Universidad de las Ciencias Informáticas
Facultad 3

Título: Sistema de Gestión del Etiquetado

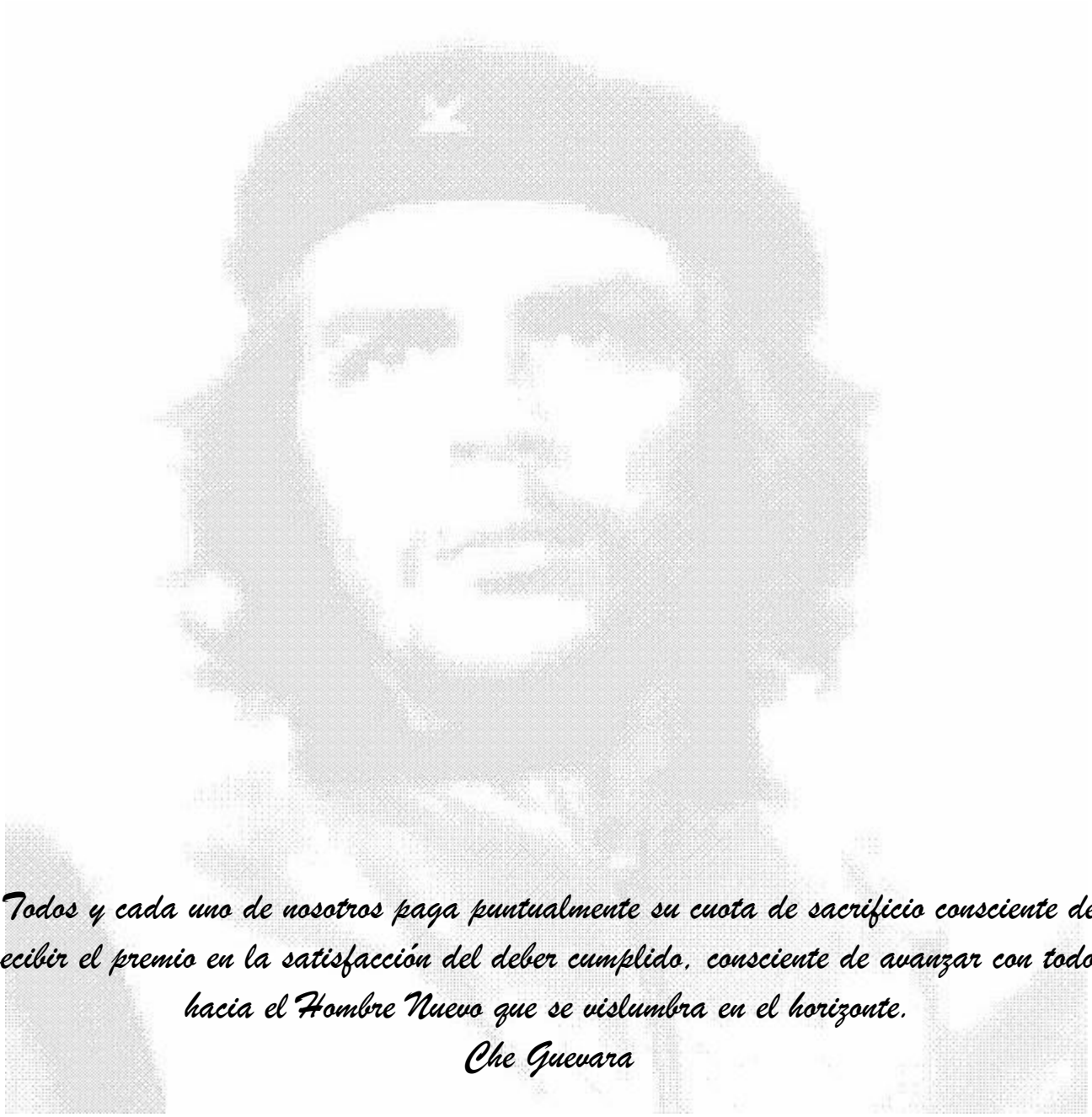
Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Pedro Carlos Gardona González.
Julio César Salgueiro Braña.

Tutor: Ing. Pedro Enrique Castiñeiras Sánchez.

Ciudad de La Habana, Junio 2009.



Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, consciente de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.

Che Guevara

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Autor: Pedro Carlos Gardona.

_____

Firma del Autor

Autor: Julio Cesar Salgueiro Braña.

_____

Firma del Autor

Tutor: Ing. Pedro Enrique Castiñeiras Sánchez.

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing. Pedro Enrique Castiñeiras Sánchez.
Graduado en el 2008 de Ingeniería en las Ciencias Informáticas en la UCI.
Profesor Adiestrado.

A nuestros amigos por estar siempre en los buenos y malos momentos.
A nuestro tutor Pedro por su paciencia, su apoyo y dedicación.
A todos los que han contribuido de una forma u otra para el desarrollo de este trabajo.
A todas las personas que nos han apoyado y se han preocupado durante el desarrollo de nuestra carrera. Siéntanse parte de este logro.

De Julio:
A mis padres Olga y Ramón que lo han dado todo por mí, para que este sueño de todos se haga realidad.
A mis demás familiares en especial a mi tía Mary que ha estado siempre presente cuando la he necesitado.
A mis amigos, los de los viejos tiempos: Jineth, Liudmila, Ludianys, Juancito, Daniel y Chirino, también agradecer a los que hice aquí en esta maravillosa escuela: la gente linda de mi grupo, Nodalis, Niudis y a alguien también muy especial: Javier
Quiero agradecerles también a un grupo de personas que han hecho posible que esto llegue a suceder hoy como son: Yoel, Hussein, Chirino, Carlos y Walfrido.
Por último agradecerle eternamente a mi hermano y compañero de tesis Pedro, de verdad, muchas gracias por todo.

De Pedro:

A mi familia, mi mamá Idania, mi abuelo Carlos y a mi tío Carlitos por apoyarme y por confiar en mí.

A mi novia Ely por sobre todo regalarme a esa hermosa bebé que tanto me inspiró, gracias por ayudarme, apoyarme y aconsejarme.

A mi compañero de tesis y hermano Julio por trabajar tan duro, gracias por todo el ánimo que me dió, lamento tu pérdida de peso.

A mis amigos Carly, Chiri, Walfro, Nodalis, Javier y Kenny por ayudarme, gracias por poder contar con ustedes.

De Julio:

Dedico este trabajo de diploma a mi familia, en especial a mis padres por confiar en mí y apoyarme en todos los pasos importantes que he dado en la vida. A mis amigos y demás que de una forma u otra han contribuido a que llegue este día.

De Pedro:

A mi bebita Heily Karla y mi novia, este es el mejor regalo que puedo darles.

A mi familia, este es nuestro sueño por el que tanto hemos luchado.

A todos mis amigos que siempre confiaron en mí.

RESUMEN

Actualmente en los equipos de desarrollo de software que trabajan con PHP existen problemas a la hora de realizar la gestión del etiquetado de las páginas Web, entiéndase por gestión del etiquetado al proceso que comienza con la toma de decisión de poner un nuevo mensaje en una interfaz determinada y ubicar los mensajes ya puestos y reeditarlos. Este problema se debe a que las herramientas que existen para realizar este proceso trabajan con las etiquetas una vez que estas están en la base de datos, pero no hay una herramienta para ir desarrollando e ir almacenando estas etiquetas. El gestor de información cuando se dispone a colocar los mensajes en un sitio Web necesita la ayuda de un desarrollador.

Para resolver estos problemas se ha propuesto desarrollar una aplicación Web que permita el trabajo con las etiquetas, la cual garantice que los gestores de información para la edición de estas no necesiten la ayuda de los desarrolladores. La aplicación permitirá obtener paquetes de recursos para las herramientas de mayor renombre existentes para la gestión del etiquetado.

En este trabajo se hace un estudio de las principales metodologías, técnicas, lenguajes, y herramientas existentes utilizadas para la creación de sistemas Web, para poder realizar una adecuada selección de las que van a ser usadas en este trabajo. Posteriormente se realiza el análisis y diseño de la aplicación, concluyéndose con la implementación de la misma y sus pruebas correspondientes garantizándose la calidad del producto.

Palabras Claves.

Etiquetas, Gestión del Etiquetado, Gestor de Información, Paquetes de Recursos.

Introducción

En la actualidad los equipos de desarrollo de software para independizar los textos de las interfaces de las aplicaciones utilizan ficheros de recursos, en estos ficheros se colocan los mensajes de las interfaces de usuarios de las aplicaciones, y se accede a estos mediante las funciones que brinda el lenguaje de programación utilizado, o mediante librerías que faciliten este proceso.

Los equipos de desarrollo que trabajan con el lenguaje PHP tienen cierta desventaja respecto a otros de diferentes lenguajes, puesto que no cuentan con herramientas que le faciliten el trabajo respecto a estos ficheros de recursos o etiquetas como también se le conocen, solo se cuenta en la actualidad con librerías como ***Gettext*** y el motor de plantillas ***Smarty***, que son utilizadas por conocidas herramientas como ***Drupal***, u otras que acceden directamente a dichos ficheros, como ***PHPFusion***, ***Moodle***, ***MediaWiki***, por solo mencionar algunas. Estas herramientas trabajan con las etiquetas una vez estos estén en la base de datos o en cualquier otra fuente de almacenamiento, pero no permiten ir desarrollando e ir almacenándolas al mismo tiempo.

A través de una investigación realizada en varios de los proyectos productivos de la Universidad de Ciencias Informáticas (UCI), se identificaron algunos de los problemas que afectan el desarrollo de la gestión del etiquetado, entre los cuales se encuentran: el gestor de información que es el responsable del contenido informativo de las páginas Web, delega todo el trabajo con los textos al programador, los desarrolladores al no saber elegir correctamente los mensajes que desean colocar en una interfaz, se dan a la tarea de editarlos en ficheros de textos, generando el problema de que no pueden ser escritos concurrentemente; añaden además que este proceso es muy largo y tedioso y en muchas ocasiones se debe repetir cuando un gestor de información no está de acuerdo con determinados mensajes.

Estos problemas condicionan la pérdida considerable de tiempo de desarrollo de los programadores, en cuestiones como la edición de textos, y elegir mensajes adecuados para una aplicación, trabajos que en otros equipos de desarrollo que usan otros lenguajes determinados, son realizados por el gestor de información y que no lo hacen con el lenguaje PHP porque no existe ninguna herramienta que lo permita. Cuando se preguntó a los miembros de los equipos de desarrollo a los que se les realizaron las encuestas, qué opinión tenían acerca de una aplicación donde los gestores de información y los desarrolladores, puedan trabajar conjuntamente en la toma de decisiones de los mensajes del sitio, y que cada cual realice su función, manifestaron que era una magnifica propuesta, y que sin dudas solucionaría muchos problemas existentes a la hora de llevar a cabo este proceso de gestión del etiquetado.

Al analizarse tal ***situación problemática*** se determinó que el ***problema*** ante el que se encuentra este estudio es: ¿Cómo implementar una herramienta que separe los textos de las interfaces, la cual independice el trabajo de los desarrolladores y los gestores de información?

El proceso de desarrollo de software constituye el ***objeto de estudio*** de este trabajo de tesis puesto que es la ciencia que estudia este fenómeno y se ha delimitado como ***campo de acción*** el desarrollo de aplicaciones Web desarrolladas con PHP.

En este trabajo se defiende la ***idea*** siguiente: Con la creación de una herramienta que logre independizar los trabajos de los programadores y los gestores de información se ganará tiempo de desarrollo, introduciéndose mejoras en el proceso de desarrollo de software.

Este trabajo tiene como ***objetivo general*** el desarrollo de una aplicación Web para la mejora de la gestión del etiquetado. Dicha meta puede ser lograda con la culminación de los ***objetivos específicos*** siguientes:

- Estudiar sobre las herramientas de la gestión del etiquetado.
- Definir las técnicas, metodologías y herramientas que se usarán para el desarrollo de la aplicación.
- Elaborar el análisis y diseño de la aplicación.
- Implementar la aplicación y hacerle las pruebas requeridas.

Para lograr los objetivos antes mencionados este trabajo debe de cumplir el siguiente grupo de ***tareas investigativas*** que servirán de guía a para el desarrollo del mismo.

- Estudio de las principales herramientas encargadas de la gestión del etiquetado.
- Investigación sobre las principales metodologías, lenguajes, técnicas y herramientas existentes para el desarrollo de aplicaciones WEB.
- Estudio del análisis y diseño de aplicaciones Web existentes.
- Investigación de los diferentes métodos de implementación y técnicas de validación.

Métodos y técnicas de investigación a utilizar.

Métodos Teóricos:

Histórico - Lógico: Para analizar la trayectoria y evolución de la metodología de desarrollo de software y demás herramientas que se utilizan durante el trabajo.

Métodos Empíricos:

Entrevista: Para recopilar las características más importantes de los procesos del negocio al cual responde el sistema.

Revisión Bibliográfica

Su objetivo fundamental es estudiar un conjunto de fuentes de información referidas al tema, así como libros, artículos, revistas, publicaciones, boletines y una especie de materiales escritos y digitalizados, que son de gran utilidad para documentar la base teórica del trabajo a desarrollar, debidamente referenciadas para futuras consultas sobre el tema.

Este trabajo de diploma está compuesto por cuatro capítulos. El **primer capítulo**, titulado ”Fundamentación Teórica”, muestra el estudio realizado del estado del arte y la fundamentación teórica, analizando las principales herramientas para la gestión del etiquetado, se examinan las distintas metodologías para el desarrollo de software, se profundiza sobre los diferentes patrones arquitectónicos y de diseño, los lenguajes de programación más utilizados en la concepción de sistemas Web, las herramientas necesarias para el desarrollo de este proyecto y el uso de framework para facilitar el desarrollo.

El **segundo capítulo** tiene como título “Características del Sistema”, en el mismo se realiza la modelación del negocio y captura de los requisitos funcionales y no funcionales. Se describe la solución propuesta para eliminar los problemas existentes en el proceso de la gestión del etiquetado. Además se describen los procesos de negocio, casos de uso, diagramas de casos de uso del negocio, diagrama de casos de uso del sistema, actores del sistema y se describen estos casos de uso.

El **tercer capítulo** se titula “Análisis y Diseño del Sistema” en el cual se definen los modelos de análisis y diseño, además de obtenerse los diagramas de interacción correspondientes a los casos de uso del sistema.

En el **cuarto capítulo** titulado “Implementación y Pruebas” se obtiene el Modelo de Implementación, se definen los principales estándares de diseño y codificación que tendrá el sistema, se exponen los principales algoritmos utilizados y se concluye realizando las pruebas al sistema.

Capítulo I. Fundamentación teórica.

1.1. Introducción.

En este capítulo, se comienza realizando un estudio del estado del arte donde se exponen las principales deficiencias de las herramientas existentes. Se fundamentan las metodologías, técnicas, herramientas y tecnologías con las cuales se realiza el sistema.

Se define como "**proceso de gestión del etiquetado**" a la actividad que comienza con la toma de decisión de poner un nuevo mensaje en una interfaz determinada y ubicar los mensajes ya puestos y reeditarlos.

1.2. Herramientas para la gestión del etiquetado.

En este epígrafe se muestran algunas de las herramientas que se encargan de la gestión del etiquetado. Estas aunque han permitido satisfacer las necesidades básicas de este proceso, aún no resuelven correctamente la gestión de las etiquetas, pues todavía presentan desventajas en cuanto a su uso, estas herramientas se presentan a continuación:

1.2.1. Módulo Gettext.

Esta herramienta se trazó como objetivo, solucionar el problema de la gestión del etiquetado siendo una de las primeras en proponerse dicha meta, pero no logró una gran evolución en este tema.
Inconvenientes:
· Requiere instalación.
· No es orientado a objeto, lo que provoca que los equipos de desarrollo que trabajen bajo el paradigma orientado a objeto se vean afectados. [1]

1.2.2. Módulos de Pear I18N o I18Nv2.

En estos módulos de Pear hay una amplia investigación sobre los datos de internacionalización.
Inconvenientes:
- Proporcionan librerías para trabajar con los diferentes mensajes una vez estos estén en la Base de datos, pero no brinda una herramienta para ir desarrollando e ir almacenando estas etiquetas. [2]

1.2.3. Smarty.

Esta herramienta cuenta con gran aceptación por algunos equipos de desarrollo, pero estos equipos aún se aquejan de los siguientes inconvenientes:

- Los ficheros de recursos deben ser editados manualmente, lo que podría acarrear problemas de concurrencia en equipos de desarrollo que necesiten modificar textos en un mismo momento.
- Cuando se hacen modificaciones en los ficheros de recursos se deben recompilar las plantillas, labor que no en muchas circunstancias requiere la implicación de algún desarrollador. [3]

Después de revisar las peculiaridades de las herramientas que se utilizan en la actualidad para solventar los problemas del etiquetado, se pudo arribar a la conclusión que hay gran necesidad de una herramienta que permita gestionar el proceso de etiquetado.

1.3. Aplicaciones Web.

En la ingeniería software se denomina **aplicación Web** a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor Web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores Web (HTML, JavaScript, Java, etc.) en la que se confía la ejecución al navegador. Las aplicaciones Web son populares debido a lo práctico del navegador Web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software a miles de usuarios potenciales. Existen aplicaciones como los webmails, weblogs, tiendas en línea que son ejemplos bien conocidos de aplicaciones Web. Es importante mencionar que una página Web puede contener elementos que permiten una comunicación activa entre el usuario y la información. Esto permite que el usuario acceda a los datos de modo interactivo, gracias a que la página responderá a cada una de sus acciones, como por ejemplo rellenar y enviar formularios y acceder a gestores de base de datos de todo tipo.

La comunicación del cliente y el servidor es vía http/s o http, protocolo principal de comunicación en una aplicación Web, el cual funciona normalmente desconectado, es decir, el cliente hace una petición al servidor, este la procesa y le devuelve el resultado, terminando la comunicación entre estos. [7]

1.3.1. Principales ventajas de las Aplicaciones Web.

Actualmente a nivel mundial las aplicaciones Web son de gran aceptación por los usuarios con acceso a Internet, esto está basado en las múltiples ventajas que ofrecen estas aplicaciones, entre las que se destacan:

- Compatibilidad multiplataforma, ya que varias tecnologías incluyendo Java, Flash, ASP, Ajax, PHP entre otras permiten un desarrollo efectivo de programas soportando todos los sistemas operativos principales.
- Menos requerimientos de memoria, porque al residir y correr en los servidores del proveedor, esas aplicaciones basadas en Web usan en todos los casos la memoria de las computadoras en que ellas se ejecutan.
- Inmediatez de acceso. Las aplicaciones Web no necesitan ser descargadas, instaladas y configuradas.
- Múltiples usuarios concurrentes. Las aplicaciones Web pueden realmente ser utilizadas por múltiples usuarios al mismo tiempo.
- Alta disponibilidad. Se puede acceder a ellas a cualquier hora y desde cualquier parte del mundo si se tiene conexión a internet. [4]

1.4. Metodologías de desarrollo.

Como resultado de los avances con los que hoy se cuenta en el campo de la informática, se hace cada día más difícil seleccionar con precisión la metodología más adecuada que posibilite la obtención de los mejores resultados. El objetivo de un proceso de desarrollo es contar con un marco de trabajo claramente definido y estandarizado, que permita obtener productos que garanticen los requerimientos de calidad y se desarrollen en el tiempo estimado. Es labor del proceso de desarrollo hacer que esas medidas para aumentar la calidad sean reproducibles en cada desarrollo, de allí la importancia que amerita hacer una selección acertada. [5]

1.4.1. Proceso Unificado de Desarrollo (RUP).

RUP consta de cuatro etapas o fases por donde el software tiene que transitar para que obtenga la calidad requerida.

Inicio: El Objetivo en esta etapa es determinar la visión del proyecto.

Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.

Construcción: En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.

Transmisión: El objetivo es llegar a obtener el realce del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, el cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

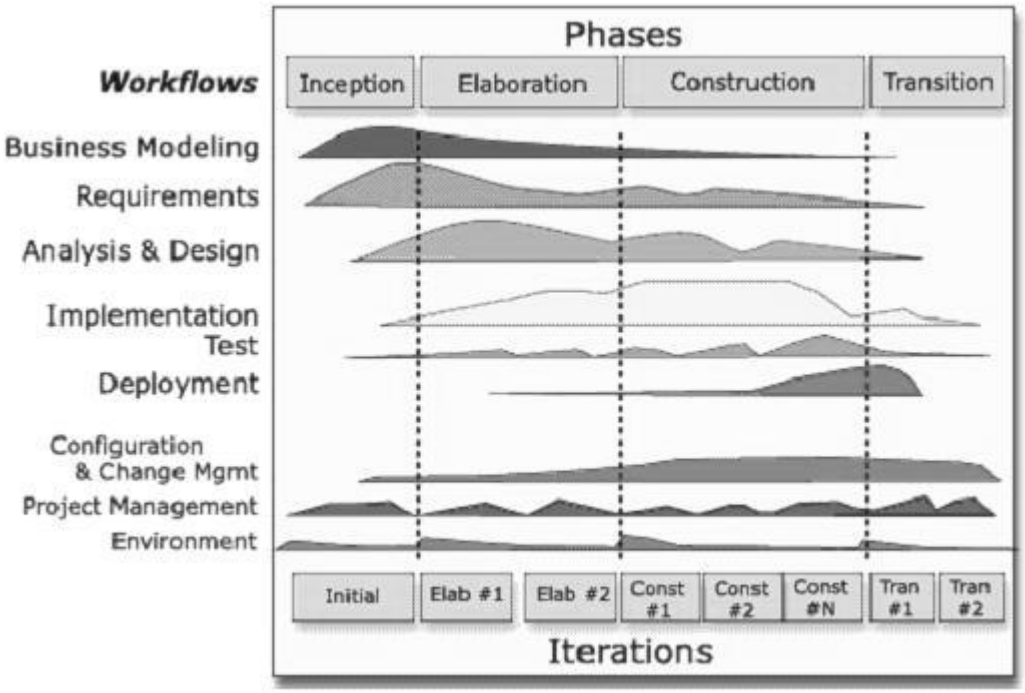


Figura 1.1 Fases de RUP.

Los ***elementos*** del RUP son:

Actividades: Son los procesos que se llegan a determinar en cada iteración.

Trabajadores: Las personas o entes involucrados en cada proceso.

Artefactos: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

El RUP presenta tres ***características*** que constituyen la esencia de todo el proceso de desarrollo las cuales son:

Dirigido por los casos de uso.

Centrado en la arquitectura.

Ciclo de vida iterativo e incremental.

Entre las ***ventajas*** de RUP, se destacan: que provee a cada miembro del equipo, un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas del desarrollo de software, además permite que todos los integrantes de un equipo de trabajo, conozcan y

compartan el proceso de desarrollo, una base de conocimientos y los distintos modelos de cómo desarrollar el software.

Otras características que constituyen además ventajas de la aplicación de esta metodología son las siguientes:

-Reconoce que las necesidades del usuario y sus requerimientos no se pueden definir completamente al principio.

-Permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema.

-Reduce el costo del riesgo a los costos de un solo incremento.

-Acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan para obtener resultados claros a corto plazo.

-Distribuye la carga de trabajo a lo largo del tiempo del proyecto ya que todas las disciplinas colaboran en cada iteración.

-Facilita la reutilización del código teniendo en cuenta que se realizan revisiones en las primeras iteraciones lo cual además permite que se aprecien oportunidades de mejoras en el diseño. [6]

1.4.2. Programación Extrema (XP).

Una metodología que como todas consta de ventajas y desventajas es la Programación Extrema (XP). De esta señalar varias cosas entre ellas que mientras que el 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. También es importante resaltar que XP intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo. Este representante debería estar en condiciones de contestar rápida y correctamente a cualquier pregunta del equipo de desarrollo de forma que no se retrase la toma de decisiones, de ahí lo de competente. En base a su opinión se definen las siguientes iteraciones del proyecto y si el cliente no está contento se adaptará el plan de raleases e iteraciones hasta que el cliente de su aprobación y el software esté a su gusto. Al contrario de otras metodologías, el código pertenece al equipo en completo, no a un programador o pareja, de forma que cada programador puede cambiar cualquier parte del código en cualquier momento si así lo necesita, dejándose en todo caso las mejoras orientadas al rendimiento para el final. Se sigue un diseño evolutivo con la siguiente premisa:

conseguir la funcionalidad deseada de la forma más sencilla posible. Estas son a grandes rasgos las principales características de XP. [8]

1.4.3. Desarrollo Guiado por Funcionalidades (FDD).

FDD se podría considerar a medio camino entre RUP y XP, aunque al seguir siendo un proceso ligero es más similar a este último. Está pensado para proyectos con tiempo de desarrollo relativamente cortos (menos de un año). Se basa en un proceso iterativo con iteraciones cortas (dos semanas) que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorizar. Un proyecto que sigue FDD se divide en 5 fases:

- Desarrollo de un modelo general.
- Construcción de la lista de funcionalidades.
- Plan de raleases en base a las funcionalidades a implementar.
- Diseñar en base a las funcionalidades.
- Implementar en bases a las funcionalidades.

Las primeras tres fases ocupan gran parte del tiempo en las primeras iteraciones, siendo las dos últimas las que absorben la mayor parte del tiempo según va avanzando el proyecto, limitándose las primeras a un proceso de refinamiento. [9]

1.4.4. Fundamentación de la Metodología a utilizar escogida.

Luego de hacer un estudio de cada una de las metodologías anteriormente planteadas se tomó la decisión de utilizar Rational Unified Process, la cual sin lugar a dudas constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. También influyó en esta decisión las facilidades que ofrece RUP en cuanto a organización, documentación que genera y sus características flexibles sobradamente probadas en la práctica, así como la experiencia existente con el uso de esta metodología en el equipo de desarrollo que realizará el producto.

1.5. Lenguajes de Modelado.

Un lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software orientado a objetos. Para la realización de este trabajo se analizarán los siguientes lenguajes de modelado.

1.5.1. Lenguaje Unificado de Modelado.

EL Lenguaje Unificado de Modelado (Unified Modeling Language - UML) es un lenguaje que proporciona un vocabulario y reglas para permitir una comunicación. Está compuesto por diversos elementos gráficos que se combinan para formar diagramas. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten, es además un método formal de modelado, aporta un mayor rigor en la especificación y permite realizar una verificación y validación del modelo realizado. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. [10]

1.5.2. Notación de Modelado de Procesos de Negocio.

La Notación de Modelado de Procesos de Negocio (Business Process Modeling Notation - BPMN) es un estándar reciente para el modelado de procesos del negocio y servicios Web, permite realizar un mejor uso de la gestión de procesos del negocio (BPM), ya que normaliza el método de notación que sirve como ayuda en la automatización de los procesos. BPMN es una notación necesaria para expresar los procesos de negocio en un único diagrama (Business Process Diagram – BPD). BPD está conformado por un conjunto de elementos gráficos. Las cuatro categorías básicas de estos elementos son: objetos de flujo, objetos de conexión, calles y artefactos.

Los objetos de flujo presentan tres elementos centrales: Evento, Actividad y Decisión. Los mismos se conectan en un diagrama para crear el esqueleto básico de la estructura de un proceso de negocio. Existen tres objetos de conexión: flujo de secuencia, flujo de mensaje y asociación.

Las calles son construidas teniendo en cuenta dos categorías diferentes, pool y lane. Pool representa un participante en un proceso y también se comporta como un contenedor gráfico para dividir un conjunto de actividades de otro pool. Lane es una sub-partición dentro de un pool y es utilizado para organizar y categorizar actividades. La versión actual de BPMN predefine sólo tres tipos de artefactos: objeto de datos, grupo y comentario. [35]

1.5.3. Fundamentación del lenguaje de modelado seleccionado.

Para la realización de este trabajo es seleccionado el Lenguaje Unificado de Modelado, ya que es el propuesto por la metodología RUP y además de presentar las siguientes ventajas: especifica todas las

decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos, también puede conectarse con lenguajes de programación (ingeniería directa e inversa). Documenta todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones) y es muy expresivo ya que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.

1.6. El lenguaje de programación PHP.

El lenguaje de programación PHP fue desarrollado originalmente por Rasmus Cerdorf en 1994 en lenguaje C, el mismo ha sido enriquecido con el transcurso de los años por la comunidad de software libre internacional, actualmente se encuentra en su versión 5, es denominado preprocesador de hipertexto (del inglés “Hypertext Pre-processor”), es un lenguaje libre y multiplataforma. Posee una amplia documentación en su página oficial posibilitando gran comprensión del mismo, se sustenta en la actualidad bajo el paradigma más difundido actualmente en el mundo que es programación orientada a objeto, incluye también la programación estructurada y servicios Web. Presenta excelente integración con todos los motores de base de datos. Cuenta con una biblioteca que trae un conjunto de funciones para realizar cualquier operación (acceso a base de datos, encriptación, envío de correo, XML, creación de PDF, entre otros). Su código es libre y se sustenta bajo la licencia GPL. PHP, es un lenguaje de programación usado normalmente para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web, es la versión libre del sistema equivalente de Microsoft ASP. Debido a la naturaleza open-source de PHP, si hay algo que actualmente no se pueda hacer en este lenguaje de programación no existe ningún impedimento para que se pueda escribir un módulo o una extensión en código C para extender la funcionalidad y hacer lo que se desee. Esto es posible por la buena documentación de la API (Interfaz de Programación de Aplicaciones) que está disponible para todos. [11]

1.6.1. Principales ventajas de PHP.

El lenguaje de programación PHP presenta una serie de ventajas, las cuales se mencionan a continuación:

- Sintaxis cómoda: PHP cuenta con una sintaxis similar a la de C, C++ o Perl. Lo más destacado ocurre a nivel semántico: el tipiado es muy poco estricto. Es decir, cuando creamos una variable no tenemos que indicar de qué tipo es, pudiendo guardar en ella datos de cualquier tipo. Esto es muy flexible y cómodo para el desarrollador, aunque los errores que se cometen pueden ser muchos más

graves y difíciles de corregir al reducirse mucho las posibilidades del intérprete para detectar incompatibilidades entre variables.

-Soporta objetos y herencia: Tiene soporte para la programación orientada a objetos, es decir, es posible crear clases para la construcción de objetos, con sus constructores, además soporta herencia, aunque no múltiple.

- Ejecución en Servidor: Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor Web justo antes de que se envíe la página a través de internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página Web con el código HTML resultante de la ejecución del código PHP compatible con todos los navegadores.

- Se puede incrustar código PHP con etiquetas HTML.

- Compatibilidad con bases de datos: Amplio soporte para una gran cantidad de bases de datos. Tiene acceso un gran número de gestores de bases de datos: Adabas D, Empress, Ingress, InterBase, FrontBase, DB2, Informix, MySQL, ODBC, Oracle, PostgreSQL, Sybase, etc.

- Se puede hacer de todo lo que se pueda transmitir por vía HTTP.

- Multiplataforma: Funciona tanto en sistemas Unix o Linux con servidor Web Apache como en sistemas Windows con Microsoft Internet Information Server, de forma que el código generado por cualquiera de éstas plataformas no debe ser modificado al pasar a la otra.

- Licencia de software libre: Es un lenguaje basado en herramientas con licencia de software libre, es decir, no hay que pagar licencias, ni existen límites en su distribución y, es posible ampliarlo con nuevas funcionalidades si así lo deseamos.

- Extensa librería de funciones: Cuenta con una extensa librería de funciones que facilitan enormemente el trabajo de los desarrolladores. [11]

1.6.2. Fundamentación de la utilización de PHP como lenguaje de programación.

Para el desarrollo de la aplicación se utilizará el lenguaje de programación PHP porque además de sus ventajas mencionadas en el epígrafe anterior, el objetivo de esta aplicación es realizar el proceso de la gestión del etiquetado a los sitios Web de los equipos de desarrollo que trabajan con este lenguaje. La selección de otro lenguaje obligaría a estos equipos a utilizar otros servidores para ejecutar esta aplicación, al utilizar PHP se permitirá que la aplicación se ejecute en los mismos nodos físicos que el sitio al que se le realizará el proceso anteriormente mencionado.

1.7. Ingeniería de Requisitos.

La Ingeniería de Requisitos es una disciplina clave en la Ingeniería de Software que forma parte de la primera fase dentro del desarrollo de un sistema informático. Estará basado en función de las necesidades planteadas por los clientes en un nivel muy general, donde se descubre, documenta, analiza y se define los servicios o componentes de lo que se desea producir, además de las restricciones que tendrá el producto o software. Su principal tarea consiste en la definición del proceso a seguir en la construcción de un software, y de facilitar la comprensión de lo que el cliente requiera. La obtención correcta de los requerimientos puede llegar a describir con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento de un sistema. Con la extracción correcta de requisitos, se pretende minimizar los problemas relacionados al desarrollo de sistemas, en proporción a la realidad de cada proyecto, con lo que se logra minimizar el tiempo en la construcción y la reducción de errores. [35]

1.7.1. Técnicas utilizadas para la captura de requisitos.

La captura de los requisitos es una actividad más humana que técnica. La ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y exacta. A continuación las principales técnicas utilizadas en la captura de requisitos.

Tormenta de ideas.

La Tormenta de Ideas es una técnica de grupo que permite la obtención de un gran número de ideas sobre un determinado tema de estudio. El principal objetivo es que los participantes muestren sus ideas de forma libre, mostrando siempre su criterio. Consiste en acumular ideas y/o información sin tomar alguna decisión aun sobre las mismas. Tiene dos características fundamentales, la **participación,** favorecen la intervención múltiple de los participantes, enfocándola hacia un tema específico, de forma estructurada y sistemática, y la **creatividad,** las reglas a seguir para su realización favorecen la obtención de ideas innovadoras. [36]

Casos de Uso.

Esta técnica es utilizada para capturar solamente los requisitos funcionales. Algunos equipos de desarrollo utilizan casos de uso como técnica para la captura de requisitos. Los casos de uso permiten mostrar el comportamiento que tendrá el sistema desde el punto de vista de los usuarios, en otras palabras el contorno representado por los actores, y el alcance representado por los requisitos funcionales expresados como casos de uso de un sistema.

1.8. Patrones.

Un patrón es una solución a un problema en un contexto, codificando conocimiento específico acumulado por la experiencia en un dominio. En términos generales, un patrón es un conjunto de información que proporciona respuesta a un conjunto de problemas similares, es decir, un patrón es una solución a un problema en un contexto, donde:

- *Contexto* son las situaciones recurrentes a las que es posible aplicar el patrón.
- *Problema* es el conjunto de metas y restricciones que se dan en ese contexto.
- *Solución* es el diseño a aplicar para conseguir las metas dentro de las restricciones.

Con la utilización de los mismos:

- Se logra una producción de software más flexible al cambio.
- Se establecen problemas pareja-solución.
- Se ayudan a especificar interfaces.
- Se promueve la reutilización del código.
- Se garantiza el uso de documentación estándar.

1.8.1. Patrones de Casos de Usos.

Los patrones de casos de uso son diseños generalmente probados en un modelo de casos de uso, junto a la descripción del contexto en el cual será usado y las consecuencias qué tendrá su aplicación en el modelo. Capturan buenas prácticas en el modelo de casos de uso, se utilizan como plantillas para saber como estructurar el modelo de casos de uso. A continuación una lista con una breve descripción de los patrones de casos de usos a tener cuenta para la realización del modelo de casos de uso del sistema:

CRUD (Creating, Reading, Updating, Deleting):

Este patrón consiste en unir casos de usos simples con fin de formar una unidad conceptual. Propone crear un caso de uso que agrupe las distintas funciones que se pueden realizar sobre una determinada parte de la información, por ejemplo: crear, leer, actualizar y eliminar.

Actores Múltiples:

También es conocido como **Rol Común** es un patrón estructural que expresa lo siguiente: cuando dos actores juegan el mismo rol sobre un caso de uso, se representa como otro actor y el resto de los actores heredan de él.

Inclusión Concreta:

Patrón estructural conformado por dos casos de uso vinculados por relación de inclusión entre el caso de uso base y el caso de uso incluido. El primero puede ser concreto o abstracto y el segundo puede ser instanciado por el mismo.

Extensión Concreta:

Patrón estructural que expresa lo siguiente: cuando dos actores están vinculados por una relación de extensión entre el caso de uso base y el caso de uso extendido, el caso de eso extendido puede ser instanciado por el mismo, además de extender el caso de uso base, este puede ser concreto o abstracto. [10]

1.8.2. Patrones Arquitectónicos.

La Arquitectura es una vista estructural de alto nivel, la cual se define muy tempranamente en el ciclo de vida y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales. El surgimiento de los patrones ha sido uno de los tres grandes temas de la ingeniería que han dominado la arquitectura de software. A continuación se presentan dos de los patrones de arquitectura más utilizados en aplicaciones Web.

1.8.2.1. Modelo Vista Controlador (MVC).

Este patrón de arquitectura está compuesto por tres capas, es una distribución fina de la secuencia de ejecución, donde se produce un evento en la capa de presentación y este rápidamente es atendido en forma completa. Las capas que componen este patrón son:

Vista: Es la encargada de mostrar los datos al usuario, es el componente que recibe un estímulo por ejemplo: (un botón en un formulario, un check box) y luego genera un evento que puede involucrar el estado de los otros controles del formulario, se encuentra enteramente en el cliente.

Modelo: Es el encargado del uso y almacenamiento de los datos, o sea la capa de acceso a datos y las entidades de negocio, es el componente asociado a entidades del dominio por ejemplo: (el cliente, la factura, sus ítems).

Controlador: Es el encargado de los procesos de negocio, este componente recibe el evento que la Vista genera y moviliza procesos del negocio que terminan cambiando el estado en el Modelo. Estos procesos de negocio suelen estar alineados a los casos de uso de la aplicación. [14]

1.8.2.2. Arquitectura por Capas.

La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario. [15]

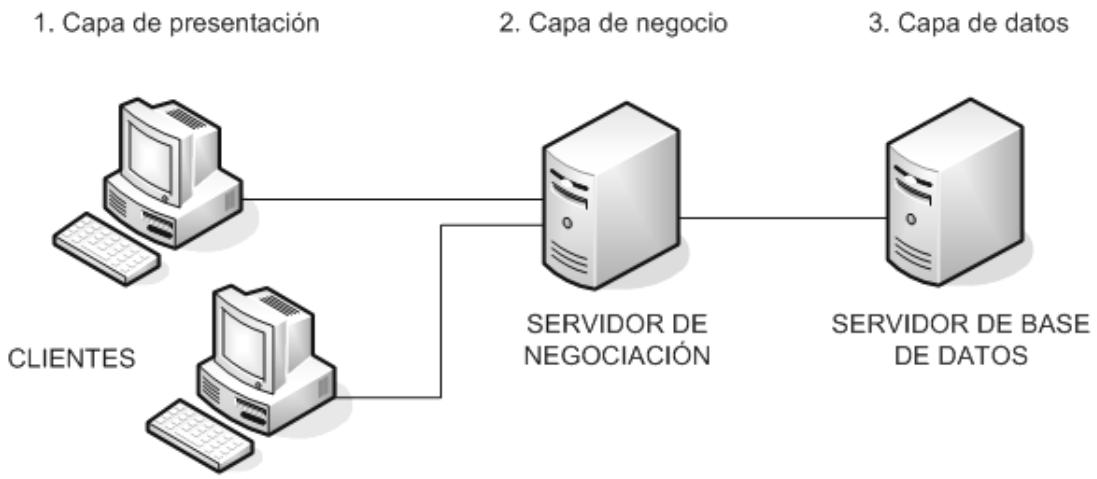


Figura 1.2 Arquitectura por Capas.

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

Capas o niveles:

Capa de presentación: es la que interactúa con el usuario (hay quien la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

Capa de datos: es donde residen los datos y es la encargada de acceder a los datos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. [15]

1.8.2.3. Fundamentación del Patrón de Arquitectura utilizado.

Para la realización del sistema se utilizará el Modelo Vista Controlador (MVC) como patrón de Arquitectura, porque el mismo es el que implementa el Codeigniter Framework utilizado como herramienta de desarrollo del sistema (véase en el epígrafe 1.11). Este patrón además presenta algunas ventajas como: proporciona bajo acoplamiento; ya que se desacopla la forma en que se muestran e introducen los datos de los modelos, brinda mayor cohesión; puesto que cada elemento del patrón se especializa en su tarea. Permite lograr mayor claridad de diseño y las vistas proveen mayor flexibilidad en la interacción con los usuarios.

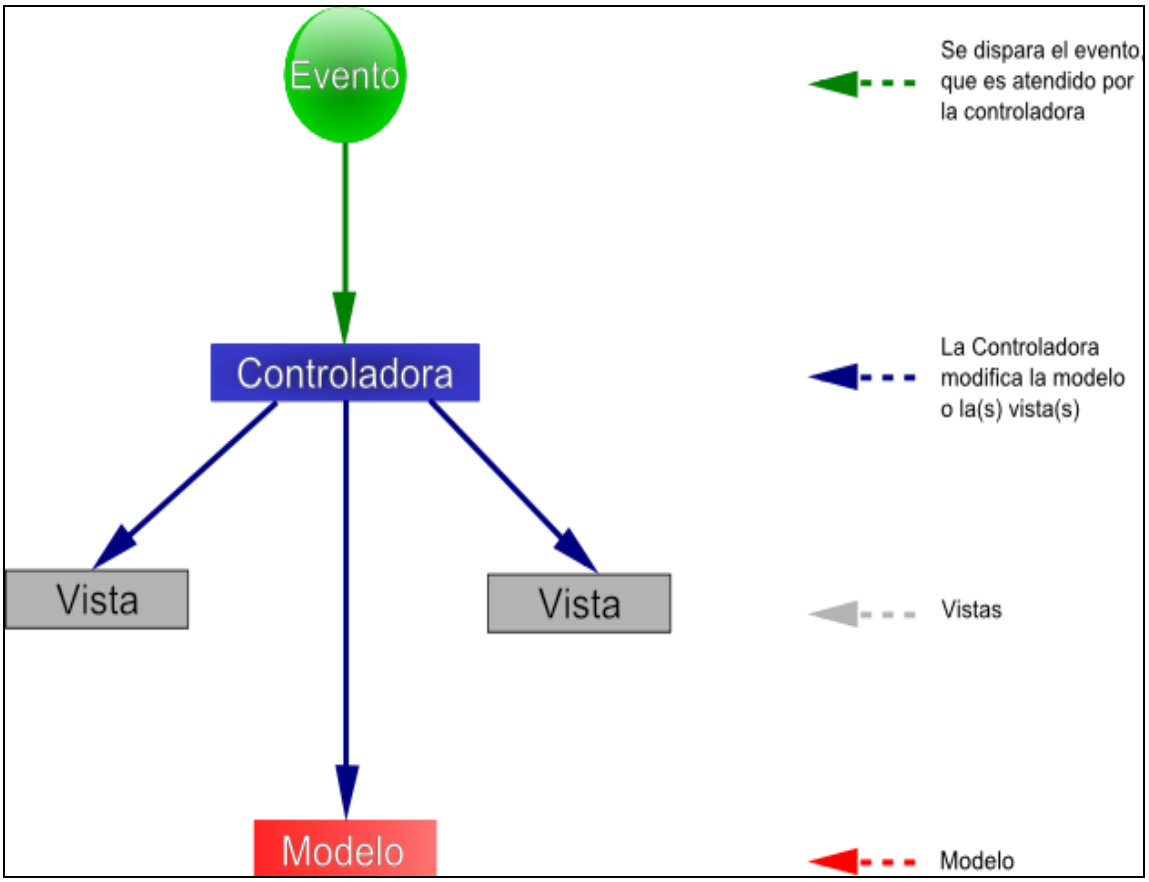


Figura 1.3 Implementación del patrón Modelo Vista Controlador según el framework Codeigniter.

1.8.3. Patrones de Diseño.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Un patrón de diseño identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades. [13]

1.8.3.1. Patrones GRASP. (General Responsibility Assignment Software Patterns).

Los patrones GRASP ó Patrones de Asignación de Responsabilidad describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades y de cierto modo ayudan a determinar las clases que estarán en el diseño. A continuación se ofrece una relación de los principales patrones GRASP y el objetivo especifico que cumple cada uno de ellos:

Experto: Cada objeto es responsable por mantener su propia información (principio de encapsulamiento) o sea que este conoce y puede informar el valor de sus atributos así como modificarlos. Si tiene relación de agregación (fuerte) con otros objetos (sus partes), también será responsable de conocer la información de ellos, de crearlos (patrón creador) y de delegarles las operaciones.

Creador: El objeto B tiene la responsabilidad de crear objetos de la clase A si:

B agrega objetos A.

B contiene objetos A.

B registra objetos A.

B usa (exhaustivamente) objetos A.

B posee la información necesaria para inicializar A.

Bajo acoplamiento: El acoplamiento es la medida de cuánto una clase esta conectada (tiene conocimiento) de otras clases por lo que éste patrón es evaluativo. Un bajo acoplamiento permite que el diseño de clases sea más independiente reduciendo el impacto de los cambios y aumentando la reutilización.

Alta cohesión: La cohesión funcional dentro de una clase es una medida que indica cuán relacionadas están las responsabilidades de una clase por lo que también éste es un patrón evaluativo. Entre más alta cohesión resulta más fácil de entender, cambiar y reutilizar.

Controlador: Consiste en Asignar la responsabilidad de manejar los mensajes eventos del sistema a una clase facilitando centralizar las actividades (tales como validación, seguridad entre otras). [10]

1.8.3.2. Patrones GoF (Gang of Four).

Patrones de creación.

Abstract Factory: proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.

Builder: separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.

Factory Method: define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
Prototype: especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.
Singleton: garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. [13]

Patrones estructurales.

Adapter: convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.
Bridge: desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente
Composite: combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.
Decorator: añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.
Facade: proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema se más fácil de usar.
Flyweight: usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.
Proxy: proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.

Patrones de comportamiento.

Chain of Responsibility: evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.
Command: encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer las operaciones.
Interpretar: dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.
Iterator: proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.
Mediator: define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

Memento: representa y externaliza el estado interno de un objeto sin violar el encapsulamiento, de forma que éste puede volver a dicho estado más tarde.
Observer: define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.
State: permite que un objeto modifique su comportamiento cada vez que cambia su estado interno, parecerá que cambia la clase del objeto.
Strategy: define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.
Template Method: define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.
Visitor: representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera. [13]

1.9. Herramientas de Modelado.

Las herramientas CASE (Computer Aided Software Engineering) son aplicaciones informáticas que sirven de gran apoyo al equipo de desarrollo en todos los aspectos del ciclo de vida de desarrollo del software, ya que reducen esfuerzo, tiempo y dinero. Con ellas se puede diseñar diagramas que ayudan a un claro entendimiento del negocio y las posibles aristas de soluciones, documentar todo el proyecto o detectar errores en fases iniciales que pueden ser arreglados a tiempo sin provocar retrasos del proyecto ni incumplimientos, entre otras facilidades que brindan. Entre las más conocidas están: [16]

1.9.1. Visual Paradigm.

La herramienta case Visual Paradigm soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. La misma ayuda en la construcción de aplicaciones con calidad y a un bajo costo. También permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, además de ser una herramienta libre y funcionar tanto en Linux como en Windows. [17]

1.9.2. Rational Rose.

Fue desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), al igual que la anterior cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. Esta

herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. Rational Rose también utiliza un proceso de desarrollo iterativo controlado (controlled iterative process development). Proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño. [18]

1.9.3. Fundamentación de la Herramienta CASE seleccionada.

Independientemente a todas las facilidades que ofrece Visual Paradigm se escogió Rational Rose como herramienta CASE para realizar el modelado visual de los procesos en el sistema debido a la mayor experiencia que se tenía en el uso de esta herramienta y a que se requieren menos requisitos de hardware. Además Rational Rose posee todas las características que los desarrolladores, analistas, y arquitectos exigen: soporte UML incomparable, completo soporte al equipo, desarrollo basado en componentes con soporte para arquitecturas líderes en la industria, facilidad de uso, integración optimizada, entre otras características que conllevan a que Rational sea reconocido como el líder tecnológico por su rol en el desarrollo del UML, lenguaje de modelado seleccionado para la realización de este trabajo.

1.10. Entornos Integrados de Desarrollo para PHP.

Los entornos integrados de desarrollo, IDE (del inglés: Integrated Development Environment) constituyen programas compuestos por un conjunto de herramientas para los programadores. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, pueden utilizarse para varios. Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación.

1.10.1. Dev-PHP.

Es uno de los destacados IDE para el lenguaje PHP, que bajo licencia GNU GPL es usado por comunidades de desarrollo por lo potente y rápido que es, al igual que por los diversos beneficios que brinda, como: navegador de clases, visor de script, integración con analizadores gramaticales, depuración de código, entre otras. El aspecto que lo deja fuera de la elección, es que solo funciona sobre plataformas de Microsoft Windows. [19]

1.10.2. Bluefish.

Es un potente editor de texto de licencia GNU GPL, muy utilizado por las comunidades de software libre para programar en gran variedad de lenguajes de programación, entre ellos el utilizado por este proyecto, PHP. Este IDE está elaborado especialmente para el desarrollo Web. Los motivos por los cuales no se decide utilizar este IDE, es que Bluefish está dirigido a diseñadores Web y programadores experimentados, además no es una herramienta multiplataforma ya que no se ejecuta sobre plataformas de Microsoft Windows. [20]

1.10.3. Zend Studio.

Es un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones Web, en lenguaje PHP. Además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. Tiene versiones para diferentes sistemas operativos Windows, Linux y MacOS. Consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene la interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración hay disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. Contiene una ayuda contextual con todas las librerías de funciones del lenguaje que asiste en todo momento ofreciendo nombres de las funciones y parámetros que deben recibir. Aunque esta ayuda contextual no solo se queda en las funciones definidas en el lenguaje, sino que también reporta ayudas con las funciones que vaya creando el programador. [21]

1.10.4. Fundamentación del entorno integrado de desarrollo seleccionado.

Debido a la popularidad del lenguaje de programación PHP existen gran cantidad de entornos de desarrollos compatibles con este lenguaje, los cuales se destacan por su potencia y competitividad, brindando un amplio espectro para la selección. Para el desarrollo de la aplicación se decidió utilizar Zend Studio, debido a que ofrece una amplia documentación, apoyada en su extensa y potente ayuda. Brinda múltiples funcionalidades como el completamiento de código, ya que cuenta con un excelente editor de texto, y además el equipo de desarrollo que realizará la implementación de la aplicación cuenta con una vasta experiencia de trabajo con el uso del entorno.

1.11. Sistemas Gestores de Base de Datos.

Los Sistemas de Gestión de Base de Datos (SGBD) son un tipo de software muy específico, dedicados a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. El propósito general de los SGBD es manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información. Existen distintos objetivos que deben cumplir los SGBD:

-*Abstracción de la información*. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

-*Independencia.* La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

-*Redundancia mínima.* Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

-*Consistencia.* En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

-*Seguridad.* Los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

-*Integridad.* Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

-*Respaldo y recuperación.* Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

-*Control de la concurrencia.* En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos y en muchos casos estos accesos ocurren de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Tiempo de respuesta. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados. [22]

1.11.1. MySQL.

MySQL es un sistema de gestión de base de datos relacional, multi-hilo y multiusuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Pertenece a Sun Microsystems desde enero de 2008. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero las empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C. Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright (derecho de copia) del código está en poder del autor individual, MySQL es propiedad y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson, y Michael Widenius. Entre sus principales ventajas se encuentran:

-*Soporta* la mayoría de los comandos del lenguaje SQL (structured query language), el estándar en bases de datos.

-*Acceso* a las bases de datos de forma simultánea por varios usuarios y aplicaciones.

-*Seguridad:* En forma de permisos y privilegios, determinados usuarios tendrán permiso para consulta o modificación de determinadas tablas. Esto permite compartir datos sin que peligre la integridad de la base de datos o protegiendo determinados contenidos.

-*Portabilidad:* SQL es también un lenguaje estandarizado, de modo que las consultas hechas usando SQL son fácilmente portables a otros sistemas y plataformas

-*Escalabilidad:* es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla.

-*MySQL* está escrito en C y C++ y probado con multitud de compiladores y dispone de APIs para muchas plataformas diferentes.

Conectividad: es decir, permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es corriente que servidores Linux o Unix, usando MySQL, sirvan datos para ordenadores con Windows, Linux, Solaris, etc. Para ello se usa TCP/IP, tuberías, o sockets Unix.

-*Es multihilo*, con lo que puede beneficiarse de sistemas multiprocesador, permite manejar multitud de tipos para columnas, permite manejar registros de longitud fija o variable. [23]

1.11.2. PostgreSQL.

PostgreSQL es un servidor de base de datos relacional orientada a objetos de software libre, como muchos otros proyectos Open Source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo.

PostgreSQL está considerado como la base de datos de código abierto más avanzada del mundo. Proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB233 u Oracle. La siguiente es una breve lista de algunas de esas características, a partir de PostgreSQL 7.1.x.

-SGBD objeto relacional: Aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas.

-Altamente extensible: Soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

-Soporte SQL: Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

-Integridad referencial: Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

-API flexible: Proporciona soporte al desarrollo fácilmente para el PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC34, Java/JDBC35, Ruby, TCL36, C/C++, y Pike entre otros.

-Lenguajes procedurales: Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido. [24]

1.11.3. Fundamentación del SGBD escogido.

PostgreSQL presenta como desventaja que por sus grandes potencialidades consume muchos recursos y carga el sistema. Para la implementación del sistema se escoge como SGBD a MySQL puesto que este gestor es muy utilizado en aplicaciones Web y su popularidad como aplicación Web está muy ligada a PHP como lenguaje a utilizar, gracias a su escalabilidad es un gestor muy rápido en la lectura y manipulación de grandes bases de datos. [24]

1.12. Framework de Desarrollo.

Un framework es diseñado con el objetivo de facilitar el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado, este puede incluir soporte de programas, bibliotecas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Permite a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

1.12.1. Framework de Zend.

Es la primera versión de su framework de desarrollo de aplicaciones para PHP de la compañía Zend. El mismo tiene como características, posee una buena documentación, estar diseñado para PHP5, y buenas capacidades de ampliación. Entre sus principales funcionalidades se encuentran:

-Proporciona funcionalidades de listas de control de acceso y gestión de privilegios.

-Proporciona un sistema de caché dividido en frontend y backend, de forma que se puedan almacenar en caché diferentes datos como resultados de funciones y páginas completas, esta información es almacenada en archivos, en memoria, en base de datos, etc.

-Simplifica la gestión de archivos de configuración y proporcionan los componentes que forma la infraestructura del patrón MVC.

-Proporciona una capa de acceso a base de datos, construida sobre PDO pero ampliándola con diferentes características y proporciona mecanismos de filtrado y validación de entradas de datos.

-Proporciona un cliente de protocolo HTTP y permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX.

-Proporciona un registro en el cual almacenar información de la aplicación que deba ser visible durante la ejecución de la petición de página, ofreciendo una forma más elegante y segura que la basada en variables globales y proporciona capacidades de búsqueda sobre documentos y contenidos.

-Permite gestionar la información de sesión y permite consumir y proveer servicios Web. [25]

1.12.2. Iguana Framework.

Iguana orientada a la creación, mantenimiento y desarrollo de sitios Web y está basada en un modelo de gestión de contenidos adaptable y extensible. Con ese fin, separa la información del sitio Web en cuatro partes: diseño, código, contenido e idioma, lo cual permite que sus equipos de desarrollo y diseño trabajen de manera paralela sin necesidad de establecer entregas parciales que afecten y condicionen cada una de las fases de desarrollo e implantación propuestas. Su funcionalidad abarca la

implantación de sitios de publicación de contenidos y noticias, foros de discusión, galerías multimedia y cuenta además con la infraestructura necesaria para el comercio en la Web (B2B, B2C). Permite al personal no técnico gestionar de forma sencilla y rápida todo tipo de contenidos. Además brinda los siguientes servicios: control de versiones, mensajería y colaboración, control de acceso, internacionalización y localización, motor de búsqueda, comercio electrónico, gestión de flujos e importación / exportación de contenido.

1.12.3. Codeigniter Framework.

Codeigniter PHP es un Framework para desarrollo de aplicaciones en PHP. Es código abierto tiene una interfaz simple y un acceso a sus librerías bien estructurado. Es liviano, bastante fácil de utilizar, altamente configurable y cuenta con un gran grupo de desarrollo. Entre sus características principales se encuentran: posee un buen rendimiento, alta compatibilidad con el estándar de las cuentas de hosting que ejecutan una gran variedad de versiones y configuraciones de PHP, evita la complejidad favoreciendo soluciones simples y cuenta con amplia documentación. [27]

1.12.4. Fundamentación del framework escogido.

Para el desarrollo de la aplicación se utilizará como framework de desarrollo el Codeigniter. Se toma esta decisión por las características y las funcionalidades que este brinda como son: Es liviano, necesita pocas configuraciones, no es necesario utilizar líneas de comando, no posee líneas de codificación y no presenta un lenguaje de plantillas.

1.13. Conclusiones.

Es de gran importancia hacer una correcta descripción de cómo se realiza el proceso de gestión del etiquetado actualmente en los equipos de desarrollo de software, por lo que se realizó un estudio profundo de la manera que se realiza este proceso y de las principales herramientas que se encargan de este trabajo. Se analizaron las principales metodologías, lenguajes y herramientas para el desarrollo de sistemas Web, concluyendo con la elección de las necesarias para el desarrollo de este trabajo.

Capítulo 2. Características del Sistema.

2.1. Introducción.

En este capítulo se ilustran conceptos fundamentales asociados al dominio del problema para lograr de esta forma un mayor entendimiento del mismo. Se analiza a profundidad el objeto de estudio haciendo especial énfasis en la descripción del flujo de eventos del problema a resolver, asegurando que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común del proceso.

2.2. Proceso de Negocio.

Se define como proceso del negocio al conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. Cada proceso de negocio tiene sus entradas, funciones y salidas. La definición del conjunto de procesos del negocio es una tarea crucial, ya que define los límites del proceso de modelado posterior.

2.2.1. Descripción General del proceso de gestión de etiquetado.

El negocio que se lleva actualmente en los equipos de desarrollo de software ha permitido satisfacer las necesidades básicas para el proceso de gestión de etiquetado, a pesar de esto existen varios aspectos deficientes y que han generado la situación problémica que se resolverá en el trabajo. El programador pide el contenido informativo al gestor de información, luego crea un conjunto de ficheros de recursos con dicha información. Para la organización de estos ficheros se colocan títulos que se refieren a la información que contendrá el fichero, y así lograr un mayor entendimiento común entre los miembros del equipo. Luego de realizar este evento el programador obtiene los paquetes de recursos (recompila los ficheros) para ser mostrados en las interfaces de usuarios. Si el gestor de información en algún momento desea cambiar el contenido guardado en estos ficheros debe comunicárselo al programador y realizar nuevamente el proceso anteriormente explicado.

2.2.2. Reglas del Negocio.

Se identificaron las siguientes reglas que debe seguir la aplicación que se desarrolle, a fin de respetar y garantizar las restricciones que existen en el negocio:

- El gestor de información es el responsable del contenido informativo del sitio Web.
- El gestor de información debe ser el encargado de editar los textos de las etiquetas.
- El desarrollador es el encargado de crear y eliminar etiquetas.
- El gestor de información no podrá crear o eliminar las etiquetas.
- El desarrollador tiene que especificar a modo de comentario en las etiquetas qué información deben contener las mismas.

2.3. Propuesta del Modelo de Negocio.

Los sistemas por muy pequeños que sean, generalmente son complicados. Por lo que se hace necesario dividirlo en piezas para su mejor comprensión y para gestionar su complejidad. El presente trabajo consiste en la implementación de una herramienta para gestionar el trabajo con las etiquetas donde se realizó un estudio detallado de cada uno de los procesos que ocurren en el proceso de gestión del etiquetado, analizando críticamente la ejecución y las actividades que se desarrollan. A partir de las necesidades, los problemas que se presentan y las deficiencias que existen, se ha planteado una propuesta de negocio.

2.3.1. Actores del Negocio.

Un Actor del Negocio representa un individuo, grupo, entidad, organización, máquina o sistema de información externos con los que el negocio interactúa. Lo que se modela como actor es el Rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. [28]

Actor del Negocio.	Descripción.
Desarrollador	Es el encargado de gestionar el trabajo con las etiquetas, es decir crear, eliminar y obtener paquetes de recursos para que sean mostrados por las interfaces.

Tabla 2.1 Actor del Negocio.

2.3.2. Trabajadores del Negocio.

Un trabajador del negocio es quien define el comportamiento y las responsabilidades de un individuo que interactúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando las entidades del negocio. [28]

Trabajador del Negocio.	Descripción.
Gestor de Información	Es el encargado de colocar toda la información contenida en el sitio Web, responsable de otorgarle dicho contenido a los desarrolladores.

Tabla 2.2 Trabajador del Negocio.

2.3.3. Diagrama de Casos de Uso del Negocio.

Un diagrama de casos de uso del negocio representa gráficamente a los procesos del negocio y su interacción con los actores del negocio.

Figura 2.1 Diagrama de Casos de Uso del Negocio.

2.3.4. Especificación del Caso de Uso del Negocio.

2.3.4.1. Caso de Uso Gestionar Etiqueta.

Caso de Uso:	**Gestionar Etiquetas.**
Actores:	Desarrollador
Trabajadores:	Gestor de información
Resumen:	El caso de uso se inicia cuando el desarrollador pide el contenido informativo del sistema para ser mostrados por la interfaz de usuario.

Tabla 2.3 Descripción del Caso de Uso ¨Gestionar Etiquetas¨.

2.3.4.2. Diagrama de Actividades.

Los casos de uso del negocio consisten en la descripción de las secuencias de actividades que, en conjunto, producen algo para el actor del negocio. El proceso consiste en un flujo básico de una o más alternativas. La estructura del mismo se describe gráficamente con la ayuda de un diagrama de actividad. [28]

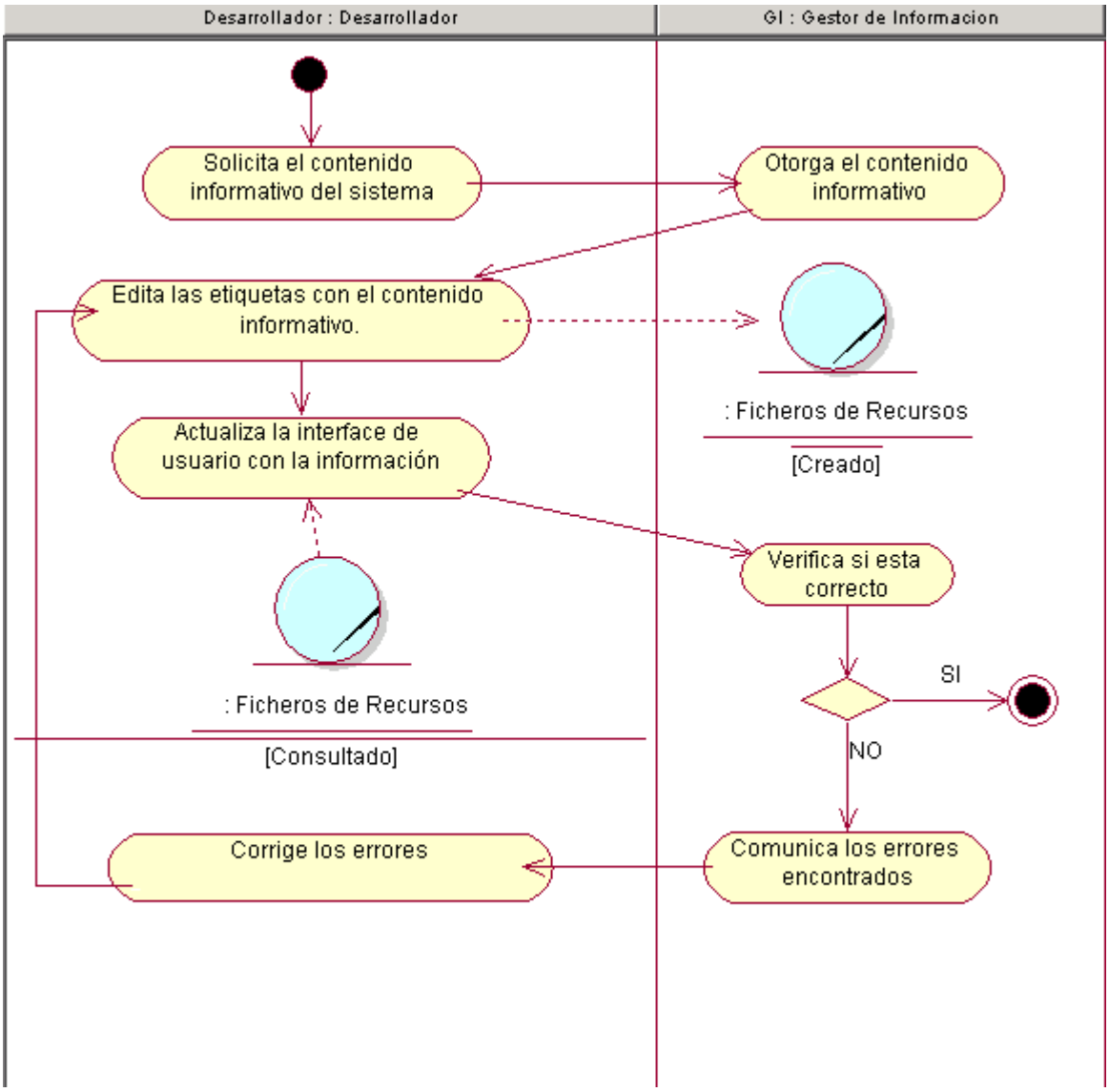


Figura 2.2 Diagrama de actividades “Gestionar Etiquetas”.

2.3.5. Modelo de objetos.

El modelo de objetos del negocio es un modelo interno de un negocio, consiste en describir cómo cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo. [28]

Figura 2.3 Modelo de Objetos.

2.4. Propuesta de Solución.

Actualmente existen problemas con la gestión de las etiquetas en los grupos de desarrollo de software que utilizan el lenguaje de programación PHP, se analizaron diferentes sistemas que actualmente se utilizan para la gestión del etiquetado, pero estos requieren siempre de la intervención de un programador por lo que el problema de la pérdida de tiempo no se logra eliminar.

En el presente trabajo se desarrollará una aplicación Web que permita que los equipos de desarrollo puedan trabajar sin detenerse en la toma de dediciones acerca de los mensajes de las interfaces, ya que dicha aplicación brinda la posibilidad de que los gestores de información puedan editar los mensajes de texto, y el programador pueda dedicarse a ordenar y agrupar estos mensajes utilizando para ello módulos. Brindando así a los equipos de desarrollo una herramienta de desarrollo colaborativo. Esta aplicación además ofrece a los equipos que la utilice la opción de generar paquetes de recursos para un gran número de herramientas de gestión del etiquetado como lo son: Gettext, Smarty 2.6.6, PHP-Fusion 6, Drupal, Moodle 1.6.

2.5. Especificación de Requisitos del Software.

El proceso de definir y especificar los requerimientos del software consiste en establecer los servicios que un producto de software debe cumplir. Proceso que lleva a la construcción de un documento conocido como Especificación de requerimientos del software, esencial para lograr un producto de alta calidad.

Obtener los Requerimientos del Software es paso esencial para la construcción de un producto de software para el posterior desarrollo de las etapas. Un error en este proceso, traería graves consecuencias en la obtención de un producto, pues podría no cumplir las expectativas del cliente. [29]

2.5.1. Requerimientos Funcionales.

Los Requerimientos Funcionales son capacidades o condiciones con las que debe cumplir el sistema a elaborar, no alteran la funcionalidad del software, se mantienen invariables sin importar con que cualidades o propiedades se relacionen. [29]

A partir del estudio y las investigaciones que se realizaron de los procesos del negocio, se obtuvieron una serie de requerimientos funcionales que ha cumplir el software:

RF 1. Iniciar Sesión. El sistema debe permitir la identificación de usuarios.

Los datos para esta operación son:

Nombre de usuario.

Contraseña.

RF 2. Obtener Listado de Etiquetas. El sistema debe permitir obtener un listado actualizado de las etiquetas.

RF 3. Obtener listas de usuarios. El sistema debe permitir obtener un listado actualizado de los usuarios.

RF 4. Gestionar Usuarios. El sistema debe permitir el trabajo con los usuarios.

Compuesto por tres escenarios:

Crear. Los datos para esta operación son:

Nombre de usuario.

Nombre.

Apellidos.

Clave.

Correo Electrónico.

Rol.

Editar. Los datos para esta operación son:

Nombre de usuario.

Nombre.

Apellidos.

Clave.

Correo Electrónico.

Rol.

Eliminar. Consiste en borrar el usuario del sistema.

RF 5. Editar Textos. El sistema debe permitir editar los textos de las etiquetas.

Textos.

RF 6. Gestionar Etiquetas. El sistema debe permitir el trabajo con las etiquetas.

Compuesto por tres escenarios:

Crear. Los datos para esta operación son:

Módulo.

Identificador.

Descripción.

Editar. Los datos para esta operación son:

Módulo.

Identificador.

Descripción.

Eliminar. Consiste en borrar la etiqueta del sistema.

RF 7. Obtener paquetes de recursos. El sistema debe permitir obtener paquetes de recursos para herramientas de gestión de etiquetado.

RF 8. Cerrar Sesión. El sistema debe permitir cerrar sesión cuando el usuario lo determine.

RF 9. Gestionar Módulos. El sistema debe permitir el trabajo con los módulos.

Los Compuesto por tres escenarios:

Crear. Los datos para esta operación son:

Nombre.

Descripción.

Editar. Los datos para esta operación son:

Nombre.

Descripción.

Eliminar. Consiste en eliminar un módulo con sus etiquetas del sistema.

RF 10. Obtener Listado de Módulos. El sistema debe permitir obtener una lista actualizada de los módulos.

2.5.2. Requerimientos no Funcionales.

Los Requisitos no Funcionales son las cualidades o propiedades que el producto debe tener. Debe pensarse en las propiedades como las características que hacen al producto o sistema atractivo, usable, confiable y fácil de usar por el usuario. [29] A continuación se muestran los no funcionales:

Apariencia.

RFN 1. La interfaz de usuario debe de ser sencilla, con un tamaño de letra mediano (arial 12) para que pueda ser observado fácilmente.

Rendimiento.

RFN 2. El sistema debe manejar toda la información relacionada con las etiquetas, la creación, eliminación y actualización de las mismas, debe efectuarse de una forma rápida (menos de 1 seg).

Usabilidad.

RFN 3. La aplicación Web debe facilitar la interacción usuario-aplicación, con el objetivo de que el usuario se sienta atraído por la tecnología evitando cualquier tipo de rechazo.

Software

Para el cliente:

RFN 4. Los usuarios tendrán acceso al sistema a través de cualquier navegador Web. Se recomienda: Firefox 3.0, Internet Explorer 6 o versiones superiores de estos.

Para el Servidor.

RFN 5. Para el funcionamiento del sistema en el servidor es necesario el Sistema Operativo Windows o Linux, Windows 98 o superior, y para Linux alguna distribución de Ubuntu u otro que tenga el usuario al alcance; además tener instalado Servidor Web AppServ.

Hardware.

RFN 6. Se necesitan como requerimientos mínimos una PC con procesador Pentium II o superior, una memoria RAM de 128 MB o superior y la conexión de red admite cualquier velocidad.

Portabilidad.

RFN 7. No es necesario especificar ningún sistema operativo para trabajar con la aplicación, ya que el producto corre sobre una Plataforma Web, codificada en PHP5 y sus sistemas de bases de datos en MySQL.

Seguridad.

RFN 8. El sistema debe controlar los diferentes niveles de acceso y funcionalidades de los usuarios en el mismo, identificando al usuario antes de que pueda realizar cualquier acción sobre el sistema.

Confidencialidad.

RFN 9. La información en el sistema estará protegida contra el acceso no autorizado, los administradores del sistema serán los encargados del control de los usuarios.

Confiabilidad.

RFN 10. El producto debe recuperarse ante cualquier fallo o error que ocurra en el sistema.

2.6. Definición de los actores del Sistema.

Un actor del sistema es una persona o la abstracción de un software que interactúa de alguna manera con el sistema: puede intercambiar información con él, ser un recipiente pasivo de información, representar el rol que juegan una o varias personas, un equipo o un sistema automatizado. [29] A continuación se definen los actores del sistema en desarrollo:

Actor	Descripción
Administrador	El administrador del sistema tiene control total sobre la aplicación y todas sus funcionalidades. Además define y controla los usuarios en el sistema, estableciendo su perfil (rol o nivel de acceso). Monitorea el trabajo de cada integrante en un equipo de desarrollo de software.
Desarrollador	Gestiona el trabajo con las etiquetas y módulos en el sistema, creándolos y eliminándolos.
	Edita los textos contenidos en las etiquetas creadas por el

Gestor de Información	desarrollador con la información correspondiente, que luego serán mostradas en las interfaces de usuarios del sitio.
Usuario	Usuario no autenticado, el cual iniciará sesión, obtendrá los paquetes de recursos y cerrará la sesión cuando lo determine.

Tabla 2.4. Definición de los actores del Sistema.

2.7. Diagrama de Casos de Uso del Sistema.

EL diagrama de Casos de Uso del Sistema es un artefacto de Ingeniería de Software que describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario, permitiendo de esta forma el establecimiento de un acuerdo entre clientes y desarrolladores sobre las condiciones y requerimientos que debe cumplir el sistema. Este modelo está formado por actores, casos de uso y las relaciones que se establecen entre estos, es decir representa gráficamente a los procesos y su interacción con los actores y constituye una entrada de gran valor para las siguientes fases de construcción de un software. [29]

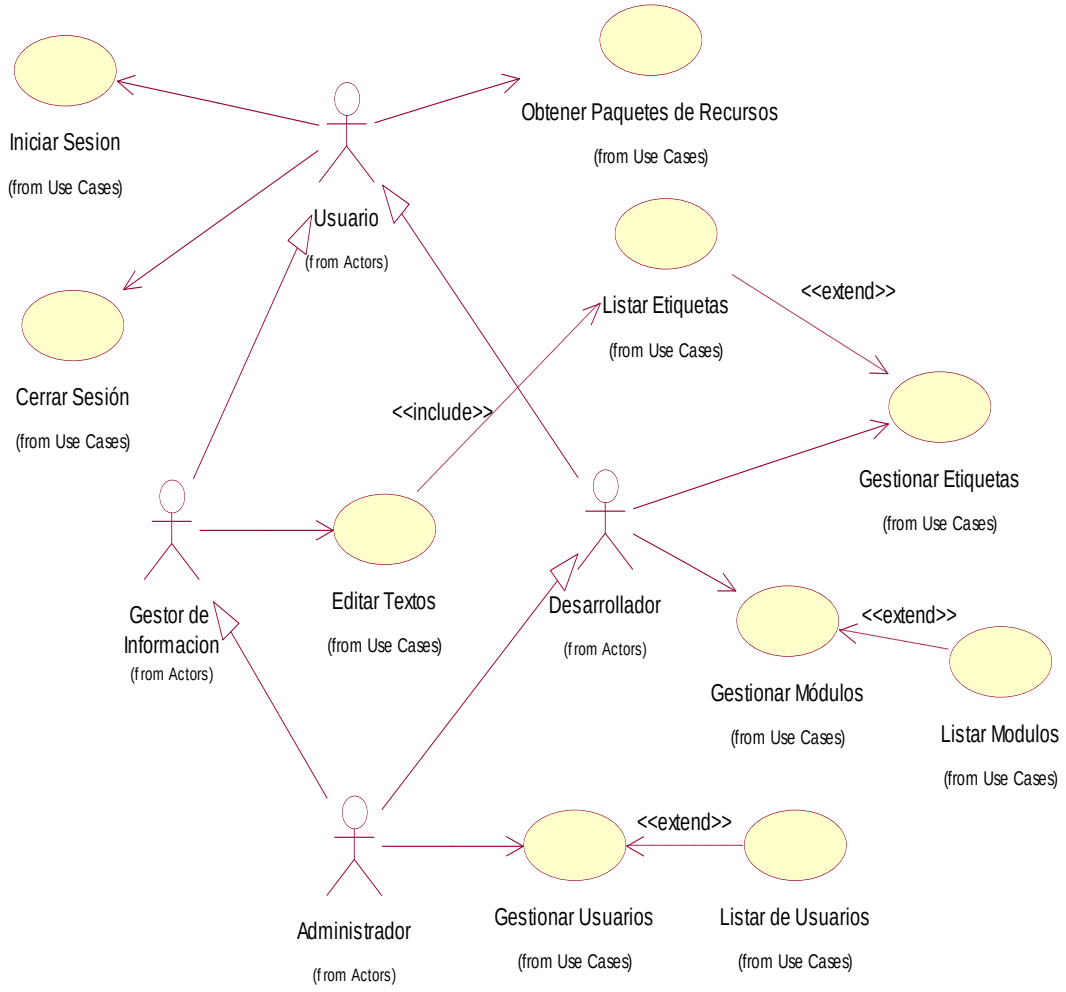


Figura 2.4 Diagrama de Casos de Uso del Sistema.

2.8. Descripción de los Casos de Uso del Sistema.

2.8.1. Descripción del Caso de Uso Gestionar Usuarios.

Caso de Uso:	Administrar Usuarios.
Actores:	Administrador.

Resumen:	El caso de Uso se inicia cuando el Administrador de Sistema necesita agregar, modificar o eliminar un usuario del sistema, ya sea un programador, gestor de información u otro administrador.
Precondiciones:	1.- Autenticación en la aplicación como administrador del sistema para que sea autorizado a acceder a esta funcionalidad. 2.- Seleccionar la funcionalidad usuarios para que el sistema muestre la lista actual de los usuarios existentes y las opciones eliminar, editar y crear.
Referencias	**RF 4 y RF 3.**
Prioridad	**Crítico.**

Flujo Normal de Eventos	
Sección "Crear"	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona crear usuarios. 3.- Introduce los datos del usuario.	2.- Muestra una interfaz con los campos para agregar usuarios. 4.- Guarda los datos del usuario en la base de datos. 5.- Muestra un mensaje de notificación, informando que el usuario fue creado satisfactoriamente.
Prototipo de Interfaz: Ver Anexo 4.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona crear usuarios. 3.- Introduce los datos del usuario.	2.- Muestra una interfaz con los campos para agregar usuarios. 4.- Muestra un mensaje de error, cometido al introducir los datos, especifica el campo donde sucedió el error.
Prototipo de Interfaz: Ver Anexo 4.	

Poscondiciones	Permite que los campos sean llenados nuevamente.
Flujo Normal de Eventos	
Sección "Editar"	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona el usuario que va a ser editado. 3.- Introduce los datos del usuario.	2- Muestra una interfaz con los campos a editar. 4.- Guarda los datos del usuario en la base de datos. 5.- Muestra un mensaje de notificación, informando que el usuario fue editado satisfactoriamente.
Prototipo de Interfaz: Ver Anexo 4.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona el usuario que desea editar. 3.- Introduce los datos del usuario.	2.- Muestra una interfaz con los campos para editar usuarios. 4.- Muestra un mensaje de error, cometido al introducir los datos, especifica el campo donde sucedió el error.
Prototipo de Interfaz: Ver Anexo 4.	
Poscondiciones	Permite que los campos sean llenados nuevamente.
Flujo Normal de Eventos	
Sección "Eliminar"	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona el usuario que desea eliminar. 3.- Acepta el mensaje.	2.- Muestra un mensaje de confirmación. 4.- Elimina el usuario de la base de datos. 5.- Muestra un mensaje de notificación, informando que

	el usuario fue borrado satisfactoriamente.
Prototipo de Interfaz: Ver Anexo 4.	
Poscondiciones	Muestra la lista actualizada de usuarios del sistema.

Tabla 2.5 Descripción del Caso de Uso Administrar Usuarios.

2.8.2. Descripción del Caso de Uso Iniciar Sesión.

Caso de Uso:	Iniciar Sesión.
Actores:	Usuario
Resumen:	El caso de Uso se inicia cuando se ejecuta la aplicación.
Precondiciones:	Ejecutar la aplicación (poner la dirección en el navegador).
Referencias	**RF 1.**
Prioridad	**Crítico.**
Flujo Normal de Eventos	
Sección "Autenticarse".	
Acción del Actor	**Respuesta del Sistema**
1.- Cualquier usuario ejecuta el sistema. 3.- El usuario introduce los datos requeridos.	2.- Muestra un formulario de autentificación, donde aparecerán campos para llenar como: usuario, contraseña. 4.- Valida los datos, verifica si los datos son correctos en la base de datos. 5.- Si los datos son correctos muestra una interfaz de acuerdo al rol las funcionalidades que podrá realizar.
Prototipo de Interfaz: Ver Anexo 4.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

1.- Cualquier usuario ejecuta el sistema. 3.- El usuario introduce los datos requeridos.	2.- Muestra un formulario de autentificación, donde aparecerán campos para llenar como: usuario, contraseña. 4.- Valida los datos, verifica si los datos son correctos en la base de datos. 5.- Si los datos son incorrectos se mostrará un mensaje de error cometido en la entrada de datos.
Prototipo de Interfaz: Ver Anexo 4.	
Poscondiciones	Mostrará nuevamente la interfaz para la autentificación.

Tabla 2.6 Descripción del Caso de Uso Iniciar Sesión.

2.8.3. Descripción del Caso de Uso Editar Textos.

Caso de Uso:	Editar Textos.
Actores:	Gestor de Información.
Resumen:	Se inicia cuando un Gestor de Información se dispone a insertar la información que tendrán las etiquetas.
Precondiciones:	El usuario tiene que estar autenticado como gestor de información para aparezca esta funcionalidad. Debe de existir en el sistema al menos una etiqueta para que sea editado su texto. Debe seleccionar la funcionalidad etiquetas para que el sistema muestre la lista actual de etiquetas existentes.
Referencias	**RF 5 y RF 2.**
Prioridad	**Crítico.**
Flujo Normal de Eventos	
Sección "Editar Texto".	

Acción del Actor	Respuesta del Sistema
1.- El gestor de información selecciona una etiqueta. 3.- El gestor de información introduce el texto en la etiqueta.	2.- Muestra una interfaz (editor de texto) con los datos de la etiqueta seleccionada. Estos datos se mostrarán a modo de información, el único campo que podrá cambiar será (texto). 4.- El sistema actualiza el contenido del texto en la base de datos. 5. Muestra un mensaje de notificación
Prototipo de Interfaz: Ver Anexo 4.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	Se almacenará la etiqueta con el texto editado.

Tabla 2.7 Descripción del Caso de Uso Editar Textos.

2.8.4. Descripción del Caso de Uso Gestionar Etiquetas.

Caso de Uso:	Gestionar Etiquetas.
Actores:	Desarrollador.
Resumen:	Se inicia cuando un desarrollador se dispone a realizar alguna operación sobre los etiquetas, ya sea crear, eliminar o editar (puede ser cambiar la etiqueta hacia otro módulo).
Precondiciones:	1.- El usuario tiene que estar autenticado como desarrollador para aparezca esta funcionalidad. 2.- Seleccionar la funcionalidad etiquetas para que el sistema muestre la lista actual de los etiquetas.
Referencias	**RF 6 y RF 2.**
Prioridad	**Crítico.**

Flujo Normal de Eventos	
Sección "Crear"	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la opción crear etiqueta. 3.- Introduce los datos de la etiqueta a crear. .	2.- El sistema muestra una interfaz para crear etiquetas con los campos a llenar. 4.- Guarda la etiqueta en la base de datos. 5.- Muestra un mensaje de notificación, informando que la etiqueta fue creado satisfactoriamente.
Prototipo de Interfaz: Ver Anexo 4.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la opción crear etiqueta. 3.- Introduce los datos de la etiqueta a crear. .	2.- El sistema muestra una interfaz para crear etiquetas con los campos a llenar 4.- Muestra un mensaje de error, cometido al introducir los datos, especifica el campo donde sucedió el error.
Prototipo de Interfaz: Ver Anexo 4.	
Poscondiciones	Permite que los campos sean llenados nuevamente.
Flujo Normal de Eventos	
Sección "Editar"	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la etiqueta que va a ser editada. 3.- Introduce los datos de la etiqueta a editar.	2- El sistema muestra una interfaz para editar etiquetas con los campos a llenar. 4.- Guarda los datos de la etiqueta en la base de

	datos. 5.- Muestra un mensaje de notificación, informando que los datos de la etiqueta han sido guardados satisfactoriamente.
Prototipo de Interfaz: Ver Anexo 4.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la etiqueta que va a ser editada. 3.- Introduce los datos de la etiqueta.	2.- Muestra una interfaz con los campos para editar etiquetas. 4.- Muestra un mensaje de error, cometido al introducir los datos, especifica el campo donde sucedió el error.
Prototipo de Interfaz: Ver Anexo 4.	
Poscondiciones	Permite que los campos sean llenados nuevamente.
Flujo Normal de Eventos	
Sección "Eliminar"	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la etiqueta a eliminar. 3.- Acepta el mensaje.	2.- Muestra un mensaje de confirmación. 4.- Elimina la etiqueta de la base de datos. 5.- Muestra un mensaje de notificación, informando que la operación se realizó satisfactoriamente.
Prototipo de Interfaz: Ver Anexo 4.	
Poscondiciones	Muestra la lista actualizada de etiquetas del sistema.

Tabla 2.8 Descripción del Caso de Uso Gestionar Etiquetas

2.8.5. Descripción del Caso de Uso Obtener Paquetes de Recursos.

<table>
<tr><td>Caso de Uso:</td><td colspan="2">Obtener Paquetes de Recursos.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario.</td></tr>
<tr><td>Resumen:</td><td colspan="2">Se inicia cuando cualquier usuario del sistema se dispone a obtener los paquetes de recursos para la herramienta deseada.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">El usuario tiene que estar autenticado, debe existir al menos un texto para ser generado.</td></tr>
<tr><td>Referencias</td><td colspan="2">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 “Paquetes de Recursos”.</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1.- Selecciona la opción obtener paquetes de recursos.

3.- Selecciona la herramienta.

5.- Guarda el archivo en la dirección deseada</td><td>2.- Muestra una lista con las herramientas.

4.- Obtiene los paquetes de recursos y le da la opción de descarga en un archivo compactado.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz: Ver Anexo 4.</td></tr>
<tr><td colspan="3">Flujos Alternos</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2"></td><td></td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
<tr><td>Poscondiciones</td><td colspan="2">En la dirección especificada se guardará una carpeta con los paquetes de recursos.</td></tr>
</table>

Tabla 2.9 Descripción del Caso de Uso Obtener Paquetes de Recursos.

2.8.6. Descripción del Caso de Uso Gestionar Módulos.

Caso de Uso:	Gestionar Módulos.
Actores:	Desarrollador.
Resumen:	Se inicia cuando un desarrollador se dispone a trabajar con los módulos, necesarios para agrupar las etiquetas, ya sea crear, modificar o eliminar uno de estos.
Precondiciones:	1.- El usuario tiene que estar autenticado como desarrollador para aparezca esta funcionalidad. 2.- Seleccionar la funcionalidad módulos para que el sistema muestre la lista actual de los módulos existentes.
Referencias	**RF 9 y RF 10.**
Prioridad	**Crítico.**

Flujo Normal de Eventos	
Sección "Crear"	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la opción crear módulo. 3.- Introduce los datos del módulo a crear. .	2.- El sistema muestra una interfaz para crear módulos con los campos a llenar. 4.- Guarda el módulo en la base de datos. 5.- Muestra un mensaje de notificación, informando que el módulo fue creado satisfactoriamente.
Prototipo de Interfaz: Ver Anexo 4.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la opción crear módulo. 3.- Introduce los datos del módulo a crear.	2.- El sistema muestra una interfaz para crear módulos con los campos a llenar

.	4.- Muestra un mensaje de error, cometido al introducir los datos, especifica el campo donde sucedió el error.
Prototipo de Interfaz: Ver Anexo 4.	
Poscondiciones	Permite que los campos sean llenados nuevamente.
Flujo Normal de Eventos	
Sección "Editar"	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona el módulo que va a ser editado, 3.- Introduce los datos del módulo a editar. ..	2- El sistema muestra una interfaz para editar módulos con los campos a llenar. 4.- Guarda los datos del módulo en la base de datos. 5.- Muestra un mensaje de notificación, informando que los datos del módulo han sido guardados satisfactoriamente.
Prototipo de Interfaz: Ver Anexo 4.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona el módulo que va a ser editado. 3.- Introduce los datos del módulo.	2.- Muestra una interfaz con los campos para editar módulos. 4.- Muestra un mensaje de error, cometido al introducir los datos, especifica el campo donde sucedió el error.
Prototipo de Interfaz: Ver Anexo 4.	
Poscondiciones	Permite que los campos sean llenados nuevamente.
Flujo Normal de Eventos	
Sección "Eliminar"	
Acción del Actor	**Respuesta del Sistema**

1.- Selecciona el módulo a eliminar. 3.- Acepta el mensaje.	2.- Muestra un mensaje de confirmación, con la lista de etiquetas que contiene el módulo. 4.- Elimina el modulo y las etiquetas pertenecientes a este de la base de datos. 5.- Muestra un mensaje de notificación, informando que la operación se realizó satisfactoriamente.
Prototipo de Interfaz: Ver Anexo 4.	
Poscondiciones	Muestra la lista de módulos del sistema.

Tabla 2.10 Descripción del Caso de Uso Obtener Paquetes de Recursos.

La descripción de los restantes casos de uso se encuentra**:**

Anexo 5.

Descripción del Caso de Uso Cerrar Sesión.

Descripción del Caso de Uso Listar Etiquetas.

Descripción del Caso de Uso Listar Módulos.

Descripción del Caso de Uso Listar Usuarios.

2.9. Validación de los requerimientos.

El principal propósito de este análisis es validar que todos los requerimientos fueron recogidos en al menos un caso de uso así como demostrar la ausencia de ambigüedades de los requisitos capturados. El equipo de desarrollo hizo una revisión exhaustiva de cada requerimiento teniendo en cuenta todos los detalles y determinó que la especificación de requerimientos estaba totalmente terminada, por recoger todas las funcionalidades requeridas ya sean primarias, secundarias, auxiliares y opcionales para el completo funcionamiento de la aplicación.

2.9.1. Métrica para la calidad de especificación de los requisitos.

Se aplicará la métrica que mide la especificidad de los requisitos, haciendo que estos se puedan entender de una manera fácil y se puedan probar.

Q1 = nui / nr donde,

Q1: es un valor que cuanto más cerca esté de 1 menor será la ambigüedad de la especificación.

nui: es el número de requisitos para los que todo el equipo de desarrollo tuvieron la misma interpretación.

En este caso nui = 20, pues para todos los requisitos el equipo de desarrollo tuvo la misma interpretación.

nr: es la cantidad de requisitos en una especificación, y se calcula usando la formula:

nr = nf + nnf donde,

nf : es el número de requisitos funcionales.

nf = 10

nnf: es el número de requisitos no funcionales.

nnf = 10

Calculando el valor de nr.

nr = nf + nnf

nr = 10 + 10

nr = 20

Calculando el valor de Q1.

Q1 = nui / nr

Q1 = 20 / 20

Q1 = 1

Una vez aplicada esta métrica se determinó que los requisitos no eran ambiguos, ya que el equipo de desarrollo siempre interpretó de igual forma el significado de los requisitos y por lo tanto el valor que tomaría Q1 en este caso es 1. [30]

2.9.2. Métrica para determinar el número de requisitos que no son considerados en ningún CU.

El propósito de esta métrica es confirmar que cada uno de los requerimientos funcionales han sido registrados en al menos un caso de uso que representará su funcionalidad. Para esto se ha tomado la especificación de los requisitos y el modelo de casos de uso del sistema y utilizando la técnica conocida como QFD (Quality Function Deployment) se verificó que todo requerimiento podrá ser implementado a través de algún caso de uso, y que todo caso de uso satisface algún requerimiento. El esquema QFD (Quality Function Deployment) es una matriz que representa las tasas de calidad, en las cuales las filas

representan los “qué”, o sea, la lista de requerimientos, mientras que las columnas representan los “cómo”, es decir, como se llevan a cabo los requerimientos en casos de uso. [30]

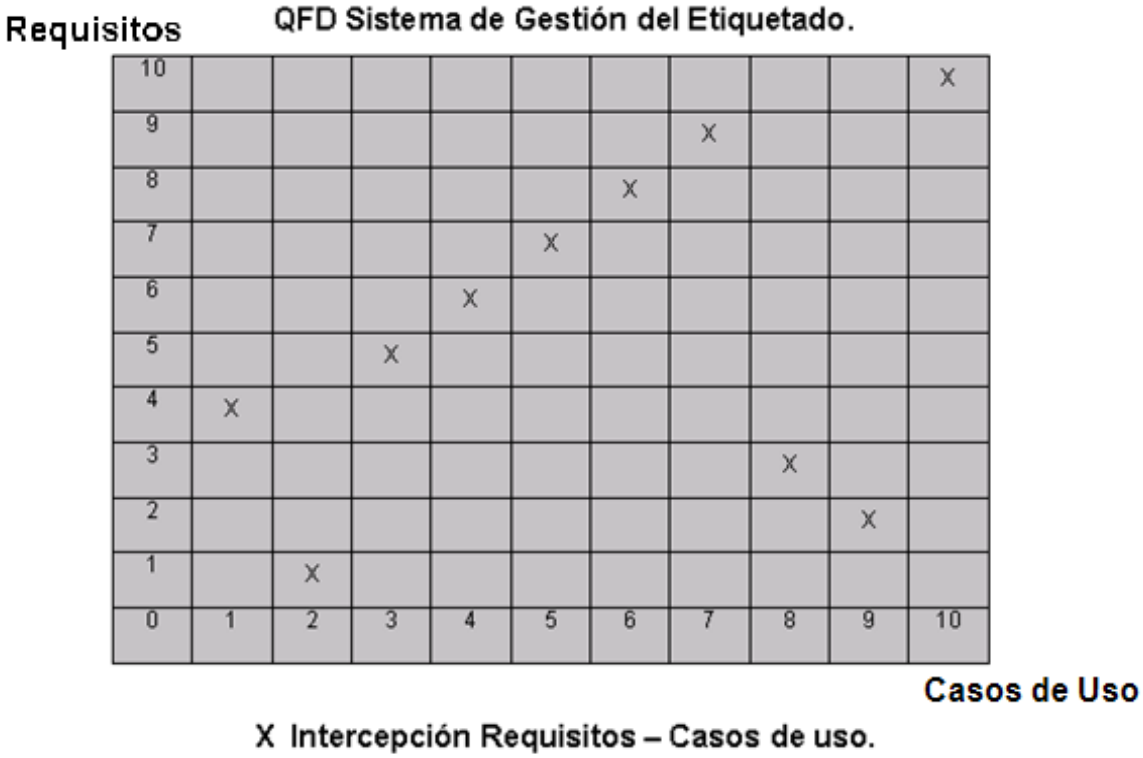


Figura 2.5 Esquema QFD.

Al aplicar el esquema QFD (Quality Function Deployment) a los casos de uso del sistema, se comprobó que todos requerimientos tienen al menos una cruz marcada, por lo que estos están incluidos en al menos un caso de uso, lo que constituye el objetivo de esta métrica.

2.10. Conclusiones.

Este capítulo constituye un paso de avance en el desarrollo del presente trabajo puesto a que, al obtener los artefactos correspondientes al Modelo de Negocio, se crearon las condiciones de entrada al Modelo del Sistema. Se realizó la captura de los requisitos, los cuales darán soporte a las funcionalidades y cualidades que presentará la aplicación a desarrollar. Se realizó la descripción de los casos de uso del

sistema mostrando a un nivel mas elevado las operaciones que realizará la aplicación y se verificaron tanto los requisitos como los casos de uso al aplicarles métricas para demostrar la validez de los mismos.

Capítulo 3. Análisis y Diseño del Sistema.

3.1. Introducción.

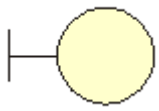
En este capítulo se realiza el análisis y diseño del sistema que se desarrollará. Se utilizan herramientas y metodologías que incluyen principios y diagramas que muestran la forma en que los componentes de software interactúan y se comportan dentro del sistema. Además se tratan temas específicos sobre los modelos de análisis, diseño y sobre su construcción como una propuesta RUP para el desarrollo del software; todo esto para lograr al final un eficiente diseño del producto.

3.2. Modelo de análisis.

En la construcción del modelo de análisis se tienen que identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye el Diagrama de clases del análisis, que por lo general se descompone para agrupar las clases en paquetes. Esta descomposición tiene impacto por lo general en el diseño e implementación de la solución. [31]

3.2.1. Clases de análisis.

Se basan en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos. RUP propone clasificar a las clases en: [31]

	**Clase Interfaz:** Modela la interacción entre el sistema y sus actores.
	**Clase Control:** Coordina la realización de uno o unos pocos casos de uso coordinando la actividades de los objetos que implementan la funcionalidad del caso de uso.

	**Clase Entidad:** Modela la información que posee larga vida y que es a menudo persistente.

Tabla 3.1. Clases del Análisis.

3.2.2. Modelo de clases del análisis.

Es un modelo de objetos que describe la realización de los Casos de Uso, y cuales sirven como abstracción del artefacto: Modelo de diseño. El modelo de análisis contiene los resultados del análisis de los casos de uso, instancias del artefacto: clases del análisis. En adición a esto, en el modelo de análisis es donde se refinan los requisitos, no se toma en cuenta el lenguaje de programación a usar en la construcción, ni se define la plataforma en la que se ejecutará la aplicación ya que el objetivo principal del análisis es garantizar una total comprensión de los requisitos del software y no precisar cómo se implementará la solución propuesta. Para la construcción del modelo de análisis se tienen que identificar las clases que describe la realización de los casos de uso y sus relaciones entre ellas y con esta información se desarrollan los Diagramas de Clases del Análisis para el sistema objeto de estudio.[31]

3.2.3. Diagrama de clases del análisis.

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas cosas. Son específicamente unos diagramas estáticos que muestran qué es lo que interactúa, pero no cómo interactúa.

A continuación se muestra algunos de los diagramas de clases del análisis, elaborados específicamente para desarrollar la realización de los casos de uso del sistema. [31]

3.2.3.1. Diagrama de clases del análisis del CU Obtener Paquetes de Recursos.

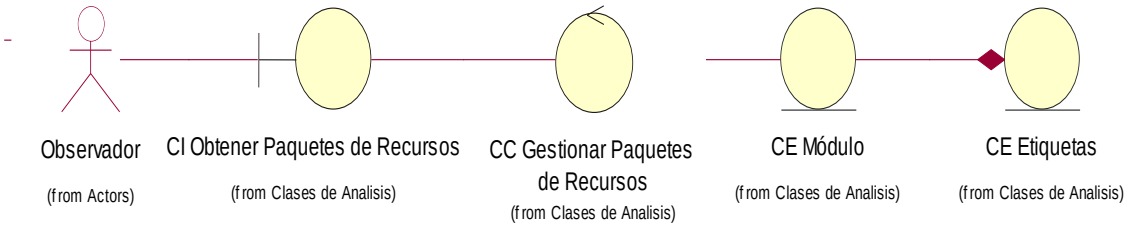


Figura 3.1. Diagrama de clases del análisis del Caso de Uso Obtener Paquetes de Recursos.

3.2.3.2. Diagrama de clases del análisis del CU Editar Textos.

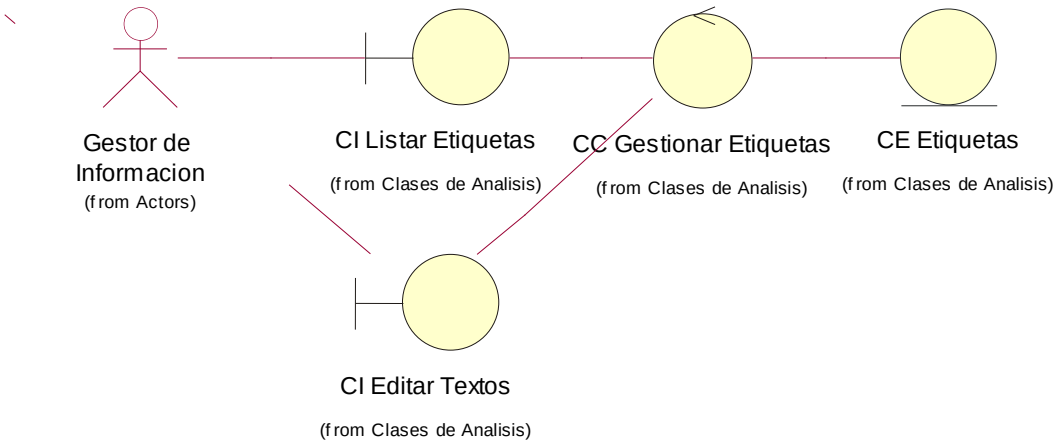


Figura 3.2 Diagrama de clases de análisis para el caso de Uso Editar Textos.

3.2.3.3. Diagrama de clases del análisis para el CU Gestionar Etiquetas.

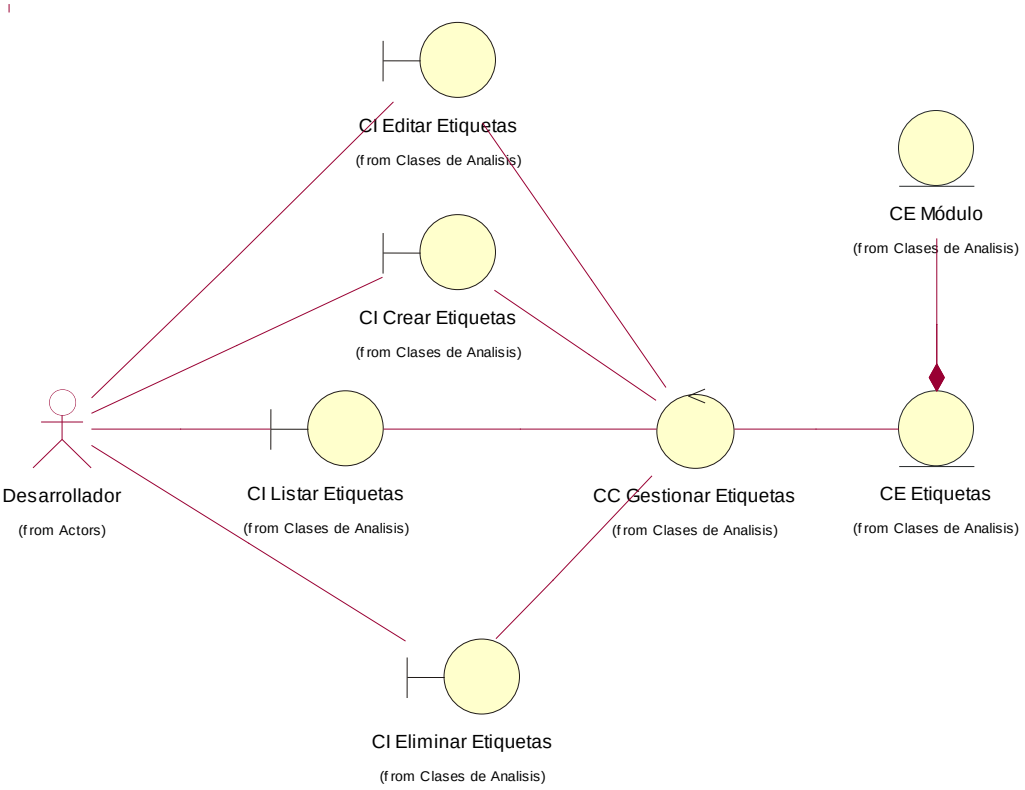


Figura 3.3 Diagrama de clases de análisis para el caso de Uso Gestionar Etiquetas.

3.2.3.4. Diagrama de clases del análisis para el CU Gestionar Módulos.

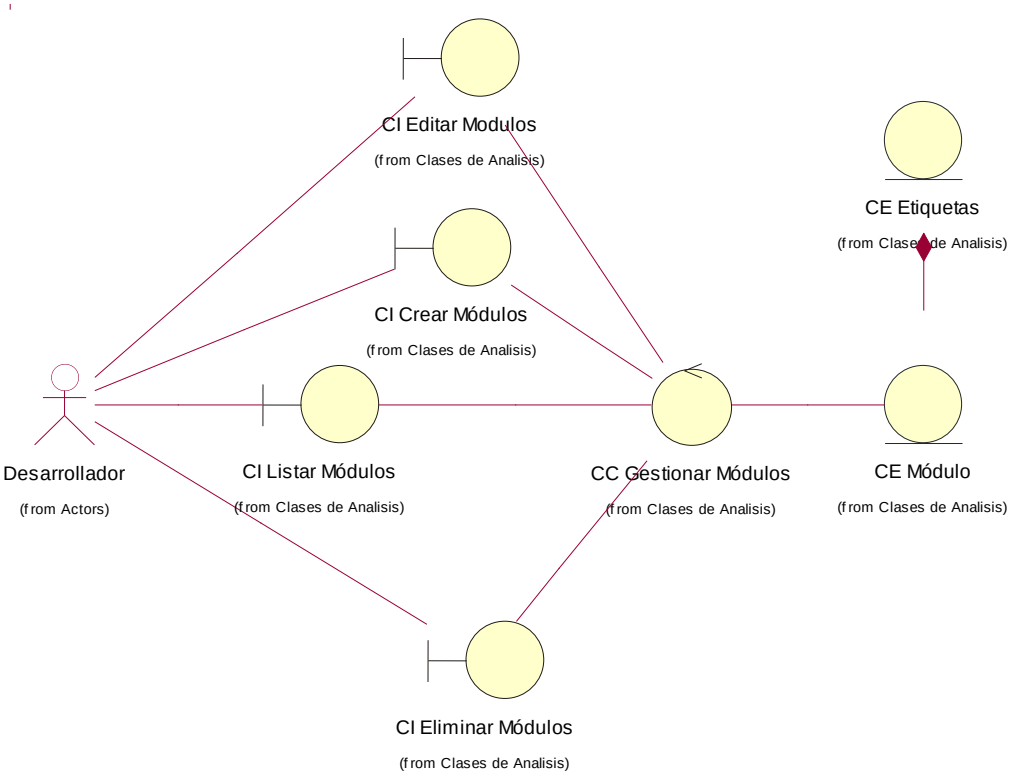


Figura 3.4 Diagrama de clases de análisis para el Caso de Uso Gestionar Módulos.

Los demás diagramas de clases de análisis se encuentran:

Anexo 2:

Diagrama de clases de análisis para el Caso de Uso Administrar Usuarios.

Diagrama de clases de análisis para el Caso de Uso Iniciar Sesión.

3.3. Diagramas de Interacción.

Los diagramas de interacción muestran una interacción concreta: un conjunto de objetos y sus relaciones, junto con los mensajes que se envían entre ellos. Modelan el comportamiento dinámico del sistema; el flujo de control en una operación. Describe la interacción entre objetos, los objetos interactúan a través de

mensajes para cumplir ciertas tareas. Las interacciones proveen un comportamiento y típicamente implementan un Caso de Uso. [32]

Existen dos tipos de diagramas de interacción en UML:

3.3.1. Diagrama de Colaboración (dimensión estructural).

Útiles en la fase exploratoria para identificar objetos. Permiten representar una disposición espacial de la estructura de los objetos en ejecución. La estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes por los enlaces. [32]

3.3.1.1. Diagrama de colaboración del Caso de Uso Gestionar Módulos.

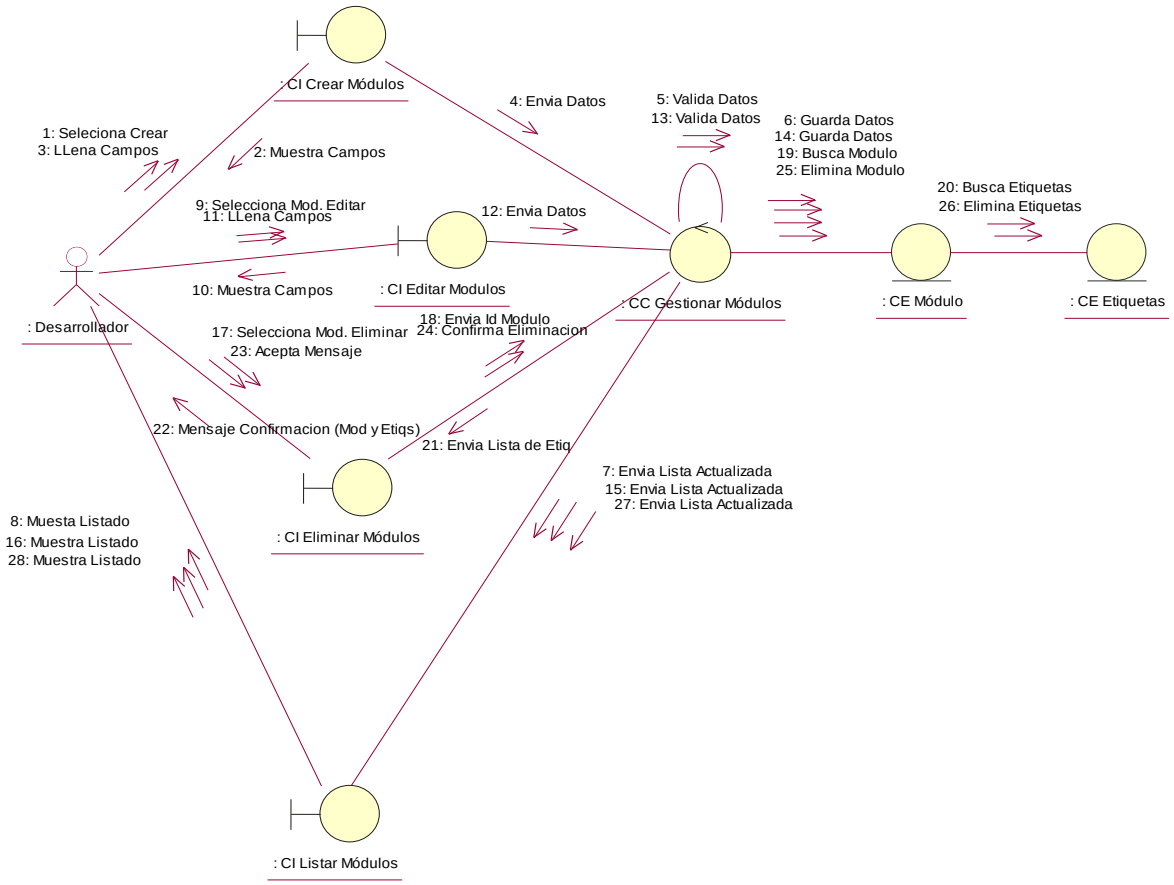


Figura 3.5 Diagrama de colaboración del caso de uso Gestionar Módulos.

3.3.1.2. Diagrama de Colaboración Gestionar Etiquetas.

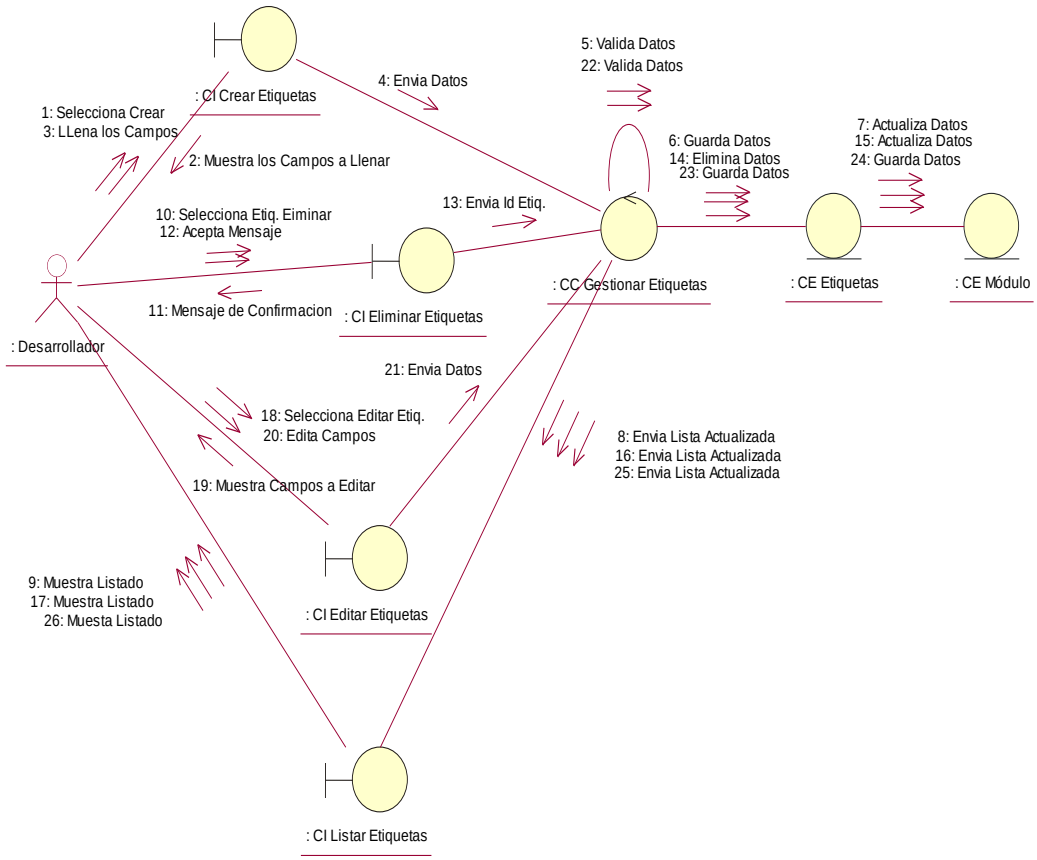


Figura 3.6 Diagrama de colaboración del Caso Uso Gestionar Etiquetas.

Los demás diagramas de colaboración se encuentran:

Anexo 3:

Diagrama de colaboración del Caso Uso Iniciar Sesión.

Diagrama de colaboración del Caso Uso Administrar Usuarios.

Diagrama de colaboración del Caso Uso Obtener Paquetes de Recursos.

Diagrama de colaboración del Caso Uso Gestionar Editar Textos.

3.3.2. Diagramas de Secuencia (dimensión temporal).

Muestra la secuencia cronológica de mensajes entre objetos durante un escenario concreto. La vida de cada objeto viene dada por una barra vertical. El tiempo transcurre de arriba abajo. [32]

3.3.2.1. Diagrama de Secuencia del caso de uso Editar Textos.

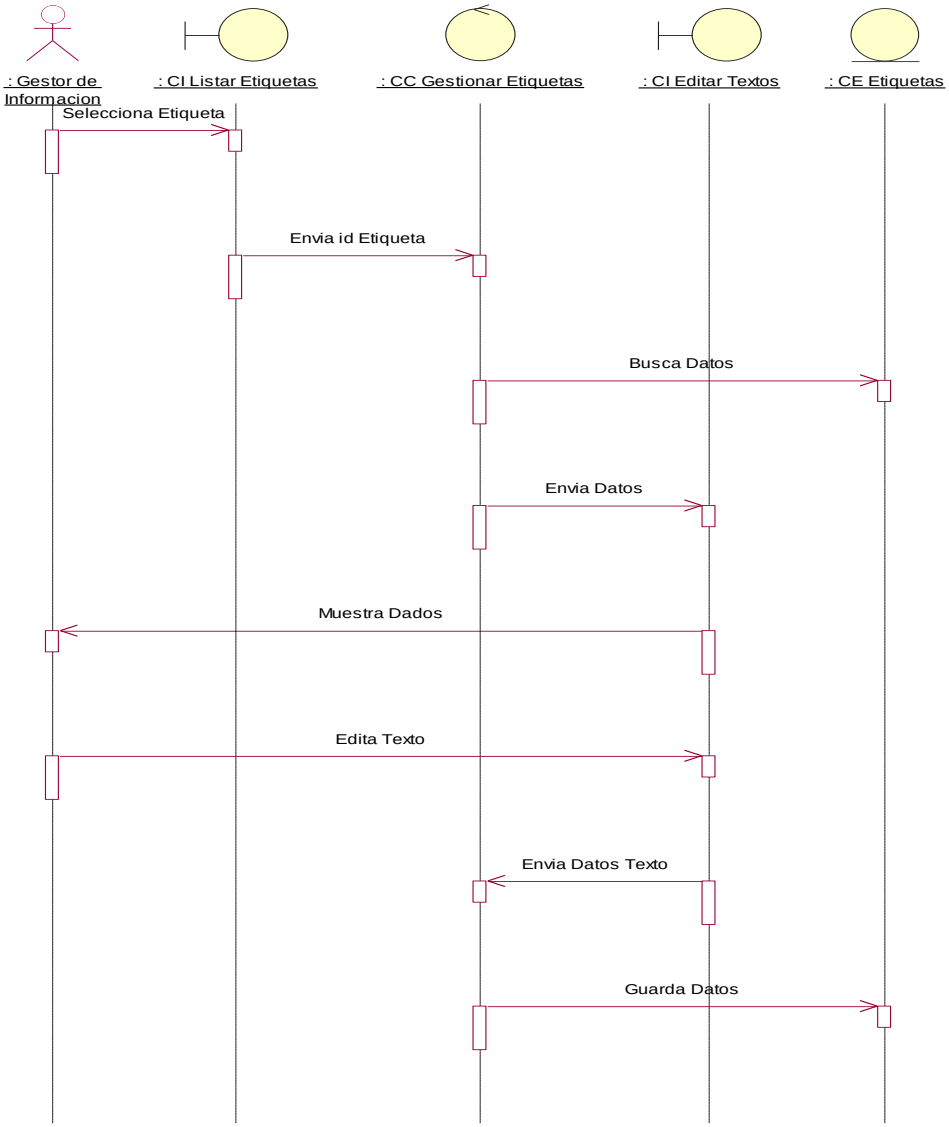


Figura 3.6 Diagrama de secuencia del caso de uso Editar Textos.

3.3.2.2. Diagrama de Secuencia caso de uso Obtener Paquetes de Recursos.

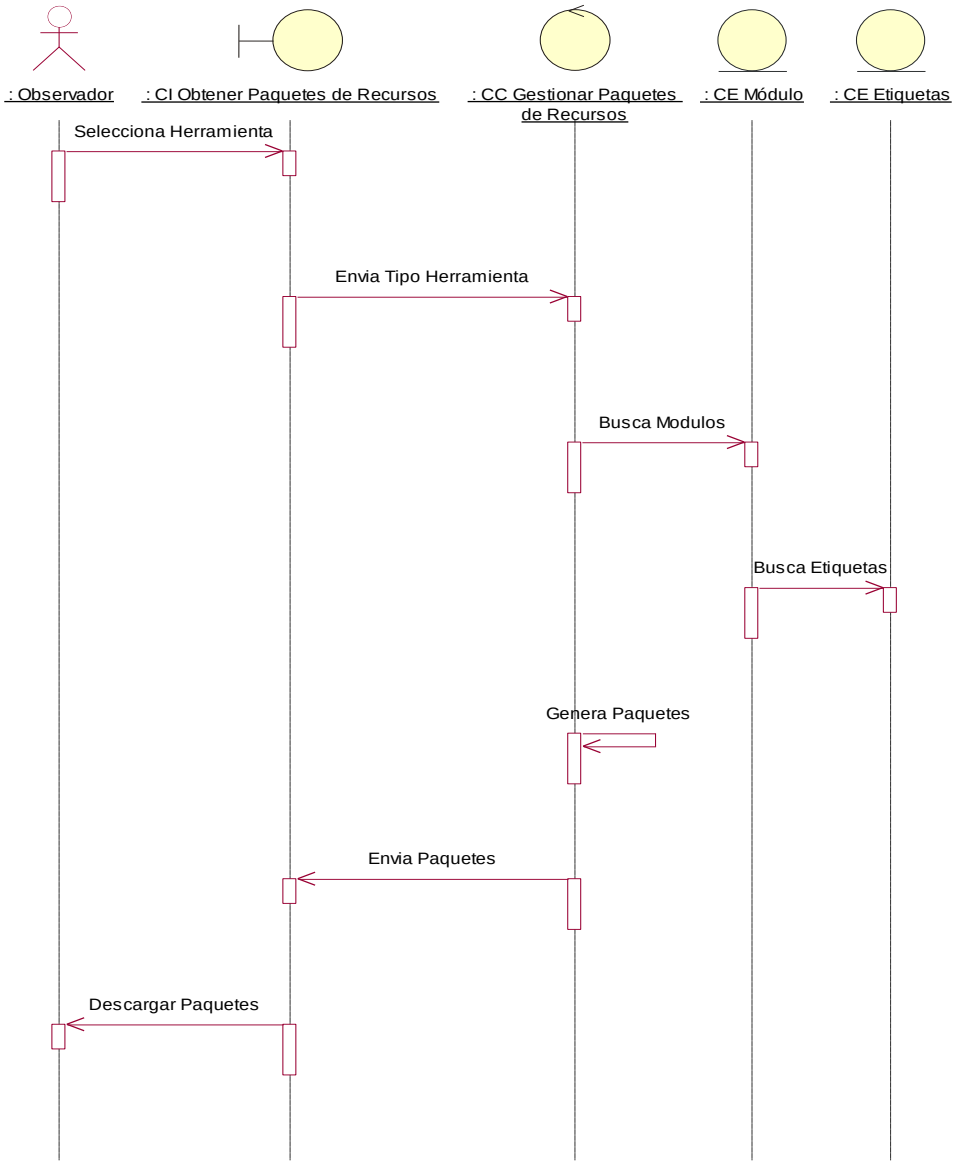


Figura 3.7 Diagrama de secuencia del caso de uso Obtener Paquetes de Recursos.

Los demás diagramas de secuencia se encuentran:

Anexo 4:

Diagrama de secuencia del Caso Uso Iniciar Sesión.

Diagrama de secuencia del Caso Uso Administrar Usuarios.

Diagrama de secuencia del Caso Uso Gestionar Módulos.

Diagrama de secuencia del Caso Uso Gestionar Etiquetas.

3.4. Modelo de Diseño.

El modelo de diseño es un modelo de objetos que adquiere una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución concurrencias y, tecnologías de interfaz de usuario. Además ayuda a descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia. [33]

3.5. Diagrama de clases del Diseño.

Un diagrama de clases de diseño es un diagrama que muestra un conjunto de interfaces, colaboraciones y sus relaciones. Los diagramas de clases de diseño se utilizan para modelar principalmente la vista de diseño estática de un sistema. Esto incluye modelar el vocabulario del sistema, las colaboraciones o esquemas. Los diagramas de clases, son importantes, no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa. [33]

	**Server Page:** Representa la página Web que tiene código que se ejecuta en el servidor. Este código interactúa con recursos en el servidor. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página. Esta clase sólo puede tener relaciones con objetos en el servidor.

	**Client Page:** Una instancia de Página Cliente es una página Web, con formato HTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles para páginas cualquier función dentro de la página. Cada página cliente es construida por una sola página de servidor.
	**Html Form:** Colección de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada del formulario (Text Field, Text Area, Button, Label, Radio Button, Radio Group, Select, Check Box y Hidden Fields).

Figura 3.8 Clases del Diseño.

3.5.1. Justificación de los Patrones de Diseño utilizados.

Durante el diseño del sistema se proponen utilizar patrones generales de software para asignar responsabilidades (GRASP), que constituyen principios básicos a tener en cuenta cuando se quiere construir eficazmente un software orientado a objetos. Entre los más evidentes en el diseño propuesto se encuentran los patrones: Experto, Creador, Bajo Acoplamiento, Alta Cohesión y Controlador.

Para contribuir a una implementación eficiente se hizo necesario el estudio de los patrones (GOF, Gang Of Four), de ellos se seleccionaron dos: Singleton y Observer, los cuales se explicarán a continuación.

3.5.1.1. Singleton (patrón solitario).

Con este patrón se garantiza una única instancia de aquellas clases que se desee tener una sola en toda la aplicación, proporcionando un punto de acceso global a dichas clases. Tiene como ventajas que reduce el espacio de nombres y es una mejora sobre las variables globales. Es usado debido a la necesidad de trabajar con el mismo objeto en distintos momentos y distintos subsistemas. Específicamente en el SGE, el desarrollador crea un módulo para agrupar las etiquetas y ese mismo módulo va a ser llamado desde las controladoras SP Paquetes de Recursos para realizar la funcionalidad generar paquetes de recursos para la herramienta deseada.

3.5.1.2. Observer (observador).

Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos. Específicamente en el SGE existe una dependencia de de uno-a-muchos entre módulos y etiquetas, cuando un módulo es eliminado se eliminan automáticamente las etiquetas pertenecientes a este.

A continuación se mostrará el diseño propuesto para el Sistema de Gestión del Etiquetado.

3.5.2. Iniciar Sesión.

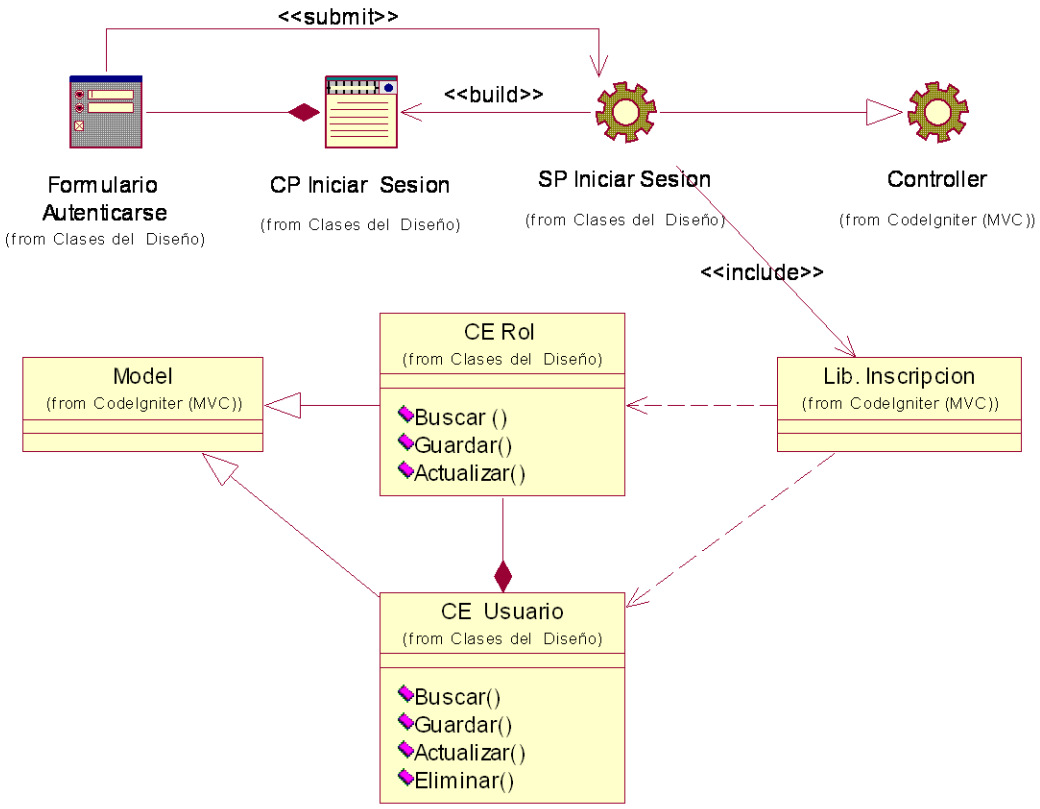


Figura 11. Diagrama de clases del diseño del Caso de Uso Iniciar Sesión.

3.5.3. Obtener Paquetes de Recursos.

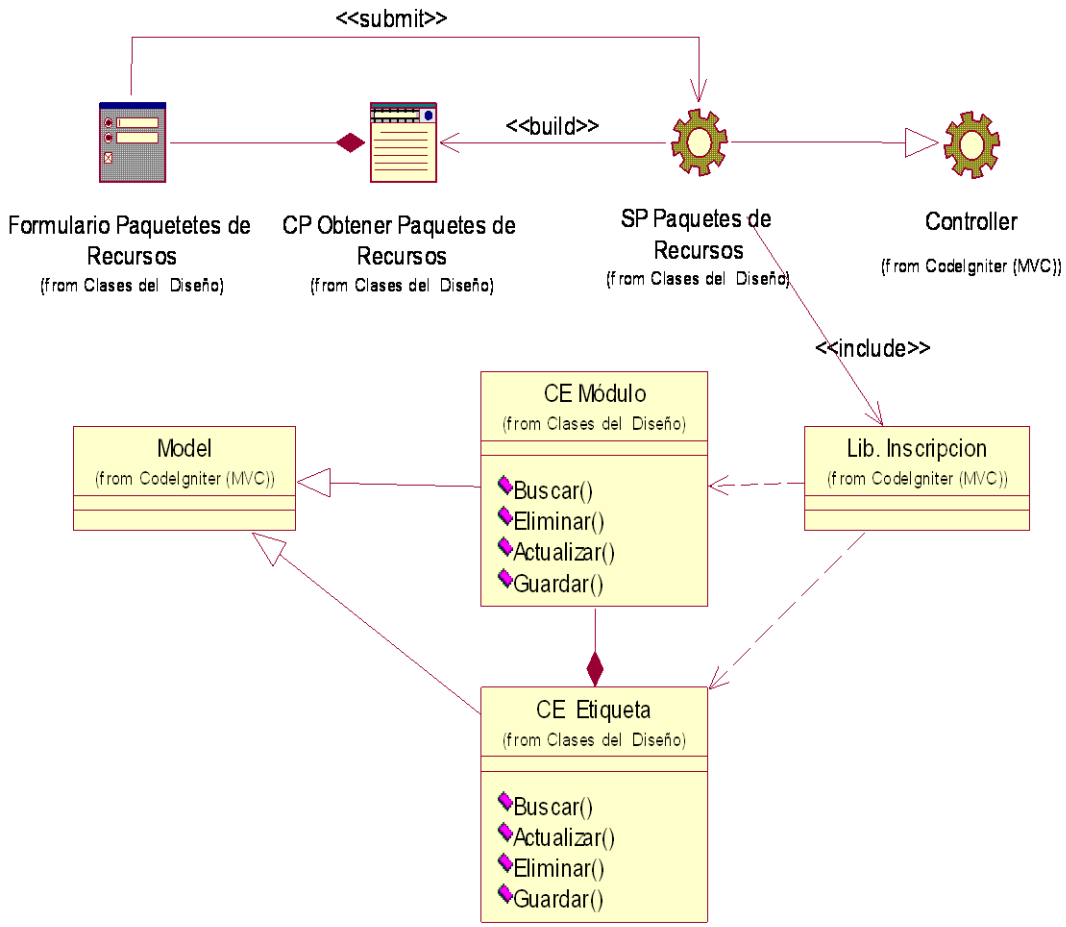


Figura 3.9 Diagrama de Clases del Diseño del CU Obtener Paquetes de Recursos.

3.5.4. Gestionar Etiquetas.

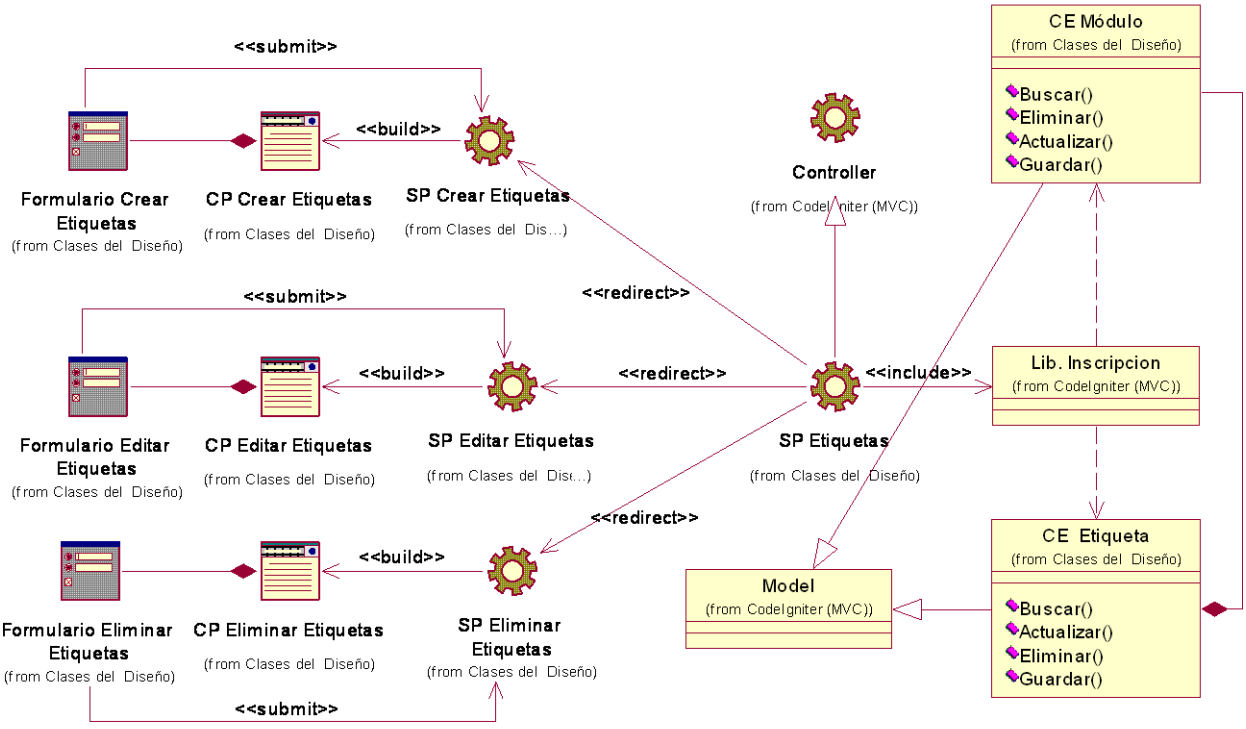


Figura 3.10 Diagrama de Clases del Diseño para el Caso de Uso Gestionar Textos.

3.5.5. Gestionar Usuarios.

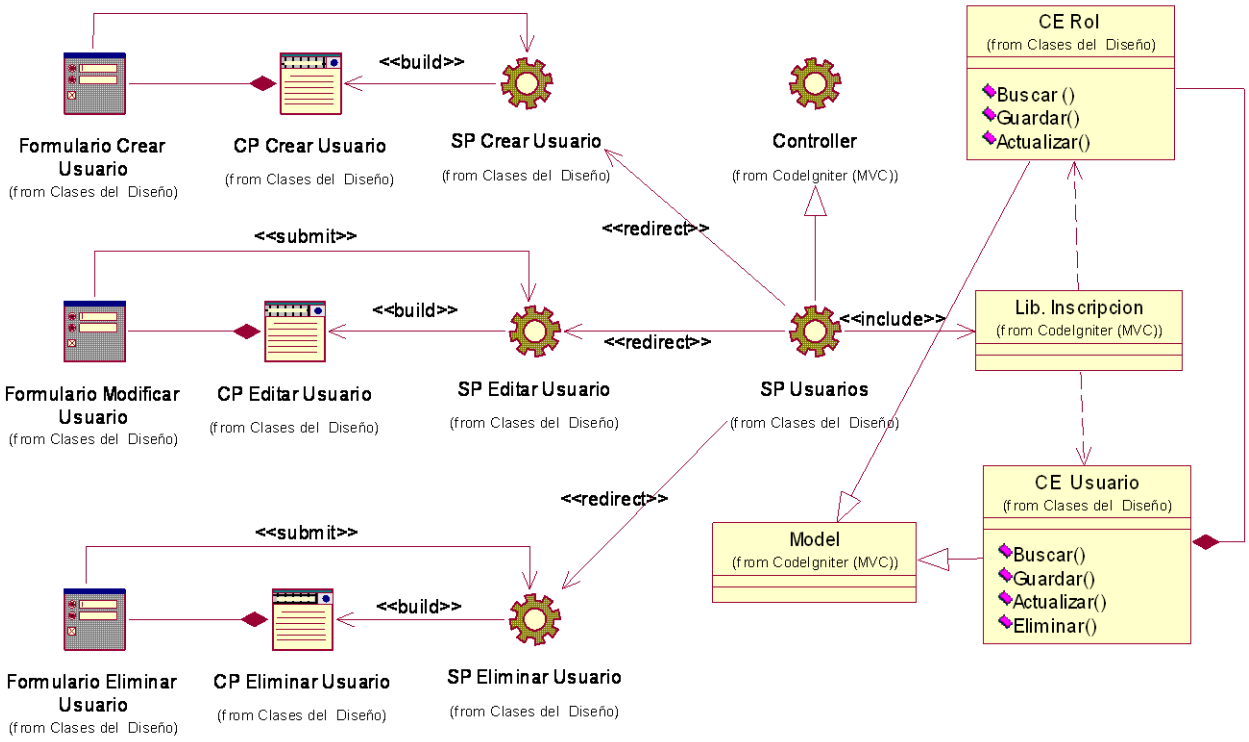


Figura 3.11 Diagrama de Clases del Diseño para el Caso de Uso Administrar Usuarios.

3.5.6. Gestionar Módulos.

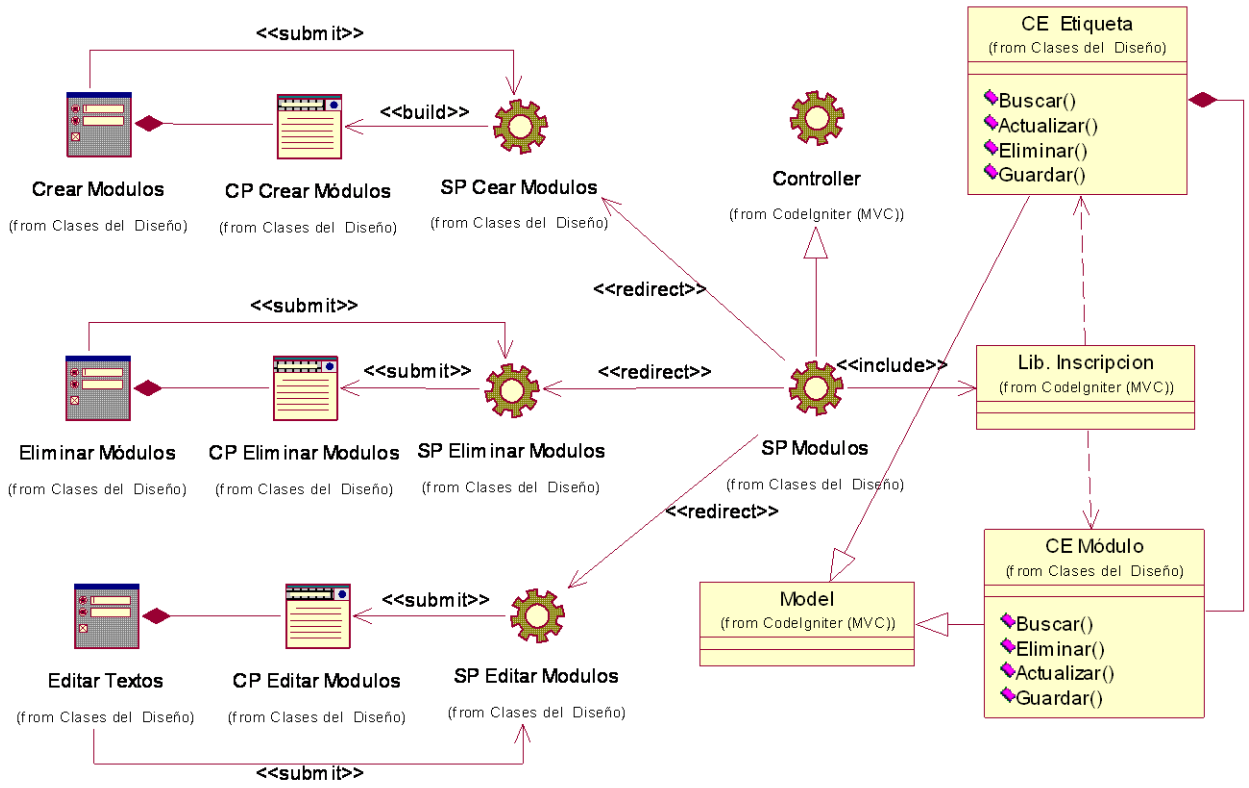


Figura 3.13 Diagrama de Clases del Diseño para el Caso de Uso Gestionar Módulos.

En el diseño se modelaron algunas clases y librerías necesarias para el funcionamiento del sistema, a continuación se explicará el objetivo de cada una de ellas.

Controller: Clase controladora perteneciente al Codeigniter de la cual heredan todas las clases controladoras del sistema, las funcionalidades que tiene implementado el framework para este tipo de clases.

Lib. Inscripción: Librería del Codeigniter que se ubica entre las clases controladoras y los modelos del sistema, su objetivo principal es la validación de datos.

Model: Clase entidad perteneciente al Codeigniter de la cual heredan todas las clases entidades del sistema, las funcionalidades que tiene implementado el framework para este tipo de clases.

3.6. Diagrama de Clases Persistentes.

Una clase persistente es una clase entidad que tiene la capacidad de mantener su valor en el espacio y en el tiempo, y el diagrama de clases persistentes no es más que dichas clases y las relaciones que se establece entre ellas (asociación, agregación/composición).

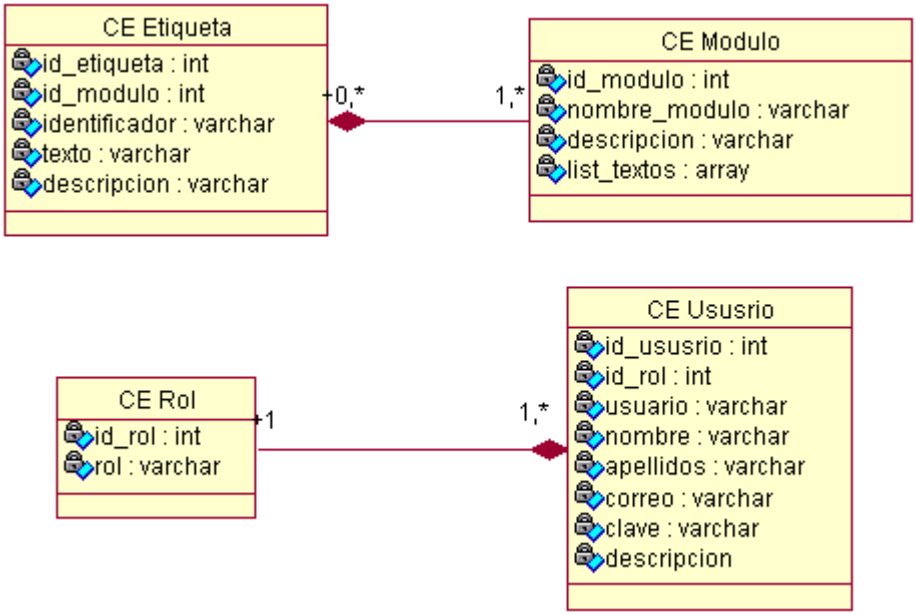


Figura 3.14 Diagrama de Clases Persistentes.

3.7. Métricas orientadas a clases.

Los objetivos principales de estas métricas son**:** comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado a nivel del proyecto. Estas métricas hacen hincapié en el encapsulamiento, la herencia y la complejidad de las clases.

3.7.1. Tamaño de Clase (TC).

Cuando existe un TC grande se afectan los parámetros de calidad definidos por esta métrica. Se reduce la reutilización de las clases, la implementación se hace más compleja, las pruebas son difíciles de realizar y aumenta la responsabilidad de las clases.

Para medir el tamaño de clase se tienen en cuenta los siguientes aspectos:

- Total de operaciones, ya sean las propias o las heredadas de las clases padres e interfaces que implementen.
- Cantidad de atributos, tanto los de ella, como los de las clases padres.
- Promedio general de los dos anteriores para el sistema completo.

Para evaluar las métricas son necesarios los umbrales. En este caso las clases se clasifican en tres grupos según su tamaño, los que se representan en la siguiente tabla junto con los umbrales seleccionados para su clasificación. [30]

El valor de los Umbrales se calcula mediante la suma de la cantidad de atributos y la cantidad de operaciones de una clase.

Tabla 9 Valores de los umbrales para TC

Clasificación	**Valores de los umbrales**
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Tabla 3.2 Valores de los umbrales para TC

La tabla 3.3 ilustra las clases del sistema aplicándole la métrica seleccionada.

No	**Nombre**	**Cantidad Atributos**	**Cantidad Operaciones**	**Umbral**	**Tamaño**
1	CC Gestionar Usuarios	1	4	5	Pequeño
2	CE Usuario	8	4	12	Pequeño
3	CE Rol	2	3	5	Pequeño
4	CC Gestionar Etiquetas	1	4	5	Pequeño
5	CE Etiqueta	5	4	9	Pequeño
6	CE Módulo	4	4	8	Pequeño
7	CC Gestionar Paq. Recursos	6	24	30	Medio
8	CC Gestionar Módulos	1	4	5	Pequeño

Tabla 3.3 Tamaño de las clases.

La mayoría de las clases que conforman el sistema están dentro de la categoría de pequeñas, lo que demuestra que el sistema no es complejo. Los resultados obtenidos son positivos según esta métrica, como se puede ver en las siguientes tablas.

Clasificación	Cantidad Clases
Pequeño	7
Medio	1
Grande	0

Tabla 3.4 Cantidad de clases por clasificación.

Cantidad Clases	Promedio Atributos	Promedio Operaciones
8	3.5	6.375

Tabla 3.5 Resultados de la Métrica TC.

3.7.2. Relaciones entre Clases (RC).

Esta métrica está dada por la cantidad de relaciones de uso que existe entre las distintas clases que forman el diseño propuesto. Se le aplica a las mismas clases que le fue aplicada la métrica TC. Los aspectos de calidad que se miden son: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas. [30]

No	Nombre	Relaciones de uso
1	CC Gestionar Usuarios	2
2	CE Usuario	1
3	CE Rol	1
4	CC Gestionar Etiquetas	2
5	CE Etiqueta	1
6	CE Modulo	1
7	CC Gestionar Paq. Recursos	2
8	CC Gestionar Módulos	2

Tabla 3.6 Cantidad de relaciones de uso entre las clases.

Para medir el acoplamiento según los resultados de esta métrica, algunos especialistas plantean los siguientes valores.

Categoría	Relaciones de uso	Cantidad de Clases
Ninguno	0	0
Bajo	1	4
Medio	2	4
Alto	>2	0

Tabla 3.7 Acoplamiento.

Los demás parámetros de calidad que mide esta métrica dependen del valor promedio de las dependencias de uso de todas las clases, en este caso ese promedio es de 1.5.

Categoría	Criterio	Cantidad de Clases
Baja	<=Prom.	4
Media	>Prom. Y <=2*Prom.	4
Alta	>2*Prom.	0

Tabla 3.8 Cantidad de Pruebas y Complejidad de Mantenimiento

Categoría	Criterio	Cantidad de Clases
Baja	>2*Prom.	4
Media	>Prom. Y <=2*Prom.	4
Alta	<=Prom.	0

Tabla 3.9 Reutilización.

De manera general los resultados de esta métrica son positivos. El acoplamiento existente entre las clases es muy bajo, a pesar de que no existen clases con ningún acoplamiento, el 50% es bajo y el otro 50% es medio, y no existen clases con alto acoplamiento. El nivel de reutilización de las clases es bastante bueno, el 50% de las clases pueden ser reutilizadas. Pasa lo mismo con la cantidad de pruebas y la complejidad de mantenimiento, el 50% de las clases son fáciles de reparar y la cantidad de pruebas a realizar es relativamente corta.

3.7.3. Árbol de profundidad de herencia (APH).

Esta métrica está definida por la máxima longitud que exista entre el nodo y la raíz del árbol. Donde el nodo es una clase hija que hereda de una clase, y así sucesivamente hasta llegar a la raíz. A medida que esa longitud aumenta, entonces se van heredando más operaciones y atributos por las clases hijas. Se hace difícil predecir el comportamiento de las clases que se encuentran en los niveles más bajos del árbol. Esta tiene sus ventajas y desventajas. Si los valores de APH son grandes, entonces se garantiza que se reutilice gran cantidad de código; pero al mismo tiempo hace que el diseño sea más complejo. Esto provoca un mayor acoplamiento entre las clases. [30]

A continuación se muestra los valores de los umbrales con respecto al acoplamiento según plantea esta métrica.

Valores de los Umbrales.	**Acoplamiento.**
<= 5	bajo
> 5	alto

Tabla 3.10 Valores de los umbrales para la medición del acoplamiento

En el Sistema de Gestión del Etiquetado no se hizo necesario hacer demasiado uso de la herencia. Esta fue empleada solamente para las clases controladoras que heredan de la clase controller del codeigniter, y las clases entidades que heredan de la clase model, también del framework. Aplicando esta métrica al diseño propuesto se obtienen resultados que demuestran su poca complejidad, el árbol de profundidad de herencia toma valor 2, por lo que existe bajo acoplamiento y es de fácil reparación. En la siguiente figura se muestra un diagrama donde se representa una de las herencias en el SGE.

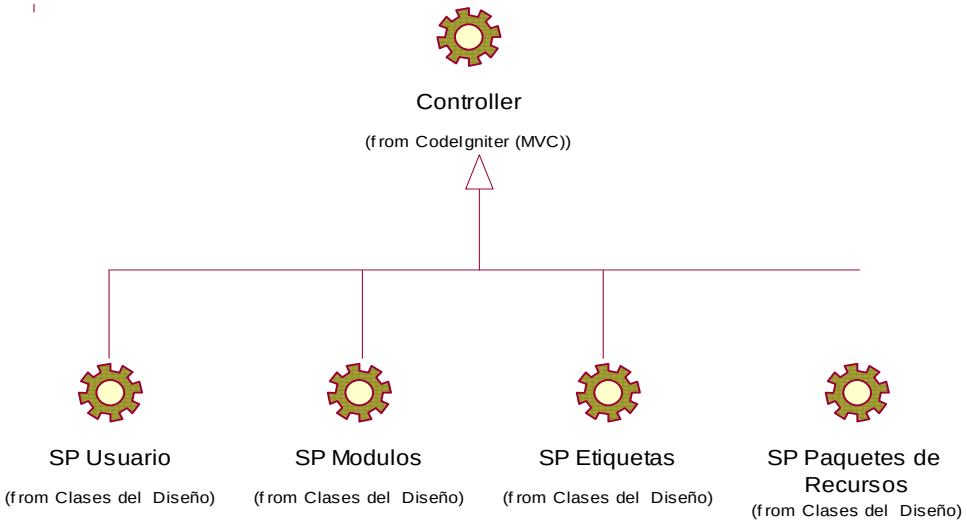


Figura 3.15 Herencia entre las clases controladoras.

3.8. Conclusiones.

Luego de tratar aspectos referentes al análisis y diseño del sistema, y obtener los principales artefactos de este flujo se puede concluir que: la realización del modelo de análisis de forma detallada facilitó la entrada al diseño. El uso de patrones de diseño y arquitectura permitió obtener un diseño flexible y de mayor calidad, la modelación del diseño contribuirá a la organización del trabajo de los desarrolladores y a entender mejor el funcionamiento del futuro sistema, Las realizaciones de los casos de uso aportarán la información necesaria para el proceso de implementación. Las métricas aplicadas a los artefactos obtenidos validan de forma general la calidad del modelo de análisis y diseño del SGE.

Capitulo IV: Implementación y Pruebas.

4.1. Introducción.

En el presente capítulo se modelan los artefactos correspondientes al flujo de trabajo implementación. Se especifican los estándares utilizados para la realización de la aplicación y se realizan pruebas de caja negra para garantizar la calidad del producto.

4.2. Modelo de Implementación.

Los diagramas de despliegue y componente conforman lo que se conoce como modelo de implementación. Estos describen los componentes a construir, su organización y la dependencia entre nodos físicos en los que funcionará el sistema. [34]

4.2.1. Diagrama de Componentes.

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación.
A continuación se muestra el diagrama de componentes correspondiente para la implementación del sistema. [34]

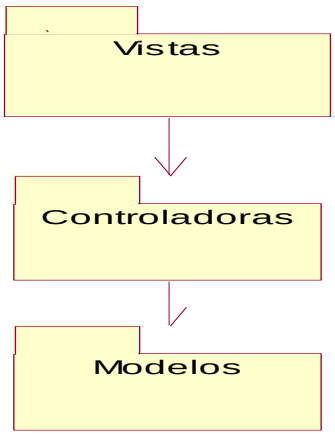


Figura 4.1 Diagrama de Componentes Sistema de Gestión del Etiquetado (paquetes).

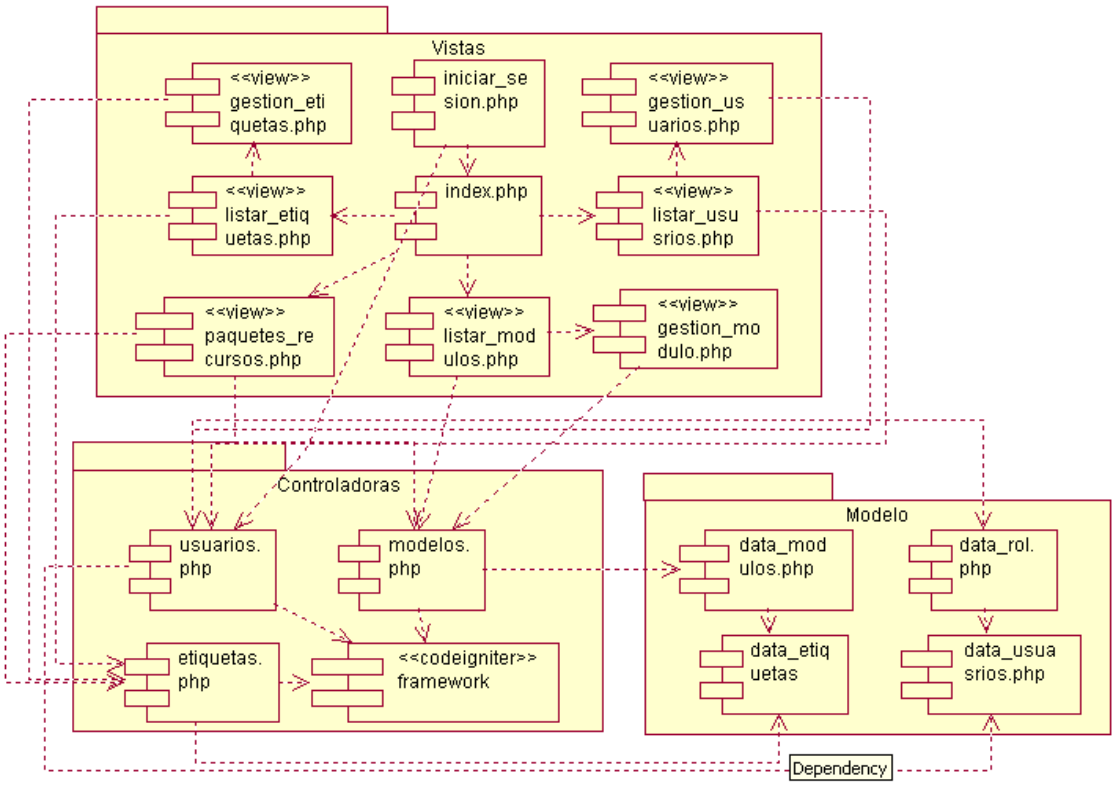


Figura 4.2 Diagrama de Componentes Sistema de Gestión del Etiquetados (componentes).

4.2.2. Diagrama de Despliegue.

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los componentes software que representan manifestaciones del código en tiempo de ejecución (los componentes que sólo sean utilizados en tiempo de compilación deben mostrarse en el diagrama de componentes). [34]

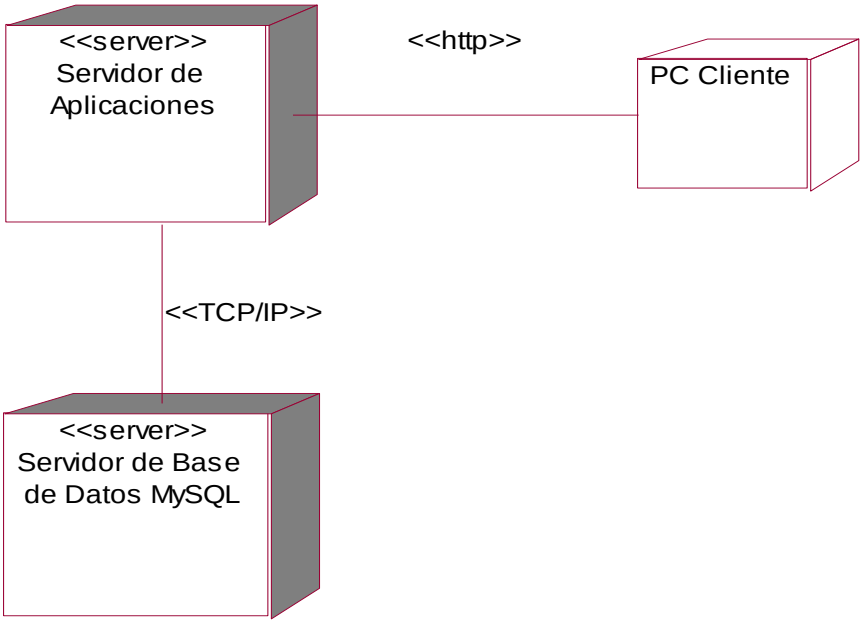


Figura 4.3 Diagrama de despliegue Sistema de Gestión del Etiquetado.

4.3. Estándares utilizados.

En el campo de la informática se define un estándar como: "una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad. También significa un modelo o guía que se sigue para realizar un proceso o para no desviarnos de un objetivo". También puede decirse que estándar en informática, es un conjunto de especificaciones técnicas utilizadas para unificar el desarrollo de hardware o de software.

4.3.1. Estándares de Diseño.

En el diseño de una interfaz para un sitio Web es necesario tomar en cuenta la interacción de un usuario con un objetivo determinado y un espacio virtual, el sitio Web, donde se encuentra el objeto buscado por el usuario. Deben evitarse problemas comunes como el exceso de entretenimiento e interactividad innecesarias, éstos influyen negativamente en el acceso del usuario a la información ofrecida y/o buscada dentro del sitio. Algunos efectos multimedia pueden ser muy atractivos, pero dentro de un sitio público traen consigo problemas de carga lenta del sitio Web, poca compatibilidad en diferentes navegadores y versiones de los mismos.

Para el diseño del Sistema de Gestión de Etiquetado se tiene en cuenta que la aplicación servirá como herramienta para equipos de desarrollo de software por lo que no es necesario utilizar efectos de

multimedia o archivos de video que puedan ralentizar el rendimiento de la aplicación. Esto no significa que la misma no cumpla una adecuada interactividad con el usuario proporcionándole una navegación fácil y una interfaz atractiva.

Figura 4.4 Página de Inicio del Sistema de Gestión del Etiquetado.

Entre los principales estándares a cumplir se encuentran:

- El género de todos los actores que aparezcan en el prototipo será masculino por ejemplo: usuario, desarrollador, gestor de información, administrador.
- En el título de los formularios se pondrá el nombre de la acción en infinitivo, la capitalización es igual a la de las etiquetas por ejemplo: crear etiqueta, crear módulo, crear usuario, cambiar clave etc.
- En las ventanas de Información y Error se mostrará en rojo y amarillo el contenido del mensaje (parte superior) y en el campo donde ocurrió el error se rellenará en rojo con los botones de “aceptar” y “cancelar” (parte inferior).

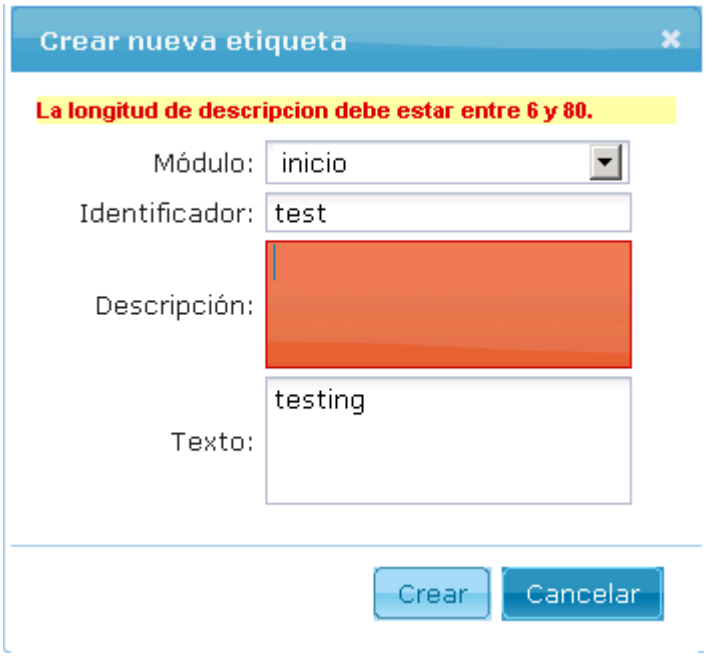


Figura 4.5 Mensajes de Error.

- Las ventanas de advertencia estarán en el mismo lugar de las de error (parte superior), la mismas brindaran la información precisa de los campos a llenar.

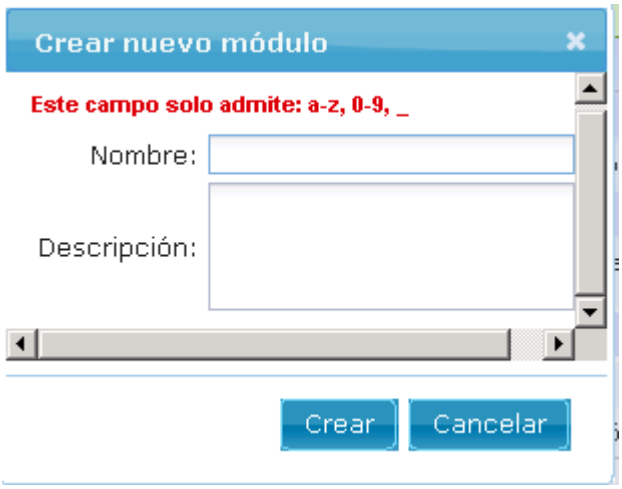


Figura 4.6 Mensajes de Advertencia.

- Los mensajes notificación de las operaciones satisfactorias, aparecerán en verde e informaran al usuario sobre la acción realizada.

Figura 4.7 Mensajes de Notificación.

4.3.2. Estándares de Codificación.

Actualmente se encuentran estándares de codificación para la mayoría de los lenguajes existentes. El uso de los mismos, partiendo de las convenciones definidas, permite una mejor comunicación entre los programadores, creando las condiciones para la reusabilidad y mantenimiento de los sistemas. Para definir el estilo de codificación a seguir en la aplicación se utilizó la notación estándar establecido por el equipo de desarrollo.

- Todo el código de la aplicación deberá ser escrito en español.
- El nombre de las clases comenzará con la letra mayúscula.
- Los atributos y variables serán escritos en minúscula, y en caso que sea necesario para su identificación se separara utilizando el underscore, por ejemplo: nombre _ modulo, cantidad _ etiquetas, etc.
- Los comentarios deben ser breves y escritos al final de la línea y de la siguiente forma: //esto es un comentario……
- Los nombres de las funciones se escribirán de la misma forma que los atributos y tendrán la siguiente estructura para las llaves que marcan el inicio y fin del cuerpo de la función.

```
private function generar_contenido($modulo)
{
    $fin_linea = "\n";

    $contenido = "# ".$modulo['nombre_modulo'].$fin_linea.$fin_linea.
                 "# ".$fin_linea.
                 "# FILE GENERATE BY : ".nombre_completo()")".$fin_linea.
                 "# DATE : ".date("d/m/Y h:i:s a").$fin_linea.
                 "# ".$fin_linea.
                 "#   MODULE NAME :".$fin_linea.
                 "#               ".$modulo['nombre_modulo'].$fin_linea.
                 "#   DESCRIPTION :".$fin_linea.
                 "#              ".$modulo['descripcion'].$fin_linea.
                 "# ".$fin_linea.
                 '['.$modulo['nombre_modulo'].']'.$fin_linea;

    $cantidad_etiquetas = count($modulo['etiquetas']);

    foreach ($modulo['etiquetas'] as $pos => $etiqueta)
    {
        $contenido .= $etiqueta['identificador'].' = '.$etiqueta['texto'].$fin_linea;
    }

    return $contenido;
}
```

Figura 4.8 Ejemplo del código del Sistema de Gestión del Etiquetado.

4.4. Modelo de Pruebas.

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces. Las pruebas es el proceso de ejercitar un programa con la intención específica de encontrar errores previos a la entrega al usuario final y van encaminadas a garantizar la calidad del software en todo momento del desarrollo. Durante este flujo de trabajo se verifica el resultado de la implementación, planificando, diseñando e implementando las pruebas necesarias, llegando a crear así los casos de prueba. Luego se concluye con su realización y el manejo de los resultados, siendo el Modelo de Prueba el principal.

4.4.1. Casos de Pruebas.

Un caso de prueba especifica una forma de probar el sistema incluyendo las entradas con las que se ha de probar, los resultados esperados y las condiciones bajo las que ha de probarse. Un caso de

prueba se deriva de un caso de uso en el modelo de casos de uso o de una realización de caso de uso en el modelo de diseño, lo cual estos casos de prueba permiten validar los requerimientos funcionales del sistema.

4.4.1.1. Pruebas de Caja Negra.

Las Pruebas de Caja Negra son llevadas a cabo sobre la interfaz del software, pretendiendo demostrar que las funciones del software son operativas, que la entrada se acepta de forma apropiada, que se produce una salida de forma eficiente y que la integridad de la información externa se mantiene. Se derivan conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales del programa.

Cuando un usuario trata de autenticarse pueden ocurrir 2 posibles situaciones, las mismas se describen a continuación en la tabla 4.1.

Entrada.	Respuesta del Sistema.	Condiciones.	Resultado de la Prueba.
El usuario introduce su nombre de usuario incorrecto o la contraseña incorrecta.	El sistema muestra el siguiente mensaje de error “Usuario o clave incorrecta”.	La contraseña y el usuario deben ser escritos correctamente.	Satisfactorio.
El usuario deja algún campo vacío al autenticarse.	El sistema muestra el siguiente mensaje de error “Usuario o clave incorrecta”.	El usuario no llena todos los campos necesarios para acceder al sistema e intenta entrar al sistema.	Satisfactorio.

Tabla 4.1 Caso de Prueba del CU Iniciar Sesión.

Cuando el desarrollador desea crear una nueva etiqueta pueden ocurrir 5 posibles situaciones, las mismas se describen a continuación en la tabla 4.2.

Entrada.	Respuesta del Sistema.	Condiciones.	Resultado de la

			Prueba.
El desarrollador desea crear una nueva etiqueta e introduce un identificador que ya existe en el módulo seleccionado.	El sistema muestra el siguiente mensaje de error “Ya existe ese identificador de etiqueta en el módulo seleccionado”.	El desarrollador introduce el identificador de etiqueta que ya existe en ese mismo módulo.	Satisfactorio.
El desarrollador desea crear una nueva etiqueta e introduce un identificador que excede la longitud de identificador.	El sistema muestra el siguiente mensaje de error “La longitud de identificador debe estar entre 3 y 15 caracteres”.	El desarrollador introduce un identificador de etiqueta que no se encuentra entre 3 y 15 caracteres.	Satisfactorio.
El desarrollador desea crear una nueva etiqueta e introduce una descripción que excede la longitud de descripción.	El sistema muestra el siguiente mensaje de error “La longitud de la descripción debe estar entre 6 y 80 caracteres”.	El desarrollador introduce una descripción de etiqueta que no se encuentra entre 6 y 80 caracteres	Satisfactorio.
El desarrollador desea crear una nueva etiqueta y olvida colocarle un identificador.	El sistema muestra el siguiente mensaje de error “La longitud de identificador debe estar entre 3 y 15 caracteres”.	El desarrollador no introduce el identificador a la nueva etiqueta.	Satisfactorio.
El desarrollador desea crear una nueva etiqueta y olvida colocarle una descripción.	El sistema muestra el siguiente mensaje de error “La longitud de la descripción debe estar entre 6 y 80 caracteres”.	El desarrollador no introduce la descripción a la nueva etiqueta.	Satisfactorio.

Tabla 4.2 Caso de Prueba del CU Gestionar Etiquetas.

Cuando el desarrollador desea crear un nuevo módulo pueden ocurrir 5 posibles situaciones, las mismas se describen a continuación en la tabla 4.3.

Entrada.	Respuesta del Sistema.	Condiciones.	Resultado de la Prueba.
El desarrollador desea crear un nuevo módulo y coloca un nombre que ya existe.	El sistema muestra el siguiente mensaje de error “El nombre de módulo especificado ya existe”.	El desarrollador pone al nuevo módulo un nombre que ya existe.	Satisfactorio.
El desarrollador desea crear un nuevo módulo y coloca un nombre que contiene caracteres fuera de rango.	El sistema muestra el siguiente mensaje de error “Este campo solo admite: a-z, 0-9, _”.	El desarrollador utiliza algún caracter fuera del rango a-z, 0-9, _ para llamar al nuevo módulo.	Satisfactorio.
El desarrollador desea crear un nuevo módulo y coloca un nombre fuera de rango.	El sistema muestra un mensaje de error “La longitud de nombre debe estar entre 3 y 15 caracteres”.	El desarrollador pone al nuevo módulo un nombre que esta fuera del rango.	Satisfactorio.
El desarrollador desea crear un nuevo módulo y coloca una descripción que fuera de rango.	El sistema muestra un mensaje de error “La longitud de descripción debe estar entre 3 y 50 caracteres”.	El desarrollador pone al nuevo módulo una descripción fuera de rango.	Satisfactorio.
El desarrollador deja algún campo vacío al crear el nuevo módulo.	El sistema muestra en dependencia del campo incompleto mensajes de error “La longitud del nombre debe estar entre 3 y 15 caracteres”, “La	El desarrollador no llena todos los datos para crear un nuevo módulo.	Satisfactorio.

	longitud de descripción debe estar entre 3 y 50 caracteres".		

Tabla 4.3 Caso de Prueba del CU Gestionar Módulos.

Cuando el administrador desea crear un nuevo usuario pueden ocurrir 3 posibles situaciones, las mismas se describen a continuación en la tabla 4.4.

Entrada.	**Respuesta del Sistema.**	**Condiciones.**	**Resultado de la Prueba.**
El administrador introduce un usuario con todos sus datos ya registrado.	El sistema muestra el siguiente mensaje de error "El nombre de usuario especificado ya existe".	El usuario introducido con todos sus datos ya se encuentra registrado con acceso al sistema.	Satisfactorio.
El administrador introduce una dirección de correo incorrecta para el nuevo usuario.	El sistema muestra el siguiente mensaje de error "El correo debe ser: Ej: user@domain.com".	El correo registrado en el sistema es una dirección no válida.	Satisfactorio.
El administrador deja algún campo vacío o inserta los datos fuera de rango.	El sistema muestra mensajes de error en dependencia del campo vacío o fuera de rango: "La longitud de usuario debe estar entre 3 y 15 caracteres". "La longitud de nombre debe estar entre 3 y 50 caracteres". "La longitud de apellidos	El administrador no llena todos los Campos, o los datos de usuario entrados por el administrador están fuera de rango.	Satisfactorio.

	debe estar entre 6 y 80 caracteres". "La longitud de usuario debe estar entre 3 y 15 caracteres". "La longitud de clave debe estar entre 5 y 20 caracteres".		

Tabla 4.4 Caso de Prueba del CU Gestionar Usuarios.

Nada tiene mayor efecto en la satisfacción del usuario final con respecto al producto deseado que una vista clara de lo que espera, éstas expectativas deben ser verificadas y validadas. Los casos de pruebas reflejan los requerimientos que serán verificados. Se realizó la selección de los requerimientos más apropiados o críticos para realizarle esta prueba. En todos los casos los resultados de la prueba fueron satisfactorios lo que evidencia que el sistema ante algún problema o violación responderá de la manera correcta.

4.5. Conclusiones.

Este capítulo se centró principalmente en realizar el despliegue de la aplicación y los distintos componentes que la conforman, se expusieron algunos de los estándares utilizados en el desarrollo del sistema, concluyéndose con la realización de pruebas de caja negra sobre las interfaces de la aplicación para validar los datos de entrada y salida, y demostrar que las funcionalidades del software son operativas.

Conclusiones.

De forma general se realizaron todos los objetivos trazados en el presente trabajo de diploma, a continuación se expresa cómo se cumplimentaron.

Se demostró la necesidad de los equipos de desarrollo de Software que trabajan con el lenguaje de programación PHP de contar con un sistema que facilite el trabajo con las etiquetas. Para esto se planteó la modelación de los flujos de trabajo de negocio, requerimientos, análisis y diseño, así como implementación y prueba, lo cual posibilitó un desarrollo eficiente del sistema.

Se realizó además un análisis exhaustivo de las diferentes metodologías, técnicas, lenguajes y herramientas actuales para desarrollar el sistema concebido de la mejor forma. Siempre tratando de resolver los problemas de la gestión del etiquetado y realizando un software lo más estable y seguro posible.

La realización del Modelo de Negocio permitió entrar seguidamente al Levantamiento de Requisitos donde se obtuvo todos sus artefactos correspondientes, los cuales permitieron tener una idea mas clara del sistema a desarrollar. Se realizó el modelo de análisis y diseño del sistema de acuerdo a las necesidades y requerimientos que se capturaron, para posteriormente comenzar con el despliegue de la aplicación y los componentes que lo conforman.

Siempre con el objetivo de obtener un sistema con calidad, se fue validando el mismo, para ellos se utilizaron métricas, tanto para los requisitos como para el análisis y diseño del sistema. Se concluyó con la realización de pruebas de caja negra sobre las interfaces del sistema para validar las funcionalidades del mismo.

Recomendaciones.

Como recomendaciones de este trabajo de tesis se tienen:

- **Desarrollar nuevas Funcionalidades:**

Mensajes entre los usuarios: Es una idea muy significativa ya que permitirá una mayor colaboración y comunicación entre los miembros equipos de desarrollos.

Crear grupos de desarrollo: Para lograr una mayor organización en el equipo de desarrollo, seria interesante que el sistema permitiera asignar un grupo de módulos determinados a uno o a un grupo de gestores de información, esto permitirá repartir y monitorear el trabajo.

- **Realizar una ayuda para el sistema.**
- **Obtener paquetes de recursos para otras herramientas.**

Bibliografía.

[9] **Molpeceres, Alberto.** Procesos de desarrollo: RUP, XP y FDD. [En línea] 15 de 12 de 2002. http://www.javahispano.org/contenidos/archivo/71/metodos_desarrollo.pdf.

[8] Programación Extrema (XP) . [En línea] http://homepages.mty.itesm.mx/al1031357/XP.ppt.

[7] Aplicaciones Web. [En línea] http://www.esenciahumana.com.mx/Servicios/AplicacionesWeb/AplicacionesWeb.html
.

[6] **Jacobson, I. Booch, G. Rumbaugh, J.** *El Proceso Unificado de Desarrollo de Software.* Ciudad de La Habana : Félix Varela, 2004.

[5] **Cuaresma, Sergi Blanco.** Metodologías de desarrollo. [En línea] 14 de feberero de 2008. http://www.marblestation.com/?p=644.

[4] Ventajas y Beneficios de una Aplicación Web. [En línea] http://www.esenciahumana.com.mx/Servicios/AplicacionesWeb/VentajasBeneficiosAplicaciones.html.

[3] Smarty Template Engine. [En línea] http://www.smarty.net.

[2] Package Information: Translation. [En línea] http://pear.php.net/package/Translation.

[15] **Zamora.** Arquitectura por Capas. [En línea] http://www.utpl.edu.ec/blog/zamora/2008/02/13/arquitectura-por-capas/.

[13] **Gracia, Joaquin.** Diseño de Software Orientado a Objetos. [En línea] 27 de Mayo de 2005. http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

[11] **Alvarez, Ruben.** Introduccion a la programacion en PHP. [En línea] http://www.desarrolloweb.com/articulos/303.php.

[10] **Larman, Craig.** *UML y PATRONES.Introducción al análisis y diseño orientado a objetos.* Ciudad de La Habana : Felix Valera, 2004.

[1] **Sánchez, Pedro Enrique Castiñeiras.** Sistema de gestión del etiquetado. [En línea] http://www.informaticahabana.com/?q=trabajo&trid=766.

[17] Visual Paradigme para UML. [En línea] [Citado el: 12 de Diciembre de 2008.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_para_Windows_14718_p/].

[19] Dev-PHP IDE. [En línea] 21 de mayo de 2007. http://libredecargo.blogspot.com/2007/05/dev-php-ide.html.

[20] BlueFish. [En línea] Abril, 2009.
http://bluefish.openoffice.nl/

[21]. **Alvarez, Miguel Angel.** Zend Studio. [En línea] 4 de junio de 2003.
http://www.desarrolloweb.com/articulos/1178.php.

[23] MySQL. [En línea] http://mysql.conclase.net/curso/index.php.

[24] MySQL vs. PostgreSQL. [En línea] http://www.fedora-es.com/node/189.

[25] Zend Framework. [En línea] http://devzone.zend.com/article/4524-Zend-Framework-1.8.0-Released.

[22] Sistema Gestores de Bases de Datos. [En línea] 1 de Noviembre de 2004.
http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

[16] Herramienta Case. [En línea]
http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

[26] Iguana PHP Framework. [En línea]
http://www.iguanait.com/es/software/tecnologia/iguana_php_framework.

[27] CodeIgniter. [En línea] http://codeigniter.com/.

[28] Conferencia # 2 Face de Inicio. *Ingenieria de Software I.* [En línea]
http://teleformacion.uci.cu/mod/resource/view.php?id=11569.

[30]. **Pressman, Roger S.** *Ingenieria de Software: Un enfoque Practico.* 5ta. Ciudad de La Habana : Felix Varela, 2005. pág. 958.

[31] Conf_7_FT_Análisis_y_Diseño. [En línea]
http://teleformacion.uci.cu/mod/resource/view.php?id=13151.

[29] Requisitos. [En línea] http://teleformacion.uci.cu/mod/resource/view.php?id=12444.

[32] Flujo de Trabajo Análisis y Diseño.Diagramas de Clases y Diagramas de Interacción. [En línea]
http://teleformacion.uci.cu/mod/resource/view.php?id=13166.

[33] Extensiones_para_WEB. [En línea] http://teleformacion.uci.cu/mod/resource/view.php?id=14069.

[34] Diagramas_de_componentes_y_de_despliegue. [En línea]
http://teleformacion.uci.cu/mod/resource/view.php?id=14072.

[35]. **Ugarte, Jorge.** BPMN estandar para modelamiento de procesos. [En línea]
http://www.slideshare.net/gugarte/bpmn-estandar-para-modelamiento-de-procesos-presentation.

[14]. **Dagum, Diego.** Layers vs. Tiers: la Parábola de Shemp y Curly. [En línea]
http://diegumzone.spaces.live.com/blog/cns!1AD5096D63670065!415.entry.

[36] ¿Qué es Ingeniería de Requisitos (IR)? [En línea] 7 de marzo de 2008.
http://danielvn7.wordpress.com/2008/03/27/%C2%BFque-es-ingenieria-de-requisitos-ir/.

[37] Tormenta de Ideas. [En línea]
http://www.fundibeq.org/metodologias/herramientas/tormenta_de_ideas.pdf.

Glosario de Términos.

A

Adabas D: es un sistema de base de datos relacional.

AJAX: Asynchronous JavaScript and XML. Java script Asíncrono y XML. Es una técnica de Desarrollo Web para crear aplicaciones interactivas, en las que se puede enviar y recibir información de los servidores sin necesidad de recargar las páginas completamente.

API: Interfaz de Programación de Aplicaciones es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

ASP: es un framework para aplicaciones Web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios Web dinámicos, aplicaciones Web y servicios Web XML.

C

C: lenguaje de programación creado en 1972 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B.

C++: Es un lenguaje híbrido, que se puede compilar y resulta más sencillo de aprender para los programadores que ya conocen C. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Las principales características son abstracción (encapsulación), el soporte para programación orientada a objetos (polimorfismo) y el soporte de plantillas o programación genérica (templates). Es un lenguaje que abarca tres paradigmas de la programación: La programación estructurada, la programación genérica y la programación orientada a objetos.

D

Drupal: es un sistema de gestión de contenido para sitios Web. Permite publicar artículos, imágenes, u otros archivos y servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos.

DB2: es un motor de base de datos relacional que integra XML de manera nativa.

DBase: fue el primer Sistema de gestión de base de datos usado ampliamente para microcomputadoras

E

Etiquetas: Lugar donde se encuentra el contenido informativo de un sitio Web, también se utiliza el término textos.

F

Flash: es una aplicación en forma de estudio de animación que trabaja sobre *"Fotogramas"* destinado a la producción y entrega de contenido interactivo para diferentes audiencias alrededor del mundo sin importar la plataforma.

Framework: estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

G

Gettext: Es una biblioteca de GNU de internacionalización. Comúnmente es usada para escribir programas con interfaz en múltiples idiomas.

GNU: Conjunto de programas desarrollados por miembros de la Fundación por el Software Libre, son de uso gratuito (FSF- Free Software Foundation).

GNU/LINUX: Es un sistema operativo, es una implementación de libre distribución UNIX para computadoras personales (PC), servidores, y estaciones de trabajo. Es multitarea, multiusuario, multiplataforma y multiprocesador.

GPL: Es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software Libre (General Public License).

Gestión del Etiquetado: Proceso de colocar las etiquetas en el sitio para que pueda ser visualizado por la interfaz de usuario.

Gestor de Información: Responsable del contenido informativo de una página Web, también es conocido como Arquitecto del Software.

H

HTML: Hypertext Markup Language. Lenguage de tags estandarizado para la creación de documentos para la Web.
HTTP: Hypertext Transfer Protocol. Protocolo de nivel de aplicación usado extensivamente en Internet para el acceso a documentos.

I

IDE: Entorno Integrado de Desarrollo, es un programa compuesto por un conjunto de herramientas para un programador, puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios.
Informix: es el sistema de bases de datos más popular después de Oracle, fue concebido y diseñado por Roger Sippl a finales de los años 1970.
InterBase: es un Sistema de Administración de Base de Datos Relacionales (RDBMS) desarrollada y comercializada por la compañía Borland Software Corporation y actualmente desarrollada por su filial CodeGear.

J

Java: Lenguaje de programación orientado a objetos con el que se puede realizar cualquier tipo de programa, es un lenguaje muy extendido, es un lenguaje independiente de la plataforma, es compilado en un bytecode que es interpretado desarrollado por la compañía Sun Microsystems a principios de los 90.

L

Linux: Sistema Operativo**.**

M

MediaWiki: es un software libre, el paquete wiki está escrito en PHP, al principio para el empleo sobre Wikipedia. Ahora es usado por varios otros proyectos de la Fundación no lucrativa Wikimedia y por muchos otros wikis.
Moodle: Moodle es un Sistema de Dirección de Curso, también conocido como un Sistema de gestión de Etiquetas o un Estudio Virtual del Ambiente. Esto es un uso libre de la Web donde los educadores pueden soler crear sitios de estudio eficaces en línea.
MySQL: Software de gestión de bases de datos relacionales, bajo licencia GNU GPL.

O

ODBC: Es un estándar de acceso a bases de datos que utilizan los sistemas Microsoft. las siglas significan Open DataBase Connectivity.
Oracle: es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle Corporation.

P

Perl: Lenguaje de programación de scripts multiplataforma muy utilizado para realizar actividades de procesamiento de texto.
PHP: Hypertext Preprocessor. Ambiente script del lado del servidor que permite crear y ejecutar aplicaciones Web dinámicas e interactivas, con PHP se pueden combinar páginas HTML y scripts con el objetivo de crear aplicaciones más potentes.
PHPFusion: un sistema de dirección ligero escrito en PHP. Este utiliza una base de datos mySQL para almacenar sus datos, e incluye un sistema simple y comprensivo de administración.
Plug-in: En informática son aplicaciones que no funcionan por si solas sino que interactúa con otra aplicación para aportarle una función o utilidad específica.
Paquetes de Recursos: Se obtiene a partir de los textos, son archivos que serán utilizados por una herramienta determinada para realizar la gestión del etiquetado.

R

RF: Requisito Funcional**.**
RNF: Requisito no Funcional.

S

Servidor de aplicaciones: En Informática se denomina un servidor en una red de computadoras que ejecuta ciertas aplicaciones, como lo son los servidores de Apache, Internet Information Server.
SGBD: Sistema de gestión de Base de Datos, es básicamente el software que permite la utilización y la actualización de los datos almacenados en una o varias bases de datos por los usuarios.
SGE: Sistema de Gestión del Etiquetado.
Smarty: Es un motor de plantillas para PHP, utilizado para separar el código PHP del HTML.
Software: Sistemas o Aplicaciones expresadas en un lenguaje de máquina.
SQL: Lenguaje de Consulta Estructurado, lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.
Sybase: es la compañía de software empresarial más grande enfocada exclusivamente a la gestión y movilización de información, desde el centro de datos, hasta el punto de acción.

T

TCP/IP: Familia de protocolos de Internet, su nombre se debe a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP).

W

Windows: Sistema Operativo**.**

X

XML (Extensible Markup Language): Es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos.

Anexos.

Anexo 1: Encuestas.

Encuestado: Alejandro Martínez Brito
Proyecto Rehabilitación
Área Temática Sistemas Especializados
Fac. 7

1. ¿Cuáles son los problemas que has detectado cuando has trabajado en un proyecto Web? ¿Has presentado problemas con los textos en las interfaces?
El primer problema es el trabajo en equipo. La mayoría trabajan individualmente cuando lo más lógico y productivo es el trabajo en equipo.
Si, en varias ocasiones he tenido problemas con el texto en las interfaces.

2. ¿Cómo resuelves actualmente el trabajo con estos textos?
Trato siempre de que los ficheros tengan la misma codificación (Unicode UTF8) además de trabajar con los códigos ASCII de los caracteres extraños (áéíóúñ…).

3. Cuando se determina que hay que modificar estos mensajes, ¿Qué haces?
Este es un proceso un poco pesado pues hay que buscar en cada una de las interfaces y cambiar los mensajes.

4. Si tuvieras a tu disposición una aplicación que te ayudara con este problema, ¿Cómo influiría eso en tu trabajo?
Me ayudaría con el tiempo de desarrollo, teniendo en cuenta el trabajo que cuesta primeramente buscar los mensajes en cada una de las interfaces y luego modificarlos.

Encuestado: Addiel Matos.
Proyecto: Generador de Reportes Dinámicos.

1. ¿Cuáles son los problemas que has detectado cuando has trabajado en un proyecto Web? ¿Has presentado problemas con los textos en las interfaces?
Problemas con el control ortográfico, es decir con palabras con tildes y con muchos de los caracteres especiales.
Si, a menudo presento problemas con los textos de la interfaces.

2. ¿Cómo resuelves actualmente el trabajo con estos textos?
Se resuelve usando un archivo que define variables globales, además de tratar siempre de que todos los ficheros tengan la misma codificación.

3. Cuando se determina que hay que modificar estos mensajes, ¿Qué haces?
Bueno, cuando tengo que modificar los mensajes comienza un proceso algo tedioso ya que hay que ir por todas las interfaces buscando y posteriormente cambiándolos.

4. Si tuvieras a tu disposición una aplicación que te ayudara con este problema, ¿Cómo influiría eso en tu trabajo?
Sería algo muy bueno no solo para mí, sino para todos los equipos de desarrollo que utilizan PHP puesto que mejoraría en cuanto al rendimiento de las aplicaciones, calidad del producto además de mejorar el tiempo de desarrollo.

Encuestado: Onier Hogguit Rodríguez.
Proyecto: Sistema de Gestión Fiscal.

1- ¿Cuáles son los problemas que has detectado cuando has trabajado en un proyecto Web? ¿Has presentado problemas con los textos en las interfaces?

Los principales problemas que he encontrado al trabajar en un proyecto Web se me han presentado cuando no es un trabajo en equipo, puesto que unos programan de una forma muy distinta a los demás, esto conlleva a que muchas veces no entendemos lo que lo que esta hecho, por otra parte la forma en que se pensó para dar solución al problema, la solución a esto es trabajar en equipo para

trabajar de tal forma que sea lo mas común para todos, dando la posibilidad de que cualquier integrante pueda continuar el trabajo de otro.
Respecto a los problemas con los textos de las interfaces si se me han presentado algunos problemas.

2- ¿Cómo resuelves actualmente el trabajo con estos textos?

Trato de mantener la misma codificación en todo el proyecto, es decir buscar la manera de hacer las cosas lo mas común posible.

3- Cuando se determina que hay que modificar estos mensajes, ¿Qué haces?

Simplemente los modifico uno a uno en cada una de estas interfaces, lo cual me demora en mi trabajo pues hay que buscarlos en cada una de las interfaces.

4-Si tuvieras a tu disposición una aplicación que te ayudara con este problema, ¿Cómo influiría eso en tu trabajo?

Bueno estoy seguro de que me ahorraría mucho trabajo, y se ganaría tiempo de desarrollo para el proyecto.

Encuestado: Walfrido Serrano.
Proyecto: Gestión Documental.

¿Cuáles son los problemas que has detectado cuando has trabajado en un proyecto Web? ¿Has presentado problemas con los textos en las interfaces?

Algunos de los problemas mas comunes a la hora de trabajar en un proyecto Web son: la falta de las interfaces para guiarse a la hora de hacer alguna vista para el sistema (prototipos) y no contar con los textos que deben tener estas interfaces, ya que estos son fundamentales a la hora de guiar al usuario para un mejor entendimiento dichas interfaces.

2. ¿Cómo resuelves actualmente el trabajo con estos textos?

Trato de ser lo mas claro y sugerente posible, con el objetivo de que le usuario tenga idea de que se trata la interfaz. En ocasiones no encuentro los más adecuados.

3. Cuando se determina que hay que modificar estos mensajes, ¿Que haces?

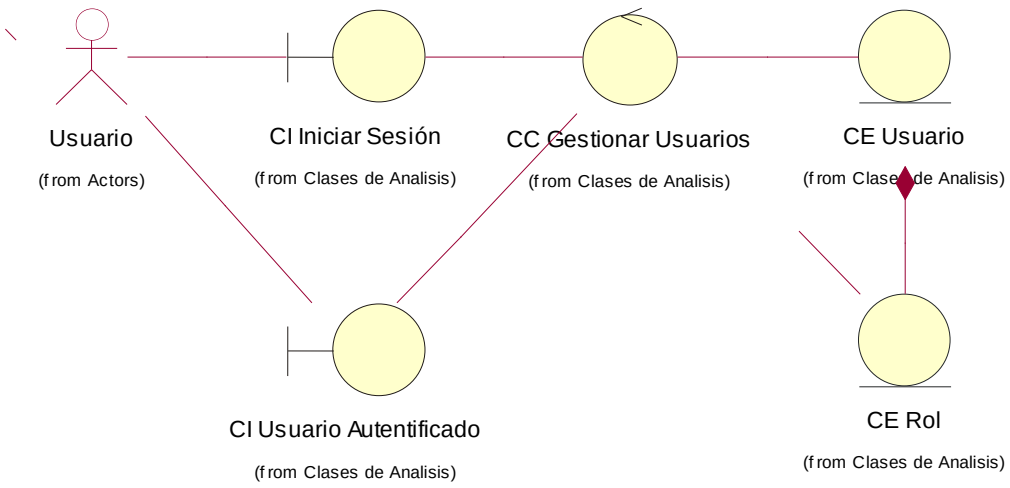
Esto conlleva a un atraso en el proyecto ya que esto debe ser unas de las principales cosas que se deben definir a la hora de comenzar el proyecto, además de que este trabajo de modificar los mensajes que ya existen es tedioso puede traer consigo faltas de ortografía etc.

4. Si tuvieras a tu disposición una aplicación que te ayudara con este problema, ¿Cómo influiría eso en tu trabajo?

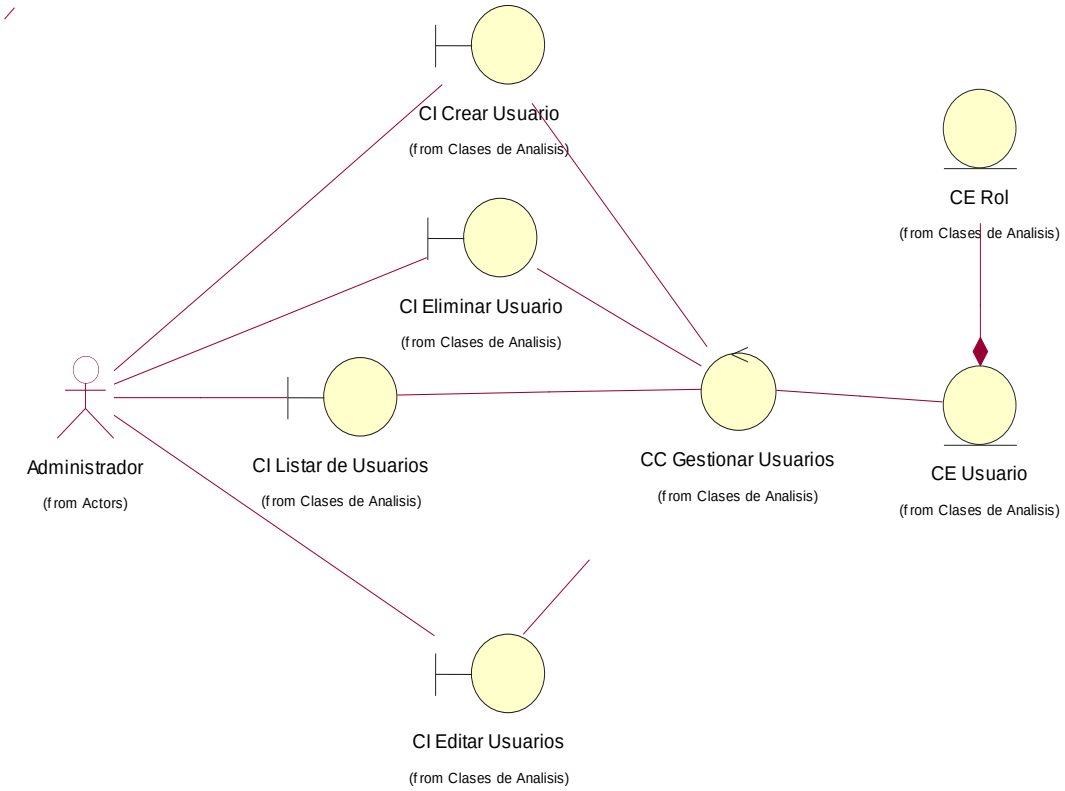
Pienso que seria de gran ayuda y que de alguna forma me ahorraría trabajo y tiempo de desarrollo.

Anexo 2: Diagrama de Clase de Análisis.

Caso de Uso Iniciar Sesión.

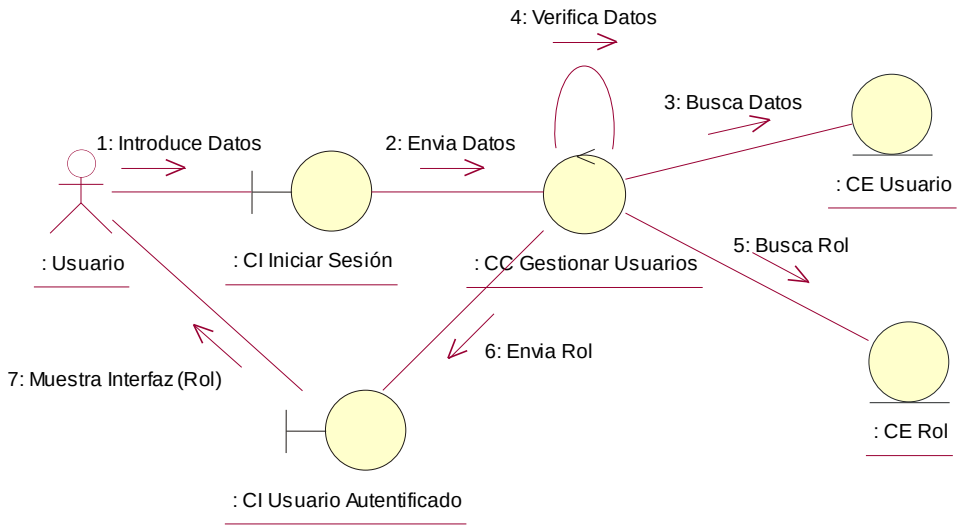


Caso de Uso Gestionar Usuarios.

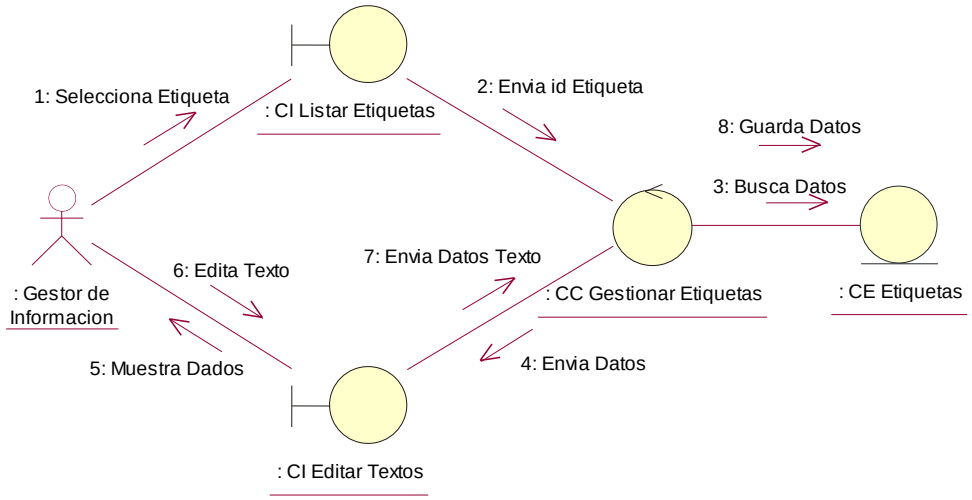


Anexo 3: **Diagramas de Colaboración:**

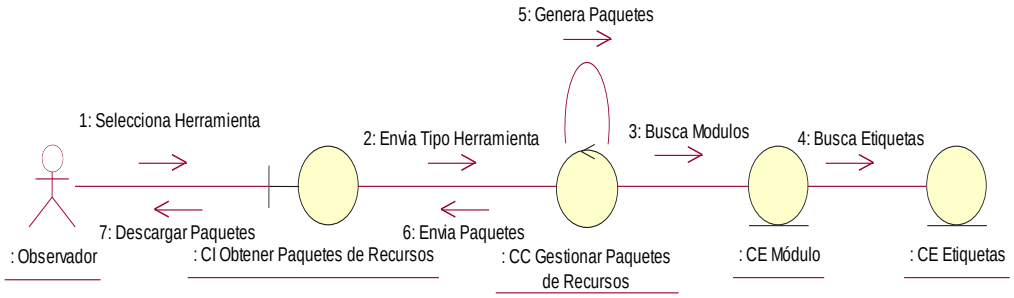
Iniciar Sesión.



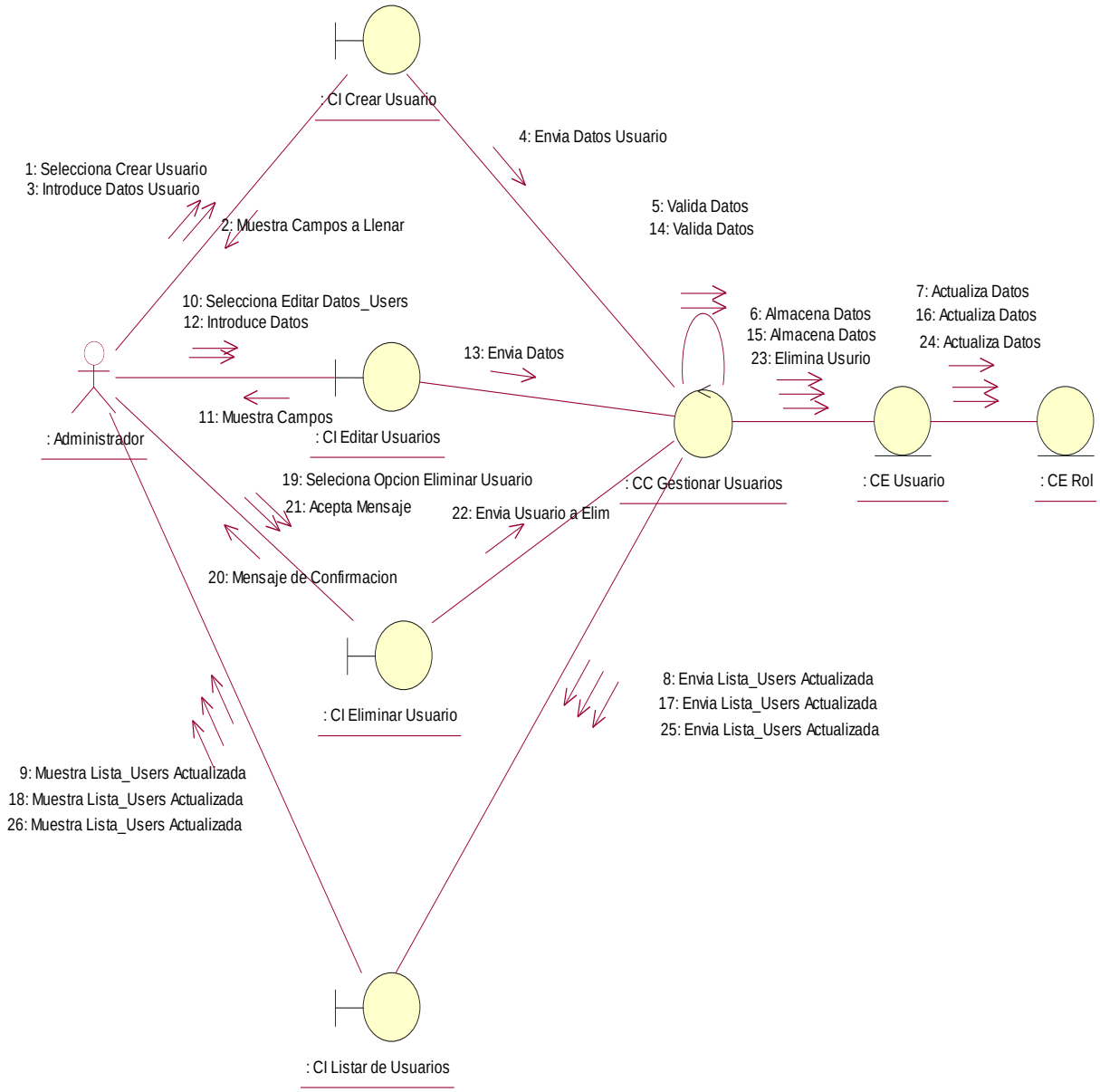
Editar Textos.



Obtener Paquetes de Recursos.

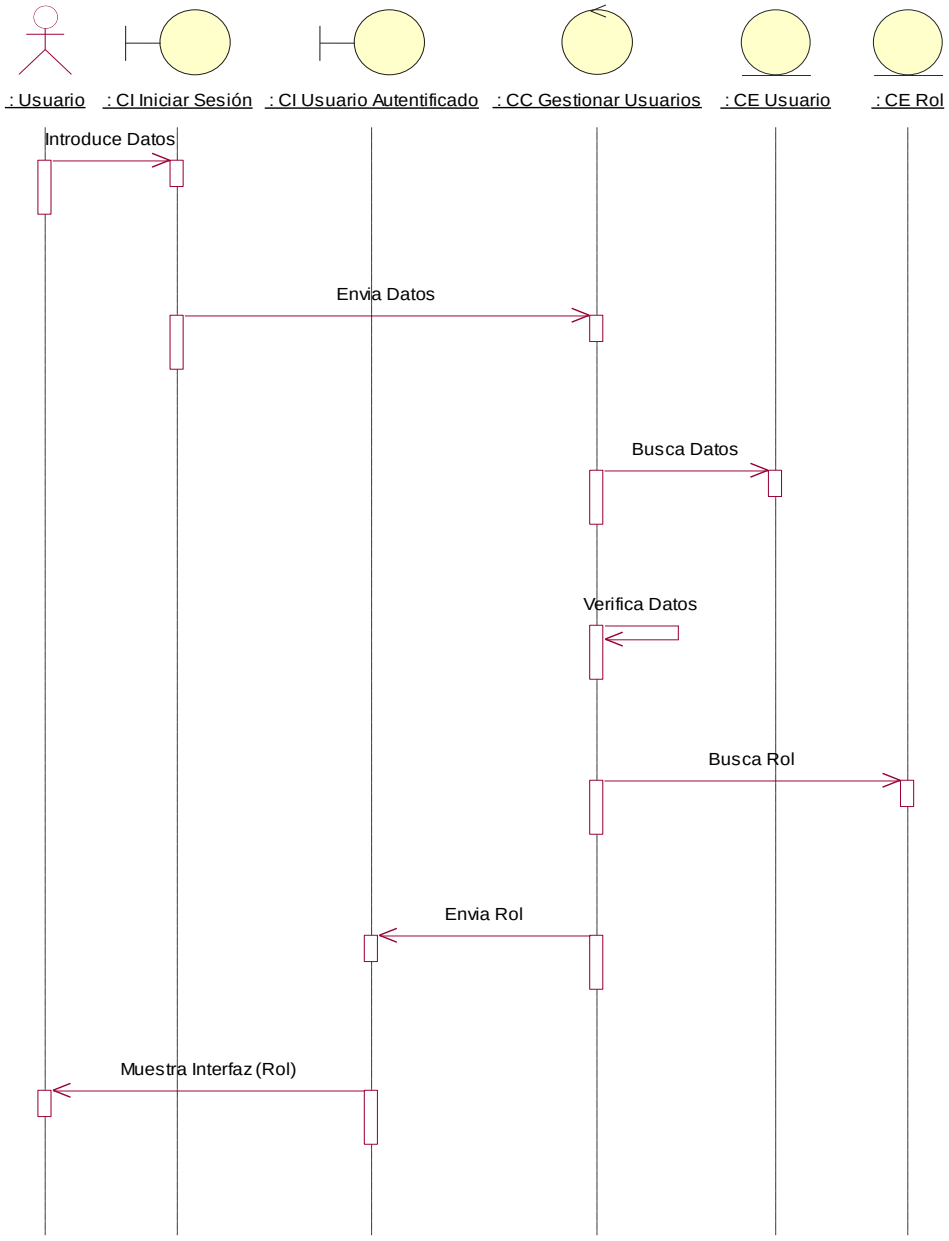


Gestionar Usuarios.



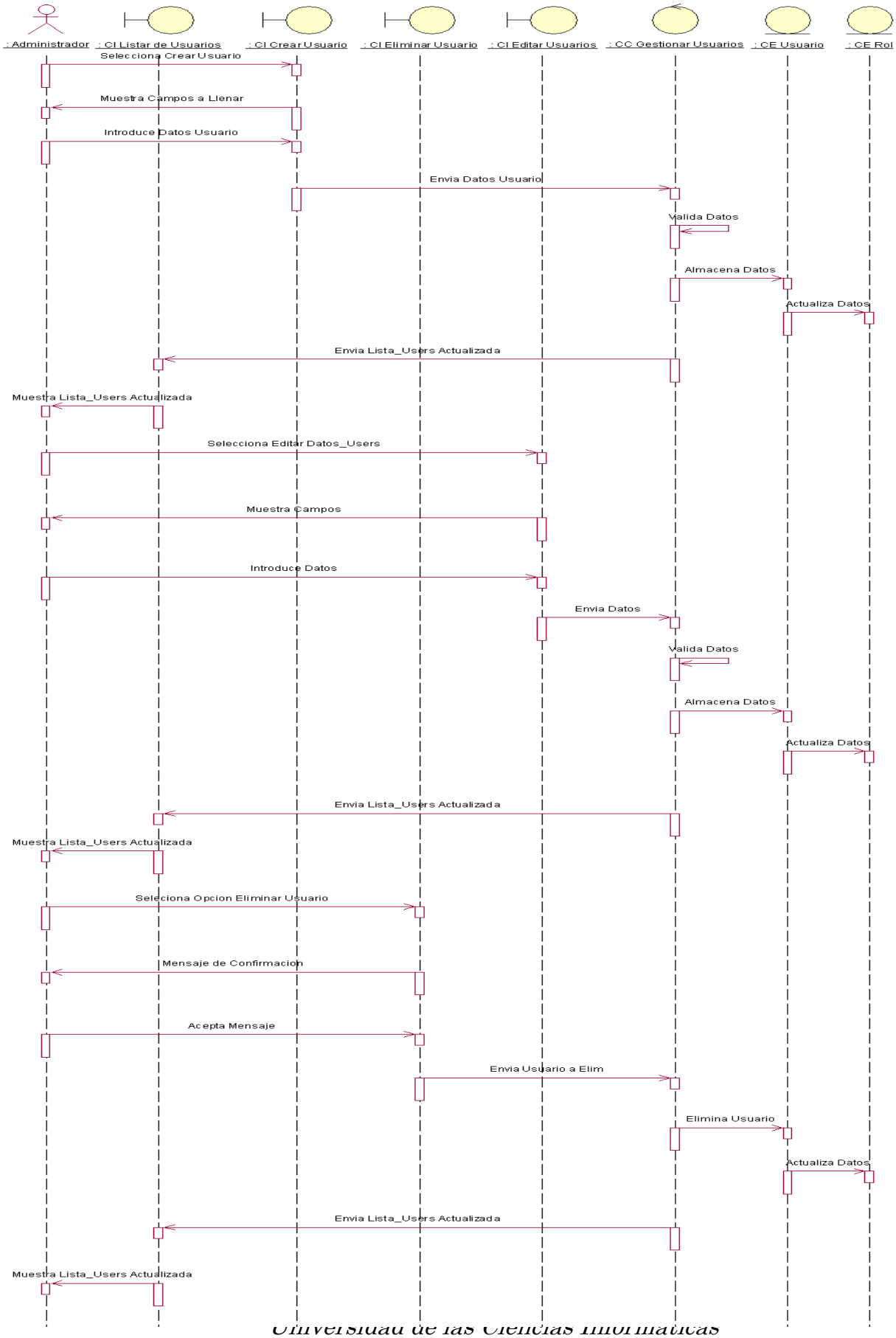
Anexo 3: Diagramas de Secuencia.

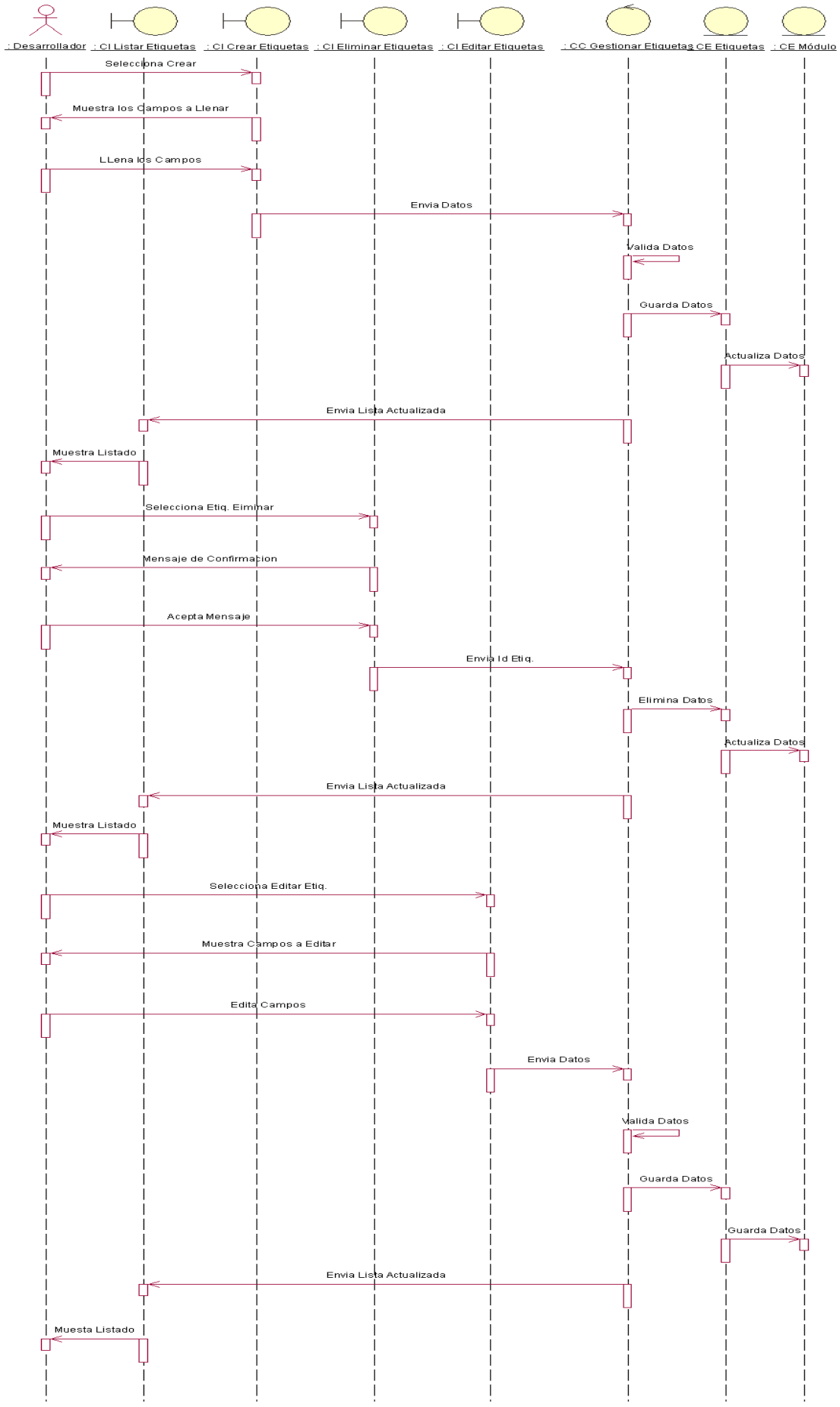
Iniciar Sesión.

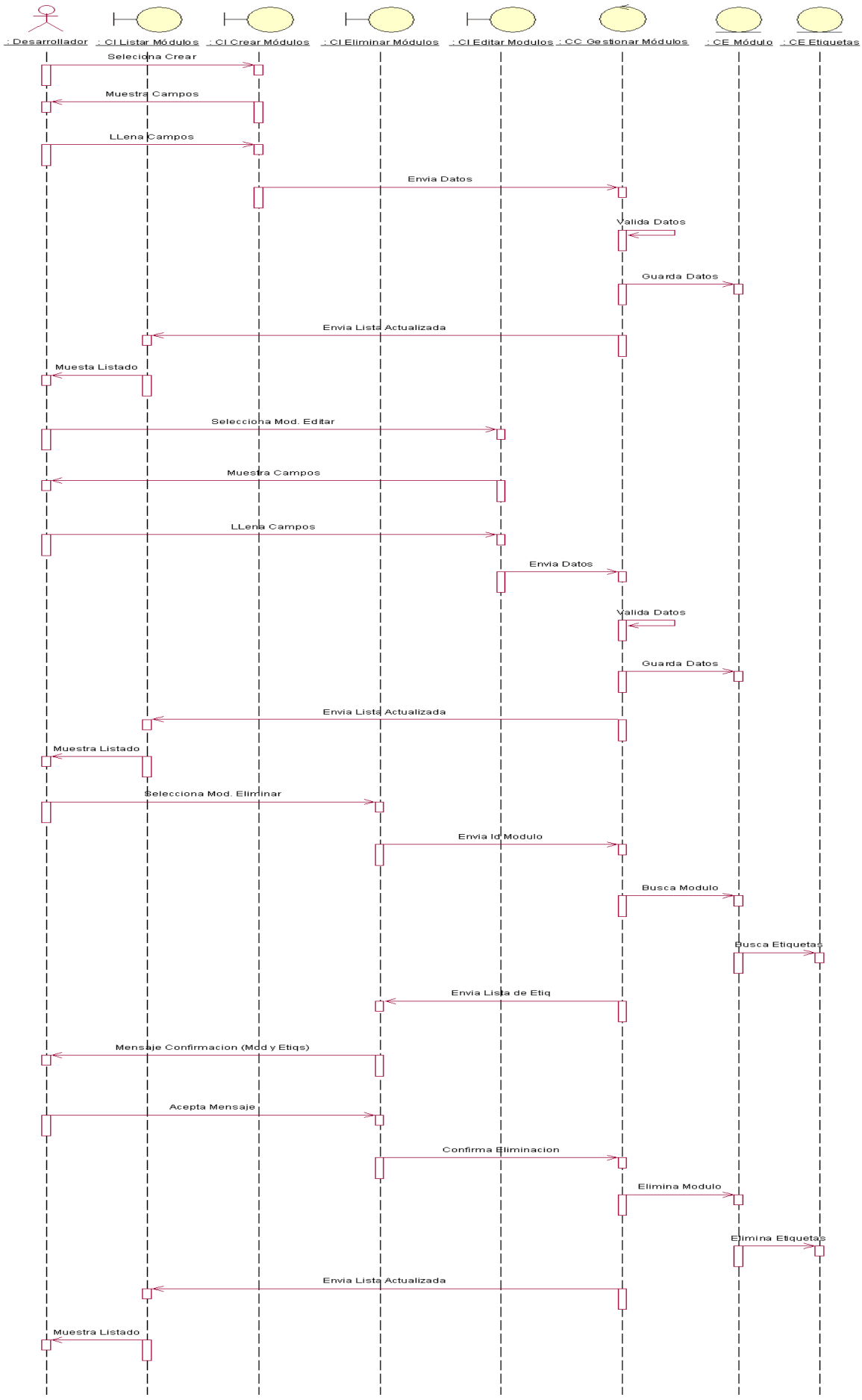


En el siguiente orden:

Administrar Usuarios.
Gestionar Etiquetas
Gestionar Módulos


: Administrador
: CI Listar de Usuarios
: CI Crear Usuario
: CI Eliminar Usuario
: CI Editar Usuarios
: CC Gestionar Usuarios
: CE Usuario
: CE Rol
Selecciona Crear Usuario
Muestra Campos a Llenar
Introduce Datos Usuario
Envia Datos Usuario
Valida Datos
Almacena Datos
Actualiza Datos
Envia Lista_Users Actualizada
Muestra Lista_Users Actualizada
Selecciona Editar Datos_Users
Muestra Campos
Introduce Datos
Envia Datos
Valida Datos
Almacena Datos
Actualiza Datos
Envia Lista_Users Actualizada
Muestra Lista_Users Actualizada
Selecciona Opcion Eliminar Usuario
Mensaje de Confirmacion
Acepta Mensaje
Envia Usuario a Elim
Elimina Usuario
Actualiza Datos
Envia Lista_Users Actualizada
Muestra Lista_Users Actualizada


: Desarrollador
: CI Listar Etiquetas
: CI Crear Etiquetas
: CI Eliminar Etiquetas
: CI Editar Etiquetas
: CC Gestionar Etiquetas
CE Etiquetas
: CE Módulo
Selecciona Crear
Muestra los Campos a Llenar
LLena los Campos
Envia Datos
Valida Datos
Guarda Datos
Actualiza Datos
Envia Lista Actualizada
Muestra Listado
Selecciona Etiq. Eimnar
Mensaje de Confirmacion
Acepta Mensaje
Envia Id Etiq.
Elimina Datos
Actualiza Datos
Envia Lista Actualizada
Muestra Listado
Selecciona Editar Etiq.
Muestra Campos a Editar
Edita Campos
Envia Datos
Valida Datos
Guarda Datos
Guarda Datos
Envia Lista Actualizada
Muesta Listado


: Desarrollador
: CI Listar Módulos
: CI Crear Módulos
: CI Eliminar Módulos
: CI Editar Modulos
: CC Gestionar Módulos
: CE Módulo
: CE Etiquetas
Selecciona Crear
Muestra Campos
LLena Campos
Envia Datos
Valida Datos
Guarda Datos
Envia Lista Actualizada
Muesta Listado
Selecciona Mod. Editar
Muestra Campos
LLena Campos
Envia Datos
Valida Datos
Guarda Datos
Envia Lista Actualizada
Muestra Listado
Selecciona Mod. Eliminar
Envia Id Modulo
Busca Modulo
Busca Etiquetas
Envia Lista de Etiq
Mensaje Confirmacion (Mod y Etiqs)
Acepta Mensaje
Confirma Eliminacion
Elimina Modulo
Elimina Etiquetas
Envia Lista Actualizada
Muestra Listado

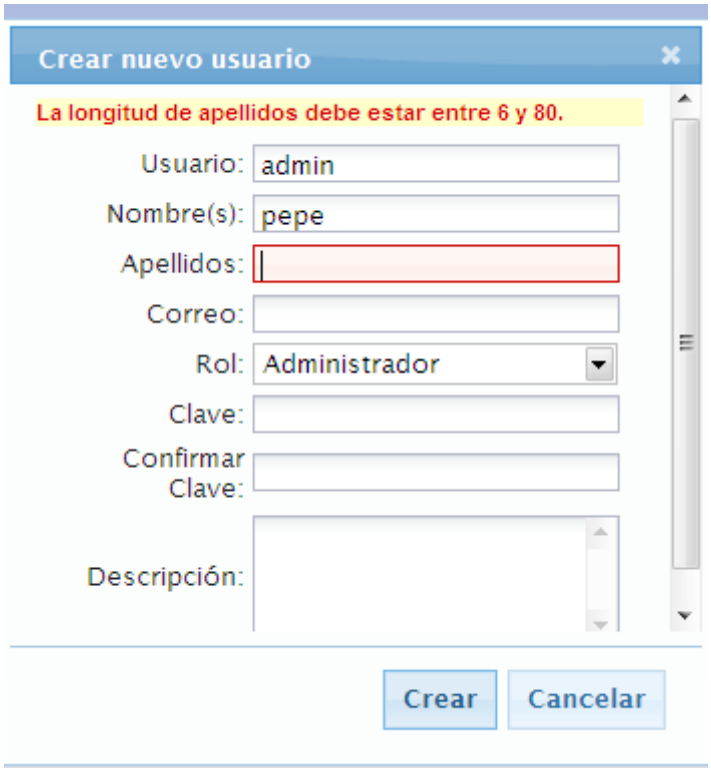
Anexo 4: Prototipos de Interfaz.

Caso de uso: Iniciar Sesión. Flujo normal de eventos.

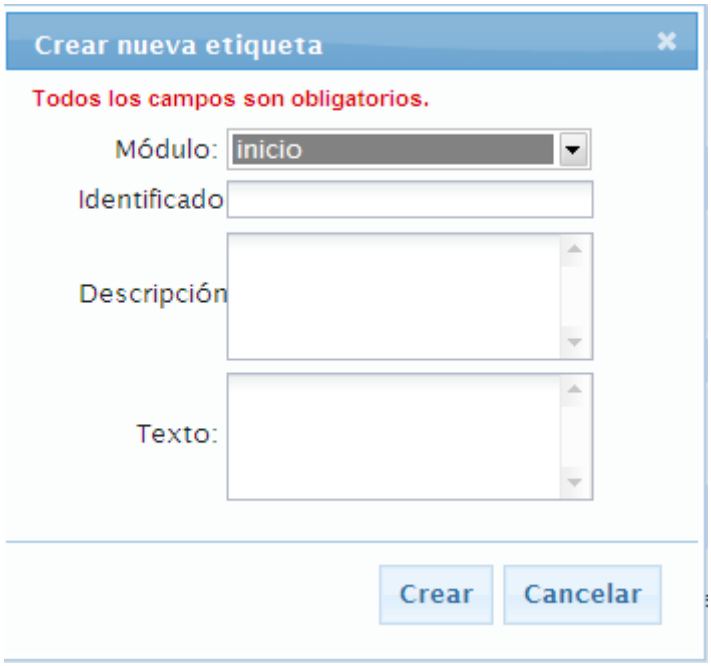
Caso de uso: Iniciar Sesión. Flujo alterno de eventos.

Caso de uso: Administradr Usuarios. Flujo normal de eventos.

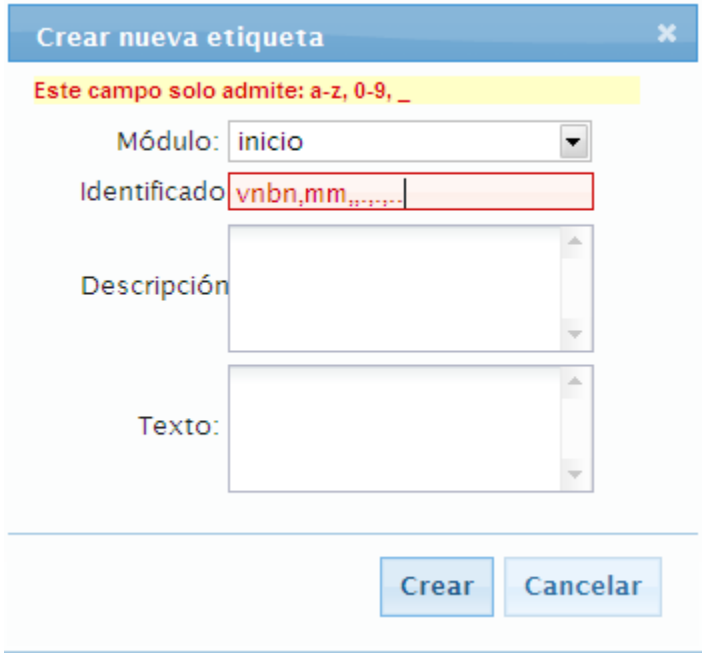
Caso de uso: Administradr Usuarios. Flujo alterno de eventos.



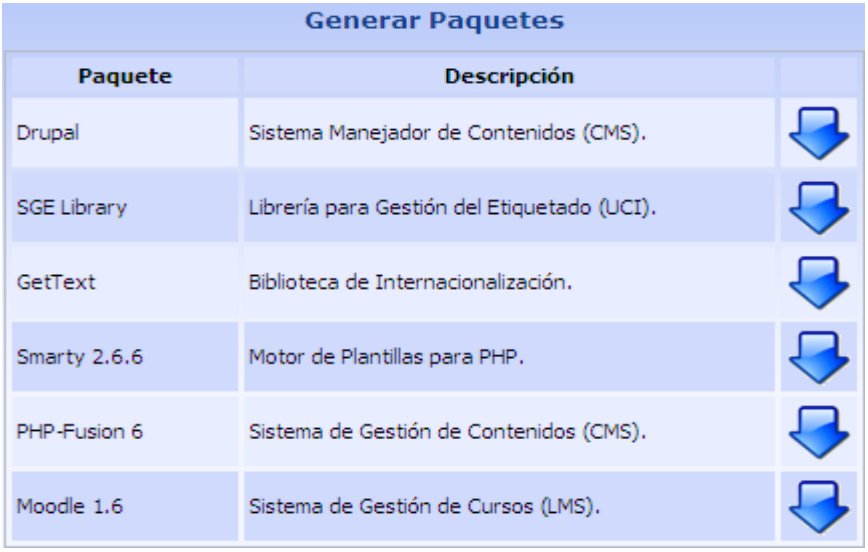
Caso de uso: Gestionar etiquetas. Flujo normal de eventos.



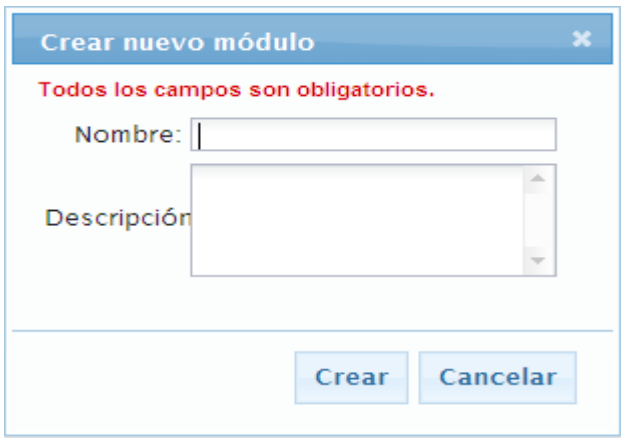
Caso de uso: Gestionar etiquetas. Flujo alterno de eventos.



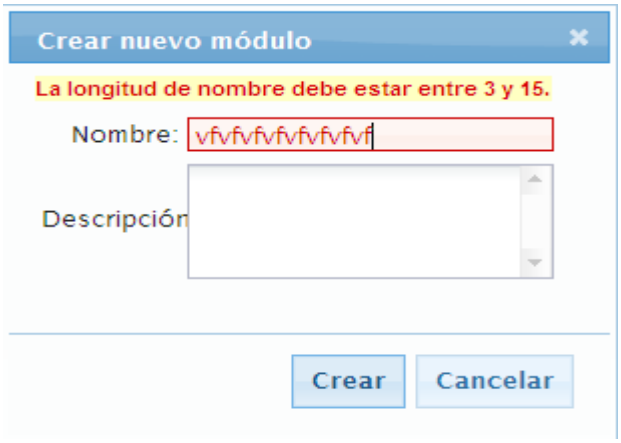
Caso de uso: Obtener paquetes de recursos. Flujo alterno de eventos.



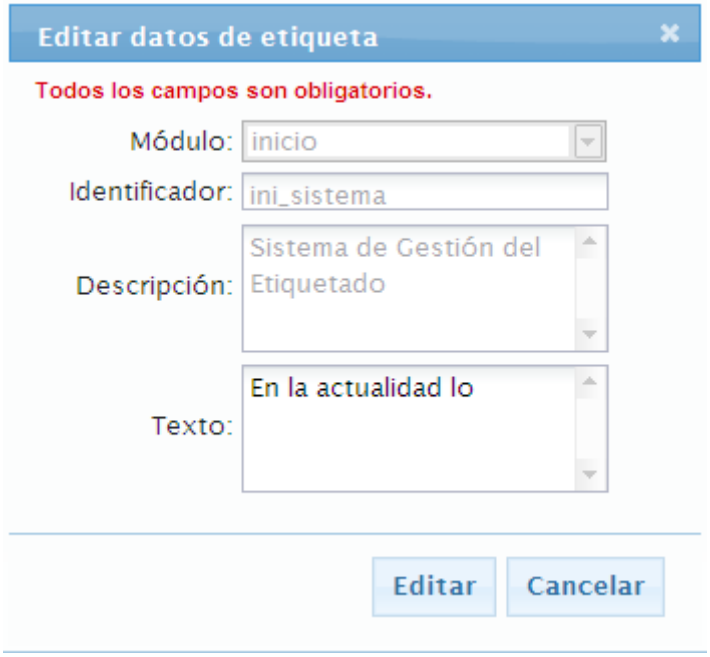
Caso de uso: Gestionar módulos. Flujo normal de eventos.



Caso de uso: Gestionar etiquetas. Flujo alterno de eventos.



Caso de uso: Editar etiquetas. Flujo alterno de eventos.



Anexo 5: Descripción de casos de uso.

Descripción del caso de uso: Cerrar Sesión.

Caso de Uso:	Cerrar Sesión
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando cualquier usuario se dispone a cerrar su sesión.
Precondiciones:	Debe estar logueado en el sistema.
Referencias	RF
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la funcionalidad cerrar sesión. 3. - Acepta el mensaje.	2.- Muestra un mensaje de diálogo "Esta seguro que desea cerrar su sesión" con las opciones "Aceptar" y "Cancelar". 4.- Cierra la sesión del usuario. 5.- Muestra una interfaz para permitir el acceso al sistema.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la funcionalidad cerrar sesión. 3.- Cancela el mensaje.	2.- Muestra un mensaje de diálogo "Esta seguro que desea cerrar su sesión" con las opciones "Aceptar" y "Cancelar".
Prototipo de Interfaz	

Poscondiciones	El usuario no estará logueado en el sistema.

Descripción del caso de uso: Listar Etiquetas.

Caso de Uso:	Listar Etiquetas
Actores:	Gestor de Información, Desarrollador.
Resumen:	El caso de uso se inicia cuando un Gestor de Información o un Desarrollador se disponen a listar las etiquetas que existen actualmente en el sistema.
Precondiciones :	El usuario debe tener el nivel de acceso necesario para poder realizar esta funcionalidad.
Referencias	RF
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la funcionalidad etiquetas	2.- Muestra la lista actual de las etiquetas que se encuentran en el sistema.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	

Poscondiciones	El sistema mostrará la lista de etiquetas con las funcionalidades que podrá realizar el usuario según su rol.

Descripción del caso de uso: Listar Módulos.

Caso de Uso:	Listar Módulos.
Actores:	Desarrollador.
Resumen:	El caso de uso se inicia cuando un Desarrollador se dispone a listar las los módulos que existen actualmente en el sistema, para poder

	efectuar el trabajo con los mismos.
Precondiciones :	El usuario debe de estar logueado en el sistema como Desarrollador.
Referencias	RF
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. - Selecciona la funcionalidad módulos.	2.- Muestra la lista actual de los módulos que se encuentran en el sistema.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	El sistema mostrará la lista de módulos.

Descripción del caso de uso: Listar Usuarios.

Caso de Uso:	Listar Usuarios.
Actores:	Administrador del Sistema.
Resumen:	El caso de uso se inicia cuando un Administrador se dispone a listar las los usuarios que existen actualmente en el sistema.
Precondiciones :	El usuario debe de estar logueado en el sistema como Administrador.
Referencias	RF
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.- Selecciona la funcionalidad usuarios.	2.- Muestra la lista actual de los

	usuarios que se encuentran en el sistema.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	El sistema mostrará la lista de usuarios.

Anexo 6. Plantillas Utilizadas.

Plantilla para Actores del Sistema:

Actor	Descripción

Plantilla para Descripción de Casos de Uso del sistema:

Caso de Uso:	
Actores:	
Resumen:	
Precondiciones:	
Referencias	
Prioridad	
Flujo Normal de Eventos	
Sección “”	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

Prototipo de Interfaz	
Poscondiciones	