UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

TÍTULO: Módulo de administración de la estructura organizacional y las categorías de dominios en el sistema AiresProxyAudit.

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

AUTORES: Leonardo Rojas Rodríguez

Roberto Rogelio Marto Ramírez

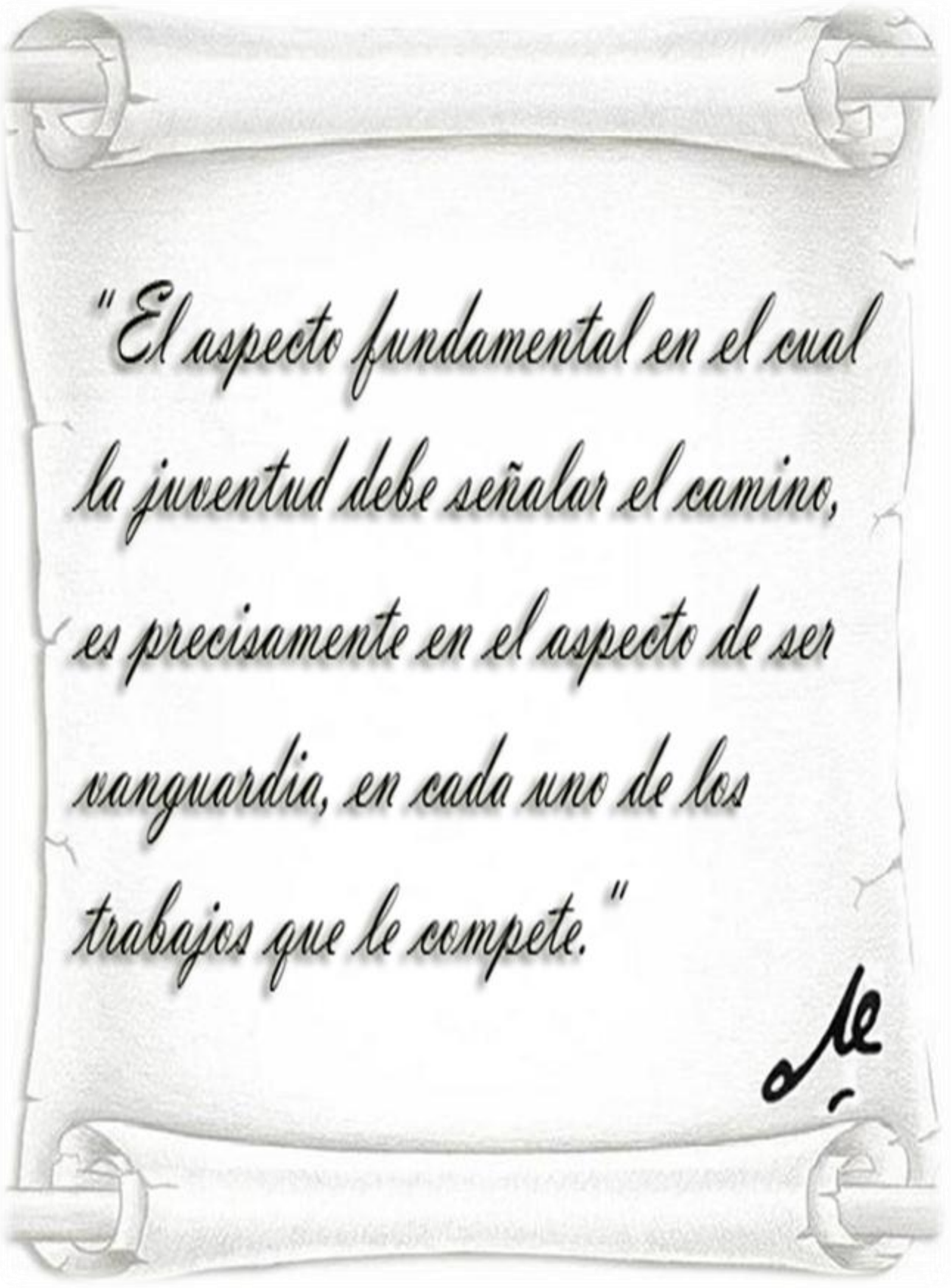
TUTORES: Ing. Mailin Diéguez Pavón

Ing. Yonny Mondelo Hernández

CO-TUTOR: Ing. Miguel Ángel Chávez Alfonso

Ciudad de La Habana, febrero 2013

"Año 54 de la Revolución"


"El aspecto fundamental en el cual
la juventud debe señalar el camino,
es precisamente en el aspecto de ser
vanguardia, en cada uno de los
trabajos que le compete."

DEDICATORIA

LEONARDO:

Dedico mi trabajo a las personas que me rodean en especial a mi familia, que es y será la piedra angular de mi futuro y las manos que me sostienen e impulsan por esta vida.

A mi madre que nunca dudó de mi aún cuando fallé, a mi padre que nunca me dejó de apoyar y a mi hermano que a pesar de ser menor que yo siempre se comportó como mi padre.

A mi abuelo quien siempre me inspiró a estudiar y superarme, a mi abuela que es la almohada donde reposan mis pesares y alegrías. A mis tíos que siempre celebraron con orgullo tener un sobrino ingeniero que estudiase en la Habana.

Dedico mi esfuerzo y consagración a mi tutora Mailin quien no me dejó desde un inicio divagar o apartarme de mi objetivo y aún cuando los resultados no fueron los mejores me siguió alentando a no desfallecer o amilanarnos frente a las adversidades.

ROBERTO:

A mis abuelos, por la sabiduría,

a mi abuela por el amor,

a mi abuelo por el ejemplo.

A mis padres, por la luz,

a mi madre, por el camino.

a mi padre, por la fuerza.

A mi tía, por el sacrificio.

A mis hermanas por el futuro.

A mi prima por la compañía.

A mis amigos por la confianza.

A mi novia por lo entregado.

A Dios.

AGRADECIMIENTOS

LEONARDO:

Mis agradecimientos a las personas que de una forma u otra contribuyeron a que mi sueño se hiciera realidad.

A todas mis compañeros en el laboratorio 18 en especial a Rubén y Chávez que siempre estuvieron presentes para cualquier problema o necesidad.

A todos aquellos que son mis amistades y fueron mi familia en la universidad, a Chachi, Elsa, Herminio, Mairelis, Osmel, Mercedes, Marbelis, Juan, José Antonio, Marianela, José, Daniela y su familia. No los olvidaré aunque me marche, que aquí tengo un hogar y una familia con quien contar, aunque me niegue a convertirme en palestino.

A Eduardo y su familia que son como mis parientes, por su alegría y compañía en los fines de semana que iba a visitarlos, y por dejarme ser uno más.

ROBERTO:

En especial a mi familia, que lo ha dado todo por mí, por protegerme solo lo necesario y dejar que cometa mis propios errores.

A mi abuela, a mi abuelo, a mi mamá, a mi tía, a mi papá, a mi prima Heidy y a mis hermanitas Kamila y Vanessa, por ser lo inimaginable y recordarme cada día que la solución está en el querer, por ser la antorcha de mi pensamiento, mi fuerza y el manantial de mi amor.

A mi novia Ailsa por estar todo este año conmigo en las buenas y malas, a ella le debo el ponerle ganas a todo, por ser tan perfeccionista y difícil, por hacerme ver luz incluso donde todo estaba encendido y por exigirme tanto cuando todo estaba hecho, sin ella la meta no hubiera sido alcanzada.

A Murgo el Infeliz, al Tomeguín de Camagüey, al Ted el Peluchito, a Tomás el Emo, al Ruso, a Tulino, al Flaco: a estos no tengo nada que agradecer porque han hecho todo mal, son la causa de mis malas noches, de mis malos ratos, de mis bajas, de mi pérdida de tiempo, son la causa del desorden que siempre había en mi cuarto y que yo de buen amigo siempre les recogía, al final termino igual que ellos, el futuro estaba escrito, pero así y todo son lo mejor que me llevo de esta universidad y los voy a extrañar cantidad, no se imaginan cuánto.

A mi compañero de tesis y a mis tutores por su paciencia.

A los amigos que no mencioné y a los que ya no son parte de mi vida.

A todos los que de una forma u otra han contribuido a la realización de este sueño.

DECLARACIÓN DE AUTORÍA

Se declara que somos los únicos autores de este trabajo y autorizamos a la Facultad 1 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2013.

_____ _____

Roberto Rogelio Marto Ramírez Leonardo Rojas Rodríguez

_____

Mailin Diéguez Pavón

DATOS DE CONTACTO

Tutor: Ing. Mailin Diéguez Pavón. Ingeniero en Ciencias Informáticas, UCI 2010. Especialista del centro de producción CIDI, Facultad 1.

Correo electrónico: mdieguez@uci.cu . Teléfono: 837 2353

Tutor: Ing. Yonny Mondelo Hernández. Ingeniero en Ciencias Informáticas, UCI 2010. Profesor Adjunto del centro de producción CIDI, Facultad 1.

Correo electrónico: ymondelo@uci.cu . Teléfono: 837-2237

Co-tutor: Ing. Miguel Ángel Chávez Alfonso. Ingeniero en Ciencias Informáticas, UCI 2010. Especialista del centro de producción CIDI, Facultad 1.

Correo electrónico: machavez@uci.cu . Teléfono: 837 3092

OPINIONES Y AVALES.

Peña Tecnológica
... donde convergen personas y tecnologías

CERTIFICADO

A: MÓDULO DE ADMINISTRACIÓN DE LA ESTRUCTURA ORGANIZACIONAL Y LAS CATEGORÍAS DE DOMINIOS EN EL SISTEMA AIRESPROXYAUDIT.

Por: Haber obtenido la categoría DESTACADO

Entregado en la Universidad de la Ciencias Informáticas, La Habana, a los 10 días del mes de abril del año 2013.

Ing. Héctor Ochil Pérez
Presidente del Comité Organizador del Evento
Presidente del Consejo Científico DTS-MININT

RESUMEN

El Centro de Ideoinformática de la Facultad 1 de la Universidad de las Ciencias Informáticas (UCI), posee dentro de sus líneas directrices el desarrollo de diferentes herramientas vinculadas a la Analítica Web, con el objetivo de apoyar la toma de decisiones de los directivos en las empresas en aras de optimizar el uso de Internet y el control del cumplimiento de las regulaciones establecidas en la resolución 127/2007 por el Ministerio de la Información y las Comunicaciones (MIC). En conjunto con la Dirección de Redes y Seguridad Informática de la UCI, se desarrolló el sistema Analizador Inteligente de Registros Proxy para Auditores (AiresProxyAudit). Este sistema cuenta con el módulo de administración, el cual permite la agrupación de usuarios en entidades o departamentos lógicos, elemento necesario para realizar auditorías dirigidas hacia usuarios que pertenecen a una misma área. Además tiene diferentes mecanismos para la creación de dicha estructura y la importación/exportación de los diferentes usuarios o grupos de usuarios a partir de los datos a los que se tiene acceso, a través de los servicios de información.

Otro elemento importante de este módulo es que permite asignar permisos de auditores a los usuarios que cuenten con este privilegio, lo cual permitirá generar reportes sobre los grupos de usuarios de navegación de las áreas que puede auditar. El hecho que AiresProxyAudit cuente con el módulo de administración lo convierte en una herramienta que facilita el control y cumplimiento de las Políticas de Seguridad Informática en cualquier institución.

PALABRAS CLAVES: administración de usuarios, gestión de estructura organizacional, permisos de auditores, proxy.

ABSTRACT

The Ideoinformática Center of the Faculty # 1 of the University of Informatics Sciences, has within its guidelines the development of different tools related to Web Analytics with the aim of supporting decisions making of managers in enterprises in order of optimizing the use of Internet and the monitoring of compliance with the regulations established by resolution 127/2007 by the Ministry of Informatics and Communications (MIC). In conjunction with the Department of Network and Information Security of the UCI, was developed the system Intelligent Analyzer of Proxy Records for Auditors (AiresProxyAudit). This system has the management module, which allows the grouping of users or department's logical entities, which is necessary for audits aimed at users who belong to the same area. It also has different mechanisms for the creation of this structure and importing / exporting of different users or groups of users based on the data that is accessed through information services.

Another important element of this module is that it allows assigning auditor's permissions to users who have this privilege, which will generate reports on group's user navigation areas that can be audited. The fact that AiresProxyAudit has the management module makes it a tool that facilitates the control and enforcement of informatics security policies in any institution.

KEYWORDS: user's management, organizational structure management, auditor's permissions, proxy.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción.

Con el avance de las Tecnologías de la Información y las Comunicaciones (TIC), en el ámbito nacional existen centros estatales y educacionales del nivel superior que cuentan con el servicio de acceso a Internet, entre ellos la Universidad de las Ciencias Informáticas (UCI). En el país se cuenta con 2361 nombres de dominio bajo .cu registrados por el Centro Cubano de Información de Red (CUBA-NIC) hasta abril del 2013, de los cuales 31 pertenecen a educación y 13 a salud, catalogados como dominios genéricos de segundo nivel(1). Mediante este servicio, el usuario dispone de una vía para acceder a la información ofrecida por multitud de servidores que se encuentran dispersos por todo el mundo y distribuida en una gran variedad de contenidos, que van desde simple texto hasta contenidos multimedia, dígase gráficos, sonidos, animaciones o videos; como una alternativa en el proceso de autoaprendizaje, principalmente en lo que respecta a la búsqueda o intercambio de información para su uso en actividades de investigación y desarrollo.

Los logros alcanzados en los últimos años en la informatización de la sociedad cubana, con el incremento de las tecnologías de la información en todos los sectores y en particular de las redes informáticas y sus servicios asociados, requieren de la adopción de medidas que garanticen un adecuado nivel de seguridad para su protección y ordenamiento. La seguridad de las organizaciones, sistemas y redes de información están constantemente amenazadas por diversas fuentes que incluyen ataques de distintos tipos y orígenes, factores que aumentan los riesgos a los que están expuestos los servicios y protocolos utilizados, así como el contenido de la información tratada en dichos sistemas. Todo ello puede afectar severamente la confidencialidad, integridad y disponibilidad de la información que se encuentra en los servidores de la institución. La resolución número 127/2007 del Ministerio de la Informática y las Comunicaciones regula las normas y procedimientos en materia de seguridad informática para las instituciones en todo el territorio nacional(2).

Cada vez son más las personas que necesitan o usan el acceso a la Web para realizar su quehacer diario y son más las instituciones que requieren de servidores proxy para el control del acceso y la navegación por Internet. El funcionamiento de los servidores proxy consiste en interceptar las conexiones de red que un cliente hace a un servidor de destino por varios motivos posibles, como seguridad, rendimiento, anonimato y control de la navegación por la red. Esta función de servidor proxy puede ser realizada por un programa o dispositivo, el cual se encuentra ejecutándose en un

servidor o un host de acceso a la red, el aumento del uso de esta tecnología se debe a su flexibilidad y al hecho de que se integra con un número creciente de sistemas (3).

Entre las principales ventajas de emplear un servidor proxy se encuentra la posibilidad de analizar los registros de navegación generados por los usuarios que se conectan a la red a través del sistema. Los registros de navegación almacenan los eventos que se producen en el servidor como consecuencia de la navegación de los usuarios y contienen información como la dirección IP[1] del usuario, la fecha y la hora de la petición, los objetos solicitados y el resultado de la operación solicitada; información que puede ser almacenada en un servidor para su posterior análisis. Como vía de solución al engorroso trabajo de analizar los registros proxy se utilizan algunas herramientas que realizan este proceso, pues interpretar estos registros de forma manual es casi imposible, debido a la cantidad de información y la difícil comprensión de los registros almacenados. Destacando que las aplicaciones solo están accesibles a los administradores de red y en muchos casos no presentan una interfaz amigable, lo que impide a los usuarios comunes conocer sus trazas.

La Dirección de Redes y Seguridad Informática de la UCI no cuenta con una herramienta que permita la administración de usuarios o grupos de usuarios de navegación en una estructura jerárquica, que pueda brindar una visión única respecto a las actividades que realizan los miembros de la institución en la Web; una herramienta que posibilite la detección temprana, de hechos extraordinarios durante la navegación de los usuarios y así mitigar las vulnerabilidades y debilidades propias, que puedan ser utilizadas para atentar contra la universidad.

El Centro de Ideoinformática de la facultad 1 de la UCI, posee dentro de sus líneas directrices la intención de desarrollar diferentes herramientas vinculadas a la analítica web, con el objetivo de apoyar la toma de decisiones de los directivos en las empresas en aras de optimizar el uso de Internet y el control del cumplimiento de las regulaciones establecidas. En conjunto con la Dirección de Redes y Seguridad Informática de la UCI, se desarrolló el sistema Analizador Inteligente de Registros Proxy (AiresProxy) como una herramienta de análisis de la navegación que realizan los usuarios en Internet, con la capacidad para generar reportes sobre la navegación de los usuarios, el cual permite realizar reportes estadísticos de la navegación a través de los registros que se almacenan en el servidor proxy de la institución.

[1] Siglas del Inglés *Internet Protocol (Protocolo de Internet)*

El sistema AiresProxy no realiza el procesamiento de información sobre grupos de usuarios ni la asignación de permisos de auditores, razón por la cual ha sido implementado el nuevo sistema Analizador Inteligente de Registros Proxy para Auditores (AiresProxyAudit). Este último actualmente no permite la agrupación de usuarios en entidades o departamentos lógicos, elemento necesario para realizar auditorías dirigidas hacia grupos de usuarios. Las áreas y usuarios actualmente se pueden identificar a partir de los servidores LDAP[2], los cuales son servidores de datos optimizados para dar una respuesta rápida a consultas de lectura y están orientados al almacenamiento de datos de los usuarios a modo de directorio, o a partir de grupos definidos por los jefes de cada área, pero debido a que el proceso no se ha automatizado, es necesario tener diferentes mecanismos para la creación de dicha estructura y la importación/exportación de los diferentes usuarios o grupos de usuarios a partir de los datos a los que se tiene acceso, a través de los servicios de información.

En estos momentos el sistema no permite la gestión de grupos de usuarios y de auditores de navegación, además existe un número limitado de clasificaciones de dominios. Para la adición de nuevas clasificaciones resulta necesario realizar modificaciones en el código fuente, lo que limita la extensibilidad del sistema. Además el cálculo de la cuota establecida para cada usuario se realiza de forma estática impidiendo la adición de nuevas reglas asociadas a las clasificaciones de los diferentes dominios y los horarios.

Teniendo en cuenta la información preliminar se identifica para la presente investigación el siguiente **problema a resolver**: ¿Cómo contribuir al control de las actividades en la Web de los grupos de usuarios de navegación por Internet? Definiéndose como **objeto de estudio** el proceso de gestión de usuarios. Enmarcándose en el **campo de acción** la gestión de usuarios y grupos de usuarios en sistemas informáticos. Se plantea como **objetivo general** de la investigación desarrollar un módulo que permita la gestión de los grupos de usuarios y las categorías de dominios en el sistema AiresProxyAudit. Para darle cumplimiento al objetivo trazado se determina que los **objetivos específicos** a realizar durante la investigación sean los siguientes:

[2] Siglas del Inglés *Lightweight Directory Access Protocol (Protocolo Ligero de Acceso a Directorios)*

1. Elaborar el marco teórico conceptual y el estado del arte respecto a las tecnologías actuales para el módulo de Administración del sistema AiresProxyAudit.
2. Diseñar el sistema de administración de la estructura organizacional y las categorías de dominios para AiresProxyAudit.
3. Implementar las funcionalidades que permitan gestionar los grupos usuarios de la UCI, la asignación de roles y permisos de auditores y la categorización de los dominios.
4. Validar el correcto funcionamiento del módulo de Administración.

El cumplimiento exitoso de los objetivos específicos expuestos anteriormente contribuye a la contrastación y/o verificación de la **idea a defender** de esta investigación: desarrollar un módulo de administración para el sistema AiresProxyAudit, permitirá gestionar los grupos de usuarios de navegación y las categorías de dominios, para lograr la realización de auditorías dirigidas hacia grupos de usuarios.

Como **posible resultado** se obtendrá un módulo para la gestión de los usuarios, grupos de usuarios de navegación y categorías de dominios en el sistema AiresProxyAudit, mediante el cual se diseñe la estructura lógica de la empresa de forma manual o mediante la importación de los usuarios o grupos de usuarios a través de diferentes mecanismos, y se administren los permisos de los auditores sobre las entidades o áreas de la estructura, además de los usuarios de cada grupo y se gestione la categorización de los dominios.

Para el desarrollo de la investigación se emplean **métodos científicos teóricos**. Dentro de los cuales se utilizan el Método Histórico – Lógico y el de Análisis y Síntesis. A continuación se explica el por qué de la selección de los mismos.

1. Histórico – Lógico: Se aplica para realizar el seguimiento de la evolución del objeto de estudio y una predicción de lo que puede hacerse en el futuro, en este caso se realiza una investigación de las funcionalidades de AiresProxy que puedan ser aplicadas a procesos de administración de la estructura organizacional.
2. Análisis y Síntesis: Se utiliza para identificar y analizar las diversas funcionalidades y procesos que intervienen en AiresProxy, que pueden ser aplicados a la administración de la estructura organizacional y su posterior síntesis

a partir de la selección de las funcionalidades, conforme a las necesidades de la Universidad de las Ciencias Informáticas.

El presente trabajo está estructurado en 3 capítulos, como se describen a continuación:

El capítulo 1 "Fundamentación Teórica".

En este capítulo se exponen los principales conceptos que contribuyen al mejor entendimiento del problema en cuestión. Se especifican detalladamente todos los argumentos que esclarecen el objeto de estudio, a partir del análisis del estado del arte de los sistemas homólogos, enfocados a la administración de usuarios y grupos de usuarios en sistemas informáticos.Se efectúa un análisis de las metodologías de desarrollo, los lenguajes de modelado y herramientas, seleccionando las que serán utilizadas en el desarrollo del presente trabajo.

El capítulo 2 "Características del sistema".

En este capítulo se exponen las características del sistema, contiene una descripción detallada del flujo de los procesos que dan surgimiento al problema. Se muestra la propuesta de solución a partir del modelo de dominio que facilita la comprensión de su funcionamiento. Se describen los requisitos funcionales y no funcionales que debe tener el módulo a desarrollar para satisfacer las necesidades del cliente. También se define el diagrama de Casos de Uso del sistema, así como la descripción de los mismos.

En el análisis y diseño del sistema, se describe la arquitectura del módulo mostrando los patrones de diseño a emplear en la implementación del mismo. Además, se identifican algunas de las ventajas de usar el *framework* Symfony[3] en el desarrollo de aplicaciones web. Se describe el modelo de diseño y el diagrama de despliegue.

El capítulo 3: "Implementación y pruebas".

En este capítulo se muestran los diagramas de componentes y la descripción de los mismos, para así lograr comprender los elementos de *software* del módulo y las relaciones entre ellos. También se muestran las pantallas principales del módulo desarrollado, así como las pruebas realizadas en la validación del sistema implementado.

[3] www.symfony.com

Capitulo #1 – Fundamentación Teórica.

1.1 Introducción.

En el presente capítulo se establecen los fundamentos teóricos de la investigación y los conceptos asociados al dominio del problema a resolver. El estudio del estado del arte se realiza un análisis sobre los sistemas homólogos en el ámbito nacional e internacional para la posible inclusión de funcionalidades en la solución a desarrollar, además del análisis de los procesos de administración de usuarios y grupos de usuarios, para su agrupación en áreas lógicas y la posterior auditoría de la navegación a partir de uno o más usuarios de un área en la estructura.

1.2 Conceptos asociados al dominio del problema.

Con el propósito de tener una mejor comprensión de los temas que serán abordados en el capítulo, directamente relacionados con el objeto de estudio de la investigación, se describen a continuación un grupo de conceptos entre los que se destacan: Registros de Navegación, *Software* Libre, Servidor Proxy, *Framework* de Desarrollo, Estructura Organizacional, Administración.

1.2.1 Registros de Navegación.

Un registro de navegación es un fichero generado por el servidor donde se almacenan los eventos ocurridos durante la actividad en la Web en un rango de tiempo determinado. Es usado por los profesionales de seguridad informática para registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación.

La mayoría de los registros son almacenados o desplegados en el formato estándar, el cual es un conjunto de caracteres para dispositivos comunes y aplicaciones. De esta forma cada log generado por un dispositivo en particular puede ser leído y desplegado en otro diferente.

También se le considera como aquel mensaje que genera el programador de un sistema operativo, alguna aplicación o algún proceso, en virtud del cual se muestra un evento del sistema.

Un registro de servidor es un archivo, o varios, creado y almacenado por el servidor de forma automática, de la actividad realizada por el mismo; tal servidor mantiene un

historial de las peticiones de página por los usuarios, formado con la información que se agrega normalmente con la solicitud, incluyendo dirección IP del cliente, fecha de la solicitud/tiempo, página que solicita, código HTTP[4], bytes comunicados, agente de usuario (protocolos), y remitente (4).

1.2.2 Software Libre.

El *software* libre es la denominación del *software* que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado, y redistribuido libremente. Al tratar el tema de software libre en materia de desarrollo de software se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el *software* y distribuirlo modificado (5).

1.2.3 Servidor Proxy.

El servidor proxy es una aplicación que soporta peticiones HTTP, HTTPS[5] y FTP[6], entre otras, a equipos que necesitan acceder a Internet y a su vez, provee la funcionalidad de caché especializado que almacena de forma local las páginas consultadas recientemente por los usuarios. De esta manera, incrementa la rapidez de acceso a los servidores de información web. También puede operar de forma transparente ya que las conexiones son direccionadas dentro del mismo servidor proxy sin configuración adicional por parte del cliente para su navegación en Internet, visto de otra forma, el navegador web no necesita ser configurado para que aproveche las características del servidor proxy (6).

Su finalidad más habitual consiste en interceptar las conexiones de red que un cliente hace a un servidor de destino, por varios motivos posibles como seguridad, rendimiento, anonimato y proteger una subred interna de los posibles ataques desde una red exterior. Cuando un explorador solicita una página web almacenada en la colección (su caché) del servidor proxy, el servidor proxy la proporciona, lo que resulta más rápido que consultar la Web. Los servidores proxy también ayudan a mejorar la seguridad, ya que filtran algunos contenidos web y *software* malintencionado (7).

[4] Por sus siglas en inglés *Hyper Text Transfer Protocol (Protocolo de Transferencia de Hiper Texto)*
[5] Por sus siglas en inglés *HyperText Transfer Protocol Secure (Protocolo Seguro de Transferencia de Hiper Texto)*
[6] Por sus siglas en inglés *File Transfer Protocol (Protocolo de Transferencia de Ficheros)*

1.2.4 Framework de Desarrollo.

Un *framework* de desarrollo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. En el desarrollo de *software*, un *framework* o infraestructura digital, es un marco de trabajo, una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, con base a la cual otro proyecto de *software* puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de *software* que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio (8).

1.2.5 Estructura Organizacional.

Una estructura es un patrón establecido de relaciones entre los diferentes componentes o partes que conforman la organización y en la misma se representa el funcionamiento interno y se infiere de las operaciones reales y el comportamiento de la organización; también es una forma de dividir, organizar, coordinar las actividades e indicar la organización jerárquica y funcional de la empresa.

La estructura formal expresa los procesos de acción mutua entre sus miembros, define las especialidades de trabajo y las líneas de comunicación. Los procesos reales sin embargo, no siempre siguen estas líneas de interacción, sino que se entremezclan con procesos informales. Así, la estructura formal (prevista) se convierte en la estructura real con la intervención de los miembros de la organización (9).

1.2.6 Administración.

La Administración es el proceso cuyo objeto es la coordinación eficaz y eficiente de los recursos de un grupo social para lograr sus objetivos con la máxima productividad.

La administración se define como "el proceso de estructurar y utilizar conjuntos de recursos orientados hacia el logro de metas, para llevar a cabo las tareas en un entorno organizacional" (10).

1.2.7 Sistemas Analizadores de Registros Proxy.

Los sistemas analizadores de registros proxy son programas informáticos utilizados principalmente por empresas u organizaciones que han modernizado su accionar ubicándose en Internet para el análisis de los registros de navegación de los usuarios de la institución. Estos sistemas brindan una serie de reportes con el objetivo de visualizar información importante y necesaria en la toma de decisiones claves, que influyen en el correcto desempeño de las empresas.

Un sistema de análisis de registros proxy es una aplicación avanzada para el proceso y análisis de registros, informes y registros de eventos. El programa está orientado especialmente a los registros de navegación almacenados por servidores proxy, permitiéndole al administrador del sistema obtener información fidedigna respecto a la navegación y el funcionamiento de la navegación de los usuarios en la institución a través de Internet. La interfaz de administración es de vital importancia para estos sistemas ya que permite al usuario administrador la gestión de los reportes de navegación y de los diferentes procesos del sistema.

1.3 Análisis de soluciones existentes en el mundo.

El desarrollo de este epígrafe se centra principalmente en el análisis de soluciones existentes a nivel mundial que son utilizadas para procesos de análisis de los registros de navegación. Tres de estas aplicaciones fueron seleccionadas debido a que han sido planteadas de tal manera que se pueden ajustar para darle solución al problema a resolver planteado en el presente trabajo. Haciendo énfasis en el estudio de la administración y la gestión de usuarios, se exponen a continuación algunas propuestas de estas aplicaciones en calidad de soluciones existentes.

1.3.1 Sawmill.

Sawmill es una aplicación de análisis inteligente de registros de navegación, orientada especialmente al análisis de los registros de servidores web, pero se puede utilizar para trabajar con cualquier tipo de registros.

Características.

Esta aplicación está especialmente diseñada para analizar los registros de navegación y de acceso a servidores web, pero puede procesar cualquier registro web. Se ejecuta como un programa CGI[7] (*Common Gateway Interface*) en un servidor web, y publica

[7] Por sus siglas en inglés *Common Gateway Interface* (Interfaz de Entrada Común)

una intuitiva interfaz gráfica de usuario, que puede utilizarse desde cualquier navegador para configurar y ejecutar Sawmill o para ver estadísticas de páginas. Las estadísticas son jerárquicas, atractivas y poseen enlaces que facilitan la navegación. El programa incluye una completa documentación a pesar de encontrarse en idioma inglés (11).

Utilidades.

Sawmill ofrece un gran número de opciones, incluida una base de datos persistente, el control sobre la apariencia de las páginas de estadísticas y diversas opciones de filtrado sobre los registros. Al concluir su instalación muestra una interfaz en el navegador web que presenta, en un cuadro de selección de opciones ubicado a la izquierda, una serie de estadísticas posibles:

- Cantidad de visitas por hora, por día, por mes.
- Horas pico y horas de baja audiencia.
- Páginas más visitadas.
- Páginas de entrada y salida más frecuentes del sitio.
- Utilización de buscadores, clasificación de palabras clave empleadas para buscar.

Gestión de Usuarios

La aplicación posee un editor de administración de usuario para la administración y notificación sobre el acceso de los usuarios a la web pero no posee la gestión de grupos de usuarios de navegación. Asigna permisos de navegación a usuarios para que estos cumplan con las directrices de seguridad de la institución y permite la aplicación de diferentes reglas de acceso para los usuarios que navegan a través del sistema, facilitando la auditoría de las actividades en la Web dirigida a usuarios.

Interfaz gráfica principal del software.

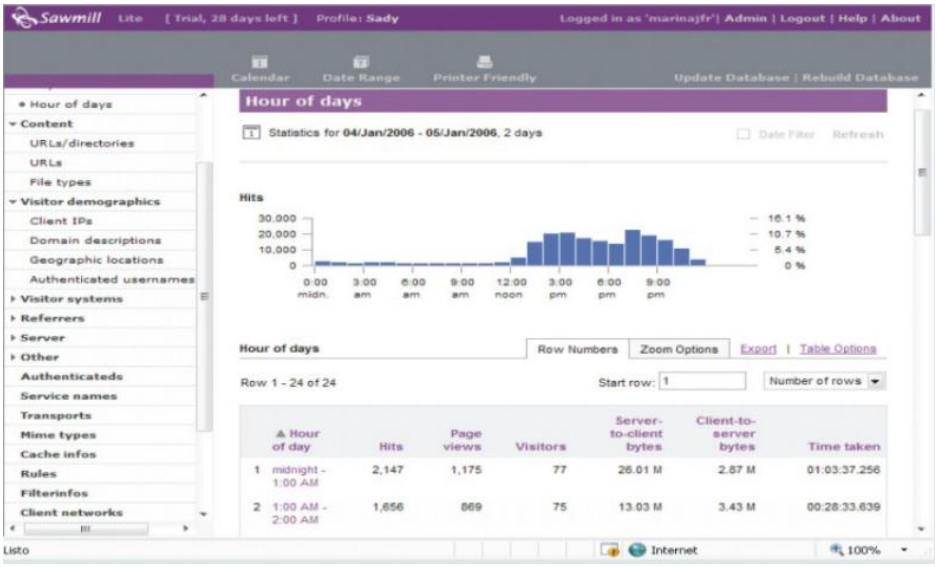


Figura # 1 Interfaz principal de la aplicación Sawmill.

1.3.2 OpenLDAP.

OpenLDAP es una implementación libre y de código abierto del protocolo LDAP (*Lightweight Directory Access Protocol*) desarrollada por el proyecto OpenLDAP. Está liberada bajo su propia licencia OpenLDAP *Public License*. LDAP es un protocolo de comunicación independiente de la plataforma. Muchas distribuciones GNU/Linux incluyen el software OpenLDAP para el soporte LDAP. Este software corre en plataformas BSD, AIX, HP-UX, Mac OS X, Solaris, Microsoft Windows (NT y derivados, incluyendo 2000, XP, Vista), y z/OS (12).

Un directorio LDAP es una base de datos optimizada para la lectura y búsqueda de información que almacena la información de manera jerárquica. Los directorios soportan opciones avanzadas de filtrado. Generalmente no soportan transacciones complejas que sí ofrecen los sistemas de gestión de bases de datos diseñadas para procesar un gran volumen de actualizaciones. Los cambios en la información almacenada en un directorio suelen ser del tipo “o todo o nada", es decir, cambios de las ramas del árbol DIT (*Directory Information Tree*) completamente, pero, aunque no estén optimizados para ello, los directorios pueden permitir cambios muy específicos. Los directorios están preparados para dar una respuesta rápida a un gran volumen de búsquedas. Disponen de mecanismos de replicación de la información en varios servidores para incrementarla disponibilidad y fiabilidad del servicio mientras se reduce el tiempo de respuesta (13).

Básicamente, Open LDAP posee tres componentes principales:

- slapd - Dominio de servidor y herramientas.
- Bibliotecas que implementan el protocolo LDAP.
- Programas cliente: ldapsearch, ldapadd, ldapdelete, entre otros.

Adicionalmente, el proyecto Open LDAP es anfitrión de los subproyectos:

- JLDAP (biblioteca de clases LDAP para Java).
- JDBC-LDAP (controlador Java JDBC – LDAP).
- ldapc++ (biblioteca de clases LDAP para C++).

Gestión de Usuarios

El directorio dispone de tres *backend* independientes para almacenar información de los usuarios de la red, los parámetros de calidad para cada una de las clases de servicio y las políticas respectivamente.

1. Información de usuarios: Se almacenan los nombres de usuario, contraseñas, grupo al que pertenecen y que SLS tienen asignado.
2. Información de los SLS: Contiene los parámetros de rendimiento de tráfico y de función de policía para cada una de las clases de servicio.
3. Políticas: Los documentos PCLS (14) y PCELS (15) definen el modo de almacenar estas políticas en el directorio. Actualmente hay almacenadas políticas de control de admisión y políticas de gestión.

Estructura del directorio

Los servidores LDAP almacenan la información en una estructura de datos jerárquica (Ver figura 2). Esta estructura puede ser distribuida, o sea, puede existir un servidor LDAP que contenga un subárbol de otro. El servidor, al recibir una consulta de un cliente, le responde o le indica dónde puede encontrar la respuesta. Los objetos se almacenan en entradas y cada entrada posee un conjunto de atributos y un Nombre Distinguido (DN, por sus siglas en inglés) que la identifica unívocamente.

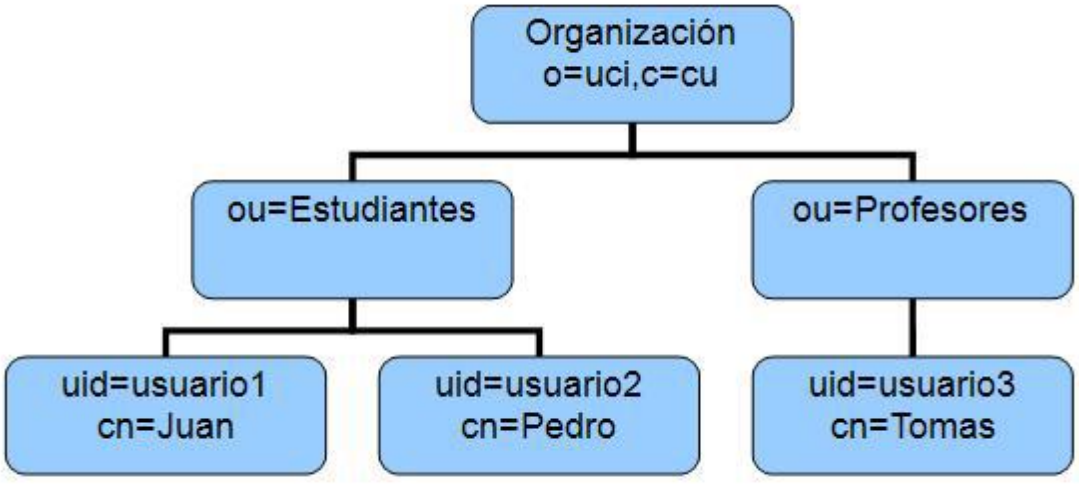


Figura # 2 Estructura de Directorio en un servidor LDAP.

Los datos del directorio se representan mediante atributos y sus valores. Algunos de los tipos de atributos más comunes son:

• **uid**: Usuario (en el caso del Directorio Activo se emplea el atributo **samaccountname**)

• **ou:** Unidad Organizacional

• **o:** Organización

• **c:** País

• **cn**: Nombre común

• **givenname**: Nombre de pila

• **sn**: Apellidos

• **mail**: Dirección de correo electrónico.

Los datos de un directorio LDAP pueden ser exportados a un fichero en formato LDIF[8]. LDIF es un formato utilizado para la realización de operaciones por lotes entre directorios según los estándares del protocolo LDAP. LDIF puede utilizarse para exportar e importar datos del directorio y permite llevar a cabo operaciones como agregar, crear o modificar (16).

1.3.3 Webmin.

Webmin es una interfaz basada en web para la administración de sistemas Unix, consiste en un servidor simple y un número de programas CGI que directamente ponen

[8] Por sus siglas en inglés *LDAP Interchange Format (Formato de Intercambio LDAP)*

al día archivos de sistema. Con esta herramienta se pueden configurar los permisos para usuarios y grupos o configurar el funcionamiento de los servidores que se encuentran funcionando en el sistema, cuotas de espacio del disco, servicios, archivos de configuración y apagado del equipo entre otros.

Webmin tiene una estructura por módulos para administrar una amplia variedad de herramientas como Apache, PHP, MySQL, PostgreSQL, DNS, Samba y DHCP entre otras. Además de permitirle al usuario la completa configuración y personalización de los servicios, le brinda la opción de administrar el sistema completamente desde un host remoto a través de la interfaz web de la aplicación (17).

Gestión de Usuarios.

El usuario por defecto es el root, una vez que se entra en Webmin se puede crear uno o varios usuarios de administración para Webmin. En el caso de que se olvide la contraseña de acceso y se tenga acceso como root al ordenador, se puede crear una nueva utilizando el comando:

```
/usr/share/webmin/.changepass.pl/etc/webmin usuario contraseña
```

En el módulo usuarios de Webmin se encuentran las diferentes opciones disponibles para definir y configurar los usuarios que tendrán acceso a Webmin. Permite al administrador del sistema crear diferentes usuarios para determinadas tareas. Por ejemplo, si el ordenador se utiliza como servidor de correo, se puede crear un usuario que tan solo tenga acceso al módulo de administración de Exim, o si se emplea como servidor de impresión se puede crear un usuario que administre las colas de impresión. De esta forma es posible crear diferentes usuarios en función de los módulos a los que tendrán acceso, delegando fácilmente la administración de determinados servicios del ordenador a diferentes usuarios y siendo posible incluso determinar qué aspectos de un determinado servicio podrá administrar.

Interfaz gráfica principal del software.

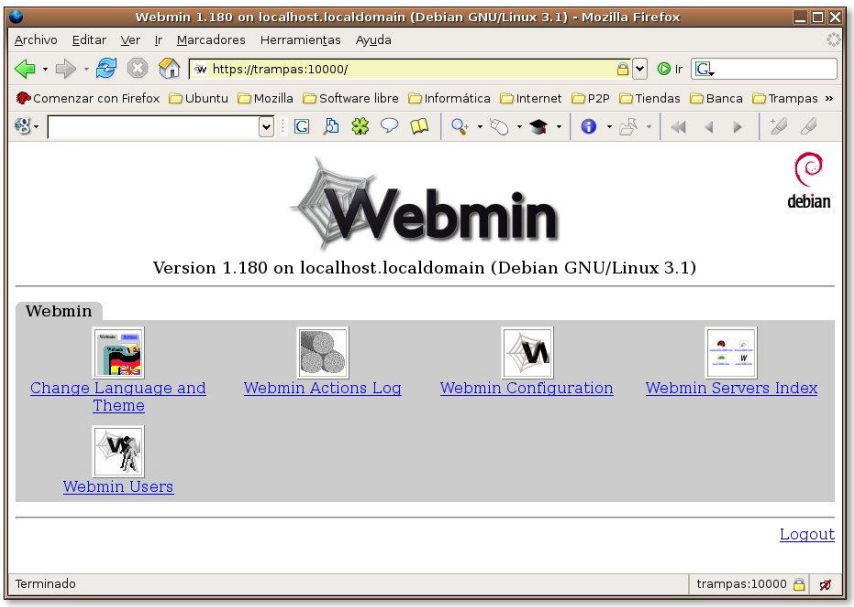


Figura # 3 Interfaz gráfica principal de Webmin.

1.3.4 Observaciones del estudio de las soluciones existentes.

El estudio de las soluciones existentes en el mundo que permiten la administración de usuarios y grupos de usuarios, ha posibilitado la opción de identificar características que deriven en funcionalidades a incluir en la solución; a partir del estudio de la herramienta OpenLDAP se decidió gestionar la información de los usuarios y grupos de usuarios en forma de árbol jerárquico. Observando el funcionamiento de los módulos de administración de la aplicación Webmin se decidió crear diferentes usuarios para la gestión de la estructura y la información de los usuarios, así como sus permisos para la interacción con el sistema. Como no se ha encontrado una herramienta que pueda ser utilizada íntegramente en la solución del problema planteado, se evidencia la necesidad de crear una herramienta o método propio.

1.4 Análisis de soluciones existentes en Cuba.

El desarrollo de este epígrafe se centra principalmente en el análisis de soluciones existentes en el ámbito nacional que son utilizadas para realizar el análisis de los registros. Haciendo énfasis en el estudio de la administración y la gestión de usuarios, se exponen a continuación las siguientes aplicaciones en calidad de soluciones existentes.

1.4.1 AiresProxy.

AiresProxy permite a directivos y empresarios contar con una herramienta auxiliar para la toma de decisiones administrativas sobre el análisis de los registros proxy de la

empresa, además los usuarios comunes pueden en todo momento tener conocimiento sobre su acceso a Internet, lo que les posibilita detectar alguna irregularidad o violación que se genere en el uso de sus cuentas.

Características.

El sistema AiresProxy cuenta con tres módulos:

1. El motor de segmentación de registros encargado de filtrar todos los registros que provienen del servidor proxy, realizando su salva en la base de datos y actualizando la cuenta de los usuarios en caso de que la empresa o institución cuente con una navegación por cuotas.
2. El motor de mantenimiento a la base de datos que se responsabiliza de eliminar los registros que lleven cierto tiempo almacenado, este tiempo es definido por el administrador de red o el encargado del sistema mediante un fichero de configuración.
3. El sistema de análisis y presentación de datos mediante una interfaz web que muestra a los usuarios los reportes de su navegación a través del proxy.

Utilidades.

Los datos generados por el servidor proxy son procesados a través de un DEMONIO que se ejecuta cada un tiempo determinado. Se recomienda la ejecución del DEMONIO cada 1 minuto con el objetivo de tener los datos actualizados con solo un minuto de desfasaje, sin embargo se puede dar la posibilidad de que los reportes sean brindados en tiempo real.

Permite la generación de reportes dinámicos sobre los registros de navegación de los usuarios ofreciendo información detallada del dominio, el IP, la fecha y hora de las peticiones además de la cantidad de datos trasmitida entre el cliente y el servidor.

Gestión de Usuarios y Grupos.

La aplicación posee un editor de administración de usuario para la administración y notificación sobre el acceso de los usuarios a la web pero no realiza la gestión de grupos de usuarios de navegación, los dominios o sus categorías.

Interfaz gráfica principal del Software.

Figura # 4 Interfaz gráfica principal de AiresProxy.

1.4.2 SmartKeeper.

SmartKeeper es un filtro de contenido web que permite aplicar las políticas de uso aceptable de Internet (AUP por sus siglas en inglés) en una institución. Está diseñado para interactuar como una lista de control de acceso (ACL por sus siglas en inglés) del servidor Squid proxy; cuenta fundamentalmente con una base de datos de URL clasificadas que rigen el acceso de los usuarios así como una interfaz que facilita las tareas de administración. Su arquitectura en forma de *plugin* permite añadir nuevas características sin necesidad de modificar las actuales y el desacoplamiento entre sus componentes permite la distribución de los mismos en la red. Dicho sistema permite mejorar la productividad de los usuarios en los centros de trabajo así como hacer un uso sano y saludable de Internet tanto para el usuario como para la institución (18).

Características.

- ✓ **Soporte para usuarios de tipo IP.**

  Es una característica muy útil en instituciones donde el control de navegación de Internet no se hace a través de usuarios. Los cibercafés o locales públicos de acceso a Internet constituyen buenos ejemplos donde se pudiera establecer diferentes configuraciones de navegación según las direcciones físicas de las máquinas.

- ✓ **Creación de políticas dinámicas.**

  Esta característica le permite al sistema adaptarse a prácticamente cualquier entorno, el administrador podrá implementar la AUP de la institución.

- ✓ **Restricciones de acceso por tiempo y por subredes. Creación de excepciones, expresiones regulares, restricciones de ficheros y escaneo de virus.**

  De manera adicional se pueden añadir reglas personalizadas en el tiempo para la denegación de ficheros y el escaneo de virus.

- ✓ **Redirector en forma de plugins (C++).**

  El núcleo del sistema se encuentra desarrollado en forma de *plugin*, permitiendo así que al sistema se puedan añadir nuevas características sin modificar la lógica actual del sistema.

- ✓ **Sistema de cuotas.**

  Cuenta con un sistema de cuotas que le permite hacer configuraciones bastante avanzadas. Permite la definición de cuotas por tiempo y por megas y por la combinación de ambas cosas.

- ✓ **Reportes de funcionamiento.**

  Cuenta con un módulo para conocer el funcionamiento de Squid, elemento que permite en ciertas circunstancias la toma oportuna de decisiones.

- ✓ **Instalación como componente y distribuida.**

  El sistema podrá ser instalado como componente en una PC previamente instalada, permitiendo así conservar las configuraciones de los administradores en sus servidores. Al mismo tiempo el sistema podrá quedar distribuido en la red, contribuyendo a un mejor aprovechamiento del hardware según los requerimientos de algunos de los componentes del sistema.

- ✓ **Análisis del tráfico entrante de Internet (Protocolo ICAP):**

  En análisis del tráfico entrante de la red amplia el espectro de posibilidades en cuanto a la incorporación de nuevas funcionalidades al sistema. El análisis de virus, detección automática de proxy anónimos y restricciones de ficheros.

Gestión de Usuarios y Grupos.

La aplicación cuenta con un módulo de administración que permite gestionar las políticas de navegación que se aplican a los usuarios, la cuota que se le asigna para el acceso a la Web y los permisos de navegación para así implementar la política de uso aceptable de Internet de forma sencilla a través de la IAW (Interfaz de Administración Web) de SmartKeeper. Entre las diversas opciones que nos brinda el sistema se encuentra la de sincronizar los usuarios a partir de un servidor LDAP agilizando el proceso de registro y control de la navegación en la institución.

Interfaz gráfica principal del Software.

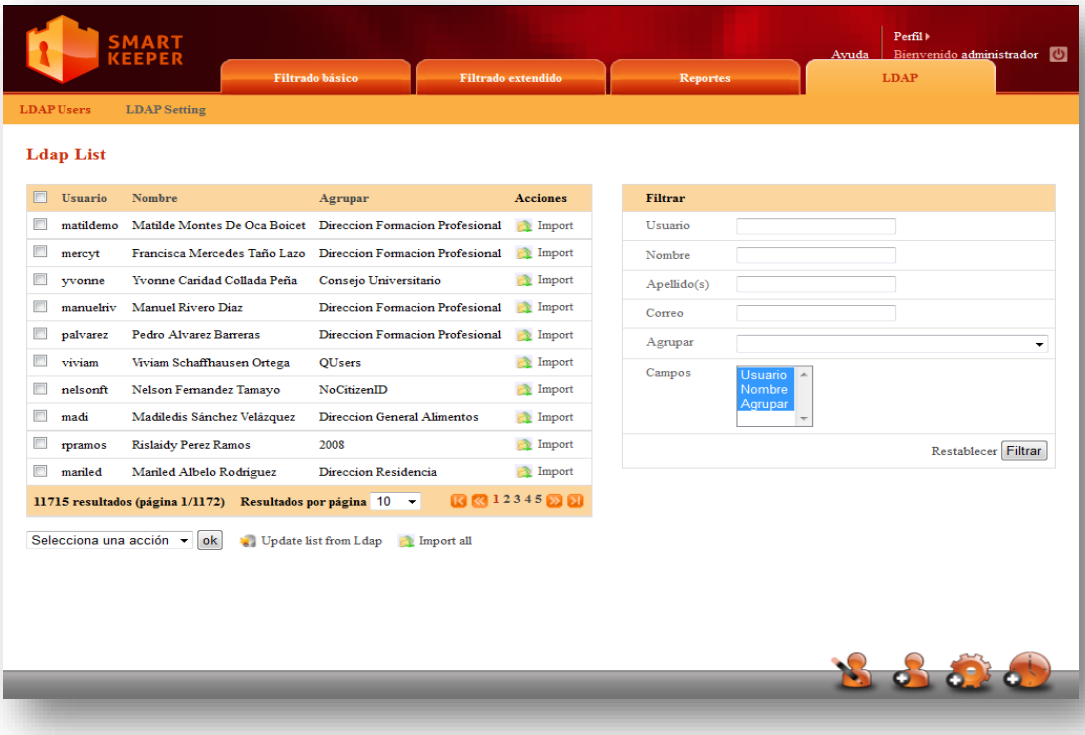


Figura # 5 Interfaz gráfica principal de SmartKeeper.

1.4.3 Observaciones del estudio de las soluciones existentes.

El estudio de las soluciones existentes en Cuba ha permitido identificar algunas características a incluir en la solución, como la posibilidad de sincronizar las áreas y los usuarios a partir de un servidor LDAP a partir del análisis del sistema SmartKeeper. Sin embargo, no se usa este módulo para la sincronización de usuarios en la solución, dado que omite el agrupamiento de usuarios a partir de las áreas lógicas o grupos definidos en el servidor, como no se ha encontrado una herramienta que pueda ser utilizada íntegramente en la solución del problema a resolver, se evidencia la necesidad de crear una herramienta o método propio.

1.5 Metodología de Desarrollo.

El empleo de una metodología de desarrollo de software es un aspecto muy importante, ya que indica el camino a seguir para obtener un producto con calidad. Existen dos grandes enfoques, las metodologías tradicionales y las metodologías ágiles. Las metodologías tradicionales están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto, en cambio, las metodologías ágiles centran su atención en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y a mantener una buena relación con el cliente (19).

OpenUP.

OpenUP es un proceso ágil y unificado, que contiene el conjunto mínimo de prácticas que ayudan a los equipos a ser más eficaces en el desarrollo de software. La metodología fue seleccionada debido a que establece una filosofía pragmática y ágil que se centra en la naturaleza colaborativa de desarrollo de software, además se trata de una herramienta agnóstica, que puede utilizarse tal cual o ampliarse para tratar una amplia variedad de tipos de proyecto.

El OpenUP está organizado en dos dimensiones diferentes pero interrelacionadas: el método y el proceso. El contenido del método es donde los elementos del método (roles, tareas, artefactos y lineamientos) son definidos, sin tener en cuenta como son utilizados en el ciclo de vida del proyecto. El proceso es donde los elementos del método son aplicados de forma ordenada en el tiempo. Muchos ciclos de vida para diferentes proyectos pueden ser creados a partir del mismo conjunto de elementos del método.

Principios del OpenUP

- Colaborar para alinear intereses y para compartir conocimiento.
- Balancear las prioridades para maximizar las necesidades de los *stakeholder*.
- Centrado en la Arquitectura.
- Desarrollo Iterativo.

Beneficios en el uso del OpenUP.

- Es apropiado para proyectos pequeños y de bajos recursos, permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito.
- Permite detectar errores tempranos a través de un ciclo iterativo.
- Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP.
- Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas.

OpenUP es un proceso mínimo, completo y extensible. Promueve técnicas ágiles y principios, mientras que tiene un ciclo de vida probado y estructurado que hace hincapié en la continua entrega de software que es valioso para los interesados en el desarrollo rápido de aplicaciones de calidad. Además es la metodología empleada para el trabajo en el proyecto productivo en el cual se desenvuelve el problema a resolver para la presente investigación(20).

1.6 Análisis del soporte tecnológico para el desarrollo del software.

La selección de las tecnologías a utilizar en el desarrollo de un software es un paso muy importante, ya que influye directamente en la calidad del producto final y en el esfuerzo necesario para obtenerlo. La interfaz web de presentación de datos de AiresProxyAudit fue creada usando el *framework* Symfony 2 y el lenguaje de programación PHP.

1.6.1 Marco de Trabajo (framework).

Un *framework* es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, con base a la cual otro proyecto de *software* es más fácilmente organizado y desarrollado. Puede incluir soporte para programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, que contribuyan a desarrollar y unir los diferentes componentes de un proyecto. Dicho de otro modo, es un producto que se utiliza como base para la programación avanzada de aplicaciones, que aporta funciones o código para realizar las tareas más básicas, puesto que en el propio *framework* ya hay implementaciones que están probadas, funcionan y no es necesario volver a programarlas. Para el desarrollo de este trabajo se propone la utilización de los *framework* que se relacionan a continuación.

Symfony 2.0.

Symfony es un *framework* diseñado para optimizar el desarrollo de aplicaciones web. Entre sus características se encuentra la de separar la lógica de negocio, la lógica de servidor y la presentación de la aplicación web, así como proporcionar varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de un sistema web complejo; además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel; es compatible con varios gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft (21).

Características de Symfony (22):

- Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Unix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además es lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

jQueryUI 1.9.0

jQuery es un *framework* para el lenguaje JavaScript, que implementa una serie de clases para programar sin tener en cuenta el navegador para el cual estará destinada la aplicación. Ofrece una infraestructura que facilita la creación de aplicaciones complejas del lado del cliente. jQuery ayuda a la creación de interfaces gráficas de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax. Este *framework* tiene licencia para su uso en cualquier tipo de plataforma.

jQuery UI es una biblioteca de componentes para el *framework* jQuery que le añaden a este un conjunto de *plugins* [9], *widgets* [10] y efectos visuales para la creación de aplicaciones web.

JqueryTreeview es una librería del *framework* que permite una transformación ligera y flexible de una lista desordenada en un árbol expandible y plegable, ideal para mejoras de navegación discreta. Proporciona algunas opciones de personalización, una extensión árbol-asíncrono y una extensión experimental para ordenar. Esta librería soporta además el uso de animaciones. Toma una lista desordenada y hace que los elementos se muestren de una forma plegable, permitiendo expandir y contraer el árbol de elementos según se requiera. También permite añadir clases y reglas para los elementos raíz y permite mostrar el árbol completo creando una vista de los elementos padres con sus respectivos hijos de una forma jerárquica (22).

Bootstrap 2.2.2

Bootstrap es un *framework* desarrollado para simplificar el proceso de creación de diseños web. Para ello ofrece una serie de plantillas CSS[11] y de ficheros JavaScript que permiten conseguir interfaces web que funcionen en los navegadores actuales; un diseño que pueda ser visualizado de forma correcta en distintos dispositivos y a distintas escalas y resoluciones; una mejor integración con las librerías usadas habitualmente, como por ejemplo jQuery; un diseño sólido basado en herramientas actuales y potentes, o estándares como CSS3/HTML5[12].

Justificación de la selección

Para los desarrolladores el empleo del *framework* agiliza y facilita el trabajo; mayormente son utilizados con el objetivo de evitar la redundancia de código y fomentar la reutilización. La selección de los *framework* anteriores se sustenta en que:

- ✓ Symfony ha sido solicitado por el usuario final, pues tiene referencias de que es fácil de instalar y configurar en la mayoría de las plataformas, es independiente del sistema gestor de bases de datos, es sencillo y flexible, implementa la mayoría de las mejores prácticas y patrones de diseño para la web. Asimismo está preparado

[9] Módulos de hardware o software que añaden una característica o un servicio específico a un sistema más grande.
[10] Pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños utilizados para dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.
[11] Siglas del Inglés *Cascade Style Sheet* (Hojas de Estilo en Cascada).
[12] Siglas del *Inglés Hyper Text Markup Language* (Lenguaje de Marcado de Hipertexto).

para aplicaciones empresariales y es adaptable a las políticas y arquitecturas propias de cada empresa.

- ✓ Symfony, para los desarrolladores genera un código fácil de leer y permite un mantenimiento sencillo. Es fácil de extender lo que posibilita su integración con librerías desarrolladas por terceros.
- ✓ jQueryUI permite la creación de interfaces gráficas de usuario con efectos dinámicos y visuales compatibles con los navegadores especificados por el cliente y mucho más atractivos para el usuario. Además la librería JqueryTreeview se utiliza para la presentación de la estructura organizacional en forma de árbol jerárquico.
- ✓ Bootstrap facilita el trabajo con el lenguaje CSS garantizando la compatibilidad entre dispositivos a distintas escalas y resoluciones.

1.6.2 Lenguajes de programación.

Los lenguajes para la Web son aquellos asimilados directamente por el navegador y que no necesitan pre tratamiento. En la actualidad existe una gran diversidad de ellos, que han surgido como consecuencia de las tendencias y necesidades de las distintas plataformas de desarrollo. El uso de estos lenguajes permite un diseño óptimo, eficiencia, seguridad y dinamismo en las aplicaciones web.

PHP 5.4.

PHP es un lenguaje orientado al desarrollo de aplicaciones web dinámicas con acceso a información contenida en bases de datos. Es sencillo y fácil de aprender, además de poseer una amplia documentación en su sitio web oficial. Entre sus características esenciales destaca el hecho de ser libre, multiplataforma, permitir el manejo de excepciones y aplicar técnicas de la programación orientada a objetos.

PHP 5 es la versión más reciente de un lenguaje útil para la creación de aplicaciones web; se trata de una renovación total que se ha extendido a numerosos aspectos, como la interacción con otras tecnologías y la sintaxis de orientación a objetos. Independientemente de sus potencialidades es imprescindible en caso de la utilización del *framework* de desarrollo Symfony.

Ventajas (21):

1. Velocidad: está escrito en C, por lo que se ejecuta rápidamente utilizando poca memoria.
2. Estabilidad: utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y

estable.

3. Simplicidad: los usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente debido a que la sintaxis es similar.
4. Puede interactuar con varios motores de bases de datos relacionales tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL y además Bases de Datos no Relacionales como MongoDB.
5. Lenguaje de código abierto, lo que posibilita la actualización y corrección de problemas por la amplia comunidad con que cuenta.
6. Multiplataforma, permite ser ejecutado bajo varias versiones de Unix, Windows y Macintosh.

Desventajas (21):

1. Todo el trabajo lo realiza el servidor ya que el código PHP se ejecuta en este. Por tanto, puede ser más ineficiente a medida que las solicitudes aumentan.
2. La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.

HTML 5.

HTML es el lenguaje de marcado predominante en el desarrollo de páginas web; HTML 5 es una actualización de HTML, el lenguaje en el que es creada la web. También es un término de marketing para agrupar las nuevas tecnologías de desarrollo de aplicaciones web: HTML 5, CSS 3 y Javascript. Aunque aún es un lenguaje en desarrollo, los principales navegadores, en sus últimas versiones, reconocen muchos de los elementos que aporta. Entre sus características resalta la reducción de la necesidad de *plugins* externos y un mejor manejo de errores y validaciones.

CSS 3.

CSS es un lenguaje para definir el estilo o la apariencia de las páginas web escritas con HTML o de los documentos XML[13]. Concede separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas web. Su novedad más importante consiste en la incorporación de mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos que a menudo complicaban el código.

[13] Siglas del Inglés *Extensible Markup Language* (Lenguaje de Marcas Extensible), es un metalenguaje extensible de etiquetas, una manera de definir lenguajes para diferentes necesidades.

JavaScript

JavaScript es un lenguaje de programación interpretado que se utiliza principalmente para crear páginas web dinámicas. Permite mejoras en la interfaz de usuario y páginas web para agregar funcionalidad, validar formularios y comunicarse con el servidor. Es un lenguaje basado en prototipos, imperativo y dinámico. La mayoría de los navegadores en sus últimas versiones interpretan código JavaScript.

Justificación de la selección

Las aplicaciones web son una de las tendencias modernas en el desarrollo de *software*. El uso de lenguajes para este tipo de implementaciones resulta indispensable para el éxito del presente trabajo. La selección de los lenguajes anteriores se sustenta en que:

- ✓ El uso de Symfony como *framework* de desarrollo deriva que sean empleados los lenguajes de programación: PHP, HTML, CSS y JavaScript.
- ✓ PHP: Symfony es un *framework* para PHP y este lenguaje proporciona a la propuesta de solución un alto grado de dinamismo y seguridad.
- ✓ HTML permite definir la estructura y el contenido de las páginas, permitiendo combinar textos, imágenes, sonidos, vídeos y enlaces a otras páginas.
- ✓ CSS permite la separación entre las normas de presentación y el contenido a mostrar en la aplicación.
- ✓ JavaScript es el encargado de interpretar numerosas instrucciones y ejecutarlas para realizar efectos y permitir la interacción con las páginas web de una manera más dinámica.

1.6.3 Sistema Gestor de Base de Datos.

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: *Database Management System*) es un sistema de *software* que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Un SGBD relacional es un modelo de datos que facilita a los usuarios describir los datos que serán almacenados en la base de datos junto con un grupo de operaciones para manejar los datos que se encuentran interrelacionados por las tablas (22).

Base de datos no relacionales.

Las bases de datos No SQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación. Las bases de datos No SQL no imponen

una estructura de datos en forma de tablas y relaciones entre ellas (no imponen un esquema pre-fijado de tablas), en ese sentido son más flexibles, ya que suelen permitir almacenar información en otros formatos como clave-valor. Además de la carencia de un esquema predeterminado, es que están pensadas para manipular enormes volúmenes de información de manera rápida y están preparadas para escalar horizontalmente sin perder rendimiento.

Las características principales de una base de datos No SQL (23):

1. Modelos de datos variables y flexibles: siempre hay un modelo, una estructura, pero la rigidez del modelo relacional desaparece.
2. Escalabilidad sencilla, pueden responder a necesidades pequeñas y a altos volúmenes de trabajo.
3. Alta velocidad de respuesta a peticiones.
4. Diferentes lenguajes de consulta (no SQL).

MongoDB.

Un estudio enfocado en las base de datos no relacionales arroja que MongoDB constituye una solución escalable y de alto rendimiento de almacenes de datos No SQL.

Es un sistema de código abierto y escrito en C++, orientado al almacenamiento de datos en documentos al estilo BSON para almacenar datos. El protocolo BSON, una extensión personalizada de JSON, soporta los tipos de datos adicionales (por ejemplo ObjectId, fecha y hora) que no forman parte de la especificación JSON. Se destaca por conservar los índices de todos los atributos y hacer mucho más flexible la agregación y procesamiento de datos(23) .

MongoDB se basa en colecciones, o sea, los datos se agrupan en conjuntos llamados "colecciones". Cada colección tiene un nombre único en la base de datos y puede contener un número ilimitado de documentos. Una colección es análoga a una tabla, excepto que no tienen un esquema definido y un documento es el análogo a una tupla de una tabla.

Teniendo en cuenta las características anteriormente descritas, se decide utilizar como SGBD a MongoDB, por lograr un mejor manejo de la información frente a un volumen considerable de datos, pues se utilizará para el almacenamiento de los datos obtenidos de los registros de navegación de los usuarios. Además de registrar la información concebida de la estructura organizacional, los grupos de usuarios, los dominios, las clasificaciones de dominios y las configuraciones de los diferentes

factores de la cuota. Permite una fácil integración con el lenguaje de programación seleccionado, además de que el framework a utilizar, Symfony 2.0, posee soporte para MongoDB.

1.6.4 Herramientas para el control de versiones.

Estos sistemas facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas, además permiten volver a una salva previa de los cambios en caso de error o pérdida de datos, además de permitir el desarrollo colaborativo por varias personas en diferentes equipos.

Subversion.

Subversion es un sistema de control de versiones, libre y de código abierto, el *software* maneja ficheros y directorios sobre el tiempo creando un árbol de ficheros en un repositorio central; el repositorio es como un ordinario servidor de archivos, salvo que recuerda todos los cambios hechos a sus ficheros y directorios, esto permite recuperar versiones antiguas de sus datos, o examinar la historia de cómo cambiaron sus datos (25).

Subversion fue seleccionado como herramienta para el control de versiones debido a que se puede acceder a su repositorio a través de la red, lo que permite que pueda ser utilizado por varias personas en diferentes equipos; la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración en el desarrollo de proyectos, lo que propicia progresar más rápidamente sin un único conducto a través del cual todas las modificaciones deben ocurrir. Y debido a que el trabajo se versiona, no tienen por qué temer que la calidad es la compensación por la pérdida de ese conducto, si algún cambio incorrecto se hace en los datos, simplemente se deshace el cambio volviendo a una versión anterior y se resuelve el problema.

RapidSVN.

RapidSVN es una veloz plataforma gráfica para el sistema de revisión de Subversion, el *software* está escrito en C++ utilizando el *framework* de desarrollo wxWidgets. Esta aplicación está licenciada bajo la v3 GNU GPL.

Ventajas (26):

- **Simple**: Proporciona una interfaz fácil de usar para las funciones de Subversion.

- **Eficiente**: Sencillo para los principiantes, pero lo suficientemente flexible como para aumentarla productividad de los usuarios de Subversion con experiencia.
- **Portable**: Funciona en cualquier plataforma en la que Subversion y wxWidgets puede ejecutar: Linux, Windows, MacOS, Solaris, etc.
- **Rápido:** Completamente escrito en C+ +.

Esta plataforma fue seleccionada como plataforma para la revisión de la herramienta de control de versiones Subversion por las facilidades que ofrece a la hora de interactuar con el mismo y lo sencillo de operar.

1.7 Herramientas para el desarrollo del software.

1.7.1 Sistema Operativo.

La plataforma seleccionada por los desarrolladores es Linux, específicamente el uso de **Debian**. Esta distribución GNU/Linux es caracterizada principalmente por ser un proyecto netamente comunitario que a lo largo de los años ha sido capaz de brindar programas de calidad empaquetados en una distribución igualmente de calidad (27).Los sistemas UNIX ofrecen una gama de prestaciones en materia de seguridad, operatividad, disponibilidad y flexibilidad que dotan a los desarrolladores de un entorno de trabajo amigable y confiable lo cual facilita las condiciones de trabajo para el desarrollo del *software*.

1.7.2 Entorno de Desarrollo Integrado (IDE).

Un Entorno de Desarrollo Integrado es una herramienta informática que facilita el trabajo a los programadores, brindándoles un conjunto de herramientas de programación.

NetBeans

NetBeans es un IDE gratuito y de código abierto, empleado en el desarrollo de aplicaciones web, de escritorio y para móviles. Se encuentran disponibles versiones de NetBeans para los sistemas operativos Windows, Linux, Macintosh y Solaris (28).

Para el desarrollo de la solución se decide utilizar la versión 7.2 de IDE, la base en la que se sustenta su elección es que permite desarrollar aplicaciones utilizando el *framework* Symfony y ejecutar los comandos de este directamente desde la interfaz del IDE. Por otra parte se posee gran dominio y experiencia en el uso de este IDE para el desarrollo de aplicaciones web en PHP.

1.7.3 Servidor Web.

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix, Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual, presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero no posee una interfaz gráfica en su configuración (29).

Para el desarrollo de la solución se decide utilizar **Apache** pues es un servidor de arquitectura modular que permite la instalación de diferentes módulos según la necesidad del usuario, lo cual facilita un mayor aprovechamiento de las prestaciones que ofrece el servidor y por su experiencia en el trabajo con el mismo anteriormente.

1.7.4 Herramienta CASE.

Visual Paradigm para UML es una herramienta de diseño multiplataforma diseñada para asistir el desarrollo de software. Soporta los principales estándares de la industria, tales como; UML, SysML, BPMN, XMI(30).

Para el desarrollo de la solución se decide utilizar Visual Paradigm para UML pues ofrece un conjunto completo de herramientas que los equipos de desarrollo necesitan para la captura de requisitos, planificación de *software*, planificación de pruebas, modelado de clases, modelado de datos y otras actividades. Todas estas características unidas a la experiencia que se posee en la utilización de esta herramienta apoyaron su elección.

1.8 Conclusiones parciales.

En este capítulo se estudiaron las disímiles aplicaciones existentes en el ámbito nacional e internacional, analizando características, ventajas y desventajas de cada una de ellas. El estudio del estado del arte ha permitido identificar algunas características a incluir en la solución al problema a resolver para la investigación, pero como no se ha encontrado una herramienta que pueda ser utilizada íntegramente en la solución, se evidencia la necesidad de crear una herramienta propia. Basado en este estudio se concluye que es necesario desarrollar un módulo de administración que pueda ser integrado al sistema AiresProxyAudit, complementando así su funcionamiento. Para este proceso de desarrollo se utilizará como metodología OpenUP, recurriendo al Visual Paradigm como herramienta de modelado, con la cual se podrá realizar el análisis y diseño de la aplicación a través del lenguaje de modelado UML. Además utilizará el lenguaje de programación PHP, para la implementación del módulo en plataforma de desarrollo NetBeans.

Capitulo #2 – Características del sistema.

2.1 Introducción.

Un flujo de trabajo es una relación de actividades que producen resultados observables. Dentro de los flujos de trabajo que propone OpenUP, se encuentra el de Requerimientos, en él se establece qué tiene que hacer exactamente el sistema que se construye, por lo que en el presente capítulo se realiza una descripción del módulo a desarrollar y se muestra el modelo de dominio para un mayor entendimiento del flujo de trabajo del mismo. Se capturan los requerimientos funcionales y no funcionales, lo que permite identificar los casos de uso y sus descripciones.

Por último se reflejan los artefactos modelados como resultado del flujo de Análisis y Diseño para la construcción de una aplicación web. Además de los diagramas de clases del diseño con estereotipos web y diagramas de secuencia, se tiene en cuenta, el modelo de despliegue, para tener una mayor visión de los elementos que interactúan en el funcionamiento de la aplicación.

2.2 Descripción del módulo.

El módulo de administración permitirá gestionar la estructura organizacional de la empresa, asignar permisos de auditorías sobre los diferentes departamentos de la institución y gestionar los usuarios que pertenecen a un departamento. Como un valor agregado el módulo de administración brindará varios reportes que visualizan elementos administrativos y estructurales.

2.2.1 Descripción de actores.

Actor	Objetivo
Administrador	Rol responsable de velar por el funcionamiento apropiado del sistema, relacionado con la configuración, permisos y gestión de las principales funcionalidades.

2.2.2 Modelado del dominio.

El modelo de dominio también conocido como Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés. Este representa clases conceptuales del dominio del problema, conceptos del mundo real, no de los componentes de *software*. Una clase conceptual también conocida como entidad puede ser una idea o un objeto físico (símbolo, definición y extensión) (31).

Teniendo en cuenta que la definición de procesos y roles del negocio se hace difícil encontrarlos, es necesario describir el funcionamiento de la aplicación mediante una serie de conceptos, entidades y sus relaciones, agrupándose en un modelo de dominio con el fin de contribuir a la comprensión del contexto del sistema.

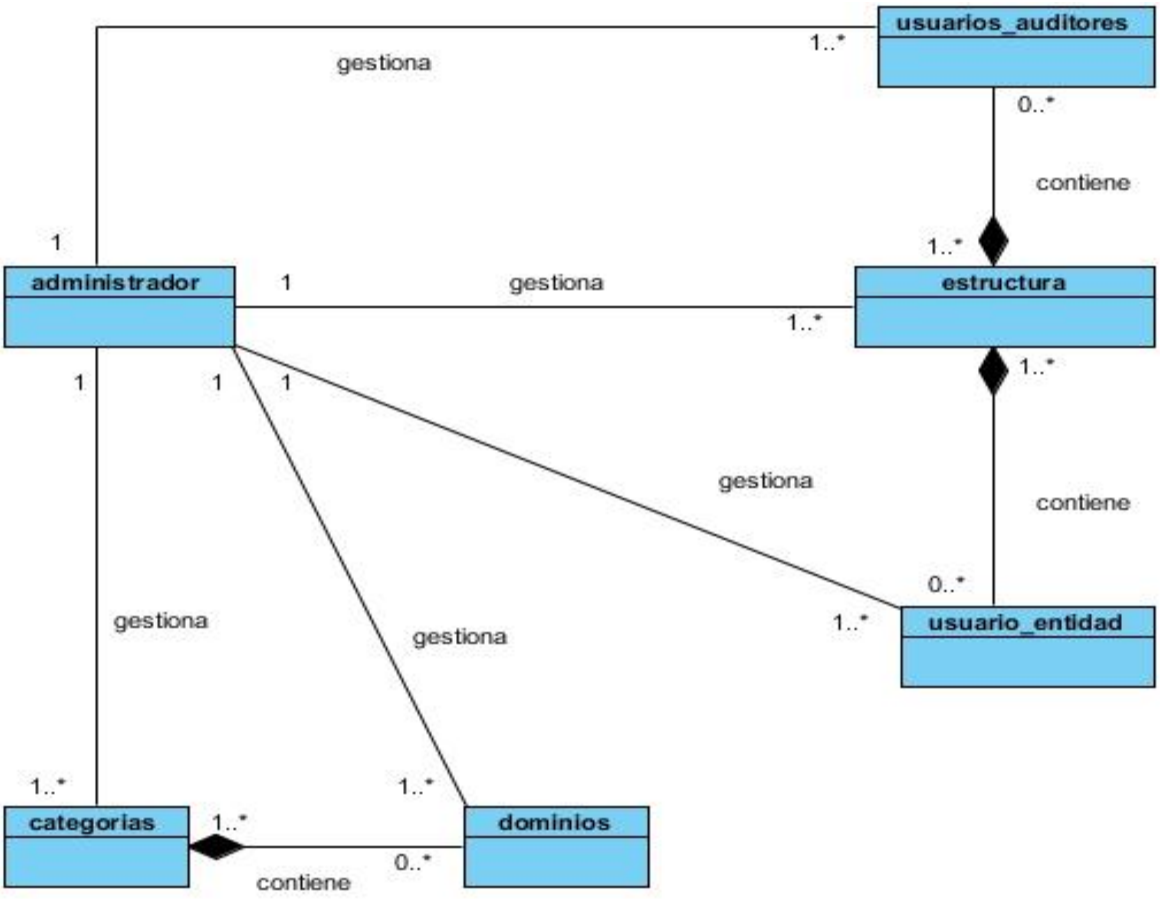


Figura # 6 Diagrama de Clases del Modelo del Dominio.

2.2.3 Descripción de Clases del Modelo del Dominio.

Administrador: Es el usuario encargado de interactuar con el sistema.

Estructura: Es donde se almacenan las áreas de la estructura.

Usuarios_auditores: Es donde se almacenan los usuarios con permisos de auditores.

Usuario_entidad: Se encarga de almacenar los usuarios de la estructura.

Dominio: Es donde se almacenan los dominios de navegación.

Categorías: Es donde se almacenan las categorías en las que se reparten los dominios de navegación.

2.3 Modelado del sistema.

2.3.1 Requisitos funcionales.

Los requerimientos funcionales son capacidades o condiciones que un sistema determinado debe cumplir, estas capacidades dan la medida de la utilidad del sistema. A continuación se describen algunos de los requerimientos funcionales críticos del sistema identificados (31).

RF #1 Adicionar entidad.
RF #2 Modificar entidad.
RF #3 Listar entidades.
RF #4 Eliminar entidad.
RF #5 Exportar estructura organizacional.
RF #6 Importar estructura organizacional
RF #7 Sincronizar estructura con LDAP
RF #8 Adicionar usuario a entidad.
RF #9 Modificar usuario.
RF #10 Buscar usuario.
RF #11 Mostrar usuarios.
RF #12 Eliminar un usuario
RF #13 Adicionar auditor a una entidad.
RF #14 Modificar auditores.
RF #15 Buscar auditor.
RF #16 Listar auditores.
RF #17 Eliminar auditor.
RF #18 Adicionar dominio.
RF #19 Listar dominio.
RF #20 Modificar dominio.
RF #21 Insertar categoría
RF #22 Listar categoría.
RF #23 Modificar categoría
RF #24 Eliminar categoría.

Nº	*Nombre*	*Descripción*	*Complejidad*	*Prioridad para cliente*
RF #1	Adicionar entidad.	El sistema debe permitir crear y guardar (a través de un campo de entrada o a través de un fichero yml) en la base de datos de MongoDB las entidades que conforman la estructura organizacional (en forma de árbol) de una empresa.	*Alta*	*Alta*
RF #2	Modificar entidad.	El sistema debe permitir editar las entidades y su información.	Alta	Alta

Tabla 1 Requerimientos Funcionales del Módulo de Administración.

En caso de necesitar consultar los demás requisitos funcionales y sus descripciones puede examinar el Anexo A.

2.3.2 Requisitos no funcionales.

Los requerimientos no funcionales detallan las propiedades o cualidades que el producto debe tener y hacen al producto atractivo, fácil de usar, rápido y confiable. Como parte de la captura de requerimientos no funcionales se determinaron los siguientes requisitos no funcionales (31).

Tipo	Nº	Nombre	Descripción
Restricciones de Diseño.	RNF #8	Restricciones de diseño del Sistema AiresProxyAudit.	• Lenguajes de programación: el lenguaje definido para el desarrollo del sistema es PHP. El procesamiento de datos y la obtención de resultados se implementarán en C++. • Sistema Operativo: el sistema se desarrollará sobre el Sistema Operativo Linux. El producto final podrá ser instalado en Sistemas Operativos Linux, optimizado para Centos 6. • Control de versiones: como sistema de control de versiones se utiliza el Subversion (SVN). • Herramienta de modelado: Se utiliza Visual Paradigm para UML 5.0 en el diseño de diagramas y artefactos. • Sistema Gestor de Base de Datos: MongoDB.

Tabla 2 Requerimientos no Funcionales del Módulo de Administración.

En caso de necesitar consultar los demás requisitos no funcionales y sus descripciones puede examinar el Anexo B.

2.4 Diagrama de Casos de Uso del Sistema.

Los Casos de Uso son una técnica para capturar y especificar los requisitos del sistema, así como para guiar su diseño, implementación y prueba. Cada Caso de Uso facilita un escenario sin ambigüedad en la interacción entre el actor y el software, así como describe un conjunto de funcionalidades que el sistema desarrollado debe poseer (32).

A continuación se muestra el Diagrama de Casos de Uso del Sistema para el módulo, en el cual muestra las interacciones entre los actores y los Casos de Uso del sistema (CUS) (ver figura 7).

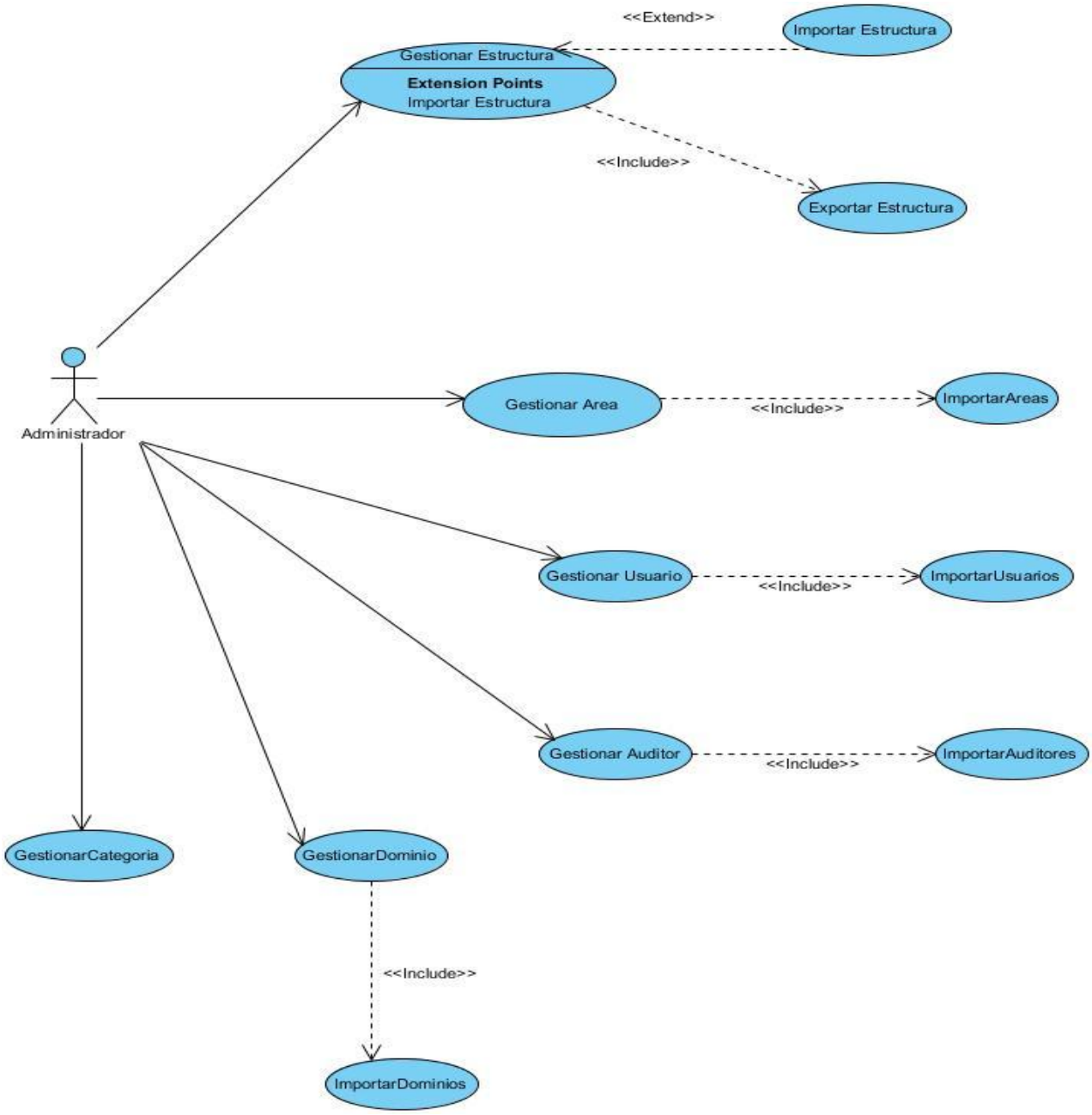


Figura # 7 Diagrama de Casos de Uso del Sistema.

2.4.1 Especificación de Casos de Uso.

CU 1. Adicionar entidad.

Objetivo	Crear y guardar la entidad (estructura organizacional en forma de árbol) de una empresa.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea crear la estructura organizacional de la empresa, el sistema muestra la interfaz correspondiente con la opción Adicionar entidad, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema.
Postcondiciones	Se adicionó la entidad (estructura organizacional) al sistema.

Flujo de eventos		
Flujo normal de eventos		
	Actor	**Sistema**
1.	Selecciona la opción “Crear entidad”.	
2.		Muestra la interfaz con la estructura organizacional predefinida.
3.	Crea la entidad base que representa la empresa.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Muestra una vista en forma de árbol con la estructura organizacional creada.
6.		Termina el CU.

Flujos alternos		
Nº Evento 3a Crear entidad hijo.		
	Actor	**Sistema**
1.	Selecciona entidad padre.	
2.		Muestra la opción “Crear entidad hijo”.
3.	Crea la entidad hijo. Ir paso 4.	
Nº Evento 3b Crear entidad a través de fichero yml.		
	Actor	**Sistema**
1.	Carga fichero yml con la estructura organizacional. Ir paso 4.	
Nº Evento 4a Datos incorrectos.		
	Actor	**Sistema**
1.		Muestra el siguiente mensaje: “Los datos de la entidad son incorrectos”. Ir paso 2.
Asuntos pendientes	Posibles mejoras al caso de uso.	

Tabla 3 CU 1. Adicionar entidad.

CU 3. Listar entidad.

Objetivo	Listar las entidades de una empresa.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea listar las entidades existentes, el sistema muestra la interfaz correspondiente con la opción Listar, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema.

	Se ha insertado al menos una entidad.	
Postcondiciones	Se listaron todas las entidades existentes.	
Flujo de eventos		
Flujo normal de eventos		
	Actor	**Sistema**
1.	Selecciona la opción “Listar entidad”.	
2.		Muestra todas las entidades existentes con su información (Nombre, usuarios y auditores que contiene).
1.		Termina el CU.
Asuntos pendientes	Posibles mejoras al caso de uso.	

Tabla 4 CU 3. Listar entidad.

En caso de necesitar consultar las descripciones de los demás casos de uso puede examinar el Anexo C.

2.5 Descripción del sistema propuesto.

2.5.1 Patrones de diseño implementados por Symfony 2.

¿Qué es un patrón de diseño?

“Cada patrón, describe un problema que ocurre una y otra vez en nuestro entorno, y describe el núcleo de la solución al problema, de manera que puede utilizarse un millón de veces, sin hacer lo mismo dos veces” (35). Los patrones de diseño resuelven problemas comunes a partir de soluciones probadas mediante un diseño orientado a objeto facilitando la reutilización de clases ya existentes. Dentro de los elementos más importantes que componen un patrón, se destacan el nombre del patrón, el problema que resuelve y la solución al mismo.

El *framework* Symfony 2 implementa una serie de patrones de diseño que hacen su arquitectura sea suficientemente robusta y a la vez flexible como para adaptarse a los casos más complejos. A continuación se especifican algunos patrones y ejemplos de uso dentro de la solución propuesta:

Front Controller (Controlador Frontal): Es un patrón de diseño web usado como único punto de entrada a la aplicación. Realiza tareas comunes a todos los

controladores, ejemplo, manejo de la seguridad, de las peticiones de los usuarios, carga de la configuración de la aplicación y delega la responsabilidad de responder a las peticiones al módulo específico que tiene la acción enviada en la petición.

Decorator (Envoltorio): Normalmente en las aplicaciones web existen contenidos que son comunes a todas las páginas que conforman la aplicación. Symfony haciendo uso de este patrón de diseño estructural delimita el código común en todas las páginas (definido en un archivo global para todas las vistas denominado *layout*) del código HTML generado como respuesta a una petición determinada (denominado plantilla). El contenido de la plantilla se integra en el *layout*, por lo que se puede afirmar que el *layout* decora la plantilla.

Expert (Experto): Es un patrón de asignación de responsabilidades (GRASP, del inglés *General Responsibility Assignment Software Patterns*) que delega las responsabilidades a quién contiene la información necesaria para cumplirlas. Tal es el caso de las clases que genera la librería Doctrine con todas las operaciones o métodos útiles que consolidan un desarrollo ágil y una abstracción del sistema gestor de bases de datos empleado.

Bajo Acoplamiento: El patrón bajo acoplamiento impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que lleve a los resultados negativos que puede producir un acoplamiento alto. No soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio. El mismo no se puede considerar de manera aislada a otros patrones como el Experto o el de Alta Cohesión, sino que necesita incluirse como uno de los diferentes principios de diseño que influyen en una elección, al asignar una responsabilidad.

Se evidencia en todo el sistema ya que las clases controladoras del sistema no se relacionan entre sí, lo que disminuye las dependencias entre las mismas al nivel de no existir alguna.

Alta Cohesión: Este patrón es un principio a tener en mente durante todas las decisiones de diseño. Constituye un objetivo subyacente a tener en cuenta continuamente. Es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño. Como beneficios que este aporta se pueden mencionar:

- ✓ Se incrementa la claridad y facilita la comprensión del diseño.
- ✓ Se simplifican el mantenimiento y las mejoras.
- ✓ Se soporta a menudo bajo acoplamiento.

Se puede observar claramente en el sistema ya que cada clase controladora se ajusta a manejar solo las responsabilidades correspondientes a las entidades con las que se relaciona, además para cada página cliente existe una página servidora encargada de manejar sus solicitudes poniéndose de manifiesto el patrón de diseño alta cohesión.

Controlador: Es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. La mayor parte de los sistemas reciben eventos de entrada externa, en cualquiera de los casos que puedan existir, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada.

Se pone de manifiesto en todo el sistema ya que cada uno de los eventos generados por el usuario es redirigido a una clase controladora que realice las operaciones solicitadas, pero siempre manteniendo las clases controladoras sin saturarse es decir siempre manteniendo la alta cohesión.

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica del mismo es encontrar un creador que necesite conectarse al objeto creado en alguna situación, eligiéndolo como el creador.

Este patrón es usado principalmente para el trabajo con las entidades que se guardan en la base de datos, ejemplo de su utilización es en el caso de uso crear estructura, ya que la clase EstructuraController es la encargada de crear los objetos de tipo estructura organizacional, que luego serán almacenadas en la base de datos debido a que obtiene todos los datos necesarios para la inicialización de los mismos.

2.5.2 Patrón arquitectónico implementado por Symfony 2.

La arquitectura Modelo-Vista-Controlador (MVC) fue introducido en los años 70 por desarrolladores de la plataforma SmallTalk y marcó un paso de avance en la arquitectura de las aplicaciones web.

La arquitectura MVC se encuentra dentro de la clasificación de los estilos de llamada y retorno y tiene como objetivo separar la lógica de negocios de la interfaz gráfica de manera que cambios en la misma no afecten la lógica de negocios y viceversa. Está compuesto por tres componentes, el modelo que representa la información que tanto el usuario como la aplicación puede manipular, la vista que implica todos los elementos que componen la interfaz gráfica y el controlador que maneja la interacción y la comunicación entre el modelo y las acciones del usuario.

Algunas ventajas de esta arquitectura son (36):

1. Facilita la agregación de múltiples representaciones de los mismos datos.

2. Crea independencia de funcionamiento.
3. Facilita el mantenimiento de errores.
4. Alta escalabilidad pues se puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware.

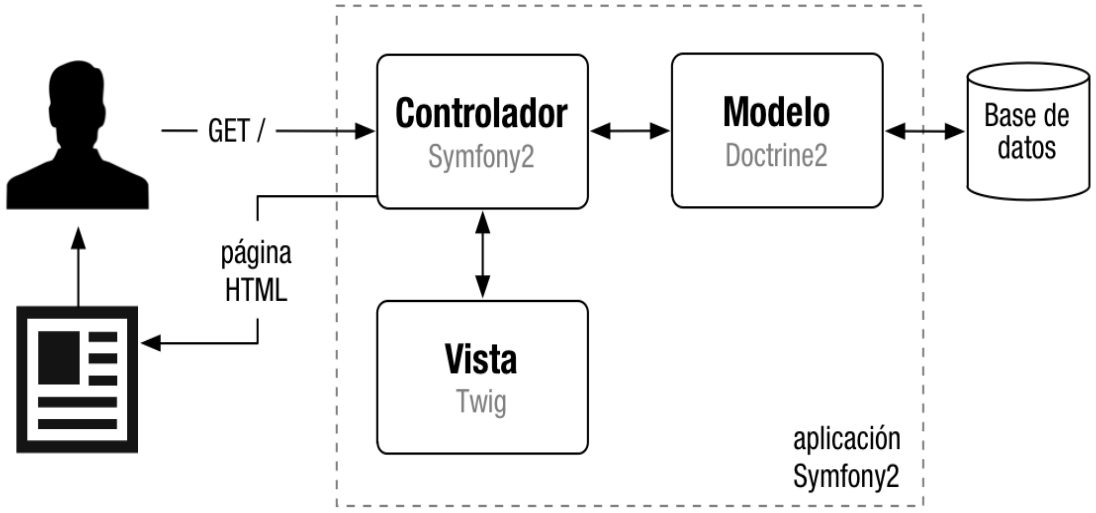


Figura # 8 Diagrama del Patrón Arquitectónico MVC.

"Symfony toma lo mejor de la arquitectura Modelo Vista Controlador y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo" (37). La capa referente al controlador dentro de la arquitectura MVC, Symfony la divide en un controlador frontal y acciones, ubicando el controlador frontal en el directorio web del proyecto.

La capa del modelo está dividida en dos capas, una para la abstracción del gestor de bases de datos y la otra es la capa de acceso a los datos, con lo que logra aplicaciones independientes a los sistemas gestores de base de datos. Una gran ventaja que tiene Symfony con respecto a muchos otros *framework* de desarrollo radica en el empleo de tareas automatizadas a través de la interfaz de líneas de comandos y el uso de la librería Doctrine o Propel para el mapeo de objetos relacionales (ORM), encargada de la generación automatizada de las clases necesarias para el acceso a los datos, a partir del esquema de la base de datos escrito en XML o YAML[14].

Por otra parte la capa de vista, está separada en tres partes: los elementos comunes a todas las páginas de la aplicación, denominada *layout*; las plantillas, encargadas de visualizar las variables definidas en el controlador; y la lógica de las vistas, definida a través de los slots y componentes (fragmentos de vistas que contienen lógica de aplicación y pueden ser reutilizables).

[14] Por sus siglas en inglés *YAML Ain't Another Markup Language (YAML no es otro Lenguaje de Marcado)*

"Symfony2 no es un *framework* MVC. Symfony2 sólo proporciona herramientas para la parte del Controlador y de la Vista. La parte del Modelo es responsabilidad del desarrollador, aunque existen librerías para integrar fácilmente los ORM más conocidos, como Doctrine y Propel" (38).La separación de la vista y el modelo trae ventajas; es posible tener diferentes representaciones de la misma información, haciendo uso del mismo código dentro del modelo. Es posible además programar el código del modelo, abstrayéndose de la representación visual que se le dará a la información.

2.5.3 Arquitectura del módulo.

Para darle cumplimiento al problema planteado se creó un proyecto Symfony 2 que consta de dos bundles:

ConfiguraciónBundle: Es el submódulo que se encarga de la gestión de la estructura organizacional, los usuarios de navegación y los auditores de navegación, así como de los permisos de los mismos sobre las diferentes áreas de la estructura.

CuotaBundle: Es el submódulo que se encarga de la gestión de los dominios de Internet y de las categorías de navegación a las cuales pertenecen para su posterior evaluación en el cálculo del consumo de la cuota de navegación.

2.5.4 Seguridad en el sistema.

En el desarrollo de aplicaciones web, uno de los aspectos más importantes, es la garantía de la confidencialidad, integridad y disponibilidad de la información. Las aplicaciones web se ven afectadas en este sentido por una serie de vulnerabilidades del propio código fuente, muchas veces por descuido o malas prácticas de programación. En este sentido Symfony implementa una serie de mecanismos para garantizar aplicaciones seguras.

Además de la seguridad que proporciona el *framework* Symfony 2, se estableció el mecanismo de autenticación al sistema basado en usuario y contraseña, y se utilizó el modelo de control de acceso basado en roles, creándose una colección en la base de datos donde se registran los usuarios con permisos de administración en el sistema. Con estos procedimientos se asegura que sólo un usuario correctamente identificado pueda acceder al sistema, garantizando la confidencialidad e integridad de los datos, y estos usuarios solo podrán accionar sobre la información precisa que por el rol establecido en el sistema pueden controlar.

2.6 Diagramas de Clases del Análisis.

Los diagramas de clases del análisis constituyen una vista de la futura composición de las clases del *software*. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama y se

modifica para satisfacer los detalles de las implementaciones (39). A continuación se muestran algunos de los diagramas de clases del análisis de la solución propuesta:

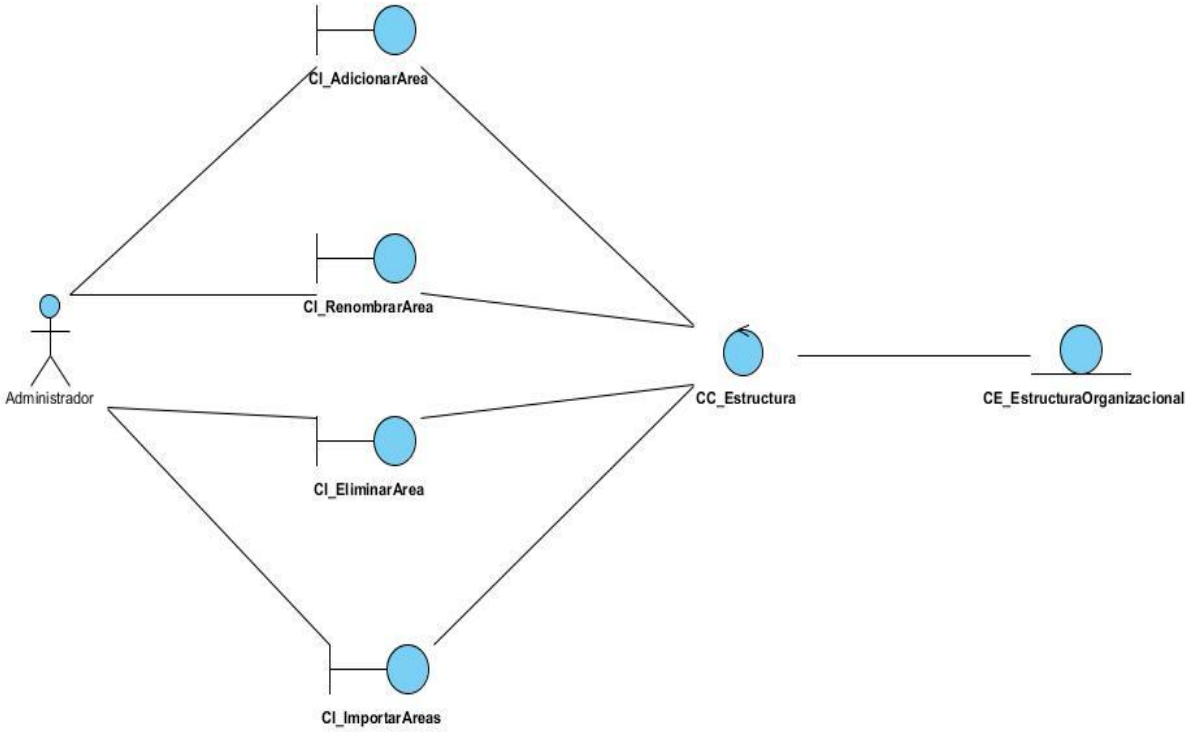


Figura # 9 Diagrama de Clases del Análisis. CU Gestionar Áreas.

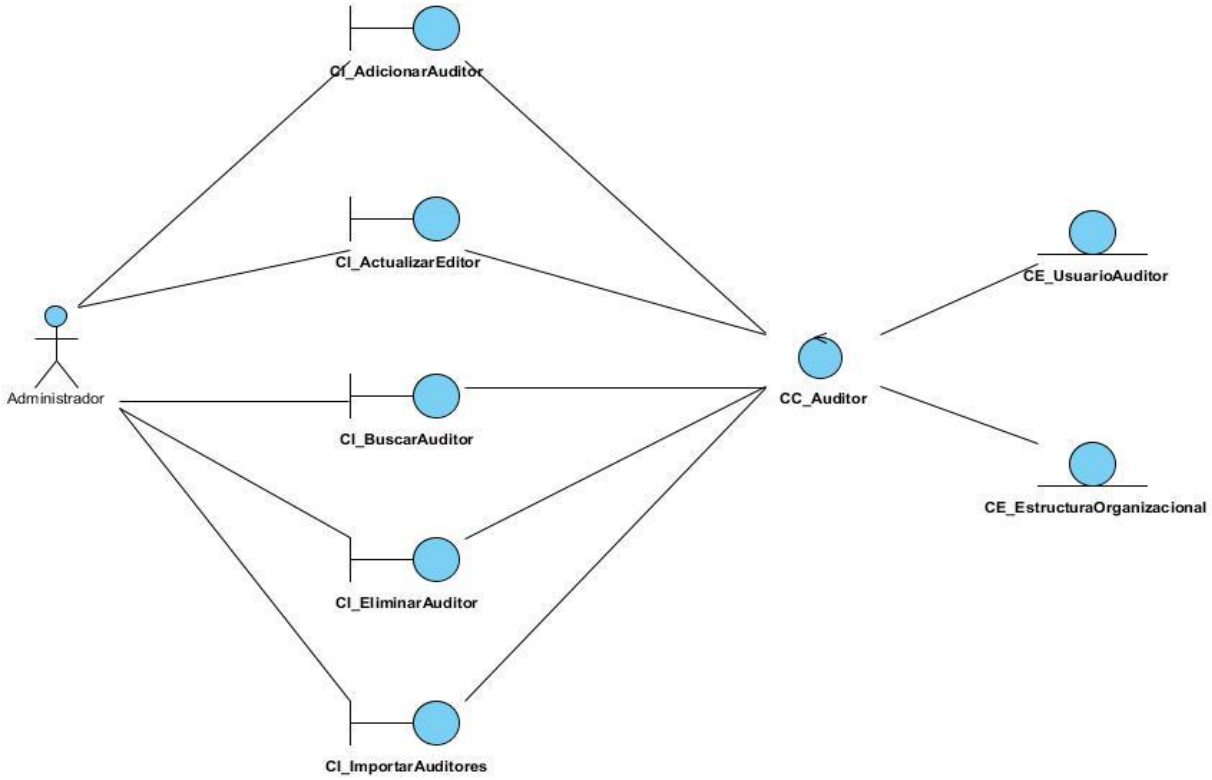


Figura # 10 Diagrama de Clases del Análisis. CU Gestionar Auditor.

En caso de necesitar consultar los diagramas de clases del análisis de los demás casos de uso puede examinar el Anexo D.

2.7 Diagramas de Clases del Diseño.

Los diagramas de clase del diseño se usan para describir la estructura de un sistema, formalizan el conocimiento del dominio de la aplicación. Las clases son abstracciones que especifican los atributos y comportamientos de un conjunto de objetos (40). Los siguientes diagramas muestran las clases del diseño con estereotipos web de los caso de uso críticos del sistema.

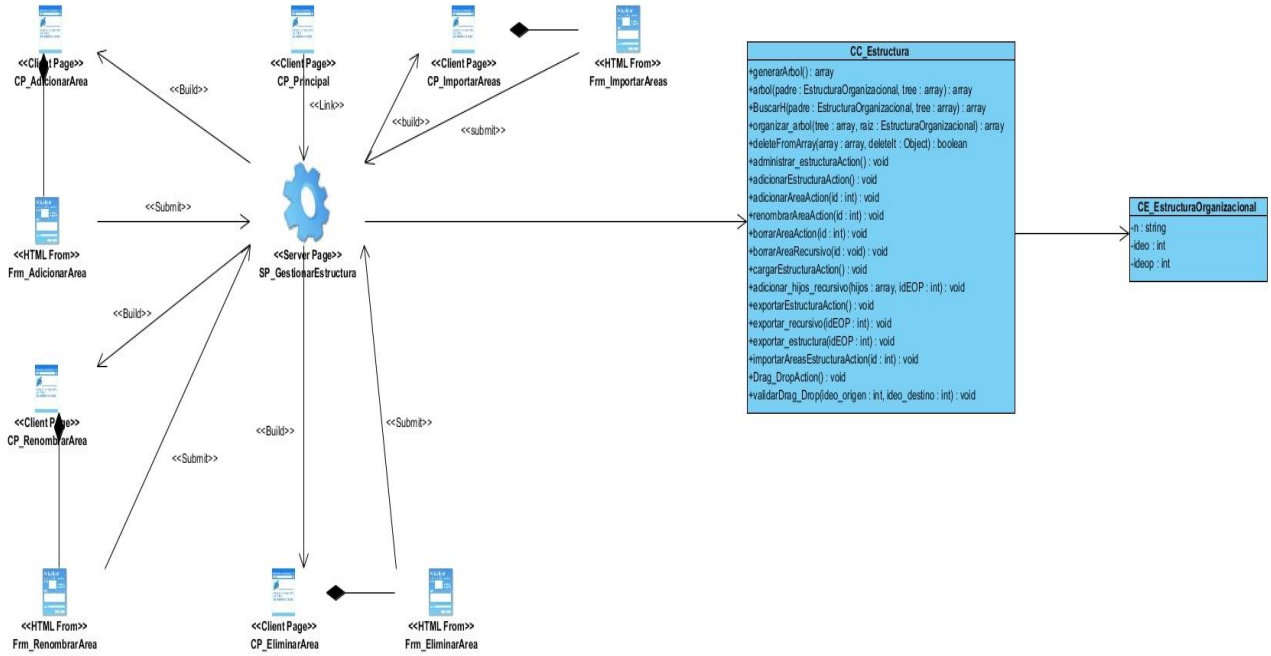


Figura # 11 Diagrama de Clases del Diseño. Gestionar Áreas.

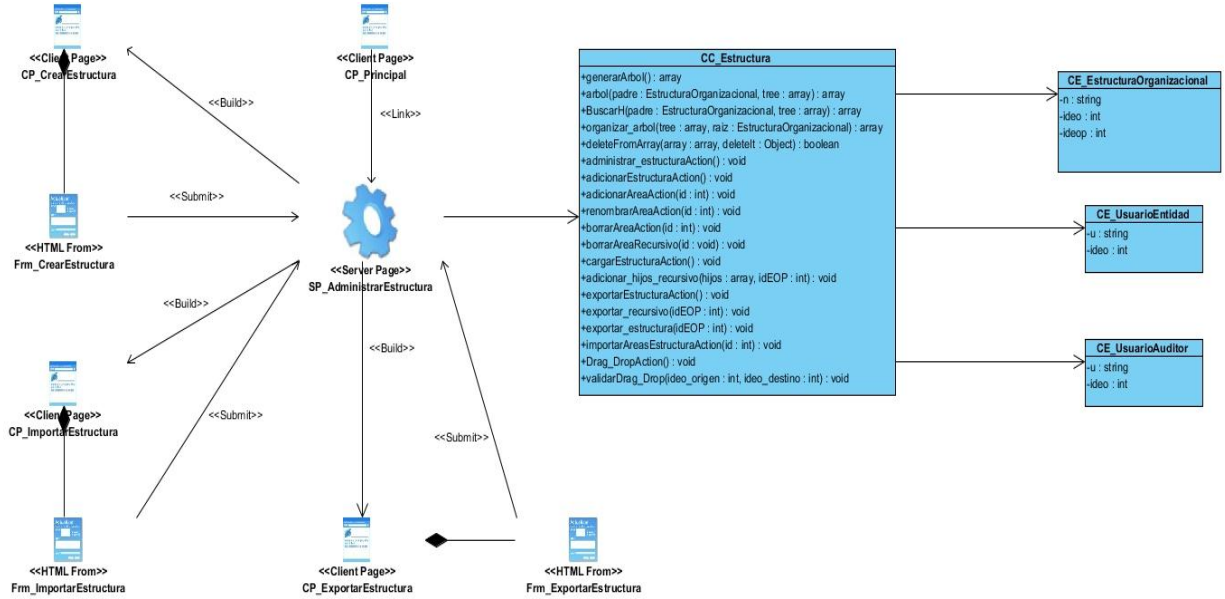


Figura # 12 Diagrama de Clases del Diseño. Gestionar Estructura.

En caso de necesitar consultar los diagramas de clases del diseño de los demás casos de uso puede examinar el Anexo E.

2.8 Diagramas de Interacción.

Los diagramas de interacción representan la forma en que un cliente u objeto se comunica entre sí en petición a un evento. No son sólo importantes para modelar los aspectos dinámicos de un sistema, sino también para construir sistemas ejecutables por medio de ingeniería directa e inversa (41).

Los siguientes diagramas muestran la interacción entre las clases y los actores de los CU críticos del sistema.

2.8.1 Diagramas de Secuencia.

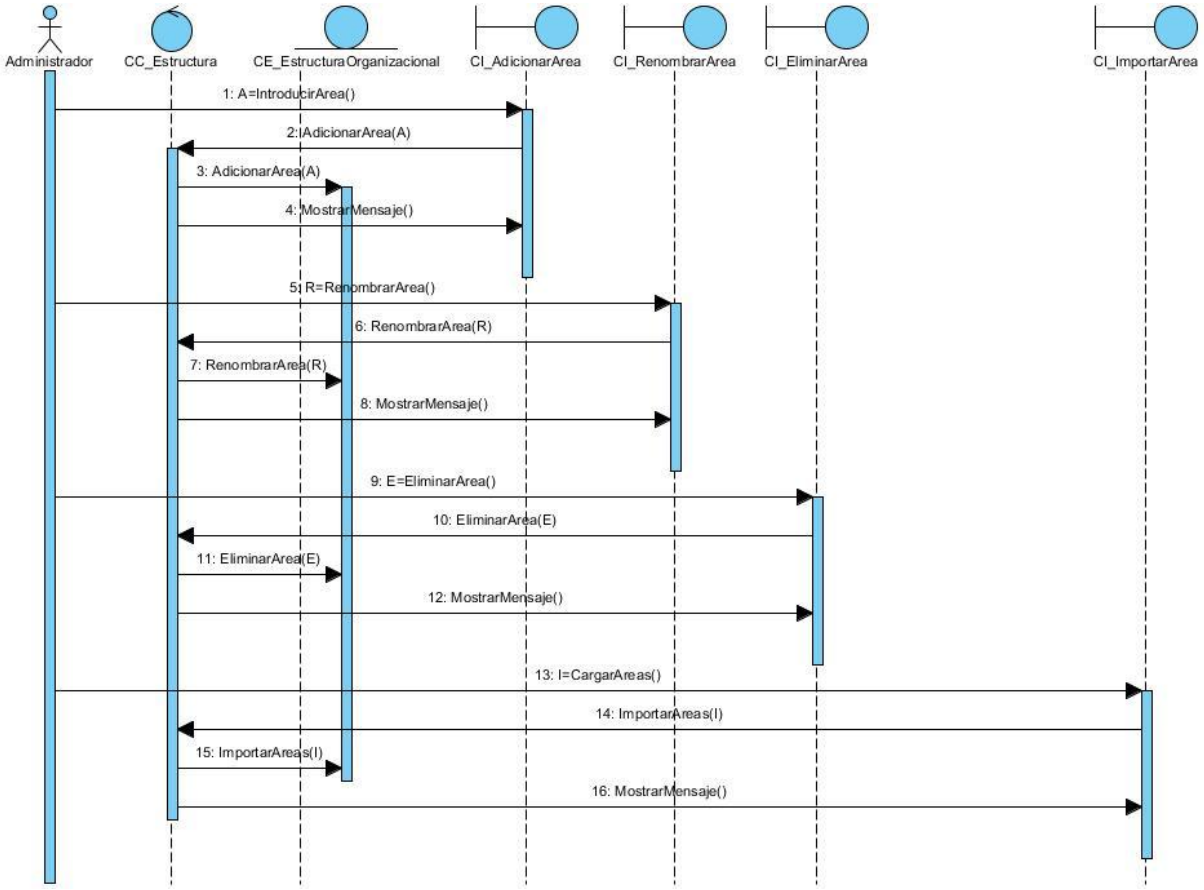


Figura # 13 Diagrama de Secuencia. Gestionar Área.

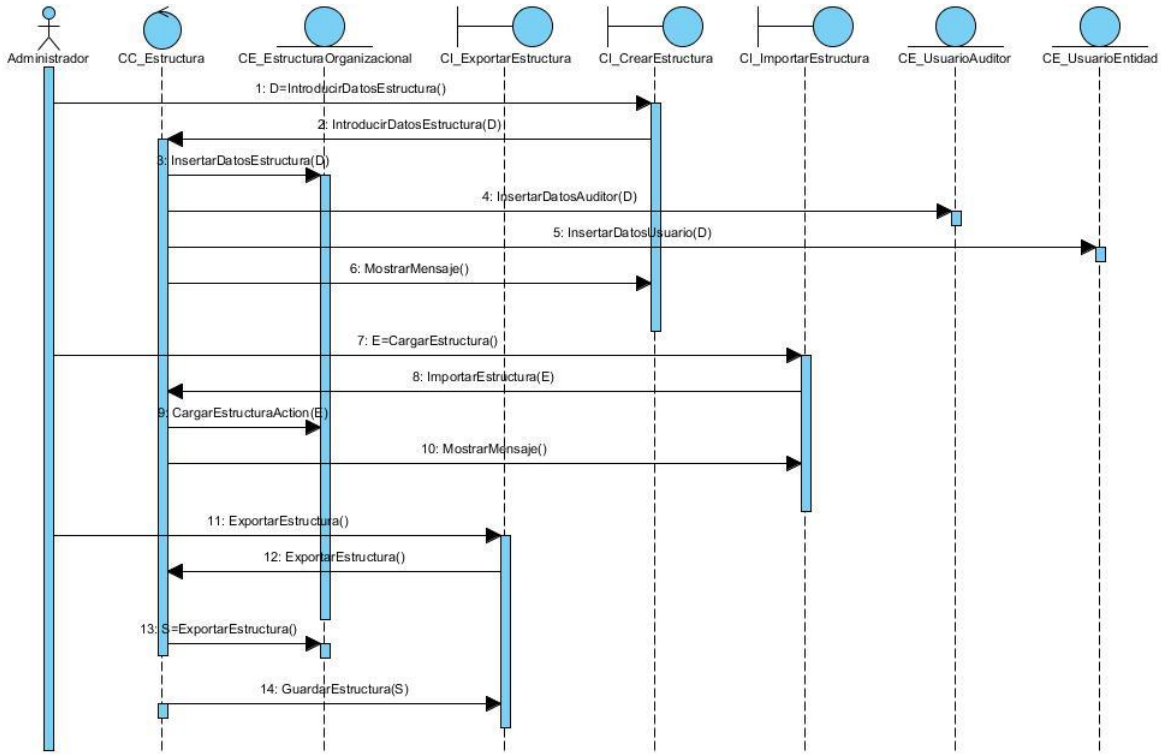


Figura # 14 Diagrama de Secuencia. Gestionar Estructura.

En caso de necesitar consultar los diagramas de secuencia de los demás casos de uso puede examinar el Anexo F.

2.9 Modelo de Datos.

Un modelo de datos es aquel que describe la representación lógica y física de los datos persistentes en el sistema (33). En el desarrollo de la solución se ha hecho necesario garantizar la persistencia de los datos.

Las bases de datos desempeñan un papel crucial en casi todas las áreas de desarrollo de software. Su diseño es una de las tareas más importantes en la construcción de un sistema que requiera manejar la persistencia de la información. El sistema que se propone hace uso de una base de datos, ya que es necesario persistir una cantidad considerable de información relacionada con los usuarios principalmente.

La colección de datos **estructura_organizacional** almacenará los datos correspondientes a las áreas en la que se encuentran agrupados los usuarios y auditores en el sistema. Representa la Estructura de una empresa con los nombres de los departamentos y sus relaciones con los padres.

campo	Significado	Tipo de dato	Ejemplo
_id	Id generado por MongoDB	Id de mongo	ObjectId(“a5dfk0d7fa”)
n	Nombre del departamento	string	UCI
id_e_o	Id numérico asignado.	float	1
id_e_o_p	Id del padre.	float	1

Tabla 5 Descripción de la colección estructura_organizacional.

La colección de datos **usuarios_entidad** almacenará los datos correspondientes a los usuarios que pertenecen al sistema. Contiene el usuario que navega por la red (al que se le realiza un reporte) y el id de la entidad (departamento o área) a la que pertenece. Un usuario puede tener varias entradas en la estructura organizacional del sistema lo que identifica que pertenece a varios departamentos.

campo	Significado	Tipo de dato	Ejemplo
_id	Id generado por MongoDB	Id de mongo	ObjectId(“asdfksdhfa”)
u	Nombre del usuario	string	machavez
id_e_o	Id que hace función de relación con la colección de datos estructura_organizacional identificando que el usuario pertenece a ese departamento.	float	1

Tabla 6 Descripción de la colección usuarios_entidad.

La colección de datos **usuarios_auditores** almacenará los datos correspondientes a aquellos usuarios del sistema con permisos de auditoría sobre las diferentes áreas de la estructura organizacional del sistema. Identifica los usuarios con permisos de auditores y los departamentos a los que tiene permisos de auditor. Un mismo usuario puede tener varias entradas en la estructura lo cual indica que tiene permisos de auditar varios departamentos y los usuarios que pertenecen a esas áreas.

Campo	Significado	Tipo de dato	Ejemplo
_id	Id generado por MongoDB	Id de mongo	ObjectId(“asdfksdhfa”)
u	Nombre del usuario auditor	string	machavez
id_e_o	Id que hace función de relación con la colección de datos estructura_organizacional identificando que el usuario tiene permiso de auditar ese departamento	float	1

Tabla 7 Descripción de la colección usuarios_auditores.

La colección de datos **categories** es la encargada de almacenar los datos de las categorías de dominios existentes en el sistema.

Campo	Significado	Tipo de dato	Ejemplo
_id	Id generado por MongoDB	Id de mongo	ObjectId(“a5dfk0d7fa”)
id1	Id numérico asignado.	float	1
name	Nombre de la categoría	string	ocio

Tabla 8 Descripción de la colección categories.

La colección de datos **domain** es la encargada de almacenar los datos de los dominios existentes en el sistema y a la categoría a las que pertenece.

Campo	Significado	Tipo de dato	Ejemplo
_id	Id generado por MongoDB	Id de mongo	ObjectId(“a5dfk0d7fa”)
id1	Id numérico asignado.	float	1
d	Nombre del dominio	string	facebook.com
cl	Nombre de la categoría a la que pertenece.	string	ocio

Tabla 9 Descripción de la colección domain.

En resumen, el modelo de datos está compuesto por las colecciones: estructura_organizacional, usuarios_entidad, usuarios_auditores, categories y domain, tal como se muestra en el modelo de datos (Ver figura 15).

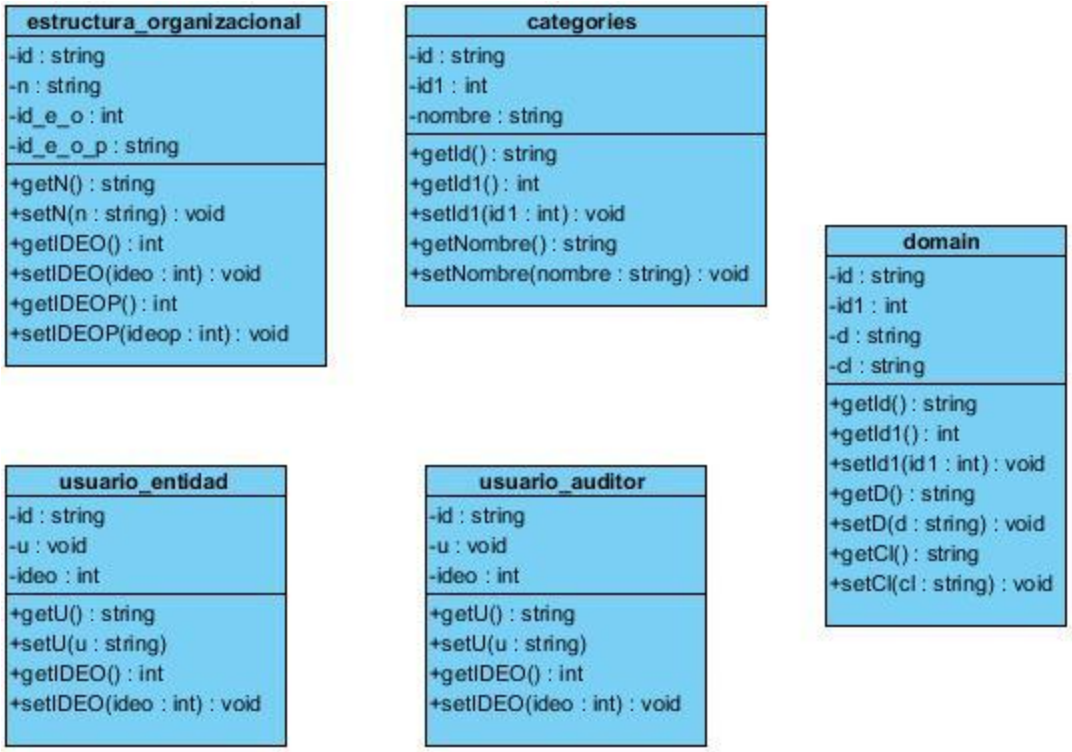


Figura # 15 Modelo físico de datos para MongoDB.

2.10 Modelo de Despliegue.

El diagrama de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos software (33). También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. El mismo está compuesto por:

Nodos: Elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.

Dispositivos: Nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.

Conectores: Expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

Como se puede apreciar en la figura # 16, el módulo desarrollado se integrará a la IAW (*Interface Web Application*) de AiresProxyAudit y será accedido por los usuarios usando un navegador web en una PC cliente, a través de una conexión por HTTPS. El módulo consulta el servidor de BD a través del protocolo TCP y realiza consultas a la base de datos de AiresProxyAudit.

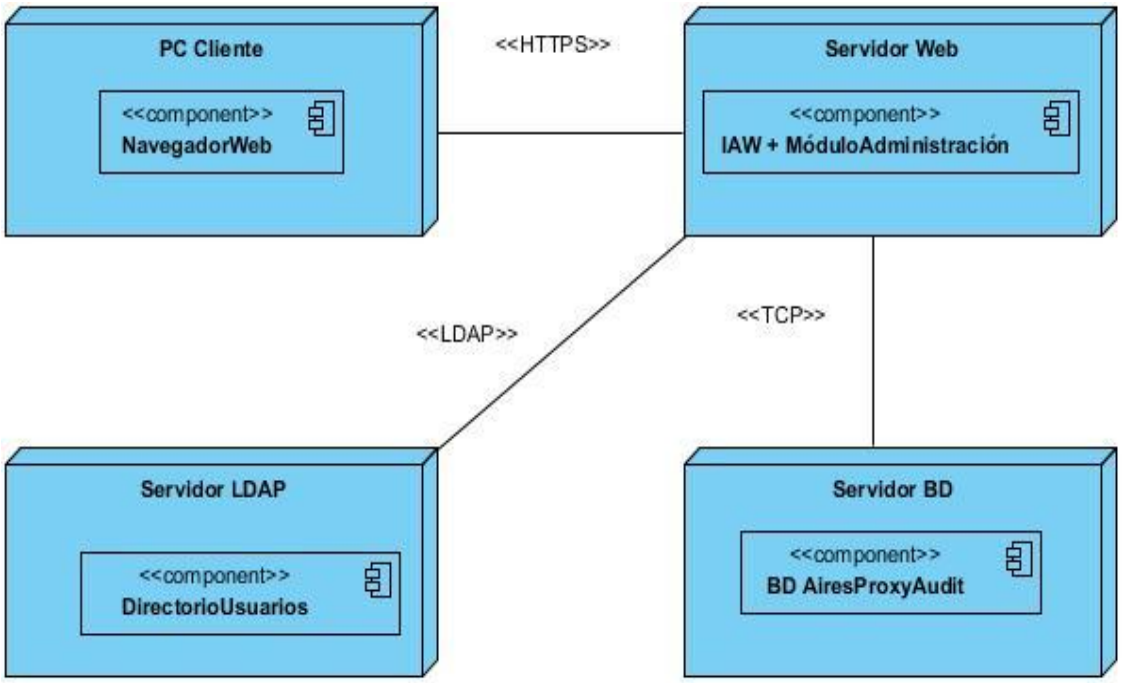


Figura # 16 Diagrama de Despliegue.

2.11 Conclusiones Parciales.

En el capítulo se realizó una descripción de los principales patrones de diseño empleados en el desarrollo del módulo, los cuales proveerán una arquitectura más sólida al sistema. A través de la modelación del análisis y del diseño se han sentado las bases para la implementación. El diagrama de despliegue permite mostrar la distribución de los componentes de software en los nodos físicos, así como los protocolos de comunicación entre dichos nodos. El refinamiento del modelo de datos permitió diseñar la base de datos del módulo.

El análisis del flujo actual de los procesos ha permitido ganar en comprensión sobre el problema presente. Las especificaciones de Casos de Uso del Sistema constituyen una herramienta útil ya que muestra detalladamente las interacciones de los actores y el sistema. Es por ello que las especificaciones y diagramas descritos en el presente capítulo han sentado las bases para comenzar la implementación del módulo propuesto.

Capitulo #3 – Implementación y pruebas.

3.1 Introducción.

En el flujo de implementación se comienza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, *scripts*[15], ficheros de código binario, ejecutables y similares. La implementación es el centro durante las iteraciones de construcción, aunque también se lleva a cabo durante la fase de transición, para tratar otros defectos encontrados en dicha fase. En el flujo de trabajo de prueba se verifica el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias (31). Las pruebas se realizan con el objetivo de validar el sistema desarrollado y detectar posibles fallos.

3.2 Diagrama de Componentes.

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos de software que intervienen en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente(31).

A continuación se muestra el diagrama de componentes del módulo desarrollado:

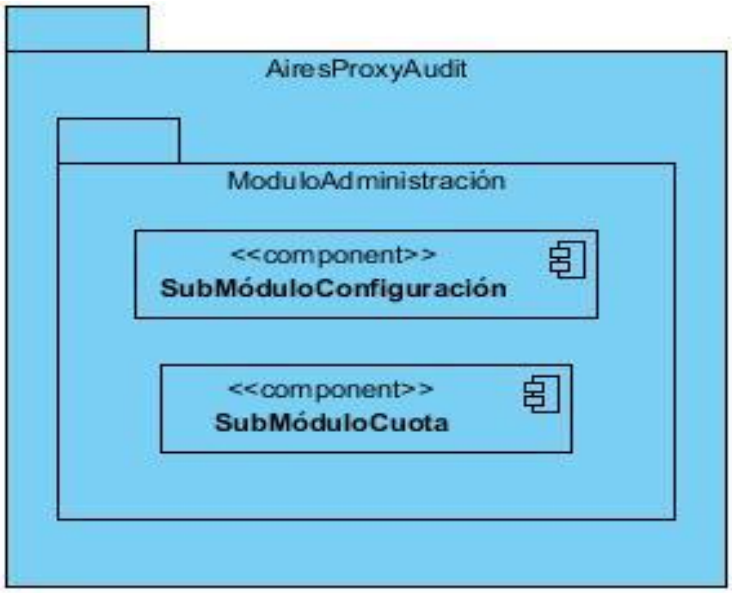


Figura # 17 Diagrama de Componentes del Módulo de Administración.

Para una mayor comprensión sobre los componentes desarrollados, se muestra una vista más detallada de los mismos, separada por sub-módulos:

[15] Ficheros de texto que contiene código ejecutable.

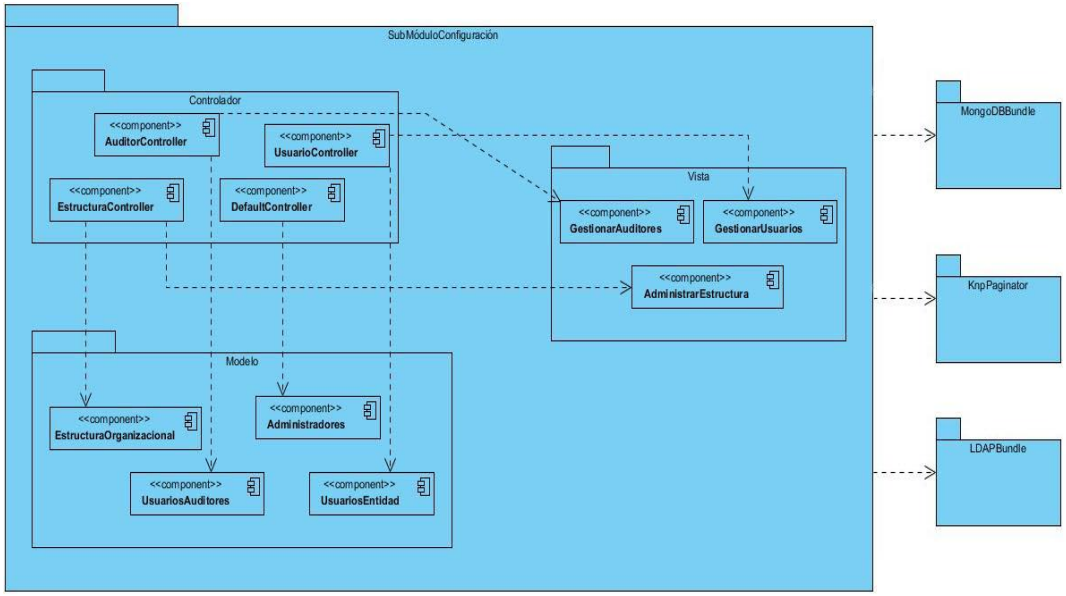


Figura # 18 Diagrama de componentes del sub-módulo configuración.

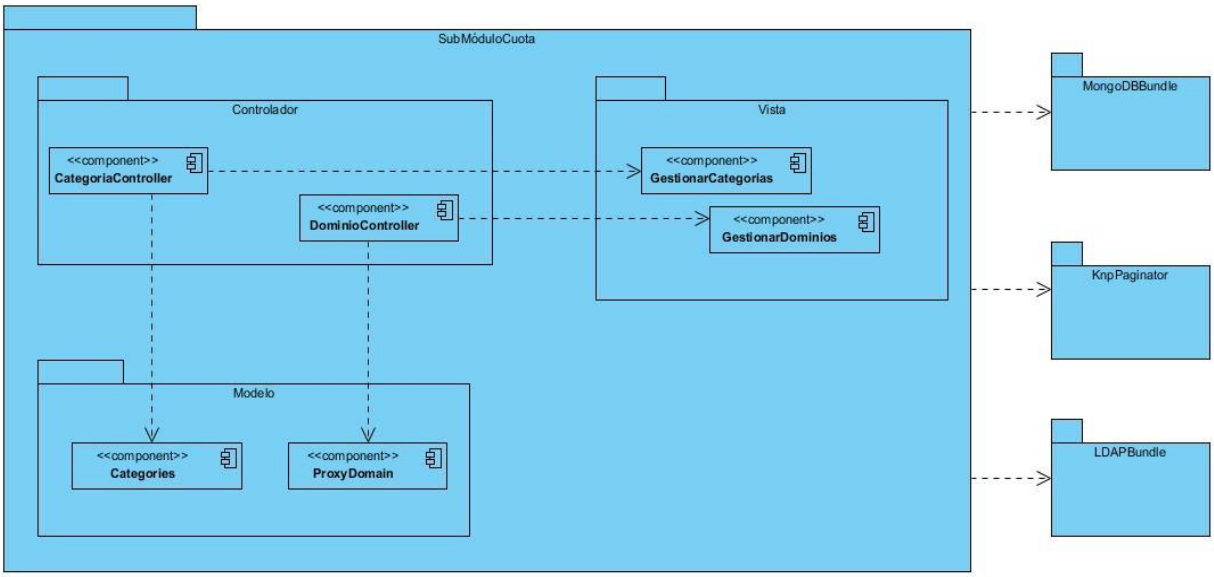


Figura # 19 Diagrama de componentes del sub-módulo Cuota.

3.3 Estándares de codificación.

Un estándar es un modelo, norma, patrón o referencia, la especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad. Para la implementación del módulo de administración se estandarizó el código, lo que permite que exista legibilidad y organización para su mantenimiento. A continuación se especifican los estándares de código a utilizar en la construcción del sistema, los cuales se basan en las normas definidas por los estándares PSR-0, PSR-1 y PSR-2, utilizados para la codificación de aplicaciones que utilizan el *framework* de desarrollo Symfony 2 (40).

El nombre de las clases debe ser declarado utilizando el estilo **StudlyCaps** que permite hacer uso de mayúsculas y minúsculas indiscriminadamente.

```
1 ....class EjEmplO()
2 ....{
3 ....//Bloque de instrucciones
4 ....}
```

El nombre de los procedimientos debe ser declarado utilizando el estilo **camelCase**, que permite definir el nombre de las funciones comenzando con minúsculas la primera palabra y el comienzo de la segunda con mayúsculas.

```
1 ....functionmiMetodo()
2 ....{
3 ....//Bloque de instrucciones
4 ....}
```

En caso de necesitar consultar los demás estándares de codificación puede examinar el Anexo G.

3.4 Interfaces principales de la aplicación.

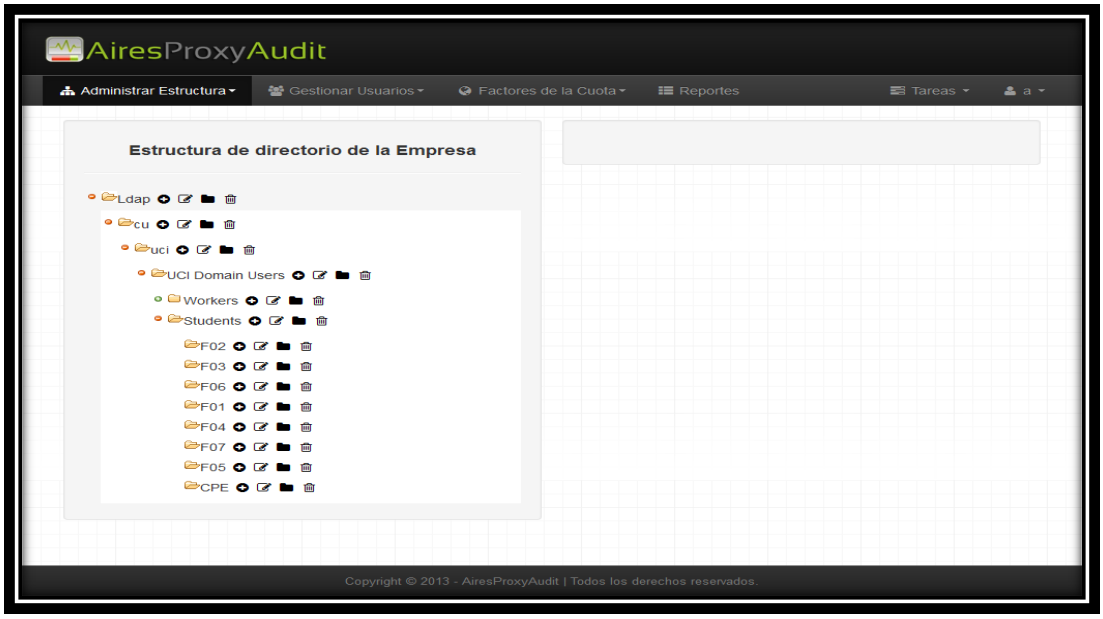


Figura # 20 Interfaz principal del módulo de administración.

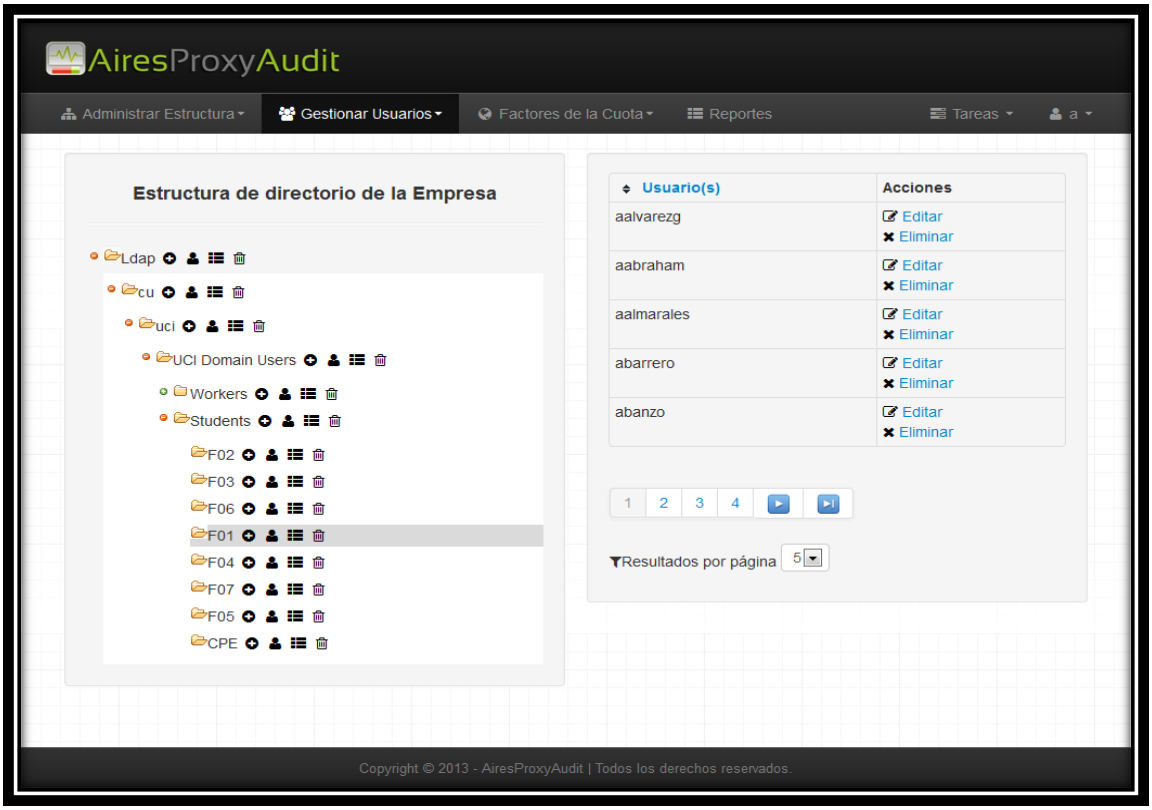


Figura # 21 Interfaz principal del submódulo de configuración.

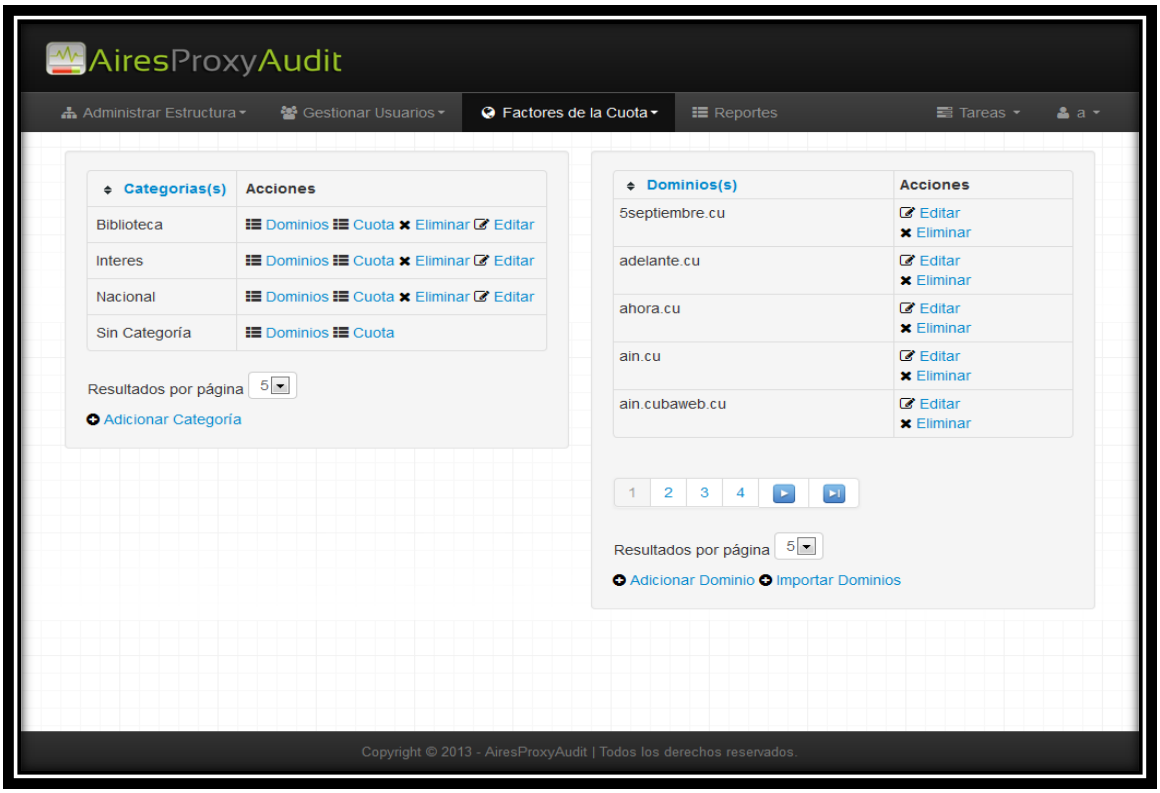


Figura # 22 Interfaz principal del submódulo de cuota.

3.5 Pruebas de software.

Pruebas de Funcionalidad.

Este tipo de pruebas se aplicaron con el objetivo de localizar fallas funcionales en el sistema, al identificar situaciones en las que las respuestas de este a determinadas acciones del usuario no se apegan a las especificaciones establecidas. Se ejecutaron este tipo de prueba para todos los casos de uso con diferentes entradas del usuario, para determinar que los resultados obtenidos fueran consistentes bajo cualquier situación, lo que permitió comprobar el cumplimiento de cada uno de ellos y el cumplimiento de los requisitos establecidos.

A continuación se detallan las pruebas de funcionalidad realizadas para el caso de uso Crear Estructura empleando la técnica de Caja Negra:

Descripción de variables - SC Crear Estructura.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	nombre	Campo de texto	No	Se debe especificar al menos una palabra.

Tabla 10 Descripción de las variables .Prueba de Funcionalidad para el CU1.

SC Crear Estructura.

Escenario	Descripción	Variable 1: nombre	Respuesta del sistema	Flujo central
EC 1.1 Datos correctos	El usuario introduce todos los valores de manera correcta en el sistema.	V	El sistema crea la estructura en la BD y muestra un mensaje informando que se creó correctamente.	El usuario llena todos los campos de manera correcta y le da clic en Adicionar o presiona el botón Enter.
EC 1.2	El usuario deja los	I	Si el navegador es	El usuario deja

Campos Vacíos	campos sin llenar.	vacía	Firefox a partir de la versión 5 el sistema muestra un mensaje que dice: "Por favor, rellene este campo" o “Please fill out this field” en dependencia del idioma que tenga el navegador.	algún campo vacío y le da clic en Adicionar o presiona el botón Enter.

Tabla 11 Prueba de Funcionalidad para el CU1 Crear Estructura.

Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Como resultado de la aplicación de este tipo de pruebas se detectaron un total de 7 no conformidades durante la realización de 3 iteraciones distribuidas de la siguiente forma (ver Figura # 23). Las no conformidades detectadas se agrupan fundamentalmente en tres tipos de errores: campos con entradas no validadas, mensajes del sistema incorrectos, funcionalidades con deficiencias (ver Figura # 24).

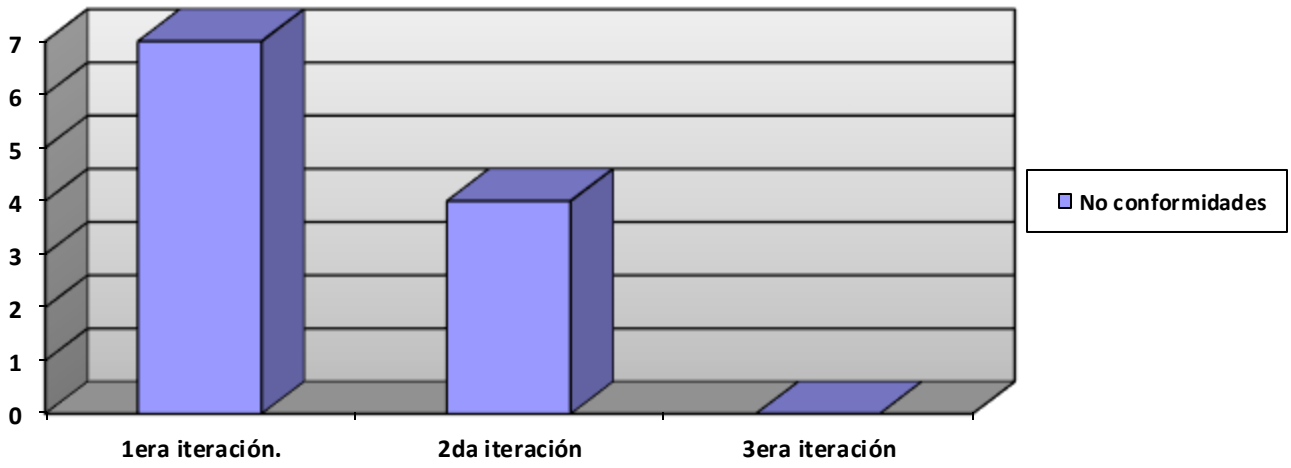


Figura # 23 No conformidades detectadas distribuidas por iteraciones.

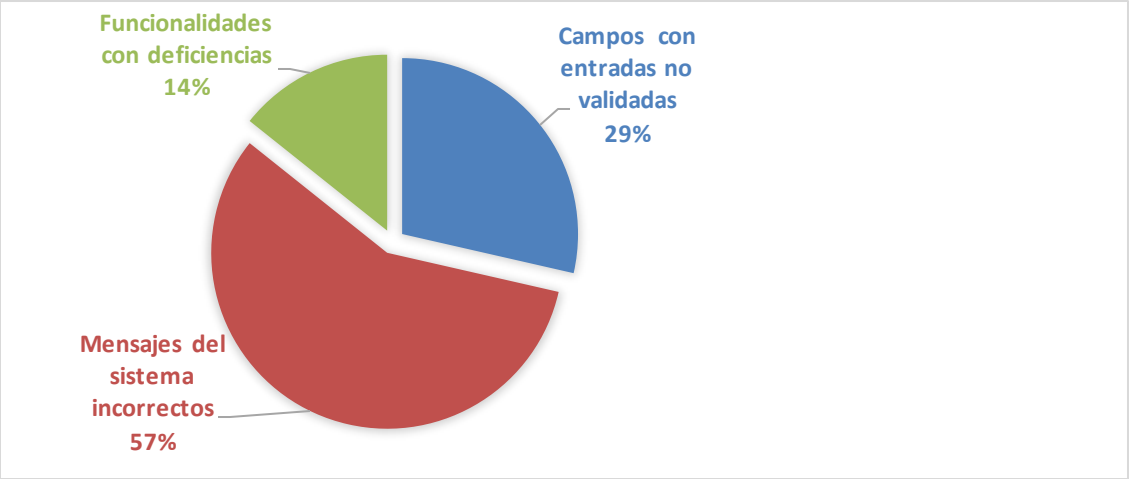


Figura # 24 Tipos de no conformidades detectadas.

Pruebas Unitarias y de Integración.

Mediante la ejecución de las pruebas unitarias a cada uno de los Bundles internos del sistema, fue posible determinar su correcto funcionamiento mediante el estudio del flujo de datos entre los diferentes componentes que los integran. La ejecución de las pruebas de integración permitió verificar la operación conjunta de cada uno de los componentes del sistema, haciendo énfasis en la interacción entre estos, lo que posibilitó la detección de incoherencias en el funcionamiento de la aplicación. También fue posible la detección de deficiencias en el manejo de entidades dependientes por parte de los componentes, ya que se eliminaban referencias y datos que luego eran utilizados por otros elementos del sistema. Luego de resueltas las deficiencias encontradas, el sistema fue sometido nuevamente a un escaneo para garantizar que habían sido resueltas, arrojando como resultado la inexistencia de errores de integración en la aplicación.

Pruebas de Carga y Estrés.

Para comprobar la escalabilidad del *software* desarrollado se hace necesario determinar la carga que puede soportar. Para ello se desarrollaron las pruebas de carga y estrés utilizando la herramienta Jmeter, lo que arrojó datos importantes acerca del estrés que puede soportar el sistema. Se realizó una simulación de carga al sistema con un total de **2060** peticiones enviadas en **20** hilos de ejecución concurrentes, arrojando un **0.0014**% de error al atender dichas peticiones y un tiempo promedio de respuesta de **4.5** segundos. Se debe señalar que la simulación fue realizada en un servidor de bajas prestaciones con las siguientes características: sistema operativo Ubuntu 12.10, microprocesador Core 2 Duo a 2.20 GHz y una memoria RAM de 2GB, considerablemente inferior a lo descrito en los requisitos. Lo

que permitió llegar a la conclusión de que los resultados son satisfactorios para el funcionamiento del módulo en un entorno real de ejecución.

Pruebas de Seguridad.

Con el objetivo de evaluar la seguridad en un primer nivel, el sistema fue sometido al chequeo de sus elementos por una lista con 15 indicadores pertenecientes a cuatro tipos de pruebas de seguridad: pruebas de autorización, pruebas de gestión de sesiones, comprobación del sistema de autenticación y validación de datos. Como resultado de la aplicación de la lista de chequeo fue posible determinar deficiencias en 5 indicadores de impacto crítico en la seguridad. Estos indicadores fueron:

1. El usuario permanecía autenticado un tiempo ilimitado en el sistema.
2. No se encriptaban los campos de usuario y contraseña del formulario de autenticación.
3. No existía una conexión segura entre el cliente y el servidor.
4. No se contaba con un certificado de seguridad que validara la conexión.
5. No se validaban los datos que se almacenaban en la base de datos.

Luego de resueltas las deficiencias encontradas el sistema fue sometido nuevamente a un escaneo de vulnerabilidades para garantizar que habían sido resueltas, arrojando como resultado la inexistencia de errores de seguridad en la aplicación.

3.6 Conclusiones Parciales.

En este capítulo se determinó cuál sería el estándar de codificación a utilizar, decisión basada en los estándares para el desarrollo de aplicaciones utilizando el *framework* Symfony2, lo que permitió lograr una mayor legibilidad y limpieza del código. Además se hizo un desglose de la aplicación en componentes, resultado del proceso de implementación, para una mayor comprensión de cómo está integrada la aplicación y sus dependencias. La realización de las pruebas funcionales, unitarias, de integración, de seguridad y de rendimiento permitieron la detección de errores y problemas para su posterior solución, además permitieron monitorear el funcionamiento de la aplicación en diferentes entornos de ejecución y carga. Como resultado de la implementación se ha obtenido un sistema funcional, completamente operativo. La interfaz web del módulo cumple con los requisitos definidos y brinda la posibilidad de gestionar las funcionalidades con facilidad. La realización de pruebas al *software* implementado ha permitido validar la solución desarrollada y comprobar que cumple con los requerimientos definidos.

Conclusiones Generales

El estudio del estado del arte permitió desarrollar una aplicación basada en las buenas prácticas empleadas por los sistemas estudiados y enfocada hacia la solución del problema identificado, así como la selección de las herramientas y tecnologías adecuadas para su desarrollo. Como resultado del trabajo realizado se logró el diseño e implementación de un módulo de administración para la gestión de los usuarios, grupos de usuarios de navegación y categorías de dominios para el sistema AiresProxyAudit. La aplicación de pruebas permitió validar el correcto funcionamiento de la aplicación desarrollada simulando ambientes reales en diferentes entornos de ejecución.

Por todo lo expuesto anteriormente se concluye que los objetivos propuestos para el presente trabajo han sido cumplidos satisfactoriamente. La aplicación desarrollada contribuirá de manera significativa a la administración de los usuarios, grupos de usuarios de navegación y categorías de dominios en el sistema AiresProxyAudit.

Recomendaciones.

Los resultados obtenidos luego del desarrollo del presente trabajo satisfacen los requerimientos definidos, no obstante para el desarrollo de futuras versiones del módulo se recomienda:

- Incluir soporte para importar las áreas y los usuarios a partir de un fichero LDIF.
- Valorar la posibilidad de poder programar desde la IAW del sistema el horario de ejecución de la tarea de sincronización.

Referencias Bibliográficas.

1. **CUBANIC.** Cubanic - Portal Cuba.cu. [En línea] UEB Servicios Web CITMATEL, 1997. http://www.nic.cu/estadisticas.php.

2. **Menéndez, Ramiro Valdés.** *RESOLUCION No. 127 /2007.* La Habana : s.n., 2007.

3. **EcuRed.** Servidor Proxy. [En línea] 14 de diciembre de 2010. http://www.ecured.cu/index.php/Servidor_proxy.

4. **Motive Ltd.** Motive Glossary. [En línea] 2000. http://www.motive.co.nz/glossary/weblogs.php.

5. **Free Software Foundation.** Sisitema Operativo GNU. *Oficial Site.* [En línea] Free Software Foundation, Inc., 1996. http://www.gnu.org/philosophy/free-sw.es.html.

6. *REVISTA FACULTAD DE INGENIERÍAS USBMed.* **UNIVERSIDAD DE SAN BUENAVENTURA MEDELLÍN.** 1, Antioquía : s.n., junio de 2012, Vol. III.

7. **Oracle.** Java. *¿Qué es un servidor proxy? ¿Cómo puedo conseguir información sobre el servidor proxy? .* [En línea] http://www.java.com/es/download/help/proxy_server.xml.

8. **Gutiérrez, Javier J.** Departamento de Lenguajes y Sistemas Informáticos. *Departamento de Lenguajes y Sistemas Informáticos.* [En línea] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

9. **Gross, Manuel.** Pensamiento Imaginactivo. *Pensamiento Imaginactivo.* [En línea] 23 de 5 de 2010. http://manuelgross.bligoo.com/content/view/801220/Conceptos-sobre-la-Estructura-Organizacional.html.

10. **Definicion .de.** ***Definicion .de.*** [En línea] http://definicion.de/administracion/.

11. Sawmill. Oficial Site. [En línea] Flowerfire, 1997. http://www.sawmill.net.

12. OpenLDAP. Oficial Site. [En línea] OpenLDAP Foundation, 2010. http://www.openldap.org/.

13. David Morón Ruano, Antoni Barba Martí. *Entorno de gestión de políticas basado en un directorio LDAP.* Cataluña : s.n., 2008.

14. Strassner, J. *"Policy Core LDAP Schema".* 2006.

15. Reyes, A. *"Policy Core Extensions LDAP Schema.".* 2007.

16. Pupo Rodríguez, Eddy Yusiel y Rodríguez Fernández, Maidelys. *Módulo para la importación y sincronización de usuarios de un LDAP en Smart Keeper.* La Habana : s.n., 2012.

17. Webmin. Oficial Site. [En línea] 2006. http://www.webmin.com.

18. Smart Keeper. *Manual de Usuario de Smart Keeper.* La Habana, La Habana, Cuba : s.n., Julio de 2012.

19. G. Figeroa, Robert, J. Solis, Camilo y A. Cabrera, Armando. *Metodologías tradicionales vs. metodologías ágiles.* 2008.

20. Eclipse. OpenUP. ***Oficial Site.*** [En línea] Eclipse, 2008. http://epf.eclipse.org/wikis/openup.

21. Potencier, Fabien. Symfony. ***Official Site.*** [En línea] www.symfony.com.

22. Potencier, Fabien. Symfony. ***Oficial Site.*** [En línea] http://symfony.com/.

23. GitHub. GitHub. ***Official Site.*** [En línea] https://github.com/jzaefferer/jquery-treeview.git.

24. Marley, Jimi. Programacion. [En línea] 2011. http://www.programacion.com/articulo/%20por_que_elegir_php_143.

25. EcuRed. EcuRed. ***Sistema Gestor de Base de Datos.*** [En línea] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

26. MongoDB. Oficial Site. [En línea] http://www.mongodb.org/.

27. O'Reilly, Media. Subversion. [En línea] 2011. http://svnbook.red-bean.com/en/1.2/svn-book.pdf.

28. RapidSVN. Oficial Site. [En línea] http://www.rapidsvn.org.

29. Software in the Public Interest, Inc. Debian. [En línea] 1997. http://www.debian.org/.

30. ORACLE. NetBeans. [En línea] http://netbeans.org.

31. The Apache Software Foundation. Apache. [En línea] http://httpd.apache.org/.

32. Visual Paradigm. *Oficial Site.* [En línea] http://www.visual-paradigm.com/product/vpuml.

33. Jacaboson, I., Booch, G., Rumbaugh J. *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000.

34. S. Pressman, Roger. *Ingeniería del Software, un enfoque práctico.* McGraw-Hill : s.n., 2001.

35. Alexander, C., Ishikawa, S. y Silverstein, M. *A Pattern Language.* Oxford University : s.n., 1997.

36. Fowler, M. *Web Server Patterns. Model View Controller, Patterns of Enterprise Application Architecture.* 2003.

37. Potencier, Fabian y Eguiluz, Javier. *Desarrollo web ágil con Symfony 2.* 2011.

38. Eguiluz, Javier y Potencier, Fabian. *Desarrollo web ágil con Symfony 2.* 2011.

39. Popkin Software and Systems. Modelado de Sistemas con UML. [En línea] 2008. http://es.scribd.com/Samirak/d/1838816-docmodeladosistemasuml.

40. Godoy Jiménez, José Manuel. Diseño de proyectos de software en código abierto. [En línea] 2002. http://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-dise%F1o-software/doc-dise%F1o-software-parte-1.pdf.

41. Salinas Caro, Patricio. Tutorial de UML. [En línea] http://www.dcc.uchile.cl/~psalinas/uml/.

42. Potencier, Fabian y Eguiluz, Javier. *Desarrollo web ágil con symfony 2.* 2012. 1.

Bibliografía

Bibliografía.

1. **Chodorow, Kristina; Dirolf, Michael.** MongoDB: The Definitive Guide. O'Reilly Media, Sept 2010. **ISBN: 978-1-449-38156-1**.
2. **Javier Eguiluz**, Desarrollo web ágil con Symfony2, 15 de octubre del 2012. Publicaciones **easybook** versión **4.8-DEV**.
3. **(UCI)Dpto. Soluciones Informáticas para Internet**, **Centro de Ideoinformática**. Trabajo con MongoDB desde Symfony2, marzo 2012.
4. **Roger S. Pressman,** Ingeniería de software 5 edición Un enfoque práctico.
5. **Potencier Fabien, Zanionotto Francois**. Symfony la Guía definitiva.
6. **Hernández León, Rolando Alfredo, Coello González Sayda.** El proceso de la investigación científica, 2011.
7. **INTECO, Laboratorio Nacional de Calidad de Software.** Ingeniería del Sotware: Metodologías y Ciclos de Vida. España: s.n., 2009.
8. **Ideoinformática, Centro.** Configuración de la metodología OpenUP. Ciudad de La Habana. Cuba: s.n., 2012.
9. **Jacobson, Ivar.** El proceso unificado de desarrollo de software. Madrid: Addision Wesley, 2000.
10. **Peña, Ing. Yadira Machado**. Planificación, Diseño y Ejecución de Pruebas Funcionales. Habana:
11. **Calisoft**, marzo-2011.
12. **B, Ing. Alexander Oré. 2008.** Calidady Software.com. *Calidady Software.com.* [En línea] Nazcasoft.com, 2008. [Citado el: 2 de 4 de 2013.] http://www.calidadysoftware.com/testing.php.
13. **PHP**, 2013. [Disponible en: http://www.php.net/manual/es/intro-whatis.php.]

Glosario de Términos.

Framework o Marco de Trabajo: Una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación, se puede considerar como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta.

Symfony: Framework de código abierto para el desarrollo de aplicaciones web en PHP.

HTML (HyperText Markup Language): Lenguaje de marcado de hipertexto compuesto de una serie de etiquetas o marcas que permiten definir el contenido y la apariencia de las páginas Web.

HTTPS (Hypertext Transfer Protocol Secure): Protocolo Seguro de Transferencia de Hipertexto. Garantiza la seguridad de las comunicaciones entre el usuario y el servidor web al que este se conecta.

IAW : Interfaz de Administración Web.

CGI (Common Gateway Interface): Interfaz de Entrada Común.

IDE (Integrated Development Environment): Entorno de Desarrollo Integrado. Herramienta que se usa para facilitar el desarrollo de software.

Plugin: Módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

TCP/IP (Transfer Control Protocol / Internet Protocol): Protocolo de Control de Transmisión/ Protocolo de Internet: Protocolo de red en los que se basa Internet y que permiten la transmisión de datos entre computadoras.

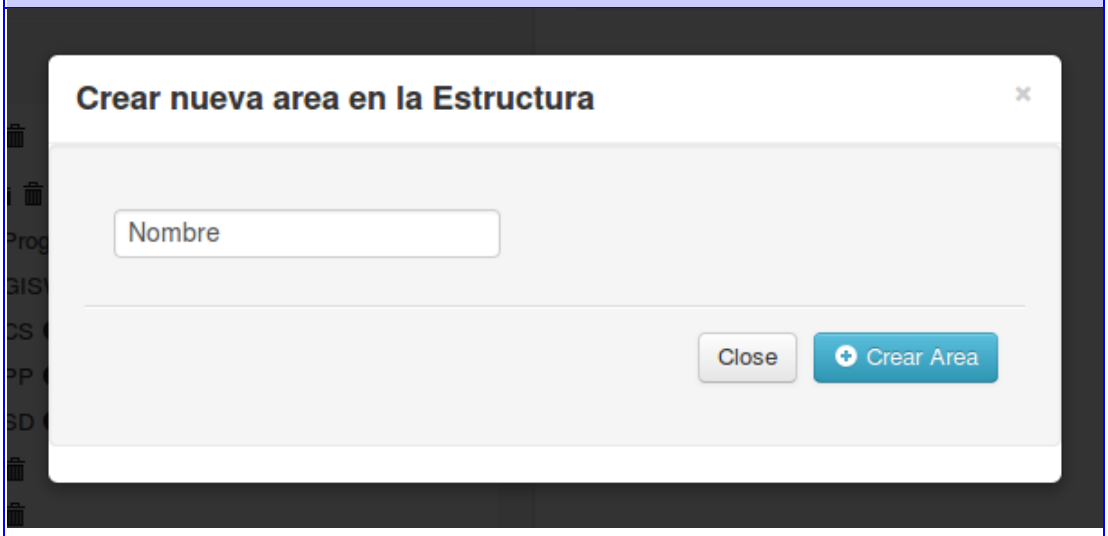
TCP: Protocolo de Control de Transferencia.

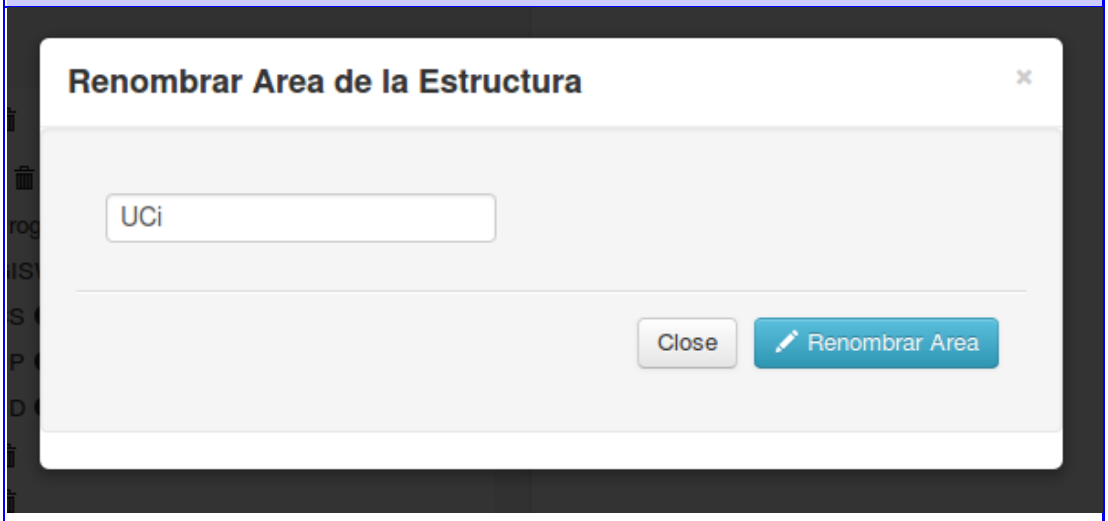
UML: Unified Modeling Language: Lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

LDIF (LDAP Interchange Format): Formato de Intercambio de LDAP. Archivo utilizado para importar y exportar información entre servidores de directorio basados en LDAP.

Anexos.

ANEXO A
Descripción de los Requisitos Funcionales del Sistema.

<table>
<tr><th>Nº</th><th>Nombre</th><th>Descripción</th><th>Complejidad</th><th>Prioridad para cliente</th></tr>
<tr><td rowspan="6">RF1</td><td>Adicionar entidad.</td><td>El sistema debe permitir crear y guardar (a través de campo de entrada o a través de fichero yml) en la base de datos MongoDB las entidades que conforman la estructura organizacional (en forma de árbol) de una empresa.</td><td>Alta</td><td>Alta</td></tr>
<tr><td colspan="4">Prototipo</td></tr>
<tr><td colspan="4"> 
</td></tr>
<tr><td>Campos</td><td>Tipos de Datos</td><td colspan="2">Reglas o Restricciones</td></tr>
<tr><td>NombreEntidad</td><td>String</td><td colspan="2">Campo obligatorio.</td></tr>
<tr><td>Observaciones</td><td colspan="3"></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF2	Modificar entidad.	El sistema debe permitir modificar la entidad seleccionada (Nombre de las entidades, Estructura lógica).	Alta	Alta
	Prototipo			
				
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	NombreEntidad	String	Campo obligatorio.	
	Observaciones			

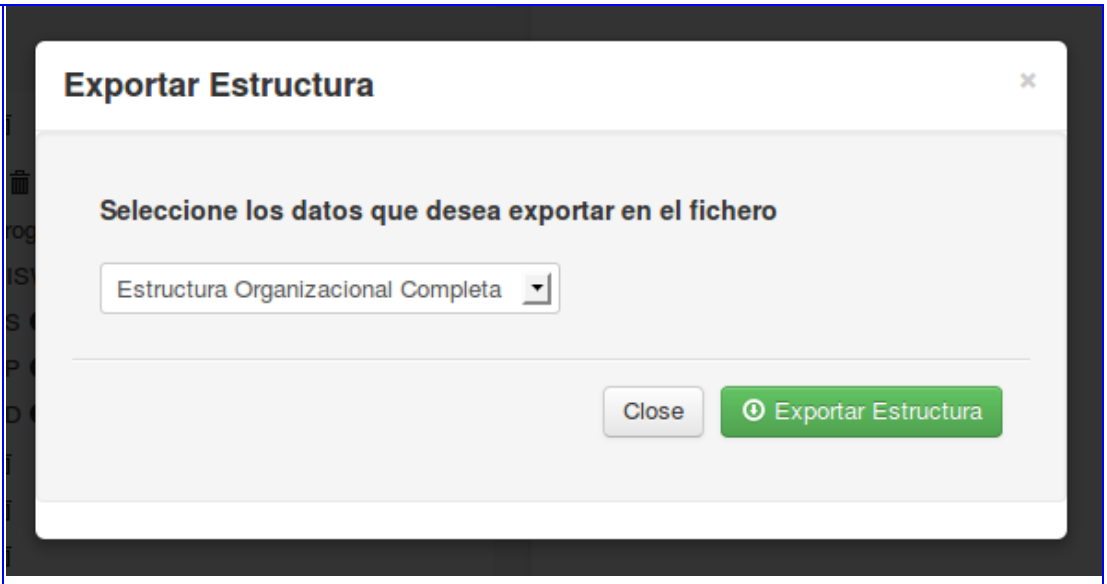
Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF3	Listar entidades.	El sistema debe permitir mostrar las entidades y su información (nombre, usuarios y auditores que contiene).	Alta	Alta
	Prototipo			

Campos	Tipos de Datos	Reglas o Restricciones	
Observaciones			

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF4	Eliminar entidad.	El sistema debe permitir eliminar una entidad de la estructura creada. Cuando se elimina la entidad son eliminadas todas las	Alta	Alta

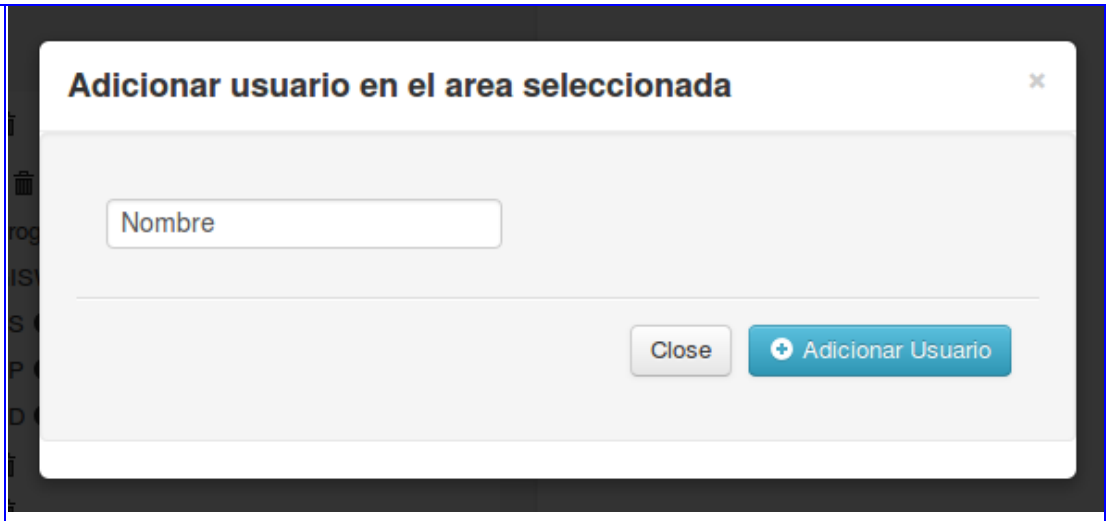
		referencias a los usuarios y auditores que pertenecen o tienen permisos para auditar dicha entidad. En caso de que el usuario pertenezca únicamente a esa entidad el mismo es eliminado del sistema.		
	Prototipo			
				
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones			

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF5	Exportar estructura organizacional.	El sistema debe permitir exportar las entidades y los usuarios que conforman la estructura organizacional de la empresa.	Alta	Alta
	Prototipo			



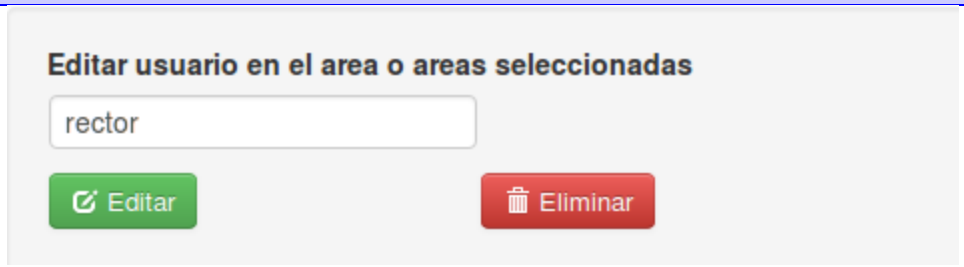
Campos	Tipos de Datos	Reglas o Restricciones	
Observaciones			

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF6	Adicionar usuario a entidad.	El sistema debe permitir adicionar un usuario a una entidad mediante un campo de entrada o partir de un fichero cargado.	Alta	Alta
	Prototipo			



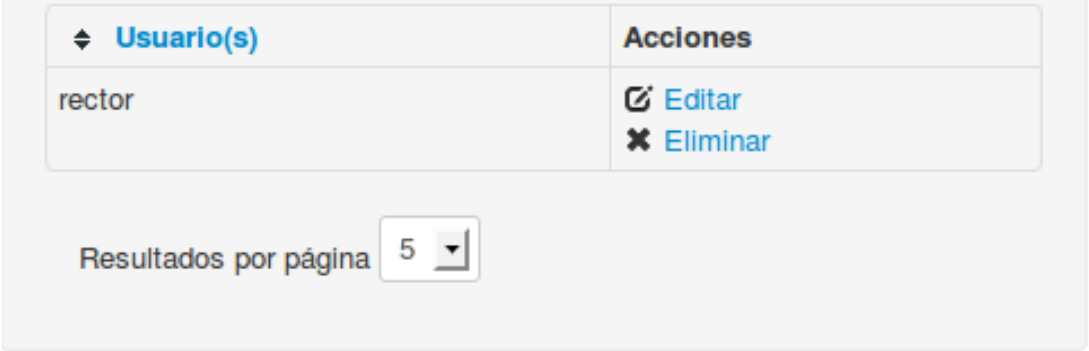
Campos	Tipos de Datos	Reglas o Restricciones
file	string	
Nombre de Usuario	String	
Observaciones		

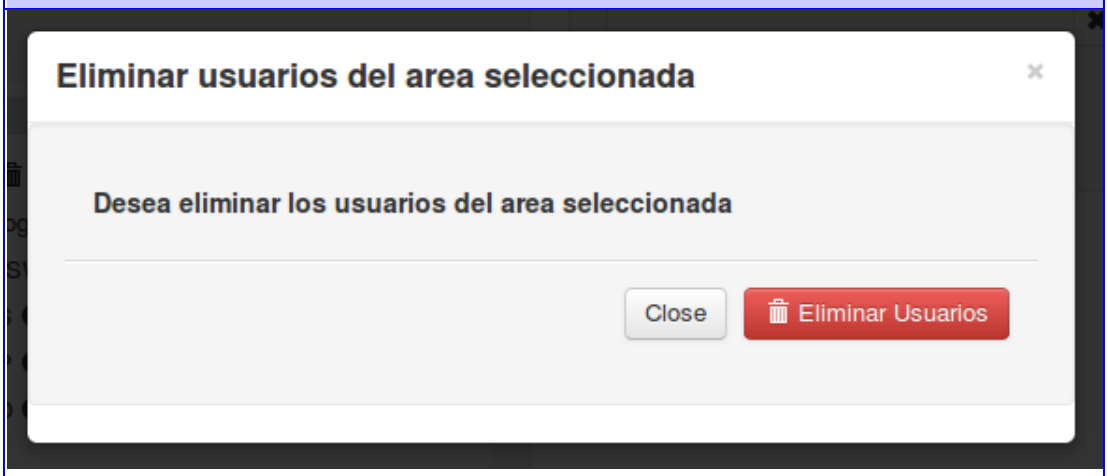
Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF7	Modificar usuario.	El sistema debe permitir modificar el usuario y las entidades a las que pertenece el mismo.	Alta	Alta
	Prototipo			



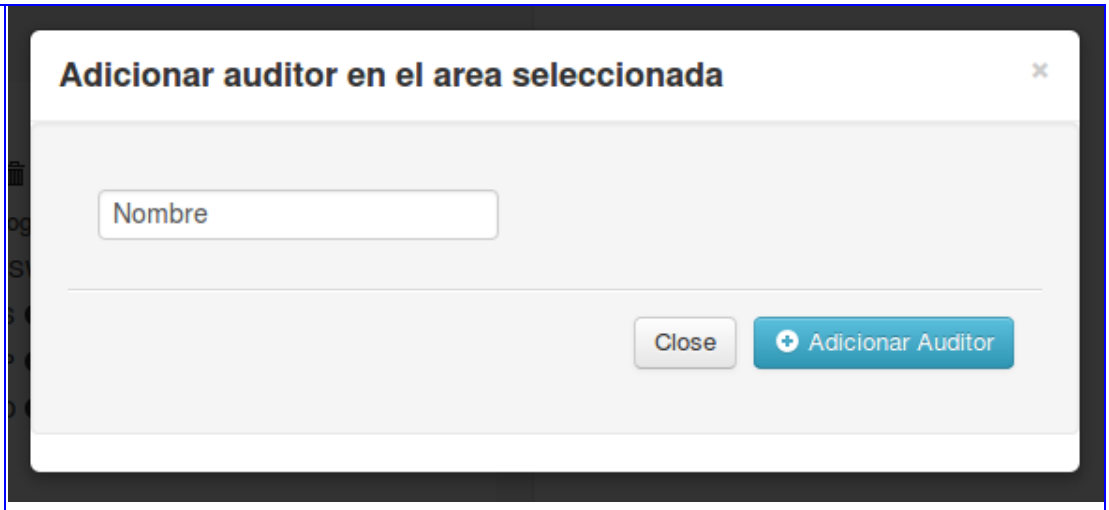
Campos	Tipos de Datos	Reglas o Restricciones
Nombre de Usuario	String	

	Observaciones			

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF8	Mostrar usuarios.	El sistema debe permitir mostrar todos los usuarios que están asociados a una entidad determinada, también debe permitir mostrar todas las entidades a las que está asociado un usuario determinado.	Alta	Alta
	Prototipo			
				
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones			

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF9	Eliminar un usuario.	El sistema debe permitir eliminar un usuario.	Alta	Alta
	Prototipo			
				
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones			

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF10	Adicionar auditor a una entidad.	El sistema debe permitir adicionar un auditor a una entidad determinada.	Alta	Alta
	Prototipo			



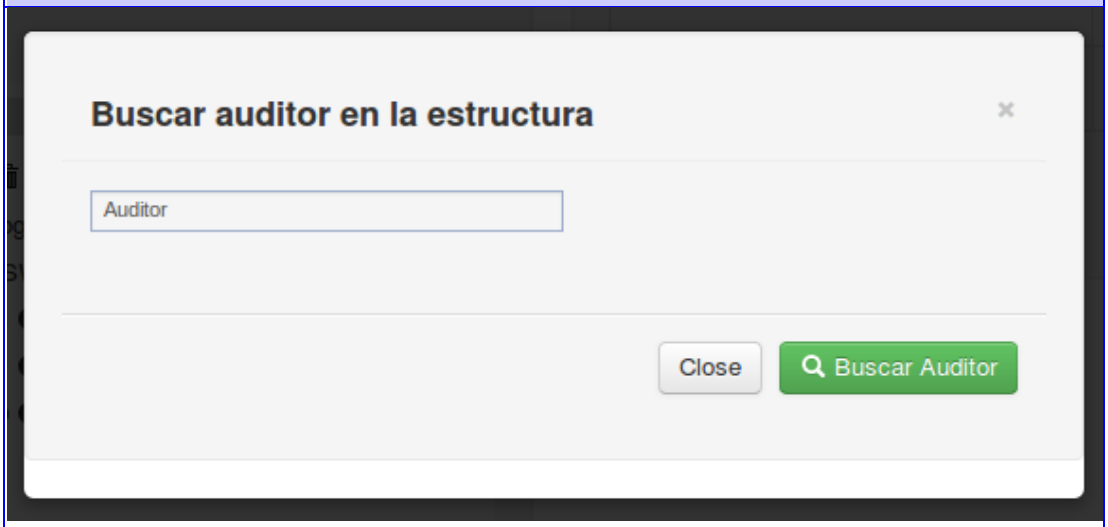
Campos	Tipos de Datos	Reglas o Restricciones
Nombre_Auditor	String	Campo obligatorio.
Observaciones		

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF11	Modificar auditores.	El sistema debe permitir modificar el auditor y los permisos de auditor sobre una entidad determinada.	Alta	Alta
	Prototipo			

Campos	Tipos de Datos	Reglas o Restricciones
Nombre_Auditor	String	Campo obligatorio.

	Observaciones			

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF12	Buscar auditor.	El sistema debe permitir generar un reporte de las entidades a las que un usuario determinado puede auditar, mostrándolo en forma de árbol.	Alta	Alta
	Prototipo			



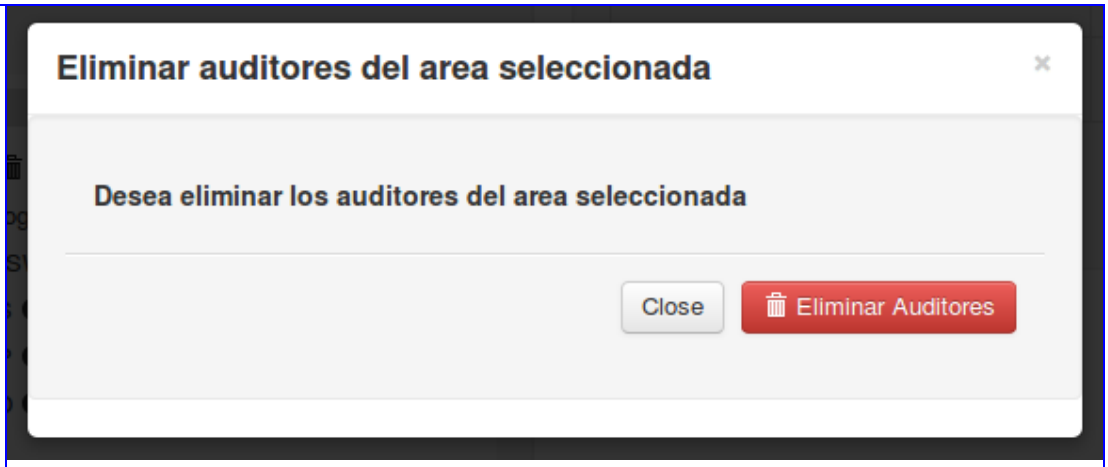
Campos	**Tipos de Datos**	**Reglas o Restricciones**
Observaciones		

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**

RF13	Listar auditores.	El sistema debe permitir verificar los usuarios que tienen permisos de auditar alguna entidad.	Alta	Alta
	Prototipo			

Campos	Tipos de Datos	Reglas o Restricciones
Observaciones		

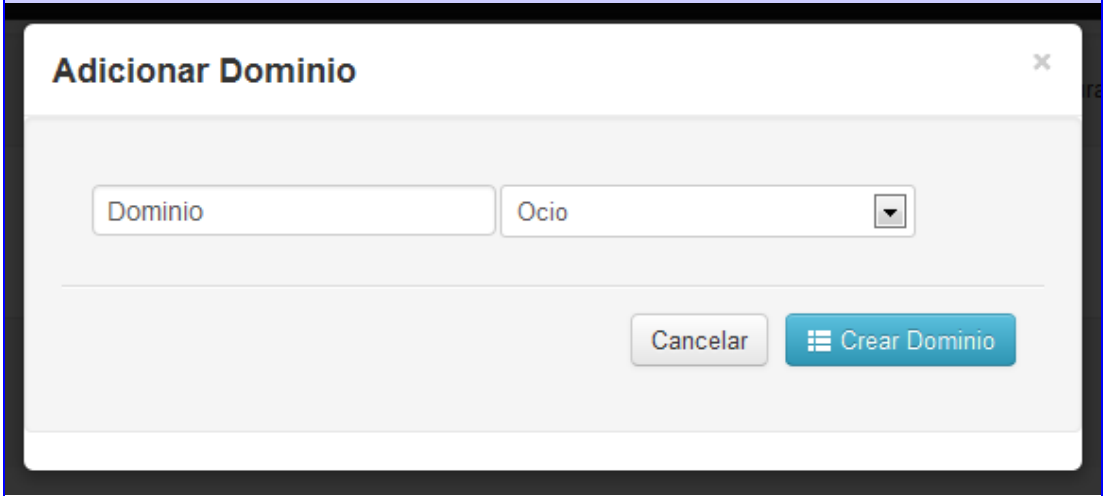
Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF14	Eliminar auditor.	El sistema debe permitir eliminar un auditor determinado de la entidad seleccionada.	Alta	Alta
	Prototipo			



Campos	Tipos de Datos	Reglas o Restricciones
Observaciones		

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF15	Adicionar dominio.	El sistema debe permitir adicionar un dominio a la categoría seleccionada.	Alta	Alta

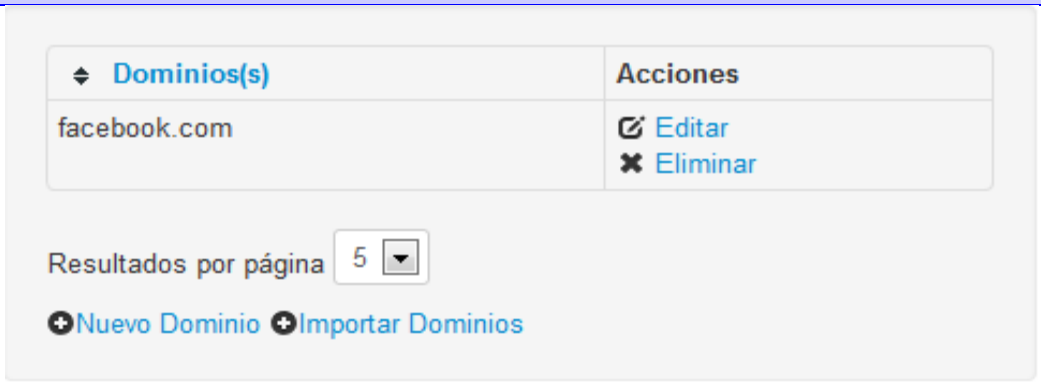
Prototipo



Campos	Tipos de Datos	Reglas o Restricciones

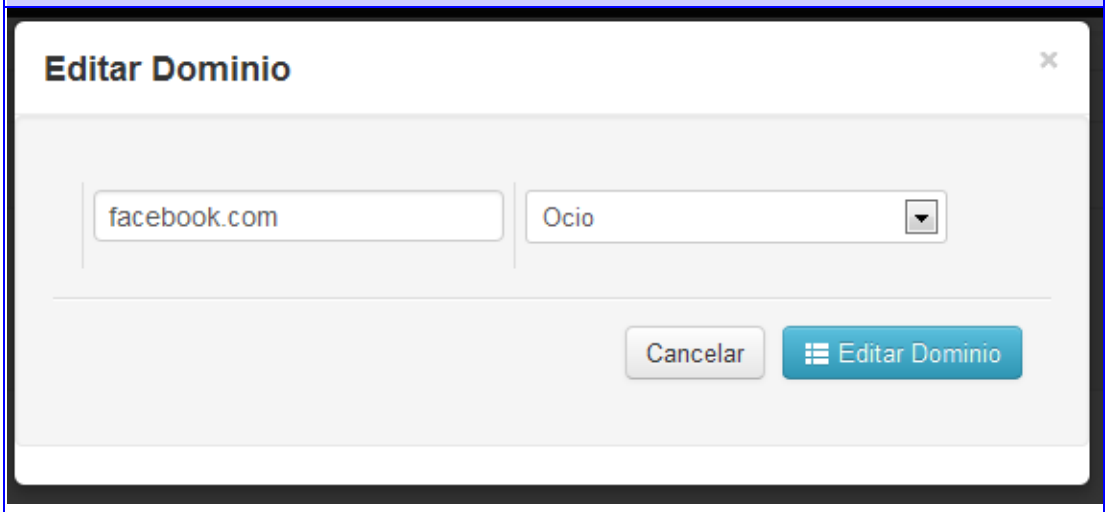
	dominio	String	Campo obligatorio	
	Observaciones			

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF16	Listar dominio.	El sistema debe mostrar los dominios existentes con su categoría, así como todos los dominios pertenecientes a una categoría dada.	Alta	Alta
	Prototipo			

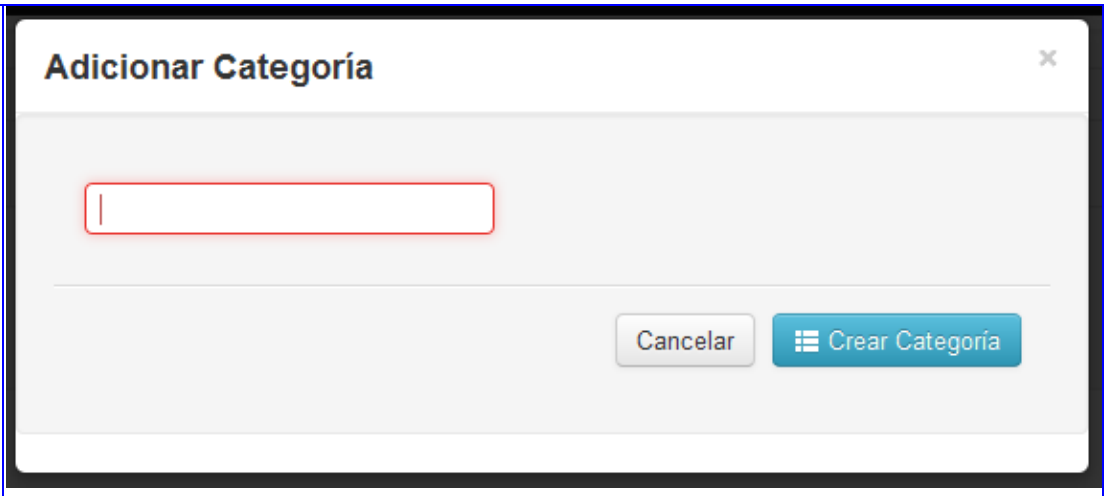


	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones			

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF17	Modificar dominio.	El sistema debe permitir modificar el nombre y la categoría perteneciente al dominio seleccionado.	Alta	Alta

	Prototipo			
				
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	dominio	String	Campo obligatorio	
	Observaciones			

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF18	Insertar categoría.	El sistema debe permitir insertar una categoría. Para insertar una categoría es necesario asociarla a un dominio. Este dominio puede existir en la base de datos o ser uno nuevo. En el caso de que el dominio ya exista en la base de datos el proceso realiza el cambio de categoría.	Alta	Alta
	Prototipo			

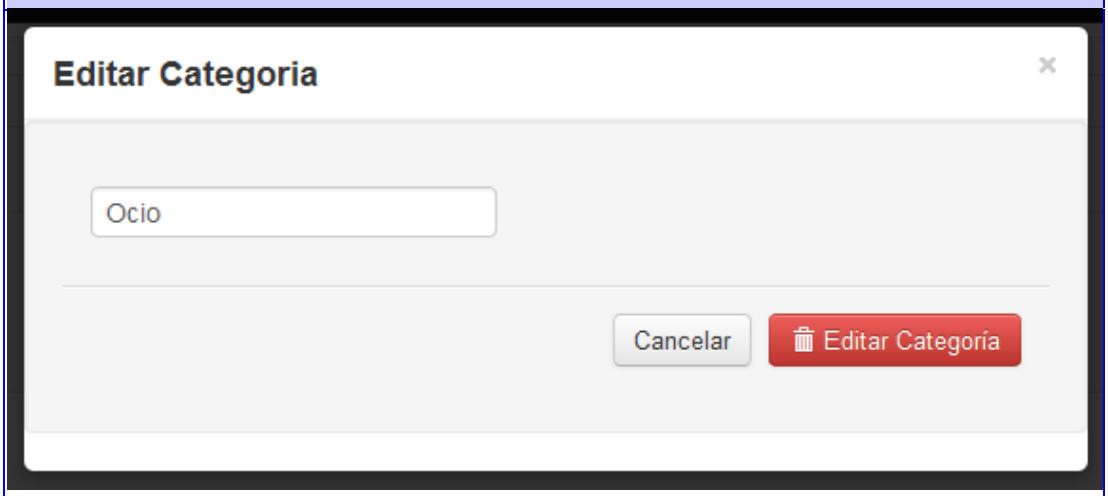


Campos	Tipos de Datos	Reglas o Restricciones
categoría	string	Campo obligatorio
dominio	String	Campo obligatorio
Observaciones		

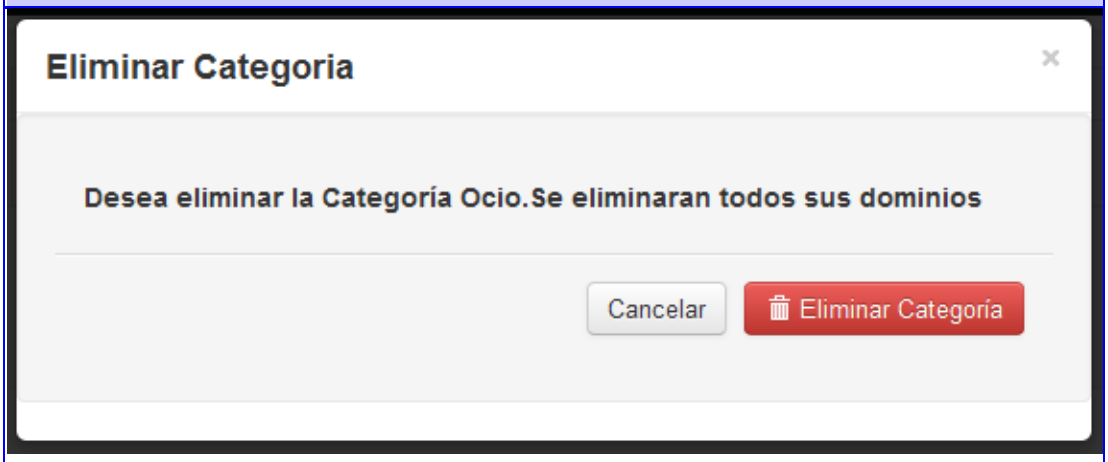
Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF19	Listar categoría.	El sistema debe mostrar todas las categorías existentes.	Alta	Alta
	Prototipo			

Campos	Tipos de Datos	Reglas o Restricciones

	Observaciones			

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF20	Modificar categoría.	El sistema debe permitir modificar el nombre de una categoría.	Alta	Alta
	Prototipo			
				
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	categoría	String	Campo obligatorio	
	Observaciones			

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF21	Eliminar categoría.	El sistema debe permitir eliminar una determinada categoría. Los dominios asociados a esta categoría deberán asociarse a la categoría “Sin Categoría”. La categoría denominada “Sin	Alta	Alta

		Categoría" no puede ser eliminada.		
	Prototipo			
				
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones			

ANEXO B.

Descripción de los Requisitos no funcionales del sistema.

Tipo	Nº	Nombre	Descripción
Usabilidad	RNF #1	Tipo de usuario final.	La aplicación podrá ser utilizada solo por los usuarios que estén registrados en el sistema.
	RNF #2	Tipo de Aplicación Informática.	El sistema de AiresProxyAudit será una aplicación Web.
	RNF #3	Finalidad	El objetivo que persigue el sistema es permitir a usuarios con permisos especiales la generación de reportes estadísticos asociados a la navegación de grupos de usuarios.
	RNF #4	Ambiente	Para el módulo Administración se necesitará: • Hardware: es recomendado 1GB de memoria RAM. Procesador familia INTEL u

			otro con una velocidad mínima de 2.10GHz. Espacio en disco duro de 400 Megabyte. • Software: Sistema Operativo: Windows XP Profesional o superior y Linux.
Tipo	**Nº**	**Nombre**	**Descripción**
Confiabilidad	RNF #5	Confiabilidad del Sistema AiresProxyAudit.	• El sistema debe identificar los privilegios de acceso de los usuarios que deseen gestionar la información. • La información manejada por el sistema es objeto de cuidadosa protección contra la corrupción y acciones inconsistentes. .
Tipo	**Nº**	**Nombre**	**Descripción**
Eficiencia	RNF #6	Eficiencia del Sistema AiresProxyAudit.	• Tiempo de respuesta por transacción: el sistema debe ser capaz de responder con rapidez a las peticiones de los usuarios. • Capacidad: el sistema debe permitir trabajar de manera concurrente a 50 usuarios.
Tipo	**Nº**	**Nombre**	**Descripción**
Soporte	RNF #7	Soporte del sistema AiresProxyAudit.	• Las opciones de instalación y mantenimiento del sistema deben ser entendibles por los usuarios finales. Los instaladores incluirán todas las librerías y *plugins* que necesite el sistema para funcionar correctamente. • Las paradas de cambios o mantenimiento no deberán interferir en el correcto desempeño del resto del sistema. • El sistema estará bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.
Tipo	**Nº**	**Nombre**	**Descripción**
Restricciones de Diseño.	RNF #8	Restricciones de diseño del Sistema	• Lenguajes de programación: el lenguaje definido para el desarrollo del sistema es PHP. El procesamiento de datos y la

		AiresProxyAudit.	obtención de resultados se implementarán en C++. • Sistema Operativo: el sistema se desarrollará sobre el Sistema Operativo Linux. El producto final podrá ser instalado en Sistemas Operativos Linux, optimizado para Centos 6. • Control de versiones: como sistema de control de versiones se utiliza el Subversion (SVN). • Herramienta de modelado: Se utiliza Visual Paradigm para UML 5.0 en el diseño de diagramas y artefactos. Sistema Gestor de Base de Datos: MongoDB.
Tipo	**Nº**	**Nombre**	**Descripción**
Requisitos para la documentación de usuarios en línea y ayuda del sistema.	RNF #9	Ayuda de usuarios del Sistema AiresProxyAudit.	El sistema deberá contar con ayuda de usuario.
Tipo	**Nº**	**Nombre**	**Descripción**
Interfaz	RNF #10	Interfaces de usuario del Sistema AiresProxyAudit.	La interfaz gráfica de la aplicación debe concebirse con un ambiente sencillo y de navegación fácil para el usuario.
	RNF #11	Interfaces Software del Sistema AiresProxyAudit.	La interfaz de software de la aplicación debe concebirse de una manera sencilla y amigable.
	RNF #12	Interfaces de Comunicación del Sistema AiresProxyAudit.	La interfaz de comunicación de la aplicación debe ser segura y confiable.
Tipo	**Nº**	**Nombre**	**Descripción**
Requisitos de Licencia	RNF #13	Licenciamiento del Sistema AiresProxyAudit.	La aplicación debe especificar los derechos de propiedad de los autores.
Tipo	**Nº**	**Nombre**	**Descripción**

Requisitos Legales, de Derecho de Autor y otros.	RNF #14	Requisitos legales y de derecho de autor.	• El sistema debe ajustarse y regirse por la ley, decretos leyes, decretos, resoluciones y manuales (órdenes) establecidos, que norman los procesos que serán automatizados. • Como producto, se distribuirá amparado bajo las normativas legales establecidas en el registro comercial emitido por las entidades jurídicas de la Universidad de las Ciencias Informáticas.
Tipo	**Nº**	**Nombre**	**Descripción**
Estándares Aplicables	RNF #15	Estándares aplicables al Sistema AiresProxyAudit.	El desarrollo del software estará regido por las normas establecidas para el Nivel 2 de CMMI aplicado en la Universidad de las Ciencias Informáticas.

ANEXO C.
Descripción de los Casos de Uso del Sistema.

CU 1. Adicionar entidad.

Objetivo	Crear y guardar la entidad (estructura organizacional en forma de árbol) de una empresa.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea crear la estructura organizacional de la empresa, el sistema muestra la interfaz correspondiente con la opción Adicionar entidad, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema.
Postcondiciones	Se adicionó la entidad (estructura organizacional) al sistema.
Flujo de eventos	
Flujo básico <Nombre del flujo básico>	

	Actor	Sistema
1.	Selecciona la opción “Crear entidad”.	
2.		Muestra la interfaz con la estructura organizacional predefinida.
3.	Crea la entidad base que representa la empresa.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Muestra una vista en forma de árbol con la estructura organizacional creada.
6.		Termina el CU.
Flujos alternos		
Nº Evento 3a Crear entidad hijo.		
	Actor	**Sistema**
1.	Selecciona entidad padre.	
2.		Muestra la opción “Crear entidad hijo”.
3.	Crea la entidad hijo.Ir paso 4.	
Nº Evento 3b Crear entidad a través de fichero yml.		
	Actor	**Sistema**
1.	Carga fichero yml con la estructura organizacional. Ir paso 4.	
Nº Evento 4a Datos incorrectos.		
	Actor	**Sistema**
1.		Muestra el siguiente mensaje: “Los datos de la entidad son incorrectos”. Ir paso 2.
Sección 1: “Nombre”		

	Actor	Sistema
1.		
2.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 2. Modificar entidad.

Objetivo	Modificar la entidad (estructura organizacional en forma de árbol) de una empresa.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea modificar la entidad, el sistema muestra la interfaz correspondiente con la opción Modificar, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos una entidad.
Postcondiciones	Se modificó la entidad (estructura organizacional).
Flujo de eventos	
Flujo básico <Nombre del flujo básico>	

	Actor	Sistema

1.	Selecciona la opción “Modificar entidad”.	
2.		Muestra la interfaz con la estructura organizacional predefinida.
3.	Selecciona la entidad deseada y la modifica.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Muestra una vista en forma de árbol con la estructura organizacional creada.
6.		Termina el CU.
Flujos alternos		
Nº Evento 3a Cambio de entidad padre.		
	Actor	**Sistema**
1.	Selecciona la nueva entidad padre.	
2.		Asigna la nueva entidad padre a la entidad seleccionada. Ir paso 4.
Nº Evento 4a Datos incorrectos.		
	Actor	**Sistema**
1.		Muestra el siguiente mensaje: “Los datos de la entidad son incorrectos”. Ir paso 2.
Sección 1: “Nombre”		
	Actor	**Sistema**
1.		
2.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		

	Actor	Sistema
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 3. Listar entidad.

Objetivo	Listar las entidades de una empresa.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea listar las entidades existentes, el sistema muestra la interfaz correspondiente con la opción Listar, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos una entidad.
Postcondiciones	Se listaron todas las entidades existentes.

Flujo de eventos

Flujo básico <Nombre del flujo básico>

	Actor	Sistema
1.	Selecciona la opción “Listar entidad”.	
2.		Muestra todas las entidades existentes con su información (Nombre, usuarios y auditores que contiene).
3.		Termina el CU.

Flujos alternos		
Nº Evento.		
	Actor	**Sistema**
1.		
2.		
Sección 1: "Nombre"		
	Actor	**Sistema**
1.		
2.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		
Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 4. Eliminar entidad.

Objetivo	Eliminar una entidad de una empresa.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea eliminar una entidad, el sistema muestra la interfaz correspondiente con la opción Eliminar, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico

Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos una entidad.
Postcondiciones	Se eliminó la entidad (estructura organizacional).

Flujo de eventos

Flujo básico <Nombre del flujo básico>

	Actor	Sistema
1.	Selecciona la opción “Eliminar entidad”.	
2.		Muestra la interfaz con la estructura organizacional predefinida.
3.	Selecciona la entidad que desea eliminar.	
4.		Muestra un mensaje de confirmación “Está seguro que desea eliminar la entidad’.
5.	Confirma que desea eliminar la entidad seleccionada.	
6.		Elimina la entidad de la base de datos, así como los permisos de auditores que pueden revisar esa entidad y todos los usuarios que pertenecen a la misma.
7.		Actualiza la vista del árbol que representa la entidad (estructura organizacional).
8.		Termina el CU.

Flujos alternos

Nº Evento 5a Cancelación de la acción.

	Actor	Sistema
1.	Cancela la acción de eliminar la entidad seleccionada. Ir paso 8.	
2.		

Sección 1: “Nombre”

	Actor	Sistema

1.		
2.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 5. Exportar estructura organizacional.

Objetivo	Exportar la estructura organizacional (entidades, usuarios, auditores) en forma de árbol de una empresa.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea exportar la estructura organizacional de una empresa, el sistema muestra la interfaz correspondiente con la opción Exportar estructura, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha creado la estructura organizacional.
Postcondiciones	Se exportó la estructura organizacional.
Flujo de eventos	
Flujo básico <Nombre del flujo básico>	
Actor	**Sistema**
1. Selecciona la opción “Exportar	

	estructura".	
2.		Muestra la interfaz correspondiente que permite guardar en formato yml la estructura organizacional.
3.	Selecciona donde quiere guardar el fichero yml de la estructura organizacional.	
4.		Termina el CU.
Flujos alternos		
Nº Evento.		
	Actor	**Sistema**
1.		
2.		
Sección 1: "Nombre"		
	Actor	**Sistema**
3.		
4.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 6.Adicionar usuario.

Objetivo	Adicionar un usuario a una entidad.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea adicionar un usuario a una entidad determinada, el sistema muestra la interfaz correspondiente con la opción Adicionar usuario, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos una entidad.
Postcondiciones	Se adicionó el usuario a la entidad deseada.

Flujo de eventos

Flujo básico <Nombre del flujo básico>

	Actor	**Sistema**
1.	Selecciona la opción “Adicionar usuario”.	
2.		Muestra la interfaz correspondiente para adicionar el usuario.
3.	Inserta los datos correspondientes al usuario a partir de campo de entrada.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Termina el CU.

Flujos alternos

Nº Evento 3a Insertar datos de usuario cargando fichero.

	Actor	**Sistema**
1.	Inserta los datos correspondientes al usuario a partir de fichero cargado. Ir paso 4.	

Nº Evento 4a Datos incorrectos.

	Actor	**Sistema**

1.		Muestra el siguiente mensaje: “Los datos del usuario son incorrectos”. Ir paso 2.
Sección 1: “Nombre”		
	Actor	**Sistema**
2.		
3.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 7.Modificar usuario.

Objetivo	Modificar un usuario.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea modificar un usuario, el sistema muestra la interfaz correspondiente con la opción Modificar usuario, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos un usuario.
Postcondiciones	Se modificó el usuario.

Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	Selecciona la opción “Modificar usuario”.	
2.		Muestra la interfaz correspondiente para modificar el usuario.
3.	Selecciona el usuario deseado y modifica sus datos, así como a las entidades a las que pertenece.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Termina el CU.
Flujos alternos		
Nº Evento 4a Datos incorrectos.		
	Actor	**Sistema**
1.		Muestra el siguiente mensaje: “Los datos del usuario son incorrectos”. Ir paso 2.
Sección 1: “Nombre”		
	Actor	**Sistema**
2.		
3.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	

Requisitos no funcionales	
Asuntos pendientes	Posibles mejoras al caso de uso.

CU 8.Mostrar usuario.

Objetivo	Mostrar todos los usuarios con su información correspondiente.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea mostrar todos los usuarios con su información correspondiente, el sistema muestra la interfaz con la opción Mostrar usuario, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos un usuario.
Postcondiciones	Se mostraron todos los usuarios existentes.

Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	Selecciona la opción “Mostrar usuario”.	
2.		Muestra la interfaz correspondiente.
3.	Selecciona la entidad de la que quiere mostrar todos los usuarios que pertenecen a la misma.	
4.		Muestra todos los usuarios pertenecientes a la entidad seleccionada.
5.		Termina el CU.
Flujos alternos		
Nº Evento 3a Seleccionar usuario.		
	Actor	**Sistema**
1.	Selecciona el usuario del cual quiere	

	mostrar todas las entidades a las que pertenece.	
2.		Muestra todas las entidades pertenecientes al usuario seleccionado. Ir paso 5.

Sección 1: "Nombre"		
	Actor	**Sistema**
6.		
7.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 9.Eliminar usuario.

Objetivo	Eliminar un usuario de una entidad.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea eliminar un usuario, el sistema muestra la interfaz correspondiente con la opción Eliminar, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema.

	Se ha insertado al menos un usuario.
Postcondiciones	Se eliminó el usuario.

Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	Selecciona la opción “Eliminar usuario”.	
2.		Muestra la interfaz con la estructura organizacional predefinida.
3.	Selecciona el usuario que desea eliminar.	
4.		Muestra un mensaje de confirmación “Está seguro que desea eliminar el usuario”.
5.	Confirma que desea eliminar el usuario seleccionado.	
6.		Elimina el usuario de la entidad a la cual pertenece.
7.		Actualiza la vista del árbol que representa la entidad (estructura organizacional).
8.		Termina el CU.
Flujos alternos		
Nº Evento 5a Cancelación de la acción.		
	Actor	**Sistema**
1.	Cancela la acción de eliminar el usuario seleccionado. Ir paso 8.	
2.		
Sección 1: “Nombre”		
	Actor	**Sistema**
3.		

4.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 10.Adicionar auditor.

Objetivo	Adicionar un auditor a una entidad.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea adicionar un auditor a una entidad determinada, el sistema muestra la interfaz correspondiente con la opción Adicionar auditor, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos una entidad.
Postcondiciones	Se adicionó el auditor a la entidad deseada.
Flujo de eventos	
Flujo básico <Nombre del flujo básico>	

	Actor	**Sistema**
1.	Selecciona la opción “Adicionar auditor”.	
2.		Muestra la interfaz correspondiente para adicionar el auditor.

3.	Selecciona la entidad a la cual quiere adicionarle el auditor y luego inserta los datos correspondientes al mismo, a partir de un campo de entrada.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Termina el CU.
Flujos alternos		
Nº Evento 4a Datos incorrectos.		
	Actor	**Sistema**
2.		Muestra el siguiente mensaje: “Los datos del auditor son incorrectos”. Ir paso 2.
Sección 1: “Nombre”		
	Actor	**Sistema**
4.		
5.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 11.Modificar auditor.

Objetivo	Modificar los datos pertenecientes a un auditor.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea modificar un auditor, el sistema muestra la interfaz correspondiente con la opción Modificar auditor, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos un auditor.
Postcondiciones	Se modificó el auditor.

Flujo de eventos

Flujo básico <Nombre del flujo básico>

	Actor	**Sistema**
1.	Selecciona la opción “Modificar auditor”.	
2.		Muestra la interfaz correspondiente para modificar el auditor.
3.	Selecciona el auditor deseado y modifica sus datos, así como a las entidades a las que pertenece.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Termina el CU.

Flujos alternos

Nº Evento 4a Datos incorrectos.

	Actor	**Sistema**
1.		Muestra el siguiente mensaje: “Los datos del auditor son incorrectos”. Ir paso 2.

Sección 1: “Nombre”

	Actor	**Sistema**
1.		
2.		

Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		
Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 12.Buscar auditor.

Objetivo	Buscar un auditor específico.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea buscar un auditor determinado, el sistema muestra la interfaz con la opción Buscar auditor, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos un auditor.
Postcondiciones	Se muestran los datos del auditor encontrado.

Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	Selecciona la opción “Buscar auditor”.	
2.		Muestra la interfaz que permite insertar el criterio de búsqueda del auditor.
3.	Inserta el criterio de búsqueda.	

4.		Muestra todos los auditores que cumplen con el criterio de búsqueda especificado.
5.		Termina el CU.
Flujos alternos		
Nº Evento 4a No encuentra auditor.		
	Actor	**Sistema**
1.		Muestra el siguiente mensaje “No hay auditores que cumplan con el criterio de búsqueda especificado”. Ir paso 5.
Sección 1: “Nombre”		
	Actor	**Sistema**
1.		
2.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 13.Listar auditor.

Objetivo	Listar todos los auditores con su información correspondiente.
Actores	Administrador.

Resumen	El caso de uso se inicia cuando el administrador desea listar todos los auditores con su información correspondiente, el sistema muestra la interfaz con la opción Listar auditor, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos un auditor.
Postcondiciones	Se listan todos los auditores existentes.

Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	Selecciona la opción “Listar auditor”.	
2.		Muestra todos los auditores, detallando el nombre y la entidad a la cual puede auditar.
3.		Termina el CU.
Flujos alternos		
Nº Evento		
	Actor	**Sistema**
1.		
2.		
Sección 1: “Nombre”		
	Actor	**Sistema**
1.		
2.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**

1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 14.Eliminar auditor.

Objetivo	Eliminar un auditor de una entidad.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea eliminar un auditor, el sistema muestra la interfaz correspondiente con la opción Eliminar, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos un auditor.
Postcondiciones	Se eliminó el auditor.

Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	Selecciona la opción “Eliminar auditor”.	
2.		Muestra la interfaz con la estructura organizacional predefinida.
3.	Selecciona el auditor que desea eliminar de la entidad escogida.	
4.		Muestra un mensaje de confirmación “Está seguro que desea eliminar el auditor”.

5.	Confirma que desea eliminar el auditor seleccionado.	
6.		Elimina el auditor de la entidad a la cual pertenece.
7.		Actualiza la vista del árbol que representa la entidad (estructura organizacional).
8.		Termina el CU.
Flujos alternos		
Nº Evento 5a Cancelación de la acción.		
	Actor	**Sistema**
1.	Cancela la acción de eliminar el auditor seleccionado. Ir paso 8.	
2.		
Sección 1: "Nombre"		
	Actor	**Sistema**
5.		
6.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 15.Adicionar dominio.

Objetivo	Adicionar un dominio al sistema.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea adicionar un dominio al sistema, el sistema muestra la interfaz correspondiente con la opción Adicionar dominio, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos una categoría.
Postcondiciones	Se adicionó el dominio a la categoría deseada.

Flujo de eventos

Flujo básico <Nombre del flujo básico>

	Actor	**Sistema**
1.	Selecciona la opción “Adicionar dominio”.	
2.		Muestra la interfaz correspondiente para adicionar el dominio.
3.	Selecciona la categoría a la cual quiere adicionarle el dominio y luego inserta los datos correspondientes al mismo, a partir de un campo de entrada.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Termina el CU.

Flujos alternos

Nº Evento 4a Datos incorrectos.

	Actor	**Sistema**
1.		Muestra el siguiente mensaje: “Los datos del dominio son incorrectos”. Ir paso 2.

Sección 1: “Nombre”

	Actor	Sistema
1.		
2.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 16.Listar dominio.

Objetivo	Listar todos los dominios con su información correspondiente.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea listar todos los dominios con su información correspondiente, el sistema muestra la interfaz con la opción Listar dominio, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos un dominio.
Postcondiciones	Se listan todos los dominios existentes.
Flujo de eventos	
Flujo básico <Nombre del flujo básico>	

	Actor	Sistema
1.	Selecciona la opción “Listar dominio”.	

2.		Muestra la interfaz correspondiente para listar dominio.
3.	Selecciona una categoría determinada.	
4.		Muestra todos los dominios que pertenecen a la categoría seleccionada.
5.		Termina el CU.
Flujos alternos		
Nº Evento 3a No seleccionar categoría		
	Actor	**Sistema**
1.	No especifica una categoría.	
2.		Muestra todos los dominios existentes con su categoría. Ir paso 5.
Sección 1: "Nombre"		
	Actor	**Sistema**
3.		
4.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 17.Modificar dominio.

Objetivo	Modificar los datos pertenecientes a un dominio.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea modificar un dominio, el sistema muestra la interfaz correspondiente con la opción Modificar dominio, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos un dominio.
Postcondiciones	Se modificó el dominio.

Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	Selecciona la opción “Modificar dominio”.	
2.		Muestra la interfaz correspondiente para modificar el dominio.
3.	Selecciona el dominio deseado y modifica su nombre, así como a la categoría a la que pertenece si así lo desea.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Termina el CU.
Flujos alternos		
Nº Evento 4a Datos incorrectos.		
	Actor	**Sistema**
2.		Muestra el siguiente mensaje: “Los datos del dominio son incorrectos”. Ir paso 2.
Sección 1: “Nombre”		
	Actor	**Sistema**

3.		
4.		

Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 18.Insertar categoría.

Objetivo	Insertar una categoría al sistema.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea insertar una categoría al sistema, el sistema muestra la interfaz correspondiente con la opción Insertar dominio, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema.
Postcondiciones	Se insertó una categoría.

Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	Selecciona la opción “Adicionar categoría”.	
2.		Muestra la interfaz correspondiente para adicionar una categoría.

3.	Inserta los datos correspondientes a la categoría.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Termina el CU.
Flujos alternos		
Nº Evento 4a Datos incorrectos.		
	Actor	**Sistema**
2.		Muestra el siguiente mensaje: “Los datos de la categoría son incorrectos”. Ir paso 2.
Sección 1: “Nombre”		
	Actor	**Sistema**
3.		
4.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 19.Modificar categoría.

Objetivo	Modificar los datos pertenecientes a una categoría.
Actores	Administrador.

Resumen	El caso de uso se inicia cuando el administrador desea modificar una categoría, el sistema muestra la interfaz correspondiente con la opción Modificar categoría, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos una categoría.
Postcondiciones	Se modificó la categoría.

Flujo de eventos

Flujo básico <Nombre del flujo básico>

	Actor	**Sistema**
1.	Selecciona la opción “Modificar categoría”.	
2.		Muestra la interfaz correspondiente para modificar la categoría.
3.	Selecciona la categoría deseada y modifica sus datos.	
4.		Verifica que los datos son correctos y los guarda en la BD.
5.		Termina el CU.

Flujos alternos

Nº Evento 4a Datos incorrectos.

	Actor	**Sistema**
1.		Muestra el siguiente mensaje: “Los datos de la categoría son incorrectos”. Ir paso 2.

Sección 1: “Nombre”

	Actor	**Sistema**
5.		
6.		

Flujos alternos

Nº Evento <Condición que dio lugar a la extensión>

	Actor	Sistema
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 20.Listar categoría.

Objetivo	Listar todas las categorías con su información correspondiente.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea listar todas las categorías con su información correspondiente, el sistema muestra la interfaz con la opción Listar categoría, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos una categoría.
Postcondiciones	Se listan todas las categorías existentes.
Flujo de eventos	
Flujo básico <Nombre del flujo básico>	

	Actor	Sistema
1.	Selecciona la opción “Listar categoría”.	
2.		Muestra todas las categorías existentes en el sistema.
3.		Termina el CU.

Flujos alternos
Nº Evento

	Actor	Sistema
3.		
4.		
Sección 1: "Nombre"		
	Actor	**Sistema**
5.		
6.		
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.		
2.		

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

CU 21.Eliminar categoría.

Objetivo	Eliminar una categoría del sistema.
Actores	Administrador.
Resumen	El caso de uso se inicia cuando el administrador desea eliminar una categoría, el sistema muestra la interfaz correspondiente con la opción Eliminar categoría, finaliza así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El administrador se ha autenticado en el sistema. Se ha insertado al menos una categoría.

Postcondiciones	Se eliminó la categoría.

Flujo de eventos

Flujo básico <Nombre del flujo básico>

	Actor	Sistema
1.	Selecciona la opción “Eliminar categoría”.	
2.		Muestra la interfaz con la estructura organizacional predefinida.
3.	Selecciona la categoría que desea eliminar.	
4.		Muestra un mensaje de confirmación “Está seguro que desea eliminar la categoría”.
5.	Confirma que desea eliminar la categoría seleccionada.	
6.		Actualiza la vista del árbol que representa estructura organizacional.
7.		Termina el CU.

Flujos alternos

Nº Evento 5a Cancelación de la acción.

	Actor	Sistema
3.	Cancela la acción de eliminar la categoría seleccionada. Ir paso 7.	
4.		

Sección 1: “Nombre”

	Actor	Sistema
7.		
8.		

Flujos alternos

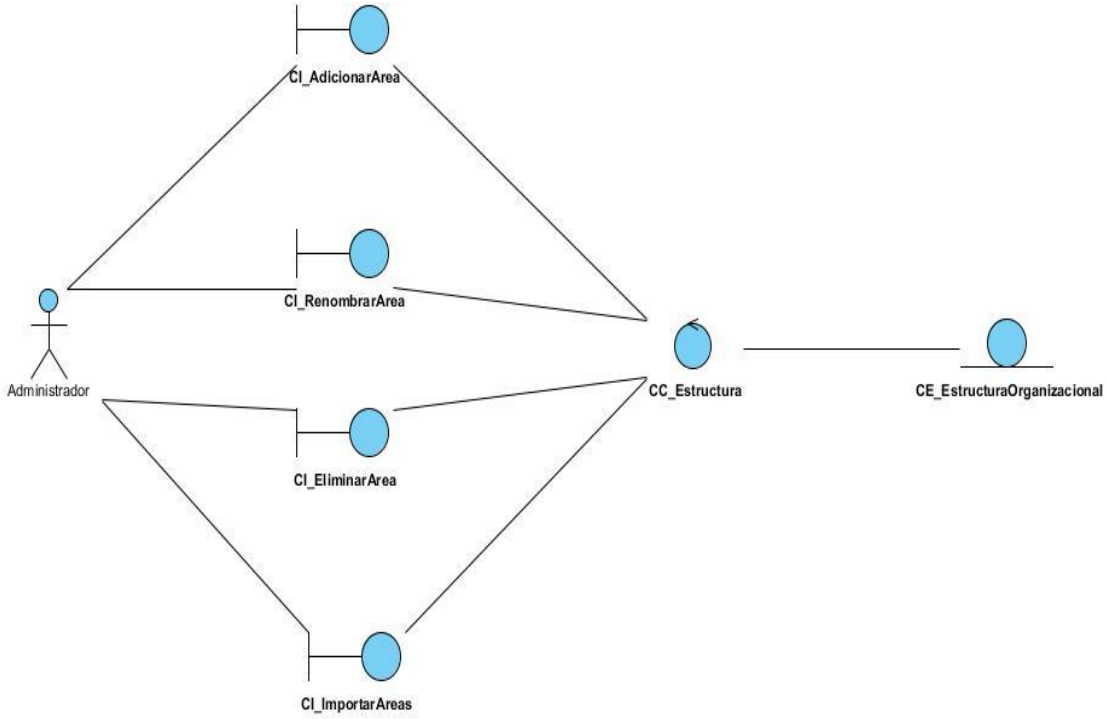
Nº Evento <Condición que dio lugar a la extensión>

	Actor	Sistema

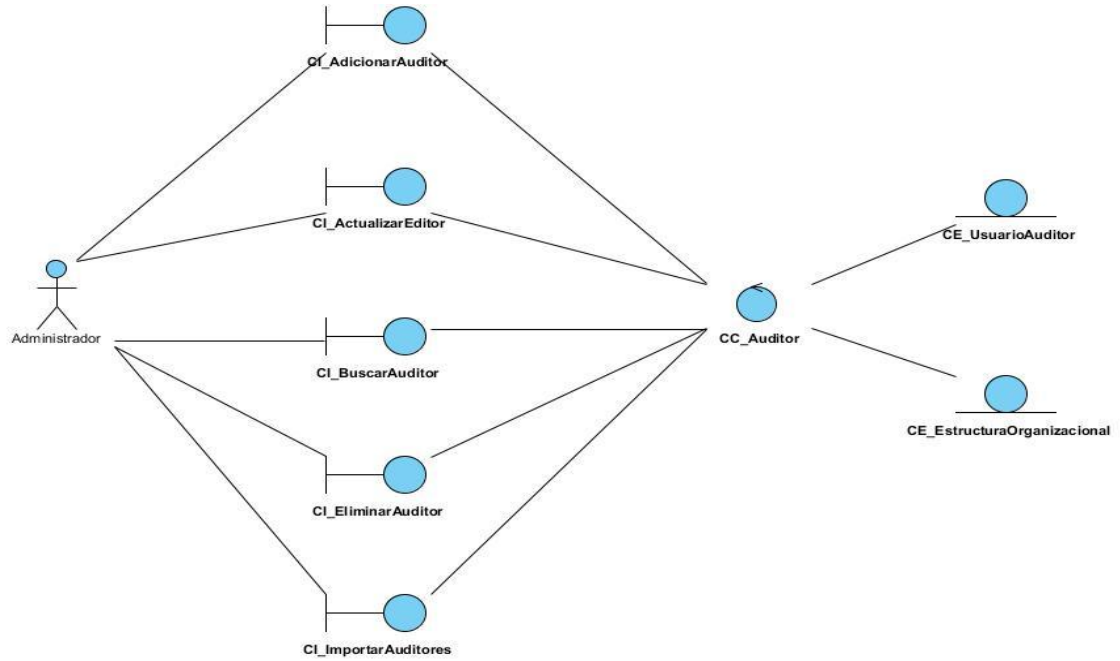
1.		
2.		
Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes	Posibles mejoras al caso de uso.	

ANEXO D.

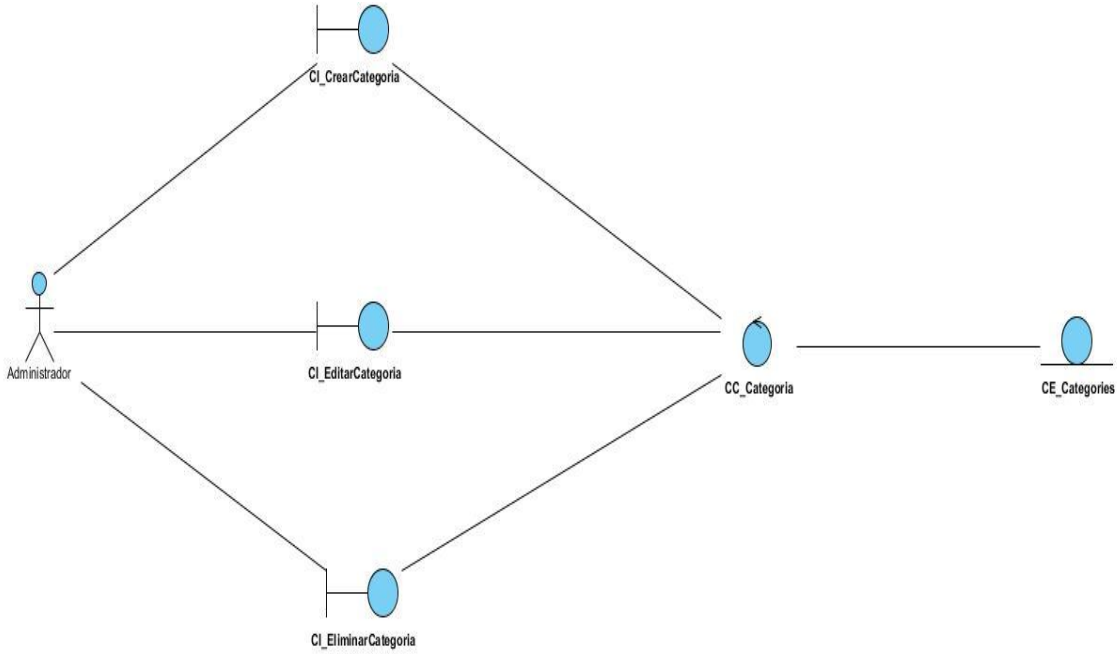
Diagramas de Clases del Análisis



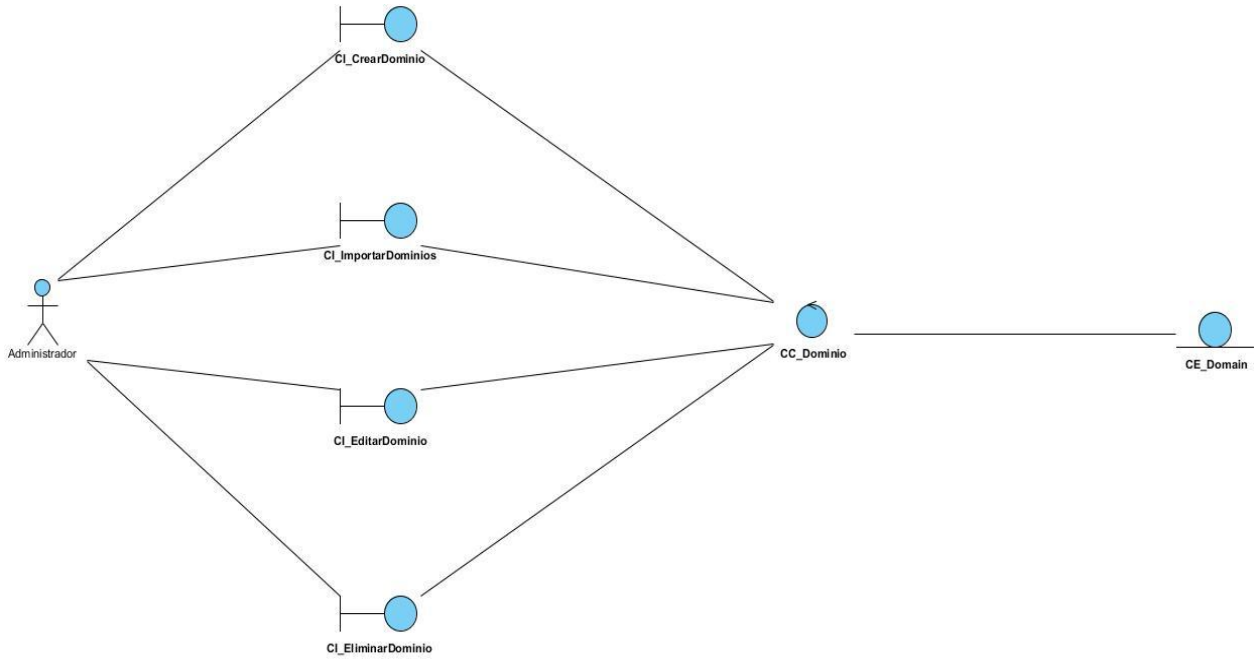
DCA_GestionarÁrea



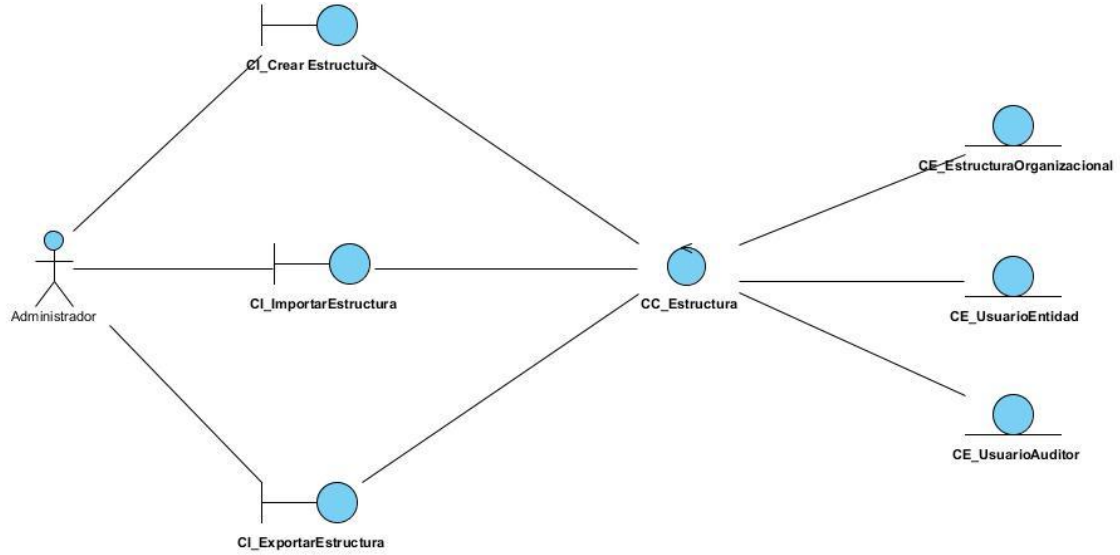
DCA_GestionarAuditor



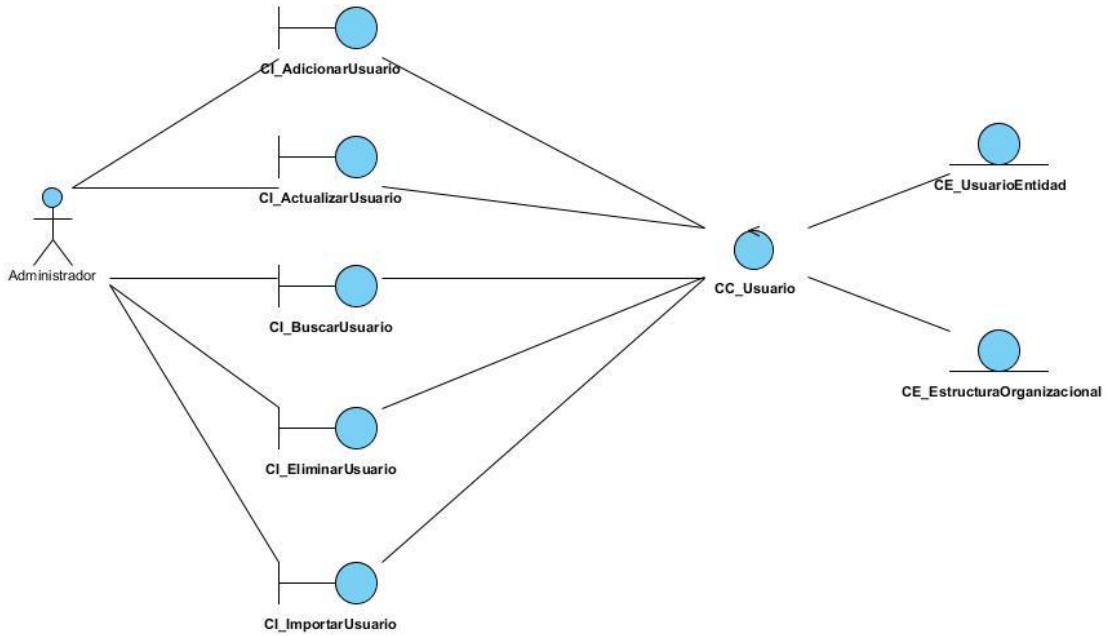
DCA_GestionarCategorías



DCA_GestionarDominios



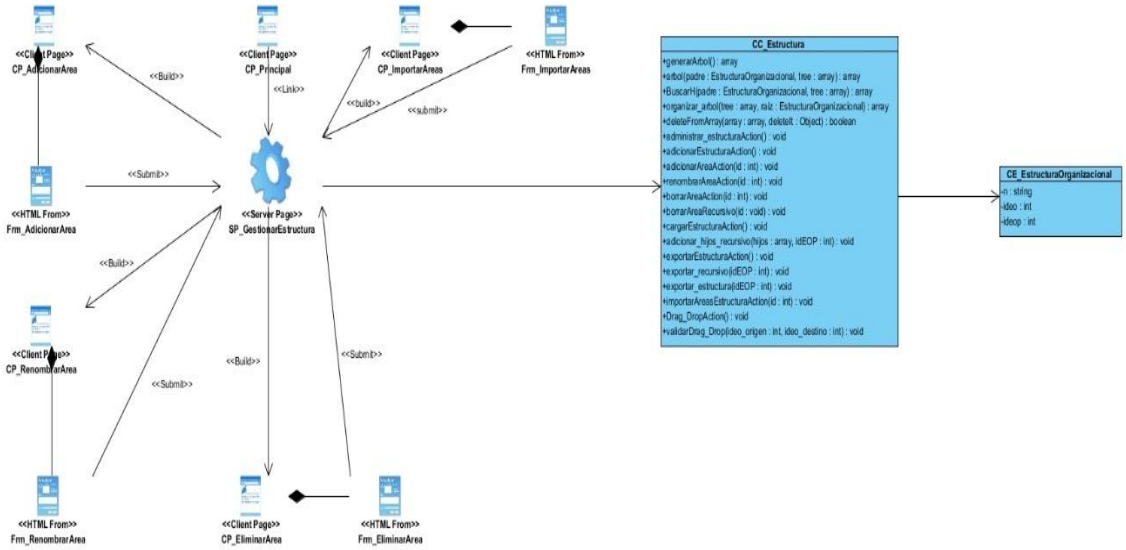
DCA_GestionarEstructura



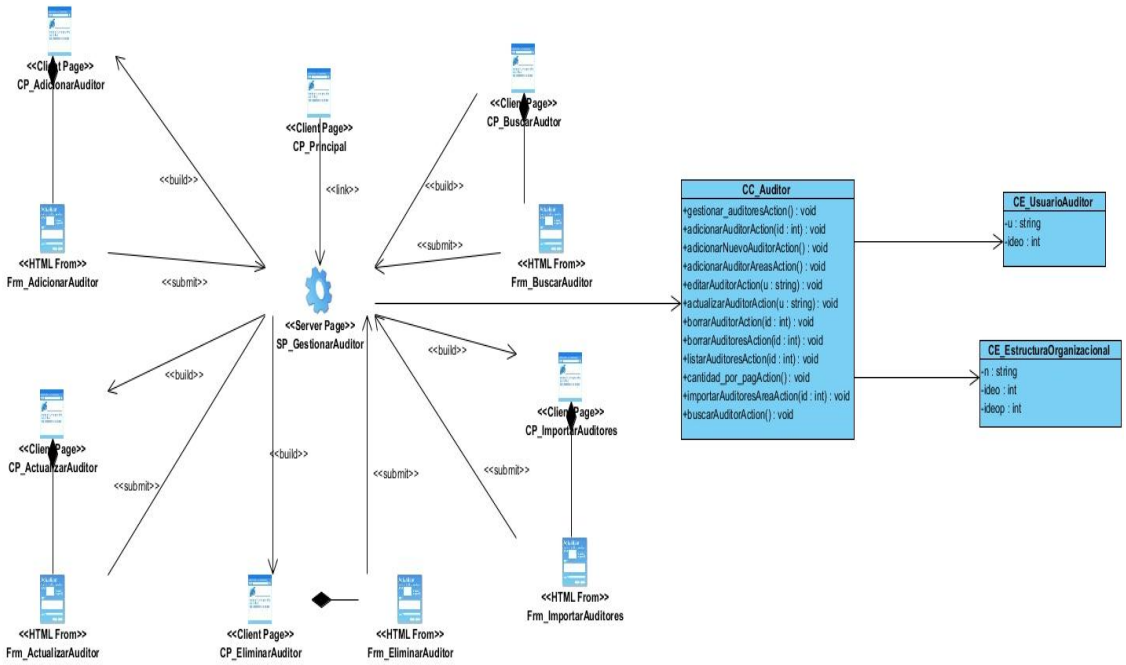
DCA_GestionarUsuario

ANEXO E.

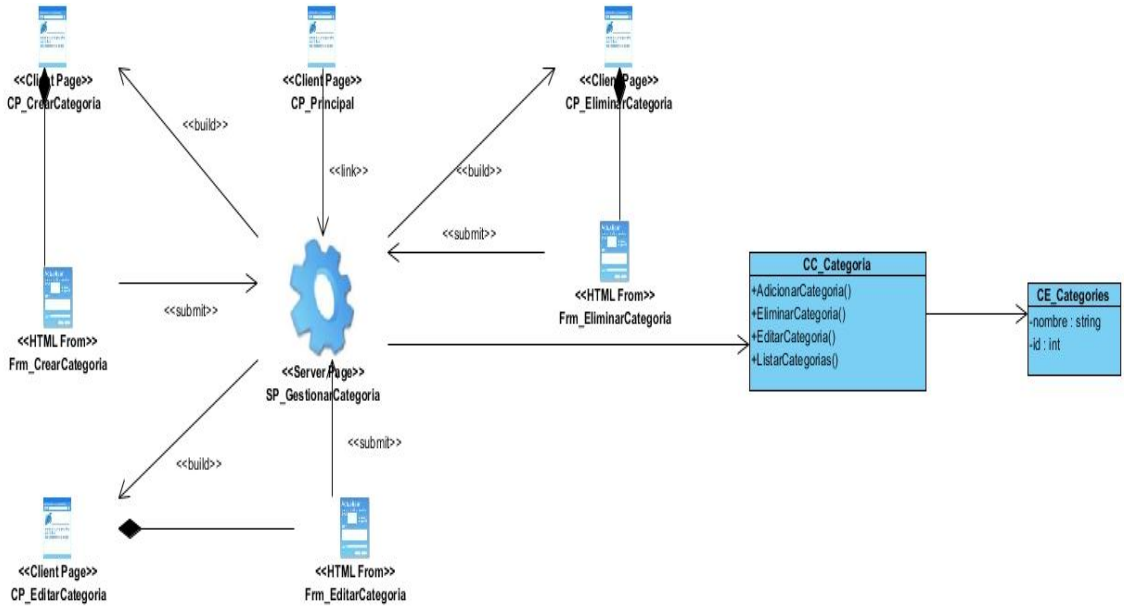
Diagramas de Estereotipos Web



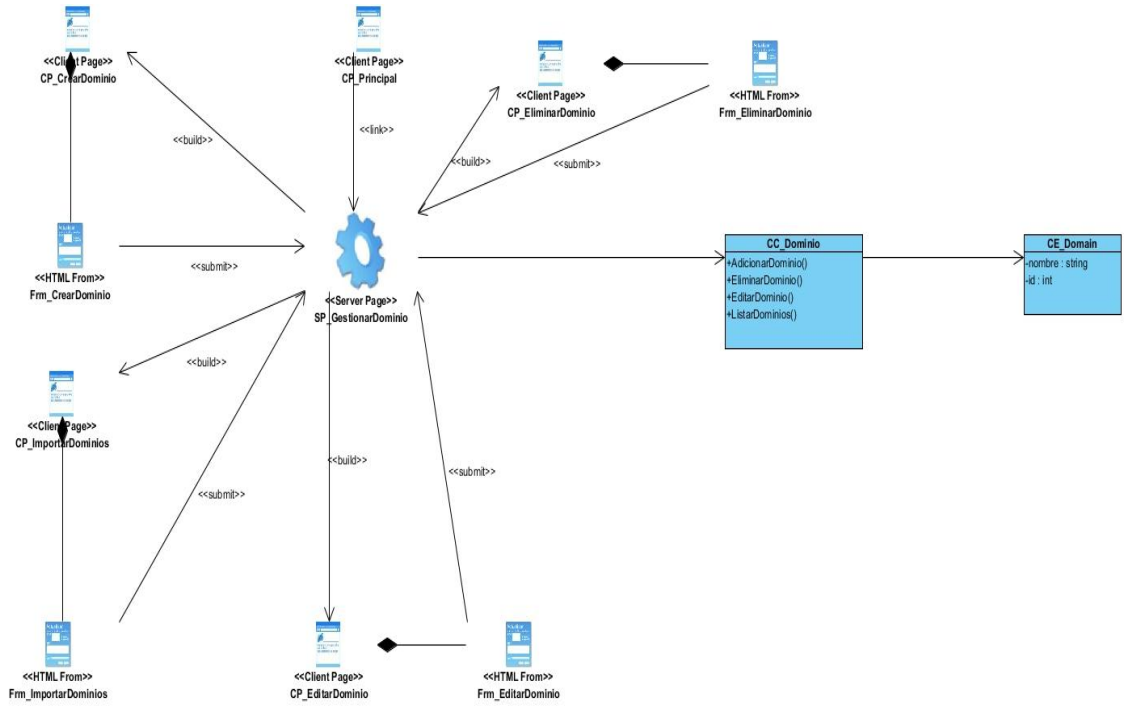
DEW_GestionarÁrea



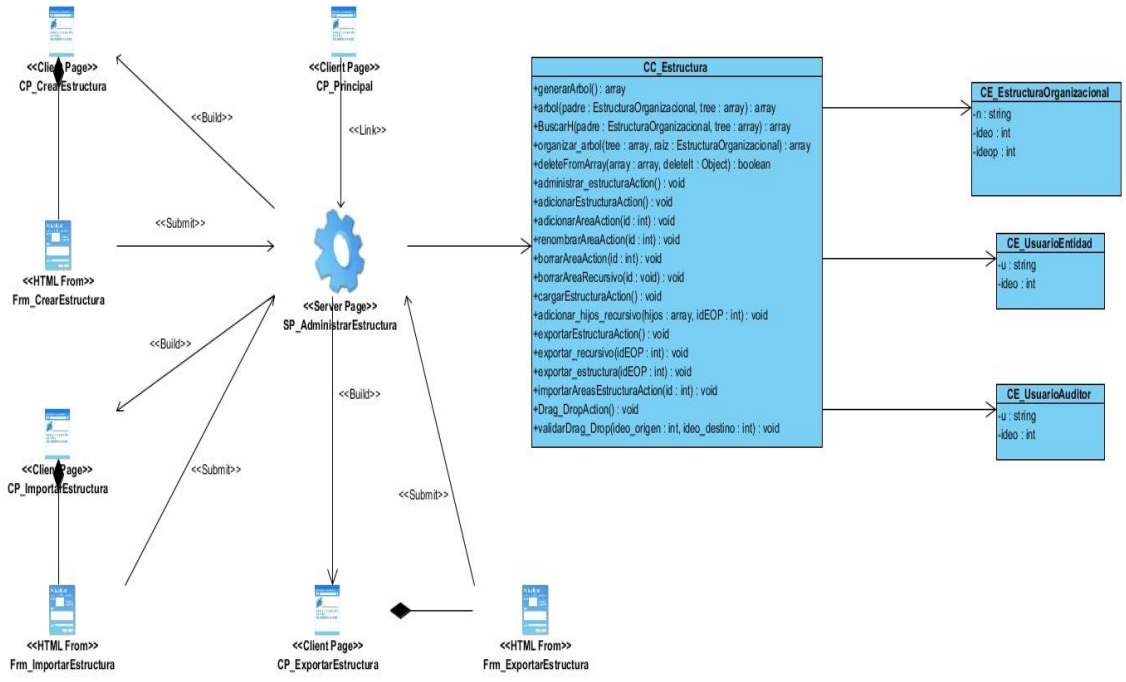
DEW_GestionarAuditor



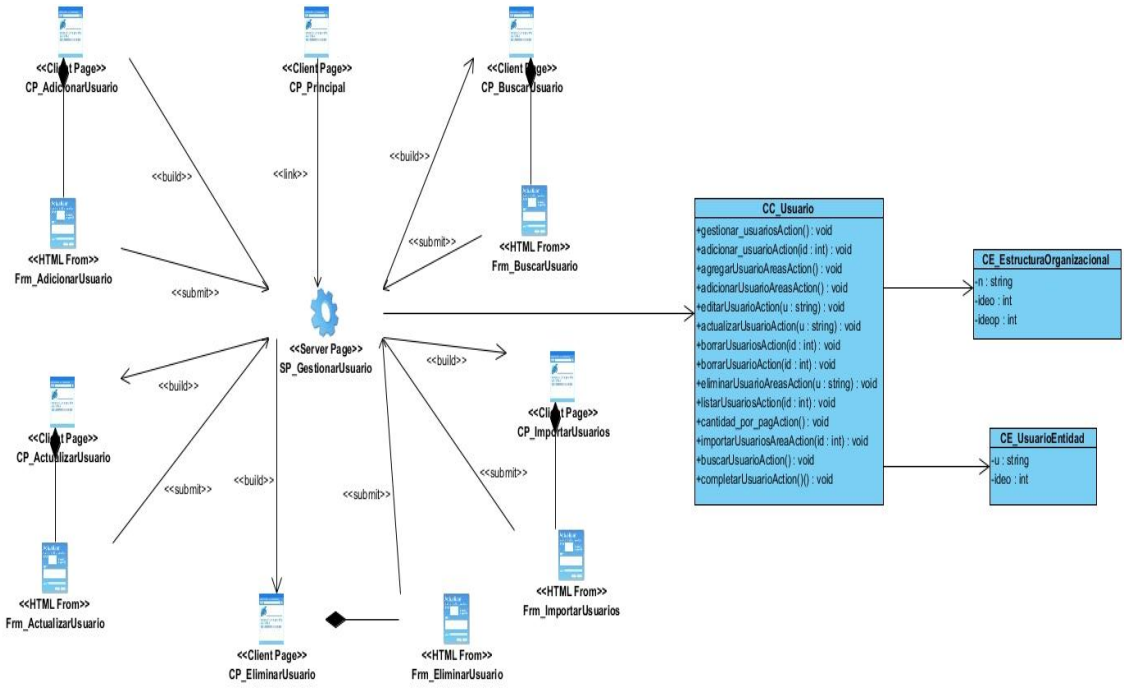
DEW_GestionarCategoría



DEW_GestionarDominio

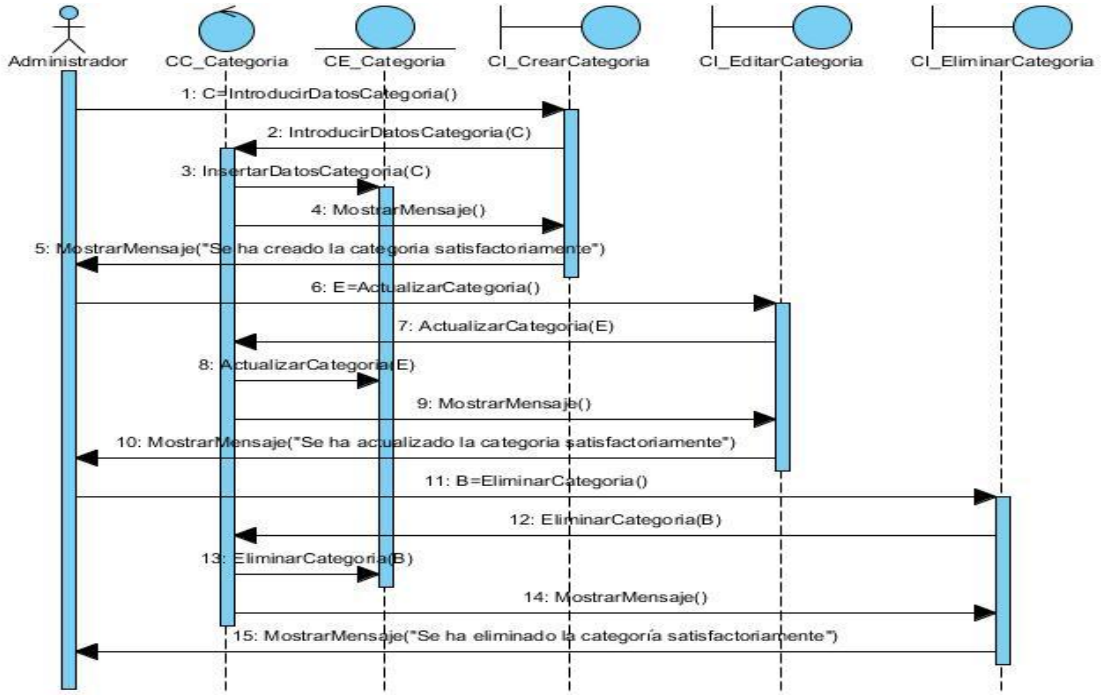


DEW_GestionarEstructura

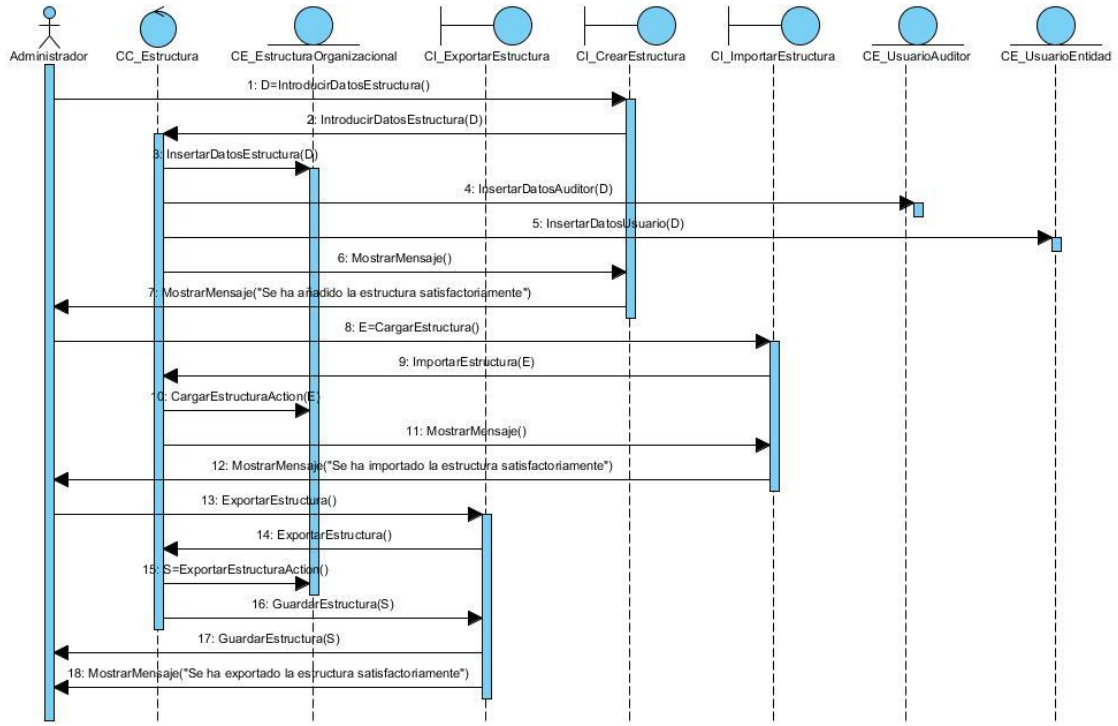


DEW_GestionarUsuario

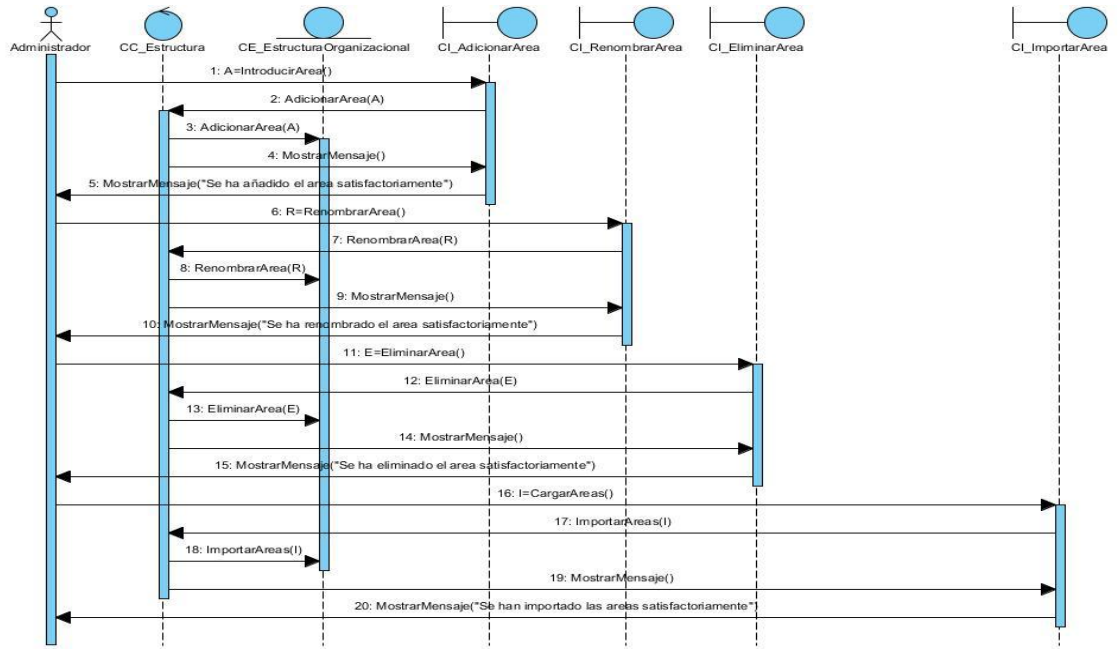
ANEXO F.
Diagramas de Secuencia



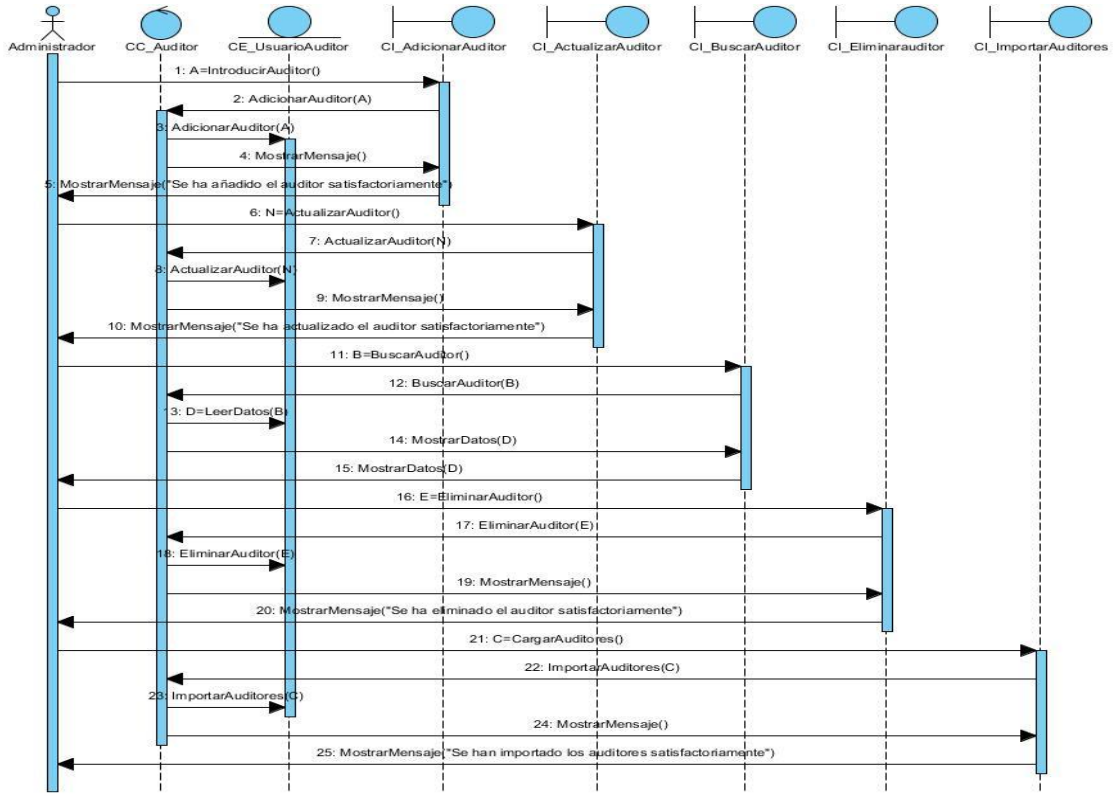
DS_GestionarCategorías.



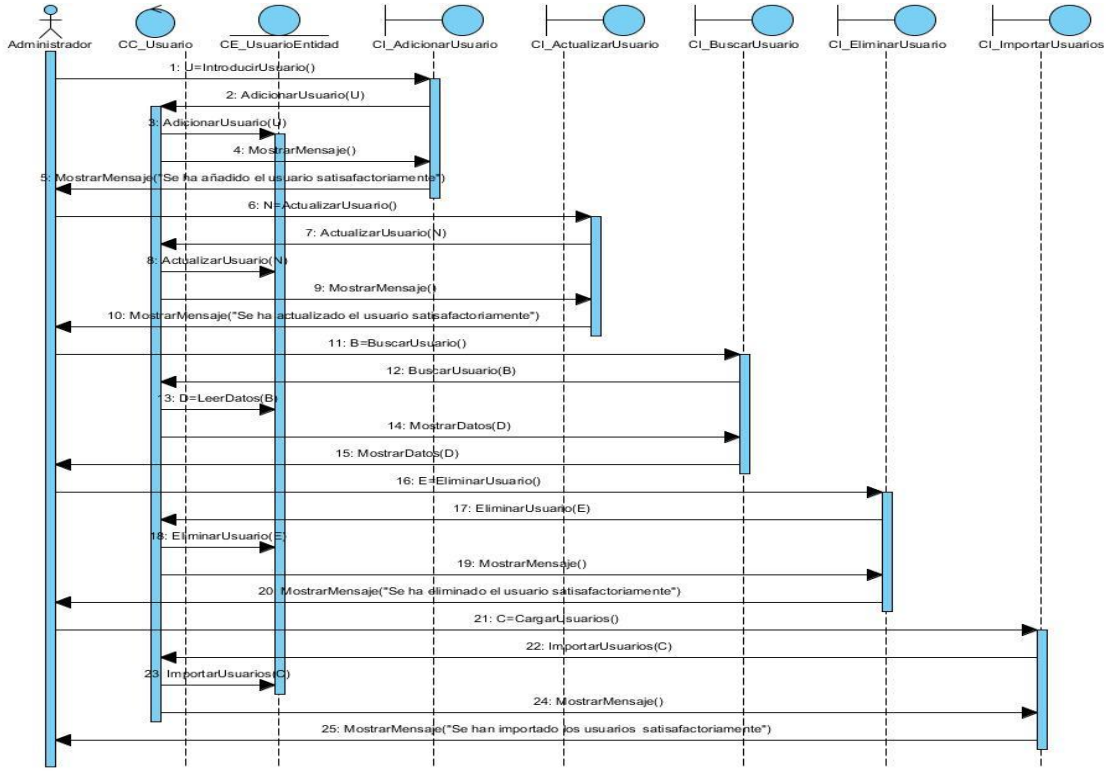
DS_AdministrarEstructura.



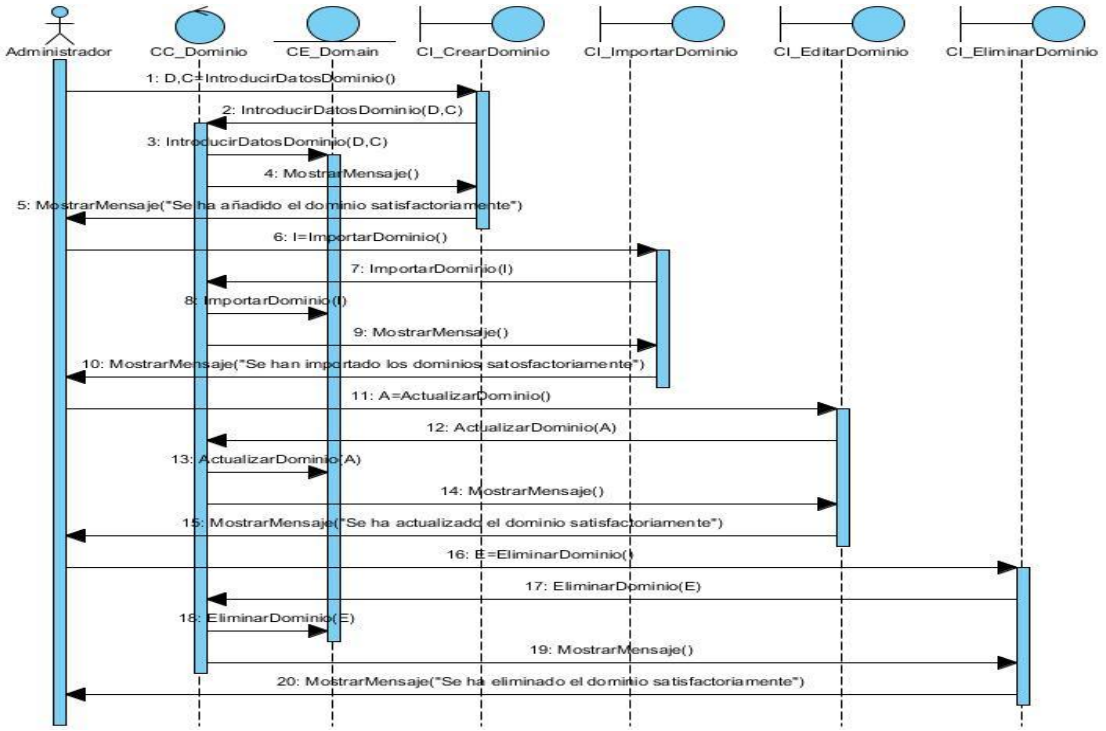
DS_GestionarÁrea.



DS_GestionarAuditor.



DS_GestionarUsuario.



DS_GestionarDominio.

ANEXO G.
Estándares de Codificacion.

El nombre de las clases debe ser declarado utilizando el estilo **StudlyCaps** que permite hacer uso de mayúsculas y minúsculas indiscriminadamente.

```
1 ....class EjEmplO()
2 ....{
3 ....//Bloque de instrucciones
4 ....}
```

El nombre de los procedimientos debe ser declarado utilizando el estilo **camelCase**.

```
1 ....functionmiMetodo()
2 ....{
3 ....//Bloque de instrucciones
4 ....}
```

Todo el código PHP debe estar encerrado en las etiquetas **<?php ?>** pero en caso de que el fichero solo contenga código escrito en PHP se omitirá la etiqueta de cierre **?>**.

```
1....//Código no PHP
2....<?php
3....//Código PHP
4....?>
```

```
1....<?php
2....//Código PHP
```

El código debe tener una identación de cuatro espacios sin tabulaciones y la longitud máxima que puede alcanzar una línea es de ochenta caracteres, las que excedan este tamaño deben ser divididas en varias líneas con un tamaño que no viole esta restricción. Además no debe escribir más de una instrucción por línea y no deben existir espacios en blanco al final de las líneas que no estén vacías.

```
1....//Código(longitud máxima 80 caracteres)
```

Las constantes de PHP **true**, **false** y **null** deben estar escritas en minúsculas.

```
1....$variable1=true;
```

Cuando en el fichero se define el espacio de nombre debe existir una línea en blanco a continuación de la declaración. Además cuando están presentes todas las declaraciones de uso se escriben a continuación de la definición de espacio de nombre, se debe utilizar la palabra reservada **use** una vez por cada declaración y debe concluir con una línea en blanco.

```
1....namespace Camino\Paquete;
2....//Línea en blanco
3....use MiClase;
4....use OtraClase;
5….//Línea en blanco
```

Se debe declarar la visibilidad de todos los atributos de las clases y no debe existir más de un atributo por línea de código.

```
1....class MiClase
2....{
3....public $atributo1;
4....public $atributo2;
5….}
```

Se debe declarar la visibilidad de todos los procedimientos y no deben existir espacios en blanco a continuación de los nombre de los mismos, ni a continuación del paréntesis abierto, ni antes del paréntesis cerrado de los parámetros. La llaves deben colocarse una en cada línea de forma que la que abre se encuentre a continuación de la declaración del método y la que cierra a continuación del bloque de instrucciones.

```
1....public function miFuncion($arg1, $arg2, $arg3)
2....{
3..../Bloque de instrucciones
4….}
```

Para las estructuras de control se define que debe existir un espacio en blanco a continuación de la palabra reservada que define la estructura de control, no debe existir un espacio en blanco ni después del paréntesis abierto, ni antes del paréntesis cerrado y debe existir un espacio en blanco entre el paréntesis que cierra y la llave que abre el bloque de instrucciones. El bloque de instrucciones debe estar identado un nivel a la derecha y a continuación debe aparecer la llave que cierra.

```
1....if ($condicion1) {
    2..../Bloque de instrucciones
3....} elseif($condicion2) {
    4..../Bloque de instrucciones
5….} else {
    6…./Bloque de instrucciones
7….}
```

```
1....while ($condicion){
2..../Bloque de instrucciones
3….}
```

```
1....for($i = 0; $i < 10; $i++){
2..../Bloque de instrucciones
3….}
```

Se debe añadir los bloques de documentación al principio de cada clase y procedimiento de la forma que se muestra a continuación.

```
/**
 * Descripción de la clase.
 */
1….class MiClase
2….{
3…./Código de la clase
/**
  * @param string $parámetro descripción del parámetro
*
* @return ValorQueDevuelve
*/
    4....public function miFuncion($parámetro)
    5....{
    6..../Bloque de instrucciones
    7….}
```