UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Centro de Informatización para la Gestión de Entidades

Versión 2.0 del componente de administración y Seguridad Aduanera de la Ventanilla Única Aduanera.

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

Autor:
Arlene Garcés González

Tutores:

Ing. Eddie Nelson Beltrán González

Ing. Leonardo D. Antúnez Naranjo

La Habana, junio de 2017
"Año 59 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser el (la) autor (a) de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____	_____
Autor	Tutor	Tutor
Arlene Garcés González	Ing. Leonardo D. Antúnez Naranjo	Ing. Eddie N. Beltrán González

DATOS DE CONTACTO

Síntesis del Tutor

Eddie Nelson Beltrán González: Ingeniero en Ciencias Informáticas. Graduado del año 2012 en la Universidad de las Ciencias Informáticas. Especialista A del Centro de Informatización para la Gestión de Entidades (CEIGE), con 5 años de experiencia en el desarrollo de aplicaciones web.

Leonardo Darell Antúnez Naranjo: Ingeniero en Ciencias Informáticas. Graduado del año 2012 en la Universidad de las Ciencias Informáticas., Especialista del Centro de Informatización para la Gestión de Entidades (CEIGE), con 5 años de experiencia en el desarrollo de aplicaciones web.

DEDICATORIA

Dedico el presente trabajo de diploma a mis abuelos que son todo para mí. A ellos que siempre fueron mi apoyo incondicional, mi sostén y mi ejemplo a seguir. A ellos que con tanto sacrificio, estoy hoy aquí en este preciso momento.

AGRADECIMIENTOS

En estos momentos los agradecimientos se hacen un poco largos, pues a lo largo de mi carrera siempre tuve personas que fueron importantes para mí. Le agradezco a toda mi familia por apoyarme y por estar siempre presente.

Les agradezco a mis abuelos por todo su sacrificio, entrega y dedicación, por hacer de mí lo que soy hoy. Por ser los mejores abuelos del Mundo. Y nunca pero nunca dejarme sola cuando más lo necesité.

Siempre le estaré agradecida a esa personita que estuvo desde el día 7 para apoyarme incondicionalmente. Por saber todos mis estados de ánimo. Por ignorarme también le agradezco, porque gracias a eso continuamos siendo novios. Por ser mi sostén, por malcriarme y complacerme en todo. A ti Yan, te agradezco el ser la persona que siempre quise para mi vida.

A mis padres Mileidys y Armando por hacer que yo existiera, y contar con ellos cuando los necesité. Aunque nunca demuestre mis sentimientos, los quiero.

Agradezco a toda mi familia y a todas aquellas personas que han pasado a formar parte de ella. Le agradezco a mi hermana por estar ahí incondicionalmente y cederme todos mis antojos. A mis tías: Maite, Mildred, Sady y Nelida. A mis tíos Camilo y Rober. También agradezco a todas mis primas y primo. A mi prima Laura, porque siempre después de una discusión de primas, siempre estuvo para apoyarme.

A toda la familia de Yan por quererme a la fuerza, por apoyarme, por dejarme probar todas mis maldades. A la abuela de Yan que fue la que más sufrió conmigo. Gracias a todos. A mi suegra decirle que no era la nuera que ella quería pero le tocó.

A mis tutores Eddy, Leo y a Isis por apoyarme y ayudarme en todo momento. Y hacer que este día todo saliera bien. Siempre les estaré agradecido.

AGRADECIMIENTOS

Clau tú también eres importante y te agradezco que me hayas tenido paciencia estos 5 años y estuvieras siempre para cuidarme, te quiero y siempre serás importante. A mis amigos Yeider y Rafael por hacer que mi familia creciera. Gracias a los caramelos de manzana de Rafa estoy graduándome hoy. A Yiyi decirle que la quiero con todos sus defectos y que me alegra que sea parte de nuestra familia, le agradezco todo su apoyo y estar siempre presente en mi vida.

A todos los que vinieron de Ciego también les agradezco compartir conmigo y soportarme. Además a Danay y a Yoiler.

A todas aquellas personas que están hoy aquí, e hicieron todo esto posible.

RESUMEN

La Ventanilla Única Aduanera es un sistema creado para gestionar los trámites requeridos para las operaciones comerciales que realiza la Aduana General de la República (AGR). El objetivo de la presente investigación es desarrollar un componente de administración y seguridad para la Ventanilla Única Aduanera que permita elevar la seguridad, adaptándose a los cambios en la estructura y niveles de acceso de la AGR.

Para lograr el objetivo se emplearon las tecnologías y herramientas establecidas por Departamento de Soluciones Financieras y Aduanales: Visual Paradigm, PHP, Symfony, Oracle, Apache, JQuery y Bootstrap. Se realiza un estudio sobre soluciones informáticas similares para la gestión de administración y seguridad de la información.

Se obtuvo un componente de administración y seguridad que brindará facilidad en la gestión de usuarios, roles, permisos, recursos y funcionalidades. Además, permitirá que los roles de usuarios se organicen en una jerarquía, asumiendo los permisos de manera descendente. Logrando una mayor seguridad en el sistema.

Palabras claves: componente, administración, seguridad, información, gestión.

ABSTRACT

The Single Customs Window is a system created to manage the procedures required for commercial operations carried out by the General Customs of the Republic (AGR). The objective of the present investigation is to develop an administration and security component for the Single Customs Window that allows to increase security, adapting to the changes in the structure and access levels of the AGR. To achieve this objective, the technologies and tools established by the Department of Financial and Customs Solutions were used: Visual Paradigm, PHP, Symfony, Oracle, Apache, JQuery and Bootstrap. A study is carried out on similar computer solutions for information management and security management.

An administration and security component was obtained that will provide facility in the management of users, roles, permissions, resources and functionalities. In addition, it will allow user roles to be organized in a hierarchy, assuming permissions in a downward manner. Achieving greater security in the system.

Keywords: Component, administration, security, information, management.

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En las últimas décadas el manejo de los nuevos avances tecnológicos ha permitido que la información pueda ser consultada desde cualquier parte del mundo, garantizando las conexiones seguras y los accesos cada vez más rápidos.

Su evolución en aplicaciones web, visualización, el incremento de dispositivos móviles y sus aplicaciones, los desafíos de las Tecnologías de la Información y las Comunicaciones (TIC's) y en especial, de los oficiales de seguridad de la información se han vuelto más complejos al momento de tratar de asegurar la información.

La seguridad de la información, según la norma ISO 27001:2005 («ISO» 2017): "consiste en la preservación de su confidencialidad, integridad y disponibilidad, así como de los sistemas implicados en su tratamiento, dentro de una organización".

Igualmente, se puede complementar con la definición del Ingeniero y Máster en Sistemas Jeimy Cano (Cano 2017): "La Seguridad de la Información es la disciplina que habla de los riesgos, de las amenazas, de los análisis de escenarios, de las buenas prácticas y esquemas normativos, que exigen niveles de aseguramiento de procesos y tecnologías para elevar el nivel de confianza en la creación, uso, almacenamiento, transmisión, recuperación y disposición final de la información".

Las organizaciones están expuestas día a día a amenazas tanto internas como externas que ocasionan robo de identidad e información, bases de datos, información sensible de clientes, pérdida de credibilidad y daños financieros que pueden afectar la sostenibilidad de la entidad. Es por ello que se hace necesario la utilización de técnicas o métodos para controlar el flujo y la integridad de la información.

Con el fin de no quedarse a espaldas de los acontecimientos que hoy existen en el mundo, Cuba pone en marcha la informatización del país. La Universidad de las Ciencias Informáticas (UCI) es uno de los centros de la educación superior que contribuye a ello. Esta institución crea y desarrolla proyectos informáticos que permiten vincular las facilidades del empleo de las TIC's en beneficio al desarrollo social y económico del país. Uno de estos proyectos es Ventanilla Única Aduanera (VUA) desarrollado por el Centro de Informatización para la Gestión de Entidades (CEIGE).

VUA es el sistema encargado de centralizar la recepción de documentos necesarios para los trámites comerciales ante la Aduana General de la República (AGR). El acceso al envío y revisión de dichos documentos debe seguir un estricto control definido por la AGR, que permita garantizar la compartimentación de la información por roles de usuario, así como la confiabilidad, accesibilidad y disponibilidad de los datos recepcionados en la VUA. Los roles de los usuarios deben seguir un orden

jerárquico asociado a la estructura definida por la AGR, que incluye además estructuras administrativas de entidades que interactúan con ella.

El componente de administración y seguridad desarrollado para la versión 1.0 de la VUA basa su sistema de autorización en la lectura y escritura de un fichero YAML[1] donde quedan registrados todos los permisos del sistema. Este enfoque tiene como desventaja que al no implementarse un mecanismo de bloqueo al escribir el fichero, no es posible evitar actualizaciones concurrentes al mismo; trayendo como consecuencia deformaciones sintácticas de la estructura del fichero y la posterior ocurrencia de fallos graves antes los cuales el sistema no es capaz de responder.

Además, el método de definir los permisos en un YAML ha demostrado no ser lo suficientemente flexible para responder a la totalidad de las necesidades organizativas de una institución como la AGR, la que a raíz de la actualización del modelo socioeconómico de Cuba ha evolucionado en los últimos tres años.

A continuación se enuncian algunos cambios ante los cuales el componente inicial ha presentado dificultades:

- Han surgido nuevas personas jurídicas como representantes y responsables de aerolínea con complejas funciones en el sistema de reportes de Información Adelantada de Pasajeros.

  Para garantizar la seguridad a estos reportes, fue necesario implementar un sistema de permisos que se ajustara a las necesidades del negocio, debido a que el componente de administración se limita a proteger rutas dentro del sistema y no provee un API para proteger otros tipos de recursos.
- El árbol de jerarquía de permisos debe ser invertido, garantizando que el nodo raíz tenga la menor cantidad posible de accesos.

  El enfoque actual establece que el nodo raíz del sistema tiene acceso a todas las funcionalidades y en niveles inferiores de la jerarquía se van eliminando estos accesos. Dicha forma de implementar el árbol no está acorde con los estándares de seguridad, porque introduce la posibilidad de recibir ataques en los que usuarios limitados puedan escalar permisos y acceder a información sensible.

---

[1] **YAML** es un acrónimo recursivo que significa "YAML Ain't Another Markup Language" (en castellano, "YAML no es otro lenguaje de marcado"). Es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl.

Se puede concluir que el componente de administración y seguridad desarrollado para la versión 1.0 de la VUA no es capaz de adaptarse a cambios en la estructura de la AGR, por lo que no asegura el nivel de seguridad requerido por un sistema que maneja datos sensibles para el comercio y la seguridad nacional. La arquitectura bajo la cual fue desarrollado dicho componente conllevaría a un costoso desarrollo para extender las funcionalidades actualmente existentes, lo cual hace complejo modificarlo para que cumpla con las necesidades anteriormente enunciadas. Es preciso además investigar y explotar nuevos conceptos de seguridad como lista de control de acceso (ACLs[2] del inglés, access control list) y Voters introducidos por el marco de trabajo Symfony y que no se incluyeron en el alcance inicial del componente de administración versión 1.0 de la VUA.

A partir de la problemática planteada anteriormente se define como **problema:** ¿Cómo elevar la seguridad de la información almacenada en la VUA teniendo en cuenta los cambios en la estructura y niveles de acceso de la AGR?

Definiéndose como **objeto de estudio**: seguridad y control de acceso en aplicaciones web. Se define según lo antes expuesto como **campo de acción:** los procesos de seguridad y control de acceso en sistemas de la AGR.

Para dar solución al problema planteado se trazó como **Objetivo:** desarrollar la versión 2.0 del componente de administración y seguridad para la VUA que permita elevar la seguridad, adaptándose además a los cambios en la estructura y niveles de acceso de la AGR.

Del objetivo general se derivan los siguientes **Objetivos específicos**:

- Elaborar el marco teórico de referencia que permita darle solución al problema planteado.
- Realizar el análisis y diseño del componente.
- Implementar un componente adaptable ante cambios en la estructura y niveles de acceso establecidos por la Aduana General de la República, y que permita elevar la seguridad de la información almacenada en la Ventanilla Única Aduanera.
- Validar la investigación utilizando la Métrica de procesos de seguridad definida por OWASP, para determinar qué tan bien los procesos de seguridad del sistema cumplen con los requisitos definidos por las políticas de seguridad y las normas técnicas seguidas por la organización.
- Probar y validar la solución propuesta mediante pruebas funcionales y de aceptación.

---

[2] **ACL** 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.

Se plantea la siguiente **Idea a defender:** si se desarrolla un componente en la Ventanilla Única Aduanera capaz de adaptarse a los cambios en la estructura y niveles de acceso en la AGR, será posible elevar la seguridad de la información almacenada en la VUA.

Con el fin de dar cumplimiento a los objetivos del presente trabajo, se concibieron las siguientes **Tareas de la investigación**:

- Caracterización de estándares y de arquitecturas orientadas a sistemas de seguridad.
- Definición y estudio de metodologías, tecnologías y herramientas a emplear para el desarrollo de la herramienta.
- Identificación de patrones a utilizar en el diseño e implementación del componente.
- Ejecución del análisis y diseño de la herramienta.
- Diseño y validación de los prototipos de interfaz de usuario.
- Implementación del componente.
- Analizar técnicas y métodos de evaluación para la validación del componente.
- Aplicación de instrumentos de medición y evaluación de la usabilidad de las interfaces de usuario.
- Validación del componente propuesto a partir de pruebas de software.
- Validación del componente con usuarios potenciales.

Los siguientes **métodos científicos** sustentan el trabajo de la investigación:

- **Histórico - Lógico:** posibilitó el estudio analítico de la trayectoria histórica de las herramientas del componente de administración y seguridad, facilitó conocer de forma general el funcionamiento y desarrollo de las mismas, permitiendo caracterizar así cada una en sus aspectos más externos.
- **Analítico – Sintético:** este método permitió el análisis de los documentos, procesos y teorías para la extracción de los elementos más importantes que se relacionan con los sistemas de gestión de administración y seguridad. Además, para buscar y analizar la información acerca de las tecnologías, metodologías y herramientas a utilizar en el desarrollo del sistema, seleccionando los principales elementos y características.
- **Método de modelación:** permitió realizar el proceso mediante el cual se crean criterios y alternativas para la propuesta de solución, facilitando la definición de los componentes de la misma, así como sus relaciones. Además, fue empleado para la realización de los artefactos que requiere la metodología seleccionada.

El presente trabajo de diploma se estructura en los siguientes capítulos:

Capítulo 1. "Fundamentación teórica": se realiza un estudio sobre soluciones informáticas similares existentes en sistemas para la gestión de administración y seguridad de la información. Además, se

analizan los principales elementos teóricos que constituyen la base de la investigación, entre los cuales se encuentran la metodología de desarrollo para el sistema, las tecnologías y herramientas a utilizar en la propuesta de solución.

Capítulo 2. "Análisis y diseño": se aborda el entorno de desarrollo definido, se explican las diferentes características que va a presentar el sistema a implementar y sus funcionalidades siguiendo los aspectos que plantea la metodología AUP-UCI.

Capítulo 3. "Implementación y validación": aborda temas correspondientes al desarrollo de la solución, describe cómo se implementan los elementos del modelo de diseño, define la estructura del producto y la construcción de la solución. Se enfatiza en la validación del sistema, con el objetivo de detectar no conformidades existentes en la aplicación y poder solucionarlas para obtener un software que responda a los requisitos definidos por el cliente.

CAPÍTULO 1. Fundamentación teórica.

Introducción al capítulo

Se realiza un estudio sobre soluciones informáticas similares para la gestión de administración y seguridad de la información. Se estudia la seguridad de la información en aplicaciones web, así como el control de acceso y sus mecanismos de autenticación, autorización y auditoría. Se analizan los principales elementos teóricos que constituyen la base de la investigación, entre los cuales se encuentran la metodología de desarrollo para el sistema, las tecnologías y herramientas a utilizar en la propuesta de solución.

1.1 Seguridad de la información en aplicaciones web

La AGR es un órgano de control que garantiza la seguridad y protección de la sociedad socialista y de la economía nacional, así como la recaudación fiscal y las estadísticas de comercio exterior. Las herramientas tecnológicas provistas por las TIC's que apoyan el logro de los objetivos estratégicos de esta organización, están expuestas a amenazas que pueden tener un impacto sobre dichos objetivos. Una manera adecuada de asegurar la información es el uso de estándares, normas y protocolos. La productividad y continuidad del negocio va a depender de la disponibilidad, integridad y confidencialidad de la información; y cualquier procedimiento que se adopte para asegurar la información de una organización.

Un sistema se considera seguro si cumple con las propiedades expuestas a continuación:

Integridad: este principio garantiza la autenticidad y precisión de la información sin importar el momento en que esta se solicita, o dicho de otra manera, una garantía de que los datos no han sido alterados ni destruidos de modo no autorizado. Para evitar este tipo de riesgos se debe dotar al sistema de mecanismos que prevengan y detecten cuándo se produce un fallo de integridad y que puedan tratar y resolver los errores que se han descubierto (Purificación Aguilera López 2010).

Confidencialidad: la Organización para la Cooperación y el Desarrollo Económico (OCDE), en sus Directrices para la Seguridad de los Sistemas de Información define la confidencialidad como «el hecho de que los datos o informaciones estén únicamente al alcance del conocimiento de las personas, entidades o mecanismos autorizados, en los momentos autorizados y de una manera autorizada». Para prevenir errores de confidencialidad debe diseñarse un control de accesos al sistema: quién puede acceder, a qué parte del sistema, en qué momento y para realizar qué tipo de operaciones (Purificación Aguilera López 2010).

Disponibilidad: la información ha de estar disponible para los usuarios autorizados cuando la necesiten. El programa MAGERIT[3] define la disponibilidad como «grado en el que un dato está en el lugar, momento y forma en que es requerido por el usuario autorizado. Situación que se produce cuando se puede acceder a un sistema de información en un período de tiempo considerado aceptable. La disponibilidad está asociada a la fiabilidad técnica de los componentes del sistema de información» (Purificación Aguilera López 2010).

La seguridad de la información está asociada a la forma en que se accede a los recursos del componente. Es necesario establecer un control de acceso, que permita a cada usuario interactuar con los recursos que tienen permitido.

1.2 Control de acceso

El control de acceso es el proceso por el cual, dada una petición de recursos, se permite o niega el acceso a los mismos en base a la aplicación de unas políticas de acceso. Evitar el acceso no autorizado y permitir la libre circulación del personal autorizado es la principal característica de un control de acceso en un sistema de seguridad (Úbeda 2015).

Los sistemas de control de accesos son un elemento clave en un sistema de seguridad, que cada día cuentan con mayor implantación, tanto si se trata de un acceso a infraestructuras físicas (hospital, hotel, fábrica) como virtuales o de red. Además permiten configurar los derechos de acceso para todos los usuarios, generar grupos con accesos a las zonas autorizadas y franjas de horario habilitadas. (Úbeda 2015).

El control de acceso comprende mecanismos de autenticación, autorización y auditoría. Sus principales objetivos son proteger datos y recursos frente al acceso no autorizado (proteger el secreto) y frente a una modificación no autorizada (proteger la integridad) a la vez que garantizar el acceso de los usuarios legítimos a los recursos (no denegación de servicio). Con el fin de conseguir estos objetivos, se controlan todos los accesos al sistema y sus recursos, y solo se permite que tengan lugar aquellos autorizados. Haciendo un símil con los sistemas físicos, en concreto los puentes levadizos y los puentes destruibles, podemos observar que ambos puentes cumplen con los dos primeros objetivos (proteger el secreto y la integridad). Sin embargo, los puentes destruibles no satisfacen el tercer objetivo puesto

[3] MAGERIT Es una metodología de análisis y gestión de riesgos de los sistemas de información. En inglés Metho-dology for Information Systems Risk Analysis and Management.

que al destruirse no pueden ser cruzados por nadie, provocando así una denegación de servicio para las peticiones autorizadas (Castellà-Roca et al. 2012) .

Todo sistema de control de acceso considera los siguientes elementos básicos (Castellà-Roca et al. 2012):

- Objetos (también llamados objetivos): son todas aquellas entidades de un sistema susceptibles de ser protegidas. En el caso de un sistema operativo pueden ser: archivos, directorios, programas, dispositivos, terminales y puertos. En el caso de una base de datos, tenemos: tablas, relaciones, vistas y procedimientos.
- Acciones: todo aquello que se puede realizar sobre un objeto. Las acciones típicas que se puede realizar sobre un fichero son: lectura, escritura, creación, eliminación. En el caso de que el objeto sobre el que se realiza la acción sea un programa, cabría añadir la opción de ejecución.
- Sujetos (también llamados iniciadores): es cualquier entidad con capacidad para requerir el acceso a objetos del sistema. Los sujetos típicos de un sistema son sus usuarios y los procesos del sistema (por ejemplo, un navegador web, un procesador de textos).

En todo sistema informático, los sujetos realizan acciones sobre los objetos. El sistema de control de acceso es el encargado de decidir si un determinado sujeto tiene permiso para ejecutar una determinada acción sobre un determinado objeto. La decisión de permitir o denegar el acceso a los recursos se realiza en base a las políticas de acceso (Úbeda 2015).

Las políticas de acceso son el conjunto de reglas que permiten determinar si un sujeto puede realizar una determinada acción (lectura, escritura, modificación, eliminación o ejecución) sobre un objeto.

La figura 1 muestra un esquema de los componentes principales de un sistema y su interacción con el sistema de control de acceso. Se observa cómo el sistema de control de acceso (en el centro de la figura) recibe peticiones de los sujetos para realizar acciones. Evalúa estas peticiones mediante el uso de una política de acceso y actúa en consecuencia permitiendo o denegando el acceso a los objetos (Úbeda 2015).

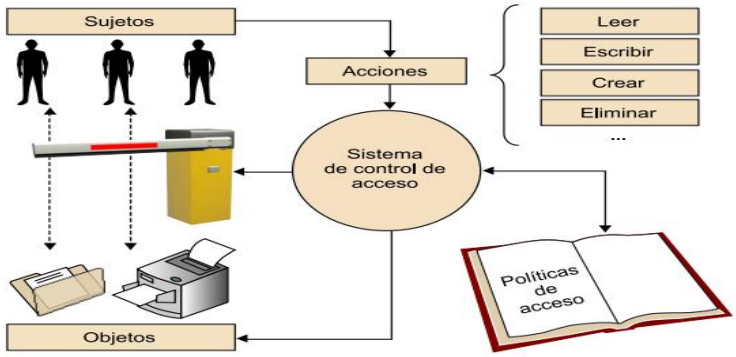

Figura 1 Esquema de control de acceso obligatorio.

1.2.1 Autenticación

El sistema debe ser capaz de verificar que un usuario identificado que accede a un sistema o que genera una determinada información es quien dice ser. Solo cuando un usuario o entidad ha sido autenticado, podrá tener autorización de acceso. Se puede exigir autenticación en la entidad de origen de la información, en la de destino o en ambas (Purificación Aguilera López 2010).

Técnicas de autenticación más comunes en la Web:

- **Autenticación básica y segura (Digest)**:

Casi todos los servidores web y de aplicación soportan el uso de autenticación básica y digest. Esto requiere que el explorador web presente un cuadro de diálogo para obtener el nombre de usuario y contraseña, y enviarlos al servidor web, el cual la procesará contra su propia base de datos de usuario, o en el caso de IIS4, con Directorio Activo5 (Stock 2005).

•La autenticación básica envía la credencial en texto claro. No debería ser usada a menos que se combine con SSL[6].

• La autenticación HTTP 1.0 Digest solo ofusca la contraseña. No debería ser usada.

• La autenticación HTTP 1.1 Digest un mecanismo de respuesta de reto, lo cual es razonablemente seguro para aplicaciones de bajo valor.

La razón principal en contra del uso de autenticación básica o digest es debido a:

• Transmisión insegura de credenciales.

---

[4] **Servicio de Información de Internet** (**IIS por sus siglas en inglés Internet Information Services)** es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows.

[5] **Directorio Activo** o **Active Directory** (AD) son los términos que utiliza Microsoft para referirse a su implementación de servicio de directorio en una red distribuida de computadores. Utiliza distintos protocolos, principalmente LDAP, DNS, DHCP y Kerberos.

[6] **Capa de puertos seguros** (**SSL por sus siglas en inglés "Secure Sockets Layer")** es un protocolo diseñado para permitir que las aplicaciones para transmitir información de ida y de manera segura hacia atrás. Las aplicaciones que utilizan el protocolo Secure Sockets Layer sí sabe cómo dar y recibir claves de cifrado con otras aplicaciones, así como la manera de cifrar y descifrar los datos enviados entre los dos.

• Ambas formas de autenticación sufren de ataques de reinyección [7] e intermediario [8].

• Ambas requieren SSL para proporcionar alguna forma de confidencialidad e integridad.

• No proporciona una gran cantidad de control a la aplicación final.

Las razones de no usar la autenticación básica y segura, no quiere decir que no son útiles. Puede ser usada para escudar sitios de desarrollo contra el uso casual o proteger interfaces administrativas de bajo valor.

- **Autenticación basada en certificado**:

La autenticación basada en certificado es ampliamente implementada en muchos servidores Web y de aplicación. El sitio Web expide certificados (o intenta confiar en certificados emitidos externamente). Los certificados públicos son cargados en la base de datos de autenticación del servidor, y comparados con las sesiones entrantes del navegador. Si los certificados coinciden, el usuario es autenticado.

La calidad de la autenticación está directamente relacionada con la calidad de la infraestructura de la llave pública para expedir certificados. Un certificado emitido a cualquiera que lo pida no es tan confiable como los certificados emitidos después de ver tres formas de identificación por foto (como pasaporte, licencia de conducir o tarjeta de identificación nacional) (Stock 2005).

- **Autenticación fuerte:**

La autenticación fuerte (como tokens, certificados) proporciona un nivel más alto de seguridad que nombres de usuario y contraseñas. La forma generalizada de autenticación fuerte es “algo que sabes, algo que tienes”. Por lo tanto, cualquier cosa que requiera un secreto (el “algo que sabes”) y autenticado como un token, llave USB, o certificado (el “algo que tienes”) es un control más fuerte que nombres de usuario y contraseñas (que es solo “algo que sabes”) o biométricos (“algo que eres”) (Stock 2005).

[7] **Ataque de reinyección** conocido como ataque de replay, es una forma de ataque de red, en el cual una transmisión de datos válida es maliciosa o fraudulentamente repetida.

[8] **Ataque de intermediario** (*man-in-the-middle*) es un ataque en el que se adquiere la capacidad de leer, insertar y modificar a voluntad, los mensajes entre dos partes sin que ninguna de ellas conozca que el enlace entre ellos ha sido violado.

- **Autenticación federada:**

La autenticación federada le permite externalizar su base de datos de usuario a un tercero, o para tener varios sitios con un enfoque SSO[9]. La principal razón de negocio para seguridad federada es que los usuarios sólo tienen que ingresar una vez, y todos los sitios que soporten esa autenticación pueden confiar en el token de ingreso y de ahí confiar en el usuario y proveer servicios personalizados. Ventajas de autenticación federada (Stock 2005):

• Reducir el número total de credenciales que sus usuarios tienen que recordar.

• Su (s) sitio (s) es (son) parte de una gran asociación comercial, como una extranet.

• Le gustaría ofrecer servicios personalizados a otros usuarios anónimos.

No debería usar autenticación federada, a menos que:

• Confía en el proveedor de la autenticación.

• Sus requerimientos de cumplimiento de privacidad son cumplidos por el proveedor de la autenticación.

1.2.2 Autorización

Autorización: consiste en dar acceso a una serie de recursos a un usuario o sistema (para ello, el usuario o el sistema previamente tendrán que haberse autenticado) (Sánchez 2011).

A continuación se analizan los diferentes modelos de control de acceso:

Control de acceso obligatorio (MAC)

Las políticas son evaluadas por el sistema entendido como un único ente central. Los sujetos del sistema no pueden rehacer/redefinir las políticas. Por ejemplo, no pueden dar permisos de acceso a otros usuarios. Los objetos y sujetos del sistema pertenecen a diversas clases de acceso. Así, para acceder a un objeto de una determinada clase hace falta que el sujeto pertenezca a una clase igual o superior (en términos de privilegios) (Úbeda 2015).

Este tipo de acceso se inspira en el funcionamiento militar en el que la información (por ejemplo, planes de ataque) solo puede ser vista por aquellas personas que gozan de un nivel de seguridad suficiente (por ejemplo, nivel de coronel) (Úbeda 2015).

[9] **Single sign-on** (SSO) es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación. Su traducción literal sería algo como «autenticación única» o «validación única».

Control de acceso discrecional (DAC)

Las políticas son gestionadas por los propietarios (sujetos) de los recursos (objetos). Los sujetos pueden modificar las políticas asociadas a los objetos del sistema. Por ejemplo, un propietario de un determinado objeto del sistema puede dar privilegios de acceso sobre ese objeto a otro sujeto. Este tipo de control de acceso es usado generalmente por los sistemas operativos (Castellà-Roca et al. 2012).

Modelo de control de acceso (RBAC)

Su principal objetivo es prevenir que los usuarios tengan libre acceso a la información de la organización. El modelo introduce el concepto de rol y asocia a los usuarios con las funciones por los que va pasando durante la vida del sistema, los permisos de acceso están asociados a los roles, el rol es un concepto típico usado en empresas para ordenar y estructurar sus actividades organizativas. RBAC permite modelar la seguridad desde de una perspectiva empresarial puesto que se pueden conectar los requerimientos de seguridad con los roles y las responsabilidades existentes en la organización. RBAC está basado en la definición de un conjunto de elementos y de relaciones entre ellos. A nivel general describe un grupo de usuarios que pueden estar actuando bajo un conjunto de roles y realizando operaciones en las que utilizan un conjunto de objetos como recursos (Domínguez et al. 2015).

En una organización, un rol puede ser definido como una función que describe la autoridad y responsabilidad dada a un usuario en un instante determinado. Entre estos elementos se establecen relaciones del tipo (Domínguez et al. 2015):

- Relaciones entre usuario y roles, modelando los diferentes roles que puede adoptar un usuario.
- Conjunto de operaciones que se pueden realizar sobre cada uno de los objetos. A los elementos de esta relación se les denomina permisos. Relaciones entre los permisos y los roles.

El modelo RBAC incluye un conjunto de sesiones donde cada sesión es la relación entre un usuario y un subconjunto de roles que son activados en el momento de establecer dicha sesión. Cada sesión está asociada con un único usuario, mientras que un usuario puede tener una o más sesiones asociada, los permisos disponibles para un usuario son el conjunto de permisos asignados a los roles que están activados en todas las sesiones del usuario, sin tener en cuenta las sesiones establecidas por otros usuarios en el sistema. RBAC añade la posibilidad de modelar una jerarquía de roles de forma que se puedan realizar generalizaciones y especializaciones en los controles de acceso y se facilite la modelización de la seguridad en sistemas complejos (Domínguez et al. 2015).

El análisis de los diferentes modelos de control de acceso, permite que se defina cuál de ellos se utilizarán en la propuesta de solución. Obteniendo que la implementación sea basado en uno de ellos, para determinar si el usuario previamente identificado y autenticado tiene permitido el acceso a los recursos del componente.

1.2.3 Auditoría

La auditoría es un análisis pormenorizado de un sistema de información que permite descubrir, identificar y corregir vulnerabilidades en los activos que lo componen y en los procesos que se realizan. Su finalidad es verificar que se cumplen los objetivos de la política de seguridad de la organización. Proporciona una imagen real y actual del estado de seguridad de un sistema de información (Purificación Aguilera López 2010).

Existen auditorías internas (las que una organización hace a su propio sistema de gestión) y externas (las que se hacen al sistema de gestión de una organización por parte de un cliente o de un tercero independiente).

Se pueden hacer auditorías a cualquier tipo de sistema para el que haya establecido un estándar o norma que describa los requisitos que debe cumplir puesto que el proceso de auditoría consiste en comprobar el cumplimiento de ese estándar. Así encontramos auditorías de sistemas de gestión de la calidad, ambiental, de prevención de riesgos laborales, de seguridad de la información y de sistemas integrados.

Para verificar que se encuentren correctamente implementados los mecanismos de autenticación y autorización se realizan diferentes pruebas. Estas pruebas permiten auditar el componente para identificar vulnerabilidades, con el objetivo de comprobar que sea seguro.

1.3 Proyecto abierto de seguridad en aplicaciones web

El software inseguro está debilitando las finanzas, salud, defensa y energía. A medida que la infraestructura digital se hace cada vez más compleja e interconectada, la dificultad de lograr la seguridad en aplicaciones aumenta exponencialmente (Wichers 2013).

El proyecto abierto de seguridad en aplicaciones web (OWASP por sus siglas en inglés) es una comunidad abierta dedicada a habilitar a las organizaciones para desarrollar, comprar y mantener aplicaciones confiables. Abogan por resolver la seguridad de aplicaciones como un problema de personas, procesos y tecnología; porque las soluciones más efectivas incluyen mejoras en todas estas áreas (Wichers 2013).

Es por ello que OWASP propone las siguientes pruebas de seguridad (Edgar D. Salazar T 2012):

- Recopilación de Información: la primera fase en la evaluación de seguridad se centra en recoger tanta información como sea posible sobre una aplicación objetivo. La recopilación de información es un paso necesario en una prueba de intrusión.
- Pruebas de gestión de la configuración: a menudo los análisis sobre la infraestructura o la topología de la arquitectura pueden revelar datos importantes sobre una aplicación web. Se pueden obtener datos como por ejemplo el código fuente, los métodos HTTP permitidos, funcionalidades administrativas, métodos de autenticación y configuraciones de la infraestructura.
- Pruebas de la lógica de negocio: las pruebas de esta categoría es la comprobación de las reglas del negocio definidas para la aplicación.
- Pruebas de Autenticación: en seguridad informática, autenticación es el proceso de intentar verificar la identidad digital del remitente de una comunicación. Esta categoría pone a prueba el sistema de autenticación de la aplicación mediante prueba de fuerza bruta y pruebas al recordatorio de contraseñas del sistema.
- Pruebas de Autorización: autorización es el concepto de permitir el acceso a recursos únicamente a aquellos que tienen permiso para ello. Las pruebas de Autorización significan entender cómo funciona el proceso de autorización, y usar esa información para saltarse el mecanismo de autorización.
- Pruebas de gestión de sesiones: la gestión de sesiones cubre ampliamente todos los controles que se realizan sobre el usuario, desde la autenticación hasta la salida de la aplicación.
- Pruebas de validación de datos: la debilidad más común en la seguridad de aplicaciones web, es la falta de una validación adecuada de las entradas procedentes del cliente o del entorno de la aplicación. Esta debilidad conduce a casi todas las principales vulnerabilidades en aplicaciones.
- Pruebas de denegación de servicio: el tipo más común de ataque de denegación de servicio (Dos) es del tipo empleado en una red para hacer inalcanzable a la comunicación a un servidor por parte de otros usuarios válidos. El concepto fundamental de un ataque DoS de red es un usuario malicioso inundando con suficiente tráfico una máquina objetivo para conseguir hacerla incapaz de sostener el volumen de peticiones que recibe. Cuando el usuario malicioso emplea un gran número de máquinas para inundar de tráfico una sola máquina objetivo, se conoce generalmente como ataque denegación de servicios distribuidos (DDoS).
- Pruebas de servicios web: los servicios web están expuestos a la red como cualquier otro servicio, pero pueden ser utilizados en HTTP, FTP, SMTP o acompañados de cualquier otro

protocolo de transporte. Las vulnerabilidades en servicios web son similares a otras vulnerabilidades como la inyección SQL, revelación de información, pero también tienen vulnerabilidades de XML.

- Pruebas de AJAX: el uso de las técnicas AJAX puede conseguir enormes beneficios en la experiencia de uso por parte de los usuarios de las aplicaciones web. Sin embargo, desde el punto de vista de la seguridad, las aplicaciones AJAX tienen una superficie de ataque mayor que las aplicaciones web convencionales, a veces son desarrolladas centrándose más en qué se puede hacer que en qué se debería hacer. Además, las aplicaciones AJAX son más complicadas porque el procesamiento se realiza tanto en el lado del cliente como en el lado del servidor.

Para la validación de la seguridad del componente a desarrollar se propone utilizar la herramienta Acunetix. Esta herramienta implementa pruebas de autenticación, autorización y de servicios web, con el objetivo de detectar vulnerabilidades en el componente de administración y seguridad en la versión 2.0.

1.4 Análisis de los componentes de administración de la seguridad

Se realiza un estudio sobre los componentes de administración de la seguridad, mostrándose a continuación cada uno de ellos:

- **Componente Boson**

SeguridadBundle es el componente encargado de proveer las funcionalidades básicas de autenticación y autorización basadas en los estándares propuestos por el lenguaje de marcado de aserción de seguridad [SAML por sus siglas en inglés]. La implementación de SAML utilizada es [SimpleSamlphp] (UCI 2017).

En el componente de Seguridad, el paquete de autorización es el encargado de validar si las credenciales mostradas por el usuario conectado son suficientes para acceder a determinado recurso del proyecto. El modelo de control de acceso utilizado es basado en roles, los que son almacenados en base de datos y asociados a cada uno de los usuarios en una relación de muchos a muchos (UCI 2017).

Este componente de seguridad utiliza estrategias de acceso que permiten o deniegan el acceso a los recursos del sistema. Para la gestión de los recursos el componente brinda una serie de funcionalidades. El componente cuando detecta algún fallo le informa al usuario. Además que la información sensible es encriptada (UCI 2017).

- **FOSUserBundle**

El FOSUserBundle es un componente o bundle para la gestión de usuarios en Symfony. Es un proyecto para versiones de Symfony2 en lo adelante, que permite el uso de un subsistema de administración de usuarios. Este bundle permite registrar nuevos usuarios, enviar un mensaje de confirmación a los usuarios recién creados, editar el perfil del usuario, encriptación de contraseñas y agrega un campo en la base de datos para poder guardar roles.

FOSUserBundle le permite asociar grupos a sus usuarios. Los grupos son una forma de agrupar una colección de roles. Las funciones de un grupo se consideran para todos los usuarios que pertenezcan a ese grupo. Este componente maneja la autenticación basada en usuario y contraseña. No implementa la gestión de autorización pues lo facilita como lo hace Symfony. Hace uso del manejo de errores y la información sensible es encriptada.

Symfony admite la herencia de roles, por lo que no siempre se necesita heredar roles de los grupos. Si la herencia de roles es suficiente para su caso de uso, es mejor usarlo en lugar de grupos, ya que es más eficiente.

Para utilizar los grupos, debe habilitar explícitamente esta funcionalidad en su configuración. La única configuración obligatoria es el nombre de clase totalmente calificado (FQCN) de su clase de grupo que debe implementar FOS \ UserBundle \ Model \ GroupInterface.

- **Componente de Administración del Sistema Ventanilla Única para el Comercio Exterior de Cuba, versión 1.0**

El componente de Administración de la VUA, versión 1.0 maneja la autenticación mediante usuarios y contraseñas. Además para la autorización hace uso del modelo de control de acceso basado en roles (RBAC), por adecuarse a las necesidades de la VUA para manejar el acceso a los recursos del sistema que necesitan ser protegidos (León y López. 2013).

Para manejar los errores cada vez que un dato es insertado incorrectamente o exista un campo obligatorio vacío, se le informa al usuario el error.

A partir del análisis del manejo de la seguridad en los componentes estudiados, se muestra a continuación las características similares:

Tabla 1 Medición de indicadores de los componentes.

Indicadores	**Boson**	**FOSUserBundle**	**Componente de Administración del Sistema VUA, versión 1.0.**
Gestión de permisos mediante jerarquía	X		

Basado en roles	X	X	X
Trazabilidad de las operaciones			X
Autenticación basada en dominios	X		
Manejo de errores	X	X	X
Encriptación de información sensible	X	X	X

Con el estudio realizado se determinaron características similares que poseen estos componentes de seguridad. Aunque no resuelven la totalidad de las necesidades existentes, se puede reutilizar y tomar de ellos buenas prácticas y principios de diseño. Como la versión 2.0 del componente de administración y seguridad de la VUA tiene que asegurar que los roles sigan un orden jerárquico, se toma como referencia el componente Boson.

Boson provee las funcionalidades básicas de autenticación y autorización. Como es un componente desarrollado en la UCI, posibilitaría constantes actualizaciones de los problemas detectados. Además permite desacoplar sus funciones, brindando la posibilidad de utilizarlos en diferentes sistemas gracias a su bajo acoplamiento.

1.5 Metodología de desarrollo de software

En la actualidad existen diferentes propuestas de metodologías, que inciden de formas distintas en el desarrollo de software. “Una metodología de desarrollo de software se fundamenta sobre tres pilares básicos: que hay que hacer y en qué orden, como deben realizarse las tareas y con que pueden llevarse a cabo. Esto es que etapas, actividades y tareas se deben acometer, que técnicas deben emplearse para realizar estas actividades y cuáles son sus herramientas software a utilizar en cada caso” (Cevallos Cadena 2015).

Para la actividad productiva de la UCI se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida de los proyectos. Para el siguiente trabajo se utiliza la metodología AUP-UCI, el componente a desarrollar es para el proyecto VUA del CEIGE, y este se rige por esta metodología. Define las fases por la que el proyecto transita: Inicio, Ejecución y Cierre. Además de los escenarios como se muestra a continuación:

- Escenario No 1: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del

negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema.

- Escenario No 2: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio.
- Escenario No 3: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad.
- Escenario No 4: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos.

De los cuatro escenarios que propone la metodología para la disciplina de requisitos, se escoge el escenario 3.

1.6 Herramientas y tecnologías empleadas

A continuación partiendo de que esta selección ha sido definida en la arquitectura del proyecto en su modelo de desarrollo, se describen las tecnologías, lenguajes y herramientas usadas para el desarrollo del componente.

1.6.1 Herramienta de modelado Visual Paradigm 8.0

El modelado de datos es el proceso de documentar un diseño de sistema de software complejo como un diagrama de fácil comprensión, usando texto y símbolos para representar la forma en que los datos necesitan fluir.

Para soportar el ciclo de vida del proceso de desarrollo del software a través de la representación de todo tipo de diagramas, se decidió utilizar el Visual Paradigm.

Visual Paradigm es una herramienta CASE (acrónimo de Computer Aided Software Engineering, en español: Ingeniería de Software Asistida por Computadora) de diseño para UML, soporta el ciclo devida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Brinda además, la posibilidad de crear, modificar y diseñar estos diagramas con rapidez y calidad, lo que ayuda a aumentar la eficiencia del sistema de análisis y

diseño de manera significativa («Visual Paradigm Features» 2017).Esta herramienta propicia un conjunto de ayudas para el desarrollo de la Versión 2.0 del componente de administración y seguridad de la Ventanilla Única Aduanera, desde la planificación, análisis y diseño del sistema.

1.6.2 Lenguaje de desarrollo PHP 5.4.

Para el presente trabajo se utiliza **PHP** (acrónimo recursivo *Hypertext Preprocessor,* en español: Hipertexto Pre-Procesado) es un lenguaje de programación interpretado que posibilita la generación dinámica de contenidos en un servidor web («PHP: Hypertext Preprocessor» 2017).

1.6.3 Marco de trabajo Symfony 2.8.

Symfony es un marco de trabajo diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación (Potencier y Zaninotto 2012).

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura Modelo-Vista-Controlador (MVC), que está formado por tres niveles (Potencier y Zaninotto 2012):

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

1.6.4 Gestor de Base de Datos Oracle 11g.

Es un Sistema de Gestión de Base de Datos Objeto-Relacional (ORDBMS), desarrollado por la corporación Oracle. Se considera como uno de los sistemas de bases de datos más completos, destacando:

- Compilación automática de PL/SQL y Java en la base de datos.
- Triggers más rápidos que en versiones anteriores, incluidas las llamadas más eficientes de triggers por fila.
- Operaciones SQL sencillas más rápidas.

- Conexiones directas más rápidas confiables con los dispositivos de almacenamiento del Sistema de Archivos de la Red (NFS, por sus siglas en inglés).
- Actualizaciones más rápidas.
- Backup/restauración más rápida de archivos grandes.
- Compresión de backup más rápida.

1.6.5 Servidor Web Apache 2.6.

Apache es un servidor web HTTP de código abierto para la creación de páginas y servicios web. Se caracteriza por ser estable, multiplataforma, modular y altamente configurable, lo cual significa que se puede adaptar para satisfacer diferentes necesidades (Montoya, Uribe y Rodríguez 2013).
Se utiliza Apache porque es el servidor web fácil de configurar y se ejecuta en gran cantidad de sistemas operativos, lo que lo hace prácticamente universal.

1.6.6 JQuery 8.0.

JQuery es un tipo de biblioteca o marco de trabajo de *JavaScript* que permite simplificar la manera de interactuar con los documentos *HTML*, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología *AJAX* al sistema (Alvarez 2010).
JQuery al igual que otras librerías, ofrece una serie de funcionalidades basadas en *JavaScript* que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de esta librería se logran grandes resultados en menos tiempo y espacio. La gran ventaja de *JQuery* es que permite cambiar el contenido de la página web sin necesidad de recargarla, utilizando *DOM* y *AJAX* de manera sencilla gracias a su sintaxis («Manual de jQuery» 2017).

1.6.7 Twitter Bootstrap 2.3.

Twitter Bootstrap es un framework CSS desarrollado inicialmente (en el año 2011) por **Twitter** que permite dar forma a un sitio web mediante **librerías CSS.** Es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías. Además es compatible con la mayoría de navegadores web del mercado («¿Que es Bootstrap?» 2017).

Se integra muy bien con las principales librerías JavaScript, por tanto con JQuery. Una gran comunidad de desarrolladores en Github[10] para dar soporte, por lo que hacen a Bootstrap un entorno de trabajo más robusto que otros frameworks.

1.7 Conclusiones parciales

- El desarrollo del marco teórico de la investigación permitió caracterizar la gestión de la seguridad en la VUA.
- El estudio realizado a diferentes soluciones informáticas denotó la presencia del componente Boson, presentando características similares a la propuesta de solución.
- Además fueron definidas las herramientas y tecnologías mayormente adecuadas para dar solución al problema planteado, teniendo en cuenta las características de estas y las necesidades descritas con anterioridad a las que da respuesta la propuesta de solución.

[10] **GitHub** es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

Introducción

En este capítulo se realiza una descripción de las características del sistema, las cuales son el punto de partida para comenzar el proceso de desarrollo. Estas permiten tener una visión objetiva acerca del producto que se desea obtener. Además, se identifican los requerimientos funcionales y no funcionales que debe cumplir. Siguiendo con la metodología AUP-UCI, Se generan los artefactos necesarios tanto para el análisis como el diseño, para formalizar y documentar toda la información que se genera durante el desarrollo de la solución.

2.1 Propuesta de solución

En toda organización social sin importar cuál sea su naturaleza, origen o propósito existen en su interior estructuras definidas ya sea formal o informalmente que permiten integrar y organizar a sus miembros de forma que cumplan con sus funciones y puedan operar mejor.

El componente de administración y seguridad en su versión 1.0, basa su sistema de autorización en la lectura y escritura de un fichero YAML. Al tener que seguir un orden jerárquico y el árbol de jerarquía de permisos debe ser invertido, no cumple con la estructura de la AGR.

Es por ello, que se tiene como propuesta de solución un componente de administración y seguridad para la Ventanilla Única Aduanera. Este necesita proveer las funcionalidades necesarias para gestionar usuarios, roles, permisos, recursos y trazas. Este componente es un Bundle de Symfony que integra y adapta los servicios del componente de seguridad de Boson.

Para el componente el administrador es el encargado en la opción de “Control de Acceso” de administrar las funcionalidades que allí se encuentran. Este les asigna a determinados usuarios los roles, los que proporcionan permisos a los usuarios para que realicen operaciones específicas en los recursos. Estos se organizan en una jerarquía**,** asumiendo los permisos de manera descendente. Los roles en un nivel inferior en la jerarquía normalmente incluirán los permisos de los roles que se encuentran en niveles superiores.

La asignación de permisos (o derechos de acceso) a los recursos para determinados usuarios, puede restringir o permitir el acceso de un determinado usuario a un recurso para su visualización de contenidos, modificación y/o ejecución (en caso de un archivo ejecutable). A mayor permiso para un usuario/sistema, más posibilidades para manipular el recurso.

2.2 Modelo de dominio

Un modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos de software, y será una entrada necesaria para varios de los artefactos que se generarán en el diseño de la solución. El modelo del dominio muestra a los modeladores las clases conceptuales significativas en un dominio del problema (Larman 1999).

Como punto de partida se utiliza el modelo de dominio, para lograr el diseño adecuado del sistema que se desea desarrollar. Constituyendo el artefacto más importante que se crea durante el análisis orientado a objetos.

2.2.1 Modelo conceptual

Un modelo conceptual es una representación de conceptos en un dominio del problema. En UML, se ilustra con los diagramas de clase, donde no se definen operaciones solo conceptos, asociaciones entre conceptos y atributos de conceptos.

El modelo conceptual explica cuáles son y cómo se relacionan los conceptos relevantes en la descripción del problema. A continuación se muestra el modelo conceptual del negocio, el cual presenta los conceptos, los atributos y las relaciones entre los conceptos que intervienen en el componente de administración y seguridad de la VUA. Es por ello que es necesario una breve descripción de los conceptos:

- Usuario: usuario que se registra en la VUA.
- Roles: es una colección de permisos definida para todo el sistema que se le asigna a los usuarios específicos en contextos específicos.
- Permisos: restringe o permite el acceso de un determinado usuario a un recurso determinado.
- Recurso: son partes del sistema que dan solución a determinado negocio, recursos a los que el usuario tiene acceso.
- Funcionalidad: recursos nomenclados en el sistema para su posterior uso, facilita la gestión del menú en el mismo.
- Trazas: registro de las operaciones realizadas por un usuario.

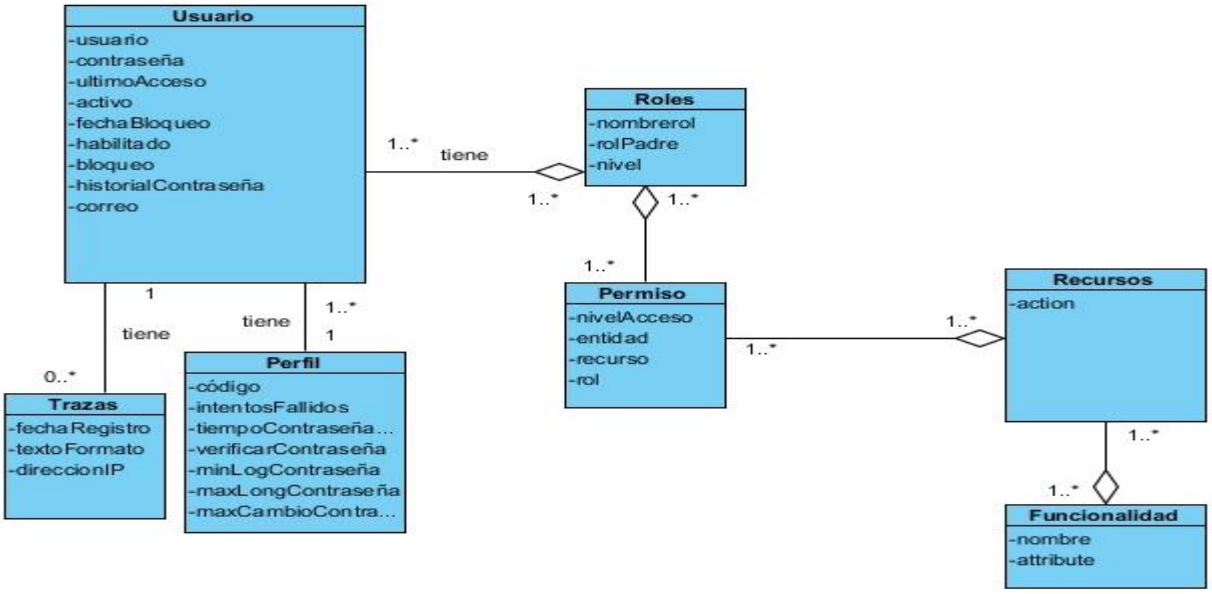

Figura 2 Diagrama del modelo conceptual.

En la (Figura 2) del modelo conceptual se puede observar, que uno o varios usuarios, que se registran a la VUA, van a pertenecer a distintos roles. Teniendo como objetivo definir los permisos para todo el sistema, se le asignarán los recursos a los que tendrán permiso. Además se engloban las funcionalidades asociadas a esos recursos. Permitiendo que cuando se le asigne un rol a un usuario, ya estén definidas todas las acciones a las que va a poder acceder en el sistema.

2.3 Especificación de los requisitos de software

Al especificar los requisitos de software, se describe cada detalle a tener en cuenta para el desarrollo del sistema. Esta captura de requisitos evita errores en etapas posteriores. A continuación se muestran las técnicas utilizadas y el listado de los requisitos funcionales definidos.

2.3.1 Técnicas para la captura de requisitos

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho. El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él.

Para la recopilación y obtención de la información se utilizaron las siguientes técnicas: la **entrevista** fue aplicada a los trabajadores de CEIGE para conocer sobre el proyecto y sus necesidades, **la observación** se llevó a cabo para percibir cómo se desarrollan las operaciones de la VUA, tomando los principales problemas que están presentando en su negocio y la **tormenta de ideas** para acumular ideas, y así tener una perspectiva general de las necesidades del sistema.

2.3.2 Requisitos Funcionales

En la captura de requisitos se definieron los siguientes 20 requisitos funcionales, donde se describen a continuación:

Tabla 2 Descripción de los requisitos funcionales.

N⁰	***Agrupación***	**Funcionalidad**	**Descripción**
RF1	Gestionar usuarios	Adicionar usuario	Se crea un nuevo usuario interno del sistema con los datos requeridos.
RF2	Gestionar usuarios	Buscar usuario	Se buscan usuarios por los criterios de búsqueda establecidos.
RF3	Gestionar usuarios	Modificar usuario	Se modifican los datos del usuario.
RF4	Gestionar usuarios	Desactivar usuario	Se desactiva uno o varios usuarios del sistema.

RF5	Gestionar usuarios	Activar usuario	Se activa uno o varios usuarios del sistema.
RF6	Gestionar usuarios	Asignar rol	Permite asignarle uno varios roles a uno o más usuarios.
RF7	Gestionar usuarios	Establecer contraseña	Permite establecer la contraseña a un usuario validando la misma contra las especificaciones del perfil.
RF8	Gestionar roles	Adicionar rol	Se crea un nuevo rol en el sistema.
RF9	Gestionar roles	Buscar rol	Se buscan roles por los criterios de búsqueda establecidos.
RF10	Gestionar roles	Modificar rol	Se modifican los datos del rol.
RF11	Gestionar roles	Eliminar rol	Se elimina un rol del sistema.

RF12	Gestionar recursos	Descubrir recursos	Se busca en el código fuente todos los recursos disponibles en el sistema y se sincronizan con los registrados en la base de datos.
RF13	Gestionar permisos	Adicionar permiso	Adiciona un permiso de uno o varios roles para uno o varios recursos
RF14	Gestionar permisos	Buscar permiso	Busca los permisos por los criterios de búsqueda establecidos
RF15	Gestionar permisos	Modificar permiso	Modifica los datos del permiso
RF16	Gestionar permisos	Eliminar permiso	Elimina uno o varios permisos del sistema
RF17	Gestionar trazas	Adicionar trazas	Adiciona trazas en el sistema
RF18	Gestionar trazas	Buscar trazas	Busca trazas en el sistema por los criterios de búsqueda establecidos

RF19	Seguridad del sistema	Autenticar usuario	Verifica las credenciales del usuario e inicia la sesión si procede
RF20	Seguridad del sistema	Cerrar sesión de usuario	Cierra la sesión del usuario conectado

A continuación se muestra la descripción textual del requisito Adicionar Usuario, donde se expone con más detalle el funcionamiento del mismo.

Tabla 3 Descripción textual del requisito Adicionar Usuario.

Precondiciones	1. El administrador encargado de la inserción debe tener los permisos necesarios y estar autenticado en el sistema. 2. El administrador debe seleccionar del menú la opción “Control de Acceso”. 3. El administrador selecciona la opción “Gestionar Usuario”.
Flujo de eventos	
Flujo básico de adicionar usuario.	
1	El administrador selecciona la opción “Adicionar”.
2	El sistema muestra una interfaz con los siguientes campos. *Ver Ilustración 1*: • Nombre Usuario. • Clave. • Confirmar. • Correo. • Activo/Inactivo.

3	El administrador introduce el usuario, el correo, la contraseña, confirma la contraseña, establece si el usuario está Activo/Inactivo.
4	El administrador selecciona la opción “Aceptar”. En caso de que todos los campos requeridos no estén llenos ver flujo alterno *4.a Datos incompletos*. En caso de que exista el usuario insertado ver flujo alterno *4.b Usuario repetido.* En caso de que la contraseña insertada no coincida con la insertada en el campo “Confirmar contraseña” ver flujo alterno *4.c No coincide la contraseña*. En caso de que la contraseña no cumpla con lo establecido por el perfil ver flujo alterno *4.d Contraseña incorrecta.* En caso de seleccionar la opción “Cancelar” ver flujo alternativo *4.e Cancelar.*
5	El sistema actualiza el listado de los usuarios.
6	Concluye el requisito.
Pos-condiciones	
1	Se adiciona correctamente el usuario.
Flujos alternativos	
Flujo alternativo 4.a Datos incompletos.	
1	El sistema muestra una alerta informando que es obligatorio el llenado de los campos marcados como requeridos.
2	Vuelve al paso 3 del flujo básico.
Pos-condiciones	
1.	Se muestra una alerta informando que es obligatorio el llenado de los campos marcados como requeridos.
Flujo alternativo 4.b Usuario repetido.	

Se muestra un mensaje de error informando que ya existe un usuario con ese nombre en el sistema.
Vuelve al paso 3 del flujo básico.
Pos-condiciones
Se muestra un mensaje de error informando que ya existe el usuario en el sistema.
Flujo alterno 4.c No coincide la contraseña.
Se le informa al usuario que no coinciden las contraseñas en los campos correspondientes.
Vuelve al paso 3 del flujo básico.
Pos-condiciones.
Se le informa al usuario que no coinciden los campos contraseña y confirmar contraseña.
Flujo alterno 4.d Contraseña incorrecta.
Se muestra un mensaje con las especificaciones que debe cumplir la contraseña según el perfil seleccionado.
Vuelve al paso 3 del flujo básico.
Pos-condiciones.
1 Se muestra un mensaje informando las especificaciones que debe cumplir la contraseña.
Flujo alternativo 4.e Cancelar.
1. Concluye requisito.
Pos-condiciones.
Concluye requisito.

Validaciones		
1. Todos los campos son obligatorios. 2. No se pueden insertar usuarios que ya existan. 3. La contraseña insertada tiene que coincidir con la insertada en el campo “Confirmar”. 4. El correo introducido debe tener la estructura de un correo. 5. El nombre del usuario no puede contener caracteres especiales exceptuando los números.		
Conceptos	**Usuario**	Visibles en la interfaz: • Usuario. • Contraseña. • Activo. • Correo. • Habilitado. • Bloqueado. Utilizados internamente • último Acceso. • fecha Bloqueado. • Sal. • fecha de expiración. • fecha de credencial. • token de confirmación.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo de interfaz de adicionar usuario.

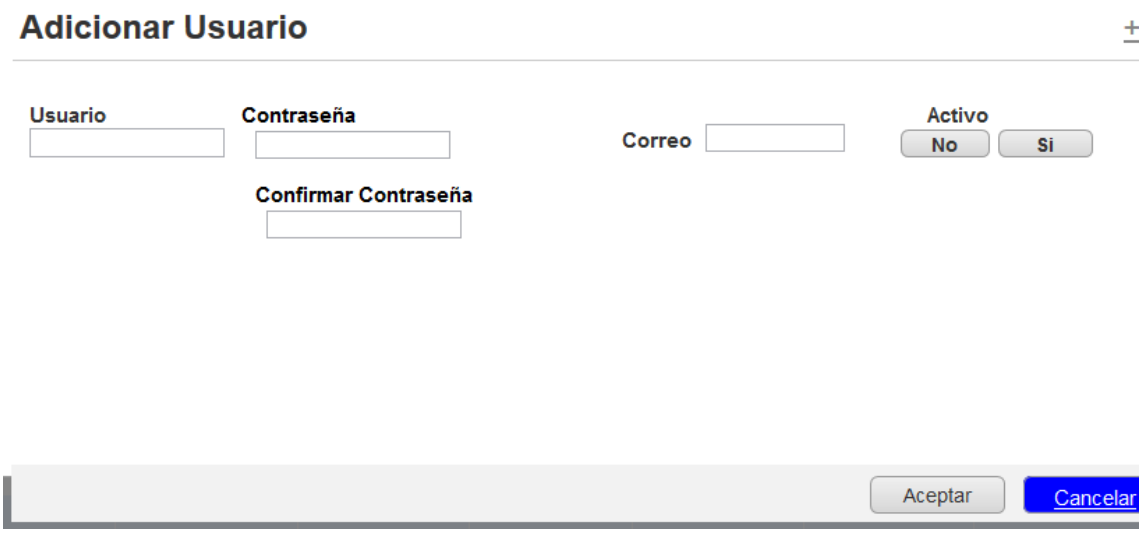

Figura 3 Prototipo de interfaz de adicionar usuario.

2.3.3 Requisitos no funcionales.

Funcionalidad

- **Precisión**

Las respuestas del sistema ante búsquedas corresponderán en un 100% a lo solicitado en los criterios.
Los resultados de operaciones de negocio en el sistema deben ser 100% correctos.
Los datos identificativos del lugar, operación o dato con error, mostrados en los mensajes de error del sistema, deberán ser 100% correctos.

Seguridad

El sistema concederá acceso a cada usuario autenticado solo a las funciones que le estén permitidas, de acuerdo a la configuración del sistema.
El sistema manejará mecanismos de encriptación para las contraseñas de los usuarios.
El sistema registrará las trazas de operaciones realizadas por cada usuario en todo momento.

El sistema garantizará el intercambio seguro de datos electrónicos usando estándares de seguridad para servicios web.

Confiabilidad

- **Madurez**

El sistema no permitirá la entrada de datos incorrectos.
El sistema impondrá campos obligatorios para garantizar la integridad de la información que se introduce por el usuario.
Ninguna información que se haya ingresado en el sistema y se haya asociado a alguna operación será eliminada físicamente de la base de datos.
El sistema contendrá un mecanismo de alertas y avisos sobre cambios de estados y/o realización de operaciones.

- **Tolerancia ante fallos**

Ante el fallo de una funcionalidad del sistema, el resto de las funcionalidades que dependen de esta deberán notificar su imposibilidad de realización debido a la causa original.
El sistema permite detectar fallos internos y notificar al administrador de la ocurrencia de estos.

Usabilidad

- **Comprensibilidad**

Todos los mensajes de error del sistema deberán incluir una descripción textual del error.
El orden de las etiquetas de funcionalidades en el menú responderá a las dependencias de ejecución de negocio.
Las etiquetas de cada funcionalidad y los campos de cada interfaz tendrán títulos asociados a su función de negocio.

Mantenibilidad

- **Diagnosticabilidad**

El sistema deberá poseer un mecanismo de almacenamiento, detección y tratamiento de errores.

- **Flexibilidad**

  El sistema permitirá agregar nuevas funcionalidades o modificar alguna existente sin romper la estructura y consistencia de los componentes.

2.3.4 Validación de requisitos

La validación de requisitos demuestra que la definición de estos responda correctamente a las necesidades del cliente. Esta es una actividad fundamental, ya que un levantamiento de requisitos con errores, que no se detecten a tiempo conduce a resultados inesperados, provocan costos excesivos y gran pérdida de tiempo. Existen diferentes técnicas para la validación de requisitos de software como: Técnica de prototipo, Diseño de casos de prueba y Revisiones Técnicas Formales.
En la validación de los requisitos de software de la propuesta de solución se aplicaron las técnicas antes mencionadas. Se diseñaron los prototipos de interfaces y se presentaron al analista, arquitecto de software para su valoración. Esto permitió que el cliente tuviera una noción de la propuesta de solución y se pudieran hacer las correcciones necesarias antes del diseño e implementación de las funcionalidades del sistema. La generación de casos de prueba permitió especificar los posibles datos de entrada, así como las salidas del sistema generándose un diseño de caso de prueba para cada requisito funcional. La analista del proyecto realizó revisiones técnicas formales a la documentación generada en la disciplina requisitos, lo que permitió realizar las correcciones necesarias antes del diseño y la implementación de la propuesta de solución.

2.4 Diseño del sistema

En el desarrollo o ciclo de vida de un sistema informático, el diseño del sistema constituye un elemento fundamental del mismo. Se centra en proporcionar la funcionalidad del sistema a través de sus diferentes componentes.

2.4.1 Patrones de Diseño Utilizados

En diseño orientado a objetos, GRASP son patrones generales de software para asignación de responsabilidades. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software (Sommerville 2005).
Para el presente trabajo se utilizaron los siguientes patrones GRASP:

- **Patrón Experto:** el patrón fue usado con el objetivo de darle a las clases la responsabilidad necesaria siempre que contaran con la información para cumplirla, logrando así un mejor comportamiento entre las clases, fáciles de comprender y mantener (Larman 1999).
- **Patrón Creador: l**a característica principal de este patrón es que permite identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases (Larman 1999).
- **Patrón Alta Cohesión:** asegura la necesidad de mantener el sistema con un bajo nivel de complejidad, es decir, una clase tiene una responsabilidad moderada en un área funcional y colabora con otras clases para llevar a cabo las tareas (Larman 1999).
- **Patrón Bajo Acoplamiento:** permite mantener bajas dependencias, bajo impacto al cambio e incrementar la reutilización (Larman 1999).

  Además se utiliza el siguiente patrón GOF:
- **Patrón Decorador:** extiende la funcionalidad de un objeto dinámicamente de tal modo que es transparente a sus clientes, utilizando una instancia de una subclase de la clase original que delega las operaciones al objeto original. Provee una alternativa muy flexible para agregar funcionalidad a una clase (Larman 1999).

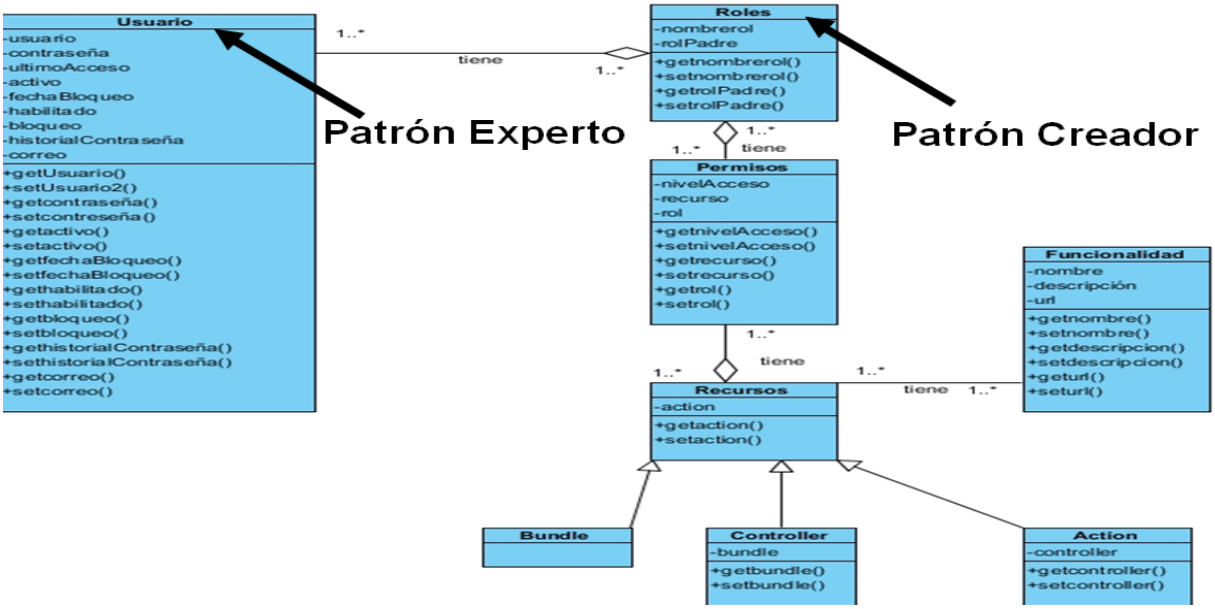

Figura 4 Ejemplos de los patrones de diseño utilizados.

.

Experto: se evidencia en la clase Usuario pues es la que contiene las funcionalidades necesarias para acceder a la información de un usuario.
Creador: este patrón se utilizó en la clase Rol, específicamente en el constructor donde se crea una nueva instancia de la lista de usuarios que tiene ese rol.
Alta cohesión: este patrón se ve reflejado en las clases Manager, donde cada clase tiene la responsabilidad de un área funcional y colabora con otras funcionalidades dentro de la clase para llevar otras funcionalidades.
Bajo acoplamiento: Se evidencia en las clases Manager porque contiene un conjunto de clases con una gran cantidad de responsabilidades y entre ellas no se relacionan, lo cual garantiza un bajo acoplamiento.
Decorador: decora la clase de Rol de Symfony, agregándole nuevas funcionalidades, pasando a constituir la nueva clase Rol de Symfony.

2.4.2 Patrón arquitectónico

El patrón de arquitectura Modelo Vista Controlador MVC (del inglés Model View Controller), permite llevar a cabo una programación multicapa, separando a la interfaz del usuario, la lógica del control y los datos de la aplicación en tres componentes diferentes. Este patrón se ve principalmente de manera más usual en aplicaciones web, donde la vista es la página (html) que se carga en el navegador web y el código que provee de datos dinámicos a la página, el modelo, es el sistema de gestión de base de datos y el controlador representa la lógica del negocio, que está formada por estos tres (Potencier y Zaninotto 2012):

- Modelo: representa toda la información con la que opera la aplicación, o sea su lógica de negocio, responde a las peticiones de estado que vienen de la vista, gestiona el comportamiento y los datos del dominio y responde a instrucciones de cambio de estado provenientes del controlador.
- Vista: gestiona la presentación de la información de la aplicación, en otras palabras convierte el modelo en una página web que facilita al usuario la interacción con ella.

Controlador: es el encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista. La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista), lo que permite un mantenimiento más sencillo de las aplicaciones. El controlador es el encargado de aislar al modelo y a la vista de los detalles del protocolo usado para las

peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica referida a los datos, lo que permite que la vista y las acciones sean independientes del tipo de gestor de bases de datos que la aplicación utiliza por citar un ejemplo.

2.5 Diagramas del diseño

Los diagramas del diseño tienen como objetivo crear los modelos que permitan un entendimiento visual de los requisitos del sistema, y preparar el ambiente para su implementación y prueba.
Los diagramas necesarios para la fase de diseño de la solución propuesta y establecidos en el modelo de desarrollo del centro son:

1. Diagrama de paquetes.
2. Diagrama de clases persistentes.
3. Diagrama de clases con estereotipos web.
4. Diagrama de Base de Datos.

A continuación se muestra un ejemplo de los diagramas realizados.

2.5.1 Diagrama de paquetes

El lenguaje UML ofrece el mecanismo paquete que permite describir los grupos de elementos o subsistemas. Un paquete es un conjunto de cualquier tipo de elementos de un modelo: clases, casos de uso, diagramas de colaboración u otros paquetes (los anidados); es un mecanismo utilizado para agrupar elementos de UML que permite organizar los elementos modelados con UML, facilitando de ésta forma el manejo de los modelos de un sistema complejo (Gutierrez 2012).
Entre sus características se tiene (Gutierrez 2012):

- Permiten dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales.
- En general, pueden tener una interfaz (métodos de clases e interfaces exportadas) y una realización de estas interfaces (clases internas que implementan dichas interfaces).
- Se pueden utilizar para plantear la arquitectura del sistema a nivel macro.

El diagrama de paquetes (Figura 5) de la presente solución cuenta con un paquete fundamental llamado Componente de Administración constituido por 4 paquetes internos; entre los que se encuentran: el paquete Services, el paquete Controlador (contiene el controlador frontal), el paquete Entidad y el paquete Resources. Este paquete está relacionado con 3 paquetes externos: el SeguridadBundle (que es el componente de seguridad de Boson), el FosUserBundle y FrontedBundle.

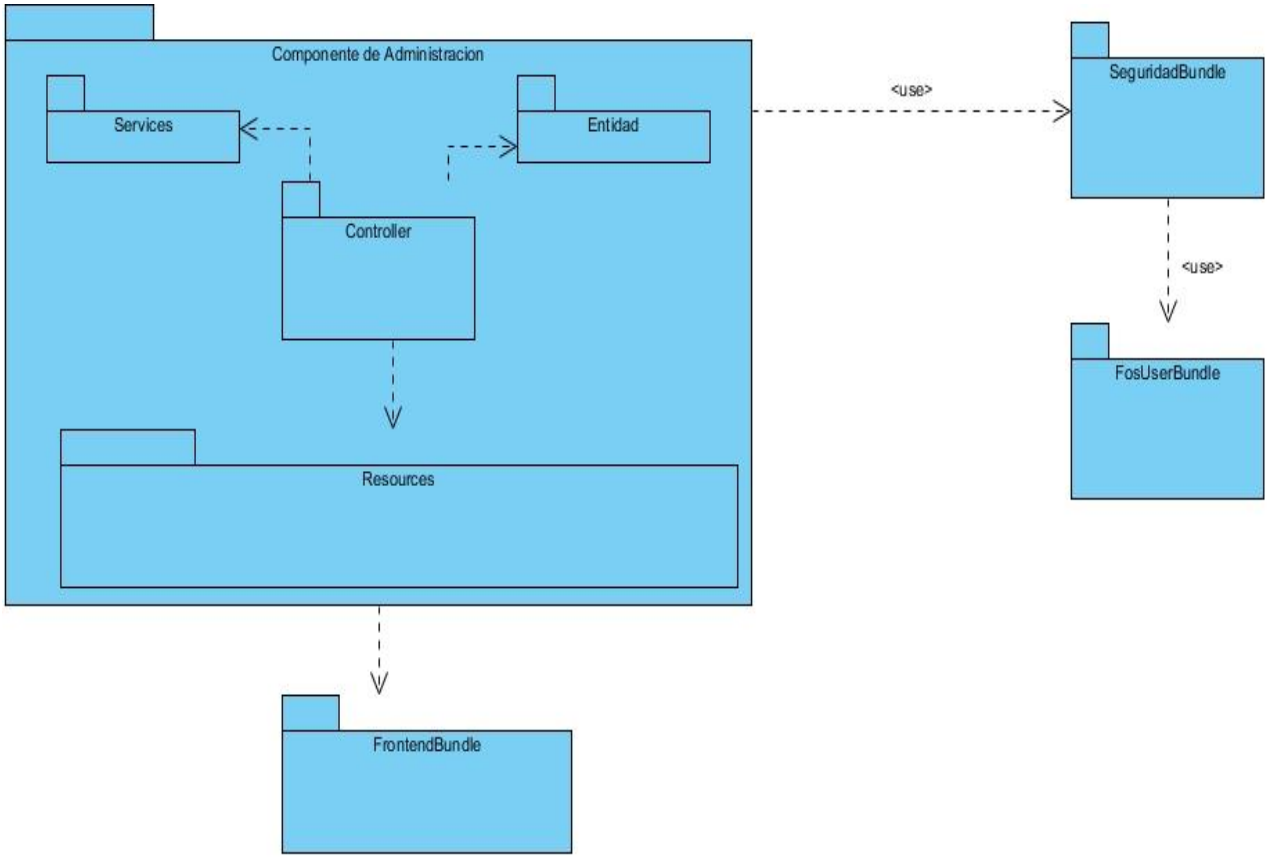

Figura 5 Diagrama de paquetes.

2.5.2 Diagramas de clases persistentes

El diagrama de clases del diseño, describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información: clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información sobre los tipos de atributos, navegabilidad y dependencias. En la figura 6, se muestra el diagrama de clases persistentes del sistema, representando un conjunto de clases, atributos y sus relaciones entre ellas.

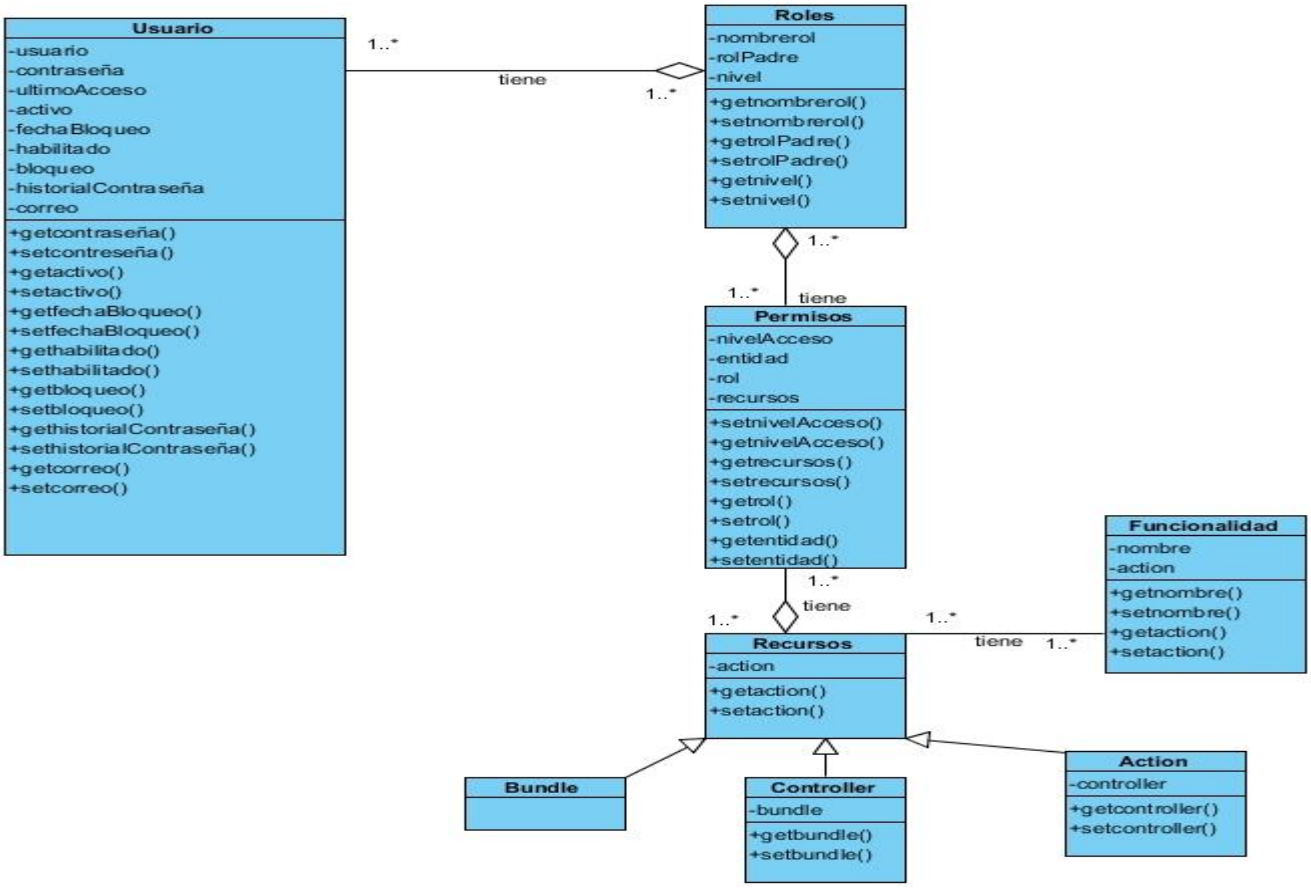

Figura 6 Diagrama de clases.

2.5.3 Diagramas de clases del diseño con estereotipos web

A continuación se describe el diagrama de clases del diseño basado en estereotipos web que se realizó durante el proceso de desarrollo perteneciente al requisito funcional Gestionar Usuario.

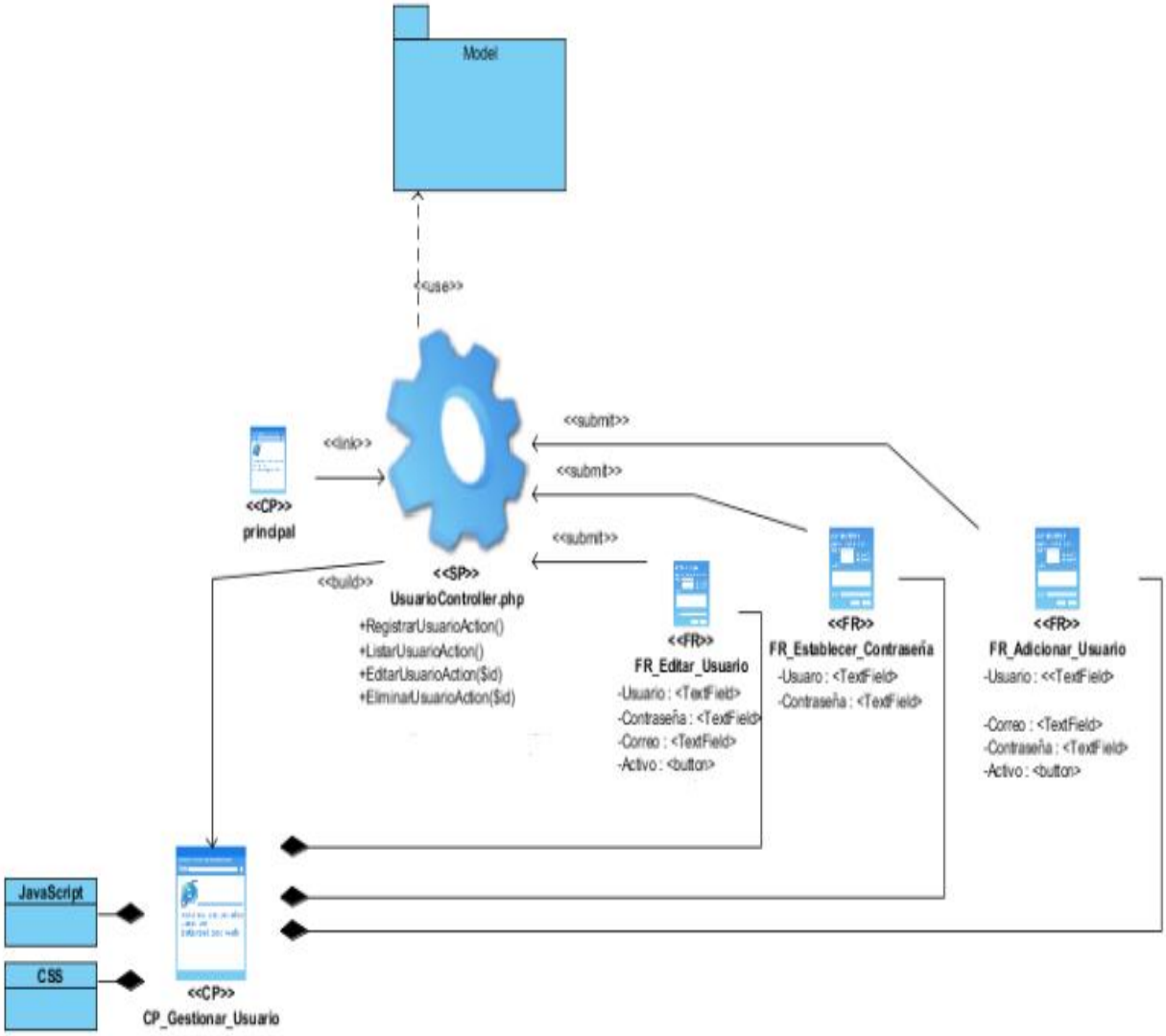

Figura 7 Diagrama de clases del diseño con estereotipos web.

2.5.4 Diagrama de Base de Datos

A continuación se muestra el diagrama de base de datos del sistema, donde sus tablas fundamentales son: usuario, permisos y recursos. Esta tabla permisos une los usuarios con ella y a su vez con los recursos, donde representa lo físico con lo que cuenta el sistema. La Base de Datos se encuentra en la tercera formal normal.

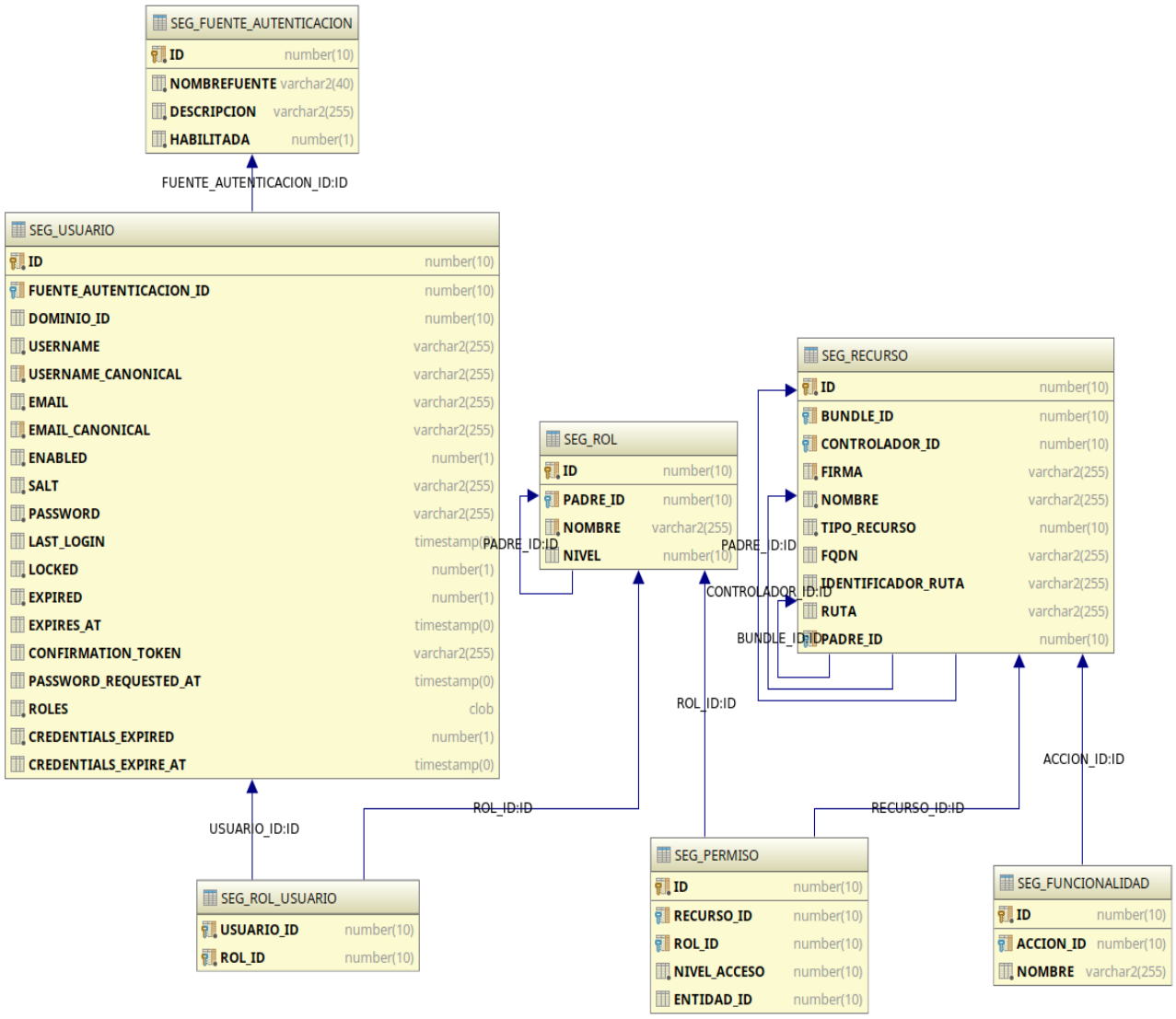

Figura 8 Diagrama de Base de Datos.

2.6 Conclusiones parciales

- En la elaboración de la propuesta de solución se identificaron 20 requisitos funcionales y 18 no funcionales cumpliendo con las necesidades del cliente y sirviendo de base para el diseño de la solución**.**
- Se generaron los artefactos definidos por el Departamento de Soluciones Financieras y Aduanales. Estos artefactos, unido al buen uso de los patrones de diseño: experto, alta cohesión, bajo acoplamiento y creador permitieron sentar las bases para la implementación del software

CAPÍTULO 3. IMPLEMENTACION Y PRUEBA

Introducción

En el presente capítulo se exponen y describen los artefactos de implementación de la solución en cuestión. Se especifican algunos de los estándares de codificación por los que se rigen los programadores en el Departamento de Soluciones Financieras y Aduanales. También se realiza el tratamiento de errores y se aplican pruebas funcionales al sistema para verificar que el comportamiento observado del software cumple con la especificación de los requisitos establecidos en el análisis. Se realizan las distintas pruebas de software para demostrar que el sistema es seguro.

3.1 Modelo de implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema.

3.1.1 Estándar de codificación

El Departamento de Soluciones Financieras y Aduanales del centro CEIGE, presenta su propia propuesta de estándar de codificación para todas las aplicaciones que se desarrollan en el mismo. Este comprende todo el código generado bajo la tecnología y el lenguaje PHP y que utilicen la arquitectura regida por la utilización del framework arquitectónico Symfony. Para el desarrollo del componente de Administración de la VUA, se utilizó este estándar de codificación. A continuación se citan algunos de los aspectos relevantes que presenta dicho documento («Estándar de codificación» 2017).

En las clases, los nombres deben estar expresados en notación UpperCamelCase[11] y no se deben utilizar guiones bajos "_", para variables, funciones y nombres de métodos; expresando con claridad cuál es el alcance y la responsabilidad de la clase.

Cuando se incluyen abreviaturas en mayúsculas no se debe incluir su nombre completo, sino que se debe utilizar la primera letra en mayúscula y el resto en minúsculas. Como se muestra a continuación:

[11] UpperCamelCase: consiste en escribir frases o palabras compuestas, eliminando los espacios intermedios y poniendo en mayúscula la primera letra de cada palabra, incluyendo la primera letra de la frase.

Ejemplo:

GetHtmlStatistic //Correcto

GetHTMLStatistic //Incorrecto

Todas las funciones definidas por los desarrolladores, deben seguir la nomenclatura CamelCase. Además, los nombres de las funciones, deben dejar reflejado claramente cuál es la acción que realiza el mismo. Los nombres de las variables deben expresar claramente el contenido de la misma. Pueden estar referidas en singular o plural.

En caso de ser bundles del negocio deben tener nombres que dejen reflejado bien claramente cuál es el propósito del mismo, en una palabra o siglas. En caso de ser mediante siglas se pondrán todas en mayúsculas. Se debe evitar mientras sea posible la utilización de palabras compuestas o la utilización de varias palabras, en caso de palabras compuestas se utilizará la notación **UpperCamelCase**[2]. El nombre oficial del bundle debe estar compuesto por el nombre del vendor seguido por el nombre del bundle. Por ejemplo, si el vendor es VU y se tiene un bundle denominado ServiciosBundle, el nombre final del bundle debe ser VUServiciosBundle.

Para más información sobre todos los estándares de codificación que se utiliza para en el CEIGE ver CIG-ADU-N-VUA_Estándares de codificación en el Expediente de Proyecto.

3.1.2 Tratamiento de errores

En el desarrollo de software el tratamiento de errores garantiza el correcto funcionamiento del sistema, logrando no afectar la calidad del mismo. Es por ello fundamental identificar y controlar los posibles problemas que puedan presentarse a la hora de interactuar con el software**.**

En el desarrollo de la solución se puede apreciar el tratamiento de errores al procesar cada uno de los documentos, donde los datos son validados antes de ser insertados o modificados en la base de datos y enviados hacía el sistema externo, comprobando que cumplen con los requisitos establecidos.

En caso de que exista algún error, es notificado al usuario para que pueda corregirlos. Además, se realizaron validaciones en las vistas, con el objetivo de garantizar la entrada correcta de los datos a la aplicación y evitando que se introduzcan datos incorrectos en los diferentes campos de los formularios. Además, se realiza tratamiento de errores con la utilización de los formularios generados por Symfony que permiten validar los campos introducidos en cuanto a tipo y número a la hora de insertar o modificar valores de la base de datos. Otro tratamiento de errores lo constituye la captura de las excepciones que puedan ocurrir al insertar valores a la base de datos, a las que se le aplica el mismo procedimiento que a los errores de las validaciones.

3.1.3 Diagrama de componentes

El diagrama de componentes describe la descomposición física del sistema de software (y, eventualmente, de su entorno organizativo) en componentes, a efectos de construcción y funcionamiento. La descomposición del diagrama de componentes se realiza en términos de componentes y de relaciones entre los mismos (Falgueras 2002).

La figura (9) muestra que el FosUserBundle provee servicios al componente Boson donde es utilizado por el Componente de Seguridad que es utilizado por la Ventanilla Única Aduanera.

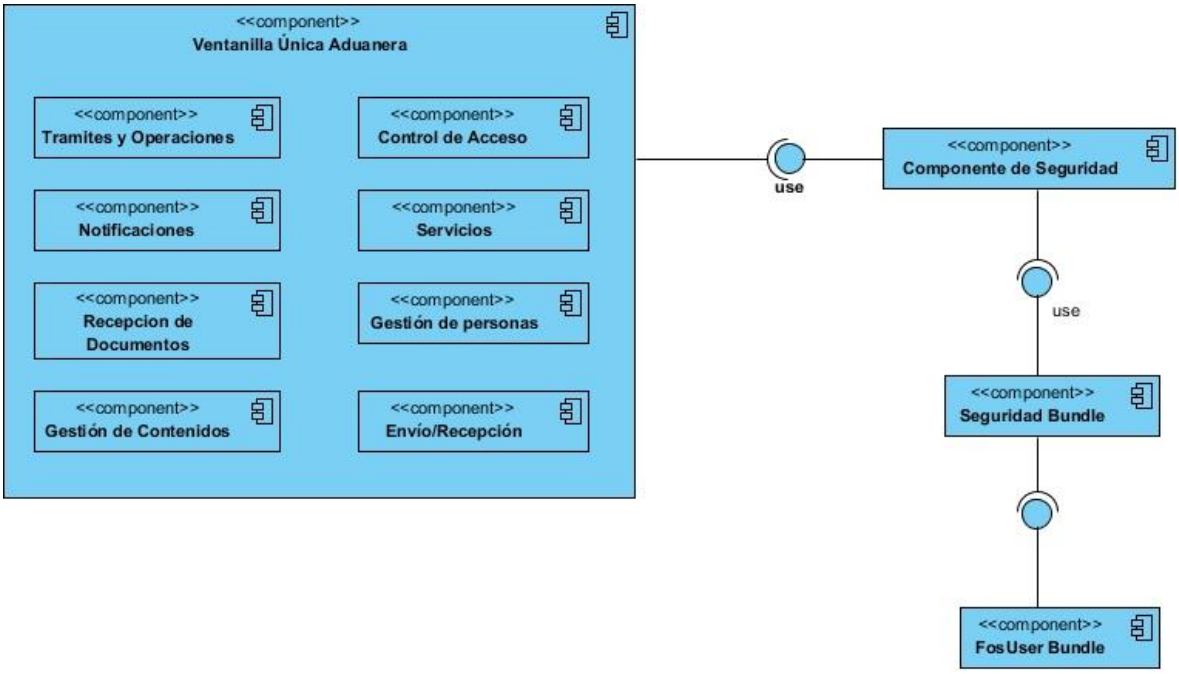

Figura 9 Diagrama de componente.

3.1.4 Implementación

La interfaz de usuario es el medio de comunicación entre el usuario y el sistema y debe cumplir con las tres reglas siguientes: dar control al usuario, reducir la carga de memoria del usuario y debe ser consistente. El cumplimiento de estas reglas permite desarrollar interfaces flexibles, en las que el usuario no realice acciones innecesarias y no deseadas, así como la interrupción de una secuencia de acciones en el instante requerido por el usuario. Además, deben desglosar la información de forma progresiva a través de una estructura organizada, orientar al usuario en las tareas a desarrollar y que esas tareas se realicen en el contexto adecuado (Pressman 1988).

A continuación se describe la interfaz de usuario del módulo de administración y seguridad que se desarrolla como parte de este proyecto y que se integra en la plataforma de Ventanilla Única Aduanera.

Figura 10 Interfaz de la Gestión de Roles.

La Figura (10) muestra la interfaz de la Gestión de Roles donde muestra 2 vistas, la de árbol y listado. En la vista de árbol se utiliza para mostrar la jerarquía de roles. La vista de listado es importante a la hora de que existan muchos roles, esta facilita la búsqueda.

En la Figura (11) se puede apreciar como adicionar un rol.

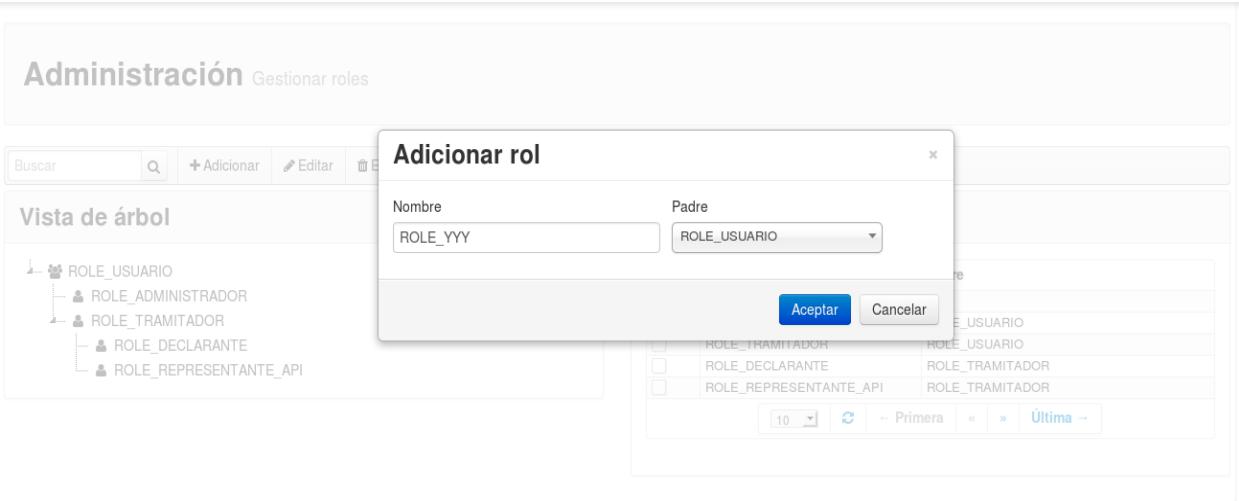

Figura 11 Interfaz de Adicionar rol.

Para el componente fue necesario implementar un sistema de permisos que se ajustara a las necesidades del negocio, para lograr este objetivo se implementa un sistema de jerarquía que crea una estructura simulando la de una institución real.

El árbol de jerarquía de permisos se hizo de forma invertida a la realizada en la versión 1.0, garantizando que el nodo raíz tenga la menor cantidad posible de accesos. En la figura (12) se muestra como a un rol se le aceptan o deniegan los permisos y los roles que pertenecen a él tienen los mismos permisos que el nodo padre.

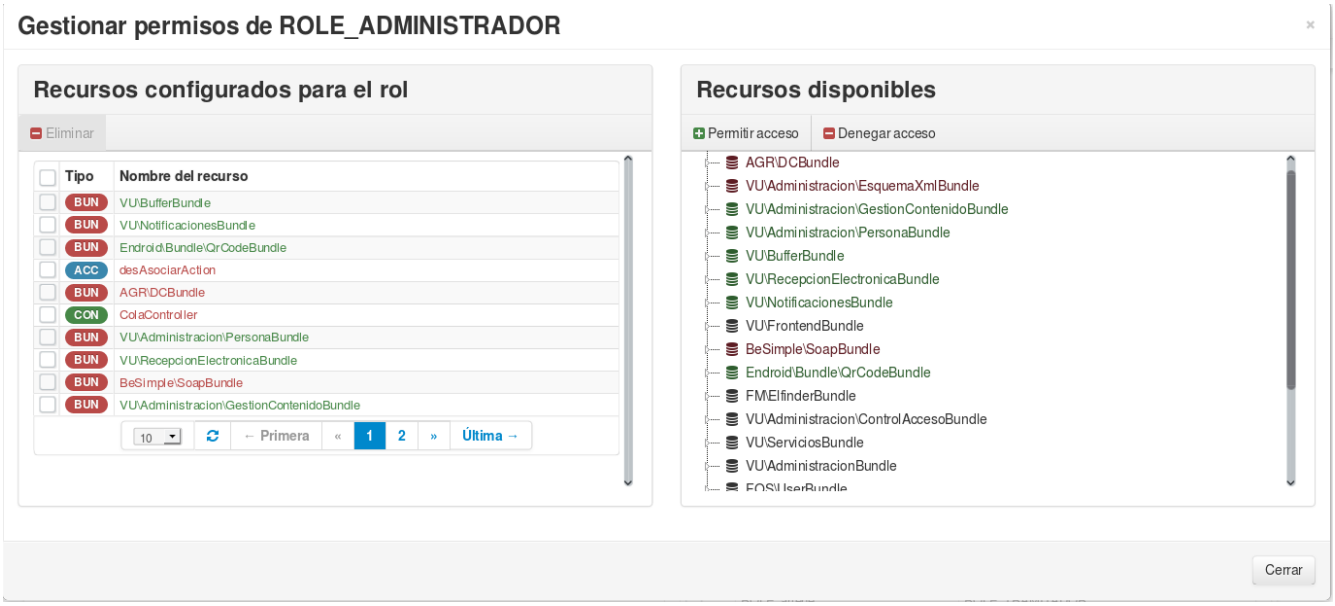

Figura 12 Interfaz de la Gestionar permisos para un rol.

3.1.5 Diagrama de despliegue

El diagrama de despliegue permite (en inglés, deployment) mostrar la arquitectura en tiempo de ejecución del sistema respecto a hardware y software. El diagrama de despliegue se utiliza en el diseño y la implementación. Se pueden distinguir componentes (como los del diagrama de componentes) y nodos, así como las relaciones entre todos éstos. Es más limitado que el diagrama de componentes, en el sentido de que representa la estructura del sistema sólo en tiempo de ejecución, pero no en tiempo de desarrollo o compilación. Sin embargo, resulta más amplio en el sentido de que puede contener más clases de elementos (Falgueras 2002).A continuación se muestra el diagrama de despliegue:

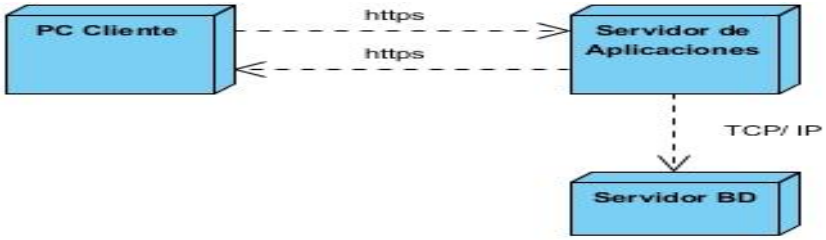

Figura 13 Diagrama de despliegue.

A continuación se procede a describir los nodos que componen el diagrama de despliegue:

PC Cliente: cliente desde el cual se accederá al Componente de Administración.
Servidor de Aplicaciones: servirá de plataforma base para el despliegue de los proyectos encargados de manejar la lógica del negocio y el sistema.
Servidor de Base de Datos: servidor donde se alojan las bases de datos que almacena Componente de Administración.

3.2 Pruebas de software

Una vez generado el código fuente, el software debe probarse para descubrir (y corregir) tantos errores como sea posible antes de entregarlo al cliente. La meta es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores; ahí es donde entran en escena las técnicas de prueba de software. Dichas técnicas proporcionan lineamientos sistemáticos para diseñar pruebas que (Pressman 1988):
1) revisen la lógica interna y las interfaces de todo componente de software.
2) revisen los dominios de entrada y salida del programa para descubrir errores en el funcionamiento, comportamiento y rendimiento del programa.

3.2.1 Pruebas funcionales aplicadas al Componente de Administración

Las pruebas funcionales están basadas en técnicas de caja negra, donde tienen una visión externa del producto software y no están centradas en el código fuente. Estas pruebas están centradas en analizar la funcionalidad. Por lo tanto los casos de prueba se basan en las diferentes entradas que pueden recibir el software y sus correspondientes valores de salida. Estas técnicas se pueden aplicar a cualquiera de los niveles de pruebas (unitarias, integración, aceptación) con diferentes niveles de abstracción en la definición de los casos de prueba (Yagüe y Garbajosa 2009).
Las pruebas de caja negra intentan encontrar errores en las categorías siguientes (Pressman 1988):
1) funciones incorrectas o faltantes.

2) errores de interfaz.

3) errores en las estructuras de datos o en el acceso a bases de datos externas.

4) errores de comportamiento o rendimiento.

5) errores de inicialización y terminación.

Aplicación de la prueba.

En el Componente de Administración de la VUA, se realizaron pruebas de caja negra, con el objetivo de demostrar que las funciones son operativas. Estas se llevaron a cabo sobre la interfaz del software, centrándose en todo momento sobre los requisitos funcionales. Para ello se elaboran diseños de casos de prueba, como el siguiente:

Diseño de caso de prueba correspondiente al requisito funcional “Adicionar Usuario”.

Condiciones de ejecución

1. El administrador debe tener los permisos necesarios y estar autenticado en el sistema.
2. El administrador debe seleccionar del menú la opción “**Control de Acceso**”.
3. El administrador selecciona la opción “**Gestionar Usuarios**”.

SC Gestionar Usuarios

Escenario	Descripción	Nombre del usuario	Contraseña	Correo	Inactivo/Activo	Habilitado/Bloqueado	Respuesta del sistema
EC 1.1 Buscar Usuario.	Permite listar los usuarios según el criterio de búsqueda especificado.	V	NP	NP	NP	NP	Muestra todos los usuarios que contengan la cadena especificada.
		usuarioarl					
		NP	NP	V	NP	NP	
				prueba@uci.cu			
EC 1.2 Adicionar usuario.	Permite adicionar usuarios en el listado de estos. Todos los campos son obligatorios excluyendo el correo.	V	V	V	V	NP	Se actualiza el listado de los usuarios.
		usuarioprueba	123Prueba*123	prueba@uci.cu	Activo		
		I	V	V	I	NP	El sistema muestra un mensaje informando que debe corregir los campos inválidos del formulario. Formato válido del nombre: -empieza siempre con letras que pueden ser mayúsculas o minúsculas seguido de caracteres alfanuméricos.
		8prueba	123Prueba*123	Vacío	Vacío		

Figura 14 Diseño del caso de prueba Adicionar usuario.

Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Usuario	Campo de texto	No	Tiene que cumplir con la siguientes condiciones: -empieza siempre con letras que pueden ser mayúsculas o minúsculas seguido de caracteres alfanuméricos. -no permite ningún caracter especial como: "@#$_/* .
2	Nombre del usuario	Campo de texto	No	Tiene que cumplir con la siguientes condiciones: -empieza siempre con letras que pueden ser mayúsculas o minúsculas seguido de caracteres alfanuméricos. -no permite ningún caracter especial como: "@#$_/* . -la longitud debe ser mayor de 4 caracteres.
3	Contraseña	Campo de texto	No	Permite cualquier caracter y solamente hasta 250 caracteres.
4	Confirmar	Campo de texto	No	Permite cualquier caracter y solamente hasta 250 caracteres.
5	Correo	Campo de texto	SI	Caracteres alfanuméricos [A..Z, a..z, 0..9] y de los caracteres especiales
6	Inactivo/Activo	Campo de selección	No	
7	Habilitado/Bloqueado	Campo de selección	No	

Figura 15 Descripción de las variables.

Resultados de las pruebas aplicadas.

La aplicación de las pruebas funcionales a la solución desarrollada, permitió encontrar no conformidades en el sistema.

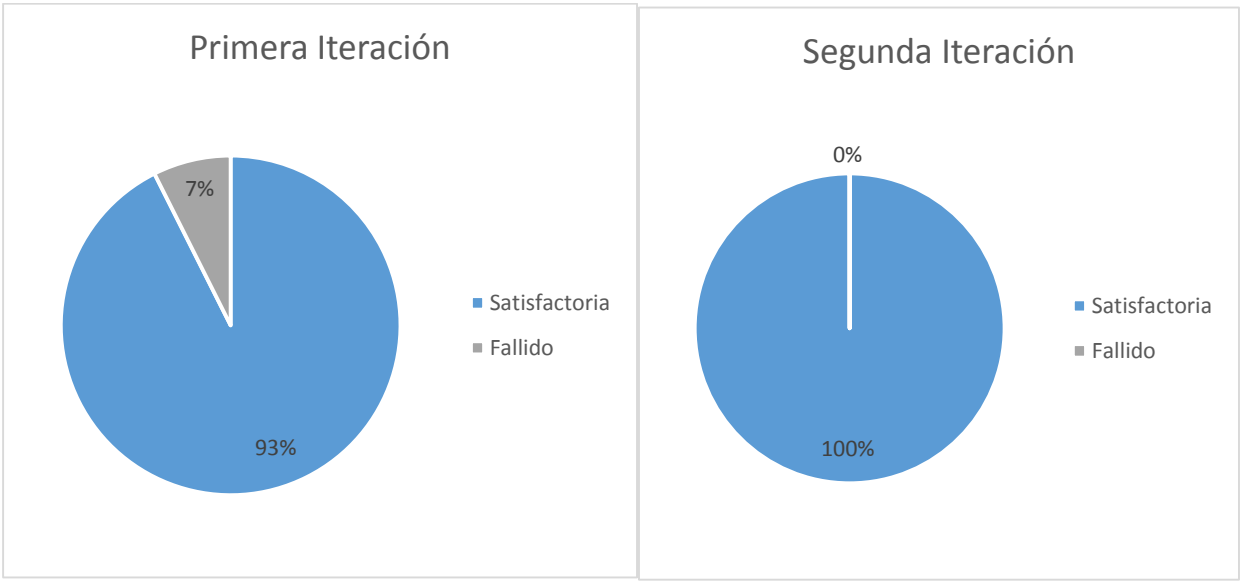

Figura 16 No conformidades encontradas en el sistema.

Como se muestra en la figura 16 en la primera iteración fueron encontradas 8 no conformidades representando 7% de total. Para la segunda iteración fueron resueltos estos errores detectados.

3.2.2 Prueba de aceptación

Las pruebas de aceptación representan aquella fase del ciclo de vida de desarrollo de software en el que el equipo de desarrollo y el área usuaria de un sistema de información tienen que garantizar que

el sistema desarrollado se corresponde con los requerimientos definidos (González et al. 2014). Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Sánchez 2014).

Resultados de la prueba de aceptación

Las pruebas de aceptación fueron realizadas por el arquitecto del proyecto y el cliente. Para ello se ejecutaron pruebas de validación al sistema desarrollado para validar el cumplimiento de los requisitos aprobados, resultando todas satisfactorias. Después de probada la implementación de los requisitos, el cliente emitió el acta de aceptación de software.

3.3 Validación de la investigación.

3.3.1 Métricas del proceso de seguridad de aplicaciones

El objetivo de las métricas del proceso de seguridad de aplicaciones es determinar qué tan bien los procesos de seguridad de aplicaciones de la organización cumplen los requisitos de seguridad definidos por las políticas de seguridad y los estándares técnicos (OWASP 2012).

Desde la perspectiva de la cobertura del proceso, uno de los objetivos de estas métricas puede ser informar acerca de la cobertura del proceso de seguridad de aplicaciones, tales como medir el modo en que las aplicaciones entran dentro del alcance de las evaluaciones de seguridad de aplicaciones, para identificar potenciales huecos en las evaluaciones de vulnerabilidad basadas en los tipos de aplicaciones y en los requisitos de seguridad de aplicaciones. Otra medición importante para la verificación de la seguridad de aplicaciones es medir el tiempo cuando los procesos de seguridad de aplicaciones son planificados o cuando son ejecutados, para identificar demoras potenciales en la revisión del código de seguridad, análisis de código fuente estáticos, hacking ético y procesos de pruebas de penetración (OWASP 2012).

- **Prueba de penetración.**

Pruebas de penetración llamadas también pentesting o hacking ético facilitan reconocer vulnerabilidades en un medio informático, permite evaluar el riesgo y estimar los procedimientos de seguridad actuales en la empresa, muestra donde falla la seguridad o es escasa y se puede utilizar para justificar la necesidad de una actualización, un mayor presupuesto para seguridad o para validar la valoración de los riesgos (Guerrero Erazo, Lasso Garces y Legarda Muñoz 2015).

Las pruebas de penetración son un conjunto de técnicas utilizadas para realizar una evaluación integral de las vulnerabilidades de los sistemas que se encuentran dentro de una red. Estas pruebas involucran la utilización de herramientas y técnicas, tales como: escáneres, crackers de contraseñas e ingeniería social (López 2004).

Objetivos de las pruebas de penetración (López 2004)

- Determinar la efectividad de las medidas de seguridad de la organización.
- Dar apoyo a la confidencialidad y protección de los recursos.
- Obtener un estado de seguridad de la información desde la perspectiva de un hacker.
- Conocer las fallas de seguridad de los activos de información.

Características de las pruebas de penetración (López 2004)**.**

- No se realiza ningún cambio en los sistemas o redes de la organización.
- No se impacta las capacidades de procesamiento durante las horas laborables o períodos críticos.
- Es obtenida autorización por parte de la empresa antes de ejecutar las pruebas.
- Es de carácter confidencial.
- No ejecuta ataques innecesarios.

Entre las diversas herramientas para el análisis de vulnerabilidades se escoge Acunetix:

Acunetix es un escáner de vulnerabilidades de aplicaciones web. La herramienta está diseñada para encontrar agujeros de seguridad en las aplicaciones web de la organización que un atacante podría aprovechar para obtener acceso a los sistemas y datos. La herramienta provee sugerencias para mitigar las vulnerabilidades identificadas y puede utilizarse para incrementar la seguridad de servidores web o de las aplicaciones que se analizan («Acunetix» 2017).

Acunetix Web Vulnerabilidad Scanner es una herramienta que será capaz de escanear sitios web en busca de posibles fallos de seguridad que puedan poner en peligro la integridad de la página publicada en Internet (Miranda 2011).

Esta aplicación ejecuta una serie de pruebas, totalmente configurables por el usuario, para identificar las vulnerabilidades tanto en la programación de la página como en la configuración del servidor detecta técnicas de hacking como pueden ser ataques de ejecución de código y de autentificación. Destaca por su completo escáner para examinar scripts en una página web, permitiendo detectar técnicas de hacking como, por ejemplo, inyección SQL, inserción de secuencias de comandos mejor conocido como Cross-Site Scripting por sus siglas en inglés (XSS), ataques de ejecución de código y ataques de autentificación. Además de por su completo conjunto de herramientas: editor HTTP, Sniffer HTTP,

Flooder HTTP, encriptador de palabras, control de vulnerabilidades, SQL, etcétera («Seguridad Informática» 2017).

Resultados de las pruebas de penetración.

Para la aplicación de las pruebas de penetración con la herramienta Acunetix se realizó la primera iteración a la versión 1.0 del componente de administración y seguridad. En ella se hizo un escaneo obteniéndose estos resultados:

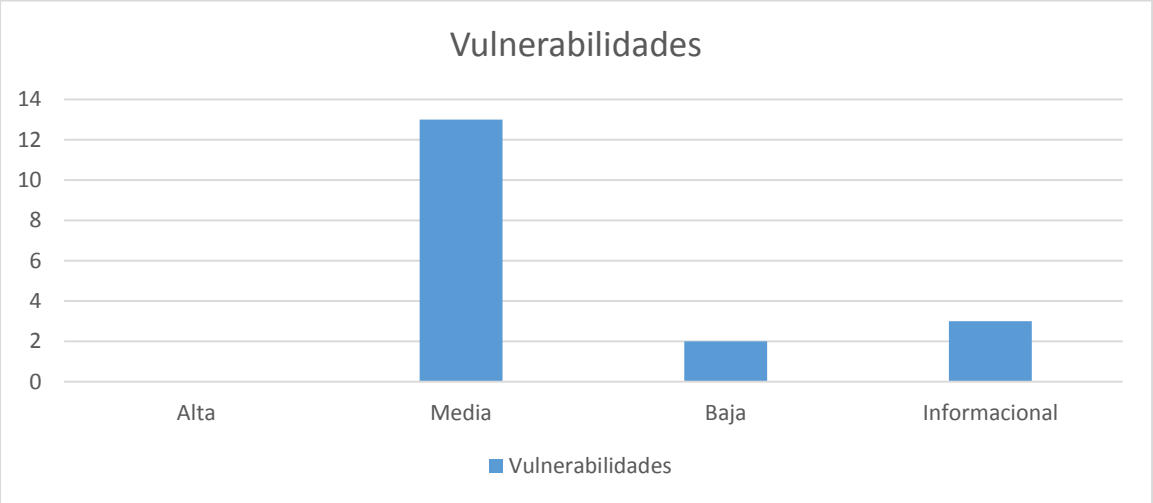

Figura 17 Reporte de vulnerabilidades.

Como se muestra en la figura (17) se entraron un total de 18 vulnerabilidades, 13 nivel medio, 2 bajo y 3 informacional. Entre las principales vulnerabilidades se encuentran:

- Formulario HTML sin protección CSRF.
- Las credenciales de usuario se envían en texto claro.
- Lista de directorio de virtual host.
- Falta el encabezado Opciones de X-Frame.
- El método OPTIONS está habilitado.

En el trabajo en la versión 2.0 del Componente de administración y seguridad para eliminar estas debilidades se llevaron a cabo las siguientes soluciones reflejándose en la siguiente tabla.

Tabla 4 Soluciones de las vulnerabilidades para la versión 2.0 del componente de seguridad.

Vulnerabilidades	**Soluciones para la versión 2.0 del componente de seguridad**

Formulario HTML sin protección CSRF.	Se incluye en token CSRF que provee Symfony como se muestra en el siguiente código: Form_widget (form_Usuariotoken).
Las credenciales de usuario se envían en texto plano.	Se resuelve implementando el servidor de aplicaciones para que utilice HTTPS.
Lista de directorio de virtual host.	En el archivo de configuración htaccess se escribe Options-Index.
Falta el encabezado Opciones de X-Frame.	En el archivo de configuración htaccess se escribe Header set x-Frame Options Deny.

Obteniéndose en la segunda iteración el resultado que se muestra en la figura (18)

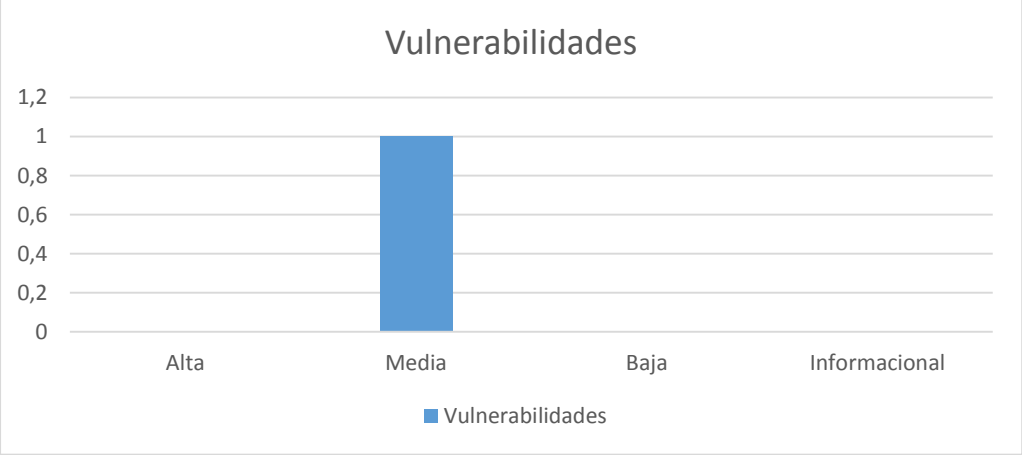

Figura 18 Reporte de vulnerabilidades para la versión 2.0.

La vulnerabilidad detectada se va del alcance del proyecto, esto implicaría un cambio en la versión de php. Para darle solución traería un cambio en los servidores de la aduana y esto se hace difícil debido a que tiene varios sistemas que están utilizando esa tecnología específica.

Con los resultados obtenidos con la aplicación de la pruebas de penetración se evidencia que el sistema realizado en la versión 2.0 es más seguro que el anterior.

3.3.2 Métricas de Usabilidad para la Eficiencia definida por ISO/IEC 9126-4

La eficiencia se mide en términos de tiempo de tarea. Es decir, el tiempo (en segundos y / o minutos) que el participante toma para completar una tarea con éxito. El tiempo que se tarda en completar una tarea se puede calcular simplemente restando el tiempo de inicio del tiempo de finalización, como se muestra en la siguiente ecuación («Usability Metrics» 2017):

$$= \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{NR}$$

Eficiencia basada en el tiempo

Donde:

N=El número total de tareas (metas)

R=Número de usuarios

Nij=El resultado de la tarea i por el usuario j

Si el usuario completa con éxito la tarea, entonces Nij=1, si no, entonces Nij=0

Tij = Tiempo empleado por el usuario j para completar la tarea i. Si la tarea no se completa correctamente, el tiempo se mide hasta el momento en que el usuario abandona la tarea.

A continuación se muestran los resultados de aplicar la Métrica de Usabilidad para la Eficiencia basada en el tiempo, aplicada a la versión 1.0 y 2.0.

Aplicación de la métrica para la versión 1.0

Las tareas a realizar son:
1. Registrar funcionalidad.
2. Crear rol.
3. Crear grupo.
4. Crear un usuario y asignar al grupo.

El número total de tareas es N=4
Para realizar la prueba se toma como muestra 2 usuarios, donde R=2

Tabla 5 Tiempo de los usuarios en realizar las tareas para la versión 1.0.

Tareas realizadas	**Tiempo en realizar las tareas de usuario1**	**Tiempo en realizar las tareas de usuario2**
Registrar funcionalidad	3	4
Crear rol	2	3
Crear grupo	1	2
Crear un usuario y asignar al grupo	2	3

Se sustituye en la fórmula para calcular:

Eficiencia basada en el tiempo =1/3+1/2+1/1+1/2+1/4+1/3+1/2+1/3=0.46

4*2

- **Aplicación de la métrica para la versión 2.0**

El número total de tareas es N=4

Para realizar la prueba se toma como muestra 2 usuarios, donde R=2

Tabla 6 Tiempo de los usuarios en realizar las tareas para la versión 2.0.

Tareas realizadas	**Tiempo en realizar las tareas de usuario1**	**Tiempo en realizar las tareas de usuario2**
Sincronizar recursos	2	**3**
Crear rol	1	**2**
Crear un usuario y asignar al grupo	1	**2**

- **Eficiencia basada en el tiempo**=1/2+1/1+1/1+1/3+1/2+1/2 =0.66
  4*2

Aplicando la métrica de usabilidad para la eficiencia basada en tiempo, a los componentes de administración 1.0 y 2.0, arroja valores de 0.44 y 0.66 respectivamente. Se comprueba con esta aplicación que la versión 2.0 es más eficiente que la anterior. Esta métrica arroja que el componente 2.0 puede realizar un 20% más de tareas que la versión anterior utilizando el mismo tiempo.

3.4 Aporte social del trabajo

El trabajo tiene un aporte e impacto fundamental en la Seguridad Nacional, pues con la realización de este componente de seguridad permite que el administrador sea más eficaz en el control de los permisos. Este componente propicia una mayor seguridad para la Ventanilla Única Aduanera, pues los daños producidos por la falta de control pueden causar pérdidas económicas.

3.5 Conclusiones parciales

- La implementación de la solución estuvo guiada por varios estándares de codificación permitiendo un entendimiento más claro del código generado.
- La aplicación de las pruebas al sistema permitió validar la investigación, donde se realizaron pruebas funcionales, de penetración y de aceptación.
- La aplicación de las pruebas funcionales a la solución desarrollada, permitió encontrar no conformidades, dando la posibilidad de que pudieran ser corregidos a tiempo y que se obtuviera un componente que responda completamente a los requisitos funcionales.
- En forma general la aplicación de las pruebas se comprobó que la solución cumple con los objetivos propuestos y con el Certificado de Aceptación del cliente, se demostró la satisfacción de los mismos.

CONCLUSIONES GENERALES

- El estudio de componentes de seguridad, la seguridad de la información y el control de acceso en aplicaciones web, arrojó que el componente Boson presenta características que se ajustan a las necesidades del sistema a desarrollar.
- Del estudio de soluciones similares y análisis con los clientes se obtuvo como resultado un total de 20 requisitos funcionales y 18 requisitos no funcionales.
- Con la implementación se obtuvo un componente que permite adaptar la Ventanilla Única al negocio de la Aduana, cumpliendo con los requisitos del cliente.
- La integración del componente de administración y seguridad, brindará mayor facilidad en la gestión de usuarios, permisos, recursos y funcionalidades. Logrando una mayor seguridad en el sistema.
- La aplicación de las pruebas funcionales a la solución desarrollada permitió encontrar 8 no conformidades, las cuales fueron corregidos a tiempo, obteniéndose un componente que responda completamente a los requisitos funcionales.
- La aplicación de la pruebas de penetración arrojó 18 vulnerabilidades en la versión 1.0 y 1 vulnerabilidad en la versión 2.0 del componente de administración y seguridad. Esta prueba evidencia que el sistema realizado en la versión 2.0 es más seguro que el anterior.
- Con las pruebas de aceptación fue validado el cumplimiento de los requisitos aprobados, emitiendo el cliente acta de aceptación de software.
- Las métricas de usabilidad para la eficiencia basadas en el tiempo demuestran que el sistema en su versión 2.0 es más eficiente que el anterior con un valor de 0.66.
- Al finalizar la investigación se logró cumplir el objetivo trazado de manera satisfactoria, obteniendo como resultado el componente de administración y seguridad aduanera de la Ventanilla Única.

RECOMENDACIONES

- Desplegar el sistema en una versión de PHP superiores a 5.6.
- Implementar y configurar el protocolo seguro HTTPS para el acceso al sistema.

REFERENCIAS BIBLIOGRÁFICAS

Acunetix. [en línea], 2017. [Consulta: 30 mayo 2017]. Disponible en: https://www.datasec-soft.com/es/acunetix.

ALVAREZ, M.A., 2010. Manual de jQuery. *Recuperado el*, vol. 17.

CANO, J.J., 2017. JOnline: La Gerencia de la Seguridad de la Información: Evolución y Retos Emergentes. *ISACA* [en línea]. [Consulta: 30 mayo 2017]. Disponible en: https://www.isaca.org/Journal/archives/2011/Volume-5/Pages/JOnline-La-Gerencia-de-la-Seguridad-de-la-Informacion-Evolucion-y-Retos-Emergentes.aspx.

CASTELLÀ-ROCA, J., FELGUERA, A., MARTÍNEZ-BALLESTÉ, A., ROMERO, J.M.P., SOLANAS, A. y GALICIA, A.V., 2012. *Identidad digital*. S.l.: UOC.

CEVALLOS CADENA, C.E., 2015. *Estudio comparativo entre las metodologías de desarrollo de software dsdm y crystal: caso práctico sitio web para la generación de pedidos de soluciones informáticas, para la carrera de ingeniería en informática y sistemas computacionales de la Universidad Técnica de Cotopaxi durante el año 2014*. S.l.: LATACUNGA/UTC/2015.

DOMÍNGUEZ, L., ERNESTO, J., SOBERANES, A., JUÁREZ LANDÍN, C. y RUEDA PAZ, J., 2015. Sistema Detector de Intrusiones ocupando una red Neuronal Artificial. ,

EDGAR D. SALAZAR T, 2012. *Pruebas de Seguridad en aplicaciones web segun OWASP* [en línea]. S.l.: s.n. [Consulta: 4 mayo 2017]. Disponible en: https://www.owasp.org/images/2/2f/OWASP_SUSCERTE.pdf.

Estándar de codificación, 2017. 2017. S.l.: s.n.

FALGUERAS, B.C., 2002. *Ingeniería del software*. S.l.: Editorial UOC. ISBN 84-8429-793-4.

GONZÁLEZ, J.F.P., MAYO, F.J.D., RODRÍGUEZ, J.J.G. y CUARESMA, M.J.E., 2014. Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental. *Ibersid: revista de sistemas de información y documentación*, vol. 8, pp. 73-80.

GUERRERO ERAZO, H.A., LASSO GARCES, L.A. y LEGARDA MUÑOZ, P.A., 2015. Identificación de vulnerabilidades de seguridad en el control de acceso al sistema de gestión documental, mediante pruebas de testeo de red en la empresa ingelec SAS. ,

GUTIERREZ, D., 2012. *UML_clase_05_UML_paquetes.pdf* [en línea]. S.l.: s.n. [Consulta: 4 mayo 2017]. Disponible en: http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.

ISO. [en línea], 2017. [Consulta: 30 mayo 2017]. Disponible en: https://www.iso.org/standard/54534.html.

LARMAN, C., 1999. *UML y Patrones*. S.l.: Pearson. ISBN 84-205-3438-2.

LEÓN, amón B. de y LÓPEZ., A.C., 2013. Componente de Administración del Sistema Ventanilla Única para el Comercio Exterior de Cuba. [en línea]. [Consulta: 4 mayo 2017]. Disponible en: file:///E:/TESIS/TESIS/tesis/version1.0.pdf.

LÓPEZ, S.G., 2004. *Estudio de pruebas de penetración para la comprobación de la seguridad de la información empresarial* [en línea]. S.l.: s.n. [Consulta: 4 mayo 2017]. Disponible en: file:///E:/TESIS/Documentacion/cap%203/prueba%20de%20penetracion/Gonzalez.pdf.

Manual de jQuery. [en línea], 2017. [Consulta: 4 mayo 2017]. Disponible en: https://www.desarrolloweb.com/manuales/manual-jquery.html#capitulos113.

MIRANDA, J.F.E., 2011. *Políticas de seguridad informática y la vulnerabilidad de los entornos Web de la Empresa Turbotech durante el año 2010*. S.l.: Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial. Maestría en Redes y Telecomunicaciones.

MONTOYA, C.E.G., URIBE, C.A.C. y RODRÍGUEZ, L.E.S., 2013. Seguridad en la configuración del servidor web Apache. *INGE CUC*, vol. 9, no. 2, pp. 31-38.

OWASP, 2012. *Guía de Seguridad en Aplicaciones para CISOs* [en línea]. S.l.: s.n. [Consulta: 21 mayo 2017]. Disponible en: file:///E:/TESIS/Documentacion/DOCUMENTACION%20Cap%201/owasp/Owasp-ciso-guide_es.pdf.

OWASP_SUSCERTE.pdf [en línea], [sin fecha]. S.l.: s.n. [Consulta: 24 enero 2017]. Disponible en: https://www.owasp.org/images/2/2f/OWASP_SUSCERTE.pdf.

PHP: Hypertext Preprocessor. [en línea], 2017. [Consulta: 4 mayo 2017]. Disponible en: http://php.net/.

POTENCIER, F. y ZANINOTTO, F., 2012. Symfony, la guía definitiva. *Potencier. org [viitattu 05.12. 2015]. Saatavissa: http://fabien. potencier. org/article/65/why-symfony*,

PRESSMAN, R.S., 1988. *Ingeniería del software*. S.l.: s.n.

PURIFICACIÓN AGUILERA LÓPEZ, 2010. *Seguridad informática*. S.l.: Editex. ISBN 84-9771-761-9.

¿Que es Bootstrap? [en línea], 2017. [Consulta: 4 mayo 2017]. Disponible en: https://raiolanetworks.es/blog/que-es-bootstrap/.

SÁNCHEZ, J., 2011. Sistemas de autenticación y autorización en internet. *Trabajo de investigación realizado en el Master« Interacción Persona Ordenador» de la Universidad Española de Lleida-Junio*,

SÁNCHEZ, T.R., 2014. Metodología de desarrollo para la Actividad productiva de la UCI. [en línea], [Consulta: 31 mayo 2017]. Disponible en: https://correo.estudiantes.uci.cu/service/home/~/?auth=co&loc=es&id=3869&part=2.

Security (current). [en línea], [sin fecha]. [Consulta: 4 mayo 2017]. Disponible en: http://symfony.com/doc/current/security.html.

Seguridad Informática. [en línea], 2017. [Consulta: 30 mayo 2017]. Disponible en: http://antisec-security.blogspot.com/2012/09/escaner-de-vulnerabilidades-parte-i.html.

SOMMERVILLE, I., 2005. *Ingeniería del software*. S.l.: Pearson Educación. ISBN 84-7829-074-5.

STOCK, A. van der, 2005. *Una Guía para Construir Aplicaciones y Servicios Web Seguros*. S.l.: Free Software Foundation.

ÚBEDA, I.L., 2015. *GeoRBAC: Dispositivo móvil en la arquitectura de control de acceso*. S.l.: s.n.

UCI, 2017. Componente: SeguridadBundle — Boson 0.1. [en línea]. [Consulta: 17 mayo 2017]. Disponible en: http://docs.prod.uci.cu/online/Boson.docset/Contents/Resources/Documents/docs/UCI/Boson/SeguridadBundle/index.html.

Usability Metrics. [en línea], 2017. [Consulta: 21 mayo 2017]. Disponible en: http://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/.

Visual Paradigm Features. [en línea], 2017. [Consulta: 4 mayo 2017]. Disponible en: https://www.visual-paradigm.com/features/.

WICHERS, D., 2013. Owasp top-10 2013. *OWASP Foundation, February*,

YAGÜE, A. y GARBAJOSA, J., 2009. Comparativa práctica de las pruebas en entornos tradicionales y ágiles. *Redalyc,(5)*, vol. 19.