reutilización	-1	1	1
Complejidad del mantenimiento	-1	1	1
Cantidad de pruebas	-1	1	1

Tabla 1. Resultados evaluados de la relación Atributos/Métricas por cada componente que integran la solución.

Calificativo	Valor
Malo	<= 0.4
Regular	>0.4 y <=0.7
Bueno	>0.7

Atributos/Métricas	Bueno	Regular	Malo
Responsabilidad	1		
Complejidad del diseño	1		
Complejidad de implementación	1		
Reutilización	1		
Complejidad del mantenimiento	1		
Cantidad de pruebas	1		

Tabla 2. Rango de valores para la evaluación técnica de los atributos de calidad evaluados por cada métrica.



Gráfica 12. Gráfica de los resultados obtenidos de los atributos de calidad evaluados en las métricas.

3.3 Conclusiones Parciales

En este capítulo se analizaron los casos de prueba y las métricas pertinentes que se le aplicaron al componente para verificar los niveles de calidad con que el mismo fue realizado. Los resultados obtenidos fueron satisfactorios luego de haber aplicado estos métodos.

Conclusiones

Al realizar un estudio del estado del arte de los componentes de autenticación de los sistemas de gestión de seguridad, tanto dentro como fuera del país, arrojó como resultado que en la actualidad la mayoría de estos sistemas están implementados rigiéndose por estándares internacionales.

Para darle solución a esta problemática se hizo un análisis, diseño e implementación del componente de autenticación de ACAXIA el cual gestiona la seguridad guiado por el estándar internacional SAML, resolviendo para su primera versión el escenario arquitectónico SSO.

La solución fue probada por el departamento de calidad del Centro de Informatización para la Gestión de Entidades (CEIGE) donde se verificó que se cumplieran todos los requisitos funcionales. Además se realizó una validación del diseño mediante la utilización de instrumentos de medición que se inspiraron en métricas para la calidad del diseño. Los resultados arrojados permitieron concluir que el diseño presenta valores positivos en indicadores de calidad tales como Reutilización, Facilidad de Mantenimiento, Complejidad del Diseño, Complejidad de Implementación, Acoplamiento, Cantidad de pruebas, entre otros. Esto apoya la afirmación de que el análisis, diseño e implementación del componente de autenticación del Sistema de Gestión Integral de Seguridad tiene una calidad excelente.

Recomendaciones

El autor que realiza el presente trabajo considera importante distinguir las siguientes recomendaciones:

- Que los sistemas web de gestión desarrollados en la UCI y en general en Cuba se suscriban a la nueva versión de ACAXIA, ya que esta brindará mayor seguridad y nuevas funcionalidades.
- Trabajar en nuevas versiones del sistema para fortalecer la gestión de la seguridad brindada por este.
- Que la nueva versión sea avalada por las instituciones pertinentes como la ONRI y por instituciones que evalúen la calidad en cuanto a seguridad respecta como Segurmática.

Anexos

Anexo 1

Total de clases	**9**
Promedio de procedimientos	**5**

	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.

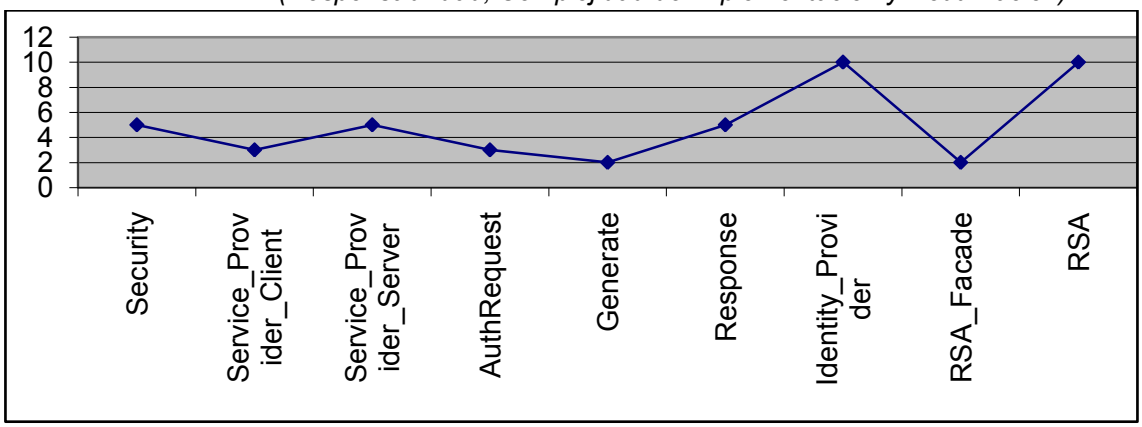
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.

Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Tabla 3. Rango de valores de para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados **No**	**Subsistema**	**Clase**	**Cantidad de Procedimientos**	**Responsabilidad**	**Complejidad**	**Reutilización**
1	Seguridad.	**Security.**	5	Baja	Baja	Alta
2	Proveedor de Servicio.	**Service_Provider_Client.**	3	Baja	Baja	Alta
3	Proveedor de Servicio.	**Service_Provider_Server.**	5	Baja	Baja	Alta
4	Proveedor de Servicio.	**AuthRequest.**	3	Baja	Baja	Alta
5	Proveedor de Servicio.	**Generate.**	2	Baja	Baja	Alta
6	Proveedor	**Response.**	5	Baja	Baja	Alta

	de Identidad.					
7	Proveedor de Identidad.	**Identity_Provider.**	10	Media	Media	Media
8	Proveedor de Identidad.	**RSA_Facade.**	2	Baja	Baja	Alta
9	Proveedor de Identidad.	**RSA.**	10	Media	Media	Media
10	Media	Media	Media			

Tabla 4. Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).



Gráfica 13. Gráfica de los resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

Anexo 2

Total de clases	**9**
Promedio de asociaciones de uso	**1,444444444**

	Categoría	**Criterio**
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

Complejidad de Mantenimiento.	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

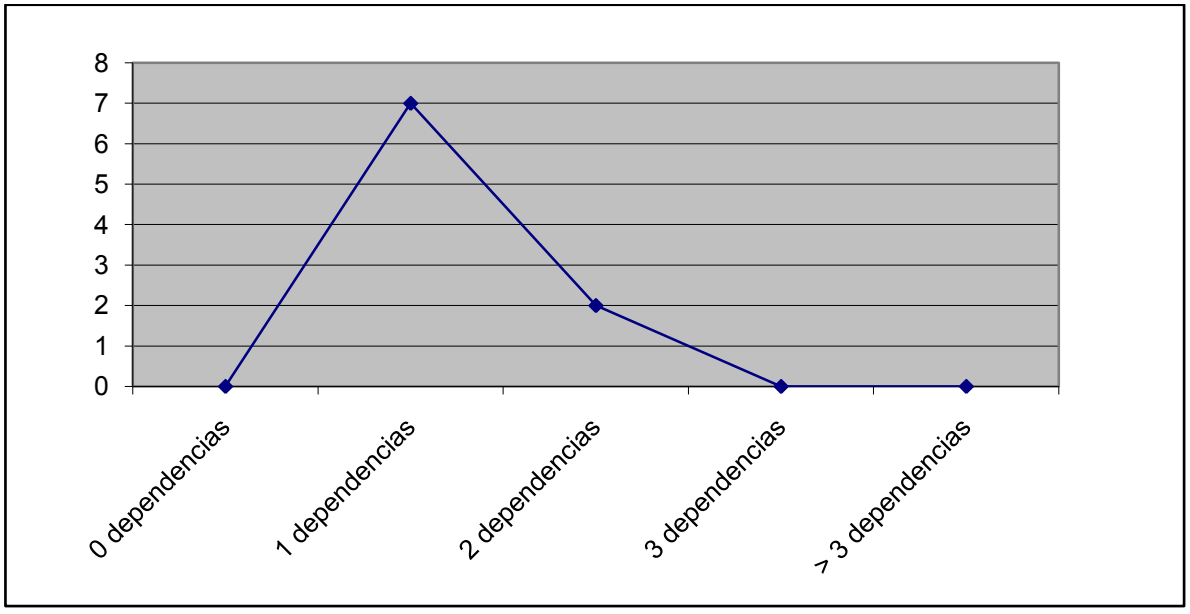
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.

Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Tabla 5. Rango de valores para la evaluación técnica de los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas) relacionados con la métrica RC.

No	Subsistema	Clase	Cant. de Relaciones de Uso	Acoplamiento	Complejidad Mantenimiento	Reutilización	Cant. P.
1	Seguridad.	**Security.**	1	Medio	Media	Media	Media
2	Proveedor de Servicio.	**Service_Provider_Client.**	1	Bajo	Baja	Alta	Baja
3	Proveedor de Servicio.	**Service_Provider_Server.**	2	Medio	Media	Media	Media
4	Proveedor de Servicio.	**AuthRequest.**	1	Bajo	Baja	Alta	Baja
5	Proveedor de Servicio.	**Generate.**	1	Bajo	Baja	Alta	Baja
6	Proveedor de Identidad.	**Response.**	1	Bajo	Baja	Alta	Baja
7	Proveedor de Identidad.	**Identity_Provider.**	2	Medio	Media	Media	Media
8	Proveedor de Identidad.	**RSA_Facade.**	1	Medio	Media	Media	Media
9	Proveedor de Identidad.	**RSA.**	1	Bajo	Baja	Alta	Baja

Tabla 6. Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas).



Gráfica 14. Gráfica de los resultados de la evaluación de la métrica RC agrupados por la tendencia de los valores.

Referencias bibliográficas

Grupo de interes especial en datos de comunicación. 2010. ACM SIGCOMM. [En línea] Enero de 2010. [Citado el: 20 de Abril de 2010.] http://www.sigcomm.org/standards/iso_stds/OSI_MODEL/.

Microsoft Corporation. 2009. Micrsoft.com. [En línea] Microsoft, Marzo de 2009. Microsoft. Microsoft.com. Microsoft Technet Seguridad. [Online] Microsoft Corporation,2009.http://www.microsoft.com/latam/technet/seguridad/articulos/bpsegcorp.mspx.

Microsoft Corporation. 2009. Micrsoft.com. [En línea] Microsoft, Noviembre de 2009. Microsoft. Microsoft.com. Microsoft Technet Seguridad. [Online] Microsoft Corporation,2009. (http://technet.microsoft.com/es-es/library/bb124149.aspx).

Letelier, Patricio. 2003. *Introducción al Proceso de Desarrollo de Software.* s.l. : DSIC, 2003.

Sánchez, Maria A. Mendoza. 2007. *Metodologías De Desarrollo De Software.* 2007.

Universidad politécnica de Valencia. 2003. *Proceso de desarrollo de software.* Valencia : s.n., 2003.

José H. Canós, Patricio Letelier y Mª Carmen Penadés. 2003. *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n., 2003.

Pérez, M. M. (2009). *Propuesta de modelo de desarrollo de software tecnológico del centro de soluciones de gestión.* Ciudad Habana: UCI.

world, G. d. (2010). Network World. Recuperado el Mayo de 2010, de http://www.networkworld.es/SAML-2.0-simplifica-la-federacion-de-identidades/seccion-/articulo-173366

Larman, Craig. *UML y Patrones, introducción al análisis orientado a objetos.* México DF : Prentice Hall Hispanoamericana, S.A, 1999. ISBN 970-1 7-0261-1.

Sommerville, Ian. *Ingeniería de Software.* Sexta edición. s.l. : Addison-Wesley, 2002. ISBN: 970-26-0206-8.

. **Lockhart, Thomas***. The PostgreSQL Development Team. s.l. : Thomas Lockhart, (1996).*

PostgreSQL Comunitario Mexico*¿Que es PostgreSql?*
Disponible en: http://www.postgresql.org.mx/?q=node/6. (2010)

Bibliografía

Equipo Arquitectura del ERP Cedrux. Especificación Técnica para el marco de la arquitectura. 2008.

Lazo Ochoa, Rene and Yzquierdo Herrera, Raykenler. El modelo de diseño del sistema HyperWeb.Módulos de Tratamiento Farmacológico y Configuración. Ciudad Habana: Universidad de las Ciencias Informáticas, 2007.

Pressman, R. Ingenería del software: Un enfoque práctico. McGraw Hill: s.n., 2000.

PHP, Comunidad. Comunidad_PHP. [Online] Universidad de las Ciencias Informáticas. http://php.uci.cu/.

UCI. Teleformación. [Online] Universidad de las Ciencias Informáticas. http://teleformacion.uci.cu.

UCI. Biblioteca. [Online] Universidad de las Ciencias Informáticas. http://biblioteca.uci.cu/.

Zend Framework Playground. Zend Framework Playground. [Online] WordPress, Septiembre 2008, 2008. [Cited: Abril 21, 2009.] http://spanish.zendfw.com/.

Microsoft Corporation. Microsoft TechNet Seguridad. [Online] 2009. http://www.microsoft.com/latam/technet/seguridad/articulos/bpsegcorp.mspx.

KumbiaPHP Framework. KumbiaPHP Framework. [Online] 2007. http://www.kumbiaphp.com.

Cake Development Corporation. Cake Development Corporation. CakePHP. [Online] Cake Development Corporation, 2007. http://www.cakedc.com/.

SAML V2.0 Contributors. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0.

[Online] http://docs.oasis-open.org/security/saml/v2.0/. March 2005.

SAML V2.0 Contributors. Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0. [Online] http://docs.oasis-open.org/security/saml/v2.0/. March 2005.

SAML V2.0 Contributors. Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0.

[Online] http://docs.oasis-open.org/security/saml/v2.0/. March 2005.

SAML V2.0 Contributors. Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0.

[Online] http://docs.oasis-open.org/security/saml/v2.0/. March 2005