Universidad de las Ciencias Informáticas

Facultad 15

Título: Sistema de Autenticación y Control de Acceso para aplicaciones del Departamento de Soluciones para la Aduana.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Adrian Naranjo Garcia

Tutor(es):

Ing. Daimara Mustelier Sanchidrian

Ing. Dennis Vázquez Prida

Ciudad de la Habana, junio del 2010.

Agradecimientos

Quiero agradecer a todas aquellas personas que han hecho posible que se realice este trabajo.

A mis tutores Dennis Vázquez y Daimara Mustelier, por dedicarme su tiempo y su paciencia, y ayudarme en todo lo que pudieron.

A Zénel Reyes por ayudarme en todo lo que me hizo falta y por ser mi amigo.

A mis compañeros de aula por aguantarme tanto tiempo.

A mis compañeros de trabajo en el proyecto por estar varias horas tratando de arreglar el mundo, y colaborar en el desarrollo de este trabajo.

A todos, muchas gracias.

Dedicatoria

A mi mamá y mi papá, Ana Iris y Julio, por darme la vida. Por apoyarme siempre y darme amor. Por enseñarme que aunque la vida no es color de rosas, siempre podemos lograr lo que nos proponemos. Simplemente, gracias por ser mis padres.

A mis hermanos, Robert, Roger y Leyanet, que siempre han compartido conmigo los momentos buenos y malos; y aunque el mundo se caiga siempre estaremos juntos.

A mi novia, por aguantarme los cinco años en la universidad y por darme su amor incondicional, es lo mejor que me ha pasado en la vida.

A mis amigos del alma, de la universidad, a los que vienen conmigo desde hace mucho; a los que están lejos y a los que ya no están, todos son una parte importante de mi vida.

A todos gracias por formar parte de este momento.

Adrian

Resumen

La seguridad de la información es un factor fundamental de cualquier institución. La incorrecta manipulación, desvío o pérdida de la misma ocasionaría daños económicos y sociales irreparables. Para impedir que esto ocurra se necesita implementar medidas de administración, control de la información y seguridad.

La Aduana General de la República (AGR) posee un sistema que le permite controlar sus procesos. Toda la información que se maneja es de suma importancia, por lo que se debe garantizar su seguridad, integridad y disponibilidad. El sistema de control de acceso que se encuentra implantado actualmente está implementado con PHP estructurado y no cumple con las mejores prácticas de programación y diseño para aplicaciones Web probadas como aceptadas.

Teniendo en cuenta que este sistema se encuentra distribuido entre todas las aduanas del país, se hace necesario el establecimiento de una aplicación que se encargue de gestionar todo lo referente a la autenticación y control de acceso de los usuarios de la entidad.

Este trabajo propone la solución a utilizar en los Sistemas Tributarios y de Aduanas desarrollados bajo la arquitectura implantada, realizándose un estudio de los mecanismos de autenticación y control de acceso para de esta forma establecer un mecanismo propio.

Palabras claves: Sistemas Tributarios y de Aduanas, seguridad, integridad.

Tabla de contenido

Introducción

La Aduana General de la República es una organización que controla el tráfico internacional de medios de transporte, mercancías y viajeros, tanto a la entrada como a la salida del país. Estas actividades están acompañadas de un alto nivel profesional y eficiencia de sus trabajadores.

En la actualidad la AGR tiene en funcionamiento diferentes sistemas informáticos que facilitan el control y la gestión de los procesos. Estos sistemas difieren en cuanto al lenguaje de programación con que fueron desarrollados, plataformas y sistemas de base de datos que utilizan.

Con el objetivo de integrar todos estos sistemas de información, la AGR apoyada en el Centro de Automatización para la Información y la Dirección (CADI) comenzó a desarrollar el Sistema Único de Aduana (SUA), encargado de integrar todos los sistemas ya instaurados, de forma que garantice la comunicación de forma rápida y segura entre ellos. A petición de la AGR se desarrolla el sistema Gestión Integral Aduanera (GINA) por parte del CADI en colaboración con la Universidad de las Ciencias Informáticas, incluyendo además el desarrollo de nuevos sistemas según las necesidades de la AGR.

Actualmente el sistema que se está desarrollando por el Departamento de Soluciones para la Aduana, perteneciente al Centro de Informatización Gestión de Entidades (CEIGE) de la facultad 15, no cuenta con una aplicación para el control de acceso de los usuarios del sistema bajo la arquitectura de Symfony[1], esto incide negativamente en la seguridad de las operaciones que se realizan en la institución.

El estudio de la problemática anterior ha desencadenado un esfuerzo en aras de un desarrollo acorde con los paradigmas actuales de programación para los nuevos sistemas del Departamento de Soluciones para la Aduana, identificando el siguiente **problema a resolver:** ¿Cómo lograr que todas las operaciones que se realizan en las aplicaciones del Departamento de Soluciones para la Aduana sean seguras?

Para llevar a cabo la investigación se ha definido como **objeto de estudio:** los sistemas de seguridad y administración para las aplicaciones desarrolladas bajo la arquitectura de Symfony.

[1] Symfony: Es un framework desarrollado con PHP 5, diseñado para optimizar y simplificar el desarrollo de aplicaciones Web

Campo de acción: El control de acceso y autenticación para las aplicaciones que se desarrollan en el Departamento de Soluciones para la Aduana bajo la arquitectura de Symfony.

Siendo el **objetivo general:** Implementar un sistema de autenticación y control de acceso que pueda ser utilizado por las aplicaciones desarrolladas en el Departamento de Soluciones para la Aduana bajo la arquitectura del framework de PHP Symfony.

Para alcanzar el objetivo trazado se debe dar cumplimiento a los siguientes **objetivos específicos**:

- Definir qué mecanismo de interpretación de las peticiones en Symfony es más factible utilizar para la validación del control de acceso de los usuarios a las aplicaciones.
- Modelar las clases necesarias para poder implementar el mecanismo seleccionado.
- Implementar los mecanismos necesarios para realizar el control de acceso de los usuarios y la autenticación a las aplicaciones desarrolladas en Symfony en el Departamento de Soluciones para la Aduana.
- Implementar una solución que permita gestionar la información referente al sistema de seguridad planteado.

Para darle cumplimiento a todos los objetivos trazados durante la investigación, se han definido las siguientes tareas:

- Procesamiento de la información obtenida acerca de los sistemas de control de acceso y autenticación desarrollados en PHP, tanto en el mundo como en Cuba, para establecer una comparación entre ellos.
- Análisis de la información sobre sistemas de autenticación y control de acceso desarrollados con Symfony, para evaluar el desarrollo de este tipo de aplicaciones.
- Estudio del plugin de seguridad sfGuardPlugin de Symfony para analizar sus ventajas y desventajas.
- Realización del análisis de la propuesta de solución que permita identificar los requerimientos significativos para el sistema y generar los artefactos correspondientes.

- Diseño de la solución propuesta para definir las clases, paquetes, relaciones y configuraciones correspondientes al sistema.
- Implementación de la solución propuesta.

El trabajo está estructurado en tres capítulos, tal y como se describe a continuación:

Capítulo 1: Se enuncian los principales conceptos relacionados con los mecanismos de control de acceso y autenticación. Se realiza un estudio de algunas soluciones existentes, concluyendo con la selección del mecanismo más indicado para desarrollar el sistema.

Capítulo 2: En este capítulo se describen todos los artefactos que se generan durante el análisis y el diseño del sistema. Definición de los requerimientos del sistema, diseño de los prototipos de interfaz de usuario, diagramas de secuencia y diagrama de paquetes.

Capítulo 3: Se hace una descripción de las clases y componentes que se generan durante la implementación del sistema.

Capítulo 1: Fundamentación teórica

1.1 Introducción.

Un sistema de control de acceso protege la confidencialidad, integridad y disponibilidad de los recursos mediante mecanismos que dificultan la entrada de usuarios no autorizados a los mismos. El control de acceso generalmente consta de dos pasos:

- En primer lugar, la autenticación, que identifica al usuario o la máquina que trata de acceder a los recursos.
- En segundo lugar, ceder los derechos, es decir, la autorización, que dota al usuario de privilegios para poder efectuar ciertas operaciones con los datos protegidos, tales como leerlos o modificarlos.

El objetivo principal de este capítulo es dar una visión global de los mecanismos de control de acceso y autorización utilizados por los diferentes frameworks de PHP.

1.2 Seguridad de los sistemas

La necesidad de mantener la integridad de la información y de proteger los activos de las tecnologías de la información (TI), requiere de un proceso de administración de la seguridad. Este proceso incluye el establecimiento y mantenimiento de roles y responsabilidades de seguridad, políticas, estándares y procedimientos. La administración de seguridad también incluye realizar monitoreos de seguridad y pruebas periódicas así como realizar acciones correctivas sobre las debilidades o incidentes de seguridad identificados. Una efectiva administración de la seguridad protege todos los activos de las TI para minimizar el impacto causado por vulnerabilidades o incidentes de seguridad (1).

El aspecto de la seguridad es un elemento muy importante en la gestión de los procesos de negocio de una organización. La seguridad de la información persigue proteger los recursos de posibles accesos y modificaciones no autorizadas (2).

Los principales objetivos de la seguridad de la información se pueden resumir en (3; 4):

- ***Confidencialidad***: Describe el estado en el cual la información está protegida de revelaciones no autorizadas.

- ***Integridad***: Significa que la información no ha sido alterada o destruida por una acción accidental o por un intento malicioso.
- ***Disponibilidad***: Referencia al hecho de que una persona autorizada pueda acceder a la información en un apropiado período de tiempo. Las razones de la pérdida de disponibilidad pueden ser ataques o inestabilidades del sistema.
- ***Responsabilidad***: Asegurar que las acciones realizadas en el sistema por un usuario se puedan asociar únicamente a ese usuario, que será responsable de sus acciones. Es decir, que un usuario no pueda negar su implicación en una acción que realizó en el sistema.

1.3 Autenticación.

Por autenticación se entiende cualquier método que permite comprobar de manera segura cierta característica de un objeto, como su origen, su no manipulación, su identidad (5). Existen dos grandes grupos dentro de los métodos de autenticación:

- ***Autenticación de mensaje:*** Garantiza la procedencia de un mensaje, identificando al usuario que lo emitió. Por definición, este tipo de autenticación involucra también la integridad de los datos, es decir, permite detectar si el contenido del mensaje fue accidental o maliciosamente alterado.
- ***Autenticación de usuario:*** Garantiza que el usuario es quien afirma ser. En este proceso se corrobora la identidad del usuario en tiempo real, es decir, en el mismo momento que se están comunicando el verificador y el usuario.

1.3.1 Autenticación de mensaje.

Las técnicas más utilizadas para proveer autenticación de mensaje son (6):

- ***MAC (Message Authentication Code):*** Los códigos de autenticación de mensajes fueron diseñados especialmente para aplicaciones donde se requiere la integridad de los datos (pero no necesariamente su privacidad). Un MAC es un tipo de función unidireccional que toma dos entradas (un mensaje y una clave secreta) y retorna una cadena de tamaño fijo, siendo imposible volver a obtener la misma salida sin conocer la clave. Así, el emisor de un mensaje M, usa la clave secreta compartida K para calcular un MAC $h_K(M)$ sobre el mensaje y envía al receptor M y $h_K(M)$. El receptor determina la identidad de la fuente del mensaje (utilizando, por

ejemplo, un campo identificador en el texto no cifrado) y separa el MAC de los datos. Luego, calcula un MAC sobre estos datos, utilizando una clave MAC compartida, y compara el MAC calculado con el recibido. Si estos valores coinciden, el receptor interpreta que los datos son auténticos y que no fueron modificados durante su transmisión.

- ***Firma Digital:*** Simula una firma real. Para ello, se genera una prueba digital que solo el emisor de un mensaje puede crear, pero que todos pueden identificar como perteneciente a ese emisor. Un cifrado utilizando la clave privada del emisor sirve como firma, porque sólo el propietario de la clave privada puede crearlo y todos pueden verificarlo con la clave pública correspondiente. El cifrado (firma) puede aplicarse a todo el mensaje o a un pequeño bloque de datos que sea función del mensaje, por ejemplo, el valor hash. Por tanto, el emisor A envía al receptor B el mensaje M y la firma calculada sobre el hash del mensaje E_{SKA} (H (M)), donde SK_A es la clave privada del emisor, E es el algoritmo de cifrado asimétrico utilizado y H (M) es el hash del mensaje. El receptor debe entonces obtener la clave pública del emisor PK_A y realizar una operación de cifrado sobre la firma del mensaje, con lo que tendría H´ (M). Posteriormente, B debe calcular el hash del mensaje y compararlo con H´ (M). Si los dos resultados coinciden, se corrobora la autenticidad y la integridad del mensaje.

1.3.2 Autenticación de usuario.

Las técnicas de autenticación de usuario pueden clasificarse en tres categorías, dependiendo de qué deba presentar el usuario para demostrar su identidad (6):

- ***Algo conocido:*** El usuario debe demostrar que conoce algún tipo de información secreta, por ejemplo, una palabra clave, un número de identificación personal (PIN), una clave privada.
- ***Algo que Posee:*** El usuario requiere para identificarse algún tipo de dispositivo, como: una tarjeta magnética, una tarjeta inteligente, un generador de contraseñas o una llave electrónica.
- ***Algo Inherente (Propio del individuo):*** Aquí se requiere alguna característica fisiológica del usuario (biometría), por ejemplo: huellas dactilares, la voz, la retina, el iris. Esta categoría puede considerarse como un caso particular de la anterior, donde el dispositivo es el propio usuario.

Las técnicas de autenticación de usuario basadas en algo conocido, se pueden clasificar a su vez en tres categorías de acuerdo al nivel de seguridad que ofrecen (6):

- ***Autenticación Débil:*** Es el tipo de autenticación más extendido, y al que los usuarios ya están acostumbrados. Aquí, una contraseña o palabra clave (*password*), asociada a cada usuario, es el secreto compartido entre el usuario y el sistema. Para identificarse, el usuario introduce un identificador (*login*) y su contraseña. El sistema comprueba entonces que esa contraseña coincida con la almacenada para dicho identificador. Ya que las contraseñas son fijas y reutilizables, si un atacante se entera de la clave de un usuario, puede suplantar su identidad fácilmente.

  Para protegerse de los posibles ataques por fuerza bruta, de los que pueden ser víctima estos sistemas, se limita el número de intentos que el usuario puede hacer desde un terminal concreto y se introducen retardos cuando la contraseña introducida es errónea.

- ***Autenticación Fuerte:*** Para este tipo de autenticación una entidad (el demandante) prueba su identidad ante otra entidad (el verificador) demostrando que conoce un secreto asociado con su identidad, pero sin revelar dicho secreto al verificador. Esto se hace respondiendo a un desafío variable. Dicha respuesta es típicamente un número elegido por el verificador (aleatoria y secretamente) al inicio del protocolo. Si un atacante obtiene la respuesta al desafío de un usuario determinado, esa información no le será útil para suplantar posteriormente a ese usuario, ya que el desafío será diferente.

  Los mecanismos de autenticación fuerte basados en criptografía simétrica, requieren que el demandante y el verificador compartan una clave secreta. Cuando el sistema cuenta con un gran número de usuarios se suele usar un servidor confiable en línea, con el que cada usuario comparte una clave. Este servidor actúa a manera de Hub[2], proporcionando una clave de sesión común a las dos partes comunicantes cada vez que quieran autenticarse una con la otra. El protocolo Kerberos provee autenticación fuerte basada en cifrado simétrico e involucra el uso de una tercera parte de confianza en línea.

[2] Hub: Término en inglés con el que se denomina al concentrador, es un dispositivo que se utiliza como punto de conexión entre los componentes de una red de área local.

En los mecanismos de autenticación fuerte basados en criptografía asimétrica, el demandante demuestra que posee su clave privada, bien sea descifrando un desafío cifrado con su clave pública o firmando digitalmente el desafío. Para no comprometer la seguridad del sistema, el par de claves usado por estos mecanismos no debe ser usado con otros propósitos. La Infraestructura de Clave Pública (PKI) utiliza autenticación fuerte basada en criptografía asimétrica para proveer sus servicios de seguridad.

- ***Autenticación de Conocimiento Nulo:*** Los protocolos de conocimientos nulos (**ZK: *Zero-knowledge***) le permiten al probador (el demandante) demostrar el conocimiento de un secreto sin revelar ninguna información útil para el verificador (más allá de la que el verificador es capaz de deducir antes de ejecutar el protocolo). Los sistemas de prueba interactivos son un ejemplo de este tipo de protocolos. Aquí el objetivo del probador es convencer al verificador de que posee algún secreto, contestando correctamente preguntas que requieren del conocimiento de ese secreto para ser respondidas. Estos protocolos emplean técnicas asimétricas pero no utilizan firmas digitales o cifradas con clave pública. Tampoco usan cifradores de bloques, números de secuencia ni sellos de tiempo (*timestamps*).

1.3.3 Arquitecturas de Autenticación

Las técnicas de autenticación descritas anteriormente implican algún tipo de clave o secreto compartido entre las partes comunicantes. Debe existir, por tanto, una infraestructura de seguridad que se encargue de la generación, distribución y gestión de los secretos. Esta infraestructura de seguridad brindará una base segura a toda la organización y deberá ser accesible por todas las aplicaciones y objetos de la organización que necesiten seguridad (7).

Durante un proceso típico de autenticación, el usuario presenta su credencial (por ejemplo: *login* y *password*) o el resultado de una operación criptográfica que involucre su credencial, a la autoridad de autenticación. Ésta valida la credencial usando los datos guardados en la base de datos. Si la credencial proporcionada por el usuario y la guardada en la base de datos coincide, o si el resultado de la operación criptográfica sobre la credencial guardada en la base de datos es igual a la información suministrada por el usuario, la identidad del usuario es considerada auténtica (8).

Sin embargo, el usuario debe compartir una credencial con cada dominio y autenticarse cada vez que quiera acceder a algún recurso. Dada la diversidad de plataformas de computación, sistemas

operativos y de software de control de acceso, lo más deseable es registrarse en múltiples sistemas una vez y simultáneamente a través de una sola transacción.

Nace aquí el registro único (SSO – *Single Sign - On*). Con SSO el usuario debe autenticarse solo una vez para acceder a múltiples sistemas. El registro único puede usarse en infraestructuras con diversas autoridades de autenticación, es decir, implementadas en diferentes plataformas y gobernadas por diversas organizaciones. Las arquitecturas más simples son las que usan un conjunto de credenciales. Las dos más importantes son (8):

- ***Sistemas Basados en Testigos (Token - Based):*** En una arquitectura basada en testigos, los usuarios obtienen un testigo temporal después de autenticarse exitosamente ante su Trusted Third Party (TTP). Este testigo puede ser guardado en el dispositivo del usuario y reutilizado para probar su identidad ante las TTP de dominios de autenticación secundarios. Para validar el testigo de un usuario, estas TTP usan métodos criptográficos basados en claves secretas establecidas previamente entre ellas y la TTP del dominio de autenticación primario. Estas claves criptográficas permiten establecer una relación de confianza entre diferentes dominios de autenticación.

- ***Sistemas basados en PKI (PKI - Based):*** En esta arquitectura, los usuarios se registran primero ante una autoridad de autenticación confiable, la Autoridad Certificadora (AC). Durante el proceso de registro los usuarios se identifican a través de un conjunto de credenciales. El software del usuario genera un par asimétrico de clave, y la clave pública es ofrecida a la autoridad certificadora para su certificación. Después de recibir las credenciales del usuario y la clave pública, la autoridad certificadora verifica si las credenciales son válidas. Si es así, genera un certificado de clave pública y se lo retorna al usuario. Este certificado y la clave son guardados de manera segura en el dispositivo del usuario u otro medio, como tarjeta inteligente. Estos son usados para probar la identidad del usuario ante otras autoridades de certificación en solicitudes de autenticación posteriores. En esta arquitectura, la relación de confianza entre la autoridad de certificación primaria y las secundarias se establece a través de un certificado expedido por la AC primaria y la AC secundaria.

1.4 Autorización.

La autorización está estrechamente ligada con la autenticación. Una vez que el usuario ha validado su identidad para acceder a algún recurso, es necesario restringir sus acciones de acuerdo a quien es,

que está tratando de hacer (9). La autorización dota al usuario de privilegios para poder efectuar ciertas operaciones con los datos o recursos protegidos.

1.4.1 Modelos de control de acceso.

Un factor fundamental para determinar el funcionamiento de todo entorno de autorización es la definición del modelo de control de acceso que se va a establecer.

Los mecanismos utilizados para restringir el acceso a los recursos son generalmente de una (o algunas veces la combinación) de dos formas (10):

- ***Control de Acceso Discrecional (DAC):*** Le deja las decisiones de control de acceso al propietario del recurso, de manera que es este quien decide que sujetos pueden realizar determinadas acciones sobre los recursos poseídos. Así, un sujeto con permisos de acceso puede otorgarlos a otro sujeto.

- ***Control de Acceso Obligatorio (MAC):*** Este control de acceso se basa en reglas establecidas por una autoridad central. MAC restringe el acceso a los objetos basándose en la sensibilidad de la información que estos contienen (representada por una etiqueta) y en la autorización formal de los sujetos para acceder a dicha información. Los objetos son considerados como entidades pasivas que almacenan información, mientras que los sujetos son entidades activas que realizan peticiones de acceso a los objetos.

Existen diversos modelos de control de acceso de tipo DAC y/o MAC (11). Los más comunes son:

- ***Control de Acceso Basado en Identidad (IBAC):*** En este caso los permisos de acceso se asocian directamente al identificador del sujeto, es decir, el nombre del usuario. Por tanto se garantiza el acceso al recurso solo cuando exista dicha asociación. Con IBAC no se asocian etiquetas de seguridad a los usuarios, por lo que este es principalmente un mecanismo de control de acceso discrecional.

  Un ejemplo de IBAC son las Listas de Control de Acceso (ACLs), encontradas comúnmente en sistemas operativos y servicios de seguridad en red. Una Lista de Control de Acceso contiene los identificadores de los usuarios junto con sus derechos de acceso a un recurso determinado, como leer, escribir, ejecutar. Esta estructura básica de autorización únicamente extiende el concepto de autenticación, ya que si el usuario no puede autenticarse correctamente ante el guardián del recurso, su solicitud de acceso es denegada.

Entre más usuarios soliciten el acceso a un recurso más identificadores contendrá la ACL, lo que dificulta el manejo de estas listas y las hace una alternativa poco escalable.

Por otra parte, la decisión de control de acceso no depende de alguna función o característica de la organización a la que pertenece el usuario sino solamente de los identificadores, así que su uso es inapropiado a nivel empresarial.

- ***Control de Acceso Basado en Roles (RBAC):*** Restringe el acceso a los recursos basándose en la función o rol que desempeña el sujeto dentro de la organización. Los permisos para acceder a un recurso son asignados a cada rol, en lugar de asociarlos directamente al identificador del sujeto. El RBAC es escalable y reduce significativamente la cantidad de información de administración necesaria, ya que los permisos no son asignados constante e individualmente a cada usuario. RBAC es primordialmente un mecanismo de control de acceso discrecional. Generalmente no tiene en cuenta las características de los recursos (más que sus identificadores) y no obtienen ninguna información relevante de seguridad del entorno.

- ***Control de Acceso Basado en Atributos (ABAC):*** En ABAC los privilegios son establecidos en base a la colección de atributos que posee el usuario y una política que los determina. ABAC es la convergencia natural de los modelos de control de acceso IBAC y RBAC. La representación de las políticas en ABAC es semánticamente más rica y expresiva. Además posee una mayor granularidad ya que puede basarse en cualquier combinación de atributos de sujeto, de recursos y de entorno.

- ***Control de Acceso Basado en Mallas (LBAC):*** En este caso una colección totalmente ordenada de etiquetas de seguridad se combina con un grupo de categorías y forman una *malla*. Con ellos se obtienen un conjunto de clases de seguridad que varía desde la más baja a la más alta (12). Generalmente el modelo LBAC es manejable cuando existe un número relativamente pequeño de etiquetas de seguridad y categorías, por lo que es solo efectivo para ciertos escenarios de seguridad poco granulados y carentes de flexibilidad y escalabilidad. LBAC es un mecanismo de control de acceso obligatorio.

1.5 Análisis de los frameworks de PHP.

En la actualidad existen un gran número de frameworks implementados en PHP, encaminados a agilizar y hacer más cómodo el proceso de desarrollo de aplicaciones empresariales. Entre los más utilizados en la comunidad de desarrolladores de PHP se encuentran Zend Framework, Kumbia,

CakePHP, CodeIgniter y Symfony. Cada uno implementa sus propios mecanismos de seguridad pero ninguno la enfoca hasta el nivel de las acciones que se pueden realizar, o sea que solo se chequea el acceso a los módulos de estas aplicaciones.

1.5.1 Zend Framework

Zend Framework es una librería de componentes escritos en PHP 5 para facilitar el desarrollo de aplicaciones Web. Está formado por muchos componentes, que están divididos por paquetes de los cuales uno se encarga de la seguridad (13). Los componentes de este paquete son(14):

- **Zend_Auth:** Se utiliza para autenticar contra un tipo particular de servicio de autenticación, tales como LDAP, RDBMS, o el almacenamiento basado en ficheros. Cada clase de Zend_Auth implementa Zend_Auth_Adapter_Interface. Esta interfaz define un método autenticar (), que una clase de adaptador debe implementar para la realización de una consulta de autenticación. Cada clase debe estar preparada antes de llamar a autenticar (). Dicha preparación incluye el adaptador de la creación de credenciales (por ejemplo, nombre de usuario y contraseña) y los valores que definen las opciones de configuración específica del adaptador, como los ajustes de conexión de base de datos para un adaptador de tabla de base de datos.

- **Zend_Session:** Este componente envuelve la actual extensión session de PHP con una interfaz de administración y gestión. Zend_Session_Namespace proporciona una interfaz estándar, orientada a objetos para trabajar con namespaces que persiste dentro del mecanismo de sesión estándar de PHP y almacena información referente a los usuarios autenticados. Los namespaces se utilizan para separar los datos de la sesión, a pesar de que existe un namespace por defecto para almacenar todos los datos juntos.

- **Zend_Acl:** Es una implementación de las Listas de Control de Acceso en PHP que permite asignar roles y permisos a usuarios o grupos de usuarios en la aplicación. Soporta la herencia de roles y recursos. Además soporta el control de acceso condicional basado en una interfaz de declaraciones. La creación de un recurso en Zend_Acl es muy simple. Zend_Acl proporciona el recurso Zend_Acl_Resource_Interface, para facilitar la creación de recursos en una aplicación. Una clase sólo necesitan implementar esta interfaz, que consiste en un método único, getResourceId (), por Zend_Acl para reconocer el objeto como un recurso.

La estructura que maneja Zend Framework no se adecúa a las características arquitectónicas de Symfony, aunque posee características que pueden adaptarse para el desarrollo de una aplicación

más flexible que permita el control de acceso y autenticación desde un sistema centralizado. Uno de los inconvenientes principales que surgen a la hora de utilizar Zend Framework es que actualmente todas las aplicaciones que se desarrollan por el Departamento de Soluciones para la Aduana se rigen estrictamente por la arquitectura de Symfony, lo que impide que se puedan integrar librerías de Zend Framework al desarrollo de las mismas.

1.5.2 Kumbia PHP Framework

Kumbia es un framework libre escrito en PHP 5 basado en el modelo MVC. Permite la separación de las reglas de negocio, lógica de aplicación y vistas de presentación de una aplicación Web. Además posee otras herramientas y clases que ayudan a acortar el tiempo de desarrollo. Fomenta la velocidad y eficiencia en la creación y mantenimiento de las aplicaciones, reemplazando tareas de codificación repetitivas (15).

Está basado en los siguientes conceptos:

- Compatible con muchas plataformas.
- Fácil de instalar y configurar.
- Fácil de aprender.
- Listo para aplicaciones comerciales.
- Convención sobre Configuración.
- Soporta muchas características de las aplicaciones Web actuales.
- Soporta las prácticas y patrones de programación más productivos y eficientes.
- Produce aplicaciones fáciles de mantener.
- Es compatible con las bases de datos disponibles actuales más usadas: PostgreSQL, MySQL, Oracle.

1.5.2.1 Mecanismos de seguridad

Para la gestión de la seguridad, Kumbia utiliza mecanismos, los cuales se describen a continuación:

- ***Listas de Control de Acceso (ACL por sus siglas en inglés)***: Este es un 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. Esta clase posee un conjunto de métodos que permiten la gestión de los permisos, objetos y acciones.

- ***Auth*:** Permite realizar el proceso de autenticación utilizando diferentes adaptadores como Digest, LDAP, Bases de Datos, Kerberos5 y Radius.
- ***Session:*** Mantiene la persistencia de sesión independiente entre aplicaciones y usuarios autenticados.
- ***Security:*** Permite detectar ataques de negación de servicios o bloquear el acceso a un cliente a través de su dirección IP.

La forma en que Kumbia implementa la administración mediante las ACL permite que se pueda controlar de forma cómoda y flexible los permisos que tiene cada usuario hasta nivel de acción. El inconveniente de las listas de control de acceso reside en que a medida que crece la cantidad de usuarios, se hace más difícil administrar los permisos correspondientes a los mismos. Por tal razón las ACLs no son una buena opción para controlar el acceso de los usuarios a los recursos.

1.5.3 CakePHP

CakePHP es un framework escrito en PHP que provee una extensa arquitectura para el desarrollo, mantenimiento y despliegue de aplicaciones. Reduce los costos de desarrollo y ayuda a los desarrolladores a escribir menos código. Un sistema de autenticación de usuarios es una parte común de muchas aplicaciones web. En CakePHP hay muchas formas para autenticar usuarios, cada una de estas provee diferentes opciones. La esencia de componente de autenticación es comprobar si el usuario tiene una cuenta con el sitio. De ser así, el componente da al usuario acceso completo a sitio.

Este componente se puede combinar con el componente ACL (Access Control Lists) para crear niveles más complejos de acceso al sitio. El componente ACL, por ejemplo, podría permitir acceso a un usuario a áreas públicas del sitio, mientras que concede a otro usuario acceso a porciones administrativas protegidas del sitio (16).

Incluye un conjunto de componentes integrados que proveen una serie de funcionalidades para tareas realizadas comúnmente. A continuación se describen cada uno de esos componentes:

- ***Acl***: Provee una interfaz para Listas de Control de Acceso, basadas en archivos ini y bases de datos.

- ***Auth***: Este componente provee un sistema de autenticación fácil de utilizar usando diferentes procesos de validación, como son los callbacks en los controladores, ACL y callbacks en los objetos.

- ***Session***: Provee una capa de almacenamiento independiente a las sesiones de PHP. Actúa como una abstracción para acceder a las variables $_SESSION, y a su vez, agrega distintos métodos útiles relacionados con este arreglo. Las sesiones pueden persistirse de varias maneras, por defecto se utilizan los métodos provistos por PHP; sin embargo existen otras opciones:

  Cake: Guarda los archivos de la sesión en el directorio temporal de tu aplicación tmp/sessions.

  Database: Guarda la información dentro de la base de datos.

  PHP: La opción por defecto. Guarda la información en el php.ini.

- ***Security***: Permite aumentar la seguridad y gestionar la autenticación HTTP. Este componente provee una forma fácil de aumentar la seguridad de tu aplicación. Con el componente Security se puede crear una interfaz para manejar autenticaciones HTTP. Se configura en el método beforeFilter () de los controladores, y tiene distintos parámetros para ajustar. Todas esas propiedades pueden configurarse directamente con los métodos creados para tal fin y que se llaman de la misma manera.

1.5.3.1 Listas de Control de Acceso.

Este framework permite definir las ACL en archivos ini o en bases de datos. Son implementadas en una estructura de árbol, que está formada por un árbol Access Request Objects (ARO) y un árbol Access Control Objects (ACO), organizando los objetos así, los permisos se gestionan de forma granular y se mantiene una visión general.

Utilizar la estructura de árbol ARO, permite definir los permisos que se aplican a un grupo de usuarios, de una sola vez. Por defecto cakePHP utiliza el sistema ACL basado en bases de datos. Consiste en un conjunto de modelos y aplicación de consola que viene con la instalación del framework. Los modelos son utilizados por cakePHP para interactuar con la base de datos, para poder almacenar y recuperar los nodos en forma de árbol (16).

Esta implementación no permite que se pueda administrar la seguridad de forma centralizada, además que no se ajusta a la arquitectura planteada por el framework de PHP Symfony.

1.5.4 CodeIgniter

CodeIgniter es un conjunto de herramientas para personas que construyen su aplicación Web usando PHP. Su objetivo es permitir al desarrollar proyectos mucho más rápido de lo que podría si lo escribiese desde cero, proveyéndole un rico juego de librerías para tareas más comunes, así como una interface simple y estructura lógica para acceder a esas librerías.

CodeIgniter gestiona la seguridad de diferentes formas. Es justamente restrictivo sobre que caracteres permitir en las cadenas URI[3] para ayudar a minimizar la posibilidad de que datos maliciosos puedan ser pasados a su aplicación (17).

Las URL sólo pueden contener lo siguiente:

- Texto alfanumérico.
- Tilde.
- Punto.
- Dos puntos.
- Guión bajo.
- Guión.

Los datos GET son simplemente anulados por CodeIgniter ya que el sistema utiliza segmentos URI en vez de las tradicionales query string de URL (a menos que la opción query string esté habilitada en su archivo config). El arreglo global GET es destruido por la clase de Entrada (input) durante la inicialización del sistema.

Desde la inicialización del sistema, todas las variables globales son destruidas, excepto aquellas que son encontradas en los arreglos $_POST y $_COOKIE. La rutina de eliminación es efectivamente la misma que register_globals = off.

[3] URI: Identificador de Recurso Uniforme (Uniform Resource Identifiers por sus siglas en inglés).

La directiva magic_quotes_runtime es apagada durante la inicialización del sistema para que no tenga que remover las barras cuando se recuperen datos de la base de datos.

CodeIgniter viene con un filtro de Cross Site Scripting (XSS). Este filtro busca técnicas comúnmente usadas para incluir Java script malicioso a sus datos, u otro tipo de código que intente secuestrar cookies.

CodeIgniter solo implementa defensas contra algunos tipos de ataque mediante funciones que le permiten tener direcciones URL más segura y mediante el tratamiento de algunas variables propias el PHP, lo que hace engorrosa la configuración. No posee un sistema de control de acceso definido, sino que estos mecanismos tienen que ser implementados por el propio desarrollador.

1.5.5 Symfony.

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de aplicaciones Web. Está desarrollado completamente en PHP 5. Separa la lógica del negocio, la lógica del servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Además automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Ha sido probado en numerosos proyectos reales y se utiliza en sitios Web de comercio electrónico de primer nivel. Es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft (18).

La configuración de la seguridad en Symfony se realiza a través de los archivos security.yml que se encuentran en el directorio config de cada aplicación. Es los mismos se especifica cuáles son las acciones seguras dentro de la aplicación. Estas configuraciones se hacen complejas en la medida en que aumenta el número de usuarios y las credenciales que tienen los mismos.

Symfony cuenta con una serie de plugins[4] que le ayudan a gestionar la seguridad de las aplicaciones, dentro de ellos se encuentra sfGuardPlugin.

[4] Plugin: Pequeño programa que añade alguna función a otro programa, habitualmente de mayor tamaño. Un programa puede tener uno o más plug-ins. Son muy utilizados en los programas navegadores para ampliar sus funcionalidades.

1.5.5.1 sfGuardPlugin

SfGuardPlugin permite incluir autenticación, autorización y otras opciones de gestión de usuarios más avanzadas que las que proporciona por defecto Symfony. Este plugin gestiona de manera muy sencilla usuarios, roles, y permisos; y para ellos se apoya en las clases de su propio modelo. Posee tareas que permiten generar los modelos, formularios y filtros.

Está compuesto por cuatro módulos:

- **sfGuardAuth:** Es el módulo que se encarga de la autenticación de los usuarios, así como del control de acceso de los mismos a las aplicaciones del sistema.
- **sfGuardGroup:** Es el módulo encargado de la gestión de grupos y roles.
- **sfGuardPermission:** Es el módulo encargado de la gestión de los permisos.
- **sfGuardUser:** Es el encargado de la gestión de usuarios.

Todas las configuraciones de este plugin se encuentran en archivos de tipo yml, lo que dificulta un poco el trabajo con las credenciales de acceso a los módulos y acciones de las aplicaciones. No permite que se controle el acceso a las aplicaciones que se encuentran en un mismo proyecto. Además de que es poco flexible para los requerimientos de seguridad de las aplicaciones que se desarrollan en el Departamento de Soluciones para la Aduana.

1.6 Sistemas realizados en la Universidad de las Ciencias Informáticas

1.6.1 Sistema de Autenticación de Aplicaciones de la Intranet

La investigación en la UCI sobre los software de gestión de seguridad arrojó como resultado la existencia del Sistema de Gestión de Sesiones basado en la arquitectura SSO cuyo objetivo es gestionar las sesiones de los usuarios en un único proceso de autenticación invisible a los ojos de los mismos, a través de un sistema con una fachada de servicios web que sea capaz de gestionar la apertura y cierre de sesiones por parte de las personas que trabajan en el dominio uci.cu. Este sistema no soluciona el problema existente, ya que sólo propone centralizar la autenticación de usuarios y mantiene la autorización como responsabilidad de cada aplicación.

1.6.2 Sistemas de Gestión de Accesos

En el año 2008 se realiza un trabajo de diploma con el nombre “Sistema de Gestión de Accesos (SGA)”, basado en la necesidad de implementar un sistema que controle centralizadamente los accesos y garantice seguridad en la red y aplicaciones de la misma, el cual brinda principalmente, los servicios de autenticación y autorización de usuarios a las disímiles aplicaciones. Asimismo soporta los mecanismos necesarios para, a través de él, obtener información de las aplicaciones, funcionalidades, roles, usuarios y grupos de usuarios de la Universidad. Este sistema tiene la desventaja de que los usuarios tienen que autenticarse cada vez que acceden a una aplicación.

1.6.3 Sistema de Gestión de Sesiones

En el 2008 se realiza el trabajo de diploma “Sistema de Gestión de Sesiones (SGS)” el cual se convierte en una iniciativa viable de protección y control que permite ofrecer servicios de valor añadido con altos niveles de seguridad y confianza a los usuarios.

La implantación de este sistema, logra mejorar la seguridad de la red en nuestra universidad, permitiendo a su vez a los usuarios disminuir su trabajo de autenticación y el tiempo que pierden los mismos en este proceso. Además en la universidad cada uno de los servicios o aplicaciones cuenta con su propio componente de seguridad, lo cual generalmente compromete la seguridad de todo el sistema, dado que el nivel de seguridad de todo un sistema es igual al nivel de seguridad del componente más inseguro que tenga.

Esta es una de las razones por las que no se usa esta solución de software pues no hace un control de roles ni de usuarios, pero si permite a los usuarios autenticarse una única vez y hacer un control de sesiones eficiente.

1.6.4 Sistema de Gestión Integral de Seguridad (SIGIS)

Por otra parte se encuentra el Sistema de Gestión Integral de Seguridad (SIGIS) desarrollado por el proyecto ERP – Cuba. Posee módulos para la gestión de usuarios, roles y grupos de usuarios, además de que posee sistema de autenticación LDAP totalmente configurable. Es un componente que cubre muchas necesidades de seguridad pero que no se acopla a la arquitectura de Symfony. Está implementado con Zend Framework y para su implantación requiere que se incluya el componente completo dentro del directorio público del proyecto de Symfony. Ya en nuevas versiones del producto

se está trabajando por cambiar su arquitectura, aunque no se ha logrado aun integrar este sistema con los desarrollados en el Departamento de Soluciones para la Aduana. Todos estos problemas traerían como consecuencia un menor rendimiento del sistema.

1.6.5 Otros sistemas

Además en la facultad 10 se llevó a cabo el desarrollo de un software de autenticación y control centralizado para la corporación Petróleos de Venezuela SA (PDVSA) capaz de mapear la credenciales de los usuarios hacia todas las aplicaciones de dicho sistema y otros sistemas heterogéneos, proporcionando una infraestructura para simular un login único para el usuario corporativo frente a una plataforma tecnológica heterogénea dentro del ambiente de aplicativos de integración, en que los usuarios puedan acceder a diferentes aplicaciones con solo un conjunto de credenciales. Esta aplicación trajo como resultado principal la importancia de mapeos entre diferentes entornos que manejan diferentes mecanismos de seguridad.

1.7 Conclusiones

En este capítulo se ha hecho un análisis de diferentes mecanismos de control de acceso implementados por algunos de los frameworks más utilizados por la comunidad de desarrollo de PHP. Debido a la estructura de la Aduana General de la República se hace necesario lograr un mecanismo más flexible que permita manejar con mayor facilidad todos los niveles de permisos a los usuarios de la entidad, para ello se tomarán las mejores prácticas de cada framework para crear una solución que se ajuste a los requerimientos de seguridad que necesitan las aplicaciones del Departamento de Soluciones para la Aduana, sin quebrantar las directivas de desarrollo de Symfony.

Capítulo 2: Análisis y diseño de la propuesta de solución

2.1 Introducción

En este capítulo se hace un levantamiento de los requisitos funcionales del sistema a implementarse, además de la descripción de los artefactos del análisis y diseño que se generan durante el flujo de trabajo. Se muestran también los prototipos de interfaz de usuario correspondientes a los diferentes módulos del sistema.

2.2 Propuesta de solución

Para darle solución al problema planteado en el Capítulo 1, se propone la implementación de un componente que tendrá la estructura de un plugin, tal y como lo plantea la arquitectura de Symfony. El mismo estará conformado por una estructura de directorio que cumple con las especificaciones requeridas por el framework.

2.2.1 Control de Acceso al sistema, subsistemas, funcionalidades y acciones

El control de acceso se manejará a nivel de usuarios y roles. Cada rol tendrá un nivel de acceso al sistema descrito en las acciones que puede ejecutar en cada uno de los módulos y aplicaciones. El sistema tendrá el debido registro de la relación tanto entre los usuarios y los roles, como entre los roles y las aplicaciones. La información referente a los usuarios se manejará mediante sesiones en tiempo de ejecución de la aplicación. El manejo de las sesiones de los usuarios se realizará a partir de la clase sfUser o una que extienda de ella. La utilización de la clase seleccionada para esto debe ser especificada en sistema de seguridad como clase manejadora de sesiones por defecto.

Para modelar el chequeo de la seguridad en las peticiones de los usuarios a las aplicaciones se deben comprobar el rol del usuario y el nivel de acceso sobre la funcionalidad que se está accediendo. Para manejar este mecanismo se proponen la siguiente variante:

El control de acceso a las aplicaciones se realizará en el controlador frontal de cada aplicación, apoyándose en los Filtros de Symfony que ejecutarán el mecanismo de autenticación en el Sistema de Seguridad. Los filtros se incluirán en la cadena de Filtros del sistema y estos se encargarán de chequear el acceso del usuario autenticado contra la aplicación/módulo/acción a la que se esté accediendo.

2.2.2 Registro de Subsistemas y sus funcionalidades de manera dinámica.

El sistema interactuará con los sistemas que estén instalados sobre el mismo proyecto, en el mismo servidor de aplicaciones de las funcionalidades a las que controlará la seguridad. Estará almacenado en forma de Plugin en el proyecto, permitiendo el acceso de sus funcionalidades a las aplicaciones que estén bajo el dominio del mismo proyecto.

Para el registro de los subsistemas y sus funcionalidades, el sistema cargará la estructura de cada aplicación con sus respectivos módulos y acciones, que se encontrará en un archivo de configuración YAML. Posteriormente se guardará en la base de datos esta estructura en forma de árbol jerárquico.

Todas estas configuraciones se hacen a través de tareas de Symfony que permiten automatizar todo el proceso de carga de las estructuras de las aplicaciones, módulos y acciones; y además se encargará de validar que las funcionalidades tengan asignadas las acciones que realmente les corresponden.

2.3 Herramientas seleccionadas para el desarrollo de la solución

2.3.1 Herramienta CASE: Visual Paradigm

La tecnología CASE (Computer Aided Software Engineering, Ingeniería de software asistida por computadoras) supone la automatización del desarrollo de software, contribuyendo a mejorar la productividad en el desarrollo de sistemas de información. Automatiza además la documentación, la generación de código, el chequeo de errores y la gestión del proyecto permitiendo así la reutilización y la portabilidad del software y la estandarización de la documentación.

Visual Paradigm es una herramienta CASE profesional que soporta todo el ciclo de vida del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones.

Entre sus características principales esta que permite el control de versiones, consta de un entorno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso. También permite el despliegue de EJB (Enterprise java Beans), la construcción de diagramas de flujo de datos, la generación de objetos Java desde la base de datos, la transformación de diagramas de Entidad-Relación en tablas de base de datos. Posee

un generador de informes, la reorganización de las figuras y conectores de los diagramas y un editor de figuras (20).

2.3.2 Entorno Integrado de Desarrollo: NetBeans 6.8

El IDE NetBeans es un entorno integrado de desarrollo galardonado, disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, y de escritorio y aplicaciones móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails , Groovy y Grails, y C / C + +.

El proyecto NetBeans es compatible con una vibrante comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación, así como una amplia selección de plug-ins de terceros.

Completo soporte para PHP 5.3: namespaces, funciones lambda y cierres. Posee además un completo soporte para Symfony, permitiendo generar proyectos y trabajar con las librerías del framework, incluyendo los archivos YAML.

Crea un proyecto PHP desde un control remoto de aplicaciones PHP, PHPUnit, la cobertura de código, FTP / SFTP y mejoras de integración (21).

2.4 Requisitos

A través de los años se ha podido constatar que los requerimientos o requisitos son la pieza fundamental en un proyecto de desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos. Además la especificación de los requerimientos es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto ya que estos son un reflejo detallado de las necesidades de los clientes o usuarios del sistema.

Un requerimiento es una descripción de una condición o capacidad que debe cumplir un sistema, ya sea derivada de una necesidad de usuario identificada, o bien, estipulada en un contrato, estándar, especificación u otro documento formalmente impuesto al inicio del proceso (22).

A continuación se muestran los requisitos funcionales y no funcionales del sistema. Para más detalles ver Anexo 1, donde se hacen las descripciones correspondientes a cada requisito funcional.

2.4.1 Requisitos funcionales

El sistema está compuesto por cinco módulos, de ellos cuarto corresponden a la autenticación y control de acceso, a continuación se relacionan los requisitos del sistema correspondientes a cada módulo.

Requisitos funcionales del Módulo suAdminAuth

R1 Requisito funcional Autenticar Usuario

R2 Requisito funcional Comprobar Acceso a Acción

R3 Requisito funcional Construir Menú de Navegación

Requisitos funcionales del Módulo suAdminDomain

R4 Requisito funcional Gestionar Dominio

- R4.1 Insertar Dominio
- R4.2 Eliminar Dominio
- R4.3 Modificar Dominio
- R4.4 Buscar y Visualizar Dominios

Requisitos funcionales del Módulo suAdminRole

R5 Requisito funcional Gestionar Rol

- R5.1 Insertar Rol
- R5.2 Eliminar Rol
- R5.3 Modificar Rol
- R5.4 Buscar y Visualizar Roles
- R5.5 Asignar Permisos a Roles

Requisitos funcionales del Módulo suAdminUser

R6 Requisito funcional Gestionar Usuario

- R6.1 Insertar Usuario
- R6.2 Eliminar Usuario
- R6.3 Modificar Usuario
- R6.4 Mostrar y Visualizar Usuarios
- R6.5 Asignar Roles a Usuarios

Otros requisitos del sistema

R7 Requisito funcional Cargar Estructura del Proyecto Mediante CLI

R8 Requisito funcional Cargar Estructura del Menú Mediante CLI

2.4.2 Requisitos no funcionales

2.4.2.1 Usabilidad

Los sistemas desarrollados por el Departamento de Soluciones para la Aduana deben prestar facilidades de usabilidad que satisfagan las necesidades de los usuarios, deben contar con un menú que les permita a los usuarios acceder a las principales funciones que son de su interés. Así como brindar a los usuarios la posibilidad de interactuar con los productos sin tener previa preparación, solo con los conocimientos necesarios del negocio, en este aspecto no se incluye la parte administrativa de las aplicaciones que si requerirán de una preparación para su manipulación. La resolución de la página se adaptará la resolución de pantalla en el cliente.

2.4.2.2 Rendimiento

Los sistemas en general deben poseer un elevado nivel de eficiencia, teniendo en cuenta que las transacciones de datos pueden ser elevadas, así como el volumen de la Base de Datos. Por tanto se requiere del diseño, desarrollo de mecanismos y técnicas de programación óptimas para el lenguaje de programación seleccionado.

2.4.2.3 Soporte

Las aplicaciones clientes deben ser capaz de correr sobre cualquier plataforma, para el caso de Windows se recomienda XP por la experiencia acumulada por los usuarios. Para la parte servidora se recomienda que corra sobre plataforma Linux.

Ser programado en PHP 5.2.7 y con un gestor de base de datos PostgreSQL 8.2.4 o superior.

2.4.2.4 Hardware

***Cliente*:** Las aplicaciones son desarrolladas para que las PC clientes de los usuarios puedan usar las aplicaciones con el menor requerimiento posible. Estos requerimientos mínimos son:

Procesador Pentium IV o superior, 333 MHz mínimo pero recomendado 1.8 GHz.

40 GB o más de capacidad.

Mínimo de memoria RAM 128 DIM.

Adaptador de Red y conectividad.

Se recomienda la utilización de clientes ligeros para aprovechar las facilidades de mantenimiento que esto brinda.

***Servidor*:** Debido a que la cantidad de datos manejados por este tipo de aplicaciones se requiere separar el servidor de aplicación y el de Base de Datos. Cada uno de ellos requiere de características de Hardware específicas.

Servidor de Aplicaciones:

CPU = 2 Dual Core AMD Opteron 64 bits a 2.2 GHz.

RAM = 2 GB.

Almacenamiento = 1 Discos SATA de 73 GB en RAID 1.

Servidor de Base de Datos:

CPU = 2 Dual Core AMD Opteron 64 bits a 2.2 GHz.

RAM = 4 GB.

Almacenamiento = 2 Discos SATA de 73 GB en RAID 1

2.4.2.5 Software

Los usuarios deben tener instalado como navegador el Mozilla Firefox 3.5, el Internet Explorer 6 o superior. Sistema Operativo Windows NT o Linux, para PC o MacOS para Macintosh. Nada de esto sería necesario para los clientes ligeros.

El servidor de aplicación debe tener instalado PHP 5.2.7 o superior, Servidor Web Apache 2.2.10 o superior. En la Base de Datos es necesario que esté instalada una versión de PostgreSQL 8.2.4 o superior.

2.4.2.6 Portabilidad

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

2.4.2.7 Seguridad

Debido a que la información con la que se va a trabajar es confidencial los sistemas deben contar con las políticas de seguridad adecuadas, la misma debe estar basada en niveles de acceso a la información de acuerdo con el nivel jerárquico que tenga el usuario autorizado por los administradores y seguridad para la transmisión de los datos. Garantizar que las transacciones de datos se ejecuten por vías seguras, ya sea en la red o por soporte físico.

2.5 Prototipos de interfaz de usuario

En este epígrafe se relacionan los prototipos de interfaz de usuario más significativos para el sistema.

2.5.1 Requisito funcional R1 Autenticar Usuario

Fig. 1 Prototipo de interfaz de usuario. Autenticar Usuario

2.5.2 Requisito funcional R4.1 Insertar Dominio

Fig. 2 Prototipo de interfaz de usuario. Insertar Dominio

2.5.3 Requisito funcional R4.2 Eliminar Dominio

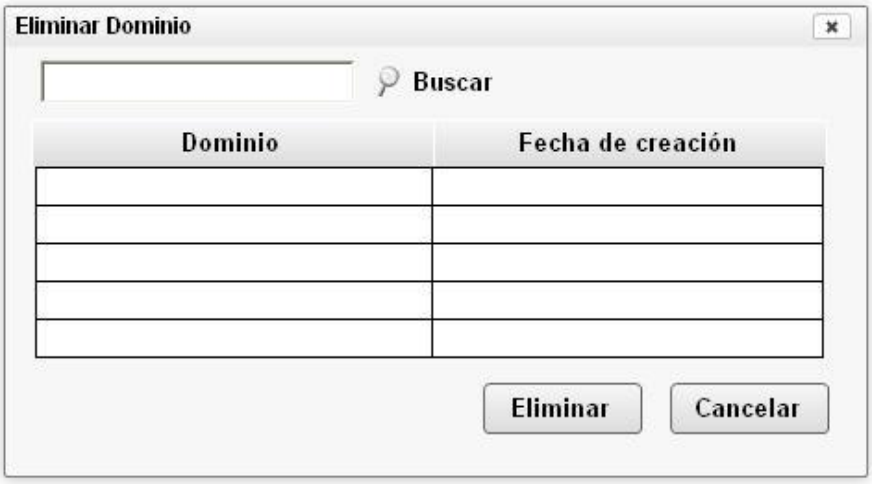

Fig. 3 Prototipo de interfaz de usuario. Eliminar Dominio

2.5.4 Requisito funcional R4.4 Buscar y Visualizar Dominios

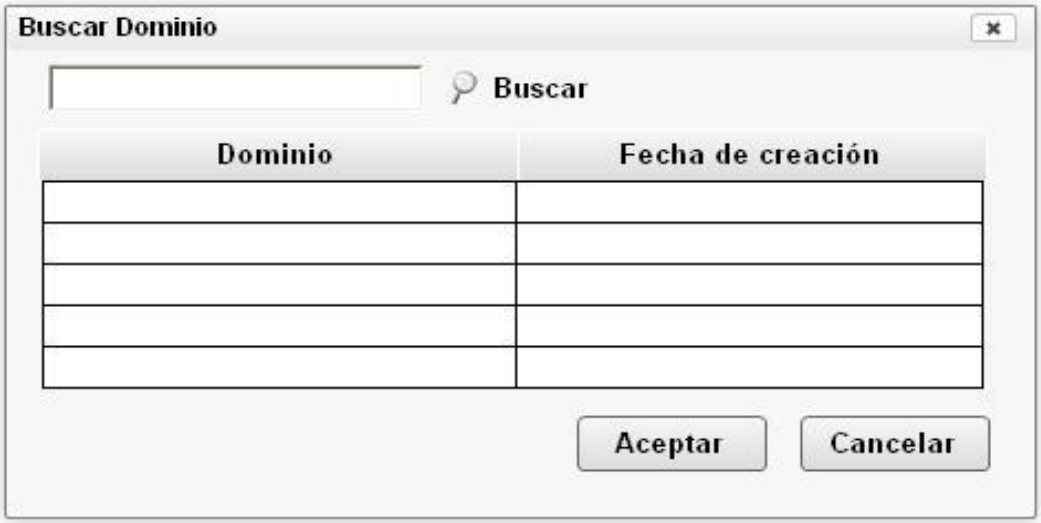

Fig. 4 Prototipo de interfaz de usuario. Buscar y Visualizar Dominios.

2.5.5 Requisito funcional R5.5 Asignar Permisos a Roles

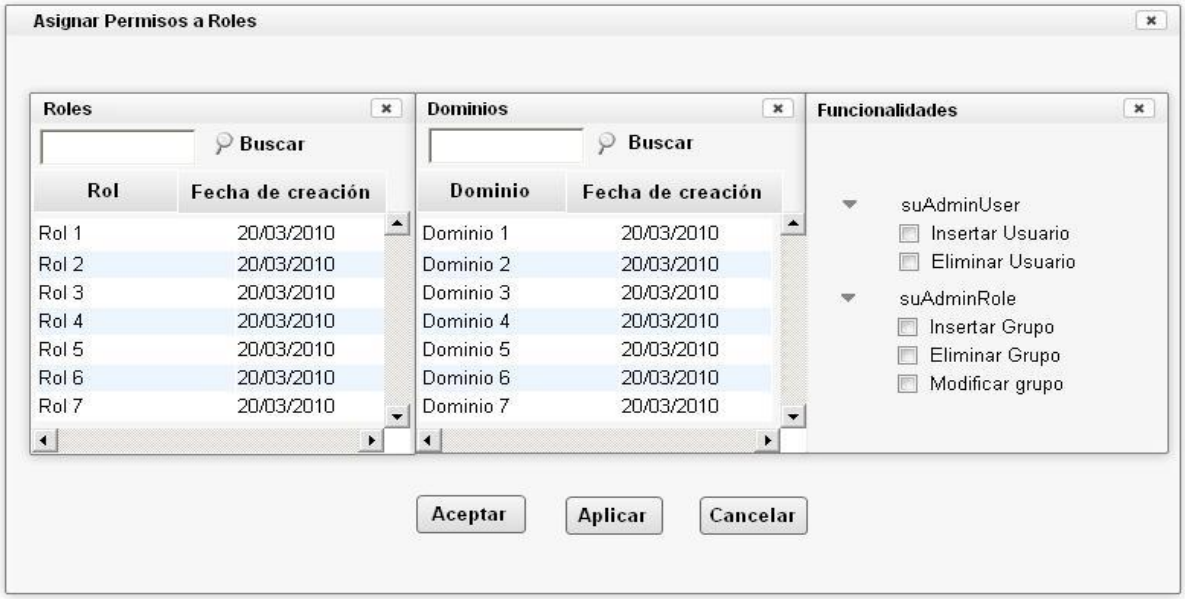

Fig. 5 Prototipo de interfaz de usuario. Asignar Permisos a Roles

2.5.6 Requisito funcional R6.5 Asignar Roles a Usuario

Fig. 6 Prototipo de interfaz de usuario. Asignar Roles a Usuarios.

2.6 Diseño de clases

En este epígrafe se presenta el diseño de las clases persistentes del sistema, sirve como una primera aproximación al diseño definitivo del modelo de datos.

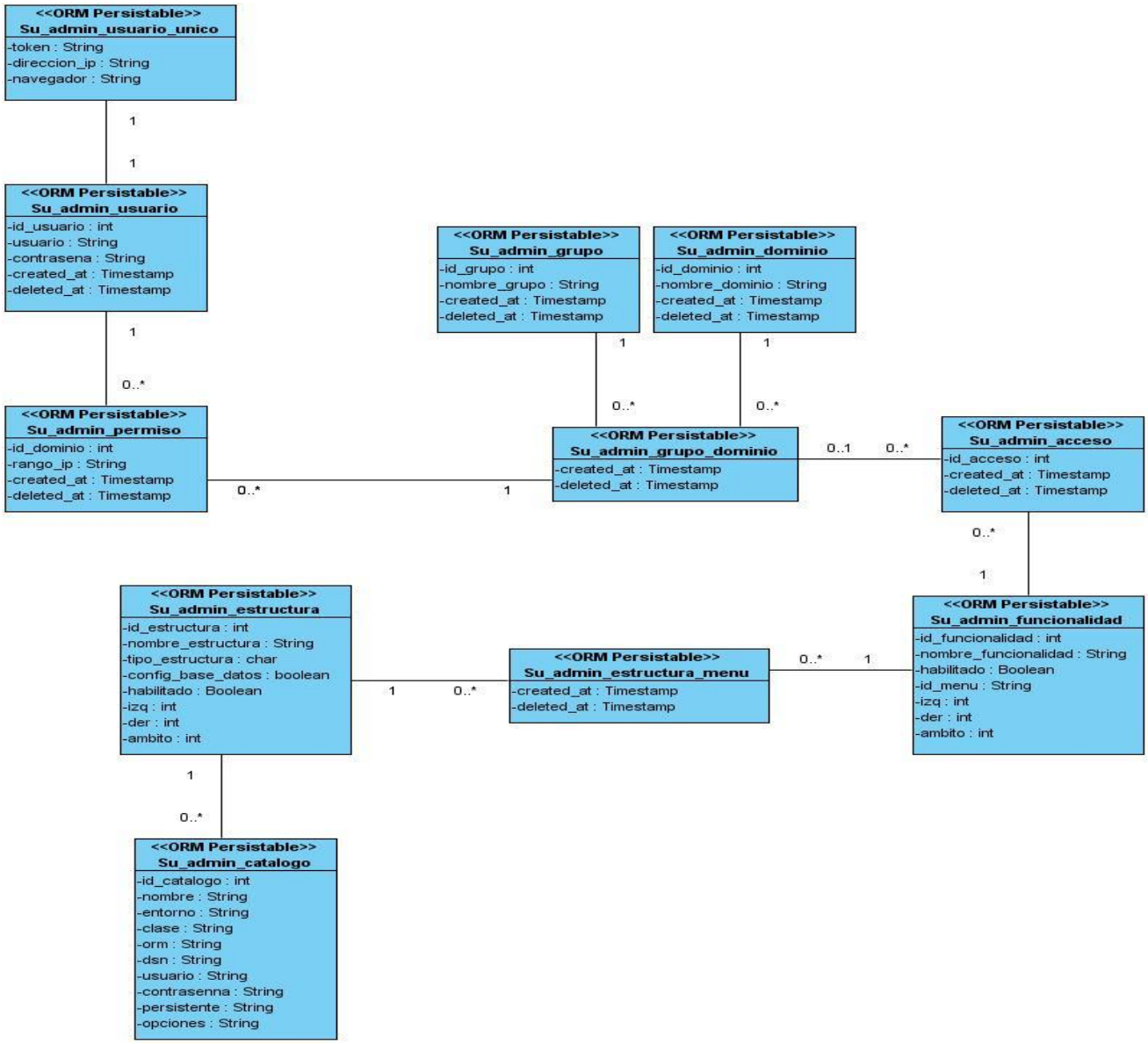

Fig. 7 Diagrama de clases

2.7 Descripción de las clases

A continuación de describe cada una de las clases que intervienen en el diagrama de clases persistentes.

Descripción de la clase suAdminEstructura

Nombre de la clase		suAdminEstructura
Estereotipo:		<<ORM Persistable>>
Descripción:		Esta clase es la encargada de manipular las estructuras de un proyecto, dígase Proyecto, Aplicación, Módulo y Acción.
Tabla correspondiente:		su_admin_estrutura
Atributos		
Nombre	**Tipo**	**Descripción**
id_estructura	Integer	Identificador de la estructura correspondiente
nombre_estructura	String	Denominación de la estructura
config_base_datos	Boolean	Booleano que dice si la estructura requiere de alguna configuración
habilitado	Boolean	Booleano que dice si la estructura se encuentra activa
izq	Integer	Valor que utiliza el ORM Propel para el trabajo con estructuras jerárquicas en forma de árbol
der	Integer	Valor que utiliza el ORM Propel para el trabajo con estructuras jerárquicas en forma de árbol
ambito	Integer	Valor que dice a que árbol pertenece la estructura

Tabla 1 Descripción de la clase suAdminEstructura

Descripción de la clase suAdminFuncionalidad

Nombre de la clase	suAdminFuncionalidad
Estereotipo:	<<ORM Persistable>>
Descripción:	Esta clase es la encargada de manipular las

		funcionalidades a las que pertenece cada acción
Tabla correspondiente:		su_admin_funcionalidad
Atributos		
Nombre	**Tipo**	**Descripción**
id_funcionalidad	Integer	Identificador de la estructura correspondiente
nombre_funcionalidad	string	Denominación de la estructura
habilitado	Boolean	Booleano que dice si la estructura se encuentra activa
id_menu	String	Identificador de la Interfaz que le corresponde a la funcionalidad
izq	Integer	Valor que utiliza el ORM Propel para el trabajo con estructuras jerárquicas en forma de árbol
der	Integer	Valor que utiliza el ORM Propel para el trabajo con estructuras jerárquicas en forma de árbol
ambito	Integer	Valor que dice a que árbol pertenece la estructura

Tabla 2 Descripción de la clase suAdminFuncionalidad

Descripción de la clase suAdminUsuario

Nombre de la clase		suAdminUsuario
Estereotipo:		<<ORM Persistable>>
Descripción:		Esta clase es la encargada de manipular información referente al usuario
Tabla correspondiente:		su_admin_usuario
Atributos		
Nombre	**Tipo**	**Descripción**
id_usuario	Integer	Identificador del usuario
nombre_funcionalidad	string	Denominación de la estructura
usuario	String	Nombre de usuario que será utilizado para acceder a la aplicación
contrasena	string	Contraseña de usuario

created_at	Date	Fecha en que se creó el usuario
deleted_at	Date	Fecha en que se deshabilitó el usuario

Tabla 3 Descripción de la clase suAdminUsuario

Descripción de la clase suAdminDominio

Nombre de la clase		suAdminDominio
Estereotipo:		<<ORM Persistable>>
Descripción:		Esta clase es la encargada de manipular información referente a los dominios
Tabla correspondiente:		su_admin_dominio
Atributos		
Nombre	**Tipo**	**Descripción**
id_dominio	Integer	Identificador del dominio
nombre_dominio	string	Nombre del dominio (es único)
created_at	Date	Fecha en que se creó el dominio
deleted_at	Date	Fecha en que se deshabilitó el dominio

Tabla 4 Descripción de la clase suAdminDominio

Descripción de la clase suAdminGrupo

Nombre de la clase		suAdminGrupo
Estereotipo:		<<ORM Persistable>>
Descripción:		Esta clase es la encargada de manipular información referente a los grupos (roles)
Tabla correspondiente:		su_admin_grupo
Atributos		
Nombre	**Tipo**	**Descripción**
id_grupo	Integer	Identificador del grupo
nombre_grupo	string	Nombre del grupo (es único)

created_at	Date	Fecha en que se creó el grupo
deleted_at	Date	Fecha en que se deshabilitó el grupo

Tabla 5 Descripción de la clase suAdminGrupo

Descripción de la clase suAdminPermiso

Nombre de la clase		suAdminPermiso
Estereotipo:		<<ORM Persistable>>
Descripción:		Esta clase es la encargada de manipular información referente a los permisos que tiene cada usuario a un grupo en un dominio determinado.
Tabla correspondiente:		su_admin_permiso
Atributos		
Nombre	**Tipo**	**Descripción**
id_usuario	Integer	Identificador del usuario
id_grupo	Integer	Identificador del grupo
id_dominio	Integer	Identificador del dominio
rango_ip	String	Rango en el que se pueden encontrar los IP de acceso a la aplicación
created_at	Date	Fecha de asignación del permiso
deleted_at	Date	Fecha en que se deshabilitó el permiso

Tabla 6 Descripción de la clase suAdminPermiso

Descripción de la clase suAdminAcceso

Nombre de la clase	suAdminAcceso
Estereotipo:	<<ORM Persistable>>
Descripción:	Esta clase es la encargada de manipular información referente a los accesos que tiene cada grupo en un dominio determinado a las funcionalidades del sistema.
Tabla correspondiente:	su_admin_acceso

Atributos		
Nombre	**Tipo**	**Descripción**
id_acceso	Integer	Identificador del acceso correspondiente
id_grupo	Integer	Identificador del grupo
id_dominio	Integer	Identificador del dominio
id_funcionalidad	String	Identificador de la funcionalidad
created_at	Date	Fecha de asignación del acceso
deleted_at	Date	Fecha en que se deshabilitó el acceso

Tabla 7 Descripción de la clase suAdminAcceso

Descripción de la clase suAdminGrupoDominio

Nombre de la clase		suAdminGrupoDominio
Estereotipo:		<<ORM Persistable>>
Descripción:		Esta clase es la encargada de manipular información referente a la relación que existe entre un grupo y un dominio.
Tabla correspondiente:		su_admin_grupo_dominio
Atributos		
Nombre	**Tipo**	**Descripción**
id_grupo	Integer	Identificador del grupo
id_dominio	Integer	Identificador del dominio
created_at	Date	Fecha de asignación del permiso
deleted_at	Date	Fecha en que se deshabilitó el permiso

Tabla 8 Descripción de la clase suAdminGrupoDominio

Descripción de la clase suAdminUsuarioUnico

Nombre de la clase		suAdminUsuarioUnico
Estereotipo:		<<ORM Persistable>>
Descripción:		Esta clase es la encargada de manipular información referente al navegador e IP que utiliza el usuario
Tabla correspondiente:		su_admin_usuario_unico
Atributos		
Nombre	**Tipo**	**Descripción**
id_usuario	Integer	Identificador del usuario
token	Integer	Token de seguridad para el usuario
dirección_ip	String	Dirección IP desde la cual puede acceder el usuario
navegador	String	Navegador con el que accede el usuario

Tabla 9 Descripción de la clase suAdminUsuarioUnico

2.8 Diagrama de Paquetes

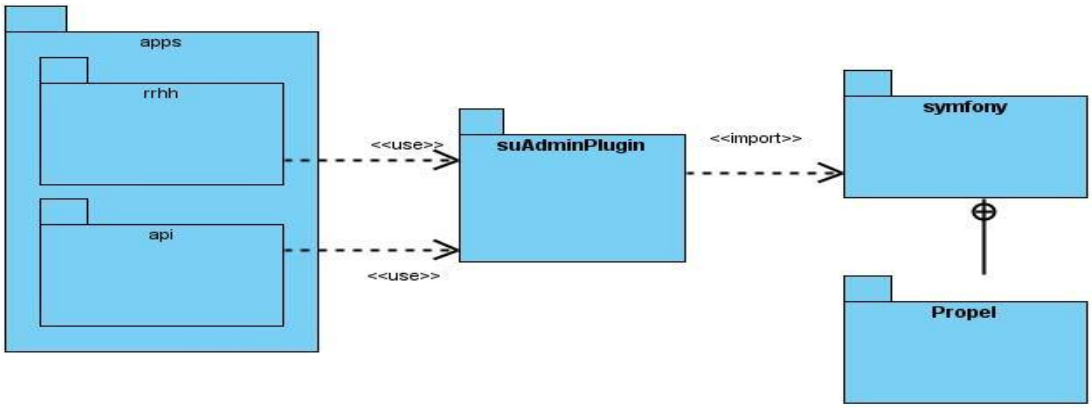

Fig. 8 Diagrama de paquetes

Todas las aplicaciones que se encuentren en el mismo proyecto utilizarán las funcionalidades del componente de seguridad suAdminPlugin.

2.9 Diseño del modelo de datos

A continuación se muestra el diagrama de entidad – relación correspondiente al componente de seguridad suAdminPlugin.

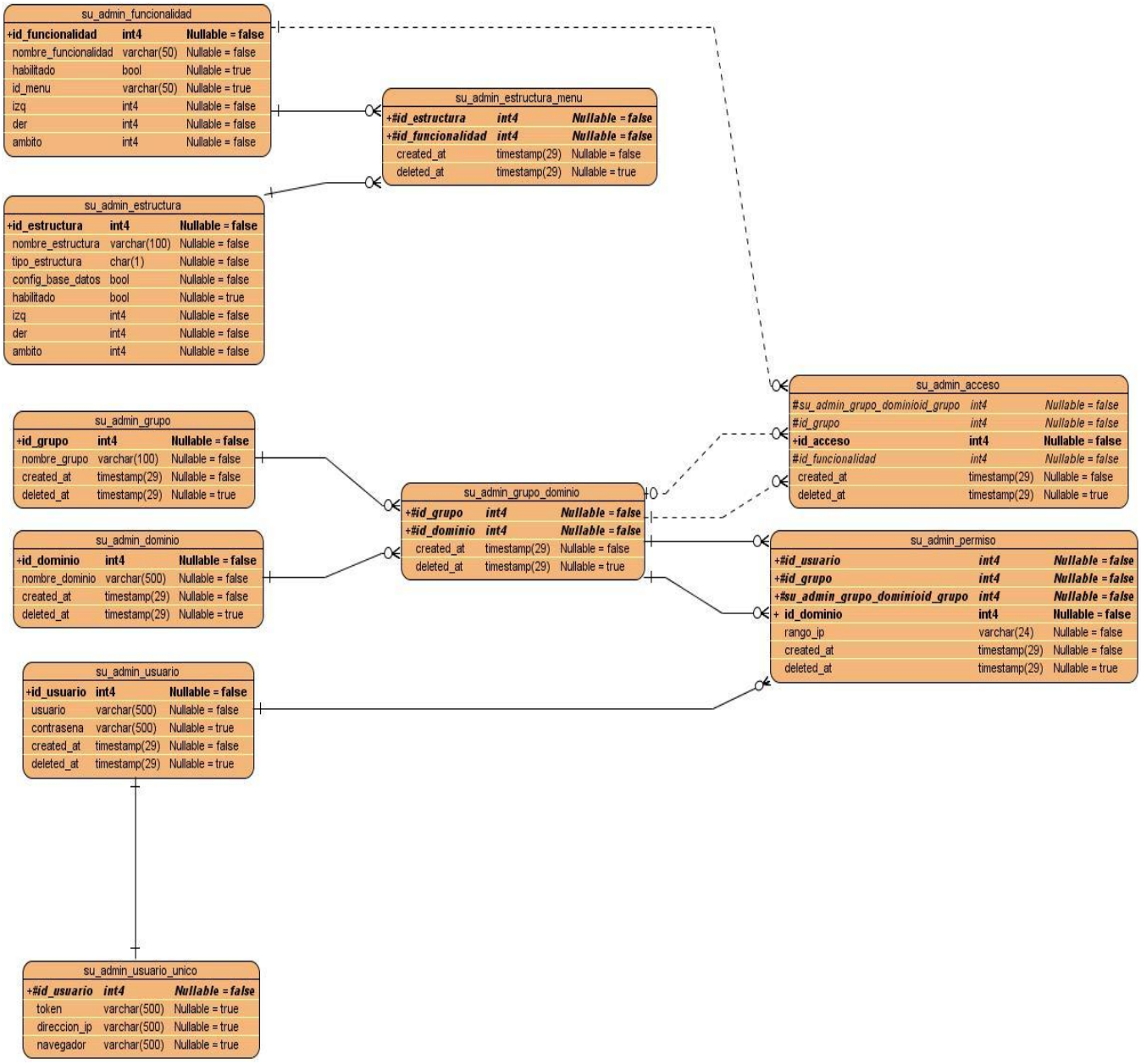

Fig. 9 Diagrama entidad relación del componente suAdminPlugin

2.10 Diagramas de clases del diseño con estereotipos Web

A continuación se muestran los diagramas de clases del diseño con estereotipos Web, para una mejor comprensión del funcionamiento del componente.

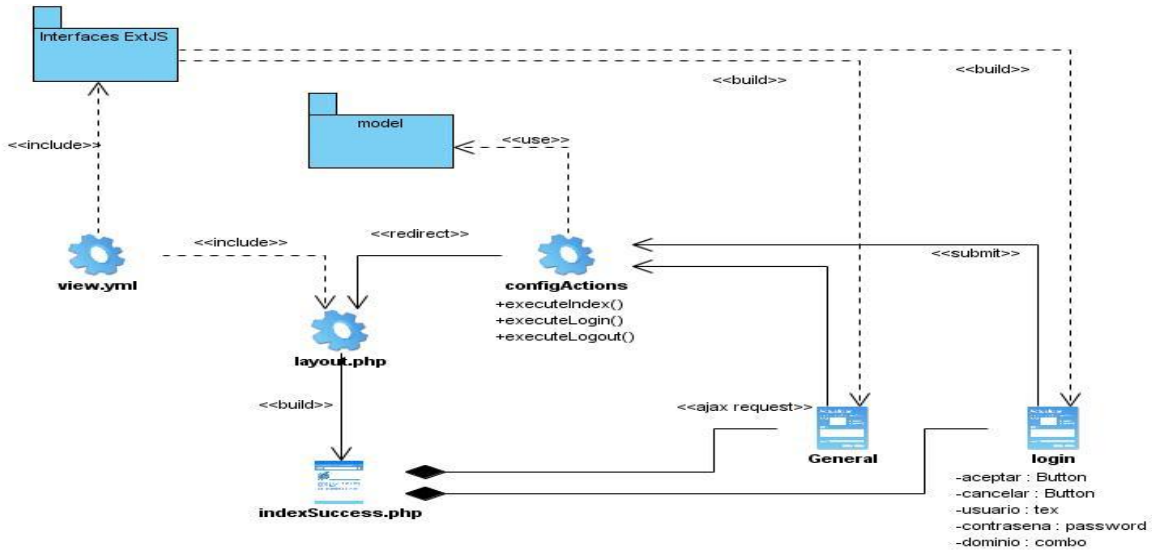

Fig. 10 Diagrama de clases del diseño con estereotipos Web. Módulo suAdminAuth

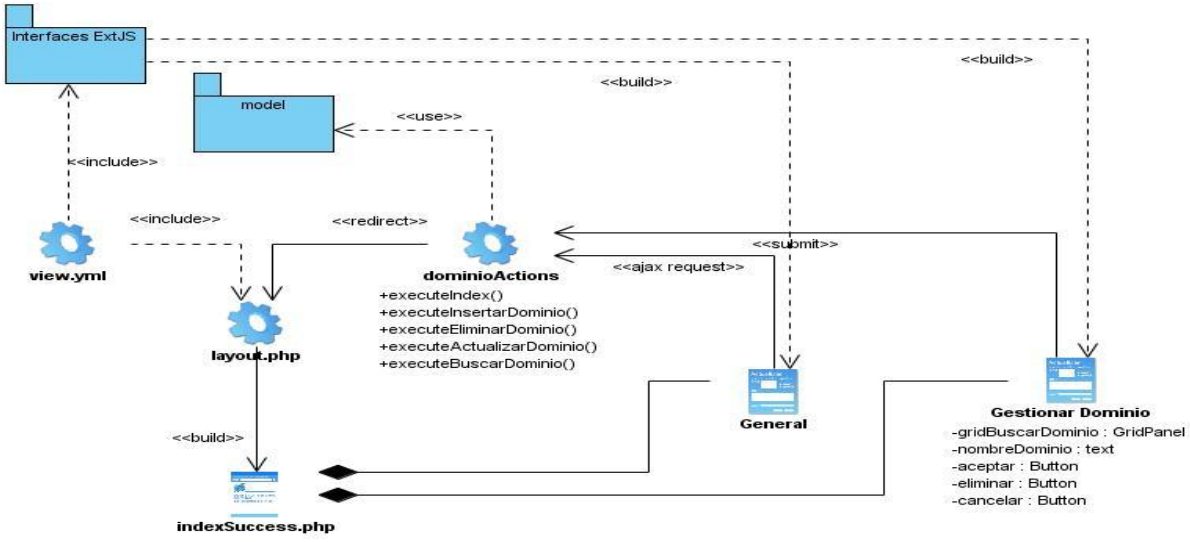

Fig. 11 Diagrama de clases del diseño con estereotipos Web. Módulo suAdminDomain

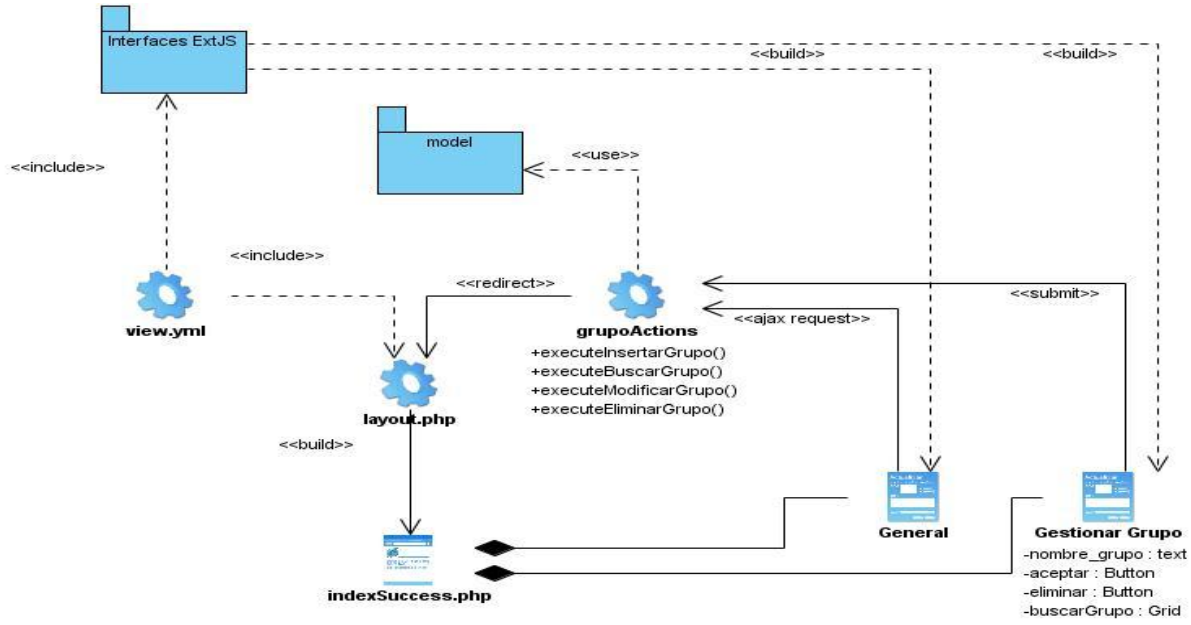

Fig. 12 Diagrama de clases del diseño con estereotipos Web. Módulo suAdminRole

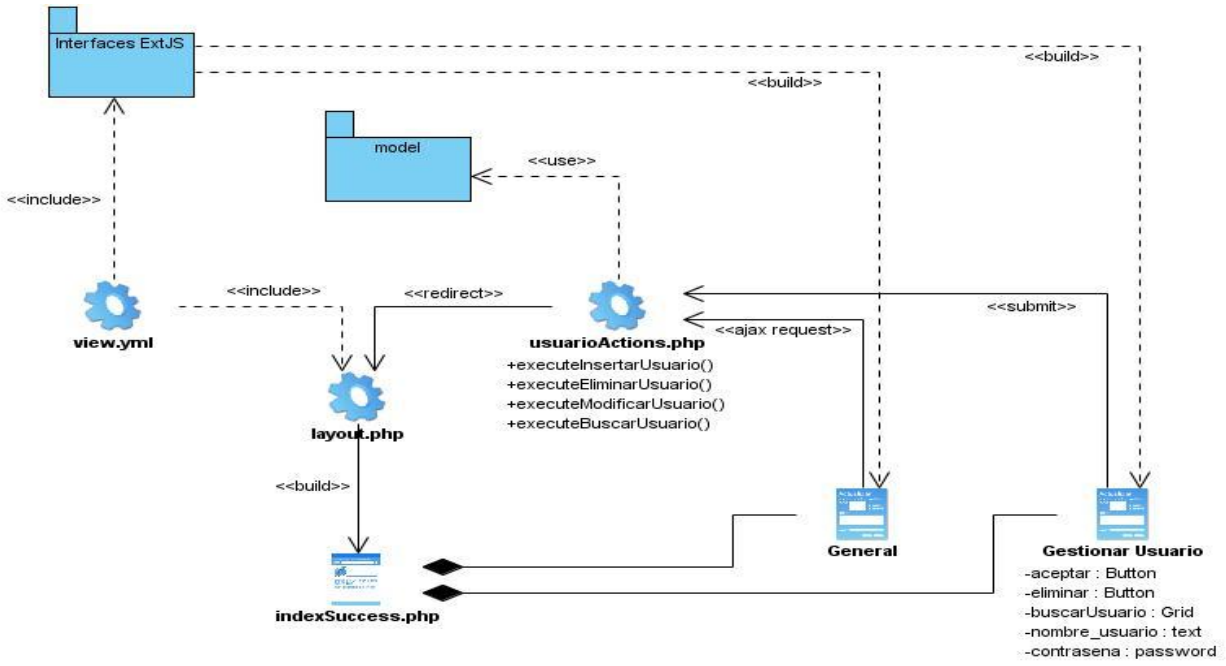

Fig. 13 Diagrama de clases del diseño con estereotipos Web. Módulo suAdminUser

2.11 Diagramas de secuencia

En este epígrafe se muestran los diagramas de secuencia de dos de los requisitos más significativos.

Requisito Funcional R6.5 Asignar Permisos a Roles

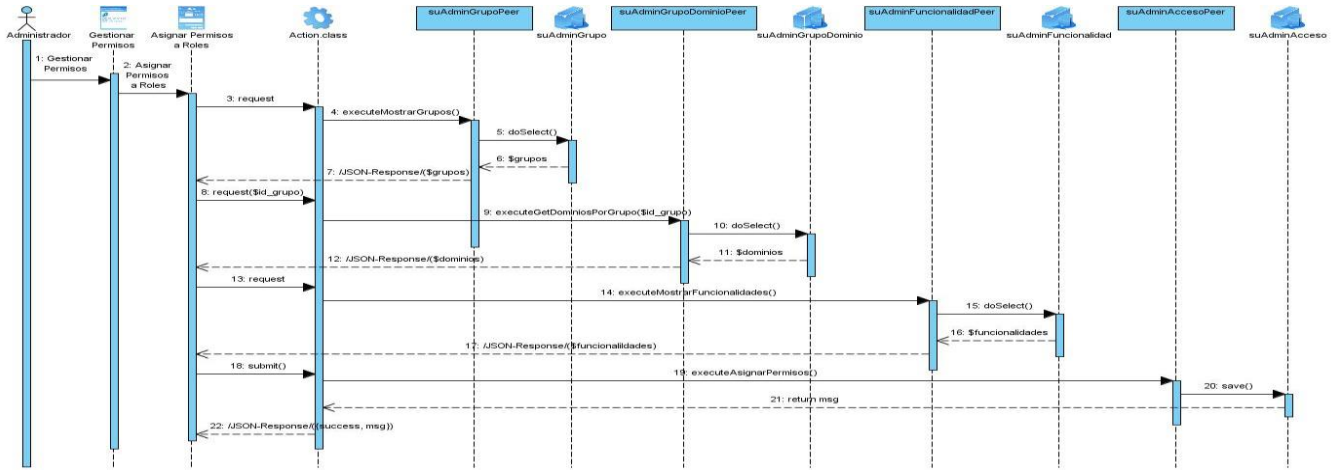

Fig. 14 Diagrama de secuencia. Asignar Permisos a Roles

Requisito Funcional R6.5 Asignar Roles a Usuarios

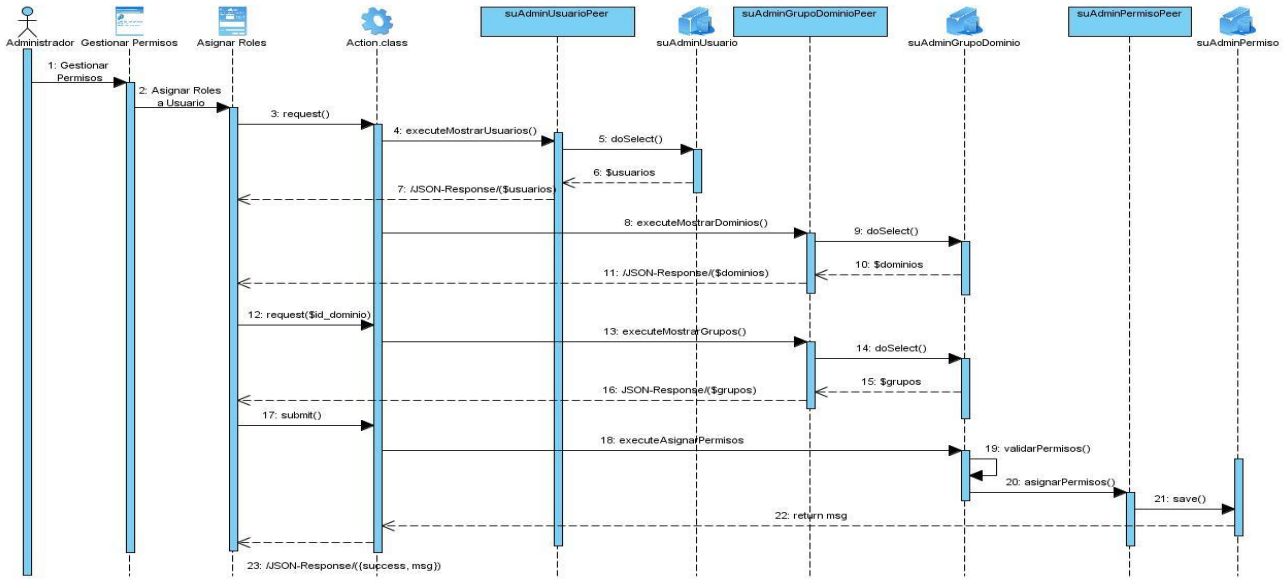

Fig. 15 Diagrama de secuencia. Asignar Roles a Usuarios

2.12 Patrones de diseño utilizados en la solución

Los patrones de diseño de software son soluciones reutilizables de problemas recurrentes que aparecen durante el proceso de diseño de software orientado a objetos.

¿Por qué surgen los patrones de diseño? Por la necesidad de transmitir la experiencia.

Con el conocimiento de estos patrones, los programadores expertos son capaces de identificar las situaciones en las que éstos tienen aplicación, y utilizarlos sin tener que detenerse para analizar el problema y vislumbrar diferentes estrategias de resolución.

A continuación se describen algunos de los patrones de diseño utilizados en la solución planteada.

2.12.1 Fachada

La utilización de este patrón está dada porque todas las configuraciones del plugin se encuentran en el directorio config. Estos archivos yml pueden ser accedidos por el componente desde cualquier lugar del mismo.

2.12.2 Singleton

A través de este patrón se accede, en la clase suAdminFilter, a la instancia del contexto, mediante el método $this->getContext (), que permite acceder a cualquier clase del núcleo de Symfony. En este caso a la sesión del usuario utilizando una instancia de sfUser ($this->getContext () ->getUser ()) y a la petición mediante el método $this-> getContext () ->getController () -> getRequest ().

2.12.3 Bajo acoplamiento

Las funcionalidades que pueden ser reutilizadas desde cualquier parte del componente, son implementadas de forma tal que el acoplamiento con otras clases sea mínimo. Tal es el caso de la clase suAdminUtil, que contendrá las funcionalidades de carga automática de los módulos del plugin por las aplicaciones. Esta clase se encuentra en el directorio lib/útil del componente.

2.12.4 Controlador

Cada módulo tiene una clase controladora llamada action.class.php que es la encargada de atender todas las peticiones y pasar los datos de la misma a las clases del modelo para su procesamiento. Al mismo tiempo es la clase que se encarga de enviar las respuestas a la vista.

2.12.5 Alta cohesión

Cada clase que pertenece al modelo tiene como responsabilidad fundamental realizar las labores que solo le competen a ella y que no son desempeñadas por otros elementos del diseño dentro del plugin. Dentro de esas labores se encuentra crear las consultas de acceso a los datos. Todas las clases están agrupadas por las funcionalidades que realizan.

2.12.6 Experto

Cada clase dentro del componente tiene la responsabilidad de utilizar únicamente la información que ella misma posee para realizar la labor para la que fue concebida. Tal es el caso de las clases del modelo que son las encargadas de toda la lógica del acceso a los datos, un ejemplo es SuAdminUserPeer.php, es la clase que se encarga de insertar, eliminar y modificar usuarios.

2.13 Conclusiones

Durante todo el capítulo se describieron los principales aspectos que se llevan a cabo durante el análisis y el diseño de la solución planteada, comenzando por el levantamiento de requisitos funcionales que contiene cada uno de los módulos correspondientes al sistema de autenticación y control de acceso. Además de una breve descripción de los requisitos no funcionales necesarios para el buen funcionamiento del sistema.

Se mostraron además los prototipos de interfaz de usuario correspondientes a cada uno de los requisitos, el diseño de clases y el modelo de datos correspondientes al sistema.

Capítulo 3: Implementación de la solución

3.1 Introducción

En este capítulo se describen los artefactos que se generan durante la implementación del sistema. Se muestra el resultado de la implementación de los componentes que utiliza la aplicación y los estándares de codificación utilizados para el desarrollo del mismo.

3.2 Diagramas de componentes

Un componente de software es una parte física de un sistema, y se encuentra en la computadora, no en la mente del analista. Ejemplos de componentes son tablas, archivos de datos, ejecutables, bibliotecas de vínculos dinámicos, documentos.

El símbolo principal de un diagrama de componentes es un rectángulo que tiene otros dos sobrepuestos en su lado izquierdo, con el nombre del componente dentro del rectángulo más grande, como se muestra en la siguiente figura:

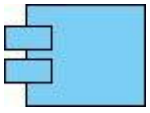

Fig. 16 Estereotipo de componente

3.2.1 Módulo suAdminAuth

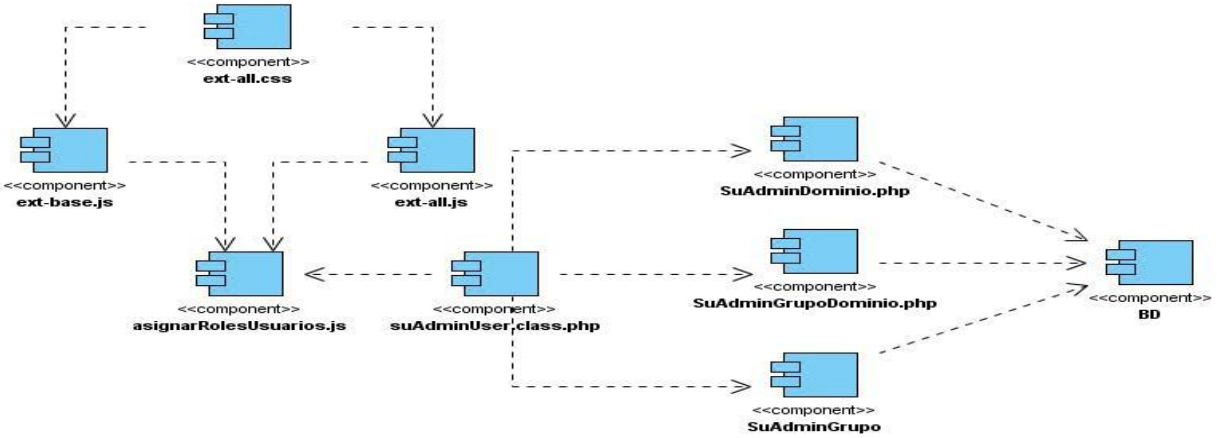

Fig. 17 Diagrama de componentes. Módulo suAdminAuth

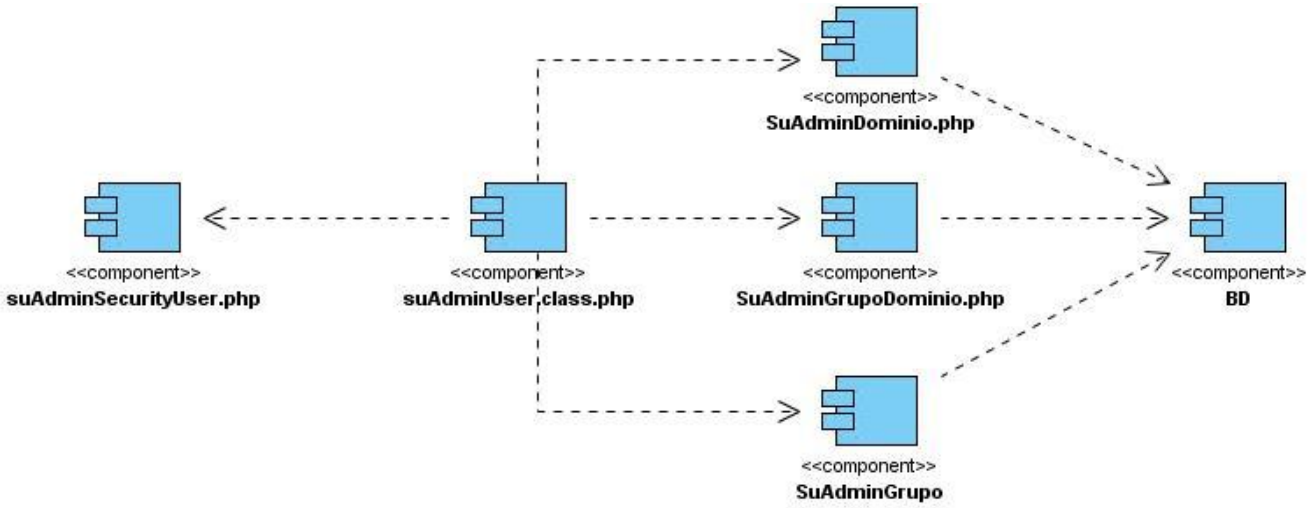

Fig. 18 Diagrama de componentes. Módulo suAdminAuth. Comprobar acceso a acción.

3.2.2 Módulo suAdminDomain

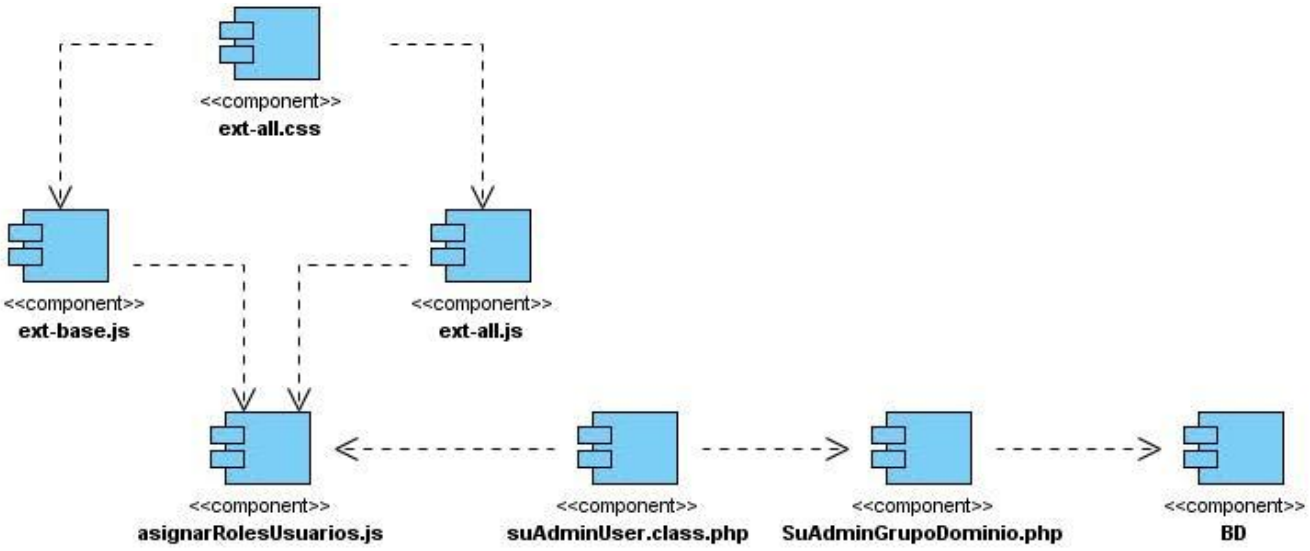

Fig. 19 Diagrama de componentes. Módulo suAdminDomain

3.2.3 Módulo suAdminRole

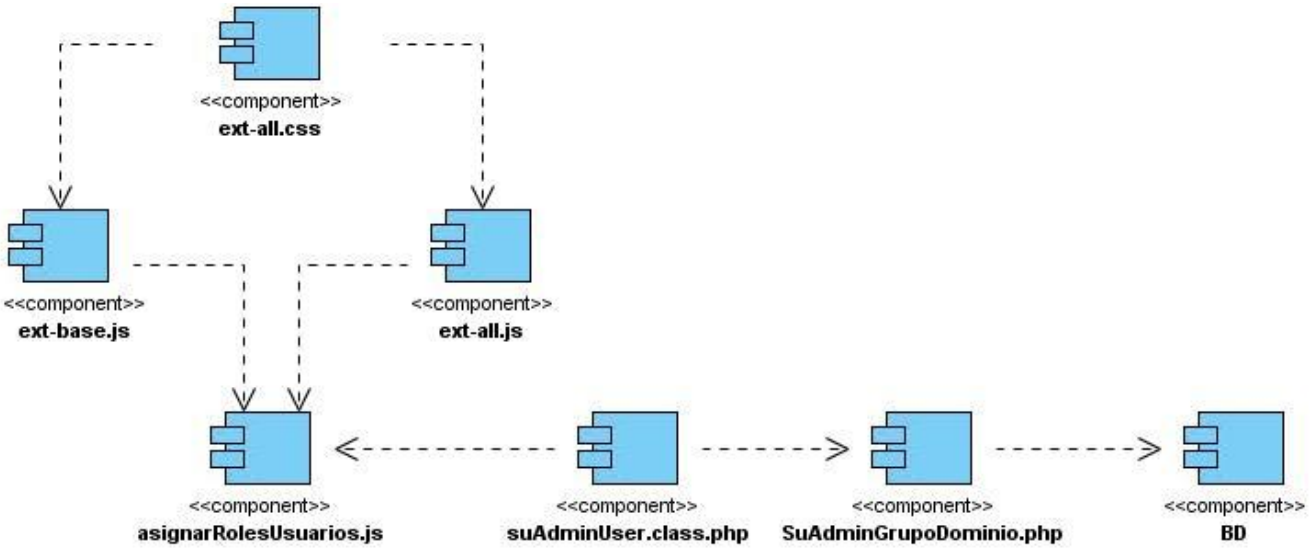

Fig. 20 Diagrama de componentes. Módulo suAdminRole

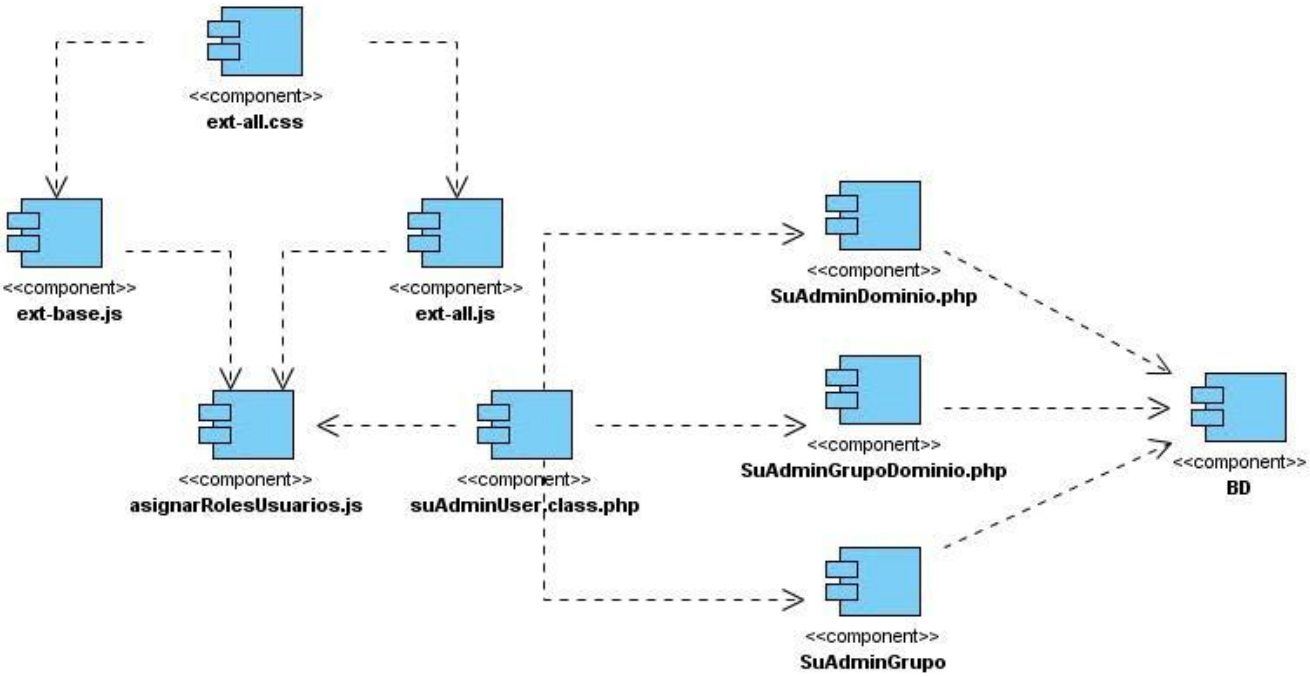

Fig. 21 Diagrama de componentes. Módulo suAdminRole. Asignar Permisos a Roles

3.2.4 Módulo suAdminUser

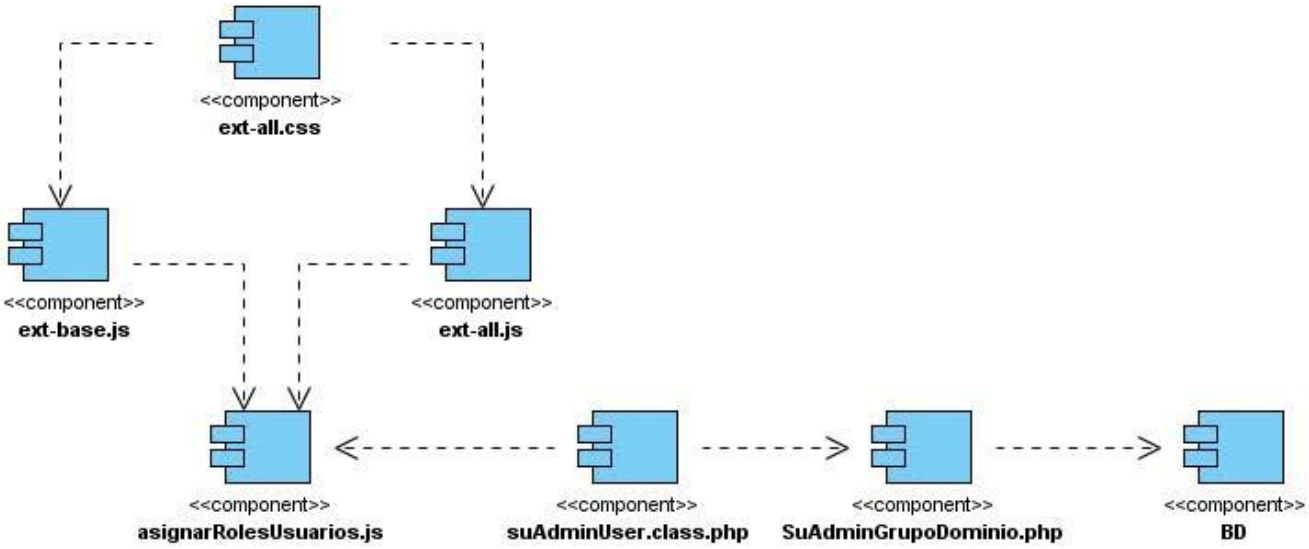

Fig. 22 Diagrama de componentes. Módulo suAdminUser

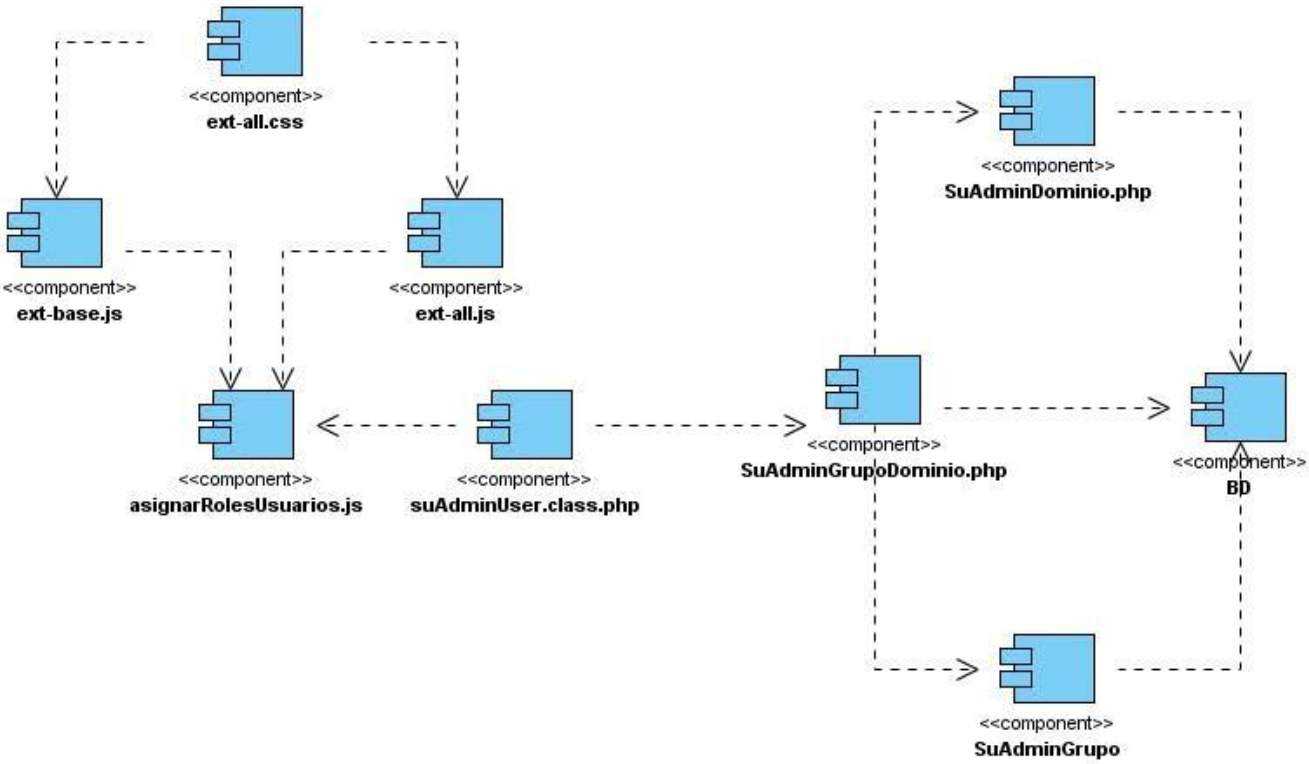

Fig. 23 Diagrama de componentes. Módulo suAdminUser. Asignar Roles a Usuarios.

3.3 Aspectos relevantes de la implementación

3.3.1 Tareas

Una tarea propia es una clase que hereda de sfBaseTask y cuyo código se encuentra en el directorio lib/task del plugin. Para el desarrollo de la aplicación se hizo necesaria la implementación de dos tareas que se describen a continuación:

suAdminEstructuraProyectoTask.class.php: Esta tarea es la encargada de leer la estructura del proyecto que se encuentra en un archivo YAML y guardarla en la base de datos. Además verifica si ya la estructura existe, en caso de ser así, la carga y sino la escribe en la base de datos.

suAdminEstructuraMenuTask.class.php: Esta es la clase encargada de asignar a cada funcionalidad la acción correspondiente en la estructura de las aplicaciones. Una funcionalidad se refiere a una agrupación de requisitos y puede incluir varias acciones. Esta tarea es muy importante en la posterior creación dinámica del menú correspondiente a los permisos que tiene cada rol en un dominio específico.

3.3.2 Filtros

El mecanismo de seguridad puede ser entendido como un filtro, por el que deben pasar todas las peticiones antes de ejecutar una acción. Según las comprobaciones realizadas por el filtro, se puede modificar el procesamiento de la petición. En este caso se implementó un filtro, que a continuación se describe.

suAdminFilter: Esta es la clase encargada de verificar si el usuario que está realizando una petición está autenticado o no, en caso de no estarlo será enviado a la página de login. Este filtro verifica que el usuario no intente acceder por la URL sin tener permisos para realizar la acción que solicita.

3.3.3 Sesiones

Symfony maneja automáticamente las sesiones del usuario y es capaz de almacenar datos de forma persistente entre peticiones. Utiliza el mecanismo de manejo de sesiones incluido en PHP y lo mejora para hacerlo más configurable y más fácil de usar. A partir de lo expuesto se creó la siguiente clase:

suAdminBasicSecurityUser.class.php: Es la clase que se encarga de manejar todo lo referente a la sesión del usuario. Maneja todos los atributos que se desean controlar en la sesión como son los

dominios a lo que el usuario tiene acceso, los roles que tiene y la aduana en la que se encuentra registrado en ese momento, además de otros parámetros necesarios para la interacción del usuario.

3.4 Conclusiones

En este capítulo se hizo un resumen de todos los artefactos que se generaron durante la implementación del sistema. Se mostraron los diagramas de componentes correspondientes a los requisitos funcionales de cada módulo. Además de describir el estándar de codificación propuesto por Symfony y el estándar para el diseño de la base de datos.

Conclusiones generales

Se considera que se ha dado cumplimiento a todos los objetivos propuestos para la investigación, quedando como resultado el sistema de control de acceso y autenticación. Para esto se llevaron a cabo una serie de tareas que tributaron al resultado final.

Durante el desarrollo del trabajo se hizo un estudio de algunos de los diferentes frameworks de PHP, se analizaron sus características en cuando a como implementan la seguridad de sus aplicaciones; además de investigar sobre aplicaciones que se desarrollan en la Universidad de las Ciencias Informáticas.

Se hizo el levantamiento de los requerimientos del sistema y se generaron todos los artefactos y descripciones correspondientes al flujo de trabajo de análisis y diseño. Se crearon los prototipos de interfaz de usuario, el diseño de las clases persistentes, el modelo de datos y los diagramas de secuencia, paquetes y clases del diseño con estereotipos web de cada requisito funcional. Se analizaron también los patrones de diseño utilizados en la solución planteada. Además se diseñaron todos los diagramas implementación que dieron paso a la programación de los componentes que forman parte de la aplicación.

El desarrollo del componente suAdminPlugin aportará mayor seguridad a las aplicaciones que se encuentren bajo la arquitectura de Symfony, y específicamente a las implementadas por el Departamento de Soluciones para la Aduana, lo que se traduce en un mayor control de las acciones que pueden realizar los usuarios sobre los sistemas que utilicen el componente.

Glosario de términos

ACL: Listas de control de Acceso. (Access Control Lists).

AGR: Aduana General de la República.

CADI: Centro de Automatización para la Información y la Dirección.

CASE: Ingeniería de Software Asistida por Computadora (Computer Aided Software Engineering).

Filtros: Son clases que contienen métodos que se ejecutan cada vez que se genera una petición.

GINA: Gestión Integral Aduanera.

HUB: Término en inglés con el que se denomina al concentrador, es un dispositivo que se usa como punto de conexión entre los componentes de una red de área local.

IDE: Entorno Integrado de Desarrollo (Integrated Development Environment).

Namespaces: Un namespace es un contexto en el que un grupo de uno o más identificadores pueden existir. Un identificador definido en un namespace está asociado con ese namespace.

Plugin: Pequeño programa que añade alguna función a otro programa, habitualmente de mayor tamaño. Un programa puede tener uno o más conectores. Son muy utilizados en los programas navegadores para ampliar sus funcionalidades.

SUA: Sistema Único de Aduanas.

TI: Tecnologías de la Información.

URI: (Uniform Resource Identifiers). Identificador de Recurso Uniforme. Los URI son cadenas que funcionan como identificadores globales que hacen referencia a recursos en la Web tales como documentos, imágenes, archivos descargables, servicios, buzones de correo electrónico y otros.

URL: Localizador de Recurso Uniforme. (Uniform Resource Locator), Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

XSS: (Cross-Site scripting) es un ataque basado en la explotación de vulnerabilidades del sistema de validación de HTML incrustado.

Recomendaciones

Después de darle cumplimiento a los objetivos propuestos y de implementar el sistema, se recomienda:

- Continuar profundizando en el tema de la seguridad en aplicaciones Web.
- Seguir desarrollando el sistema de autenticación y control de acceso en el Departamento de Soluciones para la Aduana.
- Desarrollar módulos de auditoría que permitan tener un registro de las operaciones realizadas en las aplicaciones que se encuentren utilizando el sistema de autenticación y control de acceso.

Referencias bibliográficas

1. INSTITUTE, I. G. COBIT 4.0. 2005, Disponible en: www.itgi.org. ISBN 1-933284-37-4.
2. CORPORATION, M. Microsoft TechNet [Consultado el: 24 de enero de 2010]. Disponible en: http://www.microsoft.com/latam/technet/seguridad/default.aspx.
3. GERBER, M.; SOLMS, R. V., et al. Formalizing information security requirements. 2001 vol. 9, Disponible en: http://www.emeraldinsight.com/10.1108/09685220110366768. ISBN 0968-5227.
4. GUTIÉRREZ, C.; FERNÁNDEZ-MEDINA, E., et al. A Survey of Web Services Security. Springer Berlin / Heidelberg, 2004, vol. 3043/2004, Disponible en: http://www.springerlink.com/. ISBN 978-3-540-22054-1.
5. LÓPEZ, M. J. L. Criptografía y Seguridad en Computadores. 3ra ed. 2003
6. MENEZES, A. J.; OORSCHOT, P. C. V., et al. Handbook of Applied Cryptography. 1996, Disponible en: http://www.cacr.math.uwaterloo.ca/hac/. ISBN 0-8493-8523-7.
7. ADAMS, C. y LLOYD, S. Understanding PKI. 2nd ed. Boston: Pearson Education, Inc, 2003, ISBN 0-672-32391-5.
8. CLERCQ, J. D. Single Sign-On Architectures. En Infrastructure Security. Springer Berlin / Heidelberg, 2002, vol. 2437/2002, p. 40-58.
9. STELL, A. J.; SINNOTT, D. R. O., et al. Comparison of Advanced Authorisation Infrastructures for Grid Computing. 1ra ed. IEEE, 2005, Disponible en: http://eprints.gla.ac.uk/3560/. ISBN 1550-5243.
10. ARGENTINA, C. C. N. CCN -CERT Argentina Disponible en: https://www.ccn-cert.cni.es/.
11. YUAN, E. y TONG, J. Attributed Based Access Control (ABAC) for Web Services. En IEEE International Conference on Web Services (ICWS'05). 2005.
12. SANDHU, R. S. Lattice-Based Access Control Models IEEE, 1993, ISBN 0018-9162/93/1100-0009.
13. LTD, Z. T. Zend Framework Disponible en: http://framework.zend.com/manual/en/.
14. RIGAZZI, P. Zend Framework Playground [Consultado el: 22 de enero de 2010]. Disponible en: http://spanish.zendfw.com/.
15. GUTIERREZ, A. F.; TEJEDA, D., et al. Kumbia PHP Framework. 2009,
16. CORPORATION, C. D. CakeDC Disponible en: http://cakedc.com/.
17. ELLISLAB, I. CodeIgniter Disponible en: http://www.codeigniter.com.
18. POTENCIER, F. Symfony. La guía definitiva. 2008, Disponible en: http://www.symfony.es/.
19. LTD, A. K. S. Aladdin Disponible en: http://www.aladdin.es/.

20. *Sitio oficial del Visual Paradigm. Disponible en: http://www.visual-paradigm.com.*

21. *NetBeans IDE 6.8 Release Information. Disponible en: http://netbeans.org/community/releases/68/.*

22. *CHAVES, M. A. La Ingeniería de Requerimientos y su importancia en el desarrollo de proyectos de software. Costa Rica: 2005, vol. VI, ISBN 1409-4746.*

Anexos

Anexo 1: Descripción de los requisitos funcionales del componente suAdminPlugin

Autenticar usuario

Precondiciones	No aplica.	
Flujo de eventos		
Flujo básico Autenticar Usuario		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción Autenticar Usuario.	
2	Introduce las datos necesarios para ser autenticado en el sistema: • Usuario • Contraseña • Dominio	
3		Comprueba que las credenciales son correctas. En caso contrario ver *Flujo Alterno 1.a Credenciales incorrectas.*
4		Carga las acciones a las que el usuario tiene acceso.
5		Ejecuta el requisito *Construir Menú de Navegación,*
6		Concluye la ejecución del requisito.
Pos-condiciones		
1	Se tiene acceso a los dominios que se definen por defecto.	
Flujos alternativos		
Flujo alternativo 1.a Credenciales incorrectas.		
No	**Actor**	**Sistema**
1		Muestra un mensaje de error.
2		Vuelve al paso 1.
Pos-condiciones		
1	El usuario es informado de que cometió un error.	
Relaciones	**Requisitos Incluidos**	Paso 5: Construir menú de navegación.

Comprobar acceso a acción

Precondiciones	El usuario debe existir en el sistema. El usuario debe estar autenticado.	
Flujo de eventos		
Flujo básico Comprobar Acceso a Acción		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción Comprobar Acceso a Acción.	

2	Accede a una acción en el sistema. Una acción puede ser cualquier funcionalidad que permita realizar el sistema.	
3		Verifica si el usuario tiene los permisos para acceder a la acción accedida. En caso contrario ver *Flujo Alterno 2.a: El usuario no tiene Los permisos necesarios.*
4		Concluye la ejecución del requisito.
Pos-condiciones		
1	Se determina el acceso a la acción.	
Flujos alternativos		
Flujo alternativo 2.a El usuario no tiene los permisos necesarios.		
No	**Actor**	**Sistema**
1		Redirecciona a la página de autenticación.
2		Concluye la ejecución del requisito.
Pos-condiciones		
1	El usuario puede acceder nuevamente a la página de de autenticación.	

Insertar usuario

Precondiciones	El usuario encargado de la inserción debe tener los permisos necesarios.
Flujo de eventos	
Flujo básico Insertar Usuario	

No	Actor	Sistema
1	Selecciona en el menú la opción Insertar Usuario.	
2	Ingresa los datos del usuario. • Nombre • Contraseña	
3		Valida la completitud de los datos. Si falta por llenar algún campo ver *Flujo Alterno 2.a: Datos Incompletos.*
4		Verifica que no exista un usuario con los mismos datos. En caso contrario ver *Flujo Alterno 4.a: Existe un usuario con los mismos datos*
5		Inserta el usuario.
6		Concluye la ejecución del requisito
Pos-condiciones		
1	Es insertado un usuario en la BD correctamente.	
Flujos alternativos		
Flujo alternativo 2.a Datos incompletos.		
No	**Actor**	**Sistema**

1		Muestra un mensaje de error informando que faltan datos por introducir.
2		Vuelve al paso 2 del flujo básico.
Pos-condiciones		
1	El usuario es informado de que cometió un error.	
Flujos alternativos		
Flujo alternativo 4.a Existe un usuario con los mismos datos		
No	**Actor**	**Sistema**
1		Muestra un mensaje de error informando que ya existe ese usuario.
2		Concluye la ejecución del requisito
Pos-condiciones		
1	El usuario es informado de que cometió un error.	
Validaciones		
1	Que no exista el nombre de usuario en la BD.	

Buscar y Visualizar usuario

Precondiciones	El usuario que va a realizar la búsqueda tiene que estar autenticado.	
Flujo de eventos		
Flujo básico Buscar y Visualizar Usuario		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción Buscar y Visualizar Usuario.	
2	Introduce el nombre de usuario por el que desea buscar.	
3		Verifica si existe el usuario seleccionado. En caso contrario ver *Flujo Alterno 3.a: No existe el usuario seleccionado.*
4		Muestra los datos específicos del usuario seleccionado.
5		Concluye la ejecución del requisito.
Pos-condiciones		
1	Realizada correctamente la búsqueda.	
Flujos alternativos		
Flujo alternativo 3.a No existe el usuario seleccionado.		
No	**Actor**	**Sistema**
1		Muestra un mensaje de error informando que el usuario seleccionado no existe.
2		Concluye la ejecución del requisito
Pos-condiciones		
1	Es mostrado un mensaje de error al usuario.	

Eliminar usuario

Precondiciones	Al menos debe existir un usuario en la BD. El usuario debe estar autenticado y con los permisos necesarios para eliminar.	
Flujo de eventos		
Flujo básico Eliminar Usuario		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción eliminar usuario.	
2		Muestra un listado con los usuarios.
3	Debe seleccionar el usuario que se desea eliminar.	
4		Deshabilita el usuario seleccionado de la BD, no se elimina completamente.
5		Concluye la ejecución del requisito.
Pos-condiciones		
1	Se elimina correctamente el usuario.	

Asignar Roles a Usuarios

Precondiciones	Que exista el dominio, los roles, la estructura y al menos un usuario para la asignación. El usuario debe estar autenticado y con los permisos necesarios para hacer la asignación.	
Flujo de eventos		
Flujo básico Asignar Roles a Usuarios		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción asignar roles a usuarios.	
2		Muestra un listado de los usuarios.
3		Ejecuta el requisito *Buscar y Visualizar Rol.*
4		Ejecuta el requisito *Buscar y Visualizar Dominio.*
5	Le asigna un rol al usuario seleccionado de la lista, perteneciente ya a un dominio (cada usuario tiene un único rol en cada dominio) especificando.	
6		Guarda los permisos en la BD.
7		Concluye la ejecución del requisito.
Pos-condiciones		
1	Quedan asignados correctamente los permisos.	
Relaciones	**Requisitos Incluidos**	Paso 3: Buscar y Visualizar Rol. Paso 4: Buscar y Visualizar Dominio.

Asignar Permisos a Roles

Precondiciones	Al menos debe existir un rol en la BD. El usuario debe estar autenticado y con los permisos necesarios para hacer la asignación.	
Flujo de eventos		
Flujo básico Asignar Permisos a Roles		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción asignar permisos a roles.	
2		Muestra los roles existentes.
3	Selecciona el rol al cual le la a asignar los permisos.	
4		Carga los dominios a los que pertenece el rol seleccionado.
5	Asigna las funcionalidades a las cuales puede acceder.	
6		Almacena los datos en la BD y así termina el requisito.
Pos-condiciones		
1	Quedan asignados correctamente los permisos.	

Insertar Rol

Precondiciones	El usuario encargado de la inserción debe tener los permisos necesarios.	
Flujo de eventos		
Flujo básico Insertar Rol		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción insertar rol.	
2	Ingresa el dato del rol, el cual debe tener un nombre único. • Rol	
3		Valida que no exista un rol con este nombre. En caso de existir un rol con ese nombre ver *Flujo Alterno 2.a: Existe un Rol con los mismos datos.*
4	Asigna las acciones a las cuales puede acceder, estas acciones pueden ser heredadas de otro rol.	
5		Almacena los datos en la BD.
6		Concluye la ejecución del requisito.
Pos-condiciones		
1	Queda insertado correctamente el rol.	

Buscar y Visualizar Rol

Precondiciones	El usuario que va a realizar la búsqueda tiene que estar autenticado.
Flujo de eventos	
Flujo básico Buscar y Visualizar Rol.	

No	Actor	Sistema
1	Selecciona en el menú la opción buscar y visualizar rol.	
2	Introduce el criterio de por el que desea buscar. • Rol	
3		Verifica si existe el rol seleccionado. En caso de no existir ver *Flujo Alterno 2.a: No existe el rol.*
4		Muestra los datos específicos del rol seleccionado.
5		Concluye la ejecución del requisito.

Pos-condiciones	
1	Realiza correctamente la búsqueda.

Flujos alternativos		
Flujo alternativo 2.a No existe el rol.		
No	**Actor**	**Sistema**
1		Muestra un mensaje de error informando que no existe ese rol.
2		Vuelve al paso 1.
Pos-condiciones		
1	Muestra mensaje de error.	

Modificar Rol

Precondiciones	Al menos debe existir un rol en la BD. El usuario debe estar autenticado y con los permisos necesarios para hacer la modificación.
Flujo de eventos	
Flujo básico Modificar Rol	

No	Actor	Sistema
1	Selecciona en el menú la opción modificar rol.	
2		Muestra todos los roles existentes.
3	Debe seleccionar el rol que se desea modificar.	
4		Muestra los datos específicos del rol seleccionado
5	Modifica los campos seleccionados. • Rol	
6		Actualiza los datos del rol.
7		Concluye la ejecución del requisito.

Pos-condiciones	
1	Se modifica correctamente el rol seleccionado.

Eliminar Rol

Precondiciones	Al menos debe existir un rol en la BD. El usuario debe estar autenticado y con los permisos necesarios para eliminar.
Flujo de eventos	
Flujo básico Eliminar Rol	

No	**Actor**	**Sistema**
1	Selecciona en el menú la opción eliminar rol.	
2		Muestra un listado con los roles
3	Debe seleccionar el rol que se desea eliminar.	
4		El sistema deshabilita el rol seleccionado de la BD.
5		Concluye la ejecución del requisito.
Pos-condiciones		
1	Queda eliminado correctamente el rol.	

Insertar Dominio

Precondiciones	El usuario encargado de la inserción debe tener los permisos necesarios.
Flujo de eventos	
Flujo básico Insertar Dominio	

No	**Actor**	**Sistema**
1	Selecciona del menú la opción insertar dominio.	
2	Ingresa el dato del dominio. • Dominio	
3		Valida que no exista un dominio con los mismos datos. En caso de existir un error ver *Flujo Alterno 2.a: Existe un dominio con los mismos datos*
4		Debe insertar el dominio.
5		Concluye la ejecución del requisito.
Pos-condiciones		
1	Queda insertado correctamente el dominio en la BD.	

Buscar y Visualizar Dominio

Precondiciones	El usuario que va a realizar la búsqueda tiene que estar autenticado.	
Flujo de eventos		
Flujo básico Buscar y Visualizar Dominio		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción buscar y visualizar dominio.	
2	Introduce el criterio de por el que desea buscar. • Dominio.	
3		Verifica si existe el dominio seleccionado. En caso contrario ver *Flujo alterno 2.a: No existe el dominio*
4		Muestra los datos específicos del usuario seleccionado.
5		Concluye el requisito.
Pos-condiciones		
1	Se realiza correctamente la búsqueda,	
Flujos alternativos		
Flujo alternativo 2.a No existe el dominio.		
No	**Actor**	**Sistema**
1		Muestra un mensaje de error informando que no existe ese dominio.
2		Vuelve al paso 1.
Pos-condiciones		
1	El usuario es informado de que cometió un error.	

Modificar Dominio

Precondiciones	Al menos debe existir un dominio en la BD. El usuario debe estar autenticado y con los permisos necesarios para hacer la modificación.	
Flujo de eventos		
Flujo básico Modificar Dominio		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción modificar dominio.	
2		Muestra todos los dominios existentes.
3	Debe seleccionar el dominio que se desea modificar.	
4		Muestra los datos específicos del dominio seleccionado.
5	Modifica los campos seleccionados. • Dominio	
6		Verifica si son correctos los datos

		introducidos. En caso de existir un error ver *Flujo Alterno 1.a Datos incorrectos*
7		Actualiza los datos del dominio.
8		Concluye la ejecución del requisito.
Pos-condiciones		
1	Es modificado el dominio seleccionado.	
Flujos alternativos		
Flujo alternativo 1.a Datos incorrectos		
No	**Actor**	**Sistema**
1		Muestra un mensaje de error informando los errores.
		Vuelve al paso 1.
Pos-condiciones		
1	El usuario es informado de que cometió un error.	

Eliminar Dominio

Precondiciones	Al menos debe existir un dominio en la BD. El usuario debe estar autenticado y con los permisos necesarios para eliminar.	
Flujo de eventos		
Flujo básico Eliminar Dominio		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción eliminar dominio.	
2		Muestra un listado con los dominios.
3	Debe seleccionar el dominio que se desea eliminar.	
4		Deshabilita el dominio seleccionado de la BD, pero no se elimina completamente.
5		Concluye la ejecución del requisito.
Pos-condiciones		
1	Queda eliminado correctamente el dominio.	

Cargar Estructura del Proyecto mediante Líneas de Comando

Precondiciones	Debe estar la estructura del proyecto cargada en un archivo.	
Flujo de eventos		
Flujo básico Cargar estructura del proyecto mediante líneas de comando.		
No	**Actor**	**Sistema**
1	Selecciona en el menú la opción Cargar estructura del proyecto mediante Líneas de Comando.	
2	Crea el archivo de configuración.	
3	Introduce las líneas de comando que van a leer el archivo.	

4		Ejecuta las líneas de comando que van a leer el archivo con la estructura del sistema.
5		Comprueba si la estructura del archivo existe en la BD. En caso contrario ver *Flujo Alterno 5.a: No existe la estructura en la BD.*
6		Comprueba que la estructura cargada sea igual a la existente en la BD. En caso contrario ver *Flujo Alterno 6.a: Estructuras diferentes.*
7		Concluye la ejecución del requisito.
Pos-condiciones		
1	Se carga correctamente la estructura.	
Flujos alternativos		
Flujo alternativo 5.a No existe la estructura en la BD.		
No	**Actor**	**Sistema**
1		Copia la estructura que se encuentra en el archivo.
2		Concluye la ejecución del requisito.
Pos-condiciones		
1	Se carga correctamente la estructura.	
Flujos alternativos		
Flujo alternativo 6.a Estructuras diferentes.		
No	**Actor**	**Sistema**
1		Se actualizan los cambios en la BD por la estructura que está en el archivo.
2		Concluye la ejecución del requisito.
Pos-condiciones		
1	Se carga correctamente la estructura.	

Construir Menú de Navegación

Precondiciones	No aplica	
Flujo de eventos		
Flujo básico Construir Menú de Navegación		
No	**Actor**	**Sistema**
	El usuario se autentica	A partir de los permisos que tiene el usuario con un rol y un dominio específicos, el sistema construye un menú con las acciones a las que tiene acceso el usuario.
Pos-condiciones		
1	Genera el menú de navegación correctamente.	