Universidad de las Ciencias Informáticas

Facultad 3

“Solución de autorización basada en el estándar XACML para el sistema de seguridad ACAXIA”.

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

Autores: Racielis Martin Díaz.
José Ernesto Quiroga Arencibia.

Tutor: Dr. Oiner Gómez Baryolo.

La Habana, Junio de 2013
“Año 55 de la Revolución”

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al departamento de Tecnología del centro CEIGE de la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los _____ días del mes de_____ del año _____.

José Ernesto Quiroga Arencibia

Racielis Martin Díaz

Dr. Oiner Gómez Baryolo

Pdte. del tribunal

Datos de Contacto

- ✓ **Tutor: Dr. Oiner Gómez Baryolo**

Correo electrónico: oiner@uci.cu.

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

Síntesis de los autores:

- ✓ **Racielis Martin Díaz**

Estudiante de la Universidad de Ciencias Informáticas.

Correo electrónico: rmartind@estudiantes.uci.cu

- ✓ **José Ernesto Quiroga Arencibia**

Estudiante de la Universidad de Ciencias Informáticas.

Correo electrónico: jequiroga@estudiantes.uci.cu

De Racielis:

A mis padres Julio Cesar y Leimis, a mi hermanita Ileicar y en especial a mi abuelito Manolo.

De José Ernesto:

A mi familia, en especial a mi papá, mi hermana Jenni y a Gelcy.

De Racielis:

Ahora sí, llego el momento de agradecer a cada una de las personas que de una forma u otra han contribuido a mi formación profesional y personal. Aunque las palabras nunca alcanzan, cuando lo que hay que decir desborda el alma, realizaré mi mayor intento en comunicar mis agradecimientos a:
A mi abuelito Manolo por estar siempre a mi lado y estar pendiente de mí. A mi padre Julio C. por el sacrificio continúo para que no me falte nada, gracias por quererme y por inculcarme que con esfuerzo todo se puede. A mi hermanita Ileicar que la quiero muchísimo, gracias por hacerme reír y ser mi motivación. Todo lo que soy o espero ser se lo debo a mi madre. Gracias por tu cariño, bondad y apoyo incondicional brindado en todo momento. A mi abuelita Erena porque a pesar de la distancia siempre me ha ofrecido los mejores consejos del mundo. A mis tías y primos porque a pesar de que nos vemos poco los tengo siempre presente. A mis abuelitos postizos Mire e Israel y a mi mejor amiga Lisi por la confianza depositada en mí y por levantarme los ánimos en los momentos difíciles.
A mi novio José Ernesto, no creo que lo hubiera logrado sin ti, y a su familia que los quiero muchísimo. A mis tutores Oiner y Mile por guiarme en la realización de este trabajo. Al oponente y al resto del tribunal, por las recomendaciones realizadas para que el trabajo tuviera la calidad requerida. A mis compañeros de proyecto en especial a José, Katia, Hector, Janier y Luis Angel por ayudarme y por todos los buenos momentos de esta amistad. A mis compañeros de grupo y de residencia por estar siempre ahí para compartir buenos y malos momentos durante toda mi trayectoria en la universidad que han hecho inolvidable mi estancia en la misma.
Lo que ha hecho que este sueño se haga realidad es haberlo compartido con vosotros. Gracias…

De José Ernesto:

A mi papá que me lo ha dado todo: su ejemplo, cariño, ayuda en los momentos difíciles y por ser puntal en mi formación académica y social. A mis hermanas Jenni y Yudalkis, a mi mamá, a Gelcy a mis tíos Ramón, Ana y Ernesto que siempre me han ayudado y apoyado en todo.
A mi compañera de tesis, novia y amiga Racielis que me ha tenido que soportar y ayudar durante estos casi 5 años de carrera y siempre ha estado en los momentos que más la he necesitado dándome su apoyo y amor.
A mis compañeros de proyecto Jose, Janier, Katia y Luis Angel que me han ayudado muchísimo en el desarrollo de la tesis.
A mis compañeros de grupo, en especial para el Riki, Tallys, Maricet y Jose que siempre me han brindado su ayuda incondicional cuando la he necesitado.
A mis amigos Eddy y Karel que me han acompañado y ayudado durante muchos años y por los buenos momentos que hemos compartidos juntos.
A Mile y a mi tutor Oiner por su gran ayuda en la realización de esta tesis.

"Al mundo nuevo corresponde la Universidad nueva. A nuevas ciencias que todo lo invades, reforman y minan nuevas cátedras. Es criminal el divorcio entre la educación que se recibe en una época, y la época. Educar es depositar en cada hombre toda la obra humana que le ha antecedido: es hacer a cada hombre resumen del mundo viviente, hasta el día en que vive: es ponerlo a nivel de su tiempo, para que flote sobre él, preparar al hombre para la vida. En tiempos teológicos, universidad teológica. En tiempos científicos, universidad científica".

José Martí

Resumen

En la actualidad se han desarrollado varios sistemas de información que permiten informatizar los procesos, el control de los recursos y el análisis de la información. Las actividades económicas, sociales, políticas e incluso militares muestran una profunda dependencia hacia este tipo de tecnología. Es por ello que la seguridad de los sistemas de información se ha convertido en uno de los temas de investigación más explorados en los últimos años.

Uno de los principales sistemas de información existentes en Cuba es el Sistema Integral de Gestión Cedrux el cual se desarrolla en la Universidad de las Ciencias Informáticas. Cedrux cuenta con un sistema de gestión integral de seguridad denominado ACAXIA que tiene como objetivo garantizar la gestión centralizada de la seguridad en un entorno de varias aplicaciones. El presente trabajo está enmarcado en la creación de una solución para ACAXIA que permita gestionar la autorización basada en el estándar XACML lo que facilita la capacidad de integración con otros sistemas.

Para su desarrollo se realizó un estudio que incluye diversas soluciones de autorización basadas en XACML con el propósito de investigar el proceso de autorización. Además se analizaron las herramientas y tecnologías a utilizar y mediante pruebas funcionales o de caja negra evaluar la calidad del software, descubrir errores y medir el grado en que se cumple con los requerimientos definidos. La solución de autorización basada en XACML evidencia su contribución al fortalecimiento y complejidad funcional del control de acceso, específicamente en el proceso de autorización desplegado en ACAXIA.

Palabras Claves: autorización, control de acceso, entorno tecnológico, estándar, proceso, sistema de información, seguridad.

Índice

Índice de Ilustraciones

Introducción

La colosal revolución tecnológica contemporánea ha generado en gran medida el desarrollo de complejos Sistemas de Información (SI), los cuales procesan, almacenan y distribuyen datos en una entidad. Un sistema de gestión de la información esta dictaminado como un conjunto organizado de personas, procesos y recursos, incluyendo la información y sus tecnologías asociadas, que interactúan de forma dinámica, para satisfacer las necesidades informativas que posibilitan alcanzar los objetivos de una o varias organizaciones [1, 2].

Los SI permiten informatizar los procesos, el control de los recursos, el análisis de la información y se utiliza como soporte para la toma de decisiones en una entidad, dichas acciones son alcanzadas a través de las funcionalidades de captura, almacenamiento y procesamiento de la información. Estos sistemas deben ser sencillos para gestionar la información por el personal autorizado pero a su vez presentar complejos mecanismos de seguridad para evitar brechas en su sistema de seguridad.

En un SI que es utilizado por dos o más empresas o entidades públicas, donde el acceso a los datos y a las aplicaciones es compartido, es necesario establecer mecanismos de seguridad para detectar las vulnerabilidades y los riesgos de ataque a los que están expuestos. El auge de las redes de comunicación provoca en muchos casos que exista una incorrecta manipulación de los datos afectando la confidencialidad, integridad de la información y la disponibilidad del sistema, que tiene lugar de forma accidental o intencionada [3].

Estudios realizados demuestran que más del 70% de las organizaciones son afectadas por incidentes de seguridad, los cuales varían en su forma y tipo. Aunque en los dos últimos años las compañías han incrementado hasta en un 10% el presupuesto para sus sistemas informáticos, desafortunadamente se ha identificado que las pequeñas empresas han estimado un promedio de al menos un incidente de seguridad al mes, mientras que en las grandes empresas se valora que ocurra una vez a la semana. El costo promedio de un incidente de seguridad grave en el 2012 fue de £15,000 - £30,000 para las pequeñas empresas y de £110.000 - £250.000 para las grandes empresas como se refleja en la Figura 1[1][2].

1 Fuente: Information Security Breaches Survey 2010: Technical report, Chris Potter y Grant Waterfall 2012

Figura 1: Principales tipos de infracciones en el 2012

Las amenazas y vulnerabilidades identificadas en las organizaciones han propiciado nuevas perspectivas en los procesos de seguridad informática, las cuales han dado lugar al desarrollo de normas y estándares enfocados en la protección de la infraestructura computacional incluyendo la información contenida en esta. Las empresas necesitan demostrar que realizan una gestión competente y efectiva de la seguridad de los recursos y datos que gestionan, que identifican y detectan los riesgos a los que están sometidas y que adoptan medidas adecuadas y proporcionadas para evitar incidentes. Además necesitan contar con un conjunto estructurado, sistemático, coherente y completo de normas a seguir. Los estándares brindan soluciones para los procesos de control de acceso y la capacidad de integración con otros sistemas.

El proceso de control de acceso se refiere a la protección de los recursos del sistema contra el acceso no autorizado. En particular, se define un proceso por el cual el uso de los recursos se regulan de acuerdo a políticas de control de acceso y se permite exclusivamente al personal autorizado de acuerdo con la política establecida[4]. Este mecanismo surge a partir de los problemas de seguridad, privacidad o de penetración interna o externa de usuarios maliciosos detectados con la aparición de los sistemas compartidos[5] y está determinado por tres componentes fundamentales, la identificación y autenticación, la auditoria y la autorización, en este último se enmarcará esta investigación.

En el proceso de autorización se determinan cuáles son los derechos de acceso sobre los servicios y recursos del personal o sistema anteriormente autenticado. Un modo de gestionar los permisos de autorización sobre un recurso o servicio es mediante la utilización de Listas de Control de Acceso (ACLs), las cuales manejan registros de información para determinar quién puede hacer qué. Hoy en día las

entidades elaboran su propia infraestructura de ACLs provocando que no se despliegue una solución portable y unificada, dificultando así su mantenimiento, control e integración con otros sistemas.

El Lenguaje de Marcaje de Control de Acceso Extensible (XACML) se define para expresar políticas de seguridad basadas en un Lenguaje de Marcaje Extensible (XML). Propone una solución que facilita el intercambio de mensajes con otros sistemas y establece un lenguaje unificado y portable para expresar sus políticas, definiéndolas en un modelo flexible y adaptado al nivel de detalle que sea necesario[6].

En Cuba la complejidad de los procesos de dirección, el desarrollo del capital humano y el impacto cada vez mayor de las Tecnologías de la Información y las Comunicaciones (TIC), aconsejan revisar las concepciones en torno a la gestión de la información, e integrar los SI del país, que interactúan para satisfacer las necesidades informativas relacionadas con los objetivos y planes del gobierno a todos los niveles, en los ámbitos económico, social, demográfico, geográfico, medioambiental, de funcionamiento de sus órganos y en otros que se decidan[1].

Uno de los principales SI existentes en Cuba es el Sistema Integral de Gestión de Información Cedrux el cual se desarrolla en la Universidad de las Ciencias Informáticas (UCI). Cedrux cuenta con un sistema de gestión integral de seguridad denominado ACAXIA, este tiene como objetivo general garantizar la gestión centralizada de la seguridad en un entorno de varias aplicaciones.

ACAXIA no implementa actualmente una estructura estándar para el intercambio de mensajes de autorización basado en XACML, lo que limita su capacidad de integración con otros sistemas y la posibilidad de adaptarse a entornos de despliegue heterogéneos donde existan sistemas legados que implementen el proceso de autorización basado en este estándar. Por lo anteriormente expuesto se propone modificar el proceso de autorización de este software para facilitar su integración con otros sistemas planteándose el siguiente **problema a resolver**:

¿Cómo estructurar los mensajes de autorización en el sistema ACAXIA para aumentar su capacidad de integración con otros sistemas?

Para resolver esta problemática se define como **objeto de estudio**, el proceso de autorización en Sistemas de Información, enmarcando el **campo de acción** en los estándares para el intercambio de mensajes de autorización entre Sistemas de Información.

Objetivo general: Implementar el estándar XACML en el sistema ACAXIA para aumentar su capacidad de integración con otros sistemas.

Objetivos específicos:

- ✓ Construir el marco teórico conceptual de la investigación, relacionado con los estándares más utilizados en el desarrollo de sistemas de autorización.
- ✓ Analizar y diseñar la solución propuesta.
- ✓ Implementar la solución propuesta.
- ✓ Validar la solución propuesta aplicando pruebas de caja negra y pruebas de integración con otros software.

Para desarrollar este trabajo se ha planteado la siguiente **idea a defender**: Si se desarrolla un componente de Autorización basado en el estándar XACML se aumentará la capacidad de integración de ACAXIA con otros sistemas.

El trabajo de investigación se sustenta mediante los siguientes **métodos teóricos:**

Analítico – Sintético: Permitió analizar los conocimientos generales sobre los sistemas de información estableciendo la extracción de los elementos más significativos, arrojando ideas y conclusiones mucho más prácticas y concretas.

Inductivo – Deductivo: Posibilitó una forma de razonamiento de las particularidades de la autorización dentro de los procesos de control de acceso en los SI, reflejando sus puntos comunes.

Histórico – Lógico: Posibilitó analizar el comportamiento y evolución del proceso de autorización dentro del mecanismo de control de acceso y su tendencia hacia el desarrollo de estándares que garanticen este proceso de forma simple y unificada.

El documento cuenta con la siguiente estructura:

Capítulo 1: Fundamentación teórica. Se realiza un estudio del estado del arte de los estándares para el intercambio de mensajes de autorización entre SI, así como de las tecnologías más utilizadas en este tema. Además se analizan y adoptan los principales conceptos que facilitan el estudio y compresión de la temática.

Capítulo 2: Propuesta de solución. Abarca los requisitos con los que debe cumplir la solución propuesta. También se generan los artefactos que permiten describir y entender el proceso a informatizar.

Capítulo 3: Se valida la solución aplicando pruebas de integración con otros software en un entorno real. Además se definen pruebas de caja negra utilizando la técnica de particiones equivalentes.

Se analizan los resultados obtenidos y se concluye con la valoración del aporte e impacto de estos resultados.

Finalmente se presentan las conclusiones y recomendaciones de la investigación.

Capítulo 1: Fundamentación Teórica

El proceso de autorización como mecanismo básico de seguridad puede variar dependiendo de lo que se está protegiendo. No toda la información de la organización es igual de crítica. Esta característica proporciona que en la estructura organizativa de cada entidad se establezcan sus propios mecanismos de seguridad; diversificándose así la gestión del proceso de autorización provocando que en la actualidad la capacidad de integración entre diversos SI sea limitada y compleja.

En el presente capítulo se describen los diferentes aspectos teóricos que forman la base conceptual para dar solución al problema de esta investigación. Se caracterizan los modelos de autorización, las metodologías, tecnologías, estilos arquitectónicos, herramientas de software sobre las cuales se desarrollará el sistema y los estándares existentes para llevar a cabo la solución.

1.1. Conceptos asociados al dominio del problema

1.1.1. Control de acceso en Sistemas de Información

El control de acceso es indispensable en los SI empresariales cuyo funcionamiento requiere del intercambio de recursos digitales con diferentes niveles de sensibilidad[7]. El control de acceso es el proceso de conceder permisos a usuarios o grupos para acceder a los recursos de un SI. Los sistemas de control de acceso tienen la responsabilidad de establecer las políticas de seguridad que se deben cumplir para preservar la confidencialidad, integridad, disponibilidad y trazabilidad de los recursos gestionados por los SI[8]. Según Suhendra, *una infraestructura de control de acceso involucra tres procesos fundamentales, la identificación y autenticación, la autorización y la auditoría*[9].

El concepto de control de acceso que se asume en el marco de esta investigación, es el que se define en la Resolución No. 127 /2007 del MIC de la república de Cuba. En él se expresa que “el control de acceso es un método que garantiza que solo tengan acceso a un sistema o a la información que éste contiene, aquellos debidamente autorizados para ello. Los mecanismos de control de acceso se implementan utilizando técnicas de software y de hardware y por lo general incluyen: identificación y autenticación de usuarios; limitación de acceso a ficheros, monitorización de las acciones de los usuarios y un sistema de auditoría”[10]. Se acepta este concepto por abarcar de forma específica los procesos de control de acceso.

1.1.2. Autorización

La autorización es el proceso de determinar si a un usuario ya identificado y autenticado se le permite acceder a los recursos de una manera específica. La autorización es la responsabilidad del servicio de acceso a un recurso[11].

Un usuario puede ser autenticado con cierta identidad pero puede solicitar para ser autorizado a acceder a un recurso con una identidad diferente. Cuando el usuario pide explícitamente el acceso a una aplicación o a un recurso, esto se denomina identidad de autorización. Cuando esto se lleva a cabo por una aplicación o servicio en nombre del usuario, esto se denomina suplantación. En el caso de suplantación, un usuario puede poseer una identidad de autenticación que ha sido comprobada por el proceso de autenticación[11].

La política de autorización debe ser gestionada por el administrador o agente de seguridad responsable de apoyar y llevar a cabo la política de seguridad en la organización. Una vez que una persona (o sistema) ha sido autentificado, es necesario determinar si puede ejecutar la transacción deseada. En algunos sistemas la función de autorización se limita a autorizar a realizar cualquier acción a quienes han sido correctamente autentificados. Sin embargo, cuando existe diversidad de acciones y cada una con distinto impacto, surge la necesidad de controlar más rigurosamente algunos grupos de transacciones. En general, esta necesidad se cubre con el siguiente enfoque[12]:

- ✓ Se agrupan las transacciones según el nivel de protección requerido para cada una.
- ✓ Se define para cada identidad a qué grupos puede acceder.

1.2. Modelos de control de acceso

A continuación se analizan los principales modelos que contribuyen al fortalecimiento del proceso de autorización.

1.2.1. Modelo de Control de Acceso Discrecional (DAC)

El modelo de control de acceso DAC establece que el autor de un objeto es quien decide cómo protegerlo y está autorizado a otorgar permisos sobre el mismo a otros usuarios. DAC es una forma de autorización basada en los sujetos y grupos a los que pertenece un objeto. Se dice que es discrecional en el sentido de que un sujeto puede transmitir sus permisos a otro sujeto sin la aprobación de un administrador general[13]. En la Figura 1.1[2] se ilustran las relaciones existentes en este modelo.

2Fuente: Modelo de control de acceso para sistemas de información en entornos multidominios, MSc. Oiner Gómez Baryolo 2012.

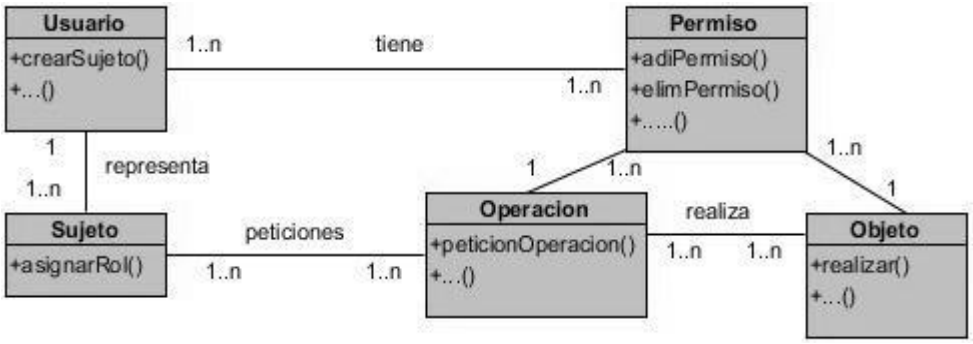


Figura 1.1: Modelo de Control de acceso Discrecional (DAC)

El carácter discrecional de DAC provoca que un sujeto acceda a objetos no autorizados para él[14].

A través de este modelo es difícil garantizar las reglas de integridad como "mínimo privilegio" o separación de tareas que son necesarias en los ambientes con procesos colaborativos. DAC es apropiado para escenarios donde existan pocos recursos y el intercambio de información sea más importante que su protección[15].

1.2.2. Modelo de Control de Acceso Obligatorio (MAC)

El MAC establece una política de seguridad multinivel, en la que los sujetos y objetos pertenecen a niveles de seguridad predefinidos por un administrador o el mismo sistema y es a éstos sistemas a los que se les conceden los permisos. En este modelo todos los sujetos y objetos son clasificados basándose en niveles predefinidos de seguridad. Para establecer las políticas por niveles, los sujetos y objetos son marcados con etiquetas de seguridad, que siguen el modelo de clasificación de la información militar (desde desclasificado hasta alto secreto)[16]. En la Figura 1.2[3] se ilustran las relaciones existentes en este modelo.

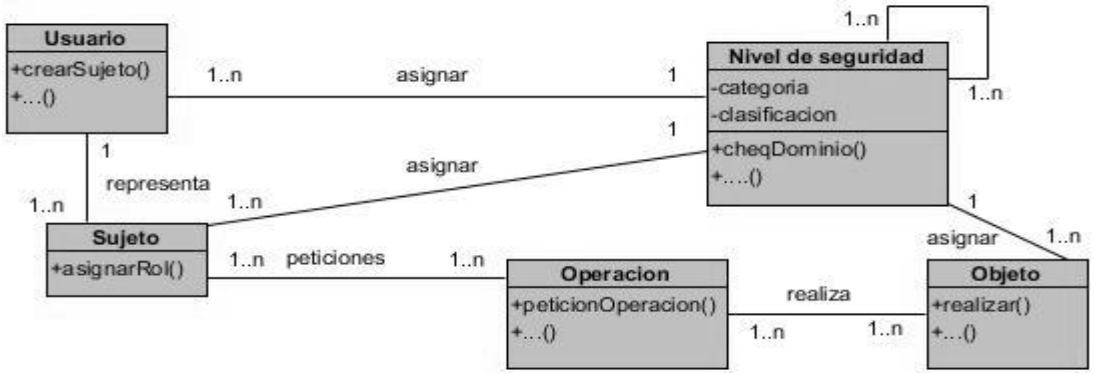


Figura 1.2: Modelo de Control de acceso Obligatorio (MAC)

3Fuente: Modelo de control de acceso para sistemas de información en entornos multidominios, MSc. Oiner Gómez Baryolo 2012.

La concepción de MAC estuvo marcada fundamentalmente por la necesidad de un modelo de control de acceso para escenarios militares. Asegurar las políticas MAC es a menudo una tarea difícil, particularmente en procesos colaborativos, ya que no proporcionan soluciones factibles producto de su falta de flexibilidad. Esta situación provoca que MAC sea vulnerable ante los ataques con troyanos, con el objetivo de lograr que un usuario pueda leer información del nivel superior y escribir en el nivel inferior. Este modelo se concentra en lograr la confidencialidad de la información por lo que deja expuesta su integridad[17].

1.2.3. Listas de Control de Acceso (ACL)

Las ACL son consideradas una implementación de DAC (válidas también para el control de acceso obligatorio), ya que en ellas se determina, mediante reglas, cuáles usuarios tienen acceso a cierta parte de la información[18].

Una ACL puede ser considerada como una estructura de datos jerárquica que puede contener múltiples ACL, cada una de ellas define un conjunto de permisos asociados a un determinado recurso. Los sujetos, objetos y permisos son los conceptos fundamentales que se utilizan para definir las reglas de acceso aplicadas directamente a los usuarios o a los grupos donde pertenecen[19]. En la Figura 1.3[4] se ilustran un ejemplo de la utilización de este modelo.

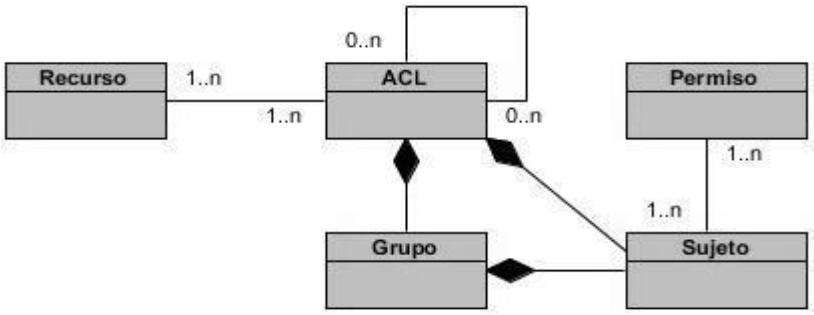


Figura 1.3: Listas de Control de Acceso (ACL)

Las ACL solo representan una estructura de almacenamiento de los privilegios autorizados de los usuarios o grupos sobre los recursos. Por esta razón no se recomienda su aplicación en SI de gran envergadura, integrados por un gran número de recursos y conceptos dinámicos que condicionan las políticas de autorización[20].

1.2.4. Modelo de Control de Acceso Basado en Roles (RBAC)

4Fuente: Modelo de control de acceso para sistemas de información en entornos multidominios, MSc. Oiner Gómez Baryolo 2012.

La definición básica de Control de Acceso Basado en Roles establece que los usuarios son asignados a roles, los permisos son asociados a roles y los usuarios adquieren permisos siendo miembros de roles[21]. RBAC es una tecnología de forma jerárquica que describe un grupo de usuarios que pueden desempeñar un conjunto de roles y realizar operaciones en las que utilizan un conjunto de objetos como recurso. Este modelo también incluye el concepto de sesión, que permite la activación y desactivación selectiva de roles, posibilitando que un usuario pueda ejercer los permisos de varios roles simultáneamente[22, 23].

RBAC brinda la posibilidad de especificar restricciones sobre la relación usuario/rol y sobre la activación de un conjunto de roles de usuario. Las restricciones pueden ser de dos tipos: estáticas o dinámicas. La Separación Dinámica de Deberes (en inglés, Dynamic Separation of Duty, DSD) al igual que la Separación Estática de Deberes (en inglés, Static Separation of Duty, SSD), limitan los permisos que son disponibles para un usuario. Sin embargo DSD difieren de las SSD por el contexto en el cual estas limitaciones son impuestas. Las DSD limitan la disponibilidad de los permisos aplicando las restricciones sobre los roles que pueden ser activados durante una sesión de usuario[24, 25].

A continuación se analizan detalladamente los componentes del modelo RBAC como solución desarrollada para fortalecer el proceso de autorización.

1.2.4.1. RBAC Centralizado

El RBAC Centralizado funciona como la relación “muchos a muchos” ya que muchos usuarios pueden ser asignados a uno o muchos roles y muchos roles pueden involucrar a más de un usuario. Éstos usuarios a su vez pueden tener muchos permisos y un permiso puede involucrar a muchos roles.

1.2.4.2. RBAC Jerárquico

RBAC Jerárquico es el modelo en el que el rol de mayor rango es el que tiene más privilegios sobre los recursos y los roles subordinados a él, adquieren algunos de los privilegios con los que cuenta el de mayor rango.

1.2.4.3. Relaciones estáticas de separación de cargas

La separación estática de cargas previene los conflictos de intereses cuando un usuario adquiere permisos y estos permisos suelen tener conflictos en los diferentes roles asignados al usuario. Para evitar que esto suceda, se le elimina al usuario algún rol, para evitar asuntos fraudulentos.

1.2.4.4. Relaciones dinámicas de separación de cargas

El objetivo principal de la separación dinámica de cargas es permitir más flexibilidad en cuanto a operaciones se refiere. Define que un usuario puede ser miembro de varios roles, siempre y cuando no se cree un conflicto de intereses cuando estos roles se activen independientemente.

La Figura 1.4[5] muestra los conceptos fundamentales que conforman este modelo RBAC.

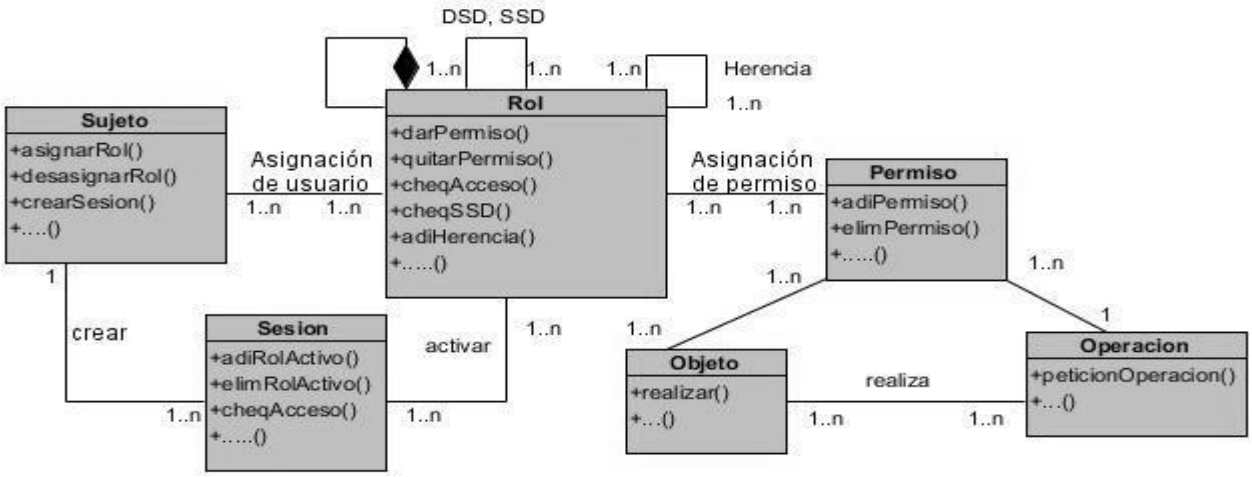


Figura 1.4: Control de Acceso Basado en Roles (RBAC)

RBAC presenta algunas deficiencias en su entorno de despliegue, a continuación se enumeran las debilidades que obtienen mayor importancia[26-29].

1. La herencia de roles genera una serie de conflictos que propician las violaciones de las restricciones establecidas.
2. Para la asignación de roles no se tienen en cuenta las características del entorno organizacional donde se despliega el sistema.
3. La gestión de privilegios tiene que ser centralizada para que los administradores mantengan la fortaleza del control de acceso.
4. Las restricciones solo se establecen a nivel de roles, sin tener en cuenta que para una determinada operación pueden existir restricciones en función de las características o atributos del recurso.

El RBAC fundamental requiere que el sistema soporte múltiples usuarios por cometido[6], múltiples cometidos por usuario, múltiples permisos por cometido y múltiples cometidos por permiso. Esto es

5Fuente: Modelo de control de acceso para sistemas de información en entornos multidominios, MSc. Oiner Gómez Baryolo 2012.

6 Función laboral en el contexto de una organización, con una semántica asociada que describe la autoridad y responsabilidad conferidas al usuario asignado a ese cometido.

importante a efectos de escalabilidad y gestión de sistemas de control de acceso. Cada uno de estos requisitos se puede satisfacer con las políticas[7] XACML.

1.3. Estándares de gestión de control de acceso

1.3.1. Lenguaje de Enmarcado de Control de Acceso Extensible (XACML)

La organización OASIS (Organization for the Advancement of Structured Information Standards) es una organización global sin fines de lucro interesada en el desarrollo, convergencia y adopción de estándares relacionados con e-business e intercambio electrónico de datos. Esta entidad ha aprobado la especificación XACML (Extensible Access Control Markup Language). [30]

XACML posibilita expresar políticas de control de acceso en forma simple y flexible. Esta especificación define un lenguaje basado en XML. Dicha especificación también define como objetivo fundamental la estructura de intercambio de mensajes de autorización y un modelo para organizar y almacenar la información de autorización. Este modelo impone una estructura base pero con flexibilidad suficiente para que cada sistema exprese las políticas de autorización de la forma más conveniente a su dominio de discurso.[6]

Los sistemas que implementen sus políticas de autorización basándose en XACML tendrán las siguientes ventajas:

- ✓ Utilizar un lenguaje unificado y portable para expresar sus políticas.
- ✓ Facilitar el intercambio de políticas con otros sistemas.
- ✓ Poder definir sus políticas con un modelo flexible y adaptado al nivel de detalle que sea necesario.

El estándar XACML propone un sistema de autorización con cuatro subsistemas, cada uno con una función bien delimitada descrita a continuación[6, 31, 32]:

PAP (Policy Administration Point) / (Punto de Administración de Política)

Es el punto en el cual se crean y administran las políticas de control. Puede ser desde un editor XML de archivos de texto hasta un sistema encargado de encapsular un lenguaje de políticas propietario en la forma de un lenguaje XACML.

PDP (Policy Decision Point) / (Punto de Decisión de Política)

7 Las políticas son la unidad básica que expresa quién puede hacer qué y bajo qué circunstancias o contexto

Es el punto responsable de evaluar un pedido de autorización a partir de las políticas definidas. Según la información que el pedido contenga y el examen de las políticas de acceso existentes, determinará si el pedido debe ser rechazado o no.

PEP (Policy Enforcement Point) / (Punto de Control de Política)

Es el punto que intercepta el pedido de autorización y lo deriva al PDP. Luego de obtener la respuesta del PDP elabora una repuesta para el sistema que hizo el pedido de autorización.

PIP (Policy Information Point) / (Punto de Información de Política)

En algunos casos la evaluación de un pedido de información puede requerir la búsqueda de información de otras fuentes. Esta información se refiere concretamente a valores de determinados atributos. En estos casos, el pedido contiene información sobre el recurso que contiene al valor del atributo y el PIP es responsable de interpretar estos datos y obtener el valor.

Para facilitar la comprensión de XACML, la Figura 1.5[8] muestra la estructura y el flujo de mensajes de autorización entre cada uno de los conceptos descritos anteriormente.

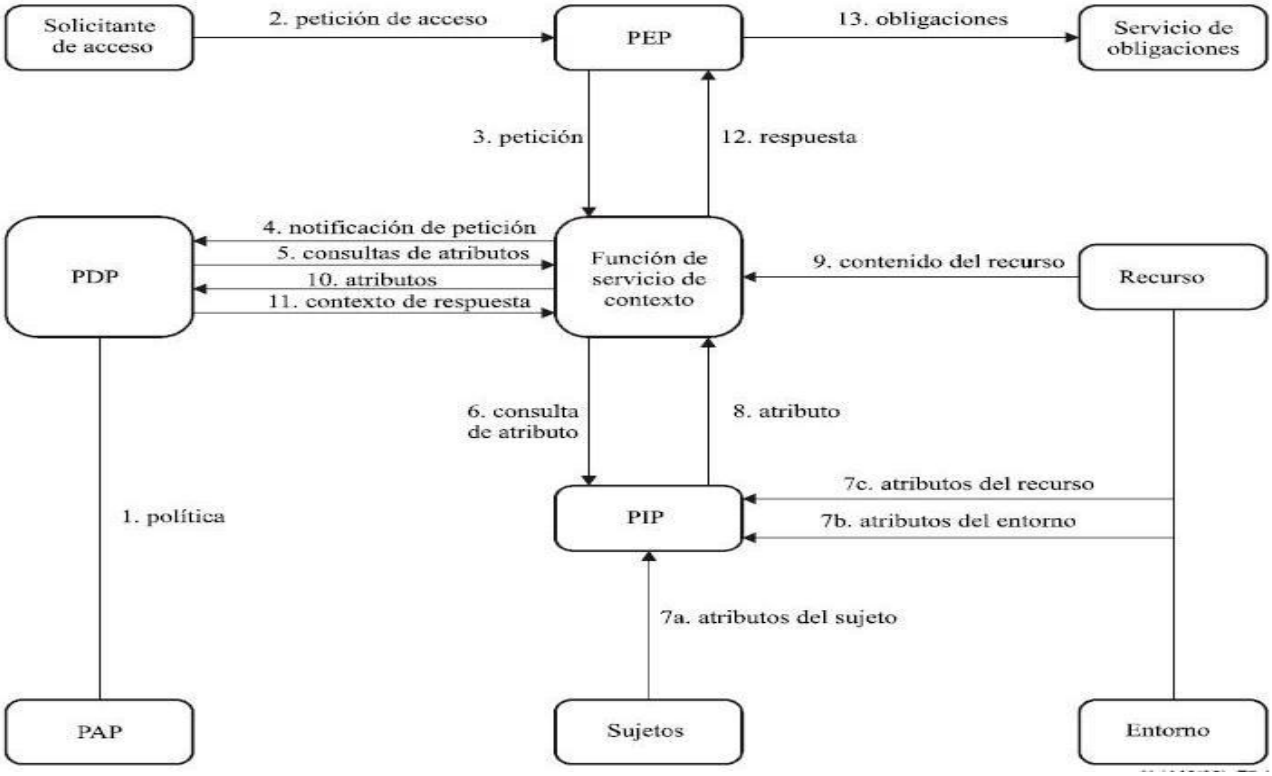


Figura 1.5: Flujo de intercambio de mensajes de autorización de XACML

8 Fuente: EXtensible Access Control Markup Language (XACML) Version 2.0, Moses, T., O.A.C. TC, Editor. 2005, OASIS: USA.

El área de impacto de XACML se centra en la estandarización de los mensajes de autorización. Por esta razón existen especificaciones donde se integra XACML con el estándar Lenguaje de Enmarcado de Aserciones de Seguridad (SAML) para incorporar la estandarización de los mensajes de autenticación y con RBAC para obtener las políticas de autorización.

1.3.2. Lenguaje de Enmarcado de Aserciones de Seguridad (SAML)

SAML, desarrollado por el Servicio de Seguridad del Comité Técnico de OASIS, es un marco de seguridad basado en XML para la comunicación de la autenticación de usuarios, el derecho, y la información de atributos. Como su nombre lo indica, SAML permite a las entidades empresariales hacer afirmaciones sobre la identidad, atributos y derechos de un sujeto (una entidad que es a menudo un usuario humano) a otras entidades, tales como una compañía asociada u otra aplicación de empresa[33].

En sus especificaciones describe la forma de intercambiar información de identificación, autenticación y autorización entre dominios aunque centra su fortaleza en la identificación y autenticación[34]. Usando sintaxis de XML, SAML establece el protocolo petición-respuesta mediante el cual los sistemas aceptan o rechazan sujetos basados en aserciones[35, 36]. Define además dos componentes fundamentales, el Proveedor de Servicios (PS) y el Proveedor de Identidades (PI) que van a ser las partes que intervienen en la comunicación SAML, para permitir que las aplicaciones puedan intercambiar información de identificación y autenticación abstrayéndose del cómo.

A pesar de las diferentes contribuciones que propone SAML al desarrollo sustentable de cada uno de los procesos de los sistemas de control de acceso, el examen crítico de este análisis estará enmarcado al estudio del intercambio de mensajes especificado por SAML con el objetivo de esclarecer y engrosar las ideas teóricas del campo de acción correspondiente a esta investigación.

Para una mejor compresión del tema se muestra en la siguiente Figura 1.6[9] el flujo de intercambio de mensajes que se produce entre el PS y el PI en el proceso de identificación y autenticación.

9 Fuente: Modelo de control de acceso para sistemas de información en entornos multidominios, MSc. Oiner Gómez Baryolo 2012

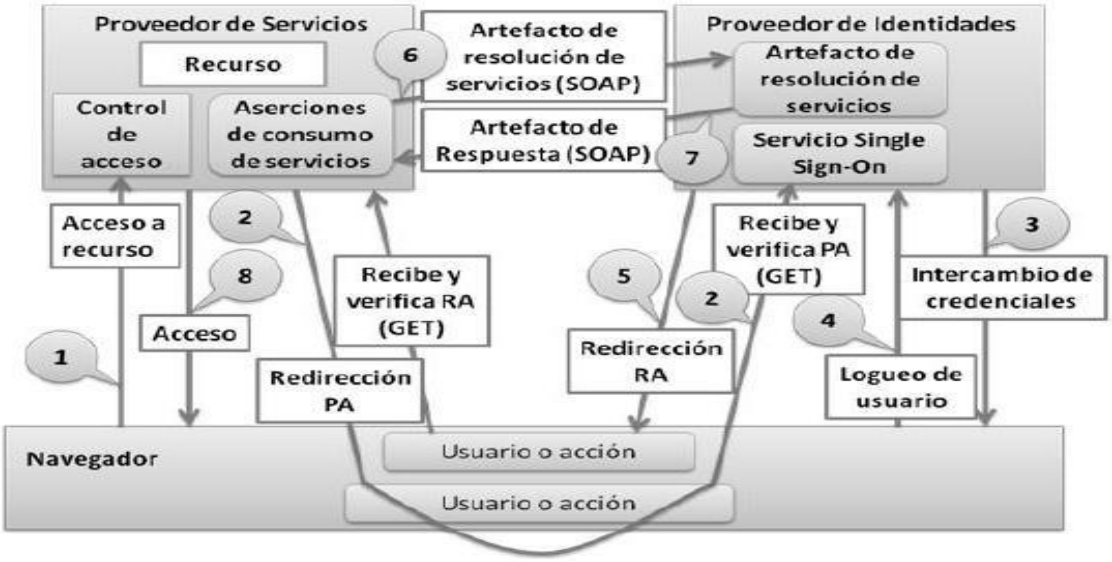


Figura 1.6: Flujo de intercambio de mensajes en la autenticación con SAML

El estándar SAML proporciona interfaces que permiten a los interesados presentar sus solicitudes de autenticación y autorización. Las solicitudes de autorización se procesan internamente y están dirigidas por el estándar XACML. XACML no solo procesa las solicitudes de autorización, también define el mecanismo para la creación de la infraestructura completa de las normas, políticas y juegos de políticas para llegar a las decisiones de autorización. SAML aborda autenticación y proporciona un mecanismo para la transferencia de autenticación y las decisiones de autorización entre las entidades cooperantes, XACML se centra en el mecanismo para llegar a las decisiones de autorización.[37]

1.4. Principales soluciones implementadas del estándar XACML

1.4.1. HERAS-AF XACML Core

HERAS-AF fue fundada en 2005 y es un proyecto oficial de código abierto desde 2006. Está muy bien establecido desde 2008. Presta asistencia al proceso de autorización entero, eso significa que todos los recursos a proteger son interceptados por un PEP y enviado a un PDP, que evalúa la solicitud, en caso de una respuesta positiva el PEP permite el acceso al recurso. Además, el mantenimiento es garantizado a través de un PAP como se muestra en la Figura 1.710.[38]

10Fuente: HERAS-AF: XACML 2.0 Implementation Developer's Guide, Department of Computer Sciences University of Applied Sciences Rapperswil 2017.

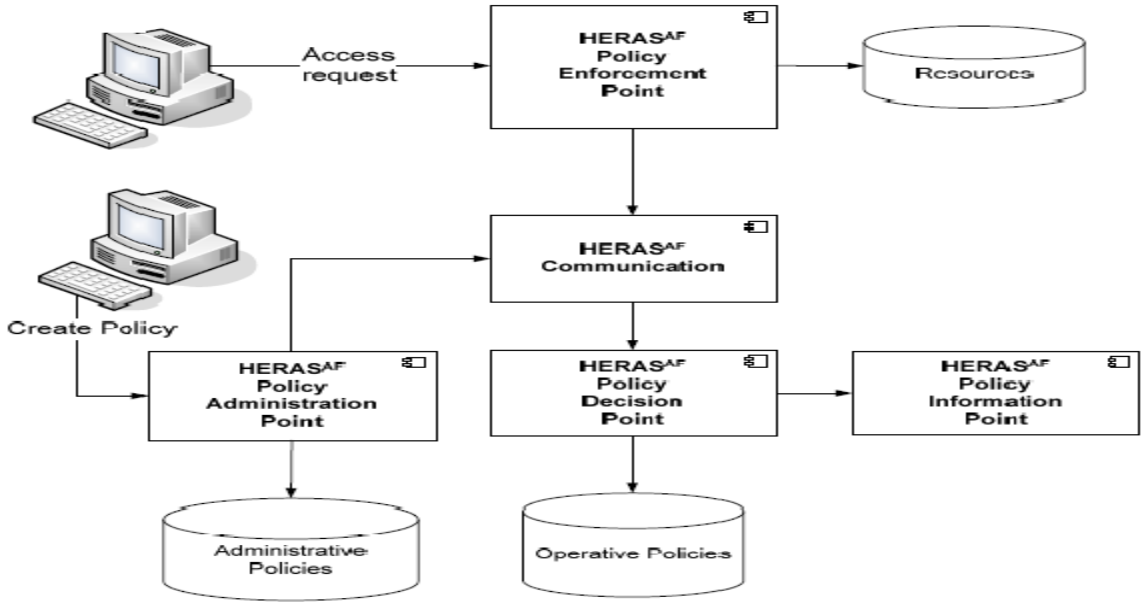


Figura 1.7: Arquitectura de HERAS-AF

El objetivo principal de HERAS-AF se basa en la interoperabilidad, capacidad de ampliación y la intercambiabilidad de artefactos de alta calidad de software. HERAS-AF XACML Core se ha convertido en un motor completo de XACML que representa la implementación de referencia de XACML 2.0. HERAS-AF recomienda a cada arquitecto o desarrollador de software utilice el HERAS-AF XACML Core en lugar de la aplicación Sun's XACML. Los componentes de software de HERAS-AF serán adoptados como un marco no intrusivo, lo que significa que solo pequeños cambios deben llevarse a cabo para integrar el software en un entorno existente.[38]

Aunque este software implementa en gran medida los componentes especificados en el estándar XACML su actual implementación se encuentra en código Java, además su solución no es adaptable al estándar de autenticación SAML y al modelo RBAC, esto imposibilita su adaptación o utilización para la solución de autorización que se desea desarrollar en ACAXIA.

1.4.2. OpenAz

Proyecto para facilitar el desarrollo de una norma basada en XACML PDP, PEP Az (autorización) y además presentar un marco de interfaz compuesto por dos elementos Interfaces XACML Az y AzService. La motivación del proyecto OpenAz es facilitar el desarrollo de una estructura de interfaz estándar, la cual servirá de base común para:[39]

- ✓ Externalizar la autorización de aplicaciones.

- ✓ Apoyar el uso de motores de políticas de autorización.
- ✓ PEP de construcción, basado en fundamentos XACML, que puede ser creado con una variedad de contextos de idiomas y tecnologías.
- ✓ Integrar la infraestructura existente mediante la abstracción de autorización dentro de un contexto XACML con el fin de desarrollar un marco uniforme de autorización.

La Figura 1.8[11] ilustra la visión de OpenAz. El AzAPI proporciona acceso de puntos de cumplimiento de políticas implementados utilizando diversas tecnologías para puntos de decisión remotas de políticas. Esta interfaz simplifica la implementación y hace que sea fácil de reemplazar un componente con otro.[39]

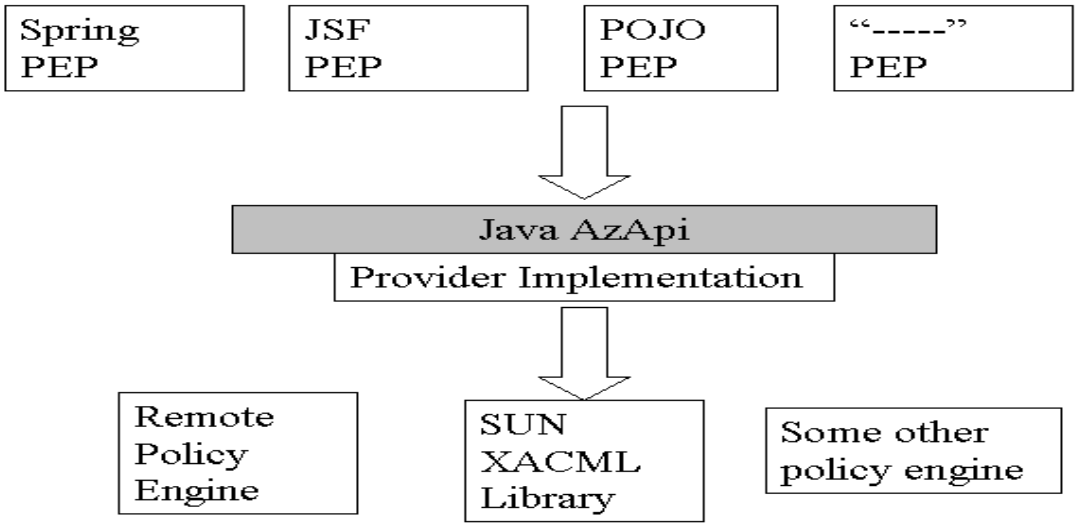


Figura 1.8: AZApi-relación con el PEP y los motores de la política

1.4.2.1. Interfaces XACML Az

Estas interfaces proporcionan un enlace de Java para la solicitud de autorización XACML y el protocolo de respuesta. A nivel conceptual, permiten a un desarrollador recoger la información requerida por el motor de autorización (AzRequestContext), y lo presentará a través de AzService y finalmente procesar la respuesta devuelta por el motor (AzResponseContext). Además de la decisión (AZ_PERMIT, AZ_DENY), la respuesta puede incluir información sobre otros atributos necesarios y obligaciones.[39]

1.4.2.2. AzService

AzService proporciona una interfaz de fábrica para que las instancias de un objeto AzRequestContext puedan ser creadas. AzRequestContext es el objeto que AzAPI toma como referencia para construir ya

11 Fuente: http://www.openliberty.org/wiki/index.php/OpenAz_Main_Page.

sea una única solicitud de autorización o una colección que puede incluir varias solicitudes de autorización. AzService implementa el método ***decidir()*** que procesa un objeto AzRequestContext, y también devuelve los resultados en un objeto AzResponseContext[39]

La solución implementada en el proyecto OpenAz propone una interfaz desarrollada en código Java que proporciona un enlace de Java para la solicitud de autorización XACML pero no garantiza la estandarización de mensajes de autorización. Además esta solución no incorpora módulos que faciliten su integración con RBAC y estándares de autenticación como SAML lo cual imposibilita su integración con el sistema ACAXIA para la solución de autorización que se pretende desarrollar.

1.4.3. Sun's XACML

Esta es una implementación de código abierto del estándar OASIS XACML, escrito en el lenguaje de programación Java. Sun XACML requiere de la plataforma Java 2 y de Standard Edition versión 1.4.0 o posterior.[40]

Este proyecto proporciona soporte completo para todas las funciones obligatorias de XACML, así como una serie de características opcionales. Existe un apoyo pleno para analizar tanto la política como los documentos de solicitud / respuesta, la determinación de la aplicabilidad de las políticas y la evaluación de las solicitudes en contra de ellas. Todos los tipos de atributos estándares, funciones, y algoritmos de combinación son compatibles.[40]

Este proyecto fue desarrollado en Sun Microsystems Laboratories y es parte de un proyecto en curso sobre Autorización de Internet en el Internet Research Group de Seguridad.[40] Incluye la estructura de mensajes de autorización declarada en el estándar XACML pero su implementación en código Java y su incompatibilidad con RBAC y SAML son inconvenientes para su utilización en la implementación de la solución de autorización del sistema ACAXIA.

1.5. Tecnologías y herramientas para el desarrollo

En este punto se tratan los conceptos fundamentales relacionados con la metodología, tecnologías, y herramientas que se proponen para el desarrollo de la solución de autorización basada en el estándar XACML para el sistema de seguridad.

1.5.1. Metodología de desarrollo

Las metodologías de desarrollo de software son un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. La finalidad de estas es garantizar la eficiencia en la generación de software.

El proceso de desarrollo de esta solución está guiado por el modelo de desarrollo de software dictaminado por el Centro de Informatización de la Gestión de Entidades (CEIGE). Este modelo define la especificación de las actividades de cada una de las fases del ciclo de vida de los proyectos del centro, además toma en cuenta los procesos de CMMI (Capability Maturity Model Integration) del nivel 2 para la Universidad de las Ciencias Informáticas (UCI) y se detallan los artefactos a generar en cada momento independientemente de las herramientas o métodos que se utilicen para ello.

1.5.2. Visual Paradigm 8.0

Es una herramienta 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 costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.[41]

1.5.3. Servidor Web Apache 2.0

El servidor HTTP Apache es un servidor web 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. Su nombre se debe a que Brian Behelendorf quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de Estados Unidos y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de Internet. El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Es un servidor altamente configurable de diseño modular.[42]

1.5.4. PostgreSQL 8.3.8

PostgreSQL se ha beneficiado también de su larga historia. Hoy en día, PostgreSQL es uno de los servidores de base de datos más avanzados disponibles. Aquí están algunas de las características que se encuentran en una distribución estándar de PostgreSQL:[43]

- ✓ Objeto-relación: En PostgreSQL, cada tabla define una clase. PostgreSQL implementa la herencia entre tablas(o, si se quiere, entre las clases). Funciones y operadores son polimórficos.[43]
- ✓ Código abierto: Un equipo internacional de desarrolladores de PostgreSQL realiza el mantenimiento. Una de las ventajas de la naturaleza de PostgreSQL de código abierto es que el

talento y el conocimiento pueden ser contratados, según sea necesario. El hecho de que este equipo es internacional asegura que PostgreSQL es un producto que puede ser utilizado de manera productiva en cualquiera de las lenguas naturales, no sólo inglés.[43]

- ✓ PostgreSQL soporta el desarrollo de aplicaciones cliente en varios idiomas. Actualmente PostgreSQL se puede conectar a C ++, ODBC, Perl, PHP, Tcl/Tky Python.[43]

1.5.5. MongoDB 1.2.2

MongoDB es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre, lo que significa que cada entrada o registro puede tener un esquema de datos diferente, con atributos o "columnas" que no tienen por qué repetirse de un registro a otro. Sus principales características son la escalabilidad, velocidad, alto rendimiento y funcionalidad y posee un sistema de consultas dinámicas y respuesta rápida.[44]

1.5.6. Subversión

Subversión es un software gratis y de código abierto de sistema de control de versiones, en inglés version control system (VCS). Subversión maneja ficheros y directorios, y los cambios introducidos en ellos, con el tiempo. Esto le permite recuperar versiones antiguas de sus datos o examinar la historia de cómo cambiaron sus datos. Subversión puede operar a través de redes, que permite que sea utilizado por personas en diferentes equipos. En algún nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos de sus respectivas ubicaciones fomenta la colaboración.[45]

1.5.7. Marco de trabajo Sauxe1.5

Sauxe es un Marco de Trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo.[46]

1.5.8. Zend Framework 1.7

Zend Framework es una librería de código abierto de componentes escritos en PHP5, orientada a objetos (OO), que facilita el desarrollo de sitios web. Zend Framework hace hincapié en la calidad del código, a través de una batería de test unitarios, utilizando PHPUnit, cubriendo alrededor del 85% del código escrito para el Framework.[47]

1.5.9. Doctrine 0.9

Doctrine es un potente y completo sistema ORM (object relational mapper) para PHP 5.2+ con un DBAL (database abstraction layer) incorporado. Se está empezando a ver su potencial, pero de la

documentación se puede decir que tiene todas las características necesarias para ser funcional en casi cualquier proyecto. Entre muchas otras funcionalidades tiene la posibilidad de exportar una base de datos existente a sus clases correspondientes y también inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Por otro lado, como la librería es bastante grande, ésta tiene un método para ser "compilada" al pasar a producción.[48]

1.5.10. ExtJS2.2

ExtJS es una librería JavaScript ligera y de alto rendimiento, construida para el desarrollo de aplicaciones web, compatibles con la mayoría de los navegadores. Incorpora un conjunto de controles para el desarrollo de interfaces web.[49] Esta librería incluye:[50]

- ✓ Componentes UI del alto performance y personalizables.
- ✓ Modelo de componentes extensibles.
- ✓ Un API fácil de usar.
- ✓ Licencias Open source y comerciales.

1.5.11. NetBeans 7.0

El IDE NetBeans es un entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. Consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, de escritorio y aplicaciones móviles utilizando la plataforma Java, así como PHP, JavaScript y Ajax, entre otros.[51]

1.6. Lenguajes de modelado y desarrollo

1.6.1. Unified Modeling Language (UML) 5.0

Es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. El estándar ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.[52]

1.6.2. PHP 5.2.6

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser incrustado en páginas HTML. La

mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP.[53]

1.6.3. XML

Extensible Markup Language (XML) es un formato simple de texto muy flexible derivado de SGML (ISO 8879). Originalmente diseñado para afrontar los retos de la gran edición electrónica, XML también está desempeñando un papel cada vez más importante en el intercambio de una amplia variedad de datos en la web y en otros lugares. XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información, además tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.[12]

1.7. Conclusiones Parciales

El estudio de los principales modelos y estándares que proveen soluciones para la gestión del control de acceso, permitió arribar a las siguientes conclusiones:

1. El proceso de control de acceso en un SI se refiere a la protección de los recursos del sistema contra el acceso no autorizado. Este está determinado por tres componentes fundamentales, la identificación y autenticación, la auditoría y la autorización, en este último se enmarcó el análisis teórico de esta investigación. De manera que la autorización es el proceso de determinar si un usuario ya identificado y autenticado se le permite acceder a recursos de información de una manera específica.
2. En la actualidad existen diversas soluciones desarrolladas para fortalecer el proceso de autorización, entre ellas se encuentran los modelos MAC y DAC los cuales presentan limitaciones por lo que no satisfacen todas las necesidades para la gestión de seguridad en sistemas de información debido a que DAC es apropiado para escenarios donde existan pocos recursos y MAC se concentra en lograr la confidencialidad de la información por lo que deja expuesta su integridad .La bibliografía consultada permitió asegurar que el modelo RBAC brinda la posibilidad de especificar restricciones sobre la relación usuario/rol y sobre la activación de un conjunto de roles de usuario. Además este último es una tecnología de forma jerárquica que describe un grupo de usuarios que pueden desempeñar un conjunto de roles y realizar operaciones en las que utilizan un conjunto de objetos como recurso. Por estas ventajas el sistema ACAXIA implementa el modelo RBAC.

3. Mediante el análisis del marco teórico se verificó que existen varios estándares para gestionar el control de acceso, pero a pesar de la diversidad no se detectó ningún protocolo que integre de forma coherente los procesos de identificación y autenticación, autorización y auditoría. Pues cada uno de estos cubren procesos específicos aunque contribuyen al fortalecimiento del control de acceso en un sistema ya que es posible integrarlos por su alto nivel de compatibilidad.
4. Es posible concluir que la utilización del estándar SAML proporciona interfaces que permiten a los interesados presentar sus solicitudes de autenticación y autorización. SAML aborda autenticación y proporciona un mecanismo para la transferencia de decisiones de autorización. Este estándar posibilita la estandarización de los mensajes de autenticación y las solicitudes de autorización se procesan internamente y están dirigidas por el estándar XACML, el cual se centra en el mecanismo para llegar a las decisiones de autorización.
5. Es gratificante documentar que el estándar XACML tiene como objetivo específico promover un mecanismo unificado de control de acceso de manera que pueda acomodarse a una amplia variedad de sistemas y dispositivos. Además XACML también define como objetivo fundamental la estructura de intercambio de mensajes de autorización y un modelo para organizar y almacenar la información de autorización, estas especificaciones del modelo propone resolver el problema planteado en la investigación.
6. El área de impacto de esta investigación se centra en la estandarización de los mensajes de autorización basado en XACML. Por esta razón se especifica integrar el estándar SAML para proteger, transportar, y solicitar instancias de un esquema XACML y con RBAC para obtener las políticas de autorización.
7. Este trabajo está enmarcado en el desarrollo de una nueva solución de autorización para el sistema de seguridad ACAXIA, basada en la implementación del estándar XACML, aumentando su capacidad de integración con otros sistemas, y la posibilidad de adaptarse a entornos de despliegue heterogéneos donde existan sistemas legados que implementen el proceso de autorización basado en este estándar.

Capítulo 2: Propuesta de Solución

2.1 Introducción

El presente capítulo brinda una descripción de las principales características de la solución propuesta para lograr un mejor entendimiento de lo que se desea lograr. Además se definen los requerimientos funcionales y no funcionales que regirán el desarrollo del componente Autorización y se desarrolla el modelo de análisis y diseño característico del mismo con los artefactos correspondientes, y se describen los artefactos de implementación de la solución en cuestión.

2.2 Descripción de la solución propuesta

Este trabajo está enmarcado en el desarrollo de una nueva solución de autorización para el sistema de seguridad ACAXIA basada en la implementación del estándar XACML con el objetivo de proveer un mecanismo unificado de control de acceso de manera que pueda aumentar su capacidad de integración con otros sistemas y la posibilidad de adaptarse a entornos de despliegue heterogéneos donde existan sistemas legados que implementen el proceso de autorización basado en este estándar. La implementación de XACML propone definir una estructura de intercambio de mensajes y un modelo para organizar y almacenar la información en el sistema ACAXIA.

Se propone integrar el estándar SAML pues este posibilita que se procesen internamente las solicitudes de autorización a través del protocolo de intercambio de mensajes que define. Además se especifica la utilización del modelo de control de acceso RBAC para obtener las políticas de autorización, este modelo brinda la posibilidad de especificar restricciones sobre la relación usuario/rol y sobre la activación de un conjunto de roles de usuario. En el siguiente epígrafe se describe el componente Autorización basado en el estándar XACML que se propone como nueva solución para el sistema ACAXIA, integrado a un sistema externo de cualquier entidad denominado “APP Externa”.

2.3 Componente de Autorización basado en el estándar XACML.

El desarrollo creciente de los SI y su uso benefactor en las diversas organizaciones genera como consecuencia la necesidad de controlar rigurosamente la seguridad y conocer cuáles son los derechos de acceso sobre los servicios y recursos del personal o sistema que los solicite. En la actualidad cada entidad establece sus propios mecanismos de seguridad, diversificándose la gestión del proceso de autorización, proporcionando que constituya un reto inigualable diseñar una estructura que permita implementar y

administrar políticas de autorización que preserven la seguridad de los recursos activos y que pueda acomodarse a una amplia variedad de sistemas y dispositivos.

Estas características proporcionan un avance significativo en el diseño e implementación del componente Autorización basado en el estándar XACML del sistema ACAXIA el cual permite:

- ✓ Integrarse a cualquier sistema externo.
- ✓ Una estructura base pero con flexibilidad suficiente para que cada sistema exprese las políticas de autorización de la forma más conveniente a su dominio.
- ✓ Administrar políticas de autorización que preserven la seguridad de los recursos activos.
- ✓ Un modelo para organizar y almacenar la información de autorización.

Para facilitar la comprensión del componente antes mencionado, la Figura 2.9[12] ilustra la estructura y el flujo de mensajes correspondiente al funcionamiento del proceso de autorización que se propone implementar en ACAXIA integrado a un sistema denominado “App Externa” que representa un sistema o dispositivo externo.

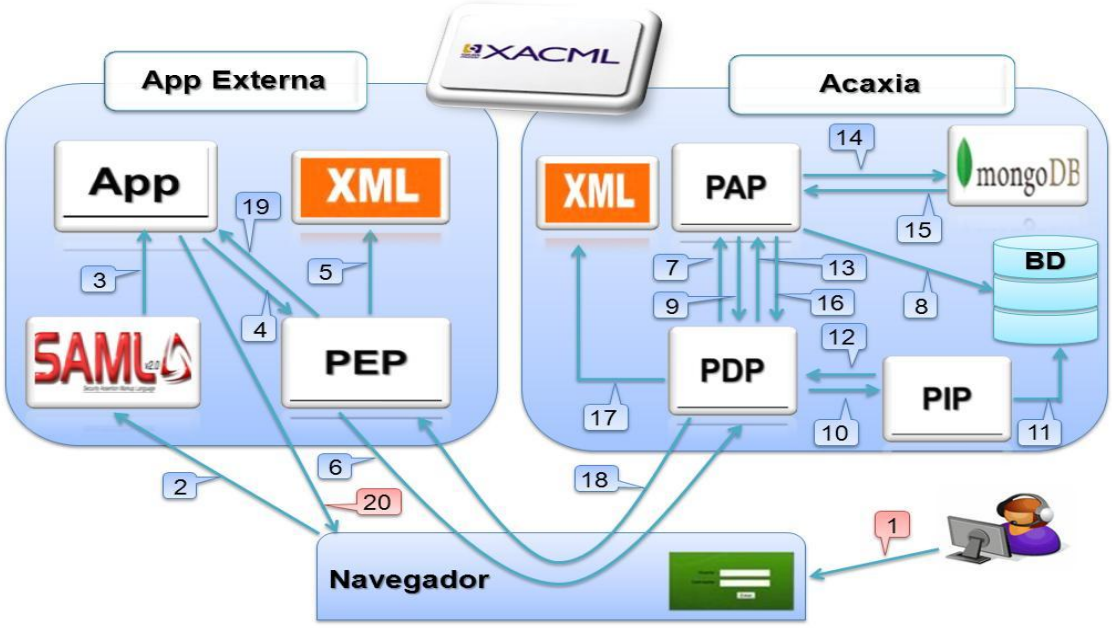


Figura 2.9: Componente de autorización basado en el estándar XACML

12 Fuente: Elaboración propia basada en la figura 1.5 representada en el capítulo 1: Fundamentación Teórica.

Se especifican a continuación cada uno de los conceptos que se ilustran en la figura 2.9 con el objetivo de contribuir a un fácil entendimiento:

- ✓ **XACML (Lenguaje de Marcaje de Control de Acceso Extensible):** Es un estándar para facilitar el intercambio de mensajes con otros sistemas y utilizar un lenguaje unificado y portable para expresar políticas de autorización.[12]
- ✓ **App Externa:** Representa un sistema o dispositivo externo el cual utiliza las funcionalidades del sistema ACAXIA para gestionar la seguridad de sus recursos y servicios.
- ✓ **SAML (Lenguaje de Enmarcado de Aserciones de Seguridad):** Es un marco basado en XML para la comunicación de la autenticación de usuarios. Describe la forma de intercambiar información de identificación, autenticación y autorización.[54]
- ✓ **App:** Clase encargada de establecer la configuración del sistema, además es la clase responsable de realizar un llamado al PEP y retribuirle los datos del usuario.
- ✓ **PEP:** El subsistema Punto de Control de Política deriva el pedido de autorización al PDP. Luego de obtener la respuesta del PDP elabora una repuesta para el sistema que hizo el pedido de autorización.[12]
- ✓ **XML (Lenguaje de Marcaje Extensible):** Lenguaje de programación diseñado especialmente para los documentos de la web. Permite que los diseñadores creen sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos entre aplicaciones y entre organizaciones. Se crea para almacenar la petición del usuario hacia un recurso y la respuesta a dicha petición.[12]
- ✓ **ACAXIA:** Este tiene como objetivo general garantizar la gestión centralizada de la seguridad en un entorno de varias aplicaciones.[54]
- ✓ **PDP:** El subsistema Punto de Decisión de Política evalúa un pedido de autorización a partir de las políticas definidas.[12]
- ✓ **PAP:** El subsistema Administración Punto de Política administra las políticas de control.[12]
- ✓ **PIP:** El subsistema Punto de Información de Política es responsable de buscar e interpretar información de otras fuentes de datos y obtener el valor.[12]
- ✓ **MongoDB:** Es un sistema de gestión de base de datos multiplataforma orientado a documentos, el cual utiliza ACAXIA para almacenar las obligaciones que tiene un usuario asignándole un rol en una entidad.[55]

- ✓ **BD:** PostgreSQL es uno de los gestores de base de datos más avanzados que utiliza el sistema ACAXIA para almacenar y gestionar la información.
- ✓ **Navegador:** Herramienta de interacción del usuario con la aplicación.

A continuación se presenta un diagrama de secuencia reflejado en la Figura 2.10[13] que muestra la interacción entre los diferentes conceptos mencionados en el proceso de autorización descrito en la Figura 2.9 y con el objetivo de fomentar la comprensión del funcionamiento de dicho proceso en el momento que se necesita acceder a un recurso.

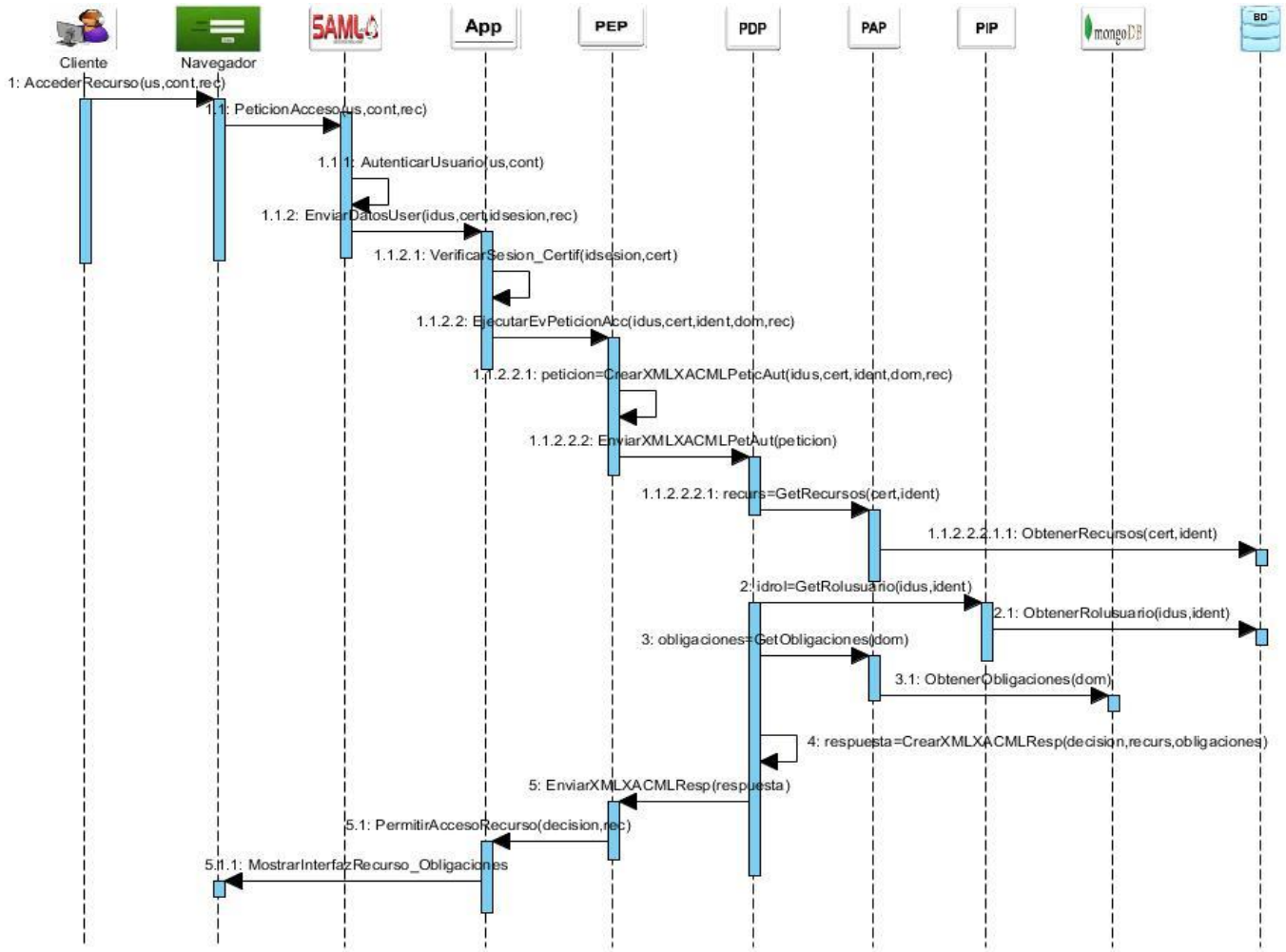


Figura 2.10: Flujo de mensajes en el proceso de autorización basado en el estándar XACML

13 Fuente: Elaboración propia basada en EXtensible Access Control Markup Language (XACML) Version 2.0, Moses, T., O.A.C. TC, Editor. 2005, OASIS: USA.

2.4 Mapa Conceptual

A continuación la Figura 2.11[14] muestra un mapa conceptual que relaciona los principales conceptos identificados a partir de las definiciones fundamentales que se van a tratar a lo largo de la investigación. Para alcanzar la calidad del proceso de desarrollo será guiado por los estándares y normativas dictadas por el Departamento de Tecnología del CEIGE. Estas normativas regulan el diseño de interfaces de usuarios, la nomenclatura de las clases, componentes y objetos a nivel de datos, estilos de codificación, validaciones a todos los niveles y la forma en la que se deben describir cada uno de los componentes desarrollados para facilitar el mantenimiento y la reutilización.

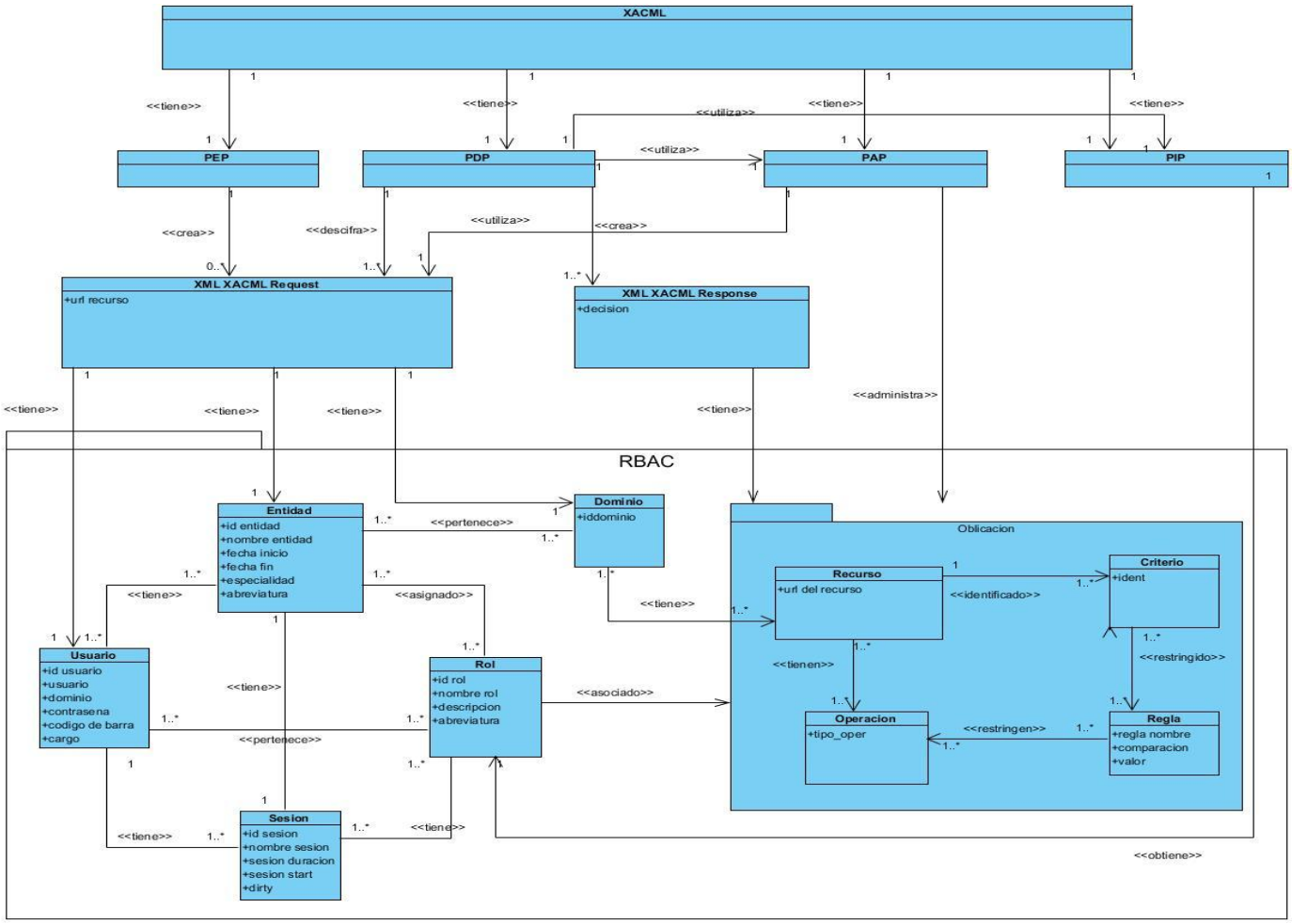


Figura 2.11: Mapa Conceptual

A continuación se definen algunos conceptos mencionados en la figura anterior:

14 Fuente: Elaboración propia basada en la figura 2.9 representada en el capítulo 2: Propuesta de Solución

- ✓ **Criterios:** Son atributos que identifican únicamente a un recurso. Ejemplo: de una cuenta el código, el estado, la organización, el usuario que la creó, la fecha, entre otros.[54]
- ✓ **Dominios:** Constituyen agrupaciones de organizaciones que tienen en común algún criterio. Ejemplo: las organizaciones de una región, de un ministerio, de un sector de la sociedad, entre otros.[54]
- ✓ **Entidad:** Forma parte de la estructura de un país y puede comportarse como un ministerio, entidad, área o cargo.[55]
- ✓ **Operaciones:** Son acciones que se realizan sobre uno o varios recursos, pueden iniciarse debido a la petición de un usuario o por configuraciones internas del SI. Ejemplo: adicionar, modificar, mostrar, eliminar, asentar, aprobar, entre otras.[54]
- ✓ **Permiso u Obligación:** es un concepto que agrupa las operaciones que se pueden realizar sobre uno o varios recursos que cumplen con una o varias reglas. Ejemplo: mostrar las cuentas de una organización (X) que fueron registradas en una fecha (Y).[54]
- ✓ **RBAC:** Es un modelo que propone que un grupo de usuarios puedan desempeñar un conjunto de roles y realizar operaciones en las que utilizan un conjunto de objetos como recurso. La definición básica de RBAC establece que los usuarios son asignados a roles, los permisos son asociados a roles y los usuarios adquieren permisos siendo miembros de roles[54].
- ✓ **Recurso:** Elemento del sistema sobre los que se establecen políticas de acceso. Puede ser una tabla, un documento, un fichero, etc.[54]
- ✓ **Regla:** Representan restricciones que evalúan uno o varios criterios. Ejemplo: las cuentas que estén en un estado (X), que hayan sido creadas por un usuario (Y) en una organización (Z).[54]
- ✓ **Rol:** Agrupa una serie de permisos sobre sistemas, funcionalidades y acciones que se le asignarán a un conjunto de usuarios.[55]
- ✓ **Sesión:** Mapea los privilegios que tiene un usuario en un espacio de tiempo, sobre los recursos de una o varias organizaciones debido a los roles que desempeña o por privilegios asignados directamente él.[54]
- ✓ **Usuario:** Cualquier persona que interactúa con el sistema y desempeña un rol determinado en el mismo.[55]
- ✓ **XML XACML Request:** Primer XML que se crea para almacenar la petición del usuario hacia un recurso.

- ✓ **XML XACML Response:** XML que se crea en respuesta al XML XACML Request Authorize User.

2.5 Requisitos de software.

Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales describen qué es lo que el sistema debe hacer para dar soporte a las funciones y objetivos del usuario. Los requisitos no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados.[56]

2.5.1 Requisitos funcionales

Los requisitos funcionales son los que definen las funciones que el sistema será capaz de realizar.

En el componente Autorización se identificaron un total de once requisitos funcionales los cuales se listan a continuación:

R1 Gestionar autorización de usuario.

Requisitos funcionales del subsistema PEP.

- R1.1 Crear XML XACML Request Authorize User.
- R1.2 Remitir XML XACML Request Authorize User.
- R1.3 Validar XML XACML Response Authorize.

Requisitos funcionales del subsistema PDP.

- R1.1 Descifrar XML XACML Request Authorize User.
- R1.2 Validar XML XACML Request Authorize User.
- R1.3 Crear XML XACML Response Authorize.
- R1.4 Remitir XML XACML Response Authorize.

Requisitos funcionales del subsistema PIP.

- R1.1 Obtener rol de usuario.

Requisitos funcionales del subsistema PAP.

- R1.1 Cargar recursos en caché.
- R1.2 Cargar obligaciones.

A continuación se describe el requisito Gestionar autorización de usuario, el resto de las descripciones se encuentran en el expediente de proyecto ubicado en:

http://10.58.19.250/svn/cigcentral/CSGTEC/Acaxia_v2.2.1.1_Beta/ExpedienteProyecto/

- ✓ **Descripción del requisito Gestionar autorización de usuario.**

Precondiciones	N/A
Flujo de eventos	
Flujo básico Gestionar autorización del usuario	
1. El usuario intenta acceder a un recurso.	
2. La clase ZendExt_APP.php del sistema verifica si la sesión está activa y si el certificado es válido.	
3. El APP del sistema realiza un llamado al subsistema PEP.	
4. El PEP valida si existen recursos en la caché.	
5. El PEP procede a crear el XML XACML Request Authorize User.(Ver descripción del requisito Crear XML XACML Request Authorize User).	
6. El sistema PEP envía la petición de acceso al susbsitema PDP para su posterior evaluación (Ver descripción del requisito Remitir XML XACML Request Authorize User).	
7. Se procede a descifrar el XML XACML Request Authorize User en el subsistema PDP (Ver descripción del requisito Descifrar XML XACML Request Authorize User.)	
8. El PDP valida la petición de acceso descifrada (Ver descripción del requisito Validar XML XACML Request Authorize User.)	
9. El PDP procede a diseñar la respuesta de acceso en formato XML.	
10. El sistema PDP crea el XML XACML Response Authorize (Ver descripción del requisito Crear XML XACML Response Authorize.)	
11. Se envía la respuesta correspondiente a la petición de acceso sobre el recurso solicitado al subsistema PEP (Ver descripción del requisito Remitir XML XACML Response Authorize User.)	
12. El subsistema PEP procede a validar la respuesta enviada por el PDP (Ver descripción del requisito Validar XML XACML Response Authorize.)	
Pos-condiciones	
N/A	
Flujos alternativos	
Flujo alternativo 3.a La sesión expiró o el certificado es inválido.	
1. El sistema re-direcciona a la ventana de logueo.	
2. El usuario inserta usuario y contraseña.	
3. Volver al paso 1 del flujo básico.	
Pos-condiciones	
N/A	
Flujo alternativo 4.a Hay recursos registrados en caché	
1. El sistema PEP valida si el recurso solicitado se encuentra en la caché.	
2. Se brinda acceso al recurso.	
3. Concluye el requisito.	
Pos-condiciones	
1 Se brinda acceso al recurso	
Flujo alternativo 4.a.a El recurso solicitado no se encuentra en la caché.	
1 El sistema reenvía un mensaje de Error" Acceso Denegado. El usuario no tiene acceso al recurso solicitado"	
Pos-condiciones	
1. Se muestra un mensaje de error y se deniega el acceso al recurso solicitado.	
Validaciones	
N/A	
Conceptos **XML XACML Request Authorize**	Visibles en la interfaz: N/A Utilizados internamente:

		Id de usuario (Token), id de certificado (Token), url del recurso, id de entidad (Token), id de dominio (Token).
	XML XACML Response Authorize	Visibles en la interfaz: N/A Utilizados internamente: decisión, recursos, obligaciones
	Sesión	Visibles en la interfaz: N/A Utilizados internamente: Id de sesión (Token), nombre.
	Certificado	Visibles en la interfaz: N/A Utilizados internamente: Id de certificado (Token).
	Recurso	Visibles en la interfaz: N/A Utilizados internamente: url de recurso.
	Obligación	Visibles en la interfaz: N/A Utilizados internamente: operación, recurso ,regla
	Regla	Visibles en la interfaz: N/A Utilizados internamente: Nombre de regla, criterio, comparación(operador),cadena o valor
	Rol	Visibles en la interfaz: N/A Utilizados internamente: Id de rol (Token)
Requisitos especiales	N/A	

Asuntos pendientes	N/A

2.5.2 Requisitos no funcionales

Son las propiedades o cualidades que el producto debe tener. A continuación se detallan algunos requisitos que el sistema posee.

Software.

- ✓ **Para el cliente:**
  - **RNF1** Navegador Mozilla Firefox.
  - **RNF2** Sistema operativo Windows o Linux.
- ✓ **Para el servidor:**
  - **RNF3** Sistema operativo Windows Advancer Server (2000 o superior) o Linux en cualquiera de sus distribuciones.
  - **RNF4** Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible.
  - **RNF5** Un gestor de base de datos PostgreSQL 8.0 o superior, se recomienda usar 8.3.9.

Hardware.

- ✓ **Para el servidor:**
  - **RNF6** Requerimientos mínimos: Procesador Pentium IV a 1GHz de velocidad de procesamiento y 1Gb de memoria RAM.
  - **RNF7** Al menos 40 Gb de espacio libre en disco duro.
- ✓ **Para el cliente:**
  - **RNF8** Requerimientos mínimos: Procesador Pentium IV a 1730Mhz con 1 Gb de memoria RAM.

Seguridad.

- ✓ **Confiabilidad:**

RNF9 La información manejada por el sistema está protegida contra acceso no autorizado.

- ✓ **Disponibilidad:**

RNF10 Los usuarios autorizados tendrán acceso a la información en todo momento.

- ✓ **Integridad:**

RNF11 La información puede ser creada, modificada y eliminada solo por las personas autorizadas.

Usabilidad.

RNF12 El componente podrá ser utilizado por cualquier usuario autorizado con conocimientos informáticos básicos.

RNF13 Debe garantizar un acceso fácil y rápido.

Rendimiento.

RNF14 Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 4 segundos.

Portabilidad.

RNF15 El componente debe ser multiplataforma, haciendo énfasis en Linux y Windows.

Estándares aplicables.

RNF16 El sistema debe utilizar el estándar SAML para la estandarización del proceso de autenticación.

RNF17 El sistema debe utilizar el estándar XACML para la estandarización del proceso de autorización.

RNF18 El sistema debe utilizar el modelo RBAC para la gestión del proceso de autorización.

2.6 Modelo de diseño

El diseño de software consiste en aplicar distintas técnicas y principios, con el objetivo de definir un producto con los suficientes detalles como para permitir su realización física[57]. Es el proceso de definición de la arquitectura software: componentes módulos, interfaces, procedimientos de prueba y datos de un sistema que se crean para satisfacer unos requisitos especificados.[58].

2.6.1 Estilos arquitectónicos

Los estilos arquitectónicos se consideran transformaciones definidas para el diseño de un sistema. Los estilos se encuentran en el centro de la arquitectura y constituyen buena parte de su sustancia. Estos profundizan en establecer una estructura para todos los componentes identificados en el sistema y una forma de organizar su arquitectura.[59]

El estilo arquitectónico definido por el Departamento de Tecnología es el **Estilo en Capas**.[60] Donde se identificaron 3 niveles o capas fundamentales, a continuación se describen y se ilustran en la Figura 2.12 [15] para un mayor entendimiento:

1. **Capa de Datos:** donde estarán ubicados el servidor de base de dato PostgreSQL y un conjunto de ficheros de configuración del sistema.

15 Fuente: Arquitectura ERP, Mena López, Grette Leydi y Tenrrero Cabrera, Marianela, 2008

2. **Capa de Acceso a Datos:** donde estará presente el framework Doctrine, utilizado para la comunicación con el servidor de datos mediante el protocolo PDO[16], también estará un Gestor de Configuración que es el encargado de comunicarse vía XML con los ficheros de configuración del sistema, y el último elemento de esta capa es la Fachada de Transferencia de Acceso Rápido cuya función es el acceso rápido tanto a los ficheros de comunicación como al servidor de datos, a través del Gestor de Configuración y Doctrine para cada caso en específico.
3. **Capa de Funcionamiento:** en esta capa se emplea el patrón de arquitectura Modelo Vista Controlador (MVC), así como el elemento de Acceso de Respuesta Rápida el cual brinda informaciones de forma vertiginosa violando las restricciones que impone el estilo en capas propuesto. De forma vertical al modelo descrito hasta este momento, estará el módulo de cacheo del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación web.

Esta gran estructura descrita, se comunica con una series de servicios web mediante protocolos SOAP[17], REST[18], HTTP[19] y HTTPS[20], que interactúan con el sistema proporcionándole un conjunto de funcionalidades tanto de seguridad como de negocio.

16PDO: Process Data Objects. Protocolo de comunicación especialmente adecuado para la transmisión rápida de datos; éste modelo define un productor y uno o varios consumidores PDO.

17 SOAP: Simple Object Access Protocol. Permite la comunicación entre aplicaciones a través de mensajes por medio de Internet.

18 REST: Transferencia Representacional de Estados. Sirve para acceder a servicios sencillamente a través de URLs.

19 HTTP: Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto. Protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP.

20 HTTPS: Versión segura del protocolo HTTP que implementa un canal de comunicación seguro y basado en SSL (Secure Socket Layers) entre el navegador del cliente y el servidor HTTP.

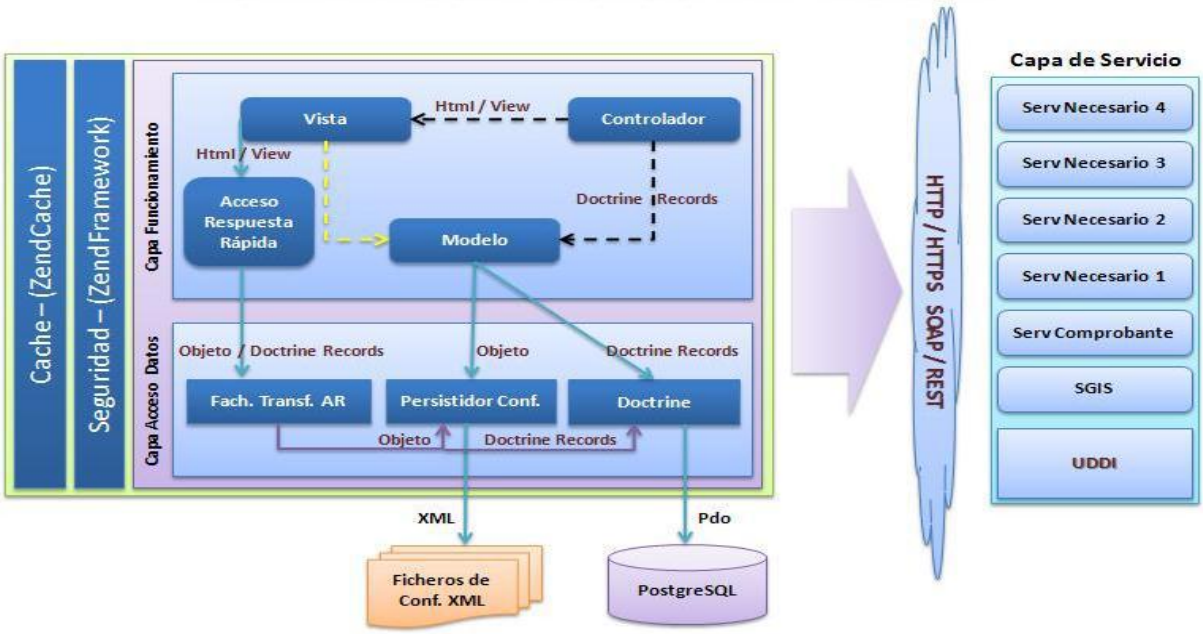


Figura 2.2: Arquitectura en Capas

2.6.2 Patrón arquitectónico

- ✓ **Patrón arquitectónico Modelo-Vista-Controlador (MVC).**

El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes [59]:

- ✓ **Modelo**. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- ✓ **Vista.** Maneja la visualización de la información.
- ✓ **Controlador.** Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Este patrón es empleado en el marco de trabajo SAUXE y determina la estructura de los paquetes internos de los componentes a desarrollar.

2.6.3 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño[61]. Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

- ✓ **Patrones GRASP**

Para el desarrollo del componente Autorización se utilizarán varios de los patrones generales de software para asignar responsabilidades o GRASP (object-oriented design General Responsibility Assignment Software Patterns). Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. Entre ellos se mencionan aquellos utilizados para el desarrollo de la solución:[55]

- **Experto:** Asigna una responsabilidad al experto en información, que es la clase que posee la información necesaria para cumplir con la responsabilidad[55]. Se puede ver el uso de este patrón en todas las clases a utilizar en el componente ya que cada clase conoce su información y es la encargada de implementar las funcionalidades que brindan información de las mismas.
- **Creador:** Tiene como objetivo asignar a la clase B la responsabilidad de crear una instancia de clase A[55]. Su uso se puede ver en la clase ZendExt_XACMLPDP_MainPDP, ya que esta clase se encarga de crear objetos de clases como ZendExt_XACMLPAP_MainPAP, ZendExt_XACMLPIP_MainPIP.
- **Alta Cohesión:** Asigna responsabilidad de modo que se mantenga alta cohesión[55]. El uso de este patrón indica que la información almacenada en las clases debe ser coherente y relacionada a lo que se maneja en dicha clase, en la solución se puede evidenciar en la clase ZendExt_XACMLPDP_MainPDP ya que agrupa toda la información de la petición de forma coherente y relacionada entre sí.
- **Bajo Acoplamiento:** Asigna responsabilidades de modo que se mantenga bajo acoplamiento[55]. El uso de éste patrón se evidencia en la poca relación existente entre las clases que conforman el componente.

✓ **Patrones GOF**

Los patrones de diseño GoF se dieron a conocer a principios de los años 90 con el libro “Design Patterns. Elements of Reusable Object-Oriented Software.”. En dicho libro se hace una recopilación de 23 patrones de diseño comunes, clasificados en tres grupos de acuerdo a su naturaleza:[61]

- **Creacionales:** concierne al proceso de creación de objetos.
- **Estructurales:** tratan la composición de clases y/o objetos.
- **De Comportamiento:** caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

A continuación se especifican los patrones GoF utilizados en el desarrollo de la solución.

1. El patrón creacional **Singleton o Solitario.**[61]

Es un patrón creacional que tiene como propósito garantizar una única instancia de una clase, proporcionando un punto de acceso global a la misma. Se puede ver su aplicación en el uso en la clase ZendExt_IOC y todos los subsistemas del componente, donde se garantiza que cada clase tenga una instancia única y permite que la misma sea accedida desde cualquier parte del sistema.

2. El patrón estructural **Fachada.**[61]

Crea una única clase que permite acceder a un conjunto numeroso y complicado de clases. En la solución se pone de manifiesto este patrón en las clases pertenecientes al paquete “services” del componente las cuales sirven de fachada a las funcionalidades brindadas por el mismo al resto de los componentes.

En el desarrollo del componente Autorización se puede especificar el uso de dicho patrón en la clase SeguridadProxyService, la cual se utiliza como interfaz de acceso a las demás interfaces y componentes del sistema y garantiza la comunicación con sistemas externos.

Como resultado el uso de estos patrones en la solución de autorización propuesta garantiza asignar a cada clase la responsabilidad que le corresponde, obtener el menor número de relaciones y dependencias entre clases y aumentar las posibilidades de reusabilidad de las mismas.

2.6.4 Diagrama de Clases

El diagrama de clases se considera como el diagrama principal de diseño y análisis para un sistema. Se desarrolla con el objetivo de describir la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

A continuación se muestra en la Figura 2.13[21] un diagrama de clases con estereotipos web en el cual se describe el proceso de autorización del sistema ACAXIA basado en el estándar XACML donde se propone definir una estructura de intercambio de mensajes. El diagrama permite visualizar las relaciones entre las clases involucradas en ACAXIA integrado a un sistema externo denominado SAUXE.

SAUXE es un marco de trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo.[54]

21 Fuente: Elaboración propia basada en la figura 2.9 representada en el capítulo 2: Propuesta de Solución

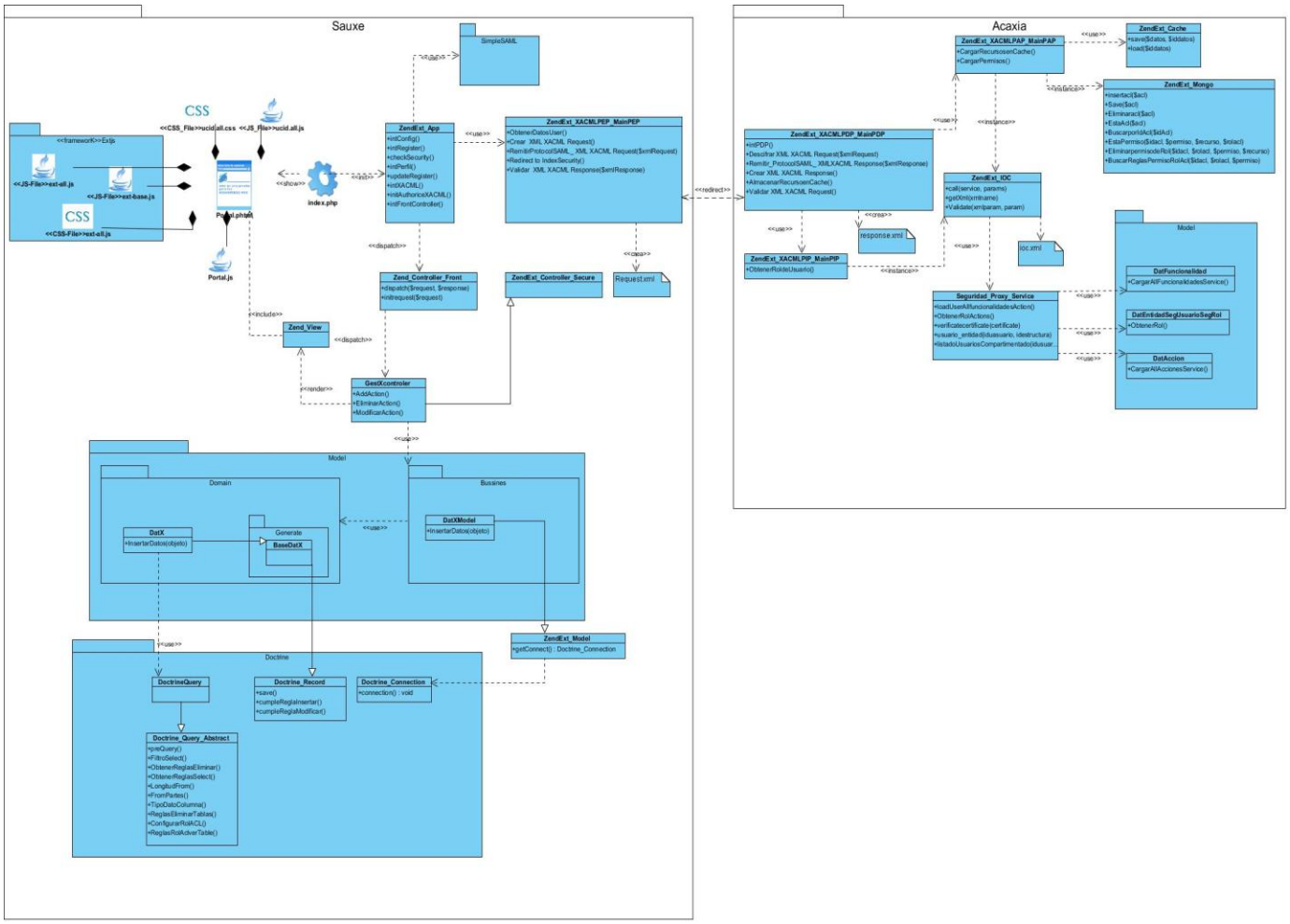


Figura 2.3: Diagrama de Clases

- ✓ **Descripción de las clases del diseño asociadas a la solución.**

Tabla 1: Clases del diseño.

Clases	Descripción
Framework ExtJs	Librería JavaScript que utiliza la clase portal.phtml.
Portal.phtml	Plantilla HTML encargada de mostrarle al usuario la interfaz correspondiente al componente con la que va a trabajar.
Portal.js	Formulario js a través del cual se maneja la información que introducen los usuarios del sistema.
Index.php	Es la clase servidora que se encarga de toda la configuración de la aplicación, re direccionar y dar seguridad. Además determina cuál es la clase controladora que le corresponde a cada página cliente.
ZendExt_App.php	Se utiliza para gestionar la lógica de negocio y se encarga de capturar los parámetros que le son enviados desde la interfaz. Además verifica si la sesión está activa y si el certificado es válido.

ZendExt_XACMLPEP_MainPEP	Es la clase que gestiona el pedido de autorización, crea un xml con los datos necesarios para realizar la petición de acceso y lo deriva a la clase ZendExt_ XACMLPDP_MainPDP(PDP). Luego al obtener la respuesta del PDP elabora una repuesta para el sistema que hizo el pedido de autorización.
ZendExt_XACMLPDP_MainPDP	Obtiene el XML enviado por la clase ZendExt_ XACMLPEP_MainPEP(PEP) y procede a evalúar un pedido de autorización .Utiliza la clase ZendExt_ XACMLPAP_MainPAP(PAP) para obtener los recursos del sistema y la clase ZendExt_ XACMLPIP_MainPIP(PIP) para obtener el rol asignado al usuario.
ZendExt_ XACMLPAP_MainPAP	Administra los recursos y obligaciones del sistema.
ZendExt_ XACMLPIP_MainPIP	Es la clase usada para obtener el rol asignado al usuario en una entidad.
ZendExt_MongoDB	La clase cuenta con funcionalidades para realizar operaciones de adición, eliminación y modificación de datos almacenados en el gestor Mongodb.
ZendExt_Cache	Es la clase usada para almacenar los recursos y obligaciones obtenidos de la clase PAP
ZendExt_IOC	Es la clase utilizada por las clases PAP y PIP para utilizar los servicios que les brinda la clase Seguridad_Proxy_Service.
Zend_Controller_Front	Se utiliza como clase intermedia entre la clase ZendExt_App.php y la controladora GestXController para ejecutar la acción sobre el recurso solicitado.
Zend_Controller_Secure	Clase de donde heredan todas las clases controladoras.
GestXController	Clase controladora que hereda de ZendExt_Controller_Secure y se define en la url del recurso solicitado. Encargada de comprobar si la acción sobre el recurso solicitado es a nivel de datos y además ejecutar el acceso al recurso.
DatXModel	Clase genérica de acceso a datos en el paquete Bussines creado internamente por el Zend Framework.
Seguridad_Proxy_Service	Esta clase sirve como fachada para la comunicación entre los componentes del sistema.
DatX	Clase genérica de acceso a datos en el paquete Domain. Puede ser cualquier clase del modelo que ejecute internamente la clase GestXController.
DatFuncionalidad	Es la encargada de obtener los recursos del sistema teniendo en cuenta el rol asignado al usuario a una entidad, la cual se obtienen a través de la clase *SeguridadProxyService*.
DatAccion	Es la encargada de obtener las acciones que tienen los recursos del sistema teniendo en cuenta el rol asignado al usuario a una entidad, se obtienen a través de la clase *SeguridadProxyService*.
DatEntidadSegUsuarioSegRol	Es la encargada de establecer las relaciones de cada usuario con los roles asociados a él en una entidad determinada, la cual se obtiene a través de la clase *SeguridadProxyService*.
DoctrineQuery	Permite la creación de consultas hacia la base de datos.
Doctrine_Query_Abstract	Es la clase usada para realizar consultas a la base de datos y presenta funcionalidades para filtrar los resultados de la consulta realizada y para determinar si un usuario tiene derecho a realizar la eliminación de un objeto de la base de datos, entre otras.
Doctrine_Record	Esta clase crea referencias de un objeto almacenado en una tabla de la

	base de datos, puede ser utilizada para conocer si un usuario puede modificar dicho objeto a través de los métodos mostrados en el diagrama.
Doctrine_Connection	Permite la conexión a la base de datos.
Request.xml	Fichero XML que almacenará los datos (idusuario, el certificado, identidad, iddominio, recurso) necesarios para evaluar la petición de acceso.
Response.xml	Fichero XML que almacenará la decisión evaluada, los recursos a los cuales tiene acceso el usuario con sus respetivas obligaciones.
SimpleSAML	Este paquete es el encargado de implementar el proceso de autenticación del usuario. Además crea e activa la sesión y valida el certificado.
Model	Paquete que contiene las clases de acceso a datos.
Doctrine	Paquete que contiene las clases mapeadas por el sistema Mapeador de Objeto Relacional (ORM)

2.6.5 Diagrama de Secuencia

Un diagrama de secuencia muestra una interacción, que representa la secuencia de mensajes entre las instancias de clases, componentes, subsistemas o actores. El tiempo fluye hacia abajo en el diagrama y muestra el flujo de control de un participante a otro.[62]

Se muestra a continuación un diagrama de secuencia en la Figura 2.14[22] que especifica el flujo de intercambio de mensajes entre las clases involucradas en el sistema ACAXIA integrado a un sistema externo SAUXE.

22 Fuente: Elaboración propia basada en la figura 2.10 representada en el capítulo 2: Propuesta de Solución

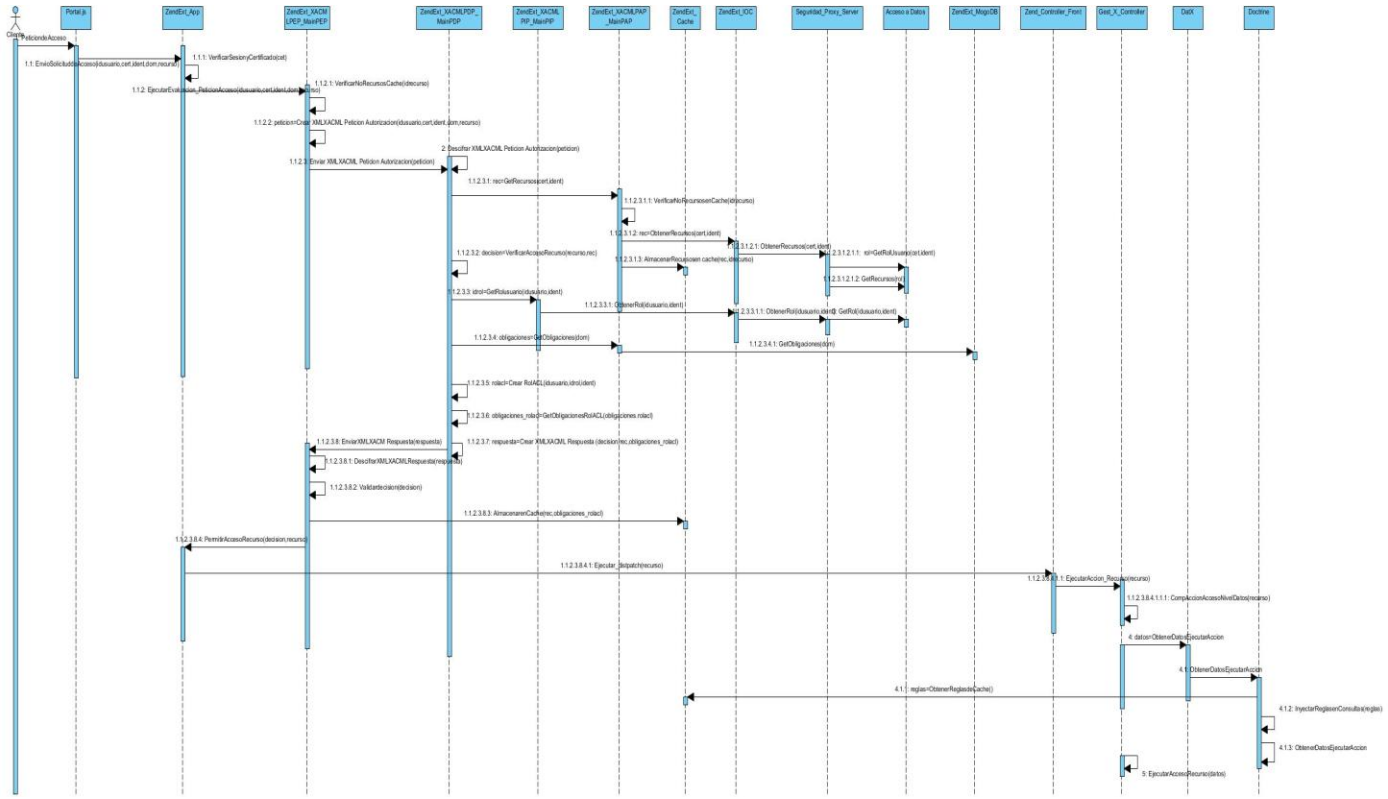


Figura 2.4: Flujo de intercambio de mensaje de autorización

2.6.6 Modelo de Datos.

Podemos definir que un modelo de datos (MD) es conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Los MD comprenden aspectos relacionados con: estructuras y tipos de datos, operaciones y restricciones.[63]

Para el componente Autorización se generan 2 modelos de datos distintos, el primero para el gestor de base de datos PostgreSQL y el segundo perteneciente a la base de datos creada en MongoDB.

- ✓ En la Figura 2.15[23] se modelan las tablas que se encuentran creadas en PostgreSQL

- **SegUsuario:** Almacena toda la información concerniente a los usuarios registrados en el sistema.
- **SegRol:** Contiene la información almacenada de los roles asignados a cada uno de los usuarios.
- **DatFuncionalidad:** Contiene toda la información sobre los recursos del sistema.
- **DatAccion:** Almacena toda la información relacionada con las acciones que se pueden realizar sobre los recursos del sistema.

23 Fuente: Elaboración propia.

- **DatEntidad_SegUsuario_SegRol:** Del servicio perteneciente al sistema Estructura y Composición se obtiene la terna resultante de la relación que existe entre las tablas *SegUsuario* y *SegRol*, donde se especifica que un usuario puede tener un solo rol en una entidad.
- **DatSistema_SegRol_DatFuncionalidad:** Del servicio perteneciente al sistema Seguridad se obtiene la terna resultante de la relación que existe entre las tablas DatSistema,DatFuncionalidad y SegRol para obtener todas las funcionalidades asignadas al rol en el sistema.
- **DatAccion_DatSistema_SegRol_DatFuncionalidad:** Del servicio perteneciente al sistema Seguridad se obtiene la tabla resultante de la relación que existe entre las tablas DatAccion, DatSistema, DatFuncionalidad y SegRol para obtener las acciones correspondientes a cada funcionalidad asignada al rol en el sistema.

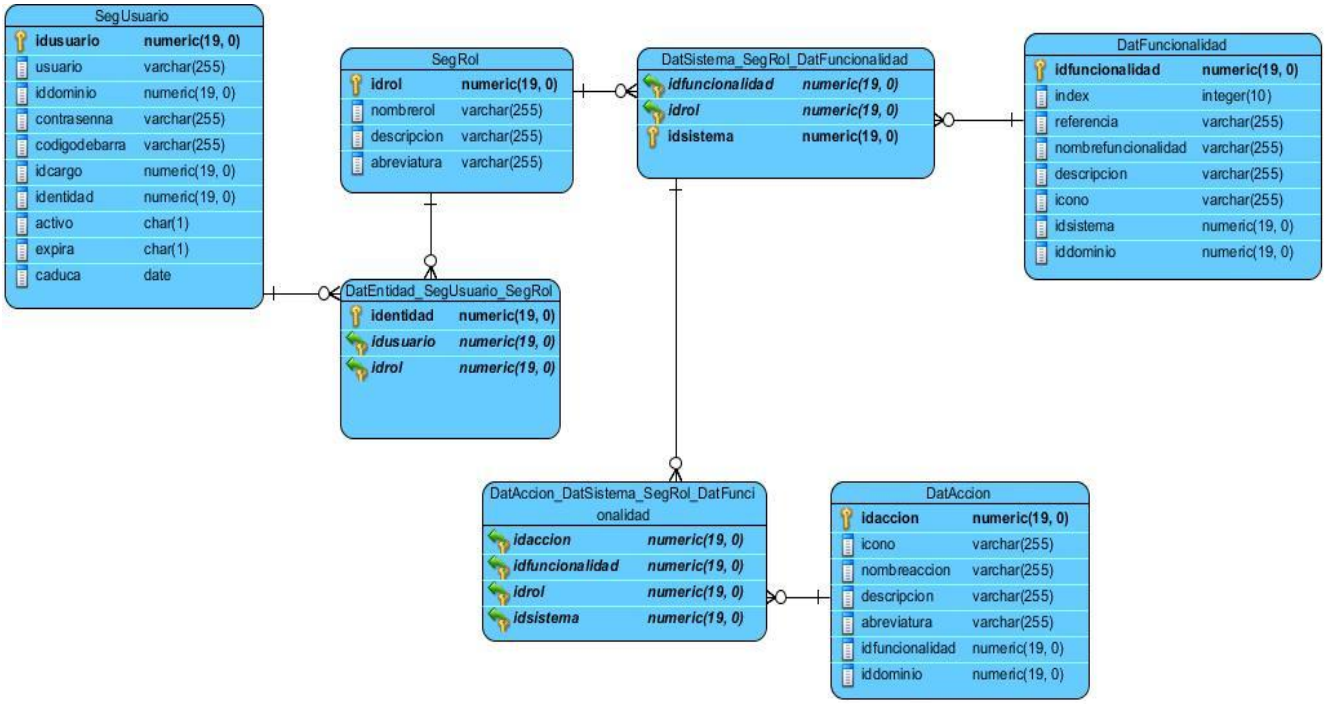


Figura 2.5: Modelo de Datos PostgreSQL

En la Figura 2.16[24] se muestra la tabla acl modelada por el gestor MongoDB. Acl almacena la información referente a la relación creada luego de la asignación de las obligaciones a usuarios que desempeñan un rol en una entidad sobre un determinado recurso.

24 Fuente: Elaboración propia

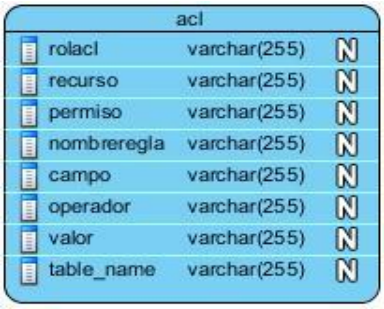


Figura 2.6: Modelo de Datos MongoDB

2.7 Conclusiones Parciales

Finalmente se presentan las conclusiones finales del presente capítulo:

1. Se específica detalladamente las principales características existentes en la nueva solución de autorización que se propone. Además se ilustra la estructura del componente Autorización basado en el estándar XACML y se describe a través del flujo de mensajes que se muestra el proceso de autorización que se desea implementar en ACAXIA integrado en un sistema o dispositivo externo.
2. En el capítulo se realizó el levantamiento y descripción de los requisitos funcionales y no funcionales identificados durante el proceso de análisis y diseño para garantizar una correcta implementación y funcionamiento de la aplicación.
3. Como parte del modelo de diseño elaborado, se estudió el estilo arquitectónico definido para el desarrollo en el departamento y se identificaron los patrones de diseño a utilizar en el desarrollo de la solución.
4. El diseño de clases modelado tiene como objetivo describir la estructura del componente Autorización del sistema ACAXIA basado en el estándar XACML y permite visualizar las relaciones entre las clases involucradas en ACAXIA integrado a un sistema externo denominado SAUXE.
5. Se confeccionó un diagrama de secuencia que especifica el flujo de intercambio de mensajes entre las clases involucradas en el sistema ACAXIA integrado a un sistema externo SAUXE para facilitar la comprensión del componente antes mencionado.
6. Se especificaron los modelos de datos con que contará la solución, los cuales contribuirán en la modelación de las tablas de base de datos utilizada en el proceso de autorización, estas serán almacenas en los gestores de base de datos PostgresSQL y MongoDB.

Capítulo 3: Implementación y Pruebas

3.1 Introducción

En el presente capítulo se exponen los componentes necesarios para realizar la implementación y despliegue de la solución de autorización. Además se especifican los estándares de codificación a utilizar para garantizar un buen entendimiento y legibilidad del código. Se exponen los principales elementos relacionados con la validación del componente Autorización implementado con el objetivo de aumentar la capacidad de integración de ACAXIA con otros sistemas.

3.2 Diagrama de Despliegue

Los diagramas de despliegue están dictaminados como un tipo de diagrama del lenguaje unificado de modelado que modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos.[64] Los diagramas de despliegue son los complementos de los diagramas de componentes que unidos proveen la vista de implementación del sistema. Describen la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos.

A continuación en la Figura 3.17 se ilustra la configuración de los elementos de hardware (nodos) que se deben utilizar para el despliegue de la solución de autorización propuesto para dar solución al problema existente.

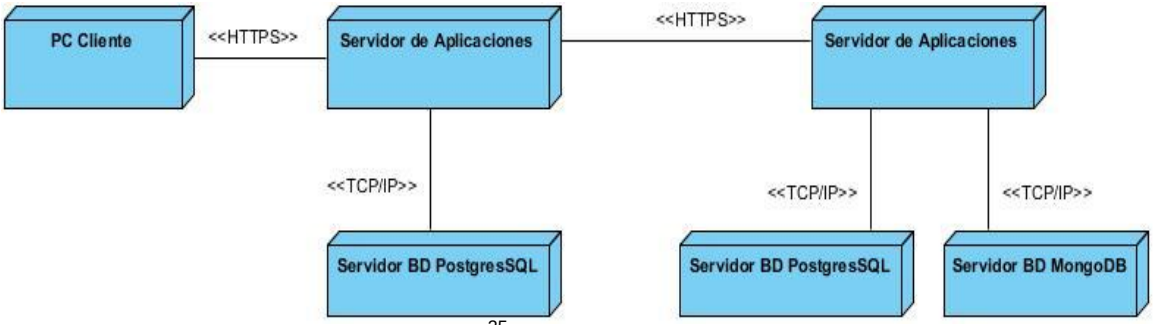


Figura 3.17[25] Diagrama de Despliegue

El usuario accede a un recurso desde su puesto de trabajo al sistema que se encuentra instalado en el servidor de aplicaciones de un sistema externo. Este se comunica a un servidor de aplicaciones donde se encuentra instalado el sistema de seguridad ACAXIA dicho servidor se conecta a los servidores de base

25 Fuente: Elaboración propia

de datos, para realizar las consultas y obtener los resultados deseados, que en este caso es evaluar si la petición de acceso al recurso solicitado es permitida, además se obtienen los permisos otorgados al usuario sobre dicho recurso.

3.3 Diagrama de Componentes

Los diagramas de componentes ilustran las partes del software que conformarán un sistema. Un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clase, usualmente un componente puede estar compuesto por una o más clases (u objetos). Estos son bloques de construcción y eventualmente pueden comprender una gran porción de un sistema.[64]

El diagrama de componente reflejado en la Figura 3.18 representa la arquitectura correspondiente a la solución de autorización dividida en diferentes componentes y especificando además las dependencias entre estos.

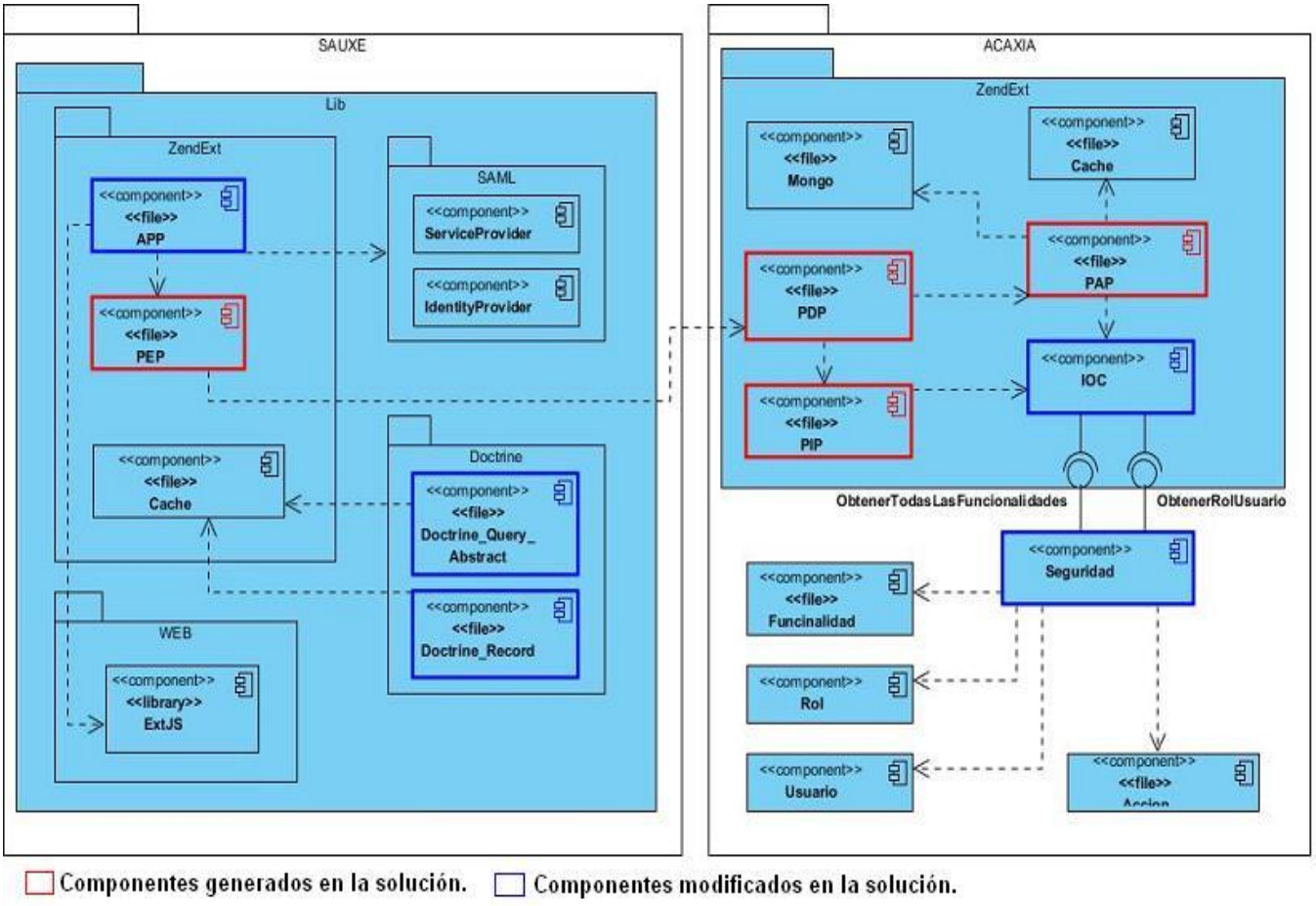


Figura 3.18[26] Diagrama de Componentes.

En el desarrollo del componente Autorización es necesario comprender las organizaciones y dependencias lógicas entre los componentes de software contenidos en ambos sistemas. En el sistema SAUXE se evidencia la interacción de varios componentes como ZendExt, Doctrine, ExtJS y SAML que a su vez se encuentran en una postura dependiente a los componentes especificados en el paquete ZendExt del sistema ACAXIA. Es importante destacar que el componente IOC hace uso de servicios internos brindados por el subsistema Seguridad con el objetivo de obtener información de otros subsistemas suscritos a ACAXIA.

3.4 Estándares de Codificación

Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible.[65] Es decir se asocia el termino de estándares de codificación a 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[66].

En el transcurso de la implementación del componente Autorización se utilizarán algunos de los estándares de codificación y normas propuestos como parte de la línea de arquitectura determinada para el desarrollo del ERP Cuba[66].

3.4.1 Estándares de nomenclatura

- ✓ **Nomenclatura de las clases desarrolladas en la implementación**

Se determina que los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*, lo que posibilita que al leer el nombre de la clase se reconozca el propósito de la misma.[66]

Ejemplo: GestionarSistema.

- ✓ **Nomenclatura según el tipo de clases**

1. Clase controladora

Las clases controladoras después del nombre llevan la palabra: “Controller”.[66]

Ejemplo: GestSistemaController.

- ✓ **Nomenclatura de las funciones**

26 Fuente: Elaboración propia

Al asignar un nombre a alguna funcionalidad se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing*, y con solo leerlo se reconoce el propósito de la misma.[66]

Ejemplo: InsertarSistema.

- ✓ **Nomenclatura de las variables**

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing*, y se comenza con un prefijo según el tipo de datos.[66]

Ejemplo: arrSistemas.

- ✓ **Prefijos para los tipos de datos**

Los prefijos a utilizar en la creación de variables serán los siguientes:[66]

Tipos de Datos	Prefijos
Arreglos	arr
Objetos	obj

3.4.2 Normas de comentariado

Es una necesidad comentar todo lo que se haga dentro del desarrollo, es decir, establecer las pautas que conlleven a lograr un código más legible y reutilizable, de manera que se pueda aumentar su mantenibilidad a lo largo del tiempo.[66]

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

- ✓ **En las clases**

Antes de la declaración de una clase se escribe una breve descripción donde se explique el propósito de la misma. Que se escribe de la siguiente forma:[66]

```
/**
* Nombre de la clase *
* Descripcion *
* @author *
* @package *(módulo)
* @subpackage *(sub módulo)
* @copyright *
* @version (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. Se escribe de la siguiente forma:[66]

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

3.4.3 Estilo del código

En la implementación cuando se escriba una sentencia en PHP la forma de utilizar los tab del mismo es la siguiente:[66]

```
<?php
//código//
?>
```

- ✓ **Sangría o indexado**

La política de sangría a utilizar en la implementación es por tab.[66]

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

Figura 3.19 [27] Estilo del código: Sangría o indexado.

- ✓ **Brazas o llaves**

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

27 Fuente: Normas y estándares de codificación, Damián Pérez Alfonso, 2009

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

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

Figura 3.20 [28]Estilo del código: Brazas o llaves.

- ✓ **Espacios en blanco**

Arreglos

La declaración de los espacios en blanco en los arreglos es como se muestra en el ejemplo.

```php
<?php
list ( $a, $b ) = array (1, 2, 3 );
$array = array (1 => 2, 2 => 3 );
$array [$i]->foo ();
$array [] = 'first cell';
?>
```

Figura 3.21[29] Estilo de código: Espacio en blanco en Arreglos.

3.5 Pruebas de Software

Las pruebas de software se definen como el proceso orientado a demostrar que un programa informático realiza las funcionalidades para las cuales fue construido. Tiene como objetivo principal evaluar la calidad de un software, además de descubrir errores y medir el grado en que el software cumple con los requerimientos definidos.[67]

3.5.1 Pruebas de Caja Negra

28 Fuente: Normas y estándares de codificación, Damián Pérez Alfonso, 2009

29 Fuente: Normas y estándares de codificación, Damián Pérez Alfonso, 2009

Las pruebas de caja negra son las que se aplican a la interfaz del software y se centran en los requisitos funcionales del sistema; permitiendo derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa.[68]

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están[68]:

- ✓ Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- ✓ Técnica del Análisis de Valores Límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

El método utilizado para realizar pruebas al componente Autorización será caja negra mediante la técnica de particiones de equivalencia ya que es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software y descubre de forma inmediata errores en las funcionalidades. Al aplicar esta técnica se definen diseños de casos de pruebas que especifican el proceso de autorización a nivel de negocio y a nivel de datos.

El componente Autorización implementado es capaz de evaluar la petición de acceso sobre cualquier recurso que el usuario solicite. Con el objetivo de describir, documentar y evaluar el proceso se utilizará como referencia la funcionalidad Gestionar usuario del módulo Configurar usuario ubicado en el componente de Seguridad del marco de trabajo SAUXE.

La Figura 3.22[30] muestra la interfaz diseñada en el sistema ACAXIA que especifica la funcionalidad Gestionar Usuario.

30 Fuente: Interfaz de Usuario correspondiente al portal del marco de trabajo SAUXE.

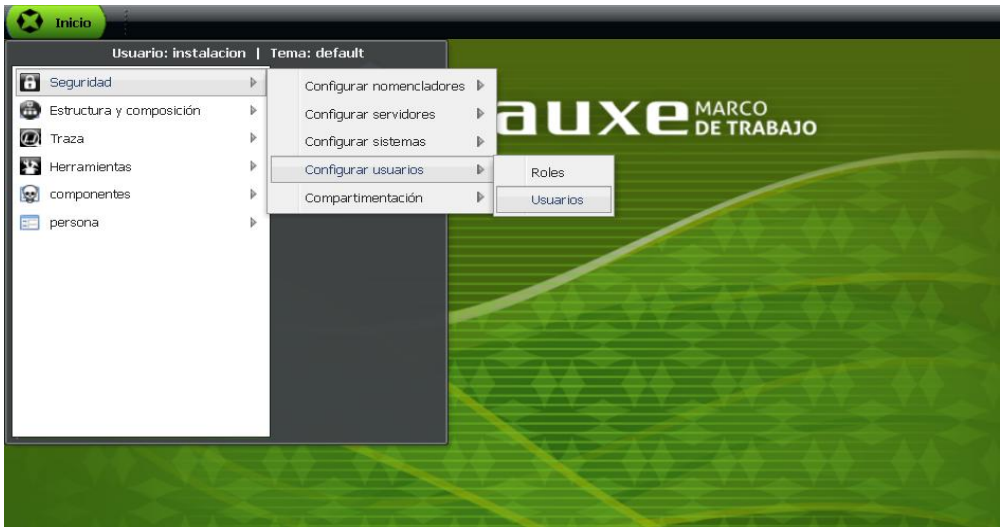


Figura 3.22 Interfaz de usuario: Funcionalidad Gestionar Usuario.

El proceso de autorización está distribuido en dos sistemas. Para realizar la validación del proceso es necesario realizar algunas configuraciones previas en el sistema ACAXIA para comprobar mediante los modelos de diseños de casos de pruebas si se obtiene una evaluación correcta del componente implementado. A continuación se describen las configuraciones previas en el sistema ACAXIA y se muestran los diseños de casos de pruebas para realizar las pruebas previstas que validan la eficiencia y calidad de las funcionalidades a cumplir del componente de autorización.

- ✓ **Proceso de autorización a nivel de negocio**

Estos diseños están enfocados a describir el proceso de autorización cuando existen peticiones de acceso a recursos que no dispongan ni requieran información de la base de datos del sistema.

- ✓ **Configuraciones previas en el sistema ACAXIA**

Las configuraciones a nivel de negocio están enfocadas a establecer privilegios sobre los recursos mediante la regulación de acciones que tiene asociada un rol determinado. De forma tal que los usuarios acceden a las funcionalidades que le son permitidas por el rol asignado.

En el sistema ACAXIA se utiliza la funcionalidad **Gestionar roles** para adicionar, modificar y regular acciones a los roles que le son asignados a los usuarios teniendo en cuenta la entidad a la que pertenecen. Se expone una imagen que específica la funcionalidad Gestionar roles correspondiente al módulo Configurar Usuarios.

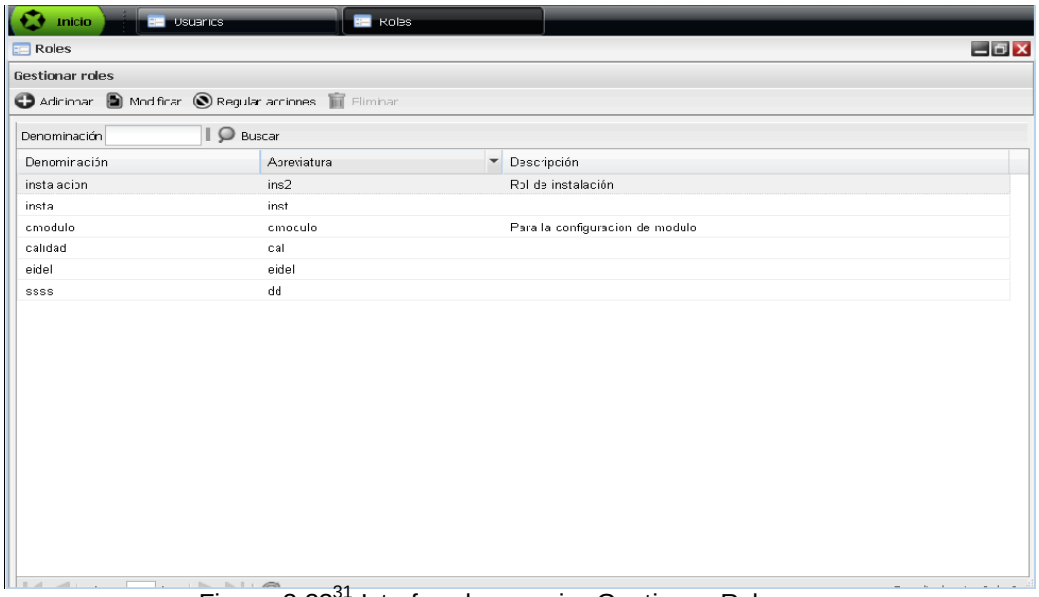


Figura 3.23[31] Interfaz de usuario: Gestionar Roles.

Para una mejor compresión, se específica el procedimiento realizado para gestionar las configuraciones en el sistema ACAXIA, estas acciones serán desglosadas por pasos:

1. **Escoger rol y regular acción**

Se busca el rol al cual se le quiere modificar las acciones y se selecciona ejecutando un clic sobre el mismo.

Se presiona el botón **Regular acciones** que aparecen en la barra de operaciones.

El sistema muestra la interfaz de usuario **Regular acciones** como se ilustra en la Figura 3.24[32] en la que se especifican los subsistemas y las funcionalidades a las que tiene accesos el rol escogido, además las acciones que le son autorizadas y en las que no posee el acceso.

31 Fuente: Interfaz de Usuario correspondiente a la funcionalidad Gestionar roles del marco de trabajo SAUXE.

32 Fuente: Interfaz de Usuario correspondiente a la acción Regular acciones del marco de trabajo SAUXE.

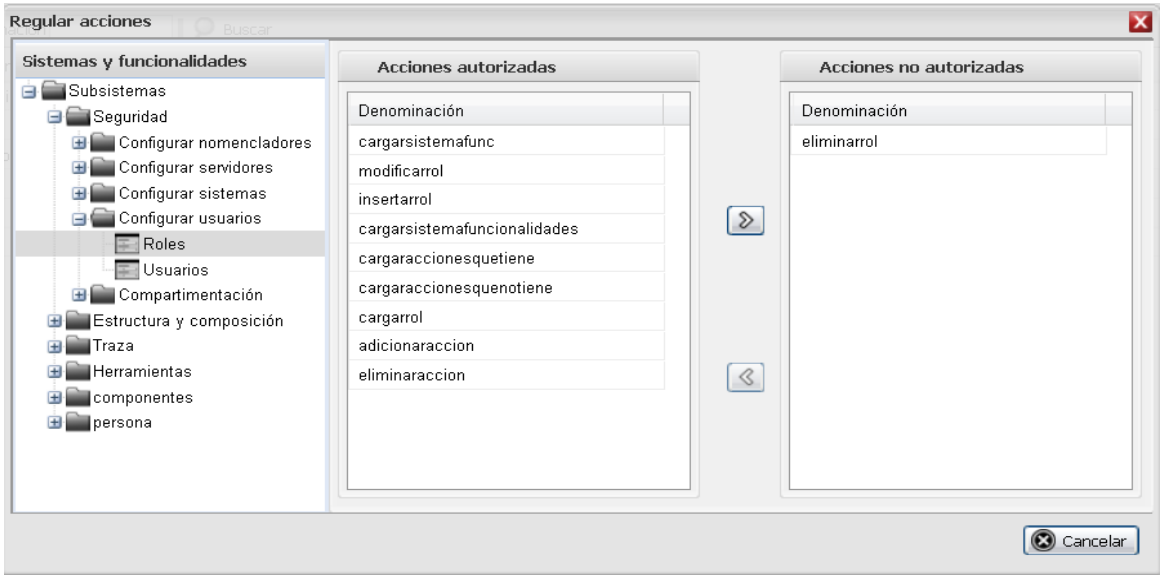


Figura 3.24 Interfaz de usuario: Regular Acciones.

2. Modificar acciones asociadas a un rol seleccionado.

Una vez el sistema muestre la interfaz Regular acciones se escoge la acción o acciones que desee regular, es decir la acción o acciones que permita autorizar y/o a las que no.

3. Resultado

Al concluir los pasos descritos anteriormente se obtiene como resultado la modificación de acciones a las que tenía acceso un rol determinado sobre una funcionalidad.

✓ **Proceso de autorización en el sistema SAUXE.**

Se modela un diseño de casos de prueba para validar el funcionamiento del proceso de autorización teniendo en cuenta las configuraciones en el sistema ACAXIA antes mencionadas.

Es importante destacar que estas pruebas se realizaran al **sistema externo SAUXE** y como interfaz de prueba **Gestionar usuarios** correspondiente al módulo Configurar usuarios mostrada en la Figura 3.25[33] la cual permite adicionar, modificar, eliminar, asignar roles, entre otras acciones a realizar sobre los usuarios registrados en la base de datos del sistema.

[33] Fuente: Interfaz de Usuario correspondiente a la funcionalidad Gestionar usuarios del marco de trabajo SAUXE.

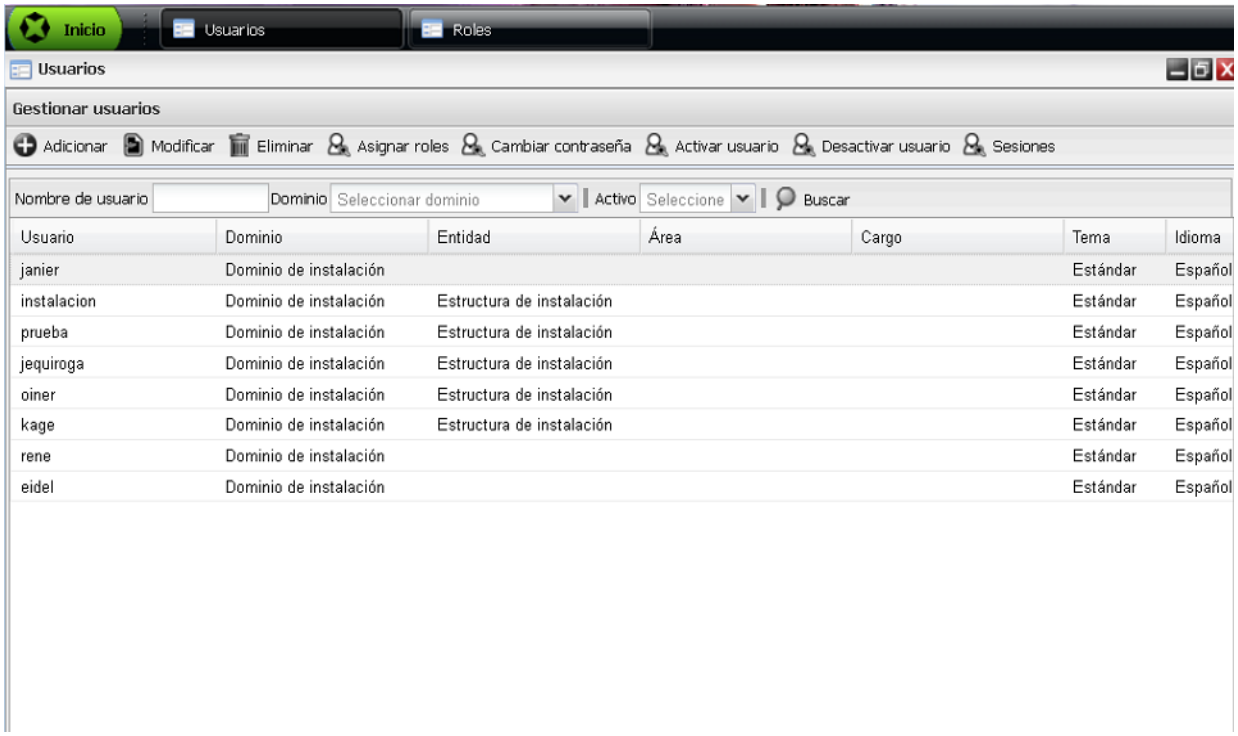


Figura 3.25 Interfaz de usuario: Gestionar usuarios

- **Diseño de Casos de prueba a nivel de negocio.**

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar autorización de usuario a nivel de negocio.	Determina si un usuario ya identificado y autenticado en el sistema puede acceder a un recurso.	EP 1.1: Petición de acceso denegada	✓ El usuario selecciona una acción (Adicionar, Modificar, Eliminar, Asignar Roles, Cambiar contraseña, etc.)de la barra de operaciones. ✓ El sistema verifica si el usuario tiene acceso sobre la acción selecciona. ✓ Se muestra un mensaje de error donde se deniega el acceso al pedido
		EP 1.2: Petición de acceso permitida	✓ El usuario selecciona una acción (Adicionar, Modificar, Eliminar, Asignar Roles, Cambiar contraseña, etc.) de la barra de operaciones. ✓ El sistema verifica si el usuario tiene acceso sobre la acción selecciona. ✓ El sistema permite al usuario ejecutar la acción selecciona.

	EP 1.3: Error en la validación interna del sistema.	✓ El usuario selecciona una acción (Adicionar, Modificar, Eliminar, Asignar Roles, Cambiar contraseña, etc.) de la barra de operaciones. ✓ El sistema verifica si el usuario tiene acceso sobre la acción selecciona. ✓ En la verificación interna se establece un error en la conexión con el PDP de XACML ✓ Se muestra un mensaje de error: "Error en la conexión con el PDP de XACML para cargar las funcionalidades."

- ✓ **Proceso de autorización a nivel de datos.**

Estos diseños están enfocados a describir el proceso de autorización cuando existen peticiones de acceso a recursos que requieran información de la base de datos del sistema.

- ✓ **Configuraciones previas en el sistema ACAXIA.**

Las configuraciones a nivel de datos están enfocadas en controlar el acceso a la información teniendo en cuenta las reglas contenidas en las obligaciones sobre los recursos, con el objetivo de evitar que un usuario manipule información perteneciente a otro usuario con un mismo rol en una misma entidad.

En el sistema ACAXIA se utiliza la funcionalidad **Compartimentar información** correspondiente al módulo de Compartimentación ubicado en el componente de Seguridad del marco de trabajo SAUXE. Esta es utilizada para adicionar, modificar, eliminar, activar y desactivar reglas a usuarios que comparten un mismo rol en una entidad con el fin de manejar la misma información indistintamente de sus funciones. Se expone una imagen que específica la funcionalidad Compartimentación de la información.

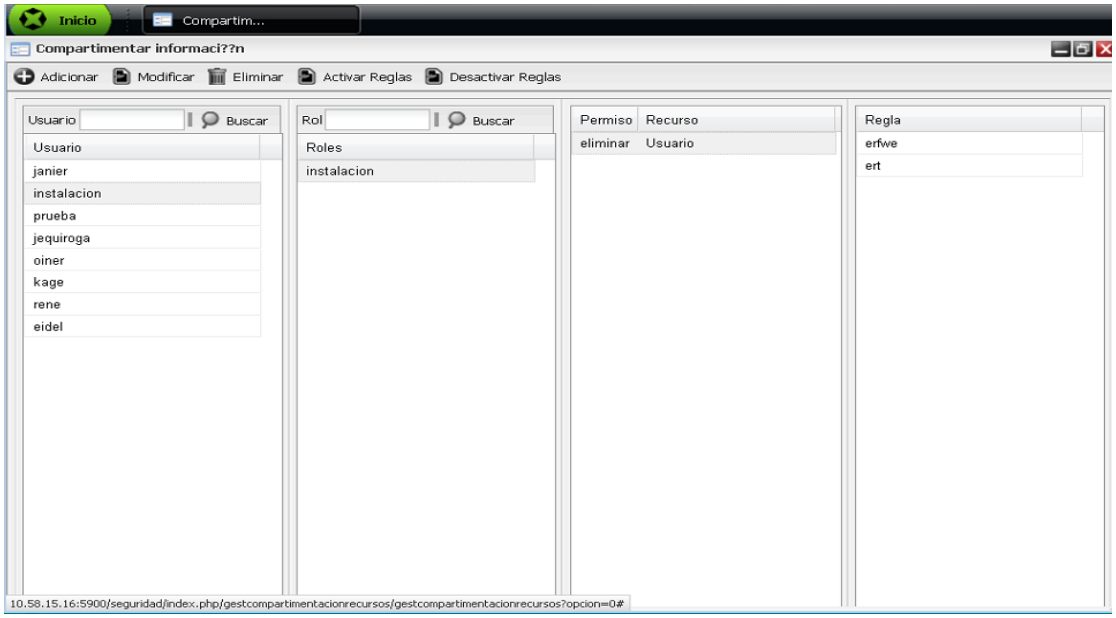


Figura 3.26 [34] Interfaz de usuario: Compartimentar Información.

Para una mejor compresión, se específica el procedimiento realizado para gestionar las configuraciones en el sistema ACAXIA, estas acciones serán desglosadas por pasos:

1. **Escoger usuario y seleccionar entidad**

Se busca el usuario al cual se le desea establecer nuevas obligaciones o permisos y se selecciona ejecutando un clic sobre el mismo.

El sistema muestra el rol que tiene asignado el usuario seleccionado y permite que el usuario (administrador del sistema ACAXIA) escoja la entidad asociada a dicho rol ejecutando un clic sobre el mismo. Se evidencia en la Figura 3.27[35] lo anteriormente mencionado.

34 Fuente: Interfaz de Usuario correspondiente a la funcionalidad Compartimentar información del marco de trabajo SAUXE.

35 Fuente: Interfaz de Usuario correspondiente a la funcionalidad Compartimentar información y Selección de entidad del marco de trabajo SAUXE.

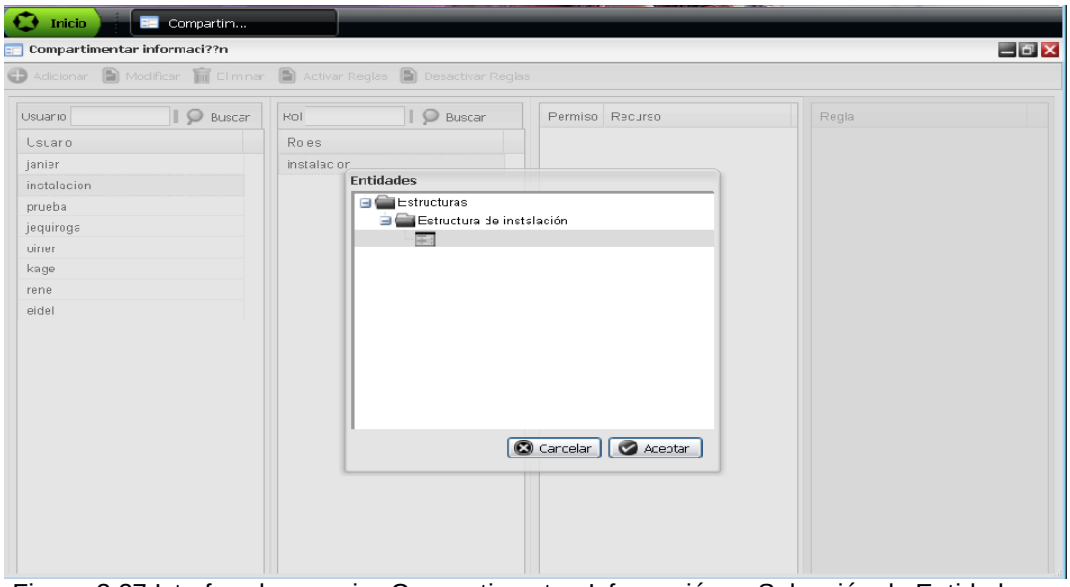


Figura 3.27 Interfaz de usuario: Compartimentar Información y Selección de Entidad.

2. **Adicionar nueva obligación o permiso a un rol en una entidad perteneciente al usuario seleccionado.**

Se selecciona la acción Adicionar ubicada en la barra de operaciones.

Una vez que el sistema muestre la interfaz **Nuevo permiso**, mostrada en la Figura 3.28[36], se especifica el tipo de operación que se desea restringir, la información o recurso sobre el cual se realizará la operación y permite definir reglas sobre el acceso.

El sistema inserta el nuevo premiso con las reglas asociadas.

36 Fuente: Interfaz de Usuario correspondiente a la acción Nuevo Permiso del marco de trabajo SAUXE.

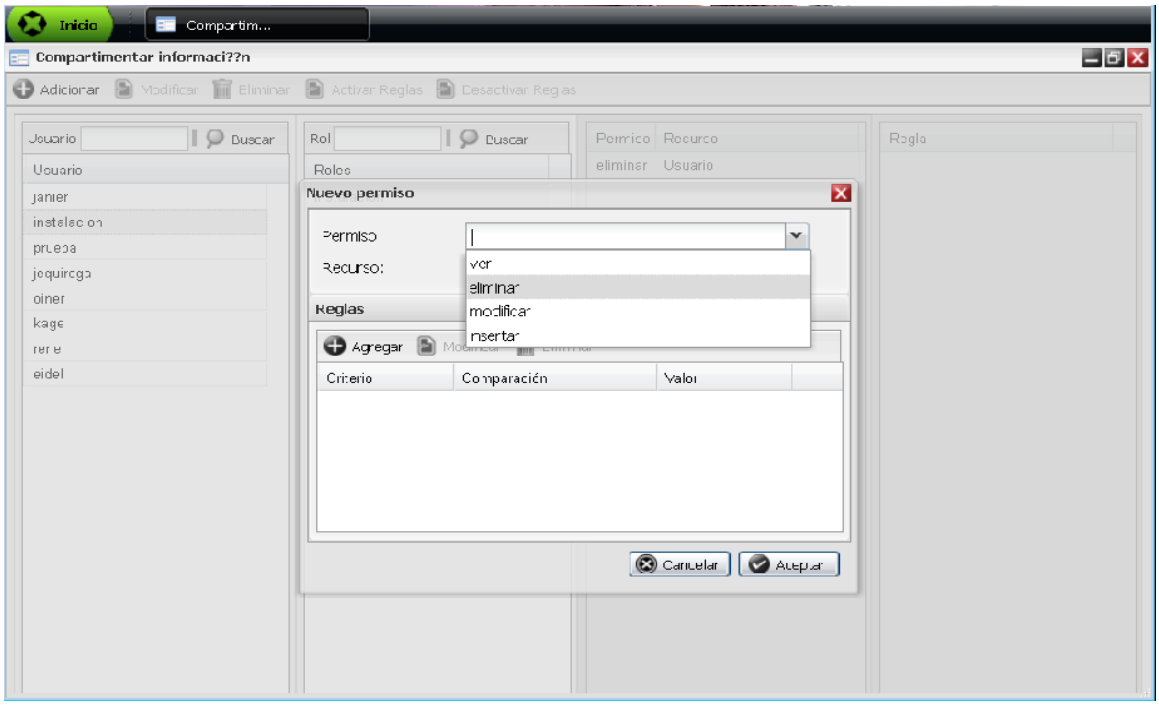


Figura 3.28 Interfaz de usuario: Nuevo Permiso

3. Resultado

Al concluir los pasos descritos anteriormente se obtiene como resultado, la adición de una obligación para los usuarios que comparten un mismo rol en una entidad con el objetivo que dominen la misma información.

✓ **Proceso de autorización en el sistema SAUXE**

Posteriormente se modela un diseño de casos de prueba para validar el funcionamiento del proceso de autorización teniendo en cuenta las configuraciones en el sistema ACAXIA antes mencionadas.

Estas pruebas se realizarán al sistema externo SAUXE y como interfaz de prueba Gestionar usuarios mostrada en la Figura 3.25.

➢ **Diseño de Casos de prueba a nivel de datos.**

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar autorización de usuario a nivel de datos.	Determina si un usuario puede acceder a un recurso teniendo en cuenta las restricciones correspondientes al rol asignado en una entidad.	EP 1.1: Permisos restringidos sobre las operaciones	✓ El usuario selecciona una acción (Adicionar, Modificar o Eliminar) de la barra de operaciones. ✓ El sistema verifica si el usuario tiene acceso sobre la acción selecciona. ✓ Se muestra un mensaje de error

		donde se específica que el usuario no tiene privilegios suficientes para ejecutar la acción sobre el objeto.
	EP 1.2: No existen permisos restringidos sobre las operaciones	✓ El usuario selecciona una acción (Adicionar, Modificar o Eliminar) de la barra de operaciones. ✓ El sistema verifica si el usuario tiene acceso sobre la acción selecciona. ✓ El sistema permite al usuario ejecutar la acción selecciona.
	EP 1.3: Permiso restringido sobre la operación (Ver)	✓ El sistema no muestra el listado de usuarios.
	EP 1.4:No existe permiso restringido sobre la operación (Ver)	✓ El sistema muestra el listado de usuarios.

3.5.2 Resultados Obtenidos

- ✓ Dichos casos de pruebas fueron evaluados y aprobados por el equipo de trabajo del departamento de Tecnología encargado de revisar y liberar el componente para ser usado como parte de ACAXIA (Anexo I).Para la aprobación y liberación del mismo, se efectuó una sola iteración de prueba, obteniendo en la misma 0 (cero) no conformidades.
- ✓ Se sugirió que durante el proceso de intercambio de mensajes entre software se utilizará Hypertext Transfer Protocol Secure (HTTPS) que es un protocolo seguro orientado a transacciones y sigue el esquema petición-respuesta; este a su vez incluye el protocolo criptográfico Secure Sockets Layer(SSL) que proporciona comunicaciones seguras por una red.

3.6 Validación de la solución como respuesta al problema científico de la investigación

La validación puede definirse de muchas formas, pero una simple definición es que la validación se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente. Es indispensable realizar un proceso de validación riguroso para la solución propuesta, donde la variable cuestionada es la **capacidad de integración** del sistema ACAXIA con otros sistemas. Actualmente la integración es un proceso que está cobrando cada vez mayor importancia ya que implica integrar componentes o módulos de uno o varios sistemas para que interactúen entre sí.

3.6.1 Pruebas de Integración

Se realizaron pruebas de integración al software con el propósito de identificar errores obtenidos de la ejecución de los requerimientos funcionales implementados, para verificar sí las especificaciones de diseños son alcanzadas y determinar sí ACAXIA presenta un alto nivel de fortaleza y rendimiento en el control de acceso, específicamente en el proceso de autorización.

Situación antes de la estandarización del componente Autorización basado en XACML

Antes de la implementación del componente Autorización ACAXIA no incluía una estructura estándar para facilitar el intercambio de mensajes con otros sistemas, lo que provocaba que dicho sistema realizara de forma insuficiente el cumplimiento de sus objetivos estratégicos. Estos están valorados en garantizar la gestión centralizada de la seguridad en un entorno de varias aplicaciones y en posibilitar adaptarse a sistemas legados que incluyan una arquitectura basada en el estándar XACML con el propósito del de aumentar la capacidad de integración con otros sistemas.

Situación después de la estandarización del componente Autorización basado en XACML

Una vez instaurada la nueva solución de autorización propuesta en la presente investigación en el sistema ACAXIA se facilita su integración con otros software. Se determinó realizar pruebas de integración entre los sistemas SAUXE y ACAXIA con la intención de diagnosticar la efectividad y medir el nivel de fortaleza de la seguridad que brinda la solución implementada.

- ✓ **Pruebas de Integración entre los software SAUXE y ACAXIA**

A continuación se describe la solución de autorización desarrollada en dos sistemas distintos.

- **Solución de autorización desarrollada en el sistema SAUXE**

SAUXE representa el dispositivo o sistema externo que realiza la petición de acceso hacia un recurso. Se específica en el mismo la implementación del subsistema PEP el cual es responsable de derivar el pedido de autorización al subsistema PDP en un formato XML. La Figura 3.29[37] ilustra la petición de autorización en formato XML realizada por este sistema.

37 Fuente: Elaboración propia

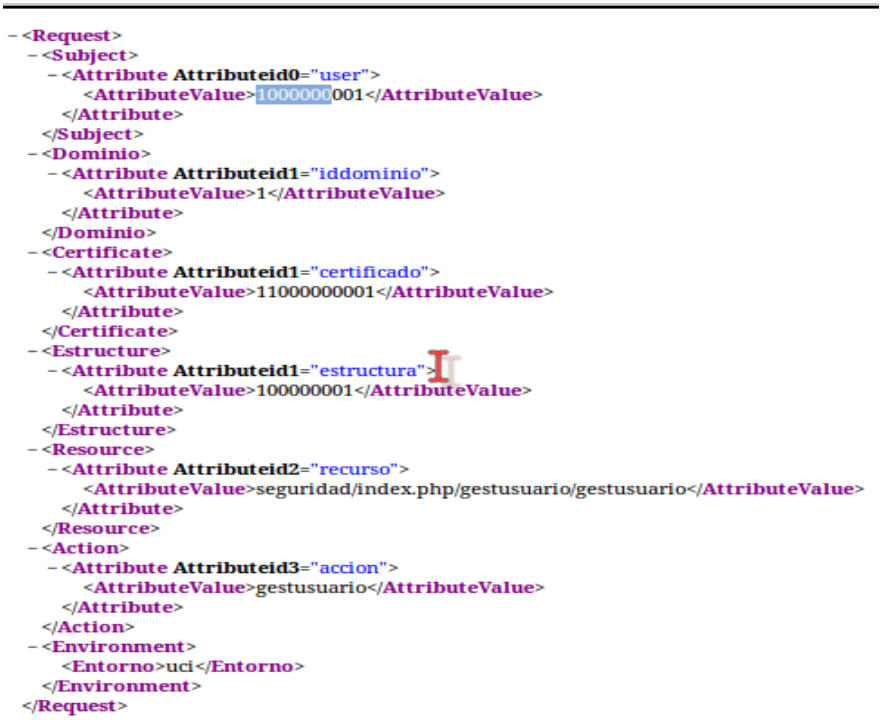


Figura 3.29 Petición de autorización en formato XML

El PEP, una vez ya obtenida la evaluación del subsistema PDP, elabora una repuesta para informar al sistema que hizo el pedido de autorización.

SAUXE permite, en caso que la respuesta de acceso sea aceptada, acceder al recurso solicitado con sus correspondientes obligaciones establecidas sobre el mismo. En caso contrario el sistema muestra un mensaje denegando el recurso: “No tiene los privilegios suficientes para acceder a este objeto. Pónganse en contacto con el administrador del sistema”.

- **Solución de autorización desarrollada en el Sistema ACAXIA**

ACAXIA representa el sistema encargado de validar la petición de acceso, para lo cual se estableció la implementación del subsistema PDP el cual es responsable de evaluar un pedido de autorización a partir de las políticas definidas y administradas en el subsistema PAP. El PDP elabora una respuesta de acceso en formato XML posteriormente enviado al PEP, en la Figura 3.30[38] se puede evidenciar la estructura de esta respuesta.

38 Fuente: Elaboración propia

ACAXIA cuenta además con el subsistema PIP que busca e interpreta información de las base de datos PostgreSQL utilizado para almacenar y gestionar la información. Además se específica la utilización del gestor de base datos MongoDB como sistema de base de datos multiplataforma orientado a documentos, el cual se utiliza para almacenar las obligaciones que tiene un usuario asignándole un rol en una entidad.

Permit

```
- <Response>
  - <Result Resourceid="seguridad/index.php/gestusuario/gestusuario">
      <Decision>permit</Decision>
    </Result>
  - <Functionalities Functionalitiesid="idfunc">
    - <Functionality>
        {"seguridad\/index.php\/gestnomdominio\/gestnomdominio": ["cargaracciones",
        modificarnomdominio","eliminarnomdominio","cargardominios","modificarclave",
        \/configcont":["cargaracciones","cargaretiquetas","closeportal","cargarclaves",
      </Functionality>
    </Functionalities>
  - <Obligations Obligationsid="idreglas">
    - <Obligation>
        [{"permiso":"eliminar","recurso":"Usuario","nombreregla":"regla1","campo":"identidad"
        ,"operador":"=","valor":"100000001","table_name":"seg_usuario"},
        {"permiso":"eliminar","recurso":"Usuario","nombreregla":"regla2","campo":"iddominio",
        "operador":">","valor":"1","table_name":"seg_usuario"}]
      </Obligation>
    </Obligations>
</Response>
```

Deny

```
- <Response>
  - <Result Resourceid="seguridad/index.php/gestusuario/gestusuario">
      <Decision>deny</Decision>
    </Result>
  </Response>
```

Figura 3.30 Respuesta de acceso en formato XML.

El intercambio de mensajes caracterizado por el esquema petición-respuesta evidencia la posibilidad de integrar, fusionar y adaptar diferentes sistemas al sistema de seguridad ACAXIA.

- ✓ **Pruebas de Integración entre los software SEPA y ACAXIA**

Otra forma de demostrar el aumento en la capacidad de integración de ACAXIA con otros sistemas, independientemente de las características y diferencias propias de cada software, se puede evidenciar a continuación.

Se decidió desarrollar una aplicación programada sobre el lenguaje Java para un entorno de escritorio denominada **Sistema de Envío de Petición de Autorización (SEPA)** que posibilita el intercambio de mensajes de autorización en el formato establecido por el estándar XACML con el propósito de valorar la eficacia, usabilidad, rentabilidad, portabilidad y la capacidad de integración del sistema ACAXIA. Además

permite medir el nivel de seguridad y fortaleza del proceso de autorización implementado. Esta solución se convirtió en uno de los aportes prácticos de la presente investigación.

A continuación se presenta la Figura 3.31[39] que caracteriza la aplicación SEPA implementada con la finalidad de enviar peticiones de autorización que tributan a políticas de control de acceso.

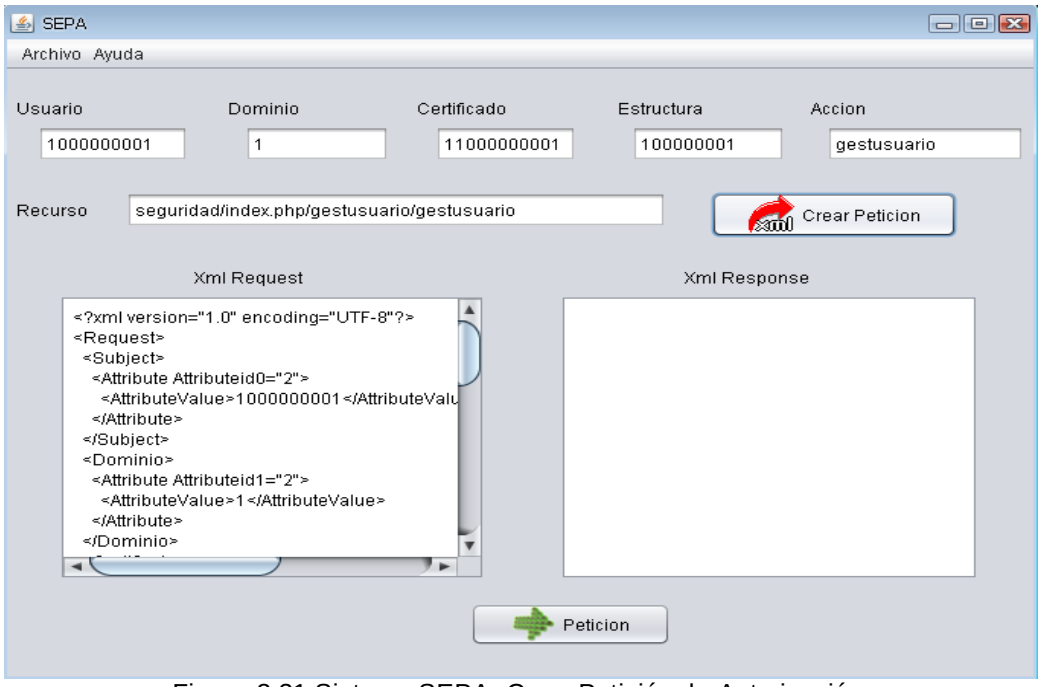


Figura 3.31 Sistema SEPA: Crear Petición de Autorización.

Esta solución de envío de peticiones permite insertar la información necesaria para acceder a un recurso. Además posibilita crear y manipular la estructura de la petición de acceso en formato XML. Para evaluar dicha petición se pulsa el botón **Petición** el cual permite establecer la comunicación con el sistema ACAXIA y a través del cual se reenvía la petición **xml Request.**

Una vez evaluada la petición, el sistema SEPA permite conocer la respuesta de acceso, las funcionalidades a las que el usuario puede acceder con sus correspondientes obligaciones a través del **xml Response** que es la respuesta enviada por el sistema ACAXIA en formato XML.

A continuación se evidencia cómo el sistema SEPA recibe el **xml Response** para cada uno de los casos (Permit o Deny).

Permit	Deny

39.Fuente: Elaboración propia

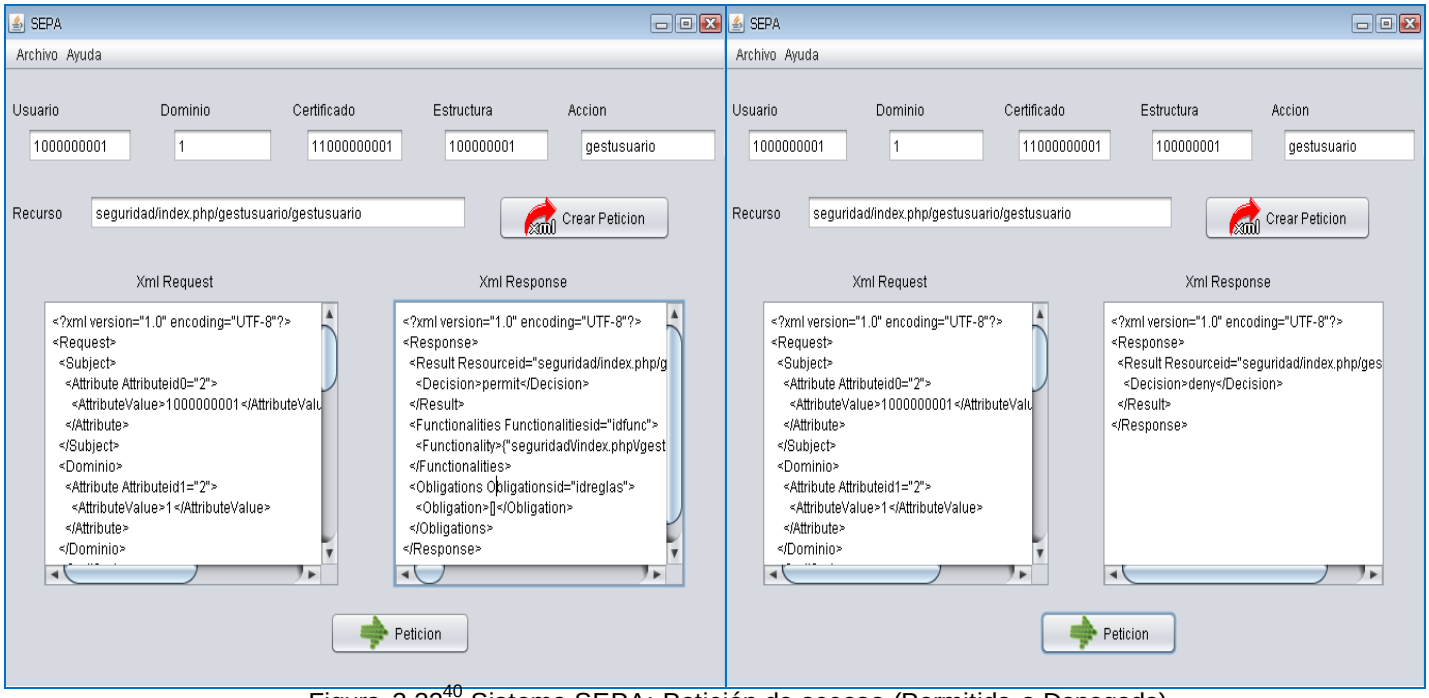


Figura 3.32[40] Sistema SEPA: Petición de acceso (Permitida o Denegada)

3.7 Resultados Obtenidos

La aplicación de la solución implementada en entornos reales arroja los siguientes resultados:

1. Se estima la importancia del tema tratado en la presente investigación y su contribución a la integración del sistema ACAXIA con otros sistemas independientemente de la tecnología sobre la que se encuentren desarrollados.
2. Se cumple el objetivo general de la investigación y se implementó con buenos resultados el estándar XACML en el sistema ACAXIA lo cual garantizó el aumento de su capacidad de integración.
3. En la implementación del componente Autorización se define una estructura de intercambio de mensajes de autorización y un modelo para organizar y almacenar la información. Estas contribuciones permiten mejorar la mantenibilidad del sistema, de forma tal que el procedimiento sea más organizado, fácil y sencillo.

40 Fuente: Elaboración propia

4. La prueba de integración entre los software ACAXIA y SAUXE evidencia el correcto funcionamiento de los requisitos funcionales desarrollados en el componente Autorización regidos por la estructura estándar de intercambio de mensajes que define XACML
5. La prueba de integración entre SEPA y ACAXIA demuestra que ACAXIA responde a peticiones de autorización, con la estructura estándar establecida, realizadas por sistemas desarrollados en Java garantizando así su integración con este tipo de tecnología.
6. ACAXIA presenta un alto nivel de fortaleza y complejidad en cuanto a la seguridad del sistema, pues éste implementa mecanismos de control de acceso que regulan y protegen el acceso a recursos y servicios.

3.8 Conclusiones Parciales

A modo de conclusión del presente capítulo se obtiene las siguientes reflexiones:

1. Se realizó el modelo de implementación y como parte de él se obtuvo el diagrama de componentes y el de despliegue de la solución.
2. Se documentaron los estándares de codificación por los que se rige la implementación del componente.
3. El desarrollo del componente Autorización basado en el estándar XACML para ACAXIA evidenció su contribución al fortalecimiento y rendimiento del control de acceso, específicamente en el proceso de autorización.
4. Se desarrolló el sistema SEPA, aplicación que tiene como objetivo el envío de peticiones de acceso. Este gestiona la evaluación de sus peticiones a través del componente Autorización desplegado en ACAXIA.
5. Para probar que el componente sí constituye una mejora para ACAXIA y verificar que éste se integre con otros sistemas, se exponen pruebas de integración entre los software SAUXE y ACAXIA y SEPA y ACAXIA.
6. Las pruebas de integración entre los diferentes sistemas arrojaron resultados satisfactorios y refleja el rendimiento y complejidad funcional del sistema ACAXIA.
7. Se resuelve el problema científico formulado en la presente investigación a través del desarrollo de un componente Autorización basado en XACML.

Conclusiones Generales

Los resultados obtenidos durante el desarrollo de la presente investigación permiten concluir lo siguiente:

1. En la bibliografía consultada no se detectó ninguna solución compatible con ACAXIA pues presentan características que limitan su rendimiento e integración.
2. Se desarrolló una solución de autorización para el sistema de seguridad ACAXIA basada en la implementación del estándar XACML.
3. La estructura definida por el estándar XACML para el intercambio de mensajes de autorización permite aumentar la capacidad de integración de ACAXIA con otros sistemas.
4. El proceso de validación demuestra el correcto funcionamiento de los requisitos funcionales implementados y el aumento de la capacidad de integración de ACAXIA con otros sistemas independientemente de las tecnología sobre la cuales estén desarrollados.

Recomendaciones

Procedentes del análisis y valoración del estudio realizado se considera significativo estimar las siguientes recomendaciones:

1. Identificar una solución criptográfica que no afecte el rendimiento del sistema.
2. Implementar en próximas versiones mecanismos que posibiliten al sistema externo el conocimiento de cambios realizados en las políticas de control de acceso en ACAXIA.

Referencias Bibliográficas

1. Oficial, G., *Decreto Ley 281. Principios de organización y funcionamiento del Sistema de Información del Gobierno*, in *10*, C. Ministerio de Justicia, Editor. 2011, Infodir. p. 29.
2. Almunawar, H.S.y.M.N., *Information Security Awareness: A Marketing Tools for Corporate's Business Processes*. 2012(3. SECURITY BREACHES AND BUSINESS PROCESSES' IMPACT): p. 5-7.
3. López, J.L.R., *Protección de la información*. 2003: España. p. 3-4.
4. Alexander Pretschner, T.M., Yves Le Traon. *Model-Based Tests for Access Control Policies*. in *International Conference on Software Testing, Verification, and Validation*. 2008.
5. Campuzano, V.H.D., *Control de acceso informático*, in *Facultad de contaducría y ciencias adminstrativas*. 2010, Universidad de Michoacana de San Nicolás de Hidalgo: Michoacán, México.
6. Moses, T., *eXtensible Access Control Markup Language (XACML) Version 2.0*, O.A.C. TC, Editor. 2005, OASIS: USA.
7. Ueda, E.T. and W.V. Ruggiero, *A Systematic Mapping on the Role-Permission Relationship in Role Based Access Control Models*. IEEE Latin America Transactions, 2012. **10**(1): p. 1243-1250.
8. Lei, Z., et al. *A Mandatory Access Control Model Based on Concept Lattice*. in *International Conference on Network Computing and Information Security*. 2011. Guilin: IEEE Computer Society.
9. Suhendra, V., *A Survey on Access Control Deployment*. Springer Berlin Heidelberg, 2011. **259**: p. 11-20.
10. MIC, *Resolución No. 127 /2007*, in *Oficina de Seguridad para las Redes Informáticas*, M.d.l.I.y.l.C.d.l.r.d. Cuba, Editor. 2007: La Habana, Cuba. p. 1-24. Disponible en: <http://ftur.uh.cu/intra/ftp/Resoluciones%20y%20Reglamentos/Resoluciones/R%20127-07%20Reglamento%20de%20Seguridad%20Informatica.pdf> .
11. Havighurst, R., *User Identification and Authentication Concepts*, L. Taylor & Francis Group, Editor. 2007. p. 1-64. Disponible en: <http://www.infosectoday.com/Articles/AU5219_C01.pdf> [Consultado 27/01/2013].
12. Esponda, S., *Sistema de Autorización para Web Services basado en XACML*, in *Facultad de Ingeniería y Tecnología Informática Carrera de Ingeniería Informática*. 2007, UNIVERSIDAD DE BELGRANO: Buenos Aires. p. 1-40.
13. Downs, D.D., et al. *Issues in Discretionary Access Control*. in *Symposium on Security and Privacy*. 1985. Oakland, USA: IEEE Computer Society.
14. Vasilyevna, N.B. *An RBAC design with Discretionary and Mandatory Features*. in *International Symposium on Ubiquitous Multimedia Computing*. 2008. Hobart, ACT: IEEE Computer Society.
15. Kim, D.-K., P. Mehta, and P. Gokhale. *Describing Access Control Models as Design Patterns Using Roles*. in *Conference on Pattern languages of programs*. 2006. New York, USA: ACM Digital Library.
16. Robles, R.J., et al. *Application of Role-Based Access Control for Web Environment*. in *International Symposium on Ubiquitous Multimedia Computing*. 2008. Hobart, ACT: IEEE Computer Society.
17. Sandhu, R.S., *Lattice-based access control models*. IEEE Computer Society, 1993. **26**(11): p. 9–19.
18. Argemí, H.I.N.J.A.M. *RBAC: Alternativa Actual para la Realización de Control de Accesos a Gran Escala*. 2003.
19. Pinagapani, S., D. Xu, and J. Kong. *A Comparative Study of Access Control Languages*. in *International Conference on Secure Software Integration and Reliability Improvement*. 2009. Shanghai: IEEE Computer Society.
20. Karjoth, G., A. Schade, and E.V. Herreweghen. *Implementing ACL-based Policies in XACML*. in *Annual Computer Security Applications Conference*. 2008. Anaheim, CA IEEE Computer Society.

21. Tao, W., L. Wei-hua, and L. Zun. *RBAC Permission Consistency Static Analysis Framework*. in *International Conference on Multimedia Information Networking and Security*. 2010. Nanjing, Jiangsu: IEEE Computer Society.
22. Abdallah, A.E. and H. Takabi. *Integrating Delegation with the Formal Core RBAC Model*. in *The Fourth International Conference on Information Assurance and Security*. 2008. Naples: IEEE Computer Society.
23. Crampton, J. and H. Khambhammettu. *A Framework for Enforcing Constrained RBAC Policies*. in *International Conference on Computational Science and Engineering*. 2009. Vancouver, BC: IEEE Computer Society.
24. Ferraiolo, D.F. and D.R. Kuhn. *Role-Based Access Controls*. in *15th National Computer Security Conference*. 1992. Baltimore MD: National Institute of Standards and Technology.
25. Cheng, D. and H. Wen. *The application of role based access control in the third party logistics information system*. in *International Conference on Information Management, Innovation Management and Industrial Engineering*. 2011. Shenzhen, China: ACM Digital Library.
26. Ma, L., S. Ma, and J. Lv. *A Dynamic Description Logic-based Formalism for RBAC*. in *Fourth International Conference on Computer Sciences and Convergence Information Technology*. 2009. Seoul: IEEE Computer Society.
27. Kuhn, D.R., E.J. Coyne, and T.R. Weil, *Adding Attributes to Role-Based Access Control*. IEEE Computer Society, 2010: p. 79-81.
28. Luo, J. and S. Xu. *Based on RBAC Security Design of University Management System*. in *International Conference on E-Business and E-Government*. 2010. Guangzhou: IEEE Computer Society.
29. Xuexiong, Y., W. Qinxian, and X. Changzheng. *A Multiple Hierarchies RBAC Model*. in *International Conference on Communications and Mobile Computing*. 2010. Shenzhen: IEEE Computer Society.
30. Anderson, A. (2005) *XACML References, 1.14*. **1.4**, 1-16.
31. Ardagna, C.A., et al. *An XACML-Based Privacy-Centered Access Control System*. in *Proceedings of the first ACM workshop on Information security governance*. 2009. Chicago, USA: ACM Digital Library.
32. Jeong, J., et al. *Integration of Single Sign-On and Role-Based Access Control Profiles for Grid Computing*. in *Frontiers of WWW Research and Development*. 2006. Harbin, China: Springer Berlin Heidelberg.
33. Pages, C. (2010) *Cover Pages*.
34. Kataoka, T., et al. *Leveraging PKI in SAML 2.0 Federation for Enhanced Discovery Service*. in *Ninth Annual International Symposium on Applications and the Internet*. 2009. Bellevue, WA: IEEE Computer Society.
35. Kaixing, W. and Y. Xiaolin. *A Model of Unite-Authentication Single Sign-On Based on SAML underlying Web*. in *Second International Conference on Information and Computing Science*. 2009. Manchester, England
36. Kohlar, F., et al. *Secure Bindings of SAML Assertions to TLS Sessions*. in *International Conference on Availability, Reliability and Security*. 2010. Krakow: IEEE Computer Society.
37. Verma, M. (2004) *XML Security: Control information access with XACML*.
38. HERAS-AF. *HERAS-AF XACML*. 2010; Available from: http://www.herasaf.org/.
39. openLiberty, C. *OpenAz Main Page*. 2012; Available from: http://www.openliberty.org/wiki/index.php/OpenAz_Main_Page.
40. SourceForge.net. *Sun's XACML Implementation*. 2006; Available from: http://sunxacml.sourceforge.net/.
41. Bustos, P.G., *Guía de Uso de la Herramienta CASE*, E.d.I. Industrial, Editor. 2010.
42. Foundation., A.S. *Apache,http server project*. 2012; Available from: http://httpd.apache.org/bug_report.html.
43. jdrake@postgresql.org, J.D.D.-. *Postgres SQL* 1996; Copyright © 1996-2013 The PostgreSQL Global Development Group [Available from: http://www.postgresql.org/docs/8.3/static/release-8-3-8.html.
44. 10gen. *mongoDB*. 2013; Copyright © 2013 10gen, Inc.:[Available from: http://www.mongodb.org.

45. Tigris.org, O.S.S.E.T. *SUBVERSION*. 2001; 2001 - 2009 CollabNet. CollabNet is a registered trademark of CollabNet, Inc.:[Available from: http://subversion.tigris.org/.
46. Pupo, Y.C., *Libro de Ayuda del Marco de Trabajo Sauxe.*, in *Facultad 15*. 2010, Universidad de las Ciencias Informáticas: La Habana. p. 72.
47. framework, Z. *Zend framework*. 2006; 2006 - 2013 by Zend Technologies Ltd.:[Available from: http://framework.zend.com.
48. doctrine. *doctrine*. Available from: http://www.doctrine-project.org/.
49. español., E.N.c.e. *Ext,Comunidad*. 2011; Available from: http://www.extjs.mx/2011/11/post-con-cursos-generales/.
50. Apps, S.E.J.J.F.f.R.D. *Ext JS Framework* 2013 [cited 2013 4 de junio]; 2013 Sencha Inc:[Available from: http://www.sencha.com/products/extjs.
51. News, N.o.C. *NetBeans IDE*. 2013 [cited 2013 4 de junio]; Available from: https://netbeans.org/community/news/show/1519.html.
52. STAR UML, T.O.S.U.; Available from: http://staruml.sourceforge.net/en.
53. Group, T.P. *PHP 5.2.6 Release Announcement*. 2001; Copyright © 2001-2013 The PHP Group [Available from: http://php.net/releases/5_2_6.php.
54. BARYOLO, O.G., *CAEM: MODELO DE CONTROL DE ACCESO PARA SISTEMAS DE INFORMACIÓN EN ENTORNOS MULTIDOMINIOS*, in *CENTRO PARA LA INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES*. 2012, UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS: La Habana. p. 130.
55. Craig, L., *UML y Patrones. Introducción al Análisis y Diseño orientado a objetos*. 1999: Mexico : Prentice Hall.
56. Ian Sommerville , P.S., *Requirements Engineering: A Good Practice Guide*, I.N.Y. John Wiley & Sons, NY, USA ©1997, Editor. 1997: New York, NY, USA.
57. *Diseño y Programación Orientada a Objetos*. 2004/2005.
58. Dr. Francisco José García Peñalvo, M.Á.C.G., Sergio Bravo Martín, *Ingeniería del Software,Principios del diseño del software*, I.T.I.S., Editor. 2008, Universidad de Salamanca – Departamento de Informática y Automática.
59. Reynoso, C.B., *Introducción a la Arquitectura de Software*. 2004: Buenos Aires,Argentina.
60. Mena López, G.L.y.T.C., Marianela., *Arquitectura ERP*. 2008.
61. Gamma, E., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Editor. 1994.
62. SEL, S.E.L. *Diagrama de Secuencia*. 2013; Software Engineering Lab (SEL-UC3M):[
63. Sevilla, D.d.L.y.S.I.E.T.S.I.I.U.d., *Bases de Datos,Modelo de datos*, U. Sevilla, Editor. 2005: Espana.
64. S.A., S. *Sparx Systems Argentina* 2000-2013 [cited 2013 mayo]; 2000 - 2013 Sparx Systems Pty Ltd.:[Available from: http://www.sparxsystems.com.ar.
65. Calleja, M.A. (2001-2013) *Carmen. Estándares de codificación*. **1**, 13.
66. Alfonso, D.P., *Normas y estándares de codificación*, in *Departamento de Tecnologia de CEIGE*. 2008, Universidad de las Ciencias Informáticas: La Habana,Cuba.
67. Software, G.d.C.y.P.d. *http://pruebasdesoftware.com/*. pruebasdesoftware.com. Gestión de Calidad y Pruebas de Software 2005; Available from: http://pruebasdesoftware.com/laspruebasdesoftware.htm.
68. Pressman, R.S., *Ingenieria del Software, Un Enfoque Practico*. 2002, McGraw-Hill Companies,.

Glosario de términos

A.

Amenaza: Hecho que puede producir un daño.

Análisis de la información: Decodificación de datos contenidos en un documento, es ejecutado por un especialista en relación con las operaciones del procesamiento de la información para facilitar la recuperación y acceso a la misma.

Auditoría: Proceso de recoger, agrupar y evaluar evidencias para determinar si un Sistema de Información salvaguarda el activo empresarial, mantiene la integridad de los datos, lleva a cabo eficazmente los fines de la organización y utiliza eficientemente los recursos.

Autenticación: Es el acto de establecimiento o confirmación de algo (o alguien) como auténtico. La autenticación de un objeto puede significar (pensar) la confirmación de su procedencia, mientras que la autenticación de una persona a menudo consiste en verificar su identidad.

Autorización: Proceso de determinar si un usuario ya identificado y autenticado se le permite acceder a recursos de información de una manera específica.

B.

Brecha de seguridad: Vía por la que se puede atacar la seguridad de un sistema.

C.

Calidad del software: Características propias del software aquellas que se quieran controlar y asegurar.

Compartimentación de la información: División de la información sensible y de los elementos y recursos de un sistema de información en unidades menores; basándose en la necesidad de conocer, a fin de reducir el riesgo de accesos no autorizados.

Componente: Unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

Confidencialidad: Se refiere a que solo personal autorizado conocerá la información.

Control de acceso: Conjunto de políticas dentro de un sistema de gestión de identidad que definen las normas de todo lo que un usuario se le permite hacer en el ámbito de aplicación de dicho sistema.

Criterios: Son atributos que identifican únicamente a un recurso.

D.

Dominios: constituyen agrupaciones de organizaciones que tienen en común algún criterio.

E.

Entidad: Forma parte de la estructura de un país y puede comportarse como un ministerio, entidad, área o cargo.

Entorno: Conjunto de circunstancias, físicas y morales, que rodean a una persona o cosa.

Estilos arquitectónicos: Se consideran transformaciones impuestas al diseño de todo un sistema.

Estándar: Los estándares son acuerdos (normas) documentados que contienen especificaciones técnicas u otros criterios precisos para ser usados consistentemente como reglas, guías, o definiciones de características para asegurar que los materiales productos, procesos y servicios se ajusten a su propósito.

F.

Funcionalidad: Coherencia entre las necesidades detectadas y los resultados que se obtienen con el uso del material. Es lo que un producto puede hacer. Probar la funcionalidad significa asegurar que el producto funciona tal como estaba especificado.

H.

Herramienta: Es un objeto elaborado a fin de facilitar la realización de una tarea mecánica.

I.

Incidentes de seguridad: Cualquier hecho o evento que crees que podría afectar a tu seguridad personal o a la seguridad de tu organización.

Identificación: Identificación es la acción y efecto de identificar o identificarse. La identificación está vinculada a la identidad, que es el conjunto de los rasgos propios de un sujeto o de una comunidad.

Integridad: La información no será alterada, borrada, reordenada o copiada. Debe permanecer en su estado original, sin haber sido alterada por personal o de forma no autorizado.

L.

Lenguaje de Enmarcado de Aserciones de Seguridad: Permite a las entidades empresariales hacer afirmaciones sobre la identidad, atributos y derechos de un sujeto a otras entidades, tales como una compañía asociada u otra aplicación de empresa.

Lenguaje de Marcaje Extensible: Lenguaje de programación diseñado especialmente para los documentos de la web. Permite que los diseñadores creen sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos entre aplicaciones y entre organizaciones.

Lenguaje de Marcaje de Control de Acceso Extensible: Es un estándar para facilitar el intercambio de mensajes con otros sistemas y utilizar un lenguaje unificado y portable para expresar políticas de autorización.

Listas de Control de Acceso (ACLs): Concepto de seguridad informática usado para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido.

M.

Mecanismos de seguridad: Es una técnica que se utiliza para implementar un servicio, es decir, es aquel mecanismo que está diseñado para detectar, prevenir o recobrarse de un ataque de seguridad.

Modelos de control de acceso: Es un conjunto definido de criterios que un administrador del sistema utiliza para definir derechos/permisos de los usuarios del/al sistema.

0.

Obligación: Es un concepto que agrupa las operaciones que se pueden realizar sobre uno o varios recursos que cumplen con una o varias reglas. Ejemplo: mostrar las cuentas de una organización (X) que fueron registradas en una fecha (Y).

Operaciones: Son acciones que se realizan sobre uno o varios recursos, pueden iniciarse debido a la petición de un usuario o por configuraciones internas del SI. Ejemplo: adicionar, modificar, mostrar, eliminar, asentar, aprobar, entre otras.

P.

Paradigmas: Significa ejemplo o modelo. Puede indicar el concepto de esquema formal de organización, y ser utilizado como sinónimo de marco teórico o conjunto de teorías.

Permiso: tipo de autorización que puede tener un rol o usuario sobre determinado recurso.

Políticas de control de acceso: El tipo de política que define quién tiene acceso a un objeto, y que operaciones pueden efectuarse en relación con el mismo.

Procesos: Conjunto de actividades o eventos (coordinados u organizados) que se realizan o suceden (alternativa o simultáneamente) bajo ciertas circunstancias con un fin determinado.

Pruebas de software: Son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o stakeholder. Son una actividad más en el proceso de control de calidad.

Punto de Administración de Política: Crea y administra las políticas de control.

Punto de Control de Política: Deriva el pedido de autorización al PDP. Luego de obtener la respuesta del PDP elabora una repuesta para el sistema que hizo el pedido de autorización.

Punto de Decisión de Política: El subsistema Punto de Decisión de Política evalúa un pedido de autorización a partir de las políticas definidas.

Punto de Información de Política: El subsistema Punto de Información de Política es responsable de buscar e interpretar información de otras fuentes de datos y obtener el valor.

R.

Recurso: Cualquier elemento que se maneje a nivel de negocio y que esté representado en una tabla de la BD.

Regla: Especificación que se le agrega a los permisos.

Requisitos de software: Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

Rol: Agrupa una serie de permisos sobre sistemas, funcionalidades y acciones que se le asignarán a un conjunto de usuarios.

S.

Seguridad informática: Es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático.

Sesión: Mapea los privilegios que tiene un usuario en un espacio de tiempo, sobre los recursos de una o varias organizaciones debido a los roles que desempeña o por privilegios asignados directamente él.

Sistema: Es un conjunto organizado de objetos o partes interactuantes e interdependientes, que se relacionan formando un todo unitario y complejo.

Software: Se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica.

T.

Toma de decisiones: Es el proceso mediante el cual se realiza una elección entre las opciones o formas para resolver diferentes situaciones de la vida en diferentes contextos: a nivel laboral, familiar, empresarial.

Tecnología: Es el conjunto de conocimientos técnicos, ordenados científicamente, que permiten diseñar y crear bienes y servicios que facilitan la adaptación al medio ambiente y satisfacer tanto las necesidades esenciales como los deseos de la humanidad.

U.

Usuario: cualquier persona que interactúa con el sistema y juega un rol determinado en el mismo.

V.

Validación: Confirmación mediante el suministro de evidencia objetiva de que se han cumplido los requisitos para una utilización o aplicación específica prevista.

Vulnerabilidad: Las vulnerabilidades son puntos débiles del software que permiten que un atacante comprometa la integridad, disponibilidad o confidencialidad del mismo. Algunas de las vulnerabilidades más severas permiten que los atacantes ejecuten código arbitrario, denominadas vulnerabilidades de seguridad, en un sistema comprometido.

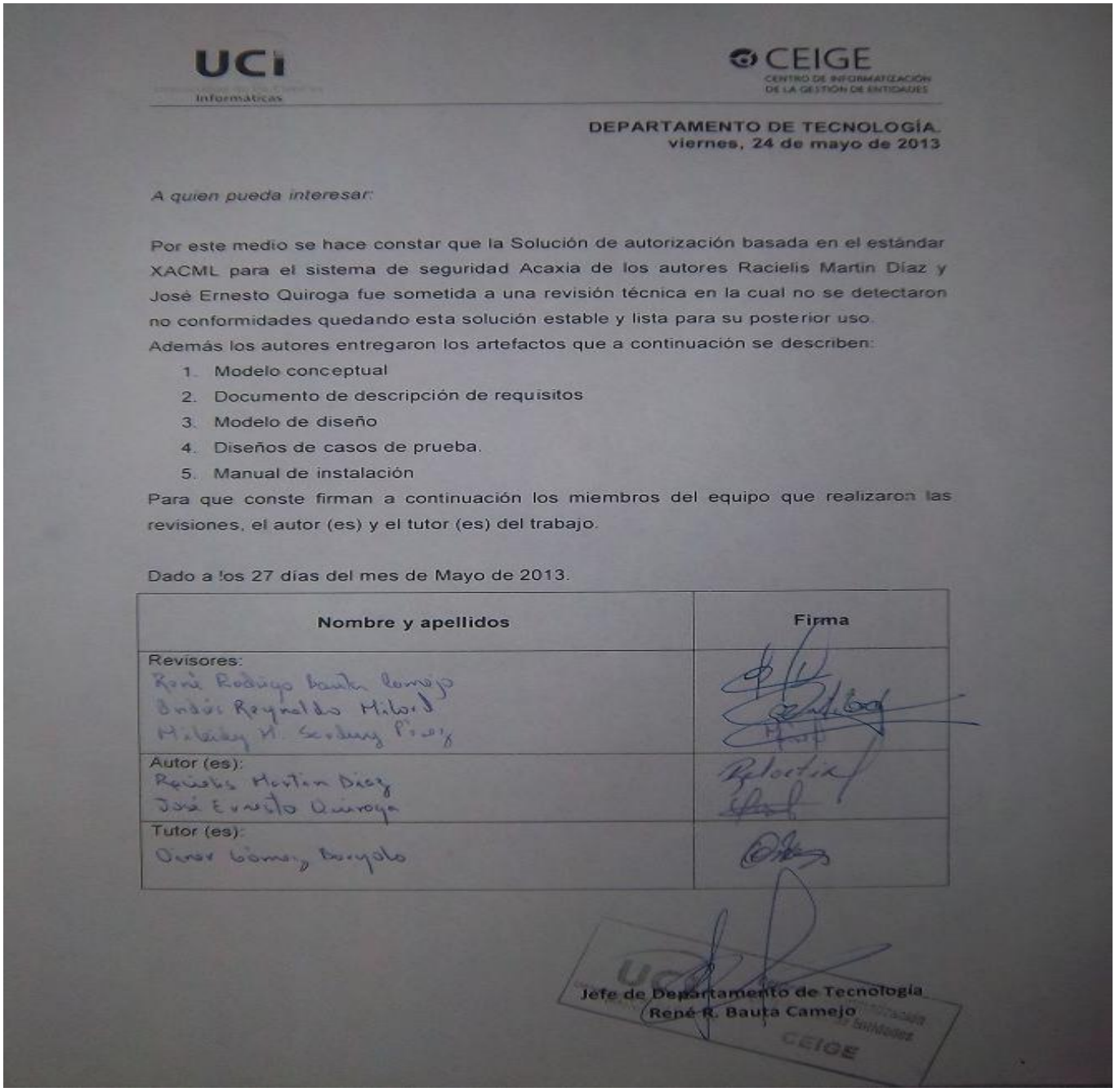
X.

XML XACML Request: Primer XML que se crea para almacenar la petición del usuario hacia un recurso.

XML XACML Response: XML que se crea en respuesta al XML XACML Request Authorize User.

Anexos

Anexo 1



UCI
Informáticas

CEIGE
CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES

DEPARTAMENTO DE TECNOLOGÍA.
viernes, 24 de mayo de 2013

A quien pueda interesar:

Por este medio se hace constar que la Solución de autorización basada en el estándar XACML para el sistema de seguridad Acaxia de los autores Racielis Martin Díaz y José Ernesto Quiroga fue sometida a una revisión técnica en la cual no se detectaron no conformidades quedando esta solución estable y lista para su posterior uso.
Además los autores entregaron los artefactos que a continuación se describen:

1. Modelo conceptual
2. Documento de descripción de requisitos
3. Modelo de diseño
4. Diseños de casos de prueba.
5. Manual de instalación

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

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

Nombre y apellidos	Firma
Revisores: René Rodrigo Bauta Camejo Andrés Reynaldo Milord Mileidy M. Serbeny Pino	
Autor (es): Racielis Martin Díaz José Ernesto Quiroga	
Tutor (es): Omar Gómez Barzaga	

Jefe de Departamento de Tecnología
René R. Bauta Camejo
CEIGE

Figura 7: Acta de liberación del Dpto. de Tecnología