Universidad de las Ciencias Informáticas

Facultad 4

Título: "Análisis y diseño del Módulo Administración para CRODA 2.0"

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

Autor: Daniel Hang Carbonell.

Tutor(es): Ing. Osvaldo Ernesto Stable Vilches.

Ing. Jorge Iturria Pozo.

Co-tutora: Ing. Isyed Rodríguez Trujillo.

La Habana, junio de 2012

Año 54 de la Revolución

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro FORTES de la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Daniel Hang Carbonell

_____ _____

Osvaldo Ernesto Stable Vilches Jorge Iturria Pozo

_____

Isyed Rodríguez Trujillo

Resumen

En el presente trabajo se realiza el análisis y diseño del Módulo Administración de la herramienta de autor web CRODA, para contribuir a erradicar ciertas limitaciones encontradas en diversos aspectos administrativos de la versión 1.0, relacionados al proceso de creación de Objetos de Aprendizaje, beneficiando la actividad de gestión de los usuarios, los roles, la configuración, la mensajería y los permisos. De esta forma se fortalece tanto la seguridad de la información como el intercambio de criterios entre usuarios no registrados y administradores; incorporando estos elementos a la versión 2.0 de dicha herramienta de autor web.

Como resultado se obtiene la fundamentación teórica para la elaboración de la investigación, utilizando una bibliografía actualizada de los temas relacionados. Se proponen las distintas herramientas para desarrollar la futura implementación del módulo. Se realiza el levantamiento de requisitos funcionales y no funcionales con que debe contar la aplicación, elaborándose los respectivos casos de usos con su descripción textual, además se realiza el análisis y diseño del Módulo Administración, obteniéndose así los artefactos generados a partir de la metodología escogida para estas fases, además de la validación de los requisitos generados durante el proceso de desarrollo del software, mediante las métricas de calidad de la especificación de requisitos, la creación de prototipos de interfaz de usuario. Se realizó el método de expertos para la verificar la eficacia de la propuesta de solución.

Palabras clave: administración, aplicación, configuración, CRODA, gestión, herramienta de autor, Objetos de Aprendizaje, permisos, rol, seguridad, Unidades de Aprendizaje, usuario.

Índice

Introducción

El uso de la informática como herramienta en el desarrollo de la administración y gestión de procesos en diversas entidades se han convertido en una tarea de vital importancia. Según Jiménez Castro, "*La administración es una ciencia social compuesta de varios principios, técnicas y prácticas cuya aplicación a conjuntos humanos permite establecer sistemas racionales de esfuerzo cooperativo a fin de lograr propósitos comunes que individualmente no es factible lograr*" (1).

En la actualidad, con el avance de las Tecnologías de la Información y las Comunicaciones (TIC) han surgido nuevas ideas que enriquecen y fortalecen la administración, con el objetivo de controlar y asegurar los procesos informatizados y recursos almacenados, en sitios, portales y aplicaciones. Las aplicaciones web, impulsoras del desarrollo del e-learning (educación a distancia), constituyen un ejemplo evidente en el control de los recursos de los Objetos de Aprendizaje (OA) en los repositorios de OA (ROA) o en el control de la gestión del aprendizaje, en los Sistemas de Gestión del Aprendizaje (LMS) o en las herramientas de autor web, para el control y aseguramiento de los materiales educativos que se encuentran en creación.

Dentro de las herramientas de autor web se encuentra CRODA, especializada en la creación de OA. Esta herramienta ha sido desarrollada en la Facultad 4 de la Universidad de las Ciencias Informáticas (UCI) por el Departamento de Producción de Herramientas Educativas del Centro de Tecnologías para la Formación (FORTES). En la versión 1.0 de CRODA el módulo de administración presenta algunas limitantes:

La administración se limita a la gestión de usuarios con solo cinco roles predefinidos, para solo un permiso sobre los OA o las plantillas[1], ya sea de creación o revisión. Esto trae como consecuencia que una persona registrada necesite acceder a la aplicación con más de una cuenta para poder realizar varias funciones. Otro elemento que gestiona la administración solamente es la configuración de la herramienta en cuanto a conexión y cantidad de OA y plantillas que puedan estar en edición. Por estas limitantes, el administrador no tiene posibilidades de controlar ni tener conocimiento de las plantillas y OA que son creados en la herramienta, ni del trabajo desarrollado por los usuarios asociados a estos.

[1] Permiten definir la estructura de los Objetos de Aprendizaje.

Por la necesidad de dar solución a las situaciones antes expuestas surge el siguiente **problema de investigación:** ¿Cómo contribuir a perfeccionar el Módulo Administración en la herramienta de autor web CRODA? Para desarrollar la investigación se define el **objeto de estudio** proceso de la administración en aplicaciones web. El **campo de acción** de la investigación se ha enmarcado en el Módulo Administración de la herramienta de autor web CRODA. El **objetivo general** en el desarrollo de este trabajo es elaborar el análisis y diseño del Módulo Administración que permita mejorar el control del contenido y los usuarios en la herramienta de autor web CRODA 2.0.

Como **idea a defender** se plantea que con la elaboración del análisis y diseño del Módulo Administración de la herramienta de autor web CRODA 2.0, se crean las bases para la implementación de nuevas funcionalidades que permitirán un mayor control de los usuarios y sus actividades de creación, revisión y publicación de los OA, plantillas y las Unidades de Aprendizaje (UoL), facilidades para el control de los roles creados y sus permisos en la herramienta, así como la relación entre usuarios no registrados y la administración.

Para desarrollar el objetivo general se han definido los siguientes **objetivos específicos**:

- Elaborar el marco teórico conceptual a partir de un estudio referente al funcionamiento de la administración en aplicaciones web.
- Elaborar el análisis del Módulo Administración para CRODA.
- Elaborar el diseño del Módulo Administración para CRODA.
- Validar la propuesta de solución del Módulo Administración para CRODA mediante el método de expertos.

Para dar cumplimiento a los objetivos definidos se proponen las siguientes **tareas**:

1. Investigación del funcionamiento de la administración en aplicaciones web.
2. Estudio de las limitaciones en la administración de la herramienta de autor web CRODA.
3. Estudio de las herramientas, metodologías y tecnologías necesarias para el desarrollado del Módulo Administración.
4. Selección de las herramientas, metodologías y tecnologías necesarias para el desarrollado del Módulo Administración.
5. Elaboración del levantamiento de requisitos funcionales y no funcionales.
6. Especificación de actores y casos de usos del sistema.

7. Elaboración de los diagramas de clases del análisis y de clases del diseño del sistema.
8. Validación de los requerimientos del sistema.

Durante la investigación los métodos científicos utilizados son:

- Métodos teóricos:

  Histórico-Lógico: Para analizar la trayectoria desde sus inicios hasta la actualidad de la administración en herramientas web.

  Analítico-Sintético: Se realiza un estudio profundo de la documentación referente al objeto de estudio, además de las propuestas de los sistemas existentes para conformar así una mejor propuesta del diseño del sistema.

  Modelación: Permite la definición y descripción de las funcionalidades que conforman el módulo administración y crear los diseños de interfaz, siendo estas tareas el principio para la integración del módulo sugerido a la herramienta de autor web CRODA.

- Métodos empíricos:

  Entrevista: Constituye un medio vital para obtener información y lograr un mejor punto de vista para desarrollar la investigación. (Ver anexo 1).

Para un mejor entendimiento de este trabajo, el mismo se ha dividido en capítulos, los cuales son descritos a continuación:

Capítulo 1: Fundamentación teórica: Se abordan términos de suma importancia mediante un análisis exhaustivo, como son el estado del arte acerca del tema y sus tendencias principales, técnicas, software y metodologías utilizadas en la propuesta de solución.

Capítulo 2: Características del sistema: Se describe una propuesta del sistema, desarrollando para ello el flujo de trabajo Requerimientos, identificando requisitos funcionales y no funcionales que debe cumplir el sistema, extrayendo de estos los disímiles casos de usos. Se realiza la descripción textual de los mismos y los prototipos de interfaz de usuario correspondientes a cada caso de uso determinado.

Capítulo 3: Análisis y diseño: Se describen los artefactos desarrollados mediante el flujo de trabajo Análisis y Diseño para entender de forma clara la anterior descripción de los requisitos. Se utilizan lenguajes de programación para darle una vista general del sistema a

los desarrolladores. Se desarrollan los artefactos del Diseño como el diagrama de colaboración, modelar el sistema, la arquitectura y definir las clases que son persistentes en la base de datos. Se realiza la validación de la propuesta de solución a través del método de expertos y otras técnicas.

Capítulo I: Fundamentación teórica

Introducción

En este capítulo se realiza un profundo análisis relacionado con las herramientas que emplean la administración web. Conjuntamente se abordan conceptos relacionados con la administración tales como la gestión de roles y usuarios. Se analiza además las metodologías de desarrollo de software, lenguajes de programación y modelado, sistemas gestores de bases de datos y las herramientas con las cuales se desarrolla el producto.

1.1. Herramientas de autor para la creación de Objetos de Aprendizaje

Las herramientas de autor son aplicaciones que permiten a los profesores y maestros realizar las actividades con mayor facilidad, ofreciéndoles guías, elementos predefinidos, ayudas y una interfaz amigable para lograr la creación de materiales educativos y/o cursos en formato digital. Estas se pueden clasificar en tres tipos, las que permiten la creación de materiales educativos digitales, las que pueden generar todos los materiales a incluir en el curso y su publicación y las que generan simulaciones (2).

En el caso de CRODA, esta se apega a la primera clasificación, ya que permite la creación de esos materiales educativos de forma flexible, duraderos y reusables. En la actualidad existen disímiles herramientas de autor con distintas particularidades como son: Ardora, eXeLearning, HotPotatoes, JClic, UDUTU, Reload y Vértice.

Para el desarrollo de la investigación se estudiaron herramientas de autor como: eXeLearning, Ardora, HotPotatoes, JClic y Reload. Estas aplicaciones son de escritorio orientadas al uso personal y carecen de un módulo de administración, debido a que su función principal es la generación de Objetos de Aprendizaje, por lo que no es necesario llevar un control de los usuarios ni la administración de configuraciones que modifiquen la experiencia de uso de roles. Otras de las herramientas estudiadas fueron UDUTU y Vértice, que son aplicaciones en línea accedidas por un gran número de usuarios en ascenso. En el caso de UDUTU su inconveniente es que necesita de herramientas complementarias que logren administrar el comportamiento de estos usuarios en el sistema y brinden una vía para personalizar la experiencia de uso de cada uno, mostrando como limitante que no se ha publicado su código ni informaciones que muestren como se logra la administración en esta aplicación. También los usuarios que han creado un Objeto de Aprendizaje necesitan pagar

una cuota para poder publicarlos en los servidores centrales de esta herramienta. Por otra parte Vértice tiene como principal desventaja que es una aplicación de pago.

CRODA

Como se aborda en la introducción de este trabajo, CRODA es una herramienta de autor web creada en la UCI, que facilita la creación de Objetos de Aprendizaje interoperables, reutilizables, accesibles y duraderos, de manera flexible, aplicando para ello el estándar para la descripción de recursos Learning Object Metadata (LOM) y el estándar de contenido Sharable Content Object Reference Model (SCORM). Esta herramienta de autor web permite la creación de cursos para los distintos entornos educativos. Sobre dicha herramienta se desarrolla la investigación que lleva por título el presente trabajo. Su módulo de administración solo realiza funciones básicas, impidiendo a su administrador llevar un mejor control sobre los datos de la aplicación, los usuarios que la componen y las acciones que realizan. Tiene definido solamente cinco roles: Administrador, creador de Objetos de Aprendizaje, revisor de Objetos de Aprendizaje, creador de plantillas y revisor de plantillas, donde cada rol solo tiene una acción a cumplir. Por lo antes mencionado se hace necesario el estudio de otros conceptos como son administración, usuarios y roles.

1.2. Administración en aplicaciones web

La administración web se define como el respaldo del desarrollo web. La misma se sustenta gracias a los administradores que mantienen los servidores funcionando y realizando un seguimiento de los registros. Esta es una actividad fundamental para cualquier operación en Internet, su control es un complemento para conocer en su totalidad el flujo de información en cualquier aplicación web. Por lo general diferentes tipos de aplicaciones, principalmente los entornos web, necesitan la gestión o configuración de los usuarios, la gestión de proyectos, definición de servidores y bases de datos a acceder, definición de dominios, gestión de sesiones activas y estadísticas para el cómputo de carga y optimizar el tiempo de respuesta, buscador de objetos, herramientas colaborativas entre desarrolladores, control de acceso para cada proyecto. Tomando como punto de partida estas características se dice que el módulo administrativo de un sistema informático basado en la web se ocupa de llevar a cabo las funcionalidades antes mencionadas con el objetivo de refinar todo su funcionamiento (3).

Uno de los elementos que se manejan en la administración web es el término usuario, así como la asignación de roles y permisos para el desarrollo de las tareas en una aplicación determinada, fortaleciendo de esta forma la seguridad de la información.

Un **usuario** es la persona que utiliza un ordenador y realiza operaciones heterogéneas con diferentes fines. A menudo es un usuario aquel que adquiere una computadora o dispositivo electrónico y que lo emplea para comunicarse con otros usuarios, generar contenido y documentos, utilizar software de diversos tipos y muchas otras acciones posibles (4).

La definición de usuario es usada generalmente en el campo de las TIC para referirse al ente que utiliza determinado hardware y/o software, mediante el cual obtiene un servicio. Las cuentas de usuarios son creadas para brindar accesibilidad a distintas personas con respecto a una misma aplicación. De forma general la persona registrada es poseedora de su propia carpeta, permisos para ejecutar determinados programas y acceso limitado a determinada red informática (interna o externa) con todo lo que esto supone. (5). Con esto se favorece una gestión de usuarios de forma eficaz.

Las **cuentas de usuario** y de grupos permiten que los usuarios se integren a un dominio y tengan acceso a los recursos de este, en función de los derechos y permisos que tengan asignados. De ahí la importancia de planificar cuidadosamente la administración de estas cuentas (6).

Una parte importante de cualquier sistema de gestión es la seguridad de la información que se maneja, hay que garantizar esa seguridad a partir de la concesión de permisos para controlar qué usuarios deben tener acceso a parte o a toda la información y quiénes no. Se puede asignar permisos solo para acceder a la información o permisos para acceder a la información y modificarla (7).

A las personas registradas en la aplicación se le asignan roles con el fin de racionar las actividades en las aplicaciones. Un **rol** o **grupo** es una manera de asociar usuarios definiendo a qué partes del sistema tienen acceso (8). Es tan sencillo como vincular a varias personas en el cumplimiento de la operación.

La **gestión de roles** dispone de una visión conjunta y la identificación de los roles técnicos ya existentes en los sistemas. Esto puede facilitar la identificación de roles funcionales o irregularidades en las asignaciones, pero también facilitar la gestión de usuarios, puesto que

es más sencillo asignar a un nuevo ente registrado los privilegios ya existentes definidos con anterioridad, que ir asignándolos poco a poco a medida que los vaya solicitando (9). Actualmente se dispone de herramientas de gestión de usuarios, que hacen de la gestión de roles un complemento vital en el perfeccionamiento de perfiles de usuarios, permitiendo actualizar los roles existentes y mantener su claridad.

Muchos sistemas manejan el concepto trámite para hacer alusión a los procesos que se realizan con el fin de lograr un resultado. En estos sistemas el módulo de administración tiene la responsabilidad de gestionar los trámites para asignarlos a los usuarios del sistema, con el propósito de llevar el control de los mismos.

Un trámite es la fase, estado o diligencia de un proceso administrativo por el que pasa un asunto para ser solucionado (10).En este caso se denomina trámite a la confección de las cuentas de usuarios en el Módulo Administración a través del registro de las personas interesadas en realizar alguna actividad en la herramienta.

La administración de sesiones es otro elemento fundamental en la administración de sistemas. Muchas de las aplicaciones que se utilizan en la actualidad requieren autenticar a los usuarios antes de concederles el acceso a toda información importante; inicialmente, la forma de reconocer los privilegios de acceso ha sido mediante el establecimiento de sesiones de usuario. Saber quién está haciendo qué en cada momento, es uno de los principios de la seguridad informática moderna; es por eso que muchas de las aplicaciones que se usan regularmente requieren la autenticación de sus usuarios antes de permitirles el acceso a cierta información o a un conjunto de servicios. La forma habitual de llevar a cabo esta tarea de certificación del usuario es el establecimiento de una sesión la cual normalmente se mantiene vigente desde que el usuario presenta sus credenciales (nombre de usuario, contraseña, certificado digital, etc.) hasta que este cierra su sesión o la actividad cesa por un período de tiempo prolongado (11).

La **seguridad** de la aplicación es uno de los requerimientos más importantes que deben ser cumplidos. Está basada en las restricciones más comúnmente usadas en cualquier sistema que la efectúe, con el uso de los conceptos de autenticación[2] y autorización[3], basados en requerir una cuenta de usuario válida y activa, además de un sinnúmero de roles que le permitirán al usuario realizar aquellas acciones que se le han permitido. Mediante el uso del

[2] Acción de un usuario que le posibilita reconocerse en una aplicación
[3] Virtud que se le concede a un usuario para realizar distintas acciones en una aplicación

Módulo Administración, todo este reconocimiento es realizado, al iniciar sesión en el sistema. De esta forma el usuario solo podrá hacer lo asignado, facilitando la confiabilidad de la aplicación. En las primeras fases de análisis y diseño del proyecto se deben definir una serie de restricciones para el acceso a la base de datos y para el registro de las operaciones de los usuarios que interactúan con ella.

Ejemplos de aplicaciones que reflejan estos elementos y presentan un Módulo Administración son las plataformas Sistemas de Gestión de Contenidos (CMS), LMS, Sistemas de Gestión de Contenidos de Aprendizaje (LCMS) y los framework de desarrollo.

1.2.1. Sistemas de Gestión de Contenidos (CMS)

Los Sistemas de Gestión de Contenidos (CMS) según sus siglas en inglés son un software que facilitan la administración y creación de contenidos en páginas web casi siempre de forma automática. Los mismos permiten publicar, editar, asignar permisos, borrar y establecer módulos accesibles para el usuario final. El CMS lo conforman dos elementos fundamentales:

- CMA (La aplicación gestora de contenidos): posibilita al autor crear, modificar o eliminar el contenido del sitio sin tener conocimientos de HTML.

- CDA (La aplicación dispensadora de contenidos): permite compilar la información del sitio web (12).

Ejemplos de CMS son Drupal y Joomla, los cuales se explican a continuación.

Drupal[4]

Drupal es un sistema de gestión de contenido muy configurable, flexible y adaptable que posibilita publicar archivos, imágenes y otros servicios como blogs. Pertenece a la familia de software libre, escrito en PHP y bajo licencia GNU/GPL. Cuenta con una gran comunidad de desarrolladores permitiendo divulgar errores encontrados y solucionarlos, también cuenta con un módulo de administración de usuarios en el cual se asignan permisos de navegación a cada uno de ellos. Estos permisos se pueden definir en la parte del perfil dentro del núcleo de Drupal. Esta aplicación basa su desempeño en el concepto de rol asignándoles identificadores (a partir de ahora ID) a cada uno de ellos, los roles anónimos, es decir,

[4]http://drupal.org/

usuarios no registrados en la aplicación tendrán asignados por ID el cero (0), y los demás el ID uno (1), así permite asignar o quitar permisos a los roles creados por defecto en la aplicación según su necesidad. De esta forma los permisos se comparten por todos los usuarios que realizan la misma función en el sistema y estos permisos a su vez se incrementarán a medida que se instalen nuevos módulos en la aplicación.

Refiriéndose a la disponibilidad del contenido el núcleo[5] del sistema permite:

- Crear contenidos de cada tipo de dato del sistema.
- Ver el contenido publicado.
- Editar el contenido de los tipos de datos tanto el propio como el de otros usuarios.
- Borrar el contenido.

Otras de las facilidades que brinda Drupal, específicamente en la versión 7 son los permisos de campos (Field Permissions), el acceso al contenido (Access Content) y el control de acceso a taxonomías (Taxonomy Access Control):

- Permisos de campo: permite gestionar los permisos de visibilidad, edición, eliminado por niveles de campos de cada uno de los tipos de datos del sistema, como son, públicos, personalizados y privados.
- Acceso al contenido: permite elegir qué contenidos son los que se desea editar, ver, dar permisos y eliminar.
- Control de acceso a taxonomías[6]: Mediante términos de taxonomías se pueden editar, borrar y crear las distintas reglas de visibilidad del contenido que permiten una mejor gestión por taxonomías en la aplicación.

Para facilitar el desarrollo de sitios web flexible Drupal crea una estructura en capas de los contenidos siguiendo una serie de elementos básicos tales como: nodos, módulos, bloques y menús, plantillas y permisos de usuarios.

[5] Especifica las funciones básicas y la arquitectura con que debe adaptarse todo proyecto.
[6] Sistema utilizado en Drupal para clasificar contenido.

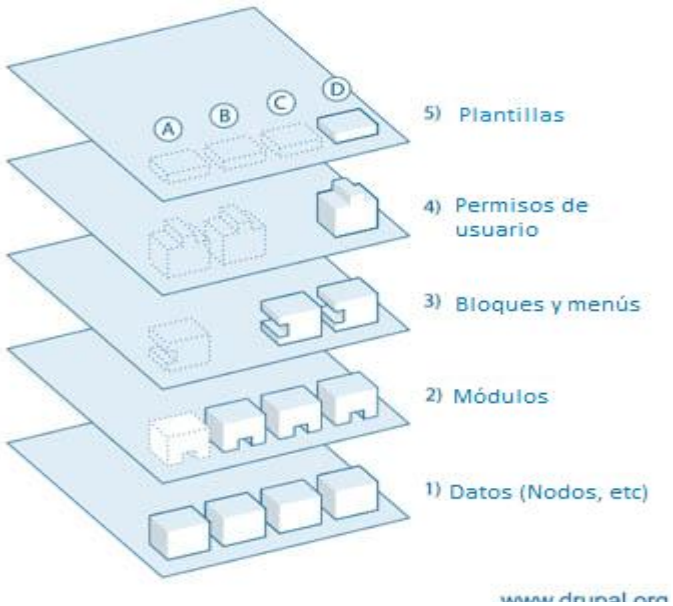

Figura 1 Arquitectura de Drupal.

Sin embargo, presenta algunas desventajas al utilizarse, como la definición del tipo de autenticación, ya que si no se define este elemento durante la instalación de la plataforma resulta complicado luego precisar en las opciones.

Por tanto esta opción no es factible utilizarla en la propuesta de solución, sin embargo resulta cómoda la manera de identificar los distintos grupos de usuarios predefinidos en la aplicación, siendo un elemento importante a tomar en cuenta para la elaboración de este trabajo.

Joomla[7]

Joomla es un CMS versátil y a la vez un framework para aplicaciones web que está disponible de forma gratuita y es de código abierto. Los usuarios en Joomla están divididos en dos grandes grupos, los que conforman la administración y los que conforman el sitio público de la aplicación.

En la administración de Joomla se encuentran los siguientes grupos de usuarios:

- Súper Administrador: Tiene acceso total a todo el sitio y a toda la administración.

[7]http://www.joomla.org/

- Administrador: Presenta casi los mismos permisos que el anterior; pero no puede acceder a las configuraciones globales, ni puede editar o eliminar al súper administrador.

- Manager: Tiene un acceso parcial, no puede acceder a la administración de componentes, a los módulos, a la configuración y a la edición de usuarios (13).

Estos usuarios administrativos también lo son del Frontend o sitio público de la aplicación y pueden administrar algunas secciones, aunque algunos tienen limitaciones para administrar esta parte del sitio.
En el Frontend de Joomla existen otros grupos de usuarios que pueden acceder solamente si han iniciado sesión, estos son:

- Registrados: Es un usuario común que solo puede acceder a las descargas y a cualquier recurso que previamente haya sido autorizado por el administrador para este tipo de usuario.

- Autor: Puede crear artículos; pero no puede editar los creados por él ni por los otros autores, además de no poder publicarlos, necesita de un administrador o un personal capacitado que lo autorice.

- Editor: Puede enviar artículos y editar sus envíos y el de los demás; pero al igual que el anterior rol no puede publicar, quedando sus escritos pendientes de validación.

- Publicador: Puede publicar, editar y enviar elementos (13).

El proceso de creación y administración de usuarios en Joomla sigue algunas normas. El súper administrador es el único que puede manipular usuarios, registrarlos, esto lo puede configurar mediante la opción "*Permitir registro de usuario*" en el panel de configuración; denegar o asignar permisos, bloquearlos, eliminarlos, cambiar contraseñas, incluso puede denegarse su acceso al sitio. Los usuarios pueden crearse y cambiar o actualizar sus datos y estados accediendo al Módulo Gestión de usuario.

Otros elementos que se pueden encontrar en el panel de configuración son:

- Gestión de artículos.

- Gestión de módulos.

- Gestión de plantillas.
- Gestión de menús.
- Gestión de configuración.
- Configuración Global.

Específicamente la “Configuración Global” permite cambiar el nombre del sitio, el tiempo de duración de una sesión activa, la dirección de la carpeta temporal del servidor, la configuración de PHP, el nombre de la base de datos, así como el usuario y contraseña para acceder a esta. Otro elemento gestionable son los permisos de los diferentes grupos de usuarios, la inclusión de módulos y plugins tanto de autenticación como de contenido.

Se está en presencia de una potente herramienta por sus particularidades y gracias a las ventajas mostradas en sus funcionalidades, como la seguridad establecidas a roles y usuarios, por lo que estas son tomadas en cuenta para incorporarlas al Módulo Administración de CRODA 2.0.

1.2.2. Sistemas de Gestión de Aprendizaje (LMS)

Los Sistemas de Gestión de Aprendizaje (LMS) son aplicaciones de software que ¨*permiten organizar materiales y actividades de formación en cursos, gestionar la matrícula de los estudiantes, hacer seguimiento de su proceso de aprendizaje, evaluarlos, comunicarse con ellos mediante foros de discusión, Chat o correo electrónico, etc., es decir, permite hacer todas aquellas funciones necesarias para gestionar cursos de formación a distancia (aunque pueden usarse como complemento en la enseñanza presencial)*¨ (14).

Ejemplo de LMS lo constituyen Moodle y Claroline, que brindan ventajas tanto a profesores, estudiantes y personal técnico de forma general, facilitando el proceso de enseñanza aprendizaje para unos y a su vez el control de las acciones realizadas en la aplicación para otros. A continuación se muestran de forma detallada.

Moodle[8]

Moodle (Entorno Modular de Aprendizaje Dinámico Orientado a Objetos) es un LMS que auxilia a los profesores en la creación de comunidades de aprendizaje en línea. Se distribuye como software libre bajo licencia GNU GPL. Está escrito en PHP. Cuenta con un módulo administración, el cual es controlado por un administrador que se define durante la instalación de la aplicación. Este administrador es el único que puede asignar o denegar permisos a otros usuarios y convertirlos a su vez en administradores del sitio. Existe además una gestión de roles donde son predefinidos los siguientes:

- Administrador: Gestiona toda la configuración del sitio.
- Creador de cursos: Permite asignar profesores y trabajar como profesor con privilegios de edición.
- Profesor: Tiene el control sobre determinado curso y de los alumnos inscritos en dicho curso.
- Profesor no editor: Tiene acceso de solo lectura a determinado curso y de los estudiantes inscritos en ese curso.
- Estudiante: Se matricula en uno o diversos cursos según sus necesidades.
- Invitado: Tienen acceso de solo lectura a la plataforma.

Accediendo al panel de edición y configuración de roles de la aplicación se pueden definir y controlar los permisos de todos los tipos de usuarios. Existen 4 tipos de permisos: ajustado, permitir, prevenir y prohibir, los cuales nos facilitan asignar roles, anular roles (solo el administrador puede realizar esta acción) y cambiar roles. No obstante, Moodle muestra ciertas deficiencias, si tratamos de elegir una versión para la instalación, por ejemplo la versión 1.9 requiere de muchos recursos con respecto al servidor de base de datos y el intérprete PHP.

[8]http://www.moodle.org/

Claroline[9]

Claroline fue desarrollado por el Dr. De Praetere en la Universidad Católica de Louvain y se ha convertido en uno de los LMS más estables de código abierto en la actualidad. Es multiplataforma, soporta diferentes idiomas, de fácil uso y sigue los estándares SCORM e Instructional Managment System (IMS)[10].

Claroline muestra tres roles fundamentales: El Administrador, el Profesor y el Estudiante; y un cuarto rol Tutor que es otorgado a un profesor para administrar ciertos foros en los cursos que imparte; pero no tiene permisos para realizar modificaciones a dicho curso.

Cuando el administrador inicia sesión en la aplicación se muestran diferentes funcionalidades que le permiten realizar su oficio en el sitio:

- Gestión de Campus: Posibilita gestionar funcionalidades y cursos, por ejemplo, crear nuevos profesores o administradores, consultarlos y a su vez hacer lo mismo con los cursos.
- Mejora de la Plataforma: Se puede hacer una actualización sobre la aplicación existente, siempre y cuando no se pierdan los cambios realizados en el sitio si se desarrolla algún componente.
- Server Information (información del servidor): Muestra información referente a todos los componentes del sitio como la versión de la aplicación, PHP y servidor Web.
- Estadísticas: Visualiza una serie de elementos como cantidad de usuarios y cursos.

Un profesor puede crear tantos cursos como desee y puede definir cuántos usuarios tienen acceso total o parcial. Puede habilitar o desactivar las herramientas disponibles para los cursos, entre estas herramientas se encuentran la agenda, los foros, documentos, grupos, trabajos y la opción **usuarios**. Esta última muestra un listado de todos los estudiantes, tutores y profesores que forman parte de un curso. Un aspecto en contra es el bajo impacto que muestra en la comunidad estudiantil, específicamente en la versión 1.4.

1.2.3. Sistemas de Gestión de Contenidos de Aprendizaje (LCMS)

Un LCMS o "*campus virtual*" es un software capaz de combinar las habilidades de crear cursos de un LMS con las capacidades de gestionar los contenidos de un CMS (12).

[9] http://www.claroline.net/
[10] http://www.imsglobal.org/

Posibilita crear de forma eficiente contenidos para el aprendizaje facilitando herramientas a autores y eruditos. Un ejemplo son ATutor y ZERA.

ATutor[11]

ATutor es un LCMS de código abierto basado en la web muy adaptable y accesible, consta de soporte a 32 idiomas, compuesto por herramientas de gerencia, autoría y colaboración. Incorpora especificaciones de empaquetamiento de contenido IMS/SCORM que posibilita a los diseñadores de contenidos crear elementos reutilizables. Puede ser administrado de forma sencilla, permitiendo gestionar estudiantes, profesores, evaluaciones y cursos.

Los roles presentes en ATutor son los siguientes:

- Administrador: Controla en su totalidad el desempeño del sitio y sus componentes
- Instructor: Crea los distintos cursos a impartir, además de editar los contenidos, los archivos referentes a dicho curso, gestionar tareas, exámenes y encuestas. También crea certificados de reconocimientos, acreditando los conocimientos adquiridos con el cumplimiento de los objetivos del curso.
- Estudiante: Tiene acceso a los disímiles cursos disponibles en el sitio.

Los usuarios en ATutor se crean en el panel de administración accediendo a la opción **Usuarios** donde muestra el listado general de los usuarios creados con sus roles asociados, a través de la función ¨Crear Cuenta de Usuario¨ muestra un formulario con los datos a llenar, entre ellos especificar si es un Estudiante o un Instructor. El conflicto de la patente del aprendizaje electrónico ATutor es un tema delicado, porque se revocaría su derecho a la patente de pizarra (15).

ZERA

ZERA es un LCMS creado en el centro FORTES de la Facultad 4 perteneciente a la Universidad de las Ciencias Informáticas. Su infraestructura está creada en el framework Symfony. Al igual que las demás aplicaciones anteriormente mencionadas ZERA consta de un Módulo Administración donde se maneja todo tipo de información como la gestión de usuarios, roles y permisos. En esta plataforma pueden crearse tantos usuarios como se deseen, los cuales serán asignados a distintos roles que a su vez se pueden crear teniendo

[11] http://www.atutor.ca

en cuenta las necesidades, aunque los básicos son los siguientes: Administrador central, Administrador local, Profesor, Estudiante, Tutor, Director, Editor, Auditor y Escritor de ayuda. Los roles tienen un nombre y una descripción y se componen de permisos que están integrados por acciones. Los roles y los permisos están compuestos por una tripleta *aplicación.módulo.componente*, por ejemplo: *backend.usuario.gestionar*. Este componente puede ser un permiso o una acción según sea el caso.

La política de esta plataforma es denegar todo, para mantener la seguridad, es decir, un usuario que accede a la aplicación por primera vez solo puede consultar el contenido de la página principal y sólo podrá realizar otras acciones si le asignan los permisos correspondientes.

1.3. Metodología y herramientas para el desarrollo del Módulo Administración

En la elaboración del análisis y diseño del módulo se adoptaron las herramientas y tecnologías seleccionadas por el proyecto CRODA y el centro FORTES para conservar la política de desarrollo en la presente migración a software libre, además este módulo será utilizado para el uso exclusivo de CRODA 2.0. A continuación se muestran dichas herramientas y tecnologías.

Metodología de desarrollo

Existen disímiles proposiciones de metodologías con una única meta, obtener mayor planificación, tiempo y un mejor costo, porque el desarrollo de software es una tarea que resulta sumamente complicada. Ejemplos de estas metodologías son:

XP[12] (eXtreme Programming)

Metodología de desarrollo clasificada dentro del grupo de metodologías de desarrollo ligeras, la misma tiene como principio básico mejorar la capacidad de intercambio de ideas entre usuarios y desarrolladores. Se basa en la rapidez de programación, además de contar en el equipo con el usuario final, facilitando de esta manera la existencia de sucesivas modificaciones en el desarrollo de la aplicación. Permite simplificar un diseño complejo para hacer más ágil el mantenimiento y el desarrollo (16).

[12]http://xprogramming.com y http://www.extremeprogramming.org,

Mientras que RUP intenta reducir la complejidad del software por medio de estructura y la preparación de las tareas pendientes en función de los objetivos de la fase y actividad actual, XP, como toda metodología ágil, lo intenta por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción (17).

XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (18).

RUP (Rational Unified Process)

El Proceso Unificado de Software (RUP) constituye una metodología de desarrollo pesada el cual está centrado en la arquitectura, es iterativo e incremental y está guiado por casos de usos. Sus elementos principales son los siguientes:

- Trabajadores (“quién”): Define las responsabilidades (rol) y el comportamiento de una persona, un grupo de personas, una máquina o sistema automatizado que trabajan en conjunto por equipos.
- Artefactos (“qué”): Elementos tangibles de un proyecto que son producidos, modificados y usados por las actividades. Pueden ser ejecutables, elementos dentro del modelo, modelos y códigos fuente.
- Actividades (“cómo”): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Flujo de Actividades (“cuándo”): Secuencia de actividades realizadas por los trabajadores y que produce un resultado de valor observable.

Las actividades se agrupan en grupos lógicos, en los cuales se definen 9 flujos de trabajo fundamentales, de los primeros 6 se consideran 3 como flujos de ingeniería y los 3 restantes como flujos de apoyo. Los flujos son: Modelado de Negocio, Requerimientos, Análisis y Diseño, Implementación, Pruebas, Despliegue, Configuración y Administración de Cambios, Administración de Proyecto y Entorno. Además, RUP está definido por 4 fases de desarrollo, las mismas son: Conceptualización (Concepción o Inicio), la cual describe el negocio y delimita el proyecto, se definen sus alcances con la identificación de los casos de uso del

sistema; Elaboración, donde se define la arquitectura que tendrá el sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen; Construcción, donde se obtiene un producto documentado y listo para su utilización, y por último la fase de Transición, en la cual el release queda listo para su instalación en condiciones reales (19)

Fundamento de selección de la metodología de desarrollo

Por las características del equipo de desarrollo y las particularidades del módulo a elaborar se decide utilizar la metodología de desarrollo de software RUP, porque genera gran documentación en la definición de las actividades realizadas por roles creando artefactos, propiciando un control de todo el proceso de avance del producto. Esto es un elemento vital para el adiestramiento del personal del proyecto que varía periódicamente, compuesto en su mayoría por estudiantes con poca práctica, posibilitando que no se pierda información que pueda ser utilizada en próximas versiones a desarrollarse de la herramienta. Además muestra una correcta guía para la especificación de requerimientos y utiliza como lenguaje de modelado a UML que es de uso viable y conocido por el equipo de desarrollo, permitiendo la fácil representación de modelos.

1.3.3. Lenguajes de Modelado

El lenguaje de modelado define la manera estándar en que intercambian criterios los miembros del equipo de desarrollo de software. Sirve de herramienta de desarrollo para realizar una representación gráfica simplificada de la realidad a través de abstracciones.

El Lenguaje Unificado de Modelado (UML) es considerado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Con el objetivo de cumplir sus funciones de ayuda a la construcción y documentación de sistemas ofrece una serie de ventajas tales como: un estándar para describir un “plano” del sistema (modelo), aspectos conceptuales como procesos de negocio y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables (19).

UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática (20).

Es importante destacar que UML es un lenguaje de modelado, no un método o un proceso, se emplea para definir, detallar y documentar los artefactos de un sistema de software. En la última versión se adicionaron diversas novedades que resuelven carencias desde el punto de vista práctico fundamentalmente. Entre los diagramas que propone UML para modelar un sistema se encuentran: Diagramas de estructura (clases, componentes, objetos, despliegue, paquetes). Diagramas de comportamiento (actividades, casos de uso, estado). Diagramas de interacción (secuencia, comunicación) (20).

Para el desarrollo del módulo se empleará UML 2.0 por todas las ventajas citadas anteriormente.

1.3.4. Herramienta CASE[13]

Una herramienta CASE (Computer Aided Software Engineering) nos permite elaborar disímiles artefactos para lograr la creación del producto sirviendo de guía una metodología de desarrollo de software. Permite generar parte del código del producto, sirve de soporte para el diseño, detección de errores y puede utilizarse en todo el proceso de desarrollo.

Visual Paradigm 6.4 para UML[14]

Visual Paradigm, herramienta de modelado que cuenta con una gama de ventajas, las cuales se nombran a continuación: es multiplataforma, cuenta con un entorno para la creación de diagramas para UML 2.0, utiliza un lenguaje común entre todos los desarrolladores para facilitar el intercambio de criterios y una completa sincronización durante todo el ciclo de desarrollo entre el modelo y el código. Además soporta aplicaciones web, varios idiomas, permite generar código en varios lenguajes de programación como C#, C++, PHP, es compatible con diferentes versiones y posibilita la creación de prototipos de interfaz de usuario.

Como se está en presencia de un software gratuito, con un entorno de trabajo amigable y con soporte de plantilla, estas características lo convierten en un programa ideal para la creación de cualquier aplicación, se utilizará el mismo en el desarrollo del módulo.

[13] http://case-tools.org/
[14] http://www.visual-paradigm.com

1.3.5. Tecnologías web

Los lenguajes de programación pueden ser divididos en dos grupos, los lenguajes del lado del cliente y los lenguajes del lado del servidor. Estos siguen la teoría de la arquitectura Cliente/Servidor para los entornos Web. Por la parte del cliente existen tecnologías como: Cascading Style Sheets (CCS), JavaScript, Hypertext Markup Language (HTML), que brindan dinamismo a la aplicación donde se utilizó y optimización en los canales de comunicación y del lado del servidor PHP (Procesador de hipertexto).

HTML (Hypertext Markup Language)

Lenguaje de marcado que predomina en la confección de páginas web, definiendo su sintaxis. Se utiliza para describir la estructura y el contenido, y presentarlo en forma de hipertexto que es el formato de las páginas web. Permite publicar documentos, fotos, imágenes y videos (21).

CSS 2.0 (Cascading Style Sheets)

Las hojas de estilo en cascada definen la estética del documento confeccionado en HTML. Separando la estructura del contenido del documento se llega a la idea fundamental del desarrollo de CSS.
Algunas ventajas del uso de CSS son:

- Control centralizado de la presentación del sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o, incluso, a elección del usuario.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño (22).

JavaScript 1.5

JavaScript es un lenguaje de programación muy potente orientado a objetos. Este lenguaje permite interactuar con HTML facilitándole a los programadores usar contenido dinámico, como crear scripts mediante los cuales se ejecuten acciones que den respuestas a los eventos solicitados por el usuario. Además permite incorporarles efectos a los diferentes componentes visuales de la página.

PHP 5.3.5[15] (Procesador de hipertexto)

Lenguaje multiplataforma del lado del servidor que facilita el desarrollo de aplicaciones web dinámicas, con una gran cantidad de documentación en su página web oficial, además de traer incluido ejemplos en su archivo de ayuda. Muestra una gran capacidad de conexión con los distintos servidores web como son MySQL, PostgreSQL. PHP es de código abierto, facilitando a los desarrolladores encontrar errores, repararlos y perfeccionarlos (23).

1.3.6. Servidor de aplicaciones web

Un servidor web representa un software creado para transmitir hipertextos, además de mantenerse en espera de una solicitud de procesadores clientes mediante un navegador web, donde se reciben las respuestas a esta petición a través de páginas web que se localizan en dicho navegador.

Apache 2.2.17

Servidor web libre open source, que gracias a su estabilidad permite que cada día se sumen más adeptos a su uso. Su licencia no es GPL; pero desciende de las licencias BSD, Berkeley Software Distribution según sus siglas en inglés, que permite configurar el código fuente a gusto siempre y cuando se reconozca el trabajo. Dentro de sus características fundamentales se encuentran:

- Es multiplataforma.
- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Es altamente configurable en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo puede tener un mayor control sobre lo que sucede en el servidor (24).

Se propone utilizar Apache en la solución del problema a resolver, ya que se está en presencia de un software gratuito, altamente configurable y es el servidor web por excelencia en la actualidad.

[15] http://www.php.net/

1.3.7. Sistema gestor de bases de datos

Un sistema gestor de bases de datos (SGBD) es un software que cumple la función de intermediario entre el usuario, la aplicación y la base de datos. Con el uso de estos gestores se logra seguridad en los datos, fácil manipulación de la información presente en las bases de datos y se elimina la redundancia.

PostgreSQL 8.4

PostgreSQL es un sistema de gestión de bases de datos relacional, orientado a objetos, de código abierto (open source), multiplataforma y cuenta con una extensa documentación, además de ejemplos. Dentro de sus potencialidades se encuentran además:

- Instalación sin límites: nadie puede ser víctima de una demanda por violar acuerdos de licencia, ya que no existe un costo asociado a la licencia del software.
- Soporte: existe una gran comunidad de expertos y entidades que brindan soporte a este gestor, de estas el Grupo Global de Desarrollo de PostgreSQL se beneficia y contribuye en su mejoramiento (25)**.**

Por todas las ventajas y las características que brinda PostgreSQL se propone como sistema gestor de base de datos a utilizar en el tratamiento de todos los datos de la aplicación para el desarrollo de este módulo y su incorporación a CRODA.

Exist -db 1.4

Para solucionar el problema relacionado con la manipulación de los Objetos de Aprendizaje y los metadatos que los integran, se eligió un sistema gestor de bases de datos idóneo para lograr la obtención del contenido de los archivos XML, como es el caso del gestor Exist -db. Este es un SGBD de código abierto y libre. Haciendo uso de tecnología XML, almacena contenido en un archivo XML del mismo formato de acuerdo a su modelo de datos. Además soporta distintos lenguajes de consulta XML como Xpath y Xquery, permite indexar documentos y dar soporte para actualizar los datos.

1.3.8. Framework de desarrollo

Los framework surgen con la idea de optimizar y hacer más fácil la creación de aplicaciones web. Están compuestos por elementos configurables, pueden incluir bibliotecas y un lenguaje interpretado para llevar a fin un proyecto. De manera general son componentes

que mediante su diseño reutilizable agilizan el desarrollo del software en el ámbito web. Existen varios framework de desarrollo en la actualidad, entre los más utilizados se encuentran: CodeIgniter y Symfony.

CodeIgniter

CodeIgniter (CI) es una aplicación web desarrollada en PHP para la creación de cualquier tipo de aplicación web bajo este lenguaje. Es un producto de código abierto, para cualquier aplicación es también libre de uso. Contiene una serie de librerías que sirven para el desarrollo de aplicaciones web y además propone una manera de desarrollarlas que debemos seguir para obtener provecho de la aplicación.

Implementa el proceso de desarrollo llamado Model View Controller (MVC), que es un estándar de programación de aplicaciones, utilizado tanto para hacer sitios web como programas tradicionales. Contiene muchas ayudas para la creación de aplicaciones PHP avanzadas, que hacen que el proceso de desarrollo sea más rápido. Define una arquitectura de desarrollo que permite programar de una manera más ordenada y contiene diversas herramientas que ayudan a hacer aplicaciones más versátiles y seguras (26).

Este framework tiene una forma clara para atender las solicitudes de los usuarios, mediante la colaboración de varios módulos como la caché y el enrutamiento de la petición.

Figura 2 Flujo de la aplicación CodeIgniter.

Un elemento importante en una aplicación web es la autenticación de los usuarios. Una deficiencia de CI es que no presenta una por defecto, haciéndose necesario la implementación de uno a través de la confección de clases controladoras y entidades, donde se especificarán los parámetros a guardar y la asociación de los diferentes roles. Por tanto no es recomendable utilizar este framework en el desarrollo del trabajo, ya que se incrementaría el trabajo del equipo de desarrollo.

Symfony 1.4.17

Symfony es un framework que facilita varias herramientas y que gracias a sus disímiles características posibilita optimizar el desarrollo de aplicaciones web complejas. Está implementado en su totalidad con PHP 5. Es compatible con la inmensa mayoría de los gestores de bases de datos como son PostgreSQL y MySQL, puede desplegarse en distintas plataformas tanto de UNIX como de Windows, consta con URLs amigables e incluye por defecto ataques contra XSS[16] y CSRF[17].

El Mapeo de Objeto Relacional (ORM) es una técnica de programación utilizada para convertir datos entre sistemas de tipos que utilizan lenguajes de programación orientados a objetos y el usado por una base de datos relacional (27). El ORM que emplea el framework Symfony por defecto es Propel aunque esto es configurable y puede usarse otro como es el caso de Doctrine, en el caso específico de CRODA es Doctrine el que se utiliza.

Se propone utilizar Symfony por su usabilidad, compatibilidad con los distintos sistemas gestores de bases de datos, por haberse publicado bajo licencia de software libre y para no perder la arquitectura de la herramienta de autor web CRODA que ha sido desarrollada sobre este framework de desarrollo y la familiaridad. Además de permitir integrar plugin como sfGuard.

sfGuardPlugin

El plugin sfGuard brinda autenticación y características de autorizo por encima de las características comunes de seguridad de este framework de desarrollo. Permite administrar usuarios, roles y permisos, además de configurar de forma flexible los módulos backend y frontend de la aplicación (27).

Para lograr ofrecer estas características basta con instalarlo y habilitarlo mediante unas simples líneas de código en una consola de comandos. La inmediata instalación propicia las siguientes clases entidades:

- sfGuardUser: para manejar los usuarios.
- sfGuardGroup: para manipular los roles.

[16]XSS (Cross-site scripting): es un tipo de agujero de seguridad basado en la explotación de vulnerabilidades del sistema de validación de HTML incrustado.
[17]CSRF (Cross Site Request Forgery): es un tipo de agujero de seguridad basado en explotar la confianza que los sitios web tienen con sus usuarios.

- sfGuardPermission: para manejar los permisos.
- sfGuardAuth: para gestionar el servicio de autenticación.

Estas clases promueven el trabajo con formularios favoreciendo la creación de listas de control de acceso (ACL) a la aplicación mediante la gestión de usuarios, roles o grupos y los permisos asociados a estos grupos de usuarios. Además facilita la extensión de sus funcionalidades, logrando una aplicación íntegra y confiable.

Por todas las ventajas expuestas anteriormente y potencialidades que manifiesta se escoge el plugin sfGuard para el desarrollo del Módulo Administración, ya que propiciará un mejor control de los usuarios, los roles y permisos correspondientes.

ExtJs 3.4

ExtJs es una librería de JavaScript que posibilita crear aplicaciones web enriquecidas usando tecnologías AJAX y DOM. Desde la versión 1.1 puede ejecutarse como una aplicación independiente. Fue creado por Jack Slocum originalmente como extensión para Yahoo User Interface (YUI), en la actualidad puede usarse como extensión para las bibliotecas jQuery y Prototype (28).
Sus características principales son:

- Componentes de interfaz de usuario personalizables.
- Buenos diseños.
- Es intuitivo.
- API extensa y fácil de utilizar.
- Se distribuye bajo licencias Open Source y comerciales.
- Gran desempeño.
- Compatible con los diferentes navegadores web.

1.4. Conclusiones

En este capítulo el estudio del estado del arte sobre el proceso de administración en aplicaciones web permitió definir los principales conceptos necesarios para el desarrollo de este trabajo. El estudio de sistemas similares permitió concluir que la herramienta de autor CRODA carece de los elementos fundamentales para la administración, además este estudio posibilitó obtener elementos necesarios a incorporar a esta nueva versión de la

herramienta de autor CRODA. Se seleccionaron como metodología de desarrollo de software RUP, como lenguaje de modelado UML y Visual Paradigm como herramienta CASE debido a las ventajas que estas proporcionan. Se propuso además un conjunto de tecnologías y herramientas para implementar el módulo en un futuro.

Capítulo 2: Características del sistema

Introducción

En el presente capítulo se define la propuesta de solución, se describen los diferentes procesos que se tratan en el objeto de estudio. Se especifican los requisitos funcionales y no funcionales que debe cumplir el sistema y se identifican los casos de usos con sus respectivos actores mediante el esbozo del Diagrama de Casos de Uso.

2.1. Modelo de Dominio. ¿Por qué?

Haciendo un análisis exhaustivo del problema en cuestión se llega a la conclusión de que el negocio del presente trabajo, tiene un bajo nivel de estructuración, donde no se definen concretamente los procesos del mismo, de ahí que se decide dar un nuevo enfoque a todo el proceso, para ello se utilizará un modelo del dominio, ya que permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo.

El modelado de dominio tiene como objetivo facilitar la comprensión de los requisitos. Este modelo debe contribuir a mejorar el entendimiento del contexto en que se basa el sistema, es un artefacto perteneciente al análisis, erigido en las reglas de UML y esboza los conceptos más importantes del contexto como objetos del dominio, ya que los une entre sí. Según Iván Garcerant**: "**Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea software u otro tipo**"** (29)**.**

Los modelos de dominio o modelos conceptuales muestran una representación visual estática de la realidad en la que se desarrollará la aplicación. Construye un diagrama con los objetos reales que se relacionan con el proyecto y sus correspondencias. Cada uno de esos objetos un nombre que lo identifica que contribuye al empleo de un lenguaje sencillo que posibilita un intercambio fluido entre el equipo de desarrollo de la aplicación.

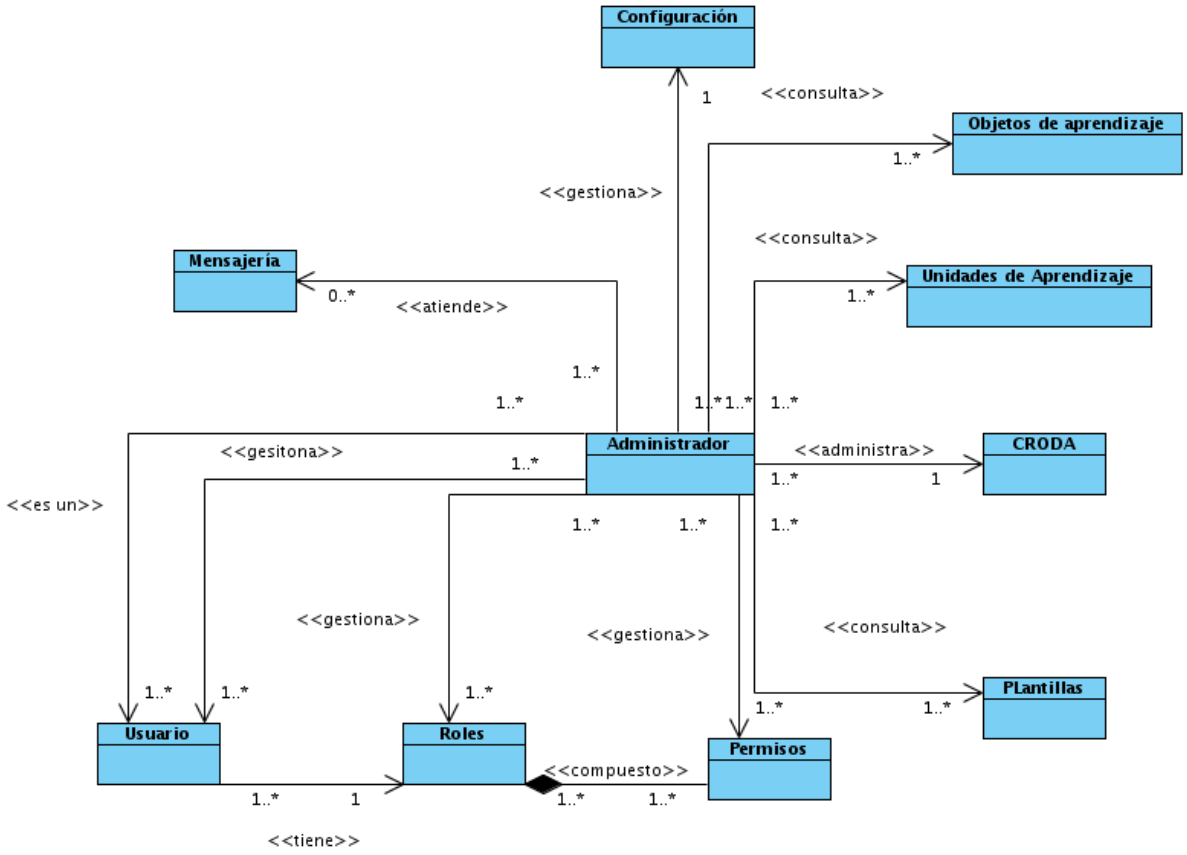

Figura 3 Modelo de dominio.

Descripción de los conceptos del modelo de dominio:

Administrador: Persona que dispone de toda las funciones administrativas de la herramienta.

CRODA: Herramienta de autor web que tiene como objetivo crear Objetos de Aprendizaje, las plantillas para la descripción de los Objetos de Aprendizaje y Unidades de Aprendizaje.

Usuario: Cualquier ente que interactúe con la herramienta de autor.

Roles: Grupos de funciones que tienen permitidas los usuarios.

Permisos: Funciones que componen un rol predeterminado.

Unidades de Aprendizaje: Diseños que reflejan cómo deben desarrollarse las acciones cognoscitivas de una actividad educativa.

Objetos de Aprendizaje: recurso con una intención formativa, compuesto de uno o varios elementos digitales, descrito con metadatos, que pueda ser utilizado y reutilizado para apoyar el aprendizaje.

Plantillas: Estructura básica que contiene los Objetos de Aprendizaje para su descripción

Configuración: Elementos que permiten el tipo de acceso a la aplicación y otras opciones como la cantidad máxima de estructuras en edición.
Mensajería: Servicio que brinda la herramienta para facilitar el contacto interno de la aplicación entre administradores y usuarios no registrados en la herramienta de autor web CRODA.

2.1.1. Descripción del diagrama de clases del modelo de dominio

El administrador es un rol asignado a un usuario registrado en la herramienta de autor web CRODA, que a la vez gestiona usuarios, roles, permisos, y la configuración de la aplicación. Además de realizar varias consultas a las plantillas, Objetos de Aprendizaje y Unidades Instruccionales, para verificar su estado o bien eliminarlos si presentan algún error. A su vez los roles van a estar compuesto por uno o varios permisos según sean necesarios. Por último el administrador atiende las inquietudes de los usuarios no registrados en la herramienta mediante un servicio de mensajería, facilitando la comunicación interna del sistema.

2.2. Requerimientos del sistema

La IEEE Standard Glossary of Engineering Terminology, define el término requerimiento como:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

Es importante no perder de vista que un requerimiento debe ser:

- Especificado por escrito: Como todo contrato o acuerdo entre dos partes.
- Posible de probar o verificar. Si un requerimiento no se puede comprobar, entonces ¿cómo se sabe si se cumplió con él o no?
- Conciso: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- Completo: Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.

- Consistente: Un requerimiento es consistente si no es contradictorio con otro requerimiento.
- No ambiguo: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

Los requerimientos pueden dividirse en requerimientos funcionales y requerimientos no funcionales. Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares.

Requerimientos funcionales.

Tras realizar un análisis de las limitaciones de la herramienta en la versión 1.0 se determinaron los siguientes requisitos funcionales.

RF 1. Adicionar usuarios. El sistema debe permitir adicionar usuarios a la aplicación.
RF 2. Editar usuarios. El sistema debe permitir editar usuarios en la aplicación.
RF 3. Eliminar usuarios. El sistema debe permitir eliminar usuarios de la aplicación.
RF 4. Consultar los usuarios. El sistema debe permitir consultar los usuarios en la aplicación.
RF 5. Adicionar roles. El sistema debe permitir adicionar roles a la aplicación.
RF 6. Editar roles. El sistema debe permitir editar los roles en la aplicación.
RF 7. Eliminar roles. El sistema debe permitir eliminar los roles de la aplicación.
RF 8. Consultar roles. El sistema debe permitir consultar los roles en la aplicación.
RF 9. Asociar permisos a roles. El sistema debe permitir asociar los permisos predefinidos en la aplicación a cualquier rol creado.
RF 10. Consultar los Objetos de Aprendizaje en edición. El sistema debe permitir consultar los Objetos de Aprendizaje que los usuarios tienen en edición.
RF 11. Eliminar los Objetos de Aprendizaje que tengan los usuarios en edición. El sistema debe permitir eliminar los Objetos de Aprendizaje que los usuarios tengan en edición.
RF 12. Mostrar la cantidad de Objetos de Aprendizaje que han sido publicados desde CRODA por colecciones y autor.
RF 13. Consultar Unidades de Aprendizaje en edición. El sistema debe permitir consultar las

Unidades de Aprendizaje que los usuarios tengan en edición.

RF 14. Eliminar Unidades de Aprendizaje que tengan los usuarios en edición. El sistema debe permitir eliminar las Unidades de Aprendizaje que los usuarios tengan en edición.

RF 15. Configurar el tipo de protocolo de acceso. El sistema debe permitir configurar el tipo de protocolo de acceso a la aplicación ya sea por protocolo HTTP o HTTPS.

RF 16. Configurar el tipo de autenticación. El sistema debe permitir configurar el tipo de autenticación en la aplicación ya sea de tipo manual o LDAP.

RF 17. Configurar el máximo número de plantillas en edición. El sistema debe permitir configurar el máximo número de plantillas en edición presentes en la aplicación.

RF 18. Configurar el máximo número de Objetos de Aprendizaje en edición. El sistema debe permitir configurar el máximo número de Objetos de Aprendizaje en edición presentes en la aplicación.

RF 19. Configurar el máximo de Unidades de Aprendizaje en edición. El sistema debe permitir configurar el máximo número de Unidades de Aprendizaje en edición presentes en la aplicación.

RF 20. Recibir mensajes de parte de los usuarios no registrados en la herramienta. El sistema debe permitir recibir mensajes de parte de los usuarios no registrados en la herramienta.

RF 21. Responder mensajes de parte de los usuarios no registrados en la herramienta. El sistema debe permitir responder mensajes de parte de los usuarios no registrados en la herramienta.

RF 22. Eliminar mensajes de parte de los usuarios no registrados en la herramienta. El sistema debe permitir eliminar los mensajes de parte de los usuarios no registrados en la herramienta.

RF 23. Consultar las plantillas que se encuentran en edición, en revisión y que han sido publicadas. El sistema debe permitir consultar las plantillas que se encuentran en edición, en revisión y que han sido publicadas en la herramienta.

RF 24. Eliminar las plantillas que se encuentran en edición, en revisión y que han sido publicadas. El sistema debe permitir eliminar las plantillas que se encuentran en edición, en revisión y que han sido publicadas en la herramienta.

Requerimientos no funcionales

Los requisitos no funcionales son las propiedades con que debe contar el sistema. Es vital definirlos para lograr un sistema agradable, seguro y confiable (30). Entre los requisitos no funcionales se encuentran:

Usabilidad

RNF1. El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.
RNF2. Debe presentar una interfaz especial para usuarios no expertos en informática muy amigable y fácil de trabajar y que pueda ser usada por los usuarios fácilmente.
RNF3. En la interacción con el software el sistema mostrará mensajes que sean de entendimiento común con el usuario, para de esta forma elevar la comunicación con el mismo.
RNF4. En los campos que sean complejos se mostrarán notas indicando ejemplos concretos para guiar el proceso.

Fiabilidad

RNF5. Una vez publicado el sistema, deberá estar disponible para los usuarios en cualquier momento.
RNF6. Las salidas que ofrece el sistema como resultado del manejo y procesamiento de la información estarán adecuadamente validadas durante los diferentes hitos de prueba, siendo estas fidedignas y con total exactitud a los valores esperados.

Eficiencia

RNF7. Tiempos de respuestas rápidas como máximo 10 segundos al igual que la velocidad de procesamiento de la información.
RNF8. El sistema funcionará de manera óptima en los navegadores Web Mozilla Firefox, Internet Explorer 6 o superior y Opera.

- **Hardware**

RNF9. La PC del servidor debe de tener al menos 1GB de RAM.
RNF10. La PC del servidor debe ser Pentium 4 o superior.

RNF11. La capacidad de almacenamiento del disco duro del servidor debe ser como mínimo 120GB.
RNF12. La PC cliente debe tener como mínimo 80GB de capacidad de almacenamiento.
RNF13. La PC cliente debe tener al menos 1GB de RAM.
RNF14. La PC cliente debe ser Pentium 4 o superior.

- **Software**

RNF15. Se utilizará PostgreSQL versión 8.4 como gestor de base de datos.
RNF16. Se utilizará Exist-db versión 1.4 como base de datos de XML nativo.

Soporte

RNF17. El soporte del sistema estará a cargo del Dpto. Instalación y Soporte del centro FORTES.

Accesibilidad

RNF18. El sistema debe permitir aumentar y disminuir el tamaño de la letra.
RNF19. Las imágenes presentes en el sistema deben poseer una descripción.
RNF20. Las interfaces deben tener vínculos entre ellas para facilitar el acceso.

Restricciones de diseño

RNF21. Para el modelado visual se deberá usar Visual Paradigm 6.4 como herramienta CASE.
RNF22. Se utilizará para la programación el lenguaje PHP 5 o superior.
RNF23. Se hará uso de Netbeans 6.9 como entorno de desarrollo.

Interfaz de software

RNF24. El sistema debe interactuar con el repositorio de Objetos de Aprendizaje RHODA. Para ello se ofrecerá una interfaz de comunicación haciendo uso del estándar SQI e IMS-DRI. Dicha interfaz contará con la vía para acceder a las funcionalidades o información que forman parte de la integración.

Interfaces de usuario

RNF25. El sistema debe ofrecer una interfaz fácil de operar. Igualmente tiene que mantener la línea de diseño de CRODA 1.0.

Interfaces de Comunicación

RNF26. El sistema debe poseer una interfaz de comunicación haciendo uso del estándar SQI e IMS-DRI.

Ayuda y documentación en línea

RNF27. El sistema debe contener un manual de usuario que les permita a sus usuarios documentarse para el trabajo con el mismo.

Requisitos Legales, de Derecho de Autor, Normas o estándares aplicables al sistema, por ejemplo estándares legales, de calidad, normas de usabilidad, otros.

- La plataforma escogida para el desarrollo de la aplicación está basada en la licencia GNU/GPL.
- Para lograr la usabilidad de la herramienta de autor se empleará la norma ISO 9241-11.
- Para lograr la interoperabilidad tanto de los contenidos creados a partir de la herramienta como la interoperabilidad de la propia herramienta con otros sistemas se utilizarán los estándares SCORM y QTI; e IMS-DRI y SQI respectivamente.
- Para la descripción de los contenidos se utilizará el estándar LOM.
- Para el derecho de autor se implementará de forma obligatoria el llenado de la subcategoría Contribución en la categoría Ciclo de vida.

2.3. Diagrama de los casos de uso identificados en el sistema

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea (31). A continuación se muestra el diagrama de casos de uso del sistema correspondiente.

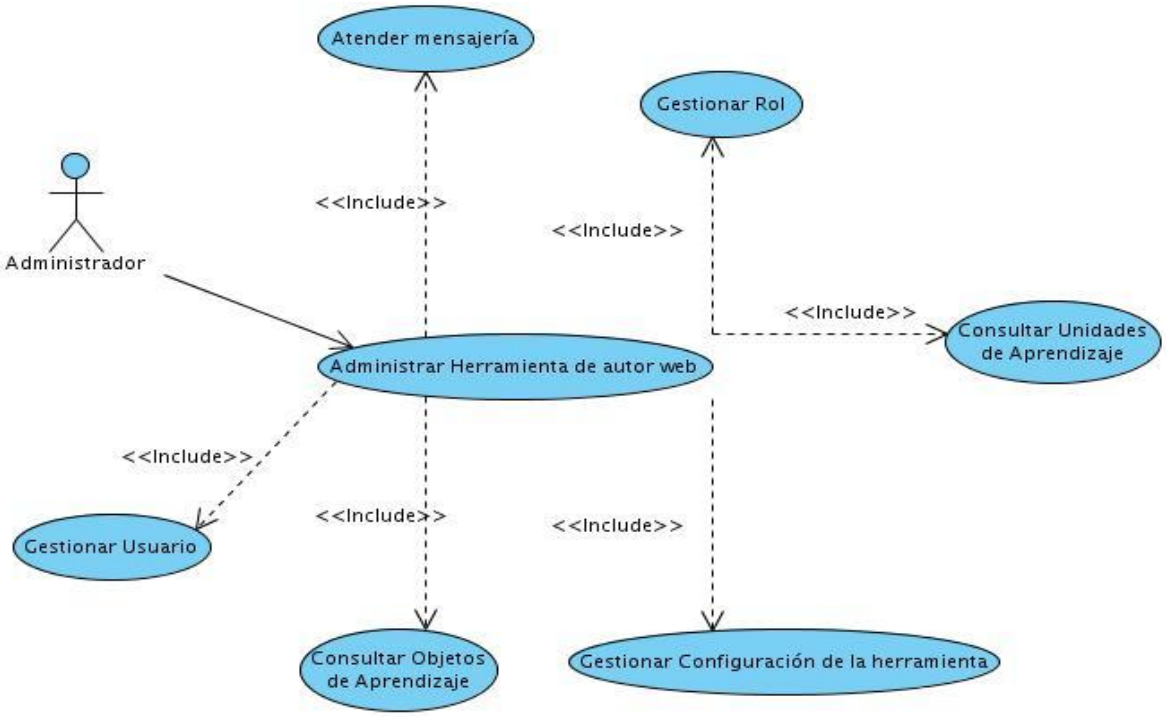

Figura 4 Diagrama de casos de uso del sistema.

2.4. Identificación de actores del sistema

El término actor significa el rol que algo o alguien juega cuando interactúa con el sistema. Un candidato a actor del sistema es cualquier individuo, grupo, organización o máquina que interactúa con los casos de uso. De acuerdo con esta idea un actor del sistema representa un tipo particular de usuario del sistema más que un usuario físico, ya que varios usuarios físicos pueden realizar el mismo papel en relación al negocio, o sea, ser instancias de un mismo actor. Una vez que se han identificado los actores del sistema, se tiene identificado el entorno externo del sistema. La siguiente tabla muestra la relación actor-módulo.

Tabla 1 Relación actor-módulo.

Actor	Descripción
Administrador	Es el rol encargado de realizar las funciones administrativas de la herramienta. Especifica la configuración del sistema, mediante la creación, edición y supresión de roles, permisos, usuarios.

2.5. Descripción de los casos de uso identificados

Mediante la descripción de los casos de usos que se identificaron, se les facilitará a los programadores una visión más clara de la interacción entre los usuarios y el sistema. Cada caso de uso viene acompañado de un diseño de prototipo de interfaz de usuario (Ver Anexo 2), facilitando un canal de intercambio efectivo entre el usuario y la aplicación. A continuación se presentan la descripción de los casos de uso críticos del sistema. La descripción de los restantes casos de uso se puede encontrar a partir del Anexo 3.

2.5.1. Administrar Herramienta de Autor Web

Tabla 2 Caso de uso Administrar Herramienta de Autor Web.

Caso de Uso:	Administrar Herramienta de Autor Web
Actores:	Administrador
Resumen:	El caso de uso habilita la administración de la herramienta.
Precondiciones:	1- El actor debe estar autenticado en la herramienta de autor CRODA.
Referencias	
Prioridad	**Crítico**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El actor selecciona la opción “Administración”	2 El sistema muestra una interfaz con las opciones: Usuarios.(ir a sección Usuarios) Roles.(ir a sección Roles) Configuración. (Ir a sección Configuración). Mensajería. (Ir a sección Mensajería) Consultar. Desplegando esta pestaña se observa los siguientes elementos: Plantillas. (Ir a la sección Plantillas). Objetos de Aprendizaje. (Ir a la sección Objetos de Aprendizaje). Unidades de Aprendizaje. (Ir a la sección Unidades de Aprendizaje).

Sección “Usuarios”	
Acción del Actor	**Respuesta del Sistema**
1 El actor selecciona la opción ¨Usuarios¨	1.1 El sistema muestra una interfaz que permite: (Ver CU incluido – Gestionar Usuario.)
Flujos Alternos	
Acción del Actor	***Respuesta del Sistema***
Sección “Roles”	
Acción del Actor	**Respuesta del Sistema**
1 El actor selecciona la opción ¨Roles¨	1.1 El sistema muestra una interfaz que permite: (Ver CU incluido – Gestionar Roles.)
Flujos Alternos	
Acción del Actor	***Respuesta del Sistema***
Sección “Configuración”	
Acción del Actor	**Respuesta del Sistema**
1 El actor selecciona la opción ¨ Configuración¨	1.1 El sistema muestra una interfaz que permite: (Ver CU incluido – Gestionar Configuración de la herramienta.)
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Sección “Mensajería”	
Acción del Actor	**Respuesta del Sistema**

1 El actor selecciona la opción ¨ Mensajería¨	1.1 El sistema muestra una interfaz que permite: (Ver CU incluido – Atender mensajería.)
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Sección "Plantillas"	
Acción del Actor	**Respuesta del Sistema**
1 El actor selecciona la opción ¨ Plantillas¨	1.1 El sistema muestra una interfaz que permite: (Ver CU incluido – Consultar Plantillas.)
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Sección "Objetos de Aprendizaje"	
Acción del Actor	**Respuesta del Sistema**
1 El actor selecciona la opción ¨ Objetos de Aprendizaje ¨	1.1 El sistema muestra una interfaz que permite: (Ver CU incluido – Consultar Objetos de Aprendizaje.)
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Sección "Unidades de Aprendizaje"	
Acción del Actor	**Respuesta del Sistema**
1 El actor selecciona la opción ¨ Unidades de Aprendizaje ¨	1.1 El sistema muestra una interfaz que permite:

	(Ver CU incluido – Consultar Unidades de Aprendizaje.)
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones	

2.5.2. Gestionar Usuarios

Tabla 3 Gestionar Usuarios.

Caso de Uso:	Gestionar Usuarios
Actores:	Administrador
Resumen:	El CU inicia cuando el administrador decide crear, modificar o eliminar un usuario de la herramienta. Finaliza cuando una de estas acciones sobre el usuario es realizada.
Precondiciones:	El usuario debe estar autenticado en la aplicación.
Referencias	RF1, RF 2, RF 3, RF 4
Prioridad	**Crítico**
Flujo Normal de Eventos	
Sección 1 "Adicionar Usuarios"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción "Usuarios".	1.1 El sistema muestra un listado de los usuarios de la herramienta además de las opciones "Adicionar, Editar y Eliminar".
2. El administrador selecciona la opción "Adicionar" usuario.	2.1 El sistema muestra el formulario correspondiente, con los campos que son necesarios. Tales como: • Nombre • Apellido • Correo electrónico

	• Usuario • Contraseña • Repetir contraseña • Ocupación • País • Rol • Activo
3. El administrador ingresa los datos necesarios y presiona el botón “Aceptar”.	3.1 El sistema adiciona el usuario mostrando un mensaje satisfactorio y regresa al listado inicial con los cambios realizados.
***Prototipo de Interfaz* Sección “Usuario”** Ver Anexo 2.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El administrador deja un campo vacío en el formulario a llenar	2.1 El sistema muestra un mensaje de error: “Existen campos obligatorios vacíos”.
Prototipo de Interfaz	
Poscondiciones	
Sección 2 “Editar usuario”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona el usuario a editar y presiona el botón “Editar”.	1.1 El sistema muestra un mensaje de confirmación: “Desea realmente editar el usuario”
2. El administrador acepta el mensaje de confirmación.	2.1 El sistema muestra el formulario correspondiente con los datos del usuario para efectuar la modificación.
3. El administrador realiza los cambios necesarios.	3.1 El sistema guarda los cambios realizados, envía un mensaje al administrador del éxito de la operación y regresa al listado inicial.
Prototipo de Interfaz	

Ver Anexo 2	
Flujos Alternos de la Sección 2	
Acción del Actor	**Respuesta del Sistema**
2. El administrador no acepta el mensaje de confirmación	2.1 El sistema no muestra el formulario para la edición del usuario.
3. El administrador deja un campo vacío en el formulario a editar.	3.1 El sistema muestra un mensaje de error: “Existen campos obligatorios vacíos”.
Prototipo de Interfaz	
Poscondiciones	
Sección 3 “Eliminar usuario”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona un usuario de la lista y seguido selecciona la opción “Eliminar”.	1.1 El sistema muestra un mensaje al administrador de confirmación para eliminar el usuario seleccionado.
2. El administrador acepta el mensaje de confirmación.	2.1 El sistema elimina el usuario seleccionado, muestra un mensaje satisfactorio referente a la operación y regresa al listado inicial.
Prototipo de Interfaz Ver Anexo 2	
Flujos Alternos de la Sección 3	
Acción del Actor	**Respuesta del Sistema**
2. El administrador no acepta el mensaje de confirmación.	2.1 El sistema no elimina el usuario seleccionado y regresa al listado inicial.
Prototipo de Interfaz	
Poscondiciones	

2.5.3. Gestionar Roles

Tabla 4 Gestionar Roles.

Caso de Uso:	Gestionar Roles
Actores:	Administrador
Resumen:	El CU inicia cuando el administrador decide crear, editar o eliminar

<table>
<tr><td></td><td colspan="2">roles. Finaliza cuando una de estas acciones es ejecutada.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">El usuario debe estar autenticado en la aplicación.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF 5, RF 6, RF 7, RF 8, RF 9</td></tr>
<tr><td>Prioridad</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección 1 “Adicionar rol”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El administrador selecciona la opción “Roles”.</td><td>1.1 El sistema muestra un listado con los distintos roles creados que conforman la herramienta.</td></tr>
<tr><td colspan="2">2. El administrador selecciona la opción “Adicionar”.</td><td>2.1 El sistema muestra los campos a completar en la sección “Rol”. Tales como:
Nombre.
Descripción.
Además de una serie de permisos de tipo administrativos, el trabajo con los metadatos y la mensajería con los usuarios no registrados en la herramienta, estos son:
Administrar la herramienta.
Administrar los usuarios.
Administrar los roles.
Administrar la configuración.
Administrar las plantillas.
Administrar Objetos de Aprendizaje.
Establecer metadatos obligatorios.
Establecer metadatos a autocompletar.</td></tr>
</table>

	• Atender la mensajería. Para la manipulación de plantillas estos son: • Crear plantillas. • Revisar plantillas. • Gestionar plantillas públicas. Para el trabajo con Objetos de Aprendizaje estos son: • Crear Objetos de Aprendizaje. • Revisar Objetos de Aprendizaje. Para el trabajo con Unidades de Aprendizaje estos son: • Crear Unidades de Aprendizaje. De tipo colaborativo ellos son: • Crear blog. • Crear foro. • Crear wiki. • Crear grupo. • Lanzar convocatoria.
3. El administrador ingresa los datos necesarios y presiona el botón “Aceptar”.	3.1 El sistema crea el rol y regresa al listado inicial.
Prototipo de Interfaz Ver Anexo 2	
Poscondiciones	
Flujos Alternos de la Sección 1	
Acción del Actor	**Respuesta del Sistema**
3. El administrador deja un campo obligatorio vacío en el formulario a llenar.	3.1 El sistema muestra un mensaje de error: “Existen campos obligatorios vacíos”.
Prototipo de Interfaz	
Poscondiciones	
Sección 2 “Editar rol”	
Acción del Actor	**Respuesta del Sistema**

1. El administrador selecciona un rol de la lista y seguido accede a la opción “Editar Rol”.	1.1 El sistema muestra un mensaje de confirmación: “Desea realmente editar el rol”.
2.El administrador acepta el mensaje de confirmación	2.1 El sistema muestra todos los datos referente al rol seleccionado.
3. El administrador realiza los cambios necesarios.	3.1 El sistema envía un mensaje al administrador del éxito de los cambios realizados.
4. El administrador acepta el mensaje de confirmación.	4.1 El sistema guarda las modificaciones hechas por el administrador.
Prototipo de Interfaz Ver Anexo 2	
Flujos Alternos de la Sección 2	
Acción del Actor	**Respuesta del Sistema**
2. El administrador no acepta el mensaje de confirmación	2.1 El sistema no muestra el formulario para la edición del rol.
3. El administrador deja un campo vacío en el formulario a editar.	3.1 El sistema muestra un mensaje de error: “Verifique los datos insertados”.
Prototipo de Interfaz	
Poscondiciones	
Sección 3 “Eliminar rol”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona un usuario de la lista y seguido selecciona la opción “Eliminar”.	1.1 El sistema muestra un mensaje al administrador si está seguro de eliminar el usuario seleccionado.
2. El administrador acepta el mensaje de confirmación.	2.1 El sistema elimina el usuario seleccionado.
Prototipo de Interfaz Ver Anexo 2	
Flujos Alternos de la Sección 3	
Acción del Actor	**Respuesta del Sistema**
2. El administrador no acepta el mensaje de confirmación.	2.1 El sistema no elimina el usuario seleccionado y regresa al listado inicial.

Prototipo de Interfaz	
Poscondiciones	

2.5.4. Gestionar Configuración de la herramienta

Tabla 5 Gestionar Configuración de la herramienta.

Caso de Uso:	Gestionar Configuración de la herramienta
Actores:	Administrador
Resumen:	El caso de uso inicia cuando el actor selecciona en el panel lateral izquierdo de la aplicación la opción Configuración
Precondiciones:	El usuario debe estar autenticado.
Referencias	RF 15, RF 16, RF 17, RF 18, RF 19
Prioridad	**Crítico**

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Configuración.	1.2. El sistema muestra una vista con las distintas opciones para configurar como son: Protocolo de acceso mostrando dos campos a seleccionar: HTTP o HTTPS. Autenticación mostrando dos campos seleccionables: Manual y LDAP. Máximo de plantillas en edición, Máximo de OA en edición, Máximo de diseños en edición.
2. El administrador selecciona la opción de autenticación LDAP	2.1. El sistema muestra los siguientes campos obligatorios a llenar: • Host. • Usuario. • Contraseña. • Contexto. • Atributos.

	• Datos.
Prototipo de Interfaz Ver Anexo 2	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El administrador deja un campo obligatorio vacío en el formulario a editar.	El sistema muestra un mensaje de error: “Existen campos obligatorios vacíos”.

2.6. Conclusiones

Debido a la poca conformación de los procesos de negocio se ha desarrollado un modelo de dominio donde se muestra una visión general del entorno donde se despliega el Módulo Administración, facilitando una base para el buen hacer del flujo de trabajo Requerimientos. Como resultados se obtuvo: el diagrama de casos de usos del sistema donde se refleja la interacción entre el actor y los casos de usos. Se realizó la descripción de los casos de uso del sistema donde se identificaron 7 casos de usos de ellos 4 clasificados como críticos, donde se agruparon los 24 requisitos funcionales. Se identificó un actor. Mediante los artefactos generados se especificó cuál es la meta del módulo, para dar paso al flujo de trabajo Análisis y Diseño, y los artefactos que se obtendrán en dicho flujo.

Capítulo 3: Análisis y Diseño del sistema

Introducción

El capítulo se basa en el flujo de trabajo Análisis y Diseño de la metodología RUP, vale destacar que esta metodología aunque contempla las disciplinas de la ingeniería de software, Análisis y Diseño en un mismo flujo de trabajo por estar estrechamente relacionadas entre sí, presenta actividades diferentes con sus respectivos artefactos. En este capítulo se hace la modelación de Análisis mediante los diagramas de clases del análisis y los diagramas de colaboración, para luego modelar el diseño a través de los diagramas de clases del diseño con estereotipos web.

3.1. Análisis

El análisis es el método por el cual se obtiene una visión de los requisitos funcionales, a través de una estructuración de los mismos. Realizando esta función se logra comprender mejor el sistema.

3.1.1. Diagrama de clases del análisis

Este diagrama responde a los requisitos funcionales planteados y son representados mediante estereotipos:

- Interfaz (boundary): se maneja toda la interacción que puede existir entre los actores y el sistema.
- Control (control): representan el comportamiento secuenciado de la lógica del negocio de uno o varios casos de uso.
- Entidad (entity): representa la información persistente que se maneja en el sistema.

En esta investigación se realizó un diagrama de clases del análisis por cada caso de uso representado. A continuación se presentan los diagramas de los casos de usos críticos. Los demás diagramas se pueden visualizar en el Anexo 4

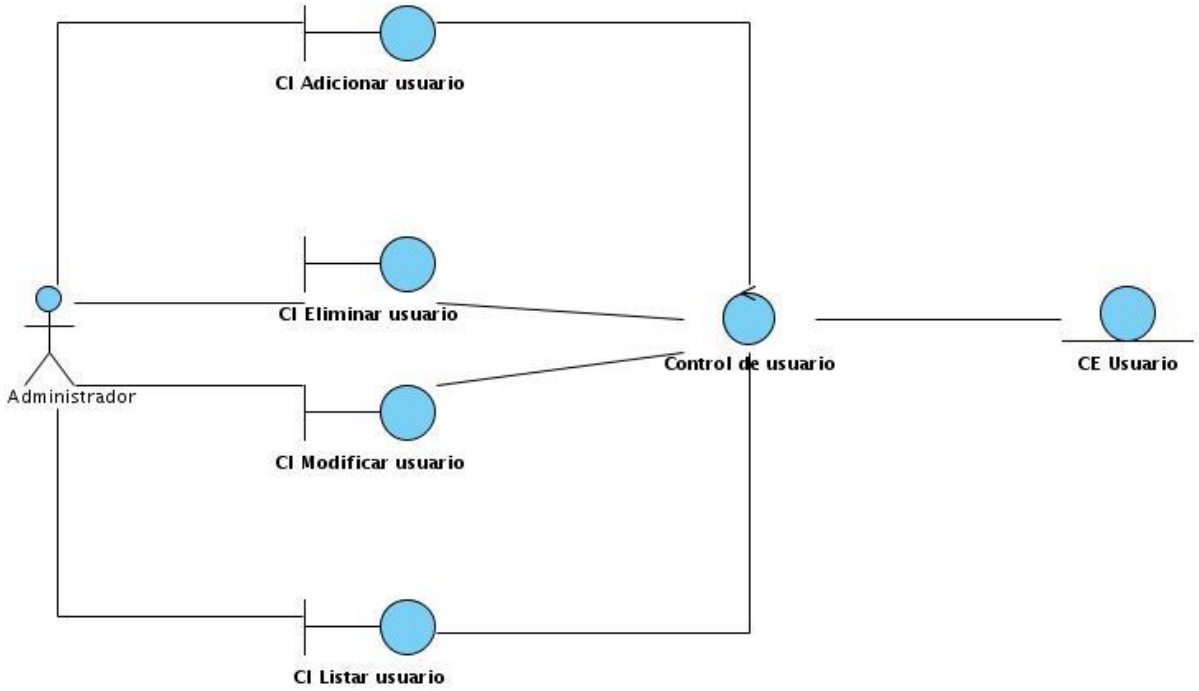

Figura 5 Diagrama de clases del análisis del CU Gestionar usuario.

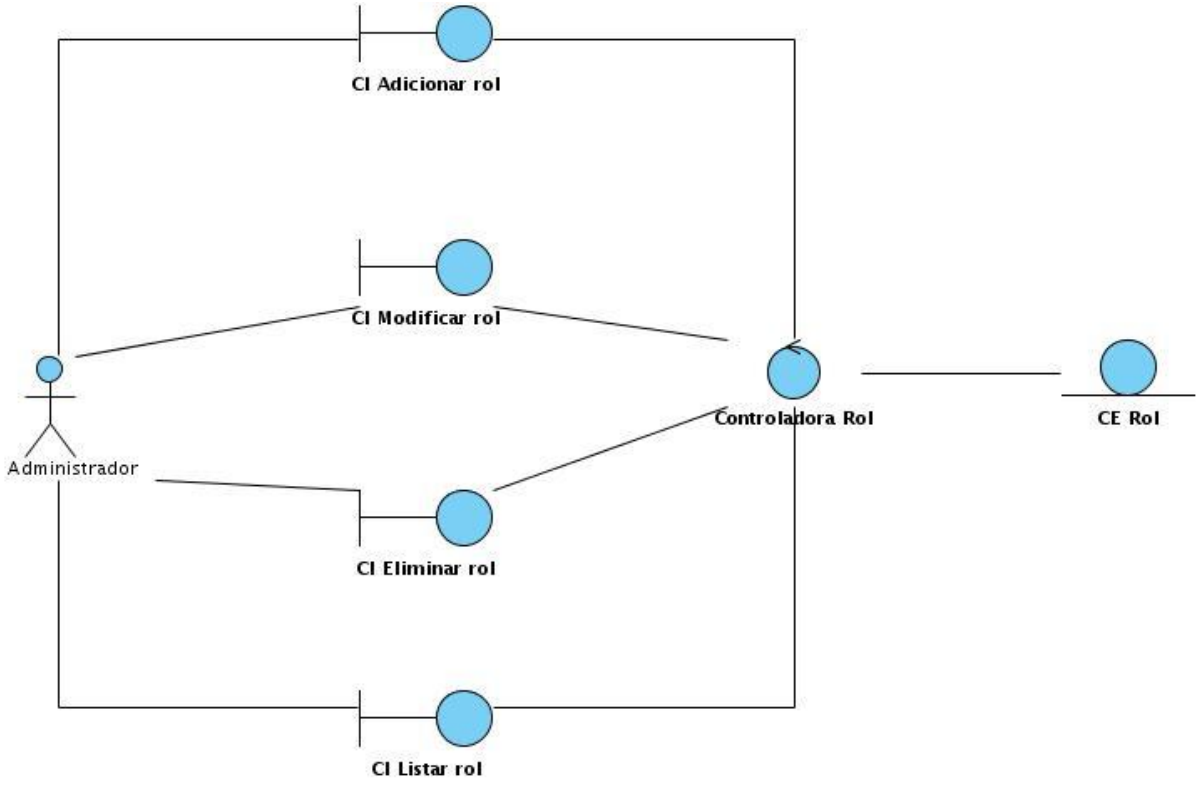

Figura 6 Diagrama de clases del análisis del CU Gestionar Roles.

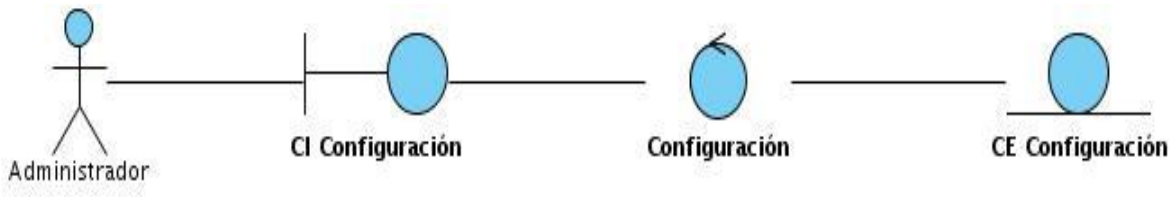

Figura 7 Diagrama de clases del análisis del CU Gestionar Configuración de la herramienta.

3.1.2. Diagrama de interacción

Los diagramas de interacción describen la colaboración entre objetos para lograr un fin. Estos diagramas se dividen en 2 grupos: de colaboración y de secuencia. En esta investigación se realizaron los diagramas de colaboración del análisis. A continuación se muestran los diagramas referentes a los casos de usos críticos. Los demás diagramas se pueden visualizar en el Anexo 5.

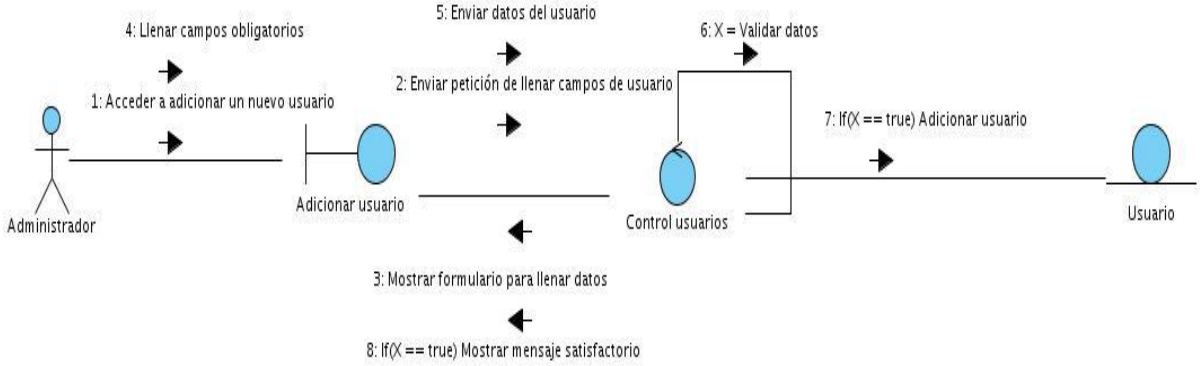

Figura 8 Diagrama de colaboración del CU Gestionar usuario. ¨Adicionar usuario¨.

Figura 9 Diagrama de colaboración del CU Gestionar usuario. ¨Eliminar usuario¨.

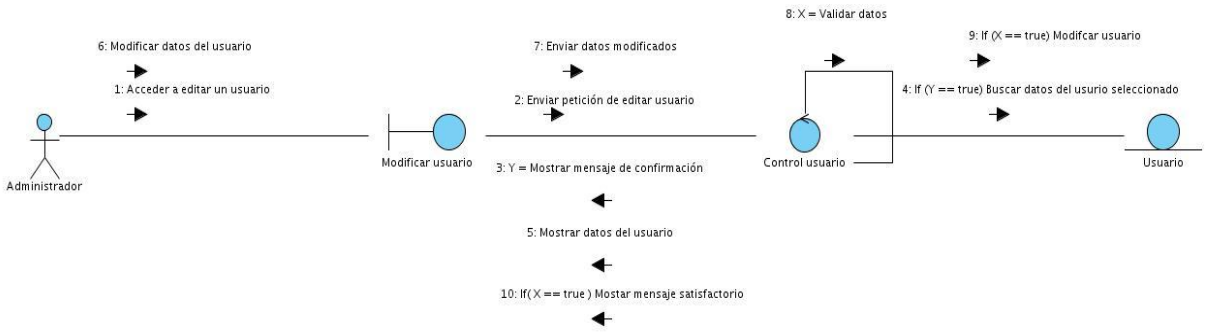

Figura 10 Diagrama de colaboración del CU Gestionar usuario. ¨Modificar usuario¨.

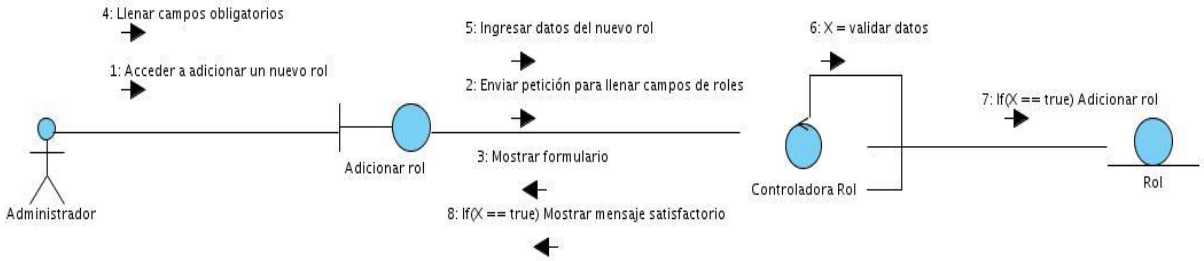

Figura 11 Diagrama de colaboración del CU Gestionar roles. ¨Adicionar rol¨.

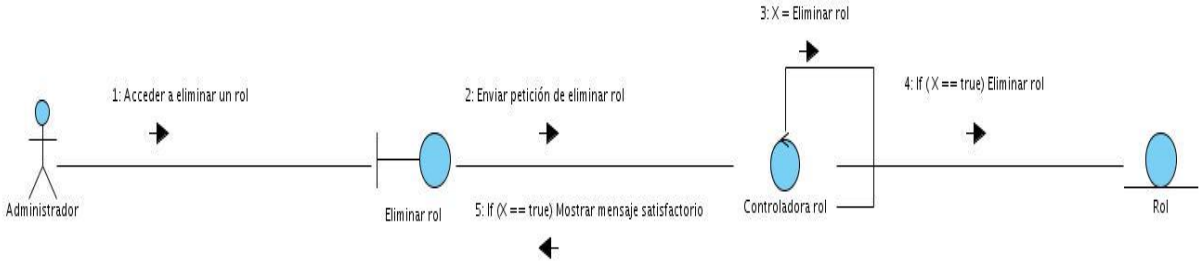

Figura 12 Diagrama de colaboración del CU Gestionar roles. ¨Eliminar rol¨.

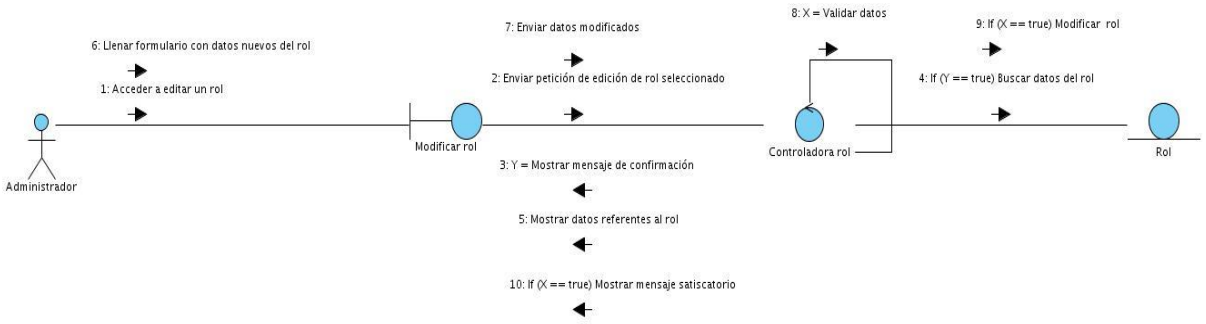

Figura 13 Diagrama de colaboración del CU Gestionar roles. ¨Modificar rol¨.

Figura 14 Diagrama de colaboración del CU Gestionar roles. ¨Listar rol¨.

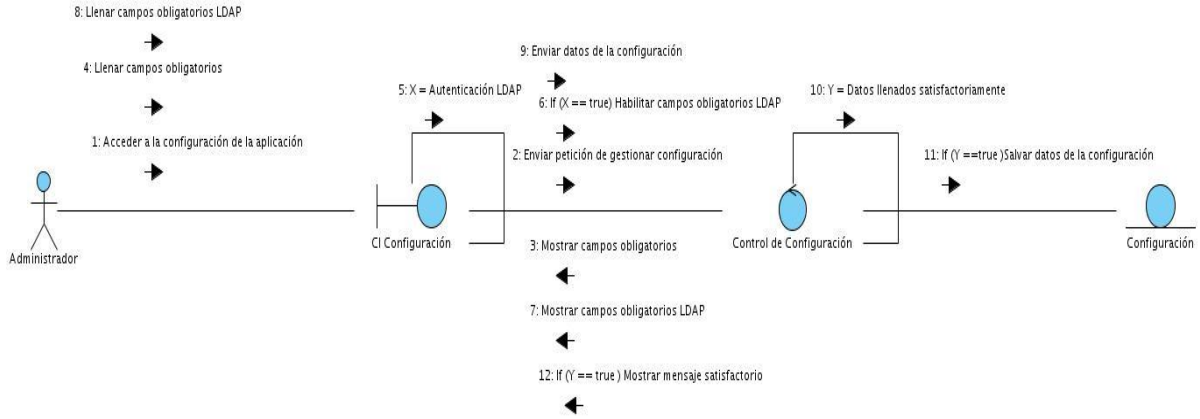

Figura 15 Diagrama de colaboración del CU Gestionar Configuración de la herramienta.

3.2. Arquitectura propuesta

La arquitectura de software es la forma que se organiza una aplicación, determina las interacciones entre sus componentes y las bases para crear su diseño y evolución. Esta fija algunas reglas y conceptos claves para que el equipo de desarrollo pueda trabajar en una misma dirección permitiendo alcanzar los objetivos pertinentes. En este trabajo se propone para la realización del diseño y la implementación el patrón clásico del diseño web conocido como arquitectura Modelo-Vista-Controlador (MVC). El framework Symfony, propuesto para el desarrollo de este módulo, está basado en este patrón que lo conforman tres niveles:

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

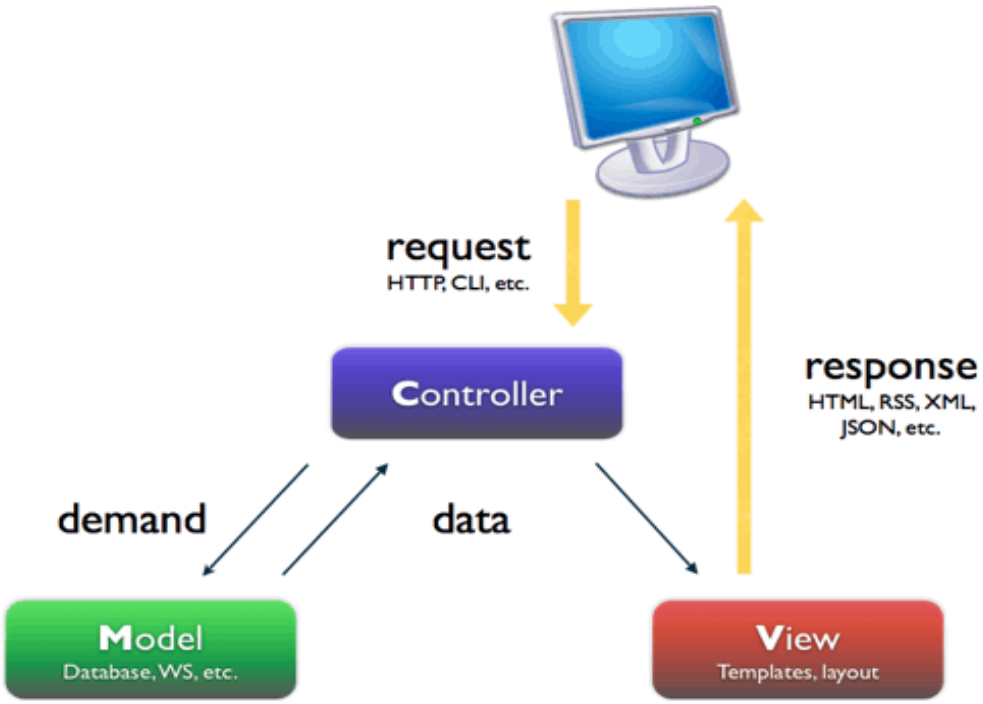

Figura 16 Arquitectura de Symfony.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. (29)

3.3. Diseño

El diseño es una disciplina que se realiza luego del análisis, haciendo un refinamiento del mismo. El diseño crea una representación o modelo del software, proporciona detalles acerca de las estructuras de datos, las arquitecturas, las interfaces y los componentes del software que son necesario para implementar el sistema (30).

3.3.1. Diagrama de clases del diseño con estereotipos web

El Módulo Administración se diseña para una herramienta web haciéndose necesario el modelado de páginas, sus enlaces, el código que se genera y a su vez el contenido cargado

dinámicamente. Para lograr esto se utilizan los estereotipos de UML para diseñar los diagramas de clases del diseño.

Tabla 6 Estereotipos web para UML.

Ícono	Estereotipo/Explicación
	<<Client Page>>: Es la página web de formato HTML, que le brinda al usuario toda la interfaz de la aplicación.
	<<Server Page>>: Es la clase encargada del código ejecutable en el servidor.
	<<Form>>: Es una colección de elementos de entradas que están contenidos en la página cliente. Se comunican con las paginas servidoras a través de una relación submit.

A continuación se presentan los diagramas de clases del diseño con estereotipos web de los casos de uso crítico. Los demás diagramas se pueden consultar en el Anexo 6.

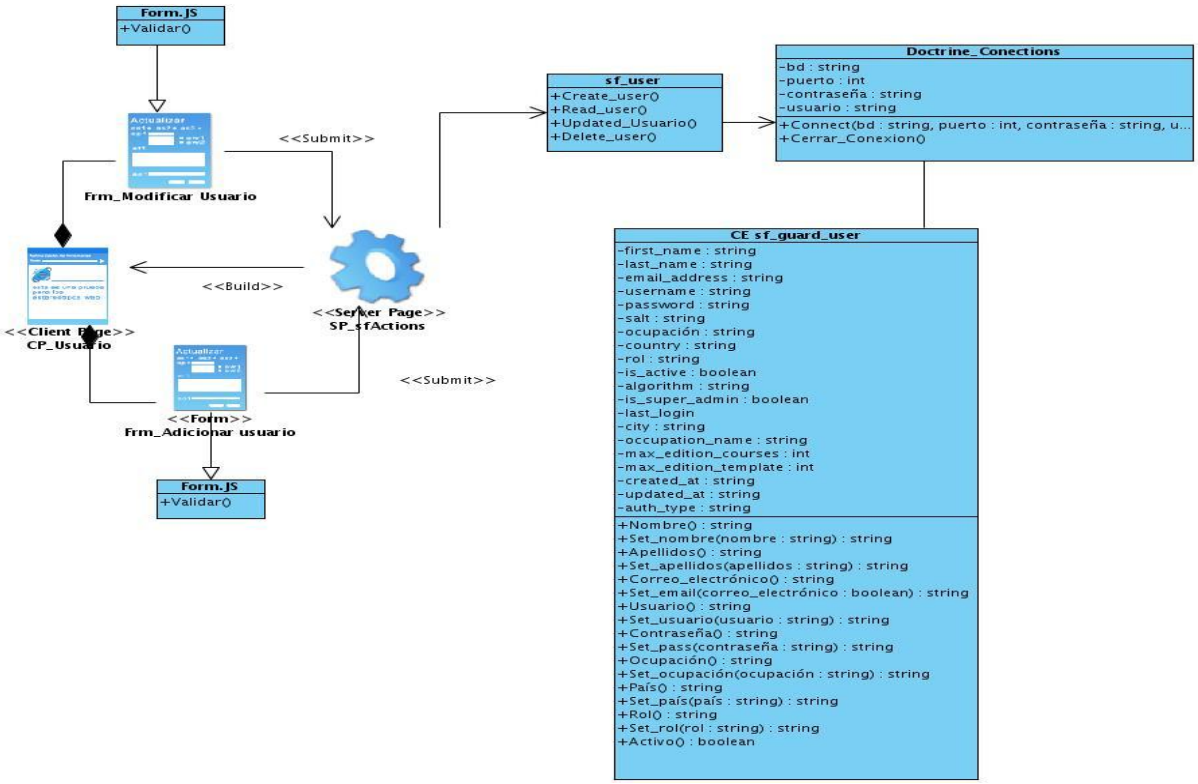

Figura 17 Diagrama de clases del diseño con estereotipos web. CU Gestionar usuario.

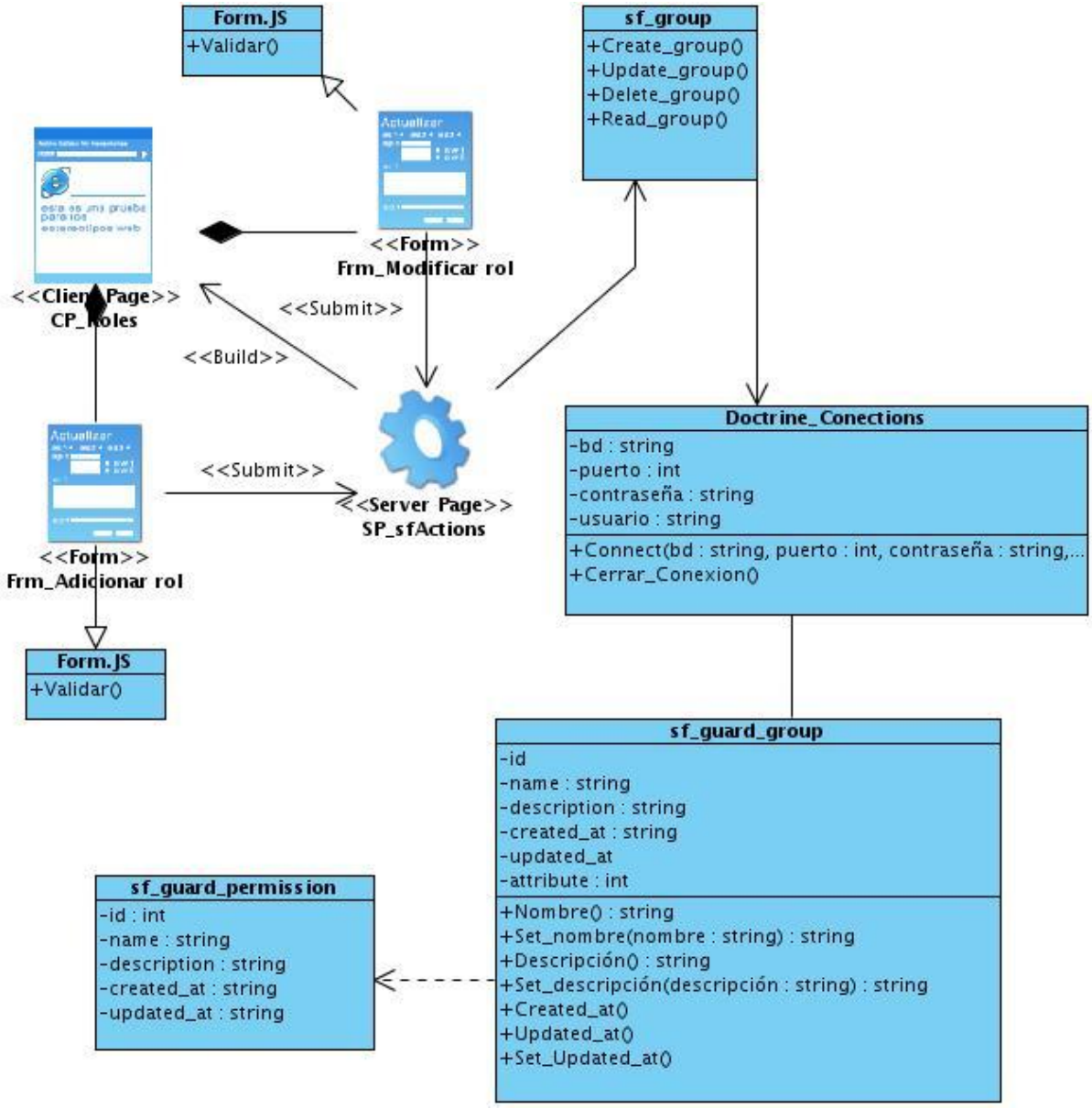

Figura 18 Diagrama de clases del diseño con estereotipos web. CU Gestionar roles.

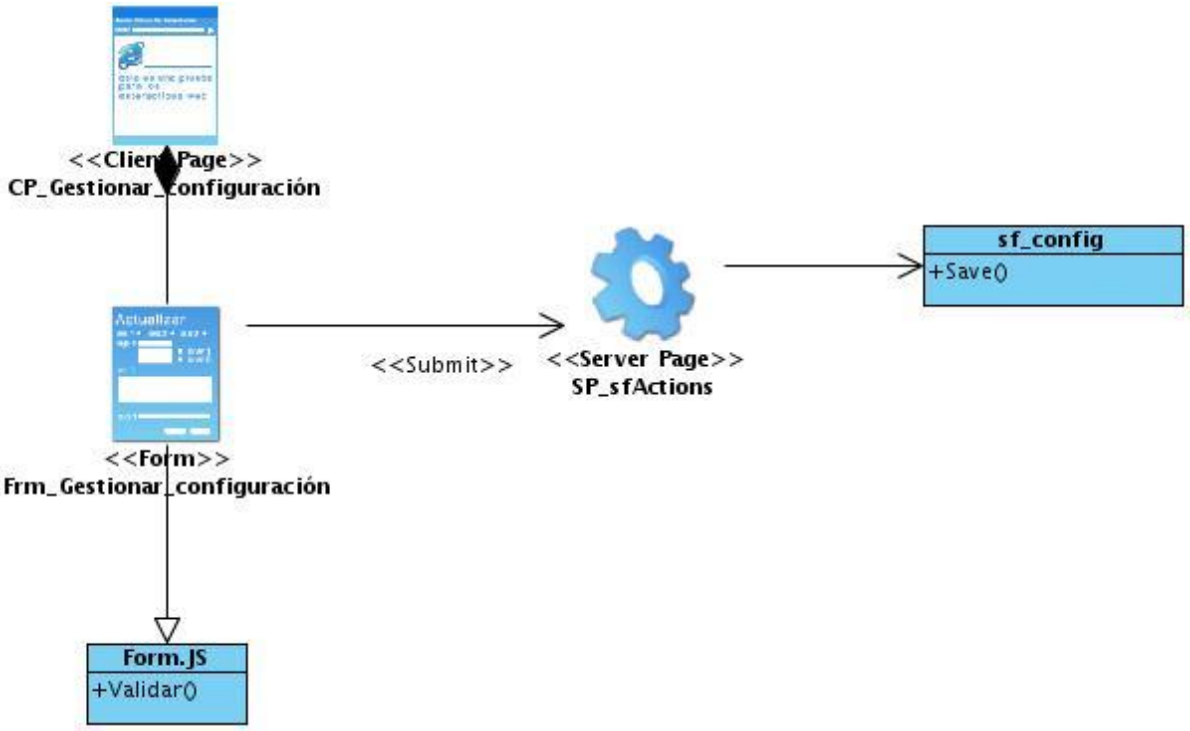

Figura 19 Diagrama de clases del diseño con estereotipos web. CU Gestionar Configuración de la herramienta.

3.3.2. Patrones de diseños utilizados

Los patrones de diseño reducen el esfuerzo del trabajo, perfeccionando la seguridad y la eficacia de los diseños. Estos están clasificados en dos grandes grupos, los GRASP (Patrones generales de software para asignar responsabilidades) y los GoF (Pandilla de los Cuatro). A continuación se muestran los empleados en el trabajo:

Patrones GRASP

En el diseño orientado a objetos, los patrones generales de software GRASP se consideran que más que patrones, son una serie de "buenas prácticas" de aplicación recomendables en el diseño de software (32).

Las responsabilidades corresponden a tareas de un objeto en cuanto a su procedimiento. Estas responsabilidades son fundamentalmente de dos tipos:

Conocer:

- Conocer los datos privados encapsulados.
- Conocer los objetos relacionados.

- Conocer las cosas que puede derivar o calcular.

Hacer:

- Hacer algo él mismo, como crear un objeto o hacer un cálculo.
- Iniciar una acción en otros objetos.
- Controlar y coordinar actividades en otros objetos (32).

Entre los patrones destacados de GRASP se encuentran:

Patrón Controlador

El patrón controlador es el que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método invocado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control (32).

Este patrón se evidencia en todas las clases Actions como sfContext, que manejan todas las peticiones web, siendo este el único lugar de acceso a la aplicación en un determinado ambiente.

Patrón Creador

El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que:

- Tiene la información necesaria para realizar la creación del objeto.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.
- Contiene o agrega la clase (32).

El uso de este patrón se evidencia en las clases Actions, permitiendo el acceso a la información almacenada en las clases entidades.

Patrón Experto

El patrón experto en información es el principio básico de asignación de responsabilidades. Nos indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión (32).

El uso de este patrón se visualiza en la definición de las clases de acuerdo a las funcionalidades que deben realizar a partir de la información manejada dentro del componente, como por ejemplo las clases controladoras y las del modelo. Ejemplo de ello es la clase sf_user, la cual gestiona las operaciones que conciernen a las funciones adicionar, modificar, eliminar y actualizar los datos de los usuarios de la aplicación.

Patrones GoF (Gang of Four)

Estos patrones de diseños fueron creados por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides conocidos por el grupo Pandilla de los Cuatro (Gang of Four) (33) y al igual que los anteriormente mencionados poseen características distintivas como ser reutilizables y efectivos en la resolución de problemas.

Patrón MVC

En la arquitectura MVC de Symfony se implementan varias clases como:

- sfController: Es la clase del controlador y se encarga de decodificar la petición y transferirla a la acción correspondiente.
- sfRequest: Guarda todos los elementos que integran la petición (parámetros, cookies, cabeceras, etc.)
- sfResponse: Posee las cabeceras de la respuesta y los contenidos. El contenido de este objetivo se convierte en la respuesta HTML que se remite al usuario.
- El singleton de contexto: almacena una referencia a todos los objetos que forman el núcleo de Symfony y puede ser accedido desde cualquier punto de la aplicación (27).

Patrón Singleton o instancia única

Pertenece al grupo de los patrones creacionales. Su función es garantizar que una clase tenga solo un objeto. Para ello la clase debe tener el constructor privado y además contar

con un método estático para devolver la única instancia. Este patrón se pone de manifiesto en la clase sfContext de Symfony, la cual hace referencias a otras clases como user, response y request, y se pueden acceder a ellas a través del método sfContext::getInstance().

Patrón Decorator o decorador

La aplicación de este patrón se puede observar en el método “decorator” de la clase abstracta sfView, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo layout.php o plantilla global guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o sea el layout decora la plantilla (27).

3.4. Modelo de datos

Según Ullman “Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos. Es formal pues los objetos del sistema se manipulan siguiendo reglas perfectamente definidas y utilizando exclusivamente los operadores definidos en el sistema, independientemente de lo que estos objetos y operadores puedan significar” (34).

El modelo de datos propuesto en la solución cuenta con un total de 5 entidades críticas. Para su construcción se tuvo en cuenta la reducción a la mínima expresión de los campos nulos y la persistencia de campos resúmenes para agilizar recuperaciones frecuentes de algunos datos.

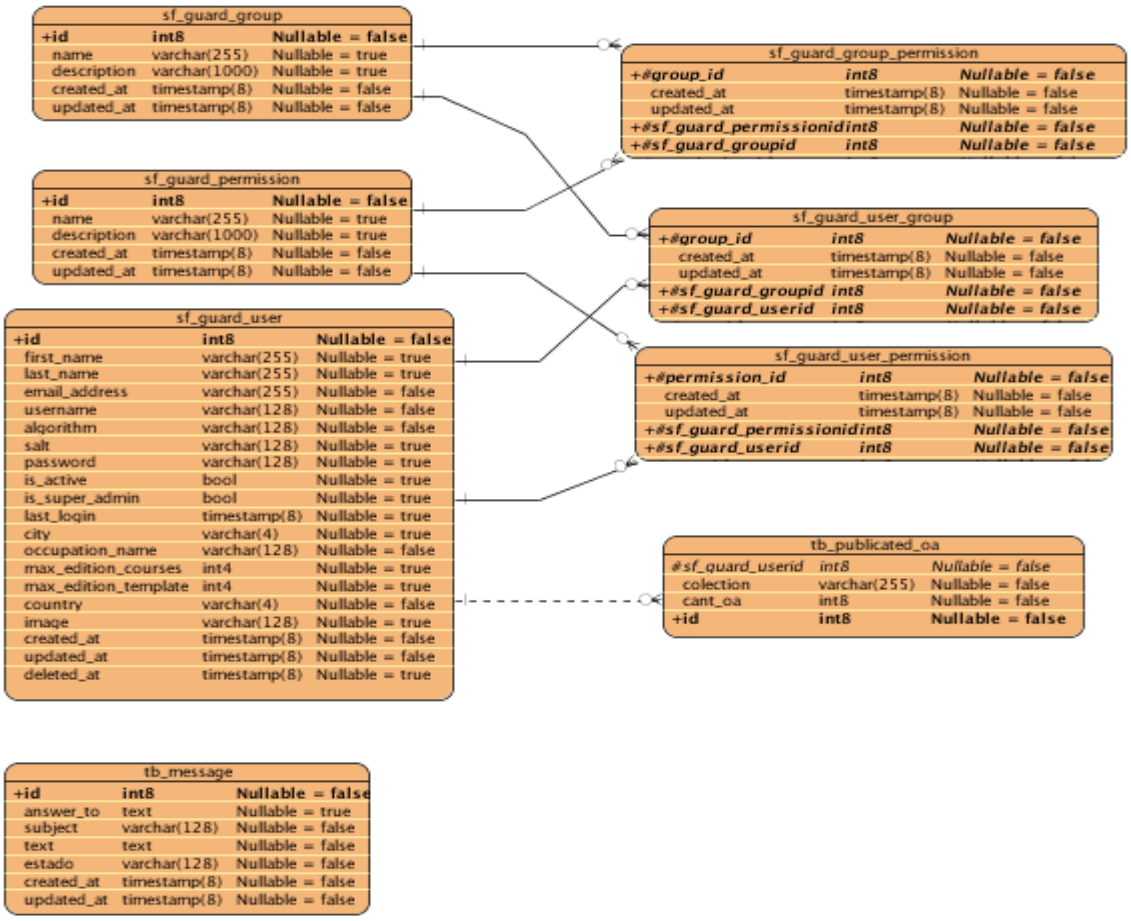

Figura 20 Modelo de datos Módulo Administración.

Descripción de las tablas

Tabla 7 Descripción de la tabla sf_guard_user de la base de datos PostgreSQL.

Nombre: sf_guard_user		
Descripción: Tabla donde se guarda los datos de los usuarios de la herramienta de autor web CRODA 2.0 haciendo uso del plugin de Symfony sfGuard.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del usuario y a su vez llave primaria de la tabla.
first_name	varchar	Nombre real del usuario.
last_name	varchar	Apellido del usuario.
email_address	varchar	Dirección de correo del usuario.
username	varchar	Nombre del usuario en la herramienta.

algorithm	varchar	Algoritmo para encriptar las contraseñas de los usuarios.
salt	varchar	Texto aleatorio que va concatenado al algoritmo de encriptación.
password	varchar	Contraseña del usuario.
is_active	bool	Si está activo el usuario en la herramienta.
is_super_admin	bool	Si es súper administrador de la herramienta.
last_login	timestamp	Último día que tuvo acceso a la herramienta.
city	varchar	Cuidad de origen.
occupation_name	varchar	Identificador del rol.
max_edition_courses	int	Máximo de cursos en edición.
max_edition_template	int	Máximo de plantillas en edición.
country	varchar	País.
image	varchar	Imagen del usuario.
auth_type	varchar	Tipo de autenticación.
created_at	timestamp	Fecha que se creó el usuario.
updated_at	timestamp	Fecha que se actualizó el usuario.
deleted_at	timestamp	Fecha en que se eliminó el usuario.

Tabla 8 Descripción de la tabla sf_guard_group de la base de datos PostgreSQL.

Nombre: sf_guard_group		
Descripción: Tabla donde se guarda los datos de los roles o grupos de la herramienta de autor web CRODA 2.0 haciendo uso del plugin de Symfony sfGuard.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del rol y a su vez llave primaria de la tabla.
name	varchar	Nombre del rol.
description	varchar	Descripción de la función del rol
created_at	timestamp	Fecha que se creó el rol.
Updated_at	timestamp	Fecha que se actualizó el rol.

Tabla 9 Descripción de la tabla sf_guard_permission de la base de datos PostgreSQL.

Nombre: sf_guard_permission		
Descripción: Tabla donde se guarda los datos de los permisos de la herramienta de autor web CRODA 2.0 haciendo uso del plugin de Symfony sfGuard.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del rol y a su vez llave primaria de la tabla.
name	varchar	Nombre del permiso.
description	varchar	Descripción de la función del rol
created_at	timestamp	Fecha que se creó el permiso.
updated_at	timestamp	Fecha que se actualizó el permiso.

Tabla 10 Descripción de la tabla tb_message de la base de datos PostgreSQL.

Nombre: tb_message		
Descripción: Tabla donde se guarda los datos de la mensajería de la herramienta de autor web CRODA 2.0.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del mensaje y a su vez llave primaria de la tabla.
answer_to	text	Nombre del destinatario.
subject	varchar	Asunto del mensaje
text	text	Cuerpo del mensaje
estado	bool	Si ha sido enviado el mensaje
created_at	timestamp	Fecha que se creó el permiso.
Updated_at	timestamp	Fecha que se actualizó el permiso.

Tabla 11 Descripción de la tabla tb_publicated_oa de la base de datos PostgreSQL.

Nombre: tb_publicated_oa		
Descripción: Tabla donde se almacenan la información de los Objetos de Aprendizaje por colección, autor y cantidad.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del Objeto de Aprendizaje y a su vez llave primaria de la tabla.

collection	varchar	Materia educativa para la que está dedicado el Objeto de Aprendizaje.
cant_oa	int	Cantidad de Objeto de Aprendizaje
sf_guard_userid	int	Identificador del autor del Objeto de Aprendizaje publicado.

3.5. Validación de la propuesta de solución

La validación consiste en determinar si un software cumple con las exigencias del cliente. Con el uso de métricas se podrá suprimir los diferentes errores en los distintos artefactos generados como son los requerimientos y los casos de uso.

3.5.1. Métricas de calidad de la especificación de requisitos

La métrica de la Calidad de Especificación de los Requisitos mide la especificidad de los requisitos, haciendo que la parte interesada pueda entenderlos de manera fácil y se puedan probar (30).

Con el conocimiento de los requerimientos (ver acápite 2.2. Requerimientos del sistema), ya se puede poner en práctica la técnica de validación. Para apreciar el resultado obtenido ver Anexo 7.

RT = RF + RNF
RT = 24 + 10 = 34

RF: número de requisitos funcionales.
RNF: número de requisitos no funcionales.
RT: total de requisitos.

Especificidad

Para determinar la especificidad o falta de ambigüedad de los requisitos, se sugiere una métrica basada en la consistencia de la interpretación de los revisores de cada requisito (35).

$\mathbf{Q_1 = R_{ii} / RT}$

R_{ii}: Número de requisitos que todos los revisores entendieron igual
RT: Total de requisitos.

Q_1: Grado de especificación de los requisitos. Mientras más cerca se encuentre de 1 menor será la ambigüedad en la especificación.

Primera revisión

Q_1 = 31/ 33

Q_1 = 0.94

Resultado: Se encontró que el 6% de los requerimientos mostraban ambigüedades, haciéndose necesaria la reelaboración de los requisitos en términos más entendibles.

Segunda revisión

Q_1 = 33/ 33

Q_1 = 1

Resultado: No se encontraron ambigüedades.

Compleción

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de completitud en la definición de los requisitos. Este valor se calcula como se muestra a continuación:

Q_2 = RC / (RC + RPE)

Q_2: Grado de completitud de los requisitos.

RC: Número de requisitos completos.

RPE: Número de requisitos pobremente especificados.

Primera revisión

Q_2 = 31 / (31 + 2)

Q_2 = 0.94

Resultado: El 6% de las definiciones de los requisitos se encontraban en un estado pobre, haciéndose necesario un estudio más profundo de los procesos de negocios. Con esto se logró una segunda revisión exitosa gracias a la mejora de la especificación de requerimientos.

Corrección

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de corrección en la definición de los requisitos. Este valor se calcula a partir de la siguiente operación:

Q_3 = (RC-RI)/RC

Q_3: Grado de validación de los requisitos.

RC: Número de requisitos validados como correctos.

RI: Número de requisitos validados como incorrectos.

Primera revisión

Q_3 = (33-0)/33

Q_3 = 1

Resultado: Todos los requisitos están definidos de forma correcta.

Consistencia interna

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 y expresa que no existen subconjuntos de requisitos contradictorios. El valor óptimo de esta métrica es el más cercano a 1. Este valor se calcula como se muestra a continuación:

Q_4 = (RE - RC)/RE

Q_4: Grado de consistencia interna.

RE: Número de requisitos especificados.

RC: Número de requisitos en conflicto con otros requisitos.

Q_4 = (33 – 0)/ 33

Q_4 = 1

Resultado: No existen requisitos en conflictos.

Estabilidad

Para medir la estabilidad de los requisitos de software, en el presente trabajo se aplicó la métrica propia para esto, la cual ofrece valores entre 0 y 1. El mejor valor es el más cercano a 1. La estabilidad de los requisitos se calcula de la siguiente forma:

E = (RT - RM)/RT

E: Valor de la estabilidad de los requisitos.

RT: Total de requisitos.

RM: Número de requisitos modificados.

Clasificación de la estabilidad

Tabla 12 Descripción de la tabla Intervalo y Clasificación.

Clasificación	Intervalo
Alta	0.85 <= **E** <= 1
Media	0.75 <= **E** < 0.85
Baja	**E** < 0.75

E = (33 - 0)/33

E = 1

Resultado: No hubo modificación en los requerimientos. Por tanto la estabilidad ha sido alta.

3.5.2. Validación a través de prototipos de interfaz de usuario (PIU)

Los prototipos de interfaz de usuario esbozan la información de forma gráfica del producto final y tienen como objetivo limar detalles en la corrección de la especificación de requisitos. Para efectuar esta técnica se seleccionaron las personas que validarían los prototipos de interfaz de usuarios: (profesores del proyecto). Se desarrollaron escenarios que permitieran al usuario obtener una mejor visualización de todo el flujo de intercambio con la aplicación. Se ejecutaron los escenarios creados, permitiendo documentar los errores encontrados, lo que facilitó la erradicación de errores en los requisitos y a su vez un mejor entendimiento con el cliente.

3.5.3. Validación de la propuesta a través del método de expertos.

Se le conoce como experto a la persona que goza de un alto grado de sabiduría en un tema determinado y se encuentra capacitada para brindar su opinión sobre la situación a tratar. Para evaluar la eficacia y el nivel de confianza del trabajo se seleccionó a un grupo de especialistas, las cuales dieron su valoración en cuanto a novedad, aporte, resultados y posibilidad de poner en práctica.

Para lograr el objetivo del método se siguió el siguiente algoritmo:

1. Selección de expertos.

Para realizar esta tarea se tuvo en cuenta la experiencia en el desarrollo de software, la labor como profesional y su conocimiento sobre la metodología de desarrollo de software. A continuación se presenta el listado de dichos especialistas.

Tabla 13 Descripción de la tabla Expertos.

ID	Expertos	
E_1	Dpto. Producción De Herramientas Educativas. Centro FORTES	**Ing. Jorge Martínez Padrón.**
E_2	Dpto. Técnicas De Programación	**Ing. Leonardo San Román Labaut.**
E_3	Dpto. Producción De Herramientas Educativas. Centro FORTES.	**Ing. Yandris Mata Cabrera.**
E_4	Dpto. Programación e ISW.	**Ing. Ángel Alberto Vázquez Sánchez.**
E_5	Dpto. IST.	**Ing. Maikel Aparicio Reytor.**

2. Enviar información a especialistas.

Le fue enviado a cada experto un resumen del trabajo para su estudio para que puedan evaluarlo según los siguientes aspectos.

Tabla 14 Descripción de la tabla Alternativas de evaluación.

Ind	Alternativas
A_1	Necesidad de mejorar la administración en la herramienta de autor web CRODA 2.0.
A_2	Nivel de comprensión de los artefactos generados en el proceso de desarrollo de software.
A_3	Contribución al control del desarrollo de software educativo.

Las alternativas serán evaluadas apoyándose la siguiente tabla de pesos.

Tabla 15 Descripción de la tabla Valor – peso.

Valor	Peso
Ninguno	1
Poco	2
Medio	3
Moderado	4

Alto	5

Evaluación dada por los expertos:

Tabla 16 Descripción de la tabla Evaluación de las alternativas.

Alternativas	E_1	E_2	E_3	E_4	E_5	EPA
A_1	5	5	5	4	3	4,4
A_2	4	4	5	5	5	4,6
A_3	4	4	4	4	4	4

EPA: Evaluación promedio de las alternativas.

3. Cálculo de grado de conformidad (GC).

Luego de conocidas las evaluaciones promedios de las alternativas se da paso al cálculo del grado de aceptación mediante la siguiente fórmula.

GC = ∑ EPA / 15

GC = 13 / 15

GC = 0.86

Tabla 17 Descripción de la tabla Probabilidad de éxito.

Probabilidad de éxito	Rangos
Ninguno	GC < 0.25
Baja	0.25 >= GC > 0.5
Media	0.5 >= GC > 0.75
Alta	0.75 > GC

4. Resultado alcanzado.

La investigación brindó como resultado un nivel alto en la calidad, los artefactos generados validados presentan una buena comprensión y se confirma la necesidad de mejorar el Módulo Administración en la herramienta de autor web CRODA. Analizando el grado de conformidad se garantiza que la propuesta de solución tiene una alta probabilidad de éxito.

3.6. Conclusiones

Con la elaboración de este capítulo permitió realizar un refinamiento del flujo de trabajo Análisis, logrando un mayor entendimiento de los requisitos funcionales. Se estableció un punto de partida para la elaboración del diseño. Por último, con la confección de los artefactos del Diseño fundados en el patrón arquitectónico modelo- vista- controlador se fomentó el módulo en un lenguaje asequible para futuras implementación. Las validaciones aplicadas a los requisitos posibilitaron detectar y enmendar errores, favoreciendo al buen desarrollo de los restantes artefactos.

Conclusiones generales

Con el desarrollo de la presente investigación se analizaron aspectos relacionados con la administración en aplicaciones web y la urgencia que amerita perfeccionar el actual Módulo Administración de la herramienta de autor web CRODA, dando por resuelto los objetivos específicos se alcanzaron las siguientes conclusiones basado en el análisis y diseño de un nuevo Módulo Administración.

- Se identificaron las tecnologías, herramientas y funcionalidades necesarias para contribuir al desarrollo del Módulo Administración para CRODA 2.0.

- Se sentaron las bases metodológicas correspondientes para garantizar la correcta implementación del Módulo Administración para CRODA 2.0.

- La validación de la propuesta de solución mediante el método de expertos arrojó resultados que garantizan la aceptación de las funciones requeridas en el Módulo Administración para CRODA 2.0.

Recomendaciones

Se recomienda para la realización de futuros trabajos, en aras de fortalecer y potenciar la herramienta de autor web CRODA:

- Desarrollar la implementación del Módulo Administración para CRODA 2.0.

Bibliografía

1. **Thompson, Janneth.** Administracion en Teoria. [Online] julio 29, 2009. [Cited: Diciembre 8, 2011.] http://administracionenteoria.blogspot.com/2009/07/definicion-de-administracion.html.

2. **About.com.** About.com. [Online] 2012. [Cited: enero 15, 2012.] http://www.about.com.

3. **Universidad Autónoma de Colombia, Instituto Superior de Pedagogía.** [Online] 2004. [Cited: febrero 9, 2012.] http://www.isp.fuac.edu.co.php.

4. **Guglielmetti, Marcos.** MasterMagazine. [Online] 2004. [Cited: febrero 9, 2012.] http://www.mastermagazine.info/termino/7056.php.

5. **HispaRed.** ABCdatos. [Online] julio 12, 2005. [Cited: febrero 9, 2012.] http://www.abcdatos.com/tutoriales/tutorial/l6783.html.

6. **Technologies, Iguana Information.** iguanait. [Online] 2012. [Cited: febrero 9, 2012.] http://www.iguanait.com/tecnologia/iguana-java-framework/gestion-de-usuarios-y-permisos-de-acceso-al-sistema.

7. **MS, Espiral.** Prosafety software. [Online] 2011. [Cited: febrero 9, 2012.] http://www.prosafety.es/gestion-de-roles-y-permisos.

8. **Siainternational.** Sistemas Informáticos Abiertos. [Online] 2012. [Cited: febrero 8, 2012.] http://www.siainternational.com/articles/06_S.htm.

9. **Diccionario Manual de la Lengua Española.** Cuba : Larousse Editorial, 2007.

10. **Xombra.** Xombra team. [Online] mayo 30, 2006. [Cited: mayo 11, 2012.] http://www.xombra.com/go_news.php?nota=2371.

11. **Torre, Aníbal de la.** adelat. [Online] 2006. [Cited: noviembre 20, 2011.] http://www.adelat.org/.

12. **Trevejo Alonso, Juan Antonio.** *Joomla! para principiantes. Aprendiendo a crear y mantener sitios web.* Primera. 2006. ISBN 84-611-3754-X.

13. **Alfredo.** Aprendizaje a Distancia. *Aprendizaje a Distancia.* [Online] Abril 25, 2009. http://aprendizajedistancia.blogspot.com/2009/04/que-es-un-lms.html.

14. **educatorsmaterials.** Educator Materials. [Online] Educator Materials, 2011. [Cited: enero 31, 2012.] http://www.educatormaterials.com.

15. *Las herramientas de autor en el proceso de producción de cursos en formato digital.* **Montero O'Farril, José L and Herrero Tunis, Elsa.** 33, Sevilla.España : Secretariado de Recursos Audiovisuales y Nuevas Tecnologías. Universidad de Sevilla., 2008, Pixel Bit Revista de medios y educación., pp. 59-72. ISSN 1133-8482.

16. **Sommerville, Ian.** *Ingeniería del Software.* Madrid : PEARSON EDUCACIÓN, SA., 2005. p. 5 y 124. ISBN:84-7829-074-5.

17. **Molpeceres, Alberto.** Scribd. [Online] diciembre 15, 2002. [Cited: enero 25, 2012.] http://es.scribd.com/doc/60812755/Http-Www-javahispano.

18. **INTECO.** Scribd. [Online] marzo 2009. [Cited: enero 25, 2012.] http://es.scribd.com/doc/62931905/45/Extreme-Programming-XP.

19. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 2004. ISBN: 9788478290369.

20. **Rumbaugh, Jame, Jacobson, Ivar y Booch, Grady.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley, 2000.

21. **Definicion ABC.** definicionabc. [Online] 2007. [Cited: enero 20, 2012.] http://www.definicionabc.com/tecnologia/html.php.

22. **desarrolloweb.** desarrolloweb.com. [Online] enero 8, 2012. http://desarrolloweb.com/css.

23. —. desarrolloweb.com. [Online] 2012. [Cited: enero 9, 2012.] http://www.desarrolloweb.com/php.

24. **Apache Software Fundation.** Apache. [Online] 2011. [Cited: enero 8, 2012.] http://httpd.apache.org/.

25. **PostgreSQL.** PostgreSQL. *PostgreSQL.* [Online] 2012. [Cited: enero 10, 2012.] http://www.postgresql.org/.

26. **Álvarez, Miguel Ángel.** desarrolloweb.com. [Online] noviembre 23, 2009. [Cited: mayo 25, 2012.] http://www.desarrolloweb.com/articulos/codeigniter.html.

27. **Potencier, Fabien and Zaninotto, François.** *Symfony La guía definitiva.* [trans.] Miguel Sanchez, Luciano A. Andrade , Martín Palacio Pentucci Javier Eguíluz Pérez. s.l. : Apress, 2008. ISBN-13: 978-1590597866.

28. **Frederick, Shea, Ramsay, Colin and Blades, Steve 'Cutter'.** *Learning Ext JS.* s.l. : Akshara Aware, 2008.

29. **Garcerant, Iván.** Tecnología y Synergix. *Tecnología y Synergix.* [Online] julio 10, 2008. [Cited: febrero 12, 2012.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

30. **Pressman, Roger s.** *Ingeniería del Software. Un enfoque práctico.* [ed.] Concepción Fernández. [trans.] Depto de lenguajes y sistemas informáticos y escuela universitaria de informática de la universidad Pontificia de Salamanca. 6ta Edición en Español. Salamanca : MC Graw Hill, 2002. p. 956. Vol. 1 y 2. ISBN: 970-10-5473-3.

31. **Sparx Systems Pty Ltd.** Sparx Systems. [Online] 2007. [Cited: febrero 20, 2012.] http://www.sparxsystems.com.ar/resources/tutorial/use_case_model.html.

32. **Saavedra, Jorge.** El mundo informático. [Online] mayo 8, 2007. [Cited: mayo 2, 2012.] http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

33. **Angelfire.** geek the planet. [Online] mayo 11, 2011. [Cited: mayo 3, 2012.] http://geektheplanet.net/5462/patrones-gof.xhtml.

34. **Ullman.** Instituto Tecnológico de Colima. [Online] 1999. [Cited: mayo 10, 2012.] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u2_1.htm.

35. **Astorga Vargas, María Ángelica.** [Online] febrero 2007. [Cited: mayo 30, 2012.] http://delfin.mxl.uabc.mx/~angelica/Metricas.pdf.

36. **SÁNCHEZ, Enrique RUIZ-VELASCO.** *Sistemas de Gestión de Aprendizaje vs. Sistemas de Gestión de Contenidos.* 2008.

37. **Ibermática.** Ibermática. [Online] 2012. [Cited: febrero 8, 2012.] http://www.ibermatica.com/ibermatica/sit/situsuarios.

38. **Smith, Andy.** *Human-computer factors: A study of users and information systems.* Londres : MacGraw-Hill, 1997. ISBN 10:0077092619 .

39. **Thompson, Ivan.** Promonegocios.net. [Online] enero 2008. [Cited: mayo 10, 2012.] http://www.promonegocios.net/administracion/definicion-administracion.html.

40. **Menéndez, Rosa.** Proyectos. [Online] 2009. [Cited: enero 11, 2012.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info36/proyectos.html#a.

41. **hispamp3.** hispamp3. [Online] junio 13, 2005. [Cited: enero 10, 2012.] http://www.hispamp3.com/2005/06/13/badblue-2-6-4.

42. **la web en la educación.** la web en la educación. [Online] 2010. [Cited: enero 23, 2012.] http://lawebenlaeducacion2010.blogspot.com/p/lms-definicion.html.

43. **Zamitiz, Ing. Carlos Alberto Román.** [Online] [Cited: Noviembre 10, 2011.] http://profesores.fi-b.unam.mx/carlos/aydoo/uml.html.

44. **B. Bernárdez, A. Durán, M. Toro.** Revista de Procesos y Métricas de las Tecnologías de la Información. [Online] Agosto 2004. [Cited: mayo 30, 2012.] http://www.google.com.cu/url?sa=t&rct=j&q=%C2%BFHay+respuestas+a+todas+las+peticiones+que+el+actor+del+caso+de+uso+hace+al+sistema+y+viceversa%3F&source=web&cd=1&ved=0CCAQFjAA&url=http%3A%2F%2Fwww.aemes.org%2Findex.php%2Frevista-de-procesos-y-metricas%2Fn. ISSN: 1698-2029.

45. **Shneiderman, Ben, et al.** *Designing The user interface, Strategies for effective Human-computer interaction.* [ed.] Addison-Wesley. 1998. ISBN-10: 0321537351.

46. **Molich, R. and Nielsen, J.** Heuristic evaluation of user interfaces. *Heuristic evaluation of user interfaces.* s.l. : Proceedings of the ACM CHI´94 Conference, 1990, pp. 249-256.
47. **Martin, Robert C.** *Design Principles and Design Patterns.* 2000.