



***UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS***

***Facultad 7***


***TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO  
EN CIENCIAS INFORMÁTICAS***

***TÍTULO: Módulo de Configuración del sistema para la Gestión de Información  
del Expediente de Proyecto en su variante Web***

***AUTORES: Alexander Alvarez Galán  
Daymi Alvarez Martínez***

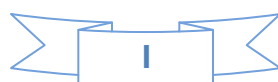
***TUTORA: Ing. Yenisel Molina Hernández  
CO-TUTOR: Ing. Dariel Fernando Reyes Prieto***

La Habana, junio 2011  
"Año 53 de la Revolución"



*Cuando crezcas, descubrirás que ya defendiste mentiras, te engañaste a ti mismo o sufriste por tonterías. Si eres un buen guerrero, no te culparás por ello, pero tampoco dejarás que tus errores se repitan.*

*Pablo Neruda.*



***DATOS DE CONTACTO***

**Tutor:** Ing. Yenisel Molina Hernández, Profesor graduado de Ingeniero en Ciencias Informática en el año 2007 en la UCI. Categoría docente: Instructor. Ha impartido las asignaturas Practica Profesional III, V y Metodología de la Investigación Científica. Se desempeña como responsable de las tesis de la facultad 7. Dirección de correo electrónico: [ymolinah@uci.cu](mailto:ymolinah@uci.cu)

**Co-Tutor:** Ing. Dariel F. Reyes Prieto, Profesor graduado de Ingeniero en Ciencias Informática en el año 2009 en la UCI. Categoría docente: Instructor Recién Graduado. Ha impartido las asignaturas Ingeniería de Software. Se desempeña como jefe de Pruebas en el grupo de Calidad del CESIM. Dirección de correo electrónico: [dfreyes@uci.cu](mailto:dfreyes@uci.cu)



**DEDICATORIA**

*Dedico este trabajo de diploma a mis padres Carmen Delia y Alfredo, por haber dedicado toda su vida a hacer de mí la persona que soy hoy, por enseñarme el camino del bien y estar para mí siempre que los necesito, por confiar en mí y ayudarme a hacer realidad mis sueños. Hoy somos ingenieros. Los quiero.*

*Alexander Alvarez Galán.*



*Dedico este trabajo a las personas más importantes en mi vida:*

*Alguien que llora con tus tristezas y ríe con tus alegrías, pero no te deja solo; alguien que te mira con los ojos y te toca el alma; alguien que lucha por ti aún cuando tu fuerzas han fallecido y es precisamente a mi madre que es la mejor mamá de este mundo, por su ternura, dedicación, por ser tan comprensiva y tolerante. Te quiero.*

*A mi papá que es el hombre más bueno, sincero y modesto de este mundo. Por apoyarme siempre en mis decisiones y confiar en mí, en las cosas que puedo lograr. Para mí ha sido, es y será un ejemplo a seguir, mi fuente de inspiración, te amo.*

*A mi hermana Dairys, que es mi faro, mi luz, la persona por la cual me rijo cada día para ser mejor, el espíritu de lucha y consagración, el ejemplo de que las metas se logran, a ti te dedico con todo mi amor este resultado, por la confianza en mí.*

*A mi sobrina Isabella, que es lo más lindo que la vida me ha brindado, por su inteligencia, su inocencia, como cuando se es pequeño, frágil y tan lleno de vida, que pasas por ella sin preocupación. Eres mi alegría.*

*A mi novio Roidel por su apoyo incondicional aún estando lejos, su paciencia, comprensión, su amor, por hacerme reír cuando lo necesitaba, y ayudarme en los momentos más difíciles. Te amo.*

*A Juan Carlos por todo su apoyo durante estos cinco años de duro batallar por lograr mi sueño, por su dedicación, por ser mi amigo, mi confidente, la persona que siempre me dio su voto de confianza, por mostrarme el camino a seguir cuando pensaba que todo estaba perdido, por toda su comprensión y por soportar mis malcriadeces. Muchas gracias.*

*A mis compañeros que me acompañaron durante todo este tiempo y a los que desgraciadamente hoy no están, siempre los llevo conmigo.*

*Daymi Álvarez Martínez.*

## **AGRADECIMIENTOS**

*A mis padres Carmen Delia y Alfredo por darme la vida y llenarme de amor, por ser mi luz y enseñarme el buen camino para triunfar en la vida y no dejarme caer, por ser los mejores padres del mundo y por estar siempre para mí. Gracias por haberme educado y hacer de mí una mejor persona. Estoy orgulloso de ser como soy, y eso se lo debo a ustedes. Hoy somos ingenieros. Los quiero.*

*A mi hermano Afredito (Pipo), muchas gracias por todas las veces que me haz defendido y apoyado, y por seguirlo haciendo. Podrá pasar el tiempo y quedar guardadas las palabras, pero nunca cambiará el amor que siento por ti y mi eterno agradecimiento por estar ahí. Te amo hermano.*

*A mi novia Yannia por ser mi mejor amiga en estos 7 años, por hacerme feliz y estar conmigo en las buenas y en las malas, por darme todo su amor y su corazón. Te amo.*

*A mis abuelas, abuelos, tías, tíos, primos, primas, sobrinas, cuñados, en fin a toda mi familia y vecinos por creer en mí.*

*A mis suegros Niuvis y Orestes por hacerme sentir parte de su familia.*

*A mis amigos de la Universidad y a todos mis compañeros de aula por apoyarme cuando los necesitaba.*

*A la aldea José Adrián (El prieto), Yasmany (Yherry), Alberto, Pedro (Peter) y Reinaldo (Rey) por ser mi familia durante los últimos 4 años de la Universidad. Los quiero y me quedo corto y si...*

*A nuestros tutores por apoyarme y defenderme durante el transcurso de la tesis.*

*A nuestro tribunal por sus constructivas críticas.*

*A todas las personas que de una forma u otra hicieron posible este sueño.*

*Alexander Alvarez Galán.*

*A mis padres por darme el regalo más grande: la vida. Por enseñarme a luchar para hacer realidad mis sueños. Por demostrarme que sí se puede. Por todo el amor y cariño infinito que siempre me han acompañado, por creer en mí. Por ayudar en mi formación como mujer, como persona y como profesional. A ustedes los amo.*

*A mi hermana Dairys por estar siempre presente, por quererme y cuidar de mí. Te quiero mucho sin tu ejemplo nunca hubiera intentado lograr llegar a donde estoy hoy.*

*A mi familia por todo el amor y el apoyo que siempre he recibido de ellos, por todo cuanto han hecho para hacerme feliz. A todos los amo.*

*A mi novio por su cariño y amor, por todo el apoyo en este tiempo tan importante para mí, por estar siempre a mi lado, por todos los momentos de felicidad. Te quiero.*

*A Juan Carlos por su cariño, apoyo y comprensión, por ser mi amigo, la persona que siempre estuvo ahí cuando lo necesité, por aportar su granito de arena en que yo sea hoy una mejor persona.*

*A mis compañeras de apartamento por su compañía, su amistad. A Cary, Yaro, Grisel, Yanicet, que venimos juntas desde primer año, gracias por su apoyo.*

*Daymi Álvarez Martínez.*

## **RESUMEN**

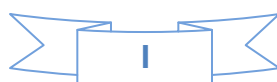
La Universidad de las Ciencias Informáticas (UCI), realiza la gestión documental de los proyectos productivos a través de un Expediente de Proyecto (EP), creado con el objetivo de recopilar y documentar toda la información referente al producto y su elaboración.

Para dar solución a las dificultades que traía consigo la confección de las plantillas que conformaban el Expediente de Proyecto, se desarrolló una herramienta que permite la gestión de la información del EP, pero presenta un conjunto de dificultades. Para darle solución a las mismas se definió como objetivo general del presente trabajo desarrollar el módulo de configuración para la variante web del Sistema de Gestión de Información del EP (GIEP).

Para el desarrollo del módulo se seleccionó como metodología RUP, lenguaje de modelado UML 2.0, como herramienta case Enterprise Architect, Se determinó como lenguaje de programación C#. El IDE a utilizar es el Visual Studio 2008, Como marco de trabajo se selecciona Microsoft .NET 3.5.

Con el desarrollo del sistema propuesto se pretende lograr un mejor manejo de la información y estructura de los EP. Permitirá gestionar los roles de los usuarios en el sistema así como establecer las reglas de acceso a las funcionalidades a las cuales el usuario tiene permiso. Además de llevar el control de las acciones realizadas por los usuarios autenticados en la aplicación y lograr la integración con la herramienta de gestión de proyecto Redmine.

**Palabras Claves:** Gestión de la Información, Gestión Documental, Expediente de Proyecto, Módulo de Configuración, Redmine.





**ÍNDICE**

**INTRODUCCIÓN**..... 1

**CAPÍTULO 1: Fundamentación Teórica**..... 5

    1.1 **Gestión de Proyectos**..... 5

    1.2 **Sistemas de Gestión Documental**..... 6

    1.3 **Expediente de proyecto**..... 9

    1.4 **Análisis de soluciones existentes**..... 9

    1.5 **Tecnologías y Herramientas**..... 14

    1.6 **Comparación de C# vs C++ y Java**..... 16

    1.7 **Entornos de Desarrollo Integrados (IDE)**..... 17

    1.8 **Metodologías de desarrollo**..... 17

    1.9 **El Lenguaje Unificado de Modelado (UML)**..... 22

    1.10 **Herramientas CASE**..... 23

**CAPÍTULO 2: Características del sistema**..... 28

    2.1 **Objeto de Automatización**..... 28

    2.2 **Modelo de dominio**..... 28

    2.3 **Levantamiento de requisitos**..... 31

    2.4 **Descripción del Sistema Propuesto**..... 35

    2.5 **Diagrama de Casos de Uso del Sistema**..... 36

    2.6 **Descripción de los Casos de Uso del Sistema**..... 38

**CAPÍTULO 3: Diseño del sistema**..... 43

    3.1 **Fundamentación del uso de patrones de Diseño**..... 43

    3.2 **Diagramas de Clases**..... 48

    3.3 **Diagramas de secuencia**..... 49

    3.4 **Descripción de las clases del diseño**..... 51

    3.5 **Descripción de la Arquitectura**..... 51

**CAPÍTULO 4: Implementación y prueba**..... 53

    4.1 **Modelo de Despliegue**..... 53

    4.2 **Modelo de Implementación**..... 54

    4.3 **Validación de la solución propuesta**..... 57



<b>CONCLUSIONES.....</b>	<b>60</b>
<b>RECOMENDACIONES.....</b>	<b>61</b>
<b>REFERENCIA BIBLIOGRÁFICA.....</b>	<b>62</b>
<b>BIBLIOGRAFÍA .....</b>	<b>64</b>
<b>GLOSARIO DE TÉRMINOS.....</b>	<b>66</b>



**ÍNDICE DE TABLAS**

Tabla 1 Criterios para la gestión documental.....	8
Tabla 2 Configuración del Alfresco.....	11
Tabla 3 Descripción del actor del sistema.....	35
Tabla 4 Descripción Textual del Caso de Uso Buscar Bitácora.....	40
Tabla 5 Descripción Textual del Caso de Uso Eliminar Bitácora.....	41
Tabla 6 Descripción Textual del Caso de Uso Exportar a PDF.....	42
Tabla 7 Descripción de la clase Agregar Proyecto.....	51
Tabla 8 Descripción de la clase Buscar Proyecto.....	51
Tabla 9 Diseño de prueba del Caso de Uso Gestionar Proyecto.....	59

**ÍNDICE DE FIGURAS**

Figura 1 Configuración del Subversion. .... 10

Figura 2 Diagrama de clases del Modelo de Dominio. .... 30

Figura 3 Diagrama de Casos de Uso del Sistema de los procesos de Configuración del Sistema. .... 36

Figura 4 Diagrama de Casos de Uso del Sistema de los procesos de Gestión de Proyecto. .... 37

Figura 5 Diagrama de Casos de Uso del Sistema de los procesos de Gestión de los EP. .... 37

Figura 6 Diagrama de clase del diseño del Caso de Uso Gestionar Proyecto. .... 49

Figura 7 Diagrama de Secuencia del Caso de Uso Gestionar Proyecto. .... 50

Figura 8 Arquitectura en capas. .... 52

Figura 9 Modelo de Despliegue. .... 54

Figura 10 Vista General Del Diagrama de Componentes. .... 55

Figura 11 Vista Detallada Del Paquete Presentación. .... 55

Figura 12 Vista Detallada Del Paquete Lógica. .... 56

Figura 13 Vista Detallada Del Paquete Acceso a Datos. .... 56

### ***INTRODUCCIÓN***

La industria del software comenzó en los últimos años de la década de 1950, donde el uso de las computadoras para los negocios se amplió rápidamente, creando una demanda enorme para las personas con experiencia de programación. Actualmente, el software es un factor primordial en el desarrollo de la tecnología, ha experimentado profundos cambios y avances sorprendentes, ya que ha facilitado y agilizado varios procesos que con anterioridad se manejaban manualmente, y además por su característica de controlar o hacer accesible, en la mayoría de los casos, los adelantos electrónicos.

El uso del software ha permitido a muchas empresas optimizar la producción, y de esta forma obtener mayores ganancias. Hace ya algunos años, Cuba se ha ido adentrando poco a poco en el desarrollo de software, creando empresas para su producción y comercialización.

La Universidad de las Ciencias Informáticas (UCI) se ha convertido, en muy poco tiempo, en centro de referencia para la Industria Cubana del Software, con el objetivo de ayudar e impulsar el desarrollo de software en el país. Esta emplea un modelo de formación vinculado a la producción y a la investigación, donde debe tenerse en cuenta todo procedimiento y plantilla que se defina. Todo esto ha impuesto un reto en cuanto a la organización de la producción, con la máxima de que la cantidad no puede afectar la calidad.

Con el objetivo de garantizar el manejo adecuado de la información surge el término Gestión Documental, que no es más que un conjunto de normas técnicas y prácticas, usadas para administrar el flujo de documentos, permite recuperar la información y determinar el tiempo en que los documentos deben guardarse, así como eliminar los que ya no son de utilidad y asegurar la conservación indefinida de los más valiosos. Por tanto, desde que un proyecto se inicia debe abrirse un expediente que recopile toda la documentación necesaria del mismo, desde su concepción hasta su finalización, para garantizar la calidad del proceso de producción del software como del producto en sí.

La confección de los diferentes artefactos que componen el Expediente de Proyecto (EP) eran llenados, en su mayoría, por los distintos roles que conforman los proyectos productivos, realizándose la elaboración de estos de forma manual y muchas veces alterándose la uniformidad en su estructura. Aunque la universidad tiene definidos patrones de arquitectura estándares para todos los software y su

documentación, estos no siempre se cumplen, en ocasiones se omite el llenado de diversos documentos y/o formularios por ser demasiados complejos o porque requieren mucho tiempo para su confección.

Algunos de los problemas que afectaban la calidad de este proceso eran:

- Existencia de información redundante en las diferentes plantillas.
- Se altera la uniformidad en la estructura de las plantillas.
- Complejidad en la estructura de algunas de las plantillas que dificultan su llenado.
- Omisión de datos en la estructura de las plantillas (Ej.: autores, fecha, módulo, versión).

Para solucionar estas dificultades, se determinó desarrollar en el grupo de calidad del Centro de Informática Médica (CESIM), la herramienta Gestión de la Información del Expediente de Proyecto (GIEP). Dicha herramienta asegura un rápido y dinámico llenado de los documentos que se generan durante el ciclo de vida de un software, lo cual evita las mayores dificultades existentes en el momento de implementada, como la información redundante y la complejidad de llenado de algunas plantillas. Además cuenta con una fácil y sencilla configuración que solo permite modificar la interfaz y la fuente<sup>1</sup>, así como seleccionar las plantillas de acuerdo al EP que se esté elaborando.

Esta herramienta al ser de escritorio, se encuentra instalada en las computadoras donde trabajan los miembros del proyecto que están autorizados a modificar el EP, la misma no utiliza una base de datos, para retroalimentarse y además los documentos generados por la herramienta no son flexibles a cambios en cuanto a su estructura.

GIEP no define las funcionalidades que permiten la integración con otras herramientas y la retroalimentación de información contenida en estas. Al ser una aplicación portable, no permite que varios usuarios actualicen el EP en tiempo real. Es decir que, los cambios realizados por un usuario son guardados en la computadora donde está trabajando, por lo tanto los cambios no pueden ser visibles por otros usuarios que se encuentren trabajando en ese EP. No admite cambiar la estructura del EP y ajustarlo a diferentes metodologías de desarrollo de software, provoca que no sea adaptable a diferentes versiones. Al no permitir cambiar el esquema del EP, impide modificarlo en cuanto a la localización de sus documentos.

---

<sup>1</sup>Ventana que permite modificar el tipo, estilo y tamaño de la letra en un documento.

Dada la situación problemática planteada, se define como **problema a resolver**: ¿Cómo mejorar la configuración de la herramienta GIEP en su variante Web?

Siendo el **objeto de estudio**: Mecanismos de configuración de aplicaciones web, enmarcado en el **campo de acción**: Mecanismos de configuración de herramientas de gestión documental. Se plantea como **objetivo general**: Desarrollar el módulo de configuración del sistema GIEP en su variante Web.

Para dar cumplimiento a los objetivos se trazó el siguiente plan de **tareas de la investigación**:

- Analizar las deficiencias de la versión desarrollada de la herramienta GIEP en cuanto su configuración.
- Analizar las herramientas de gestión de información y las funcionalidades asociadas a la configuración.
- Caracterizar las herramientas, de gestión de proyecto y gestión documental definidas en la universidad y utilizada en el CESIM.
- Evaluar las metodologías de desarrollo y herramientas de modelado, los lenguajes de programación, plataformas y Entorno Integrado de Desarrollo (IDE) así como tecnologías y librerías existentes que puedan ser usadas en el desarrollo del sistema propuesto.
- Seleccionar la metodología de desarrollo, herramienta de modelado, lenguaje de programación, plataforma y Entorno Integrado de Desarrollo (IDE) dentro de los establecidos por el CESIM.
- Analizar la estructura del expediente de proyecto de acuerdo a las distintas versiones que se utilizan en la UCI.
- Analizar los aspectos comunes entre los módulos que conformarán el sistema para su posterior integración.
- Definir la prioridad de los componentes a implementar en el sistema.
- Elaborar la documentación correspondiente a los flujos de trabajo propuestos por la metodología seleccionada.
- Implementar las funcionalidades definidas.
- Desarrollar pruebas de caja negra al sistema obtenido.

Con el desarrollo del sistema propuesto se pretende lograr un mejor manejo de la información y estructura de los EP. Permitirá gestionar los roles de los usuarios en el sistema así como establecer las reglas de acceso a las funcionalidades a las cuales el usuario tiene permiso en el sistema. Llevar el control de las acciones realizadas por los usuarios autenticados en la aplicación y lograr la integración con la herramienta de gestión de proyecto Redmine.

El presente trabajo consta de 4 capítulos estructurados de la siguiente forma:

**Capítulo 1. Fundamentación Teórica:** Contiene los conceptos fundamentales asociados a la gestión documental, así como un estudio del estado del arte sobre la configuración de diferentes sistemas de gestión de información y documental, se realiza además un análisis de las tendencias actuales en cuanto a tecnologías, herramientas y metodologías, concluyendo con la selección de las más adecuadas para el desarrollo de la solución propuesta.

**Capítulo 2. Descripción de la Arquitectura:** Especifica los requisitos funcionales y no funcionales que cumplen con dicho sistema, se describen los actores y los casos de uso del sistema.

**Capítulo 3. Implementación del sistema:** Presenta los principales artefactos para lograr la construcción del sistema como los diagramas de clases del diseño, de secuencia así como la descripción de la arquitectura definida. Se explican además los patrones de diseño que se emplean.

**Capítulo 4. Pruebas del sistema:** Describe la distribución física sobre la cual se ejecutará el sistema como el modelo de despliegue, además del diagrama de componentes y una breve descripción de un caso de pruebas.



## ***CAPÍTULO 1: Fundamentación Teórica.***

En el presente capítulo se reflejan distintos elementos que permiten fundamentar el tema abordado en el trabajo de diploma. Para lo cual, se hace alusión a los conceptos fundamentales asociados a la gestión documental, se abordan las principales características asociadas a la configuración de diferentes sistemas de gestión de información y documental, así como un análisis de las tendencias actuales en cuanto a tecnologías, herramientas y metodologías, concluyendo con la selección de las más adecuadas para el desarrollo de la solución propuesta.

### **1.1 Gestión de Proyectos.**

La gestión de proyectos es el proceso por el cual se planifica, dirige y controla el desarrollo de un sistema aceptable con un costo mínimo y dentro de un período de tiempo específico.

- **Causas de proyectos fallidos por la gestión de proyectos.**

Dentro de las principales causas por las que puede fallar un proyecto, se encuentra el hecho de que los analistas no respetan o no conocen bien las herramientas y las técnicas del análisis y diseño de sistemas, además de esto puede haber una mala gestión y dirección del proyecto. Además existen una serie de factores que pueden hacer que el sistema sea mal evaluado, entre estas están: **(Peña, 2001)**

- Necesidades no satisfechas o no identificadas.
- Cambio no controlado del ámbito del proyecto.
- Exceso de costo.
- Retrasos en la entrega.

Muchas empresas tienen su propio ciclo de vida estándar, y algunas de ellas tienen también normas sobre métodos y herramientas que han de usarse. Así ha de planificarse cada una de las tareas requeridas para completar el proyecto: **(Peña, 2001)**

- ¿Cuánto tiempo se requerirá?
- ¿Cuántas personas serán necesarias?
- ¿Cuánto costará la tarea?

- ¿Qué tareas deben terminarse antes de empezar otras?
- ¿Pueden solaparse algunas de ellas?

### 1.2 Sistemas de Gestión Documental.

#### 1.2.1 Surgimiento.

Durante siglos, la gestión documental en las organizaciones fue el dominio exclusivo de administradores, archiveros y bibliotecarios, cuyas herramientas manuales básicas eran los libros de registro, las carpetas, archivadores, cajas y estanterías en que se guardan los documentos de papel (y más tarde los audiovisuales y los documentos en soportes magnéticos u ópticos), y los ficheros que permiten hacer referencias cruzadas y una larga lista de técnicas de recuperación de información mediante sistemas de codificación y clasificación. Hace unas décadas se fueron sumando a ellos los informáticos, que son cada vez más necesarios debido a la complejidad y nivel de sofisticación que van alcanzando los sistemas computacionales de apoyo de la actividad administrativa. **(Ramirez, 2010)**

El uso del computador en la gestión documental se inicia en la práctica a partir de las grandes bibliotecas nacionales anglófonas, la Biblioteca del Congreso de los Estados Unidos de América y la British Library, que en los años 60 del siglo XX crean programas de bases de datos conocidos como MARC (Machine Readable Cataloguing) o Catalogación. Poco después se comienza también a usar registros computarizados para inventariar documentación administrativa en soporte papel. **(Ramirez, 2010)**

Cuando el uso de las tecnologías de información y comunicación se hizo común en la administración pública y privada, con el inicio de las bases de datos y la aparición de los procesadores de textos, y sobre todo con la llegada del correo electrónico, surgió la necesidad de capturar y conservar también documentos que nacen, viven y mueren en formato electrónico. Conseguir esto representó un nuevo salto en la complejidad y exigencias a los sistemas informatizados y en la forma de pensar de los administradores y archiveros. **(Ramirez, 2010)**

En la actualidad, coexisten en el mundo los más diversos sistemas de gestión documental: desde el simple registro manual de la correspondencia que entra y sale, hasta los más sofisticados sistemas informáticos que manejan no sólo la documentación administrativa, ya sea en papel o en formato electrónico, sino que además controlan los flujos de trabajo del proceso de tramitación de los expedientes, capturan información desde bases de datos de producción, contabilidad y otros, enlazan con el contenido

de archivos, bibliotecas, centros de documentación y permiten realizar búsquedas sofisticadas y recuperar información de cualquier lugar. **(Ramirez, 2010)**

### 1.2.2 Definición.

Se entiende por gestión documental al conjunto de normas técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización, permitir la recuperación de información desde ellos, determinar el tiempo que los documentos deben guardarse, eliminar los que son desechados y asegurar la conservación indefinida de los documentos más valiosos, aplicando principios de racionalización y economía. **(Ramirez, 2010)**

### 1.2.3 Criterios para la gestión documental.

Un sistema de gestión documental por lo general se refiere a las siguientes áreas: almacenamiento, recuperación, clasificación, seguridad, custodia, distribución, creación, autenticación. Para un mejor entendimiento se muestra en la tabla 1 los criterios para la gestión documental. **(Webmaster, 2009)**

Almacenamiento	¿Dónde se guardarán los documentos?
Recuperación	¿Cómo encontrar documentos necesarios? ¿Cuánto tiempo se puede pasar buscándolo? ¿Qué opciones tecnológicas están disponibles para la recuperación?
Clasificación	¿Cómo organizar los documentos? ¿Cómo asegurar que los documentos estén archivados siguiendo el sistema más apropiado?
Seguridad	¿Cómo evitar la pérdida de documentos, evitar la violación de la información o la destrucción no deseada de documentos? ¿Cómo mantener la información crítica oculta a quién no debiera tener acceso a ella?
Custodia	¿Cómo decidir qué documentos conservar? ¿Por cuánto tiempo deben ser guardados? ¿Cómo proceder a su eliminación?
Distribución	¿Cómo distribuir documentos a la persona que los necesita? ¿Cuánto se puede tardar para distribuir los documentos?
Workflow	¿Si los documentos necesitan pasar a partir de una persona a otra,

	cuáles son las reglas para el flujo de estos documentos?
Creación	¿Si más de una persona está implicada en creación o modificación de un documento, cómo se podrá colaborar en esas tareas?
Autenticación	¿Cómo proporcionar los requisitos necesarios para la validación legal al gobierno y a la industria privada acerca de la originalidad de los documentos y cumplir sus estándares para la autenticación?

Tabla 1 Criterios para la gestión documental.

#### 1.2.4 Oportunidades.

Las oportunidades de la gestión documental para las empresas son múltiples, la principal, sin duda, es la oportunidad de mejorar su productividad en el ejercicio de sus actividades y servicios hacia sus clientes. La optimización y organización de los documentos son una oportunidad de aligerar la estructura de costes, permitiendo una mayor agilidad y control sobre los gastos de la empresa, permitiendo así una mejoría del alojamiento de sus recursos y de los servicios ofrecidos.

#### 1.2.5 Ventajas.

- **Gestión y control efectivo (sencillez, rapidez y ahorro):** De una forma sencilla, la organización tiene acceso instantáneo a toda la documentación necesaria para su actividad de negocio, con las ventajas añadidas de la eliminación de desplazamientos, reducción de tiempo de consultas y tareas de archivo, ahorro de espacio físico y resolución del problema de localización de documentos. **(GIDOC INTEGRAL, S.L., 2008)**
- **Uso racional de los recursos:** La gestión documental facilita que la información se comparta y se aproveche de forma más eficiente y como un recurso colectivo. Como consecuencia, se reducen drásticamente situaciones como la duplicidad de documentos archivados, fotocopias innecesarias y dobles grabaciones de datos. **(GIDOC INTEGRAL, S.L., 2008)**
- **Seguridad y fiabilidad:** Información y documentos de gran valor para la organización, pueden custodiarse en locales de alta seguridad, garantizando su perfecto estado de conservación mientras que para el uso diario, se dispone de su réplica electrónica. **(GIDOC INTEGRAL, S.L., 2008)**
- **Productividad y valor añadido:** Una gestión documental, además de ahorro de costos, genera una productividad y valor añadido adicionales, originados por el rápido acceso a la información dentro de

la organización y su posterior distribución, sin necesidad de trasladar los documentos. (**GIDOC INTEGRAL, S.L., 2008**)

### **1.3 Expediente de proyecto.**

**Expediente de Proyecto:** Es el conjunto de documentos en soporte digital, dispuestos por orden, que informan sobre el proceso de desarrollo de software.

La UCI para elevar la calidad del producto entregado al cliente, ha diseñado un EP que está implantado en cada uno de los proyectos vigentes en la universidad. Este esquema de expediente tiene como objetivo influir en la estandarización de la documentación y la creación de una cultura de calidad en la organización.

La confección del expediente se dividió en varias etapas:

- Identificar las necesidades de documentación planteadas por CMMI v1.2.
- Identificar las necesidades propias de documentación de los proyectos de la UCI.
- Revisar plantillas propuestas por normas y estándares y adaptarlas al expediente en la UCI.
- Adicionar las plantillas que por las características de la UCI no se encuentran presentes en los modelos seleccionados.

El EP ha transcurrido por varias versiones con el objetivo de mejorar cada vez más la organización referente a la documentación del proyecto. La estructura de las distintas versiones del EP se muestra en el [Anexo 1.](#)

### **1.4 Análisis de soluciones existentes.**

En la actualidad se pueden encontrar muchas soluciones de gestores de proyectos y de gestión documental, ya sea nacional e internacionalmente. A continuación se brindan características referentes a la configuración de algunas soluciones existentes.

### 1.4.1 Herramientas desarrolladas internacionalmente.

#### ✚ Subversion.

Subversion es un software que administra el acceso a un conjunto de ficheros, y mantiene un historial de cambios realizados. El control de versiones es útil para guardar cualquier documento que cambie con frecuencia, como una novela, o el código fuente de un programa. Normalmente consiste en una copia maestra en un repositorio central, y un programa cliente, con el que cada usuario sincroniza su copia local. Esto permite compartir los cambios sobre un mismo conjunto de ficheros. Además, el repositorio guarda registro de los cambios realizados por cada usuario, y permite volver a un estado anterior en caso de necesidad.

- **Configuración.**

La configuración del Subversion, como se muestra en la figura 1, se basa específicamente en la creación de usuarios y grupos, otorgándoles a estos los permisos que pueden tener sobre un repositorio o a una carpeta determinada.

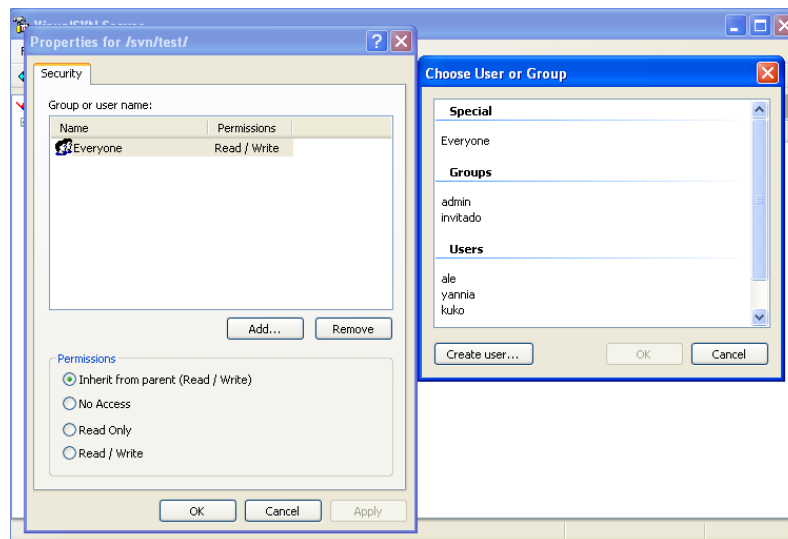


Figura 1 Configuración del Subversion.

#### ✚ Alfresco.

Alfresco es un sistema de administración de contenidos libre. Está diseñado para usuarios que requieren un alto grado de modularidad y rendimiento escalable. Incluye un repositorio de contenidos, un framework de portal Web para administrar y usar contenido estándar en portales, un sistema de administración de

contenido Web con la capacidad de visualizar aplicaciones Web y sitios estáticos y está desarrollado en Java. **(de la Fuente, 2008)**

- **Configuración.**

La configuración de Alfresco en cuanto a los permisos de los usuarios es específica, ya que la misma en dependencia del rol que interpreta, brinda los contenidos definidos para ese rol. Para un mejor entendimiento se muestra la tabla 2. **(de la Fuente, 2008)**

Lector	Editor	Contribuyente	Colaborador	Coordinador	
x	x	x	x	x	Ver
	x		x	x	Editar Contenido
		x	x	x	Crear
	x		x	x	Editar Propiedades
				x	Invitar a otros
				x	Tomar Posesión

Tabla 2 Configuración del Alfresco.

- **Redmine.**

Redmine es un gestor y planificador de proyectos con interfaz web, multiplataforma, multilenguaje y orientado a la coordinación de tareas, comunicación de participantes, y que puede especializarse en proyectos de desarrollo gracias a herramientas como la integración en un repositorio de código. **(Ceslcam, 2010)**

- **Configuración.**

La herramienta Redmine cuenta con una configuración de roles y permisos, la cual permite definir los permisos que tienen los miembros de un proyecto. Cada miembro de un proyecto tiene uno o múltiples roles para el proyecto, y además cada usuario puede tener diferentes funciones para diferentes proyectos. Se pueden crear nuevas funciones o editar las existentes. Al editar una función, se pueden definir los permisos correspondientes. **(Redmine, 2006)**

Existen dos funciones en el sistema Redmine: *No miembro* y *Anónimo*. Las mismas son de uso interno de Redmine por lo que no se pueden eliminar.

- **No miembro:** esta función le permite definir los permisos que el usuario registrado tiene en los proyectos que él no es miembro.
- **Anónimo:** esta función le permite definir los permisos que los usuarios anónimos tienen sobre los proyectos. **(Redmine, 2006)**

Como consecuencia de que los no miembros de las funciones y anónimos no se pueden asignar a un usuario o grupo específico, sino que son asignados de forma automática, los permisos de estas funciones son globales para una instalación Redmine dada. Es decir se pueden crear los usuarios con diferentes permisos en dependencia de las necesidades específicas de los proyectos. Sin embargo puede ser que en un proyecto determinado, el usuario este autorizado a realizar una actividad, pero no esté autorizado para realizar esa actividad en otro proyecto. **(Redmine, 2006)**

- **Clarity.**

Clarity es una solución global para la gestión de proyectos que permite que los gestores de cada área administren la cartera corporativa de proyectos y aplicaciones, asegurando su alineamiento con los objetivos del negocio. **(Computer Associates Int., 2005)**

- **Configuración.**

Clarity Studio es un módulo de configuración “señalar y marcar” que permite a las organizaciones crear y desplegar portales, páginas, menús y objetos empresariales personalizados, y no al revés. Los menús, páginas y vistas. Clarity utilizan procesos y un lenguaje familiar y pueden ser creados reduciéndose la necesidad de formación y aumentando la velocidad de adopción de los usuarios. Los datos críticos están



disponibles en tiempo real y en formatos que son familiares para el usuario, aumentado la productividad del usuario y la capacidad de tomar decisiones informadas. **(Computer Associates Int. (CA), 2005)**

#### **1.4.2 Herramientas desarrolladas en la UCI.**

##### **alas HIS<sup>2</sup>.**

Está conformado por módulos que responden a cada una de las áreas que se pueden encontrar en un hospital. Teniendo como característica principal la interacción, de forma tal que la información generada por cualquiera de las áreas pueda almacenarse, y al mismo tiempo fluir hasta el resto de los módulos que se encuentran interactuando. Cada uno de los módulos trabaja usando como condición necesaria la información proporcionada por el personal que interactúa con el sistema.

- **Configuración.**

El módulo controla cada aspecto de la configuración para todos los módulos que conforman al sistema. Esto evita la gestión de forma independiente en cada módulo, lo que conllevaría a un conjunto de problemas como: multiplicidad de la implementación de funcionalidades por cada módulo, la creación de usuarios con diferentes roles que al final responden a una misma persona. Entre las funcionalidades que brinda este módulo se encuentran crear, modificar, buscar, eliminar y ver datos de un usuario, rol o perfil determinado.

##### **alas RIS<sup>3</sup>**

Los RIS son sistemas ampliamente usados en las instituciones hospitalarias para la informatización de los departamentos de Diagnóstico por Imágenes. Facilita al personal de atención y organización de los servicios, las herramientas para la gestión de la información de los pacientes que son atendidos en el Departamento; así como la información de las citas otorgadas a estos, tanto para la realización de estudios de imágenes, como la realización de consultas con especialistas del Departamento. Permite la creación de las listas de trabajo para los equipos de adquisición de imágenes DICOM compatibles, así como una lista de trabajo para las consultas y especialistas.

---

<sup>2</sup>Sistema de Información Hospitalaria.

<sup>3</sup> Sistema de Información Radiológica.

- **Configuración**

El alas RIS está compuesto por varias funcionalidades una de ella es gestionar los roles y usuarios donde se puede crear, eliminar, modificar los roles de los mismos, o ver los usuarios por roles. Cuenta además con las reglas de acceso donde se pueden crear las mismas y aplicarlas a los usuarios o roles, permitiendo autorizar o denegar el acceso a determinada información, estas se aplican en el orden en que aparecen y las heredadas no pueden ser modificadas en el directorio hijo.

-  **Sistema de Gestión Penitenciario (SIGEP).**

El SIGEP constituye la solución de software para la informatización de la gestión de los privados de libertad de la República Bolivariana de Venezuela. Este módulo se encarga de simplificar el proceso de gestión de las configuraciones y definiciones de seguridad, haciéndolo más confiable y eficiente y permite gestionar las configuraciones y definiciones de seguridad mediante interfaces visuales. **(Hernández Suárez, 2008)**

- **Configuración.**

Los mecanismos de seguridad que posee el SIGEP constan de un conjunto de configuraciones y definiciones en los cuales se definen parámetros que determinan quién, a qué, cuándo y desde dónde se tiene acceso en el sistema. Permite administrar los usuarios, roles y su relación de manera sencilla y amigable, verifica los datos antes de ser almacenados y el sistema audita cada acción que se realiza sobre los mismos. **(Hernández Suárez, 2008)**

### **1.5 Tecnologías y Herramientas.**

#### **1.5.1 Lenguajes de programación.**

Los lenguajes de programación son herramientas que permiten crear programas y software. Facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computadora a utilizar. **(Ríos Aquino, et al., 2010)**

#### **1.5.2 Lenguaje C++.**

Bjarne Stroustrup crea una versión experimental denominada "C with Classes" (C con clases) hacia 1979, con la intención de proporcionar una herramienta de desarrollo para el kernel Unix en ambientes distribuidos; el objetivo de este nuevo lenguaje era mejorar algunas características del C pero

manteniendo su basamento en el mismo. Luego, en 1983 adquiere el nombre de “C++” debido a su esencia de C y con mejoras como su operador de incremento numeral (++). Es versátil, flexible, conciso y muy eficiente. C++ no es un lenguaje orientado a objetos puro, se trata simplemente del sucesor de un lenguaje de programación hecho por programadores (de alto nivel) para programadores, al cual le han ido añadiendo todos los elementos que la práctica aconsejaba como necesarios, con independencia de su belleza o purismo conceptual. **(González, 2008)**

### 1.5.3 Lenguaje Java.

Este es un lenguaje desarrollado por la compañía Sun Microsystem en los años noventa. Está inspirado en C++ y se proyectó con la finalidad de obtener un producto de pequeñas dimensiones, simple y portátil sobre diferentes plataformas y sistemas operativos ya sea a nivel de código fuente como a nivel de código binario. **(Expertos en Servicios de Consultoría Exes, 2000)**

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. Es un lenguaje que genera ficheros de clases compiladas, pero estas son en realidad interpretadas por la Máquina Virtual de Java, quien mantiene el control sobre las clases que se estén ejecutando. **(Expertos en Servicios de Consultoría Exes, S.L., 2000)**

Es un lenguaje multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la JVM. **(Expertos en Servicios de Consultoría Exes, S.L., 2000)**

Es un lenguaje seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros. **(Expertos en Servicios de Consultoría Exes, S.L., 2000)**

Su sintaxis ha sido trabajada mejorando la de C++ logrando mayor sencillez y legibilidad. Presenta mayor robustez al simplificar la gestión de memoria y eliminar las complejidades del manejo explícito de punteros. **(Expertos en Servicios de Consultoría Exes, S.L., 2000)**

Presenta capacidades avanzadas de ejecución *multi-hilo* y proporciona mecanismos de carga dinámica de clases en tiempo de ejecución. Se puede compilar y ejecutar en cualquier plataforma de sistema operativo

por ejemplo en Windows, Solaris o Linux gracias a su máquina virtual. Su desarrollo ha sido rápido y exitoso debido a la gran cantidad de grandes empresas colaboradoras que han dado su aporte para enriquecerlo.

La ejecución de programas escritos en Java suele comportarse más lenta que la de aplicaciones de otro lenguaje haciendo un uso voraz de recursos como memoria y procesador. Esto se hace más notorio si la ejecución se basa en cálculos matemáticos complejos o si la aplicación presenta un diseño cargado de componentes visuales.

### 1.5.4 Lenguaje C#.

Desarrollado por la empresa Microsoft Corporation, como una recopilación de lo mejor de C++ y Java. Es un lenguaje orientado a objetos y orientada a componentes, tiene una sintaxis muy parecida a Java y posee la potencia de C++, fue diseñado para ser sencillo, moderno y de propósito general.

## 1.6 Comparación de C# vs C++ y Java.

### 1.6.1 Ventajas frente a C++.

- Compila a código intermedio independiente del lenguaje en que haya sido escrita la aplicación e independiente de la máquina donde vaya a ejecutarse.
- Realiza la recolección automática de basura.
- Elimina el uso de punteros, en C# no son necesarios aunque permite utilizarlos.
- No hay necesidad de declarar funciones y clases antes de invocarlas.
- No existen las dependencias circulares.
- Soporta definición de clases dentro de otras.
- No existen funciones ni variables globales, todo pertenece a una clase.
- Todos los valores son inicializados antes de ser usados (automáticamente por defecto, o manualmente desde constructores estáticos).
- No se pueden utilizar valores no booleanos (enteros, coma flotante) para condicionales.
- Es mucho más limpio y menos propenso a errores.

### 1.6.2 Ventajas frente a Java.

- El rendimiento es, por lo general, mucho mejor.
- Soporta más tipos primitivos incluyendo tipos numéricos sin signo.

- Se usan indizadores que permiten acceder a cualquier objeto como si se tratase de un arreglo.
- Compilación condicional.
- Aplicaciones *multi-hilo* más sencillas de manejar.
- Soporta la sobrecarga de operadores, que aunque pueden complicar el desarrollo son opcionales y algunas veces muy útiles.
- Permite el uso de punteros cuando realmente se necesiten, como al acceder a librerías que no se ejecuten sobre la máquina virtual.

### 1.6.3 Ventajas frente al C++ y Java.

- Concepto formalizado de los métodos *get* y *set*, con lo que se consigue código mucho más legible.
- Gestión de eventos usando delegados mucho más fácil de implementar.

### 1.7 Entornos de Desarrollo Integrados (IDE).

Un Entorno de Desarrollo Integrado (IDE) es un programa que agrupa un conjunto de herramientas que viabilizan la labor de los programadores.

➤ **Visual Studio 2008** provee un nuevo lenguaje de consultas integrado para el manejo de la información, denominado Microsoft Language Integrated Query (LINQ), es fácil para programadores individuales, para poder construir soluciones que analicen y actúen sobre la información. Ofrece a desarrolladores nuevas herramientas para la fácil creación de aplicaciones conectadas en las últimas plataformas incluyendo web, Windows Vista, Office 2007, SQL Server 2008 y Windows Server 2008. Para la web, se tiene ASP.NETAJAX y otras tecnologías como Silverlight, que darán la posibilidad de crear aplicaciones con rica interfaz de usuario. **(Smith, 2008)**

➤ **MonoDevelop 2.2:** es un IDE diseñado para que los desarrolladores de .NET migren sus aplicaciones creadas en Visual Studio a Linux de forma rápida y que puedan mantener un único código base para ambas plataformas.

### 1.8 Metodologías de desarrollo.

#### 1.8.1 El Proceso Unificado de Desarrollo de Software (RUP).

Un proceso define quién está haciendo qué, cuándo y cómo para alcanzar un determinado objetivo. Un proceso de desarrollo de software es un conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto.

Este conjunto de actividades, tiene la misión de transformar los requerimientos del usuario en un producto software. Como RUP es un proceso, en su modelación define como sus principales elementos:

- **Trabajadores** (Quién): Define el comportamiento y responsabilidades de un individuo, grupo de individuos, sistema automatizado o máquina, los cuales se encargan de realizar las actividades y son los responsables de una serie de artefactos.
- **Actividades** (Cómo): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- **Artefactos** (Qué): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades.
- **Flujo de actividades** (Cuándo): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

RUP divide su ciclo de vida en cuatro fases dentro de las cuales se llevan a cabo un grupo de iteraciones.

- **Conceptualización** (Concepción o Inicio): Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
  - **Elaboración**: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.
  - **Construcción**: Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario.
  - **Transición**: El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.
- **Flujos de Trabajo.**
    - **Modelamiento del negocio**: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
    - **Requerimientos**: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
    - **Análisis y diseño**: Describe cómo el sistema será realizado a partir de las funcionalidades previstas y las restricciones impuestas, por lo que indica con precisión lo que se debe programar.

- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
  - **Prueba (Testeo):** Busca identificar los defectos y corregirlos antes de la instalación final del sistema, se verifica la integración apropiada de los componentes y que se satisfacen los requerimientos de los clientes.
  - **Instalación (Despliegue):** Produce release del producto y realiza actividades (empaquete, instalación, asistencia a usuarios, y otras) para entregar el software a los usuarios finales.
  - **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
  - **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto.
  - **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.
- **Características de RUP.**
    - Creado por Jacobson, Rumbaugh y Booch.
    - Unifica los mejores elementos de metodologías anteriores.
    - Preparado para desarrollar grandes y complejos proyectos.
    - Orientado a objetos.
    - Utiliza UML como lenguaje de representación visual.

El ciclo de vida de RUP se caracteriza por ser: dirigido por casos de uso, centrado en la arquitectura y además por ser iterativo e incremental.

Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo, obteniéndose un producto final que cuenta con la calidad requerida por el cliente. La arquitectura muestra la visión común del sistema completo, que describe los elementos del modelo que son importantes para la construcción del software, creándose de esta manera los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo. RUP propone que se

desarrolle el software mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista para la arquitectura y además que cada fase se desarrolle en iteraciones, donde en cada iteración se involucran actividades de todos los flujos de trabajo, aunque se desarrollan fundamentalmente algunos flujos más que otros.

### 1.8.2 Extreme Programming (XP).

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre.

- **Fases de XP.**

- **Fase I: Exploración.**

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

- **Fase II: Planificación de la Entrega.**

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.



### ➤ **Fase III: Iteraciones.**

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fueren la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

### ➤ **Fase IV: Producción.**

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

### ➤ **Fase V: Mantenimiento.**

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

### ➤ **Fase VI: Muerte del Proyecto.**

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

### 1.9 El Lenguaje Unificado de Modelado (UML).

Lenguaje Unificado de Modelado (UML), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de un desarrollo de software.

Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (como el Proceso Unificado Racional), pero no especifica en sí mismo qué metodología o proceso usar.

Intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado, y se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo. Permite la modificación de todos sus miembros mediante estereotipos y restricciones. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama. Está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo.

- **Objetivos de UML.**

- **Visualizar:** UML permite expresar de una forma grafica un sistema de forma que otro lo puede entender.
- **Especificar:** UML permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir:** A partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar:** los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Un modelo UML está compuesto por tres clases de bloques de construcción:

- **Elementos:** los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, entre otros).
- **Relaciones:** relacionan los elementos entre sí.

- **Diagramas:** son colecciones de elementos con sus relaciones.
- **Características de UML.**
  - Es un lenguaje de representación visual.
  - Permite combinar diversos elementos gráficos y crear diagramas.
  - Se usa solo para modelar sistemas que usan tecnología orientada a objetos.

### 1.10 Herramientas CASE.

#### 1.10.1 Rational Rose.

Es una herramienta software para el Modelado Visual mediante UML de sistemas de software. Permite Especificar, Analizar, Diseñar el sistema antes de codificarlo.

- **Características.**
  - Mantiene la consistencia de los modelos del sistema software.
  - Chequeo de la sintaxis UML.
  - Generación Documentación automáticamente.
  - Generación de Código a partir de los Modelos.
  - Ingeniería Inversa (crear modelo a partir código).
- **Ventajas.**
  - Rational Rose es la herramienta más poderosa del mercado, basado en eclipse.
  - Potente herramienta para el desarrollo de software.
  - Es una herramienta muy completa y estable como muy pocas que se han creado.
  - Facilidad de uso para el modificado y creación de nuevos diagramas.

#### 1.10.2 Visual Paradigm.

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

- **Características.**

- Producto de calidad.
- Soporta aplicaciones web.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- En adición al soporte de Modelado UML esta herramienta provee el modelado de procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java .NET y PHP.

- **Visual Paradigm ofrece:**

- Entorno de creación de diagramas para UML 2.1.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

### 1.10.3 Enterprise Architect.

- ✚ **Características.**

- **Comprensivo para UML 2.1.**

- Soporta los 13 diagramas de UML 2.1.
- Los diagramas de comportamiento incluyen: Casos de Uso, Actividades, Estado, Descripción de la interacción, Secuencia y Comunicación.
- Los diagramas de estructurales incluyen: Paquetes, Clases, Objetos, Composición, Componentes y Despliegue.

- **Interfaz de usuario intuitiva.**
  - Amplio rango de barras de herramientas, ventanas acoplables, y estilos visuales.
  - Guarda y restaura disposiciones de ventanas personalizadas.
  - Modifica y personaliza las barras de herramientas y menús.
  
- **Documentación flexible y comprensible.**
  - Documentación compatible de Word para editar posteriormente y vincular a los documentos maestros de Word.
  - Generador de informe HTML adicional para crear informes HTML detallados.
  - Los documentos en texto enriquecido se pueden vincular a los elementos del modelo y editados directamente usando el editor de texto enriquecido incorporado.
  
- **Ingeniería de código Directa e Inversa.**
  - Generador de código dirigido por plantillas completas.
  - Agrega lenguajes de destino adicionales.
  - Editor del código fuente seleccionado de sintaxis con capacidad para "guardar y sincronizar" con rapidez.
  
- **Plug-ins para vincular EA a Visual Studio.NET o Eclipse.**
  - Adaptadores de MDG Link disponibles como agregaciones separadas.
  - Enlaces para sus IDE favoritos.
  - Localiza el código fuente para las clases, atributos y operaciones en su IDE directamente desde EA.
  
- **Modelado de base de datos.**
  - Ingeniería inversa para muchos de los sistemas populares, incluyendo Oracle, SQL Server, My SQL, Access, PostgreSQL y otros.
  - Modela tablas de base de datos, columnas, claves, claves foráneas, y relaciones complejas usando UML y perfiles de modelado de datos incorporados.
  - Generación directa de los scripts DLL para crear las estructuras de base de datos.

- **Soporte de esquema XML.**

- Perfil incorporado para XSD para simplificar el desarrollo de esquemas XML usando UML.
- Genera esquemas XML complejos de modelos UML.
- Transforma modelos simples en modelos XSD usando las transformaciones MDA, luego genera XSD para guardar.

- **Ventajas:**

- Fácil de usar, rápida de entender y muy manejable.
- Genera diagramas bastante buenos gráficamente, y con más colorido que las demás herramientas.

- **Inconvenientes:**

- Es una herramienta comercial.
- La licencia de prueba es bastante restrictiva.

### 1.11 Tecnologías, metodologías y herramientas a utilizar.

Se selecciona como metodología de desarrollo RUP, es orientada a objetos y utiliza UML como lenguaje de representación visual, como lenguaje de modelado se emplea el UML ya que permite visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de desarrollo de software y es empleado para modelar sistemas que usan tecnología orientada a objetos.

Como herramienta case se selecciona Enterprise Architect, ya que brinda una interfaz de usuario intuitiva y posee un plug-ins que se integra a Visual Studio.NET. Se determinó como lenguaje de programación C# ya que es gratis, aporta características novedosas que simplifican las labores de implementación, se enriquece con la potencia de C++ y la sencillez y modernidad de Java, está debidamente estandarizado y propuesto como uno de los lenguajes a utilizar en el CESIM, plasmado en el documento de Arquitectura del Centro.

El IDE a utilizar es el Visual Studio 2008, pues provee un nuevo lenguaje de consultas integrado para el manejo de la información, denominado Microsoft Language Integrated Query (LINQ), incorpora nuevos componentes como ASP.NET, AJAX y otras tecnologías como Silverlight, que dan la posibilidad de crear aplicaciones con una amigable interfaz de usuario. Como marco de trabajo se selecciona Microsoft .NET 3.5, debido a que amplía la compatibilidad con aplicaciones móviles distribuidas, al incorporar la tecnología Windows Communication Foundation (WCF), agrega nuevas características de lenguaje como

LINQ y compatibilidad con el desarrollo de AJAX centrado en el servidor mediante un conjunto de nuevos controles y nuevas API.

Como resultado del análisis realizado durante el presente capítulo, se identificaron las principales características de los mecanismos de configuración de las herramientas y sistemas analizados, así como elementos significativos de la gestión de proyecto y documental a tener en cuenta en el desarrollo del sistema propuesto. Para el mismo se seleccionaron las herramientas, tecnologías, lenguaje de programación y metodología más adecuadas.

## ***CAPÍTULO 2: Características del sistema.***

En el presente capítulo se aborda lo referente al modelo de dominio en el cual se analizan todas las entidades y conceptos asociados al problema. Se especifican detalladamente los requisitos funcionales y no funcionales que debe cumplir el sistema, así como los actores y los casos de usos del sistema.

### **2.1 Objeto de Automatización.**

Se requiere desarrollar un módulo para la configuración del sistema de Gestión de Información del Expediente de Proyecto en su variante web, donde se pueda gestionar toda la información referente a los proyectos, estructura y organización, brindar la posibilidad de integración con las herramientas existentes para dicho proceso, con el fin de obtener los datos necesarios de ellas, administrando los usuarios, roles, niveles de acceso, proyectos, reorganizar las estructuras existentes para los documentos en el EP y guardar un registro de todas las acciones realizadas por los usuarios en el sistema.

### **2.2 Modelo de dominio.**

Un modelo de dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, teniendo como ventaja el permitir ayudar a los usuarios, clientes y desarrolladores a utilizar un vocabulario común para poder entender el contexto en que se emplaza el sistema. **(Calzado Toledano, et al., 2010)**

#### **2.2.1 Glosario de Términos del Dominio.**

❖ **Usuario.**

Persona que interactúa con la herramienta GIEP para trabajar con el EP.

❖ **EP.**

Contiene los documentos relacionados con el desarrollo de un software.

❖ **Documento.**

Archivo donde se detalla la información relacionada con un proceso del desarrollo de un software.



❖ **GIEP.**

Herramienta desarrollada por el proyecto de Calidad de la Facultad 7 para la gestión de la información del EP.

❖ **XML.**

Archivo utilizado por la herramienta GIEP para guardar y cargar información necesaria para su funcionamiento.

❖ **Plantilla.**

Artefacto del EP.

❖ **Proyecto.**

Producto de software desarrollado con un fin determinado.

2.2.2 Diagrama de clases del Modelo de Dominio.

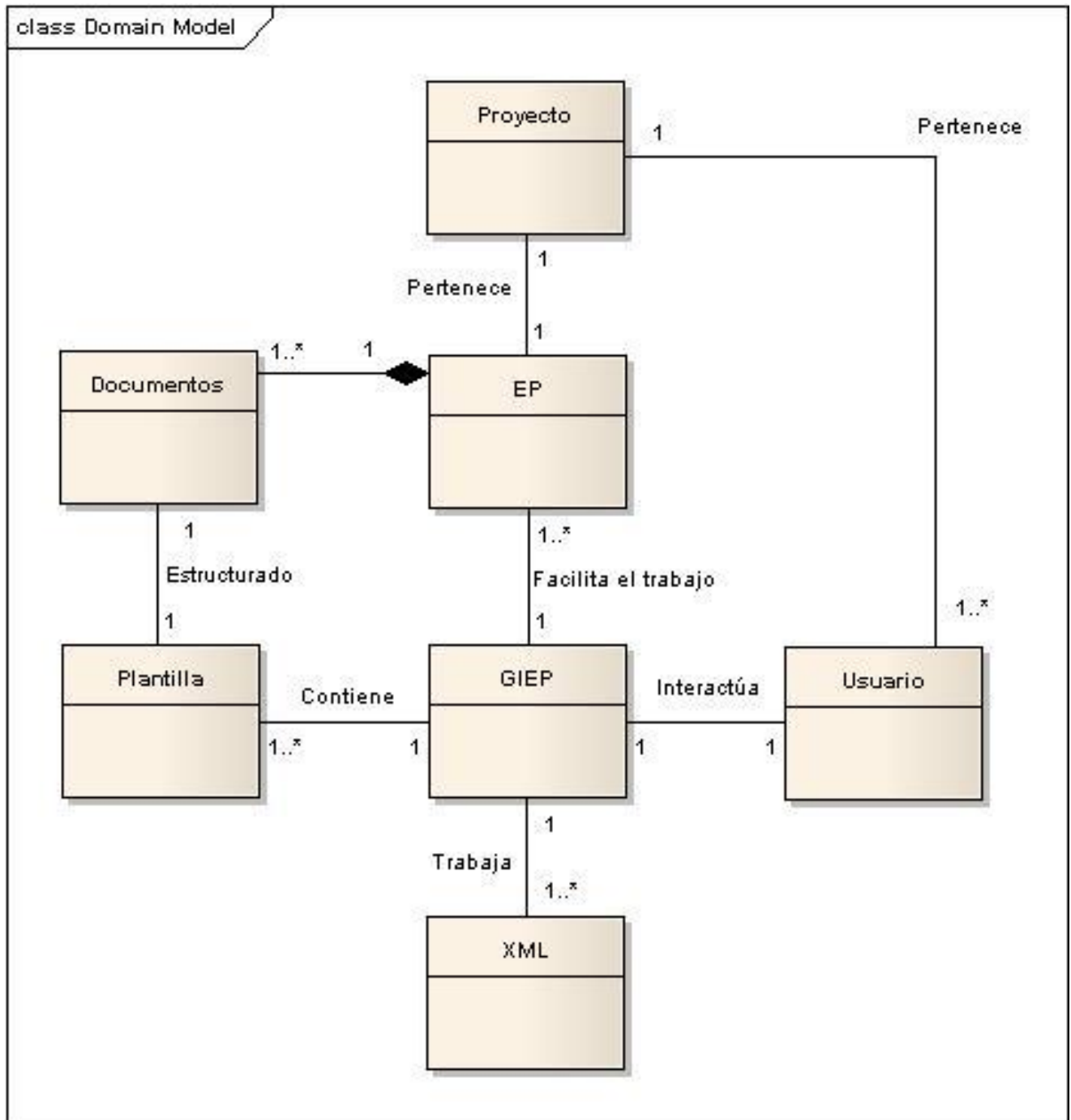


Figura 2 Diagrama de clases del Modelo de Dominio.

## **2.3 Levantamiento de requisitos.**

### **2.3.1 Requisitos funcionales del sistema.**

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen.

El sistema debe ser capaz de:

**RF1 Autenticar Usuario:** El sistema debe permitir que el usuario pueda acceder a la aplicación con su usuario y contraseña del dominio uci o con un usuario y contraseña local.

**RF2 Seleccionar Estilo:** El sistema debe permitir que se pueda seleccionar el estilo deseado a mostrar en la interfaz.

**RF3 Agregar Rol:** El sistema debe permitir que se pueda crear un rol determinado.

**RF4 Buscar Rol:** El sistema debe permitir que se pueda buscar un rol deseado por el usuario.

**RF5 Eliminar Rol:** El sistema debe permitir que se pueda eliminar un rol seleccionado.

**RF6 Modificar Rol:** El sistema debe permitir que se pueda modificar un rol seleccionado por el usuario.

**RF7 Crear Usuario:** El sistema debe permitir que se pueda crear un usuario.

**RF8 Buscar Usuario:** El sistema debe permitir que se pueda buscar un usuario.

**RF9 Asignar Rol a Usuario:** El sistema debe permitir que se pueda asignar un rol a un usuario seleccionado.

**RF10 Eliminar Usuario:** El sistema debe permitir que se pueda eliminar un usuario seleccionado.

**RF11 Agregar Proyecto:** El sistema debe permitir que se pueda agregar un proyecto.

**RF12 Buscar Proyecto:** El sistema debe permitir que se pueda buscar un proyecto deseado.

**RF13 Eliminar Proyecto:** El sistema debe permitir que se pueda eliminar un proyecto.

**RF14 Asignar Usuario a Proyecto:** El sistema debe permitir que se pueda asignar un usuario a un proyecto seleccionado.

**RF15 Importar del Redmine:** El sistema debe permitir que se pueda importar un proyecto seleccionado del Redmine

**RF16 Cargar Plantilla:** El sistema debe permitir que se pueda cargar una plantilla XML o archivo.rar seleccionada.

**RF17 Eliminar Plantilla:** El sistema debe permitir que se pueda eliminar una plantilla o archivo.rar seleccionada.

**RF18 Buscar Plantilla:** El sistema debe permitir que se pueda buscar una plantilla deseada.

**RF19 Gestionar Directorio:** El sistema debe permitir que se pueda realizar las diferentes funcionalidades sobre un expediente de proyecto.

**RF19.1 Asignar Documento a un Directorio.**

**RF19.2 Eliminar Documento de un Directorio.**

**RF20 Eliminar Estructura:** El sistema debe permitir que se pueda eliminar una estructura seleccionada.

**RF21 Buscar Estructura:** El sistema debe permitir que se pueda buscar una estructura deseada..

**RF22 Agregar Regla de Acceso:** El sistema debe permitir que se pueda agregar una regla de acceso a determinado usuario o rol.

**RF23 Buscar Regla de Acceso:** El sistema debe permitir que se pueda buscar una regla de acceso.

**RF24 Eliminar Regla de Acceso:** El sistema debe permitir que se pueda eliminar una regla de acceso seleccionada.

**RF25 Asignar Orden de Acceso:** El sistema debe permitir que se pueda asignar un orden de acceso sobre una funcionalidad determinada

**RF26 Buscar Bitácora:** El sistema debe permitir que se pueda buscar una bitácora deseada.

**RF27 Eliminar Bitácora:** El sistema debe permitir que se pueda eliminar una bitácora seleccionada.

**RF28 Exportar a PDF:** El sistema debe permitir que se pueda exportar el reporte de la bitácora en formato PDF.

### 2.3.2 Requisitos no funcionales del sistema.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades son las características que hacen al producto atractivo, usable, rápido o confiable.

#### ➤ **Usabilidad.**

- El sistema contará con una interfaz amigable al usuario, fácil de emplear, minimizando en todo los casos el número de pasos para alcanzar una funcionalidad.
- Los íconos estarán en correspondencia con las opciones del menú, teniendo una representación clara de la funcionalidad y ajustándose a los colores definidos dentro del diseño del proyecto.
- Las funcionalidades en los menús estarán agrupadas en dependencia de su objetivo.

#### ➤ **Soporte.**

- El sistema contará con la documentación asociada al proceso de desarrollo.

#### ➤ **Seguridad.**

- El sistema debe proteger la integridad de la información que se maneja otorgando los permisos necesarios a las funcionalidades en dependencia del rol o el usuario.
- Se establecerá una política en cuanto a la fortaleza y cambio de la contraseña de los usuarios locales.

#### ➤ **Portabilidad.**

- El sistema debe desarrollarse de manera tal que pueda ser instalado en diferentes plataformas.

### ➤ **Software.**

PC Cliente:

Navegador Web: Mozilla Firefox.

Sistema Operativo: Linux o Windows.

Servidor de Aplicaciones:

Internet Information Services.

Servidor de BD:

SQL Server

### ➤ **Hardware:**

PC Cliente requiere:

- Tarjeta de red.
- Al menos 256 MB de memoria RAM.
- 10 Gb de disco duro para ambos SO.
- Procesador 2.0 MHz.

Servidor de Aplicación requiere:

- Tarjeta de red.
- Al menos 1GB de RAM.
- 1GB de espacio libre en el disco para la instalación del apache y la aplicación.
- Procesador de 2.0 MHz

Servidor de Base de Datos requiere:

- Tarjeta red.
- Al menos 1GB de RAM.
- 2 GB mínimo de espacio libre en el disco duro para la instalación.
- Procesador de 2.5 GHz.

## 2.4 Descripción del Sistema Propuesto.

### 2.4.1 Descripción de los actores.

Un actor del sistema no es parte de la aplicación en desarrollo, es un agente externo que intercambia información con el mismo en pos de obtener un resultado esperado. Los actores definidos para el sistema en cuestión son:

Actor	Descripción
Usuario	Es la persona que se autentica en el sistema y una vez registrado se le asignan los roles y permisos para acceder a determinada funcionalidad.
Administrador	Es la persona que gestiona las funcionalidades de configuración del sistema y además asigna los roles y permisos a los usuarios registrados.

Tabla 3 Descripción del actor del sistema.

### 2.5 Diagrama de Casos de Uso del Sistema.

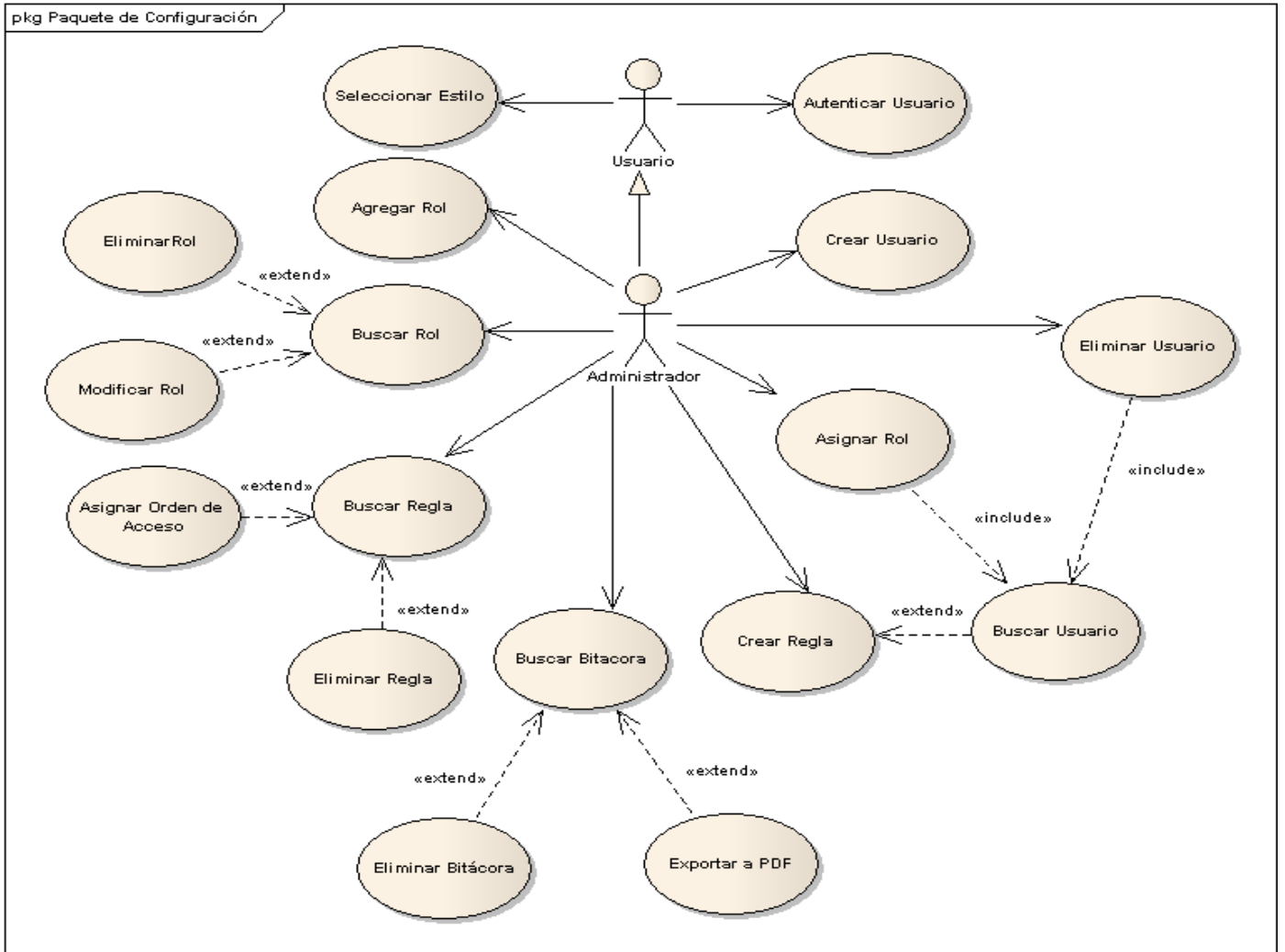


Figura 3 Diagrama de Casos de Uso del Sistema de los procesos de Configuración del Sistema.



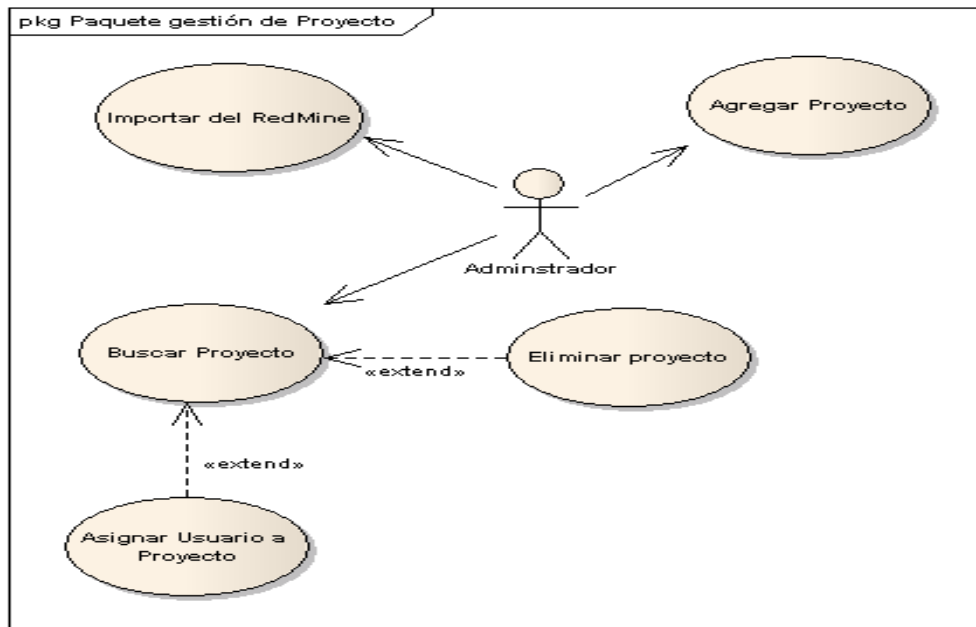


Figura 4 Diagrama de Casos de Uso del Sistema de los procesos de Gestión de Proyecto.

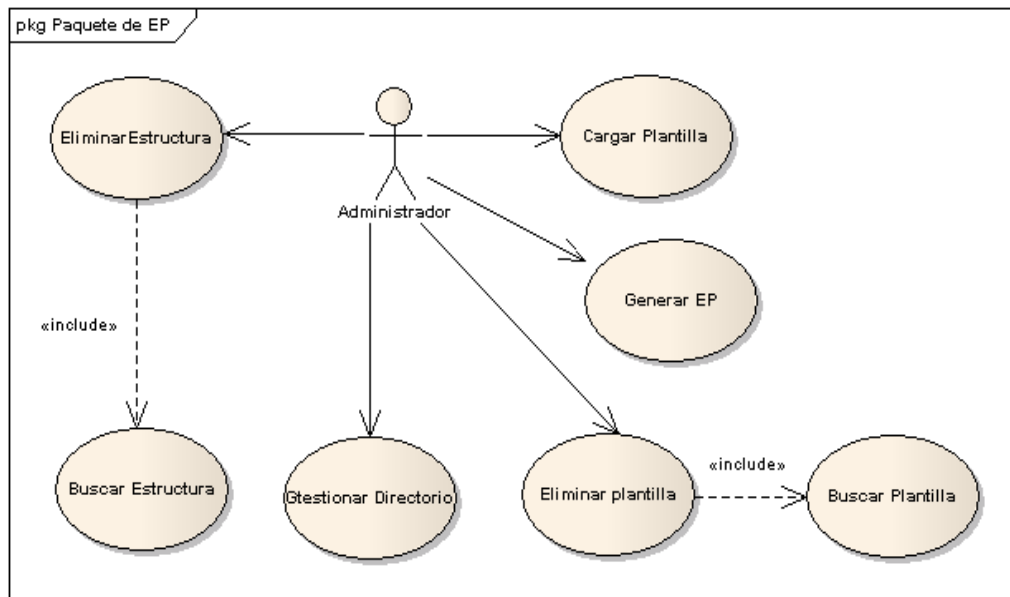


Figura 5 Diagrama de Casos de Uso del Sistema de los procesos de Gestión de los EP.

## 2.6 Descripción de los Casos de Uso del Sistema.

### 2.6.1 Descripción Textual del Caso de Uso Buscar Bitácora.

Caso de Uso:	<b>Buscar Bitácora</b>	
Actores:	Administrador	
Propósito	Este caso de uso se realiza con el objetivo de buscar una bitácora deseada.	
Resumen:	El caso de uso se inicia cuando el administrador busca una acción realizada por determinado usuario.	
Precondiciones:	El administrador debe estar autenticado	
Referencias	<b>RF26</b>	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
1-El actor selecciona la opción Buscar Bitácora	2-El sistema muestra la interfaz 29 de alasGIEP. Y permite: <ul style="list-style-type: none"> <li>• Buscar</li> <li>• Cancelar ver flujo alterno 1</li> </ul>	
3- El actor selecciona el criterio por el cual desea realizar la búsqueda y selecciona la opción Buscar	4-El sistema muestra los resultados de la búsqueda. Y permite: <ul style="list-style-type: none"> <li>• Eliminar ver alternativa 2</li> <li>• Exportar ver alternativa 3</li> </ul>	
Prototipo de Interfaz		
Interfaz 31		

**Buscar bitácora**

**Criterios de búsqueda** >>

Usuario:  Fecha:  

---

**Listado de bitácoras** 

Usuario	Fecha	Hora	Descripción	
agalan	20/06/2011	13:54:28	Ha modificado los roles del usuario agalan	
agalan	16/06/2011	12:43:18	Ha generado el Ep del proyecto alasHIS con la estructura nue	
agalan	14/06/2011	12:38:08	Ha modificado la estructura test	
agalan	16/06/2011	7:58:18	Ha subido la plantilla oricha	
agalan	11/06/2011	17:32:00	Ha modificado las reglas de acceso del directorio /Tesis/Fun	

▶ ◀

**Flujos Alternos**

**Flujo Alterno 1 “Cancelar Acción”.**

Acción del Actor	Respuesta del Sistema
2.1- El actor selecciona la opción Cancelar	2.2- El sistema muestra la interfaz anterior.

**Flujo Alterno 2 “Mostrar Mensaje”.**

Acción del Actor	Respuesta del Sistema
	4.1-El sistema muestra un mensaje notificando que “No existe información a mostrar”.

**Flujo Alterno 3 “Eliminar”.**

Acción del Actor	Respuesta del Sistema
4.2-El actor selecciona la opción Eliminar	4.3-El sistema ejecuta el Caso de uso “Eliminar Bitácora”

**Flujo Alterno 4 “Exportar”.**

Acción del Actor	Respuesta del Sistema

4.4-El actor selecciona la opción Exportar	4.5-El sistema ejecuta el Caso de uso “Exportar a PDF”.
--	---

Tabla 4 Descripción Textual del Caso de Uso Buscar Bitácora.

### 2.6.2 Descripción Textual del Caso de Uso Eliminar Bitácora.

Caso de Uso:	<b>Eliminar Bitácora</b>	
Actores:	Administrador	
Propósito	Este caso de uso se ejecuta con el objetivo de eliminar una bitácora deseada.	
Resumen:	El caso de uso se inicia cuando el administrador selecciona eliminar una acción realizada por determinado usuario.	
Precondiciones:	El administrador debe estar autenticado	
Referencias	<b>RF27</b>	
Flujo Normal de Eventos		
	Acción del Actor	Respuesta del Sistema
	1-El actor selecciona la opción eliminar Bitácora	2-El sistema muestra un mensaje de confirmación y permite: <ul style="list-style-type: none"> <li>• Aceptar</li> <li>• Cancelar ver flujo alterno 1</li> </ul>
	3- El actor acepta eliminar la bitácora seleccionada.	4-El sistema guarda los cambios efectuados.
Prototipo de Interfaz		
Interfaz 32		

Acción del Actor		Respuesta del Sistema
2.1- El actor selecciona la opción Cancelar		2.2- El sistema muestra la interfaz anterior.

Tabla 5 Descripción Textual del Caso de Uso Eliminar Bitácora.

### 2.6.3 Descripción Textual del Caso de Uso Exportar a PDF.

Caso de Uso:	<b>Exportar a PDF</b>
Actores:	Administrador
Propósito	Este caso de uso se realiza con el objetivo de generar un reporte de una bitácora deseada.
Resumen:	El caso de uso se inicia cuando el administrador elige exportar a formato PDF una bitácora según el criterio seleccionado.
Precondiciones:	El administrador debe estar autenticado
Referencias	<b>RF28</b>
Flujo Normal de Eventos	


Acción del Actor	Respuesta del Sistema																																																																
1-El actor selecciona la opción Exportar.	2-El sistema muestra un documento PDF con los datos de la bitácora como se muestra en la interfaz 30 de alasGIEP.																																																																
<p>Prototipo de Interfaz</p> <p>Interfaz 33</p>  <p><b>Reporte de Bitácoras</b></p> <table border="1" data-bbox="501 758 1216 1167"> <thead> <tr> <th><i>Usuario</i></th> <th><i>Fecha</i></th> <th><i>Hora</i></th> <th><i>Descripción</i></th> </tr> </thead> <tbody> <tr><td>agalan</td><td>23/06/2011</td><td>9:48:08</td><td>Ha modificado la estructura nueva</td></tr> <tr><td>agalan</td><td>16/06/2011</td><td>15:25:11</td><td>Ha modificado los roles del usuario jmarin</td></tr> <tr><td>agalan</td><td>22/06/2011</td><td>14:09:08</td><td>Ha importado del redmine</td></tr> <tr><td>agalan</td><td>16/06/2011</td><td>16:33:42</td><td>Ha modificado la información del documento Documento visión del EP alasHIS</td></tr> <tr><td>agalan</td><td>21/06/2011</td><td>8:32:37</td><td>Ha modificado el rol analista del sistema por analista</td></tr> <tr><td>agalan</td><td>22/06/2011</td><td>13:23:44</td><td>Ha importado del redmine</td></tr> <tr><td>agalan</td><td>23/06/2011</td><td>9:50:10</td><td>Ha generado el Ep del proyecto GPI con la estructura nueva</td></tr> <tr><td>agalan</td><td>20/06/2011</td><td>16:02:21</td><td>Ha importado del redmine</td></tr> <tr><td>agalan</td><td>16/06/2011</td><td>12:43:07</td><td>Ha generado el Ep del proyecto alasHIS con la estructura test</td></tr> <tr><td>agalan</td><td>22/06/2011</td><td>13:57:55</td><td>Ha importado del redmine</td></tr> <tr><td>agalan</td><td>16/06/2011</td><td>15:41:13</td><td>Ha creado la estructura nueva</td></tr> <tr><td>agalan</td><td>14/06/2011</td><td>22:35:10</td><td>Ha modificado las reglas de acceso del directorio /Tesis/Funcionalidades</td></tr> <tr><td>agalan</td><td>14/06/2011</td><td>10:07:13</td><td>Ha modificado las reglas de acceso del directorio /Tesis/Funcionalidades/Generador de Formularios</td></tr> <tr><td>agalan</td><td>15/06/2011</td><td>8:45:59</td><td>Ha modificado las reglas de acceso del directorio /Tesis/Funcionalidades</td></tr> <tr><td>agalan</td><td>14/06/2011</td><td>12:38:14</td><td>Ha modificado la estructura test</td></tr> </tbody> </table>		<i>Usuario</i>	<i>Fecha</i>	<i>Hora</i>	<i>Descripción</i>	agalan	23/06/2011	9:48:08	Ha modificado la estructura nueva	agalan	16/06/2011	15:25:11	Ha modificado los roles del usuario jmarin	agalan	22/06/2011	14:09:08	Ha importado del redmine	agalan	16/06/2011	16:33:42	Ha modificado la información del documento Documento visión del EP alasHIS	agalan	21/06/2011	8:32:37	Ha modificado el rol analista del sistema por analista	agalan	22/06/2011	13:23:44	Ha importado del redmine	agalan	23/06/2011	9:50:10	Ha generado el Ep del proyecto GPI con la estructura nueva	agalan	20/06/2011	16:02:21	Ha importado del redmine	agalan	16/06/2011	12:43:07	Ha generado el Ep del proyecto alasHIS con la estructura test	agalan	22/06/2011	13:57:55	Ha importado del redmine	agalan	16/06/2011	15:41:13	Ha creado la estructura nueva	agalan	14/06/2011	22:35:10	Ha modificado las reglas de acceso del directorio /Tesis/Funcionalidades	agalan	14/06/2011	10:07:13	Ha modificado las reglas de acceso del directorio /Tesis/Funcionalidades/Generador de Formularios	agalan	15/06/2011	8:45:59	Ha modificado las reglas de acceso del directorio /Tesis/Funcionalidades	agalan	14/06/2011	12:38:14	Ha modificado la estructura test
<i>Usuario</i>	<i>Fecha</i>	<i>Hora</i>	<i>Descripción</i>																																																														
agalan	23/06/2011	9:48:08	Ha modificado la estructura nueva																																																														
agalan	16/06/2011	15:25:11	Ha modificado los roles del usuario jmarin																																																														
agalan	22/06/2011	14:09:08	Ha importado del redmine																																																														
agalan	16/06/2011	16:33:42	Ha modificado la información del documento Documento visión del EP alasHIS																																																														
agalan	21/06/2011	8:32:37	Ha modificado el rol analista del sistema por analista																																																														
agalan	22/06/2011	13:23:44	Ha importado del redmine																																																														
agalan	23/06/2011	9:50:10	Ha generado el Ep del proyecto GPI con la estructura nueva																																																														
agalan	20/06/2011	16:02:21	Ha importado del redmine																																																														
agalan	16/06/2011	12:43:07	Ha generado el Ep del proyecto alasHIS con la estructura test																																																														
agalan	22/06/2011	13:57:55	Ha importado del redmine																																																														
agalan	16/06/2011	15:41:13	Ha creado la estructura nueva																																																														
agalan	14/06/2011	22:35:10	Ha modificado las reglas de acceso del directorio /Tesis/Funcionalidades																																																														
agalan	14/06/2011	10:07:13	Ha modificado las reglas de acceso del directorio /Tesis/Funcionalidades/Generador de Formularios																																																														
agalan	15/06/2011	8:45:59	Ha modificado las reglas de acceso del directorio /Tesis/Funcionalidades																																																														
agalan	14/06/2011	12:38:14	Ha modificado la estructura test																																																														

Tabla 6 Descripción Textual del Caso de Uso Exportar a PDF.

Como resultado de la investigación realizada se definió un modelo de dominio que permitió comprender los conceptos asociados al entorno del problema. Se identificaron las funcionalidades a desarrollar, los casos de uso que conformarán el sistema y las descripciones textuales de cada uno de ellos.

## ***CAPÍTULO 3: Diseño del sistema.***

En este capítulo se presentan los principales artefactos para lograr la construcción del sistema como los diagramas de clases del diseño y los diagramas de secuencia. Para un mejor entendimiento se explican además los patrones de diseño que se emplean.

### **3.1 Fundamentación del uso de patrones de Diseño.**

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se han demostrado que funcionan. No son fáciles de entender, pero una vez entendido su funcionamiento, los diseños son más flexibles, modulares y reutilizables. **(Gracia, 2005)**

#### **3.1.1 Patrones Para Asignar Responsabilidades (GRASP).**

Los patrones GRASP<sup>4</sup> describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. **(Gutierrez, 2007)**

En el sistema desarrollado se aplican estos patrones, con el objetivo de distribuir responsabilidades en las clases, y establecer sus relaciones, tratando de que no existan sobrecargas de funcionalidades ni numerosa dependencia entre ellas.

#### **➤ Patrón Experto**

**Problema:** ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos? **(Gutierrez, 2007)**

Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Si estas se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones. **(Gutierrez, 2007)**

---

<sup>4</sup> Patrones Generales de Software para Asignación de Responsabilidades (General Responsibility and Assignment Software Patterns).

**Solución:** Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. **(Gutierrez, 2007)**

**Aplicación:** Este patrón es muy utilizado por GIEP, ya que el mismo define clases como Bitácora\_Negocio, la cual posibilita la gestión de toda la información correspondiente a una bitácora determinada.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using AccesoDatos;
6 using System.Data.Linq;
7
8 namespace Negocio
9 {
10 public class Bitacora_Negocio
11 {
12     Bitacora_Datos bit = new Bitacora_Datos();
13
14     public bool Insertar(Bitacora_Entidad b)...
15
16
17
18
19
20     public List<Bitacora_Entidad> GetAllBitacora()...
21
22
23
24
25
26
27
28
29
30
31
32
33
34     public List<Bitacora_Entidad> GetBitacoraUser(string user)...
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49     public List<Bitacora_Entidad> GetBitacoraFecha(string fecha)...
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64     public List<Bitacora_Entidad> GetBitacoraUserFecha(string user, string fecha)...
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79     public bool Eliminar(Bitacora_Entidad b)...
80
81
82
83
84 }
85
86 }
```

➤ **Patrón Creador.**

**Problema:** ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?  
**(Gutierrez, 2007)**

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el



encapsulamiento y la reutilización. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. **(Gutierrez, 2007)**

**Solución:** Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A o
- B utiliza especialmente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A). B es un creador de los objetos A.

**(Gutierrez, 2007)**

**Aplicación:** En la clase Rol\_Negocio se definen y ejecutan acciones, donde se crean los objetos de las clases que representa la clase Rol\_Datos, evidenciando que la clase Rol\_Negocio es creadora de dicha entidad.

```
7 namespace Negocio
8 {
9     public class Rol_Negocio
10    {
11        Rol_Datos rol = new Rol_Datos();
12    }
13
14    public bool Insertar_Rol(Rol_Entidad r)...
28
29    public List<Rol_Entidad> GetAllRoles()...
40
41    public string[] GetRolesForUser(User_Entidad u)...
46
47
48    public string[] GetUsersInRole(Rol_Entidad r)...
52
53    public bool RemoveUsersFromRole(string[] users, Rol_Entidad r)...
57
58    public bool RemoveUserFromRoles(User_Entidad u, string[] roles)...
62
63    public bool AddUserToRoles(User_Entidad u, string[] roles)...
67
68    public bool AddUserToRole(User_Entidad u, Rol_Entidad r)...
72
73    public bool IsUserInRole(User_Entidad u, Rol_Entidad r)...
77
78    public bool DeleteRole(Rol_Entidad r)...
82
83    public bool RoleExists(Rol_Entidad r)...
87
88    public bool RemoveUserFromRoles(Rol_Entidad r)...
92
93 }
94
```

➤ **Patrón Alta Cohesión.**

**Problema:** ¿Cómo mantener la complejidad dentro de límites manejables? (Gutierrez, 2007)

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo. Esto presenta los siguientes problemas:

- Son difíciles de comprender.
- Difíciles de reutilizar.
- Difíciles de conservar.

- Las afectan constantemente los cambios. **(Gutierrez, 2007)**

**Solución:** Asignar una responsabilidad de modo que la cohesión siga siendo alta. **(Gutierrez, 2007)**

**Aplicación:** GIEP permite asignar responsabilidades con una alta cohesión, ejemplo de ello es la clase Bitácora\_Negocio que tiene la responsabilidad de definir acciones sobre una bitácora específica, así como instanciar otros objetos y acceder a las propiedades.

```
7 |
8 | namespace Negocio
9 | {
10 |     public class Bitacora_Negocio
11 |     {
12 |         Bitacora_Datos bit = new Bitacora_Datos();
13 |         public bool Insertar(Bitacora_Entidad b)
14 |         {
15 |
16 |             return bit.Insertar(b.Fecha, b.Descripcion, b.Username, b.Hora);
17 |         }
18 |
19 |         public List<Bitacora_Entidad> GetAllBitacora()
20 |         {
21 |             List<Bitacora_Entidad> ret = new List<Bitacora_Entidad>();
22 |             foreach (GetAllBitacoraResult obj in bit.GetAllBitacora())
23 |             {
24 |
25 |                 Bitacora_Entidad bi = new Bitacora_Entidad(obj.fecha, obj.hora, obj.descripcion, obj.UserName);
26 |
27 |                 ret.Add(bi);
28 |             }
29 |
30 |             return ret;
31 |         }
32 |     }

```

➤ **Patrón Controlador.**

**Problema:** ¿Quién debería encargarse de atender un evento del sistema? **(Gutierrez, 2007)**

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema. Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. **(Gutierrez, 2007)**

**Solución:** Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- el “sistema” global (controlador de fachada).
- la empresa u organización global (controlador de fachada).
- algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados “Manejador<NombreCasodeUso>” (controlador de casos de uso).

Un defecto frecuente al diseñar controladores consiste en asignarles demasiada responsabilidad. Normalmente un controlador debería delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad. **(Gutierrez, 2007)**

**Aplicación:** Todas las peticiones del sistema son manejadas por clases controladoras, las cuales se encargan de manejar todas las acciones a realizar.

```
8 using iTextSharp.text.pdf;
9 using iTextSharp.text;
10 using System.IO;
11 using System.Diagnostics;
12
13 public partial class Funcionalidades_Buscar_Bitacora_Buscar_Bitacora : BasePage
14 {
15     Bitacora_Negocio bitacora = new Bitacora_Negocio();
16
17
18
19
20     protected void btnFind_Click(object sender, EventArgs e)...
76     protected void Eliminar_Click(object sender, EventArgs e)...
127     protected void PDF_Click(object sender, EventArgs e)...
168
169
170
171
172 }
173
```

### 3.2 Diagramas de Clases.

Un diagrama de clases del diseño Web permite describir gráficamente un conjunto de clases, así como las relaciones entre ellas, logrando de esta forma una muestra del sistema importante para la

implementación. A continuación en la figura 4 se muestra el diagrama de clases del diseño del Caso de Uso Gestionar Rol. Los diagramas de diseño correspondientes a los demás Casos de Uso pueden ser consultados en el [Anexo 2](#).

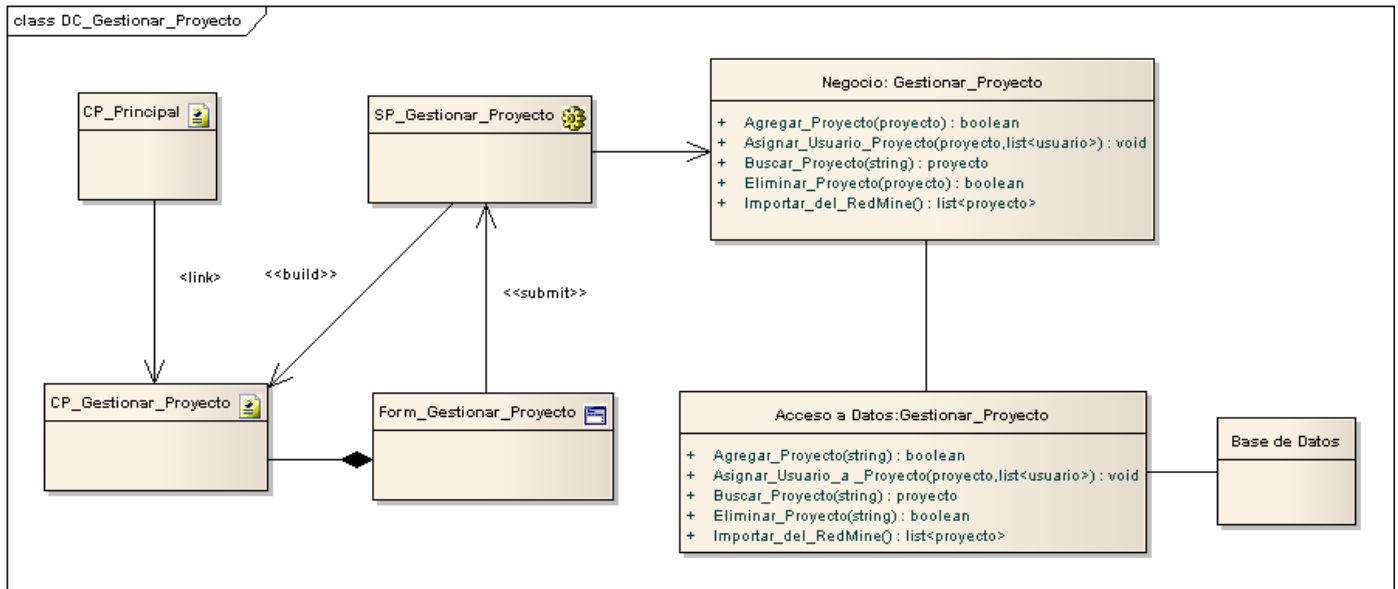


Figura 6 Diagrama de clase del diseño del Caso de Uso Gestionar Proyecto.

### 3.3 Diagramas de secuencia.

Un Diagrama de secuencia consiste en resaltar el ordenamiento temporal de los mensajes, presenta un conjunto de objetos con los mensajes que reciben y que envían entre ellos. A continuación en la figura 5 se muestra el diagrama de secuencia del Caso de Uso Gestionar Rol. Los diagramas de secuencia correspondientes a los demás Casos de Uso pueden ser consultados en el [Anexo 3](#).

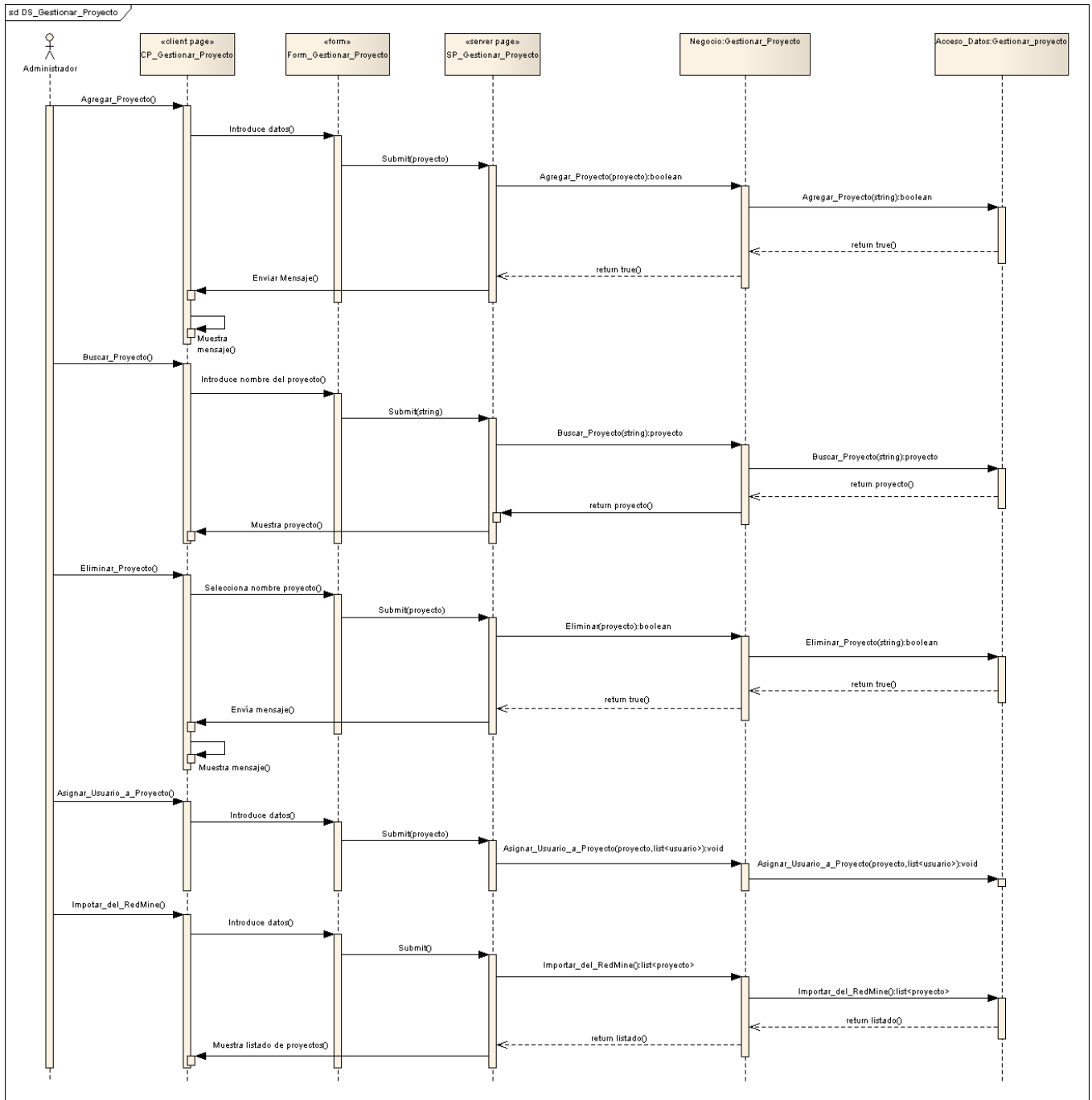


Figura 7 Diagrama de Secuencia del Caso de Uso Gestionar Proyecto.

### 3.4 Descripción de las clases del diseño.

A continuación en la tabla 32 y 33 se presenta la descripción de la clase de diseño que tiene relación con la funcionalidad presentada.

<b>NOMBRE:</b>	Agregar Proyecto
<b>Tipo de Clase:</b>	Interfaz
<b>Atributo</b>	Tipo
Nombre	Campo de texto
Proyecto Principal	Selector
Usuarios	ListBox
Miembros	ListBox
Aceptar	Botón
Cancelar	Botón
<b>Descripción:</b>	Esta clase muestra una interfaz para agregar un proyecto, mostrando un mensaje indicando el resultado de la acción.

Tabla 7 Descripción de la clase Agregar Proyecto.

<b>NOMBRE:</b>	Buscar Proyecto
<b>Tipo de Clase:</b>	Interfaz
<b>Atributo</b>	Tipo
Nombre del proyecto	Campo de texto
Buscar	Botón
Cancelar	Botón
Listado de Proyectos	Tabla
Seleccionar	Botón
Eliminar	Botón
<b>Descripción:</b>	Esta clase muestra una interfaz para buscar, seleccionar o eliminar un proyecto, mostrando el resultado de la búsqueda.

Tabla 8 Descripción de la clase Buscar Proyecto

### 3.5 Descripción de la Arquitectura.

La programación por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

La **capa de presentación** es la encargada de interactuar con el usuario, solicitándole y/o mostrándole información de ser necesaria. Esta capa se comunica únicamente con la capa de negocio.

La **capa de negocio** es la encargada de recibir y procesar las peticiones del usuario y enviar las respuestas tras el proceso. 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 almacenar o recuperar datos de él.

La **capa de datos** es la encargada de acceder a la base de datos y obtener la información. Esta capa recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio y le proporciona la información.

A continuación en la figura 6 se muestra la arquitectura empleada en el sistema desarrollado.

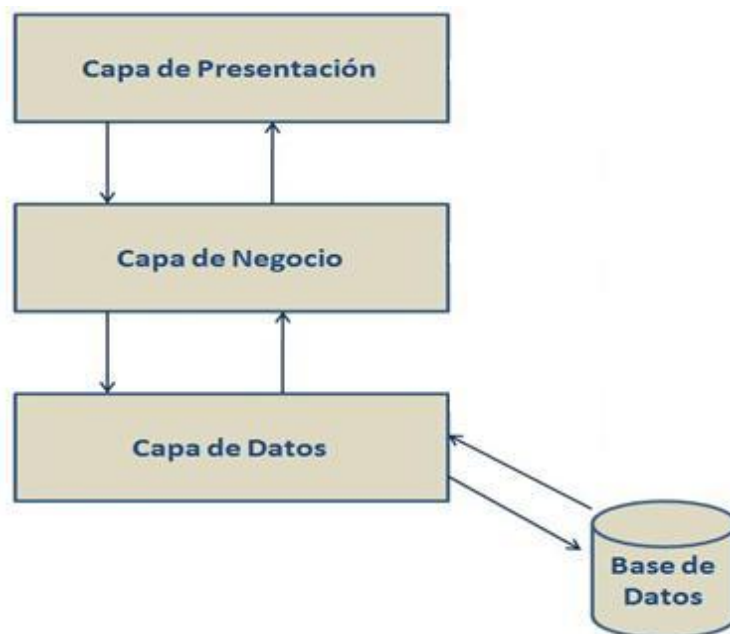


Figura 8 Arquitectura en capas.

En el presente capítulo se logró presentar la descripción del diseño a través de los diagramas de clases del diseño y de los diagrama de secuencia. Para la realización de estos diagramas se tuvo en cuenta la utilización de estereotipos web, que describen la relación entre las páginas. Se definieron los principios de diseño, abordando la descripción de la arquitectura definida, patrones y estándares a utilizar.



## ***CAPÍTULO 4: Implementación y prueba.***

En este capítulo se presentan los principales artefactos para lograr la construcción del sistema como el modelo de despliegue y el modelo de componente.

### **4.1 Modelo de Despliegue.**

Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Los nodos no son más que elementos físicos que existen en tiempo de ejecución y representan un recurso computacional que generalmente tienen algo de memoria y capacidad de procesamiento. El Diagrama de Despliegue servirá para modelar la topología de hardware sobre la cual se ejecutará el sistema. **(Morales, et al., 2010)**

El diagrama de despliegue correspondiente al sistema desarrollado, como se muestra en la figura 7, consta con una computadora cliente, en la cual los usuarios pueden visualizar e interactuar con la aplicación que se encuentra en el servidor de aplicación. Este servidor es el encargado de responder las peticiones de las computadoras clientes y a su vez está conectado al servidor de base de datos, el cual es el encargado de administrar los datos necesarios para el correcto funcionamiento de la aplicación alasGIEP.

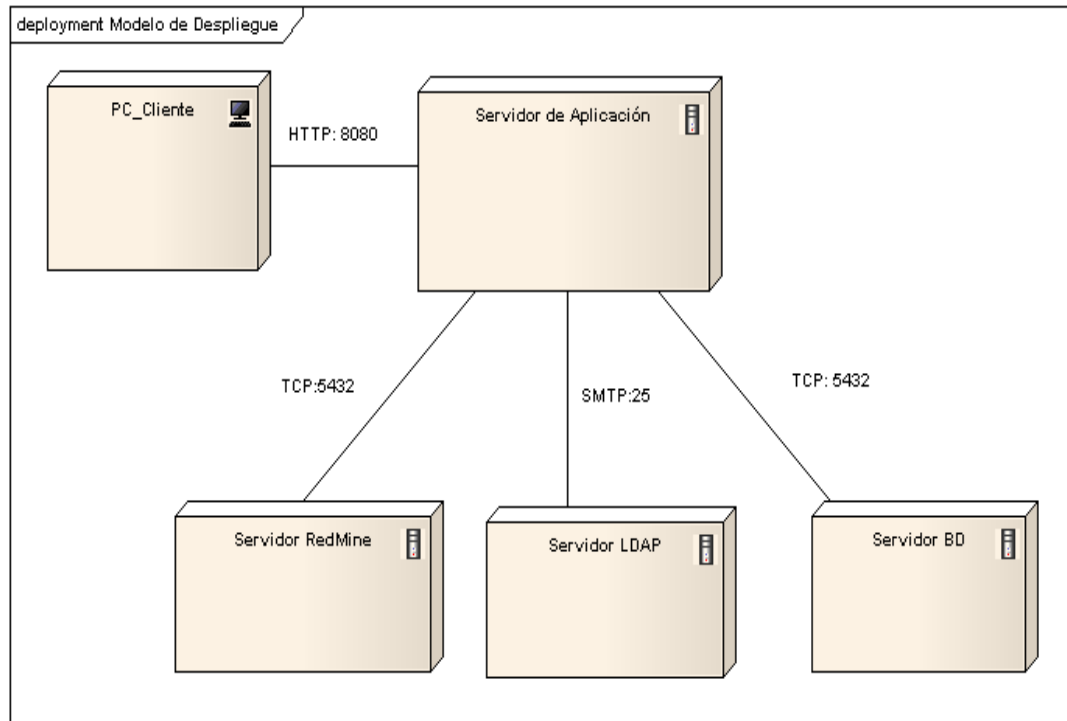


Figura 9 Modelo de Despliegue.

## 4.2 Modelo de Implementación.

Un modelo de implementación consiste en una visión general de lo que tiene que ser implementado, y un apartado para cada iteración con los componentes y subsistemas a implementar durante esa iteración, así como de los resultados software que se han de obtener y el testeo que se ha de realizar sobre ellos. **(Calzado Toledano, et al., 2010)**

### 4.2.1 Diagrama de Componentes.

A continuación se muestra en la figura 8 la vista general del diagrama de componentes correspondiente al sistema desarrollado. Para un mejor entendimiento de dicho diagrama se expone además, en las figuras 10, 11, 12 y 13 una vista detalla de los paquetes, así como los componentes que contiene cada paquete en específico.

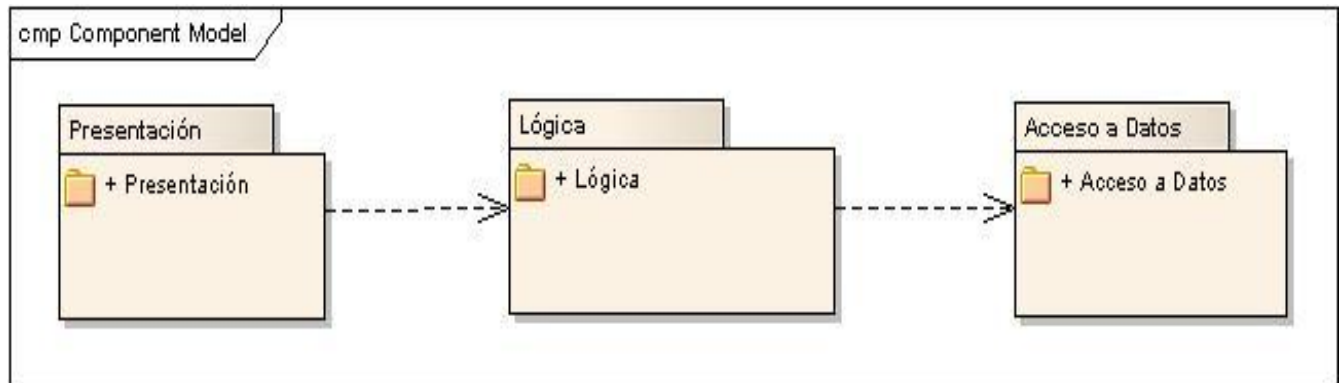


Figura 10 Vista General Del Diagrama de Componentes.

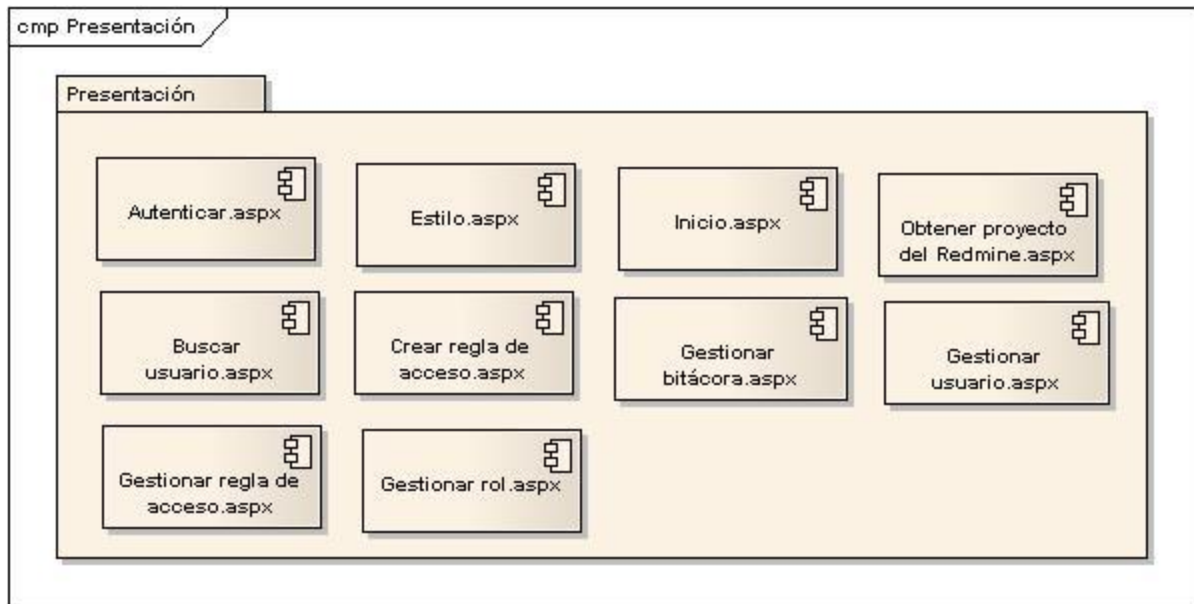


Figura 11 Vista Detallada Del Paquete Presentación.

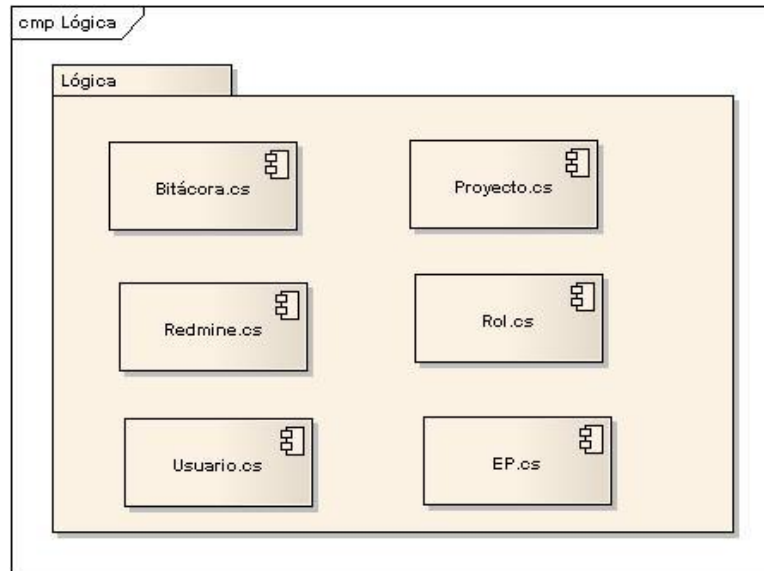


Figura 12 Vista Detallada Del Paquete Lógica.

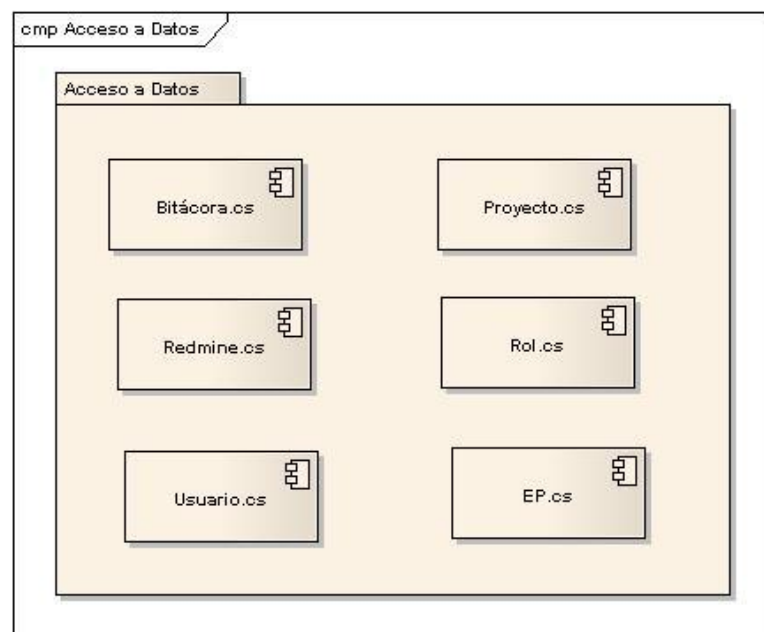


Figura 13 Vista Detallada Del Paquete Acceso a Datos.

### 4.3 Validación de la solución propuesta.

Las pruebas de un software están constituidas por un conjunto de herramientas, técnicas y métodos que tributan a la excelencia del desempeño de un programa. Técnicamente consisten en inspeccionar manualmente el código o hacer pruebas de ensayos (ejecutar el software y ver los resultados) y el objetivo principal es encontrar errores o defectos en el software. **(Calzado Toledano, et al., 2010)**

A continuación en la tabla 13 se muestra el diseño de caso de prueba de caja negra del Caso de Uso Gestionar Rol. Los diseños de casos de pruebas correspondientes a los demás Casos de Uso pueden ser consultados en el [Anexo4](#). Para ello solo es necesario conocer la interfaz y tratar de probar cada uno de los elementos que componen a la misma.

#### 4.3.1 Diseño de prueba del Caso de Uso Gestionar Proyecto.

**Descripción General:** El caso de uso se inicia cuando el usuario desea crear, buscar o eliminar un rol determinado y termina cuando el sistema realiza la operación deseada por el usuario.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad.	Flujo Central
SC1:Agregar proyecto	EC1.1: Agregar proyecto	Se indica crear un nuevo proyecto y se introduce el nombre del mismo	<ul style="list-style-type: none"><li>- El administrador introduce el nombre del proyecto a crear.</li><li>- El sistema verifica que no exista un proyecto con el mismo nombre.</li><li>- El sistema registra y visualiza el nuevo proyecto.</li><li>- El sistema valida que no existan datos incorrectos.</li><li>- El sistema muestra la interfaz principal.</li></ul>

.	EC 1.2: Agregar proyecto con campos vacíos	En este escenario se muestra el campo señalado, en caso de que no se completen los campos requeridos.	- Si el campo nombre quedó vacío se señala como campo requerido.
	EC 1.3: Crear proyecto que ya existe	En este escenario el administrador indica crear un nuevo proyecto e introduce el nombre del mismo, en caso de que el nombre se duplique.	- Si ya existe un proyecto con ese nombre el sistema emite un mensaje: "El proyecto " X " ya existe "
SC2: Eliminar proyecto	EC2.1: Eliminar proyecto	Se elimina el proyecto seleccionado.	<ul style="list-style-type: none"> <li>- El administrador busca el proyecto que desea eliminar.</li> <li>- Se selecciona la opción eliminar proyecto.</li> <li>- El sistema muestra un mensaje de confirmación.</li> <li>- El usuario confirma que desea eliminar el proyecto</li> <li>- El sistema elimina el proyecto seleccionado y actualiza los proyectos.</li> </ul>

SC3: Asignar Usuario a proyecto	EC3.1: Asignar Usuario a proyecto	En este escenario el administrador selecciona el proyecto a modificar e introduce los datos.	<ul style="list-style-type: none"> <li>- El administrador busca y selecciona el proyecto a modificar.</li> <li>- El administrador agrega o elimina miembros al proyecto seleccionado.</li> <li>- El sistema guarda los cambios efectuados.</li> </ul>
SC4: Buscar proyecto	EC4.1: Buscar proyecto	Se introduce el nombre del proyecto a buscar.	<ul style="list-style-type: none"> <li>- El administrador introduce el nombre del proyecto a buscar.</li> <li>- El sistema muestra el resultado final de la búsqueda.</li> <li>- El sistema posibilita seleccionar o eliminar dicho proyecto.</li> </ul>

Tabla 9 Diseño de prueba del Caso de Uso Gestionar Proyecto.

Como resultado de las pruebas efectuadas al sistema, se ejecutaron dos iteraciones, en una primera iteración se realizaron seis recomendaciones y se encontraron cuatro no conformidades no significativas, permitiendo darles solución en una segunda iteración, por lo se obtuvieron resultados satisfactorios.

En este capítulo se desarrollaron los principales artefactos para llevar a cabo el proceso de implementación, necesarios para la construcción del sistema. Mediante la realización del Modelo de Despliegue, se describe como está distribuida física y lógicamente la arquitectura del sistema y sus conexiones. Además la realización del Modelo de Implementación, permitió detallar los componentes creados para el desarrollo de la aplicación y la relación entre ellos.

## **CONCLUSIONES**

- Se identificaron las características de interés para el desarrollo del sistema a partir del análisis de las funcionalidades asociadas a la configuración de varios sistemas de gestión de información.
- Se determinó la arquitectura adecuada para el desarrollo del sistema.
- Se desarrolló el módulo que permitirá la configuración de la variante web de la herramienta GIEP.
- Las pruebas realizadas arrojaron resultados satisfactorios lo cual avala la calidad del módulo desarrollado.



## **RECOMENDACIONES**

Se recomienda:

- Actualizar el componente utilizado en el sistema Telerik.Web.UI.dll, para eliminar los errores que puedan lograr el mal funcionamiento de la aplicación.
- Agregar al módulo de configuración un algoritmo que permita descryptar las contraseñas del Redmine, de manera que puedan ser utilizados sus usuarios para interactuar en la aplicación desarrollada.
- Añadir un servicio con las funcionalidades que permitan la integración de la aplicación con otros sistemas.
- Se recomienda que los proyectos existentes cambien de estado y no sean eliminados de la BD.

## REFERENCIA BIBLIOGRÁFICA

- Calzado Toledano, Yannia Dalida y Arias Dieguez, Luis Carlos. 2010. **Desarrollo de un Sistema de Información Geográfica sobre tecnología libre en la Web**. Univesidad de las Ciencias Informáticas. La Habana : s.n., 2010. Tesis.
- Ceslcam. 2010. **Centro de excelencia de software libre de Castilla La Mancha**. [En línea] 2010. <http://www.ceslcam.com/noticias-del-centro/analisis-de-aplicacion-Redmine.html>.
- Computer Associates Int. (CA). 2005. **MASTER DAP**. [En línea] 2005. <http://www.master-direccion-proyectos.com/programasMaster/clarity.pdf>.
- de la Fuente, Toni. 2008. **Conferencia Internacional de Software Libre**. [En línea] Octubre de 2008. [http://www.opensourceworldconference.com/papers/Dia21/Sala%201/Tarde/de%20la%20Fuente\\_17.pdf](http://www.opensourceworldconference.com/papers/Dia21/Sala%201/Tarde/de%20la%20Fuente_17.pdf).
- Expertos en Servicios de Consultoría Exes, S.L. 2000. **exes**. [En línea] 2000. [http://www.exes.es/ManJava/index.asp?Pg=java\\_inicial\\_4\\_1.html](http://www.exes.es/ManJava/index.asp?Pg=java_inicial_4_1.html).
- GIDOC INTEGRAL, S.L. 2008. **Gidoc Integral. Glosario de términos**. [En línea] 2008. <http://www.gidocintegral.com/definiciones.html>.
- González, Carlos D. 2008. **Usabilidad Web**. [En línea] 2008. <http://www.usabilidadweb.com.ar/cpp.php>.
- Gracia, Joaquín. 2005. **IngenieroSoftware**. [En línea] 27 de Mayo de 2005. <http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php>.
- Guglielmetti, Marcos, y otros. 2005. **MASTERMAGAZINE. Definición de Configurar**. [En línea] 11 de Febrero de 2005. <http://www.mastermagazine.info/termino/4404.php>.
- Gutierrez, Jorge A. Saavedra. 2007. **El Mundo Informático**. [En línea] 8 de Mayo de 2007. <http://jorgesaavedra.wordpress.com/2007/05/08/patrones-grasp-patrones-de-software-para-la-asignacion-general-de-responsabilidadparte-ii/>.
- Hernández Suárez, Eivys. 2008. **Administración de las Configuraciones y Definiciones de Seguridad del SIGEP**. [Tesis]. Habana, Habana, Cuba : s.n., Mayo de 2008.
- Morales Mesa, Enerys. Rodríguez Martínez, Yoandry y Vega Argota, Irina Elena. 2010. **SISTEMA PARA EL ANÁLISIS CUANTITATIVO DE LOS RIESGOS PARA LOS PROYECTOS DE**

**PRODUCCIÓN DE SOFTWARE. [Documento]. Ciudad de la Habana, La Habana, Cuba : s.n., Septiembre de 2010. pág. 18.**

- Peña, Rodrigo. 2001. **GestioPolis. [En línea] Agosto de 2001. <http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/gestioproyecto.htm>.**
- Ramirez, Elvia. 2010. ***Las Poetas Centroamericanas en el Repertorio Americano de 1919 a 1959. Patrimonio Documental Digital.* Universidad De Costa Rica. Costa Rica : s.n., 2010. Tesis de Maestría.**
- Redmine. 2006. **Redmine. [En línea] 1.1.0, 2006. <http://www.Redmine.org/>.**
- Ríos Aquino, Jaime Francisco y Cortés Morales, Jorge Eduardo. 2010. **buenas tareas. [En línea] 26 de Abril de 2010. <http://www.buenastareas.com/ensayos/Lenguajes/257692.html>.**
- Smith, Mark. 2008. **developerFusion. [En línea] 26 de Marzo de 2008. <http://www.developerfusion.com/article/7633/introducing-visual-studio-net-2008-top-10-features/>.**
- Webmaster. 2009. **my partners. *Más Conocimiento.* [En línea] 22 de Octubre de 2009. [http://www.mypartnersap.com/local2/index.php?option=com\\_content&view=article&id=13&Itemid=14](http://www.mypartnersap.com/local2/index.php?option=com_content&view=article&id=13&Itemid=14).**

## **BIBLIOGRAFÍA**

- Adapting Document. (2009). Obtenido de <http://www.adapting.com/aplicaciones/gestiondocumental/>
- Calzado Toledano, Yannia Dalida y Arias Dieguez, Luis Carlos. 2010. **Desarrollo de un Sistema de Información Geográfica sobre tecnología libre en la Web**. Univesidad de las Ciencias Informáticas. La Habana : s.n., 2010. Tesis.
- Ceslcam. 2010. **Centro de excelencia de software libre de Castilla La Mancha**. [En línea] 2010. <http://www.ceslcam.com/noticias-del-centro/analisis-de-aplicacion-Redmine.html>.
- Computer Associates Int. (CA). 2005. **MASTER DAP**. [En línea] 2005. <http://www.master-direccion-proyectos.com/programasMaster/clarity.pdf>.
- de la Fuente, Toni. 2008. **Conferencia Internacional de Software Libre**. [En línea] Octubre de 2008. [http://www.opensourceworldconference.com/papers/Dia21/Sala%201/Tarde/de%20la%20Fuente\\_17.pdf](http://www.opensourceworldconference.com/papers/Dia21/Sala%201/Tarde/de%20la%20Fuente_17.pdf).
- Expertos en Servicios de Consultoría Exes, S.L. 2000. **exes**. [En línea] 2000. [http://www.exes.es/ManJava/index.asp?Pg=java\\_inicial\\_4\\_1.html](http://www.exes.es/ManJava/index.asp?Pg=java_inicial_4_1.html).
- Gestión Documental. (2005). Obtenido de Gestión Documental
- GIDOC INTEGRAL, S.L. 2008. **Gidoc Integral. Glosario de términos**. [En línea] 2008. <http://www.gidocintegral.com/definiciones.html>.
- González, Carlos D. 2008. **Usabilidad Web**. [En línea] 2008. <http://www.usabilidadweb.com.ar/cpp.php>.
- Gracia, Joaquín. 2005. **IngenieroSoftware**. [En línea] 27 de Mayo de 2005. <http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php>.
- Guglielmetti, Marcos y otros. 2005. **MASTERMAGAZINE. Definición de Configurar**. [En línea] 11 de Febrero de 2005. <http://www.mastermagazine.info/termino/4404.php>.
- Gutierrez, Jorge A. Saavedra. 2007. **El Mundo Informático**. [En línea] 8 de Mayo de 2007. <http://jorgesaavedra.wordpress.com/2007/05/08/patrones-grasp-patrones-de-software-para-la-asignacion-general-de-responsabilidadparte-ii/>.

- Hernández Suárez, Eivys. 2008. **Administración de las Configuraciones y Definiciones de Seguridad del SIGEP. [Tesis].** Habana, Habana, Cuba : s.n., Mayo de 2008.
- Introducción a RUP. (s.f.). Obtenido de <https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc>
- Larman, C. (2007). UML y Patrones.
- Morales, Enerys Mesa, Rodríguez, Yoandry Martínez y Vega, Irina Elena Argota. 2010. **SISTEMA PARA EL ANÁLISIS CUANTITATIVO DE LOS RIESGOS PARA LOS PROYECTOS DE PRODUCCIÓN DE SOFTWARE. [Documento].** Ciudad de la Habana, La Habana, Cuba : s.n., Septiembre de 2010. pág. 18.
- Peña, Rodrigo. 2001. **GestioPolis. [En línea] Agosto de 2001.** <http://www.gestipolis.com/recursos/documentos/fulldocs/ger/gestioproyecto.htm>.
- Pressman, R. (2002). Ingeniería del Software: Un enfoque práctico.
- Ramirez, Elvia. 2010. **Las Poetas Centroamericanas en el Repertorio Americano de 1919 a 1959. Patrimonio Documental Digital.** Universidad De Costa Rica. Costa Rica : s.n., 2010. **Tesis de Maestría.**
- Redmine. 2006. **Redmine. [En línea] 1.1.0, 2006.** <http://www.Redmine.org/>.
- Ríos Aquino, Jaime Francisco y Cortés Morales, Jorge Eduardo. 2010. **buenas tareas. [En línea] 26 de Abril de 2010.** <http://www.buenastareas.com/ensayos/Lenguajes/257692.html>.
- Smith, Mark. 2008. **developerFusion. [En línea] 26 de Marzo de 2008.** <http://www.developerfusion.com/article/7633/introducing-visual-studio-net-2008-top-10-features/>.
- Webmaster. 2009. **my partners. Más Conocimiento. [En línea] 22 de Octubre de 2009.** [http://www.mypartnersap.com/local2/index.php?option=com\\_content&view=article&id=13&Itemid=14](http://www.mypartnersap.com/local2/index.php?option=com_content&view=article&id=13&Itemid=14).

## **GLOSARIO DE TÉRMINOS**

- **Application Programming Interface (API):** Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas.
- **ASP.NET:** 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.
- **Clases:** Conjunto de objetos que comparten atributos, operaciones, relaciones y semántica, las mismas representan los conceptos fundamentales del sistema.
- **Diagrama de Clases:** Representación de los conceptos de importancia en el área de la aplicación, así como de las relaciones entre estos.
- **Gestión de Proyectos:** La gestión de proyectos es el proceso por el cual se planifica, dirige y controla el desarrollo de un sistema aceptable con un costo mínimo y dentro de un período de tiempo específico.
- **Gestión Documental:** Conjunto de normas técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización, permite la recuperación de información desde ellos, determinar el tiempo que los documentos deben guardarse, eliminar los que son desechados y asegurar la conservación indefinida de los documentos más valiosos.
- **Microsoft Language Integrated Query (LINQ):** Es una tecnología integrada en .NET que proporciona la capacidad para consultar o manipular diversas fuentes de datos, independientes del proveedor utilizando de forma nativa la sintaxis de cualquier lenguaje de programación soportado por .NET.
- **Rational Unified Procces (RUP):** Metodología de desarrollo tradicional. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Es

adaptable al contexto y necesidades de cada organización cuyo fin es entregar un producto de software.

- **UML:** Lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de un desarrollo de software.