Universidad de las Ciencias Informáticas

Facultad 3

"Análisis y Diseño del Módulo Administración para el Laboratorio Virtual de Criptografía"

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

Autor: Alexis Mejías Rodríguez

Tutor: Ing. Carlos Y. Hidalgo García

Ciudad de la Habana, Cuba

Junio 2011

El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres.

Ernesto Guevara de la Serna.

Declaración de Autoría

DECLARACIÓN DE AUTORÍA

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

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

Alexis Mejías Rodríguez	Ing. Carlos Y. Hidalgo García
Firma del autor	Firma del tutor

Dedicatoria

A mi madre por ser todo en mi vida.
A mi padre por brindarme su apoyo cuando más lo necesité.
A mi hermana por confiar siempre en mí.
A mi abuela Pepa por darme la educación que tengo hoy.
A mi padrastro que ha sido como un padre para mí.
A mi abuelo Celestino que siempre ha estado para ayudarme.
A mis tíos, Adolis, Nolvis, Marfa y Yamilka por apoyarme y ayudarme.
A mis abuelos Miriam y Reyes que en paz descansen, que lo dieron todo por mí.
A mi tío Arnolis que en paz descanse, por ser una de las personas que mas me ha ayudado en mi vida.

Agradecimientos

A mis abuelos Miriam y Reyes, que en paz descansen, por su apoyo incondicional.
A mi tío Arnolis, que en paz descanse, por haberme ayudado cuando pequeño.
A mis padres por su apoyo, por sus preocupaciones y por haber confiado siempre en mí.
A mis abuelos que siempre han estado tan orgullosos de mí.
A mi hermana por todo su amor, cariño y confianza.
A mi hermanastra Dayamí por quererme como un hermano.
A mi tutor Carlos Yasmany por ayudarme mucho con mi tesis y a los demás que de una forma u otra aportaron su granito de arena.
A mis amigos Alejandro, Juan José, Pedro, Yosvany y los demás que he conocido durante todo el tiempo que llevo en la Universidad.
A mis amigas Yailyn, Juliet y Arlena, por tenerme presente y contar conmigo para todo.
A mis hermanos Alexis David, Lázaro y Miguel por estar siempre cuando los necesité en los malos y buenos momentos y por brindarme su amistad incondicional.
A mi novia, por preocuparse siempre por mí y exigirme mucho cuando perdía el tiempo innecesariamente.
A todos mis tíos, primos, quienes han dado muestra en todo momento de preocupación y entrega a mi porvenir.

Resumen

El presente trabajo de diploma está encaminado a desarrollar el Análisis y Diseño del Módulo Administración del Laboratorio Virtual de Criptografía que permitirá visualizar los fundamentos teóricos de la Criptografía, así como realizar ejercicios sobre este tema, apoyando a la asignatura de Seguridad Informática.

Con el fin de desarrollar este trabajo se realizó un estudio sobre conceptos asociados a laboratorios virtuales determinando las ventajas y desventajas que poseen, metodologías de desarrollo de software, herramientas de modelado, entre otros aspectos asociados a la Ingeniería de Software. Además se confeccionó el Modelo de Dominio, Diagrama de Casos de Uso del Sistema, Diagramas de Clases de Análisis, Diagramas de Secuencia y Diagramas de Clases del Diseño correspondientes a cada Caso de Uso. Finalmente se aplicaron métricas y pruebas dirigidas a evaluar la calidad de los requisitos y el sistema, obteniéndose resultados satisfactorios.

PALABRAS CLAVE

“Laboratorio Virtual, Criptografía, Seguridad Informática, Tecnología”

Índice de Tablas

Índice de Tablas

Índice de Tablas

.

Introducción

INTRODUCCIÓN

La evolución de la humanidad y la interacción hombre-naturaleza, ha traído consigo que la especie humana se aventure hacia la investigación de lo desconocido del universo que lo rodea, permitiendo la constante experimentación y desarrollo de ciencias, las cuales están presentes en muchos sectores sociales. Entre estas ciencias se encuentra la Informática. Nuestro país se ha incorporado al desarrollo informático mundial, trazando diversas estrategias para elevar los niveles de conocimientos de esta ciencia, así como elevar el nivel cultural de los cubanos. Algunas de estas estrategias son: llevar a cada escuela del país una computadora, la creación de los Joven Club de Computación, la creación de los Cibercafé y de una universidad de nuevo tipo llamada Universidad de las Ciencias Informáticas (UCI).

La Universidad de Ciencias Informáticas se ha convertido en la fuerza impulsora del desarrollo de software en Cuba, pues las necesidades del país de desarrollarse en esta rama de la informática, han propiciado un aumento de las responsabilidades productivas del centro. Por ser una universidad de tipo único donde el estudio-trabajo se encuentra balanceado, se ha hecho necesaria la modificación paulatina del plan de estudios con vistas a reforzar la preparación del futuro Ingeniero de Ciencias Informáticas en función de adquirir habilidades y conocimientos que le brinden un mayor ámbito de actuación una vez que los estudiantes egresen de la universidad. Todo esto se ha visto materializado en la división del plan de estudios en 2 ciclos: básico que va desde 1er año hasta el 1er semestre del 3er año y profesional desde el 2do semestre de 3ro y hasta 5to año donde el estudiante se vincula directamente a un proyecto de desarrollo y la formación se enfoca desde su labor como parte de un equipo de desarrollo con tareas concretas en función de la obtención de un determinado producto de software.

Como apoyo al proceso docente – educativo y a partir de las ventajas que ofrecen las Tecnologías de la Informática y las Comunicaciones (TIC) en la educación, se ha implementado el uso de los Entornos Virtuales de Aprendizaje (EVA) y específicamente del Moodle lo que ha permitido que todas y cada una de las asignaturas se puedan transformar en más que solamente un contenido impartido por el profesor. El uso del EVA en la UCI para el proceso de enseñanza - aprendizaje es creciente, pues permite la utilización de recursos didácticos en las distintas materias, en tal sentido es importante destacar el rol que desempeñan los laboratorios virtuales, siendo una significativa herramienta de apoyo, tanto para el profesor como para el estudiante.

Dentro de las asignaturas impartidas al 4to año de la carrera en Ciencias Informáticas, se encuentra la de Seguridad Informática la que se considera de importancia vital dentro de la especialidad fundamentalmente a partir del auge de las TIC en todas y cada una de las aristas de la vida cotidiana. Uno de los temas incluidos en esta asignatura es el referente a la criptografía, el que se torna complejo, pues cuenta con un componente teórico muy fuerte y a esto se le suma la carencia de herramientas que permitan la realización de ejercicios prácticos.

Desde que se comenzó a impartir la asignatura, se ha detectado que el tema referido a la criptografía no tiene gran profundización en clases pues forma parte de una conferencia, donde el contenido se centra en explicar algunos conceptos referentes al tema, conjuntamente con unas pocas aplicaciones y por eso se necesita integrarlo al desarrollo de las TIC como única vía para lograr integrar este tema al modelo de formación propuesto donde el estudiante sea capaz de realizar su autoestudio, lleve el protagonismo de su proceso de aprendizaje y al mismo tiempo es la vía fundamental para que se pueda poner en práctica lo recibido en el aula conjuntamente con los elementos necesarios que le permitan adquirir las habilidades necesarias que requiere este contenido.

Introducción

En función de esto se decidió desarrollar un laboratorio virtual para la asignatura de Seguridad Informática y más específicamente para el tema de criptografía, donde el estudiante tuviera la posibilidad de poner en práctica los contenidos impartidos en clases por el profesor y al mismo tiempo, contara con ejercicios y aplicaciones que apoyaran su aprendizaje. Uno de los módulos más importantes del laboratorio es sin dudas el de administración pues se encarga de configurar todo el sistema para un correcto uso del mismo por parte de estudiantes y profesores. En estos momentos ese módulo no está presente por lo que se requiere de un análisis previo donde se identifiquen las necesidades reales del cliente en este sentido con el fin de traducir esas necesidades al lenguaje de los desarrolladores para lograr una posterior implementación del módulo correspondiente.

Teniendo en cuenta todas estas condiciones mencionadas, es identificado el siguiente **problema de investigación**: ¿Cómo lograr un entendimiento entre clientes y desarrolladores respecto al tema referido de criptografía del Módulo Administración para el Laboratorio Virtual de Criptografía como apoyo a la asignatura de Seguridad Informática, que permita la posterior implementación del mismo ?. Este problema se enmarca en el **objeto de estudio**: Proceso de Desarrollo de Software.

En este sentido, la investigación tiene como **objetivo general**: Realizar el Análisis y Diseño del Módulo Administración para el Laboratorio Virtual de Criptografía para la asignatura de Seguridad Informática permitiendo así el desarrollo posterior y cuyo **campo de acción** es: Análisis y Diseño dentro del Proceso de desarrollo de software para el Módulo Administración para la modelación de un Laboratorio Virtual de Criptografía.
Es definida como **idea a defender**: Realizando el Análisis y Diseño del Módulo Administración para el Laboratorio Virtual de Criptografía de la asignatura de Seguridad Informática permitirá el desarrollo posterior de una aplicación que los estudiantes puedan utilizar para adquirir los conocimientos necesarios desde el punto de vista teórico y práctico.

Para darle cumplimiento al objetivo general planteado se definen los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Realizar los modelos de análisis y diseño.
- Validar los artefactos obtenidos.

Métodos de Investigación:

Métodos Teóricos:

- ✓ Análisis Histórico – Lógico: Para profundizar en los antecedentes de la utilización de las Tecnologías de la Información y las Comunicaciones en los procesos de enseñanza – aprendizaje y sus tendencias actuales.

- ✓ Analítico - Sintético: Se utiliza en la revisión bibliográfica, el estudio de reportes e informes sobre el estado de la infraestructura tecnológica de la UCI, la consulta de documentos rectores de la política de la UCI sobre la Informatización, la tendencia actual al aprendizaje auto gestionado y la introducción de las TIC en el proceso de enseñanza-aprendizaje.

Métodos Empíricos:

- ✓ Entrevista: Para la recopilación de información especializada o dirigida a directivos, profesores y alumnos que interactúan con las TIC en el proceso de enseñanza – aprendizaje de la asignatura de Seguridad Informática en la Facultad 3 de la Universidad de las Ciencias Informáticas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capitulo se abordarán, diferentes temas entre los que se encuentran: qué es un laboratorio virtual, los diferentes tipos que existen, los beneficios e inconvenientes que estos presentan en conjunto con su función dentro del proceso de enseñanza. Se realiza además un estudio sobre metodologías de desarrollo, herramientas CASE (por sus siglas en inglés Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) y lenguajes de modelado que se van a utilizar en la realización de este trabajo.

1.1 Laboratorios Virtuales.

Muchos han sido los autores que han definido el concepto de Laboratorio Virtual, en lo adelante se citan algunos ejemplos:

Según James P. Vary, un laboratorio virtual es un “espacio electrónico de trabajo concebido para la colaboración y la experimentación a distancia con objeto de investigar o realizar otras actividades creativas, y elaborar y difundir resultados mediante tecnologías difundidas de información y comunicación”.(Vary, 2000)

Julián Monge Nájera define un laboratorio virtual como “simulaciones de prácticas manipulativas que pueden ser hechas por el estudiante lejos de la universidad y el docente”. (Monge-Nájera, 1999).

Luego de un ánalisis a las distintas definiciones de varios autores, se define que un Laboratorio Virtual no es más que un entorno de experimentación para realizar prácticas de laboratorio, que faciliten ejercitar los contenidos de las asignaturas. Existen diversos tipos de laboratorios virtuales, según la bibliografía consultada, existen tres clasificaciones que se mencionan a continuación.

1.1.1 Laboratorios Virtuales de Software.

Los laboratorios virtuales de software, son laboratorios desarrollados como un programa de software independiente destinado a ejecutarse en la máquina del usuario, y cuyo servicio no requiere de un servidor Web. Es el caso de programas con instalación propia, que pueden estar destinados a plataformas Unix, Linux, M.S. Windows e incluso necesitar que otros componentes de software estén instalados previamente, pero que no necesitan los recursos de un servidor determinado (como bases de datos o módulos de software de servidor) para funcionar. También determinados laboratorios virtuales pensados inicialmente como aplicaciones Java accesibles a través de un servidor Web se pueden considerar de este tipo si funcionan localmente y no necesitan recursos de un servidor en concreto.

1.1.2 Laboratorios Virtuales Web.

En contraste con el anterior, este tipo de laboratorio se basa en un software que depende de los recursos de un servidor determinado. Estos recursos pueden ser determinadas bases de datos, software que requieren ejecutarse en su servidor o la exigencia de determinado hardware para ejecutarse. No son

programas que un usuario pueda descargar en su equipo para ejecutar localmente de forma independiente.

1.1.3 Laboratorios Virtuales Remotos.

Los laboratorios virtuales remotos son aquellos que permiten operar remotamente cierto equipamiento, bien sea didáctico como maquetas específicas, o industrial, además de poder ofrecer capacidades de laboratorio virtual. En general, estos requieren de equipos servidores específicos que les den acceso a las máquinas a operar de forma remota, y no pueden ofrecer su funcionalidad ejecutándose de forma local. Otro motivo que los hace dependientes de sus servidores es la habitual gestión de usuarios en el servidor. (HERÍAS, 2003)

1.2 Ventajas de los Laboratorios Virtuales.

El uso de laboratorios virtuales tiene sin duda muchos beneficios, permite al estudiante buscar información, en él se pueden relacionar fenómenos con sus consecuencias, se pueden repetir los eventos o fenómenos cuantas veces se requiera. Se incorporan las Tecnologías de la Información y Comunicación en las prácticas educativas y sociales para beneficio de los estudiantes. El aprendizaje está basado en simulaciones. Pero además de esto tiene otras ventajas más significativas como son:

- Simula situaciones que en la realidad tendría escasas posibilidades de realizarlas. (Estrada, 2007)
- Se convierten en una ayuda interactiva para el aprendizaje de contenidos difíciles de demostrar en la realidad.
- Es una herramienta de auto aprendizaje. (Herreros, 2005)
- El laboratorio virtual acerca y facilita la realización de experiencias a un mayor número de alumnos, aunque alumno y laboratorio no coincidan en el espacio.
- Permite simular fenómenos y modelos físicos, conceptos abstractos, mundos hipotéticos, controlar la escala de tiempo.
- Se fomenta un aprendizaje constructivista.
- Los estudiantes aprenden por cuenta propia fomentando la capacidad de análisis el pensamiento crítico y la utilización de tecnología informática.
- Al no verter productos químicos a la atmósfera ni a los desagües, se favorece la preservación del medio ambiente.
- Evita que productos tóxicos y perjudiciales para la salud entren en contacto con nuestros educandos.
- Permite que el profesor analice los resultados desde su ordenador y en cualquier momento del día.
- El profesor puede controlar en todo momento lo que los alumnos están realizando a través de su propio ordenador.

1.3 Desventajas de los Laboratorios Virtuales.

Acompañado a estos beneficios también se presentan algunos inconvenientes donde se corre el riesgo de que el alumno se comporte como un simple espectador. Es preciso que el estudiante realice una actividad ordenada y progresiva, conveniente a alcanzar objetivos básicos concretos, además el alumno no utiliza

elementos reales en el laboratorio virtual, lo que provoca una pérdida parcial de la visión de la realidad y los resultados son menos atractivos para los estudiantes. (Herreros, 2005)

- El laboratorio virtual no puede sustituir la experiencia práctica altamente enriquecedora del laboratorio tradicional. Ha de ser una herramienta complementaria para formar a la persona y obtener un mayor rendimiento.
- Es necesario que todos los estudiantes dispongan de un ordenador personal.
- El centro y las aulas han de disponer de conexión a internet de banda ancha.
- No tienen en cuenta las ideas de los alumnos durante su proceso de aprendizaje.
- Hay ciertos laboratorios virtuales que son difíciles de manejar por lo que nuestros estudiantes han de tener un cierto nivel de conocimiento de internet.
- Los resultados son menos llamativos para los educandos perdiendo calidad en la educación.

1.4 Estudio Realizado al Laboratorio Virtual Para Educar.

Existe una gran cantidad de laboratorios virtuales, desarrollados para diversas materias. Entre estos laboratorios virtuales se puede encontrar, laboratorios de Química, laboratorios de Física, laboratorios de Biología Molecular, laboratorios de Electrónica, laboratorios de Visual Estudio y el Laboratorio Virtual Para Educar por mencionar algunos.
Despues de realizado un estudio al Laboratorio virtual Para Educar, se puede decir que es una serie de espacios de internet con contenidos de disciplinas básicas de la enseñanza media especialmente diseñados, que se complementan con un foro virtual para el intercambio de ideas y el seguimiento de proyectos de enseñanza. Este Laboratorio Virtual fue propuesto por el Ministerio de Educación, Ciencia y Tecnología, Educ.ar S.E. y el Programa Alianza por la Educación de Microsoft de Argentina, junto a un grupo de especialistas de las distintas disciplinas.
Está destinado a los Docentes de Nivel Medio y Polimodal de todo el país que deseen participar de un ejercicio de comunidad virtual que favorezca la efectiva inclusión de las nuevas tecnologías en diferentes áreas curriculares, y construir nuevos conocimientos y propuestas en forma colaborativa. Fue diseñado con varios objetivos:

- ✓ Brindar oportunidades de especialización para los docentes de todo el país a través de internet, centradas en el análisis de las tecnologías de la información y la comunicación en su relación con las áreas de conocimiento.
- ✓ Promover los vínculos entre docentes en un entorno tecnológico, de modo tal de favorecer el reconocimiento de la influencia de la tecnología, a partir de un contacto cotidiano con las herramientas.
- ✓ Formar comunidades académicas virtuales de discusión, como una de las formas más adecuadas de construcción del conocimiento en la actualidad.
- ✓ Orientar el desarrollo de propuestas que contribuyan al mejoramiento de proyectos institucionales.

Su estructura en una primera etapa desarrolla ocho zonas que cubren las áreas disciplinares de Lengua, Literatura, Matemática, Biología, Física, Química, Historia y Geografía. Está expenso a cambio según las áreas disciplinares vayan ampliándose a la vez que se generen nuevas comunidades interdisciplinares en torno a la discusión de temas y problemas que irán publicándose mensualmente.
Cada espacio virtual está compuesta por un Núcleo Teórico que incluye cuatro dimensiones: un recorrido histórico, el estado del arte, la influencia de las nuevas tecnologías en el área de referencia y el análisis de

las tradiciones de enseñanza; y un Núcleo de Herramientas que incluye cuatro categorías: un centro de información, un archivo de documentos, un banco de materiales para la enseñanza y una serie de propuestas de enseñanza. Estos materiales fueron desarrollados por reconocidos especialistas de las diferentes disciplinas que además acompañarán a los docentes en las distintas etapas del proyecto.

Luego del estudio realizado a los diferentes tipos de laboratorios y específicamente del Laboratorio Virtual Para Educar, se llegó a la conclusión de que todos estos laboratorios están diseñados con el objetivo de interactuar con el usuario. Se convierten en una herramienta muy potente para apoyar el mejor desempeño de los estudiantes en las distintas asignaturas, ya que brindarán una gran gama de documentación y facilitarán la realización de ejercicios prácticos.

1.5 Técnicas de Captura de Requisitos.

Entrevista
La entrevista es una técnica muy utilizada en la actualidad mediante la cual se extrae la mayor parte de la información acerca del dominio del problema y las necesidades de los usuarios finales. Las entrevistas pueden clasificarse como, entrevistas abiertas o entrevistas cerradas donde en dependencia del interés del equipo de desarrollo del software se precisan algunos requisitos del sistema y terminologías. De esta forma la realización de entrevistas puede ser de vital importancia para definir los límites del sistema. (Duram, 2000)

Tormenta de ideas
La tormenta de ideas no es más que una reunión con un grupo reducido, que generalmente no deben exceder de diez personas para que cada uno de los participantes exponga sus propias ideas acerca de las funciones que el sistema debe cumplir. (Escalona, y otros, 2002)

Prototipos
El uso de prototipos es una técnica muy importante para la elicitación de los requisitos, a través de esta técnica se realiza una simulación del sistema la cual podría constituir una versión inicial. El uso de prototipos es de gran ayuda para el diseño de los prototipos de interfaz de usuario y mediante ellos se pueden descubrir requisitos incompletos o inconsistentes (Pressman, 2002).

1.6 Metodologías que se Utilizan para el Desarrollo de Software.

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería, también su propósito es establecer un contrato social entre todos los participantes en un proyecto para conseguir la solución más eficaz con los recursos disponibles.

1.6.1 Proceso Unificado de Desarrollo de Software (RUP).

La metodología Rational Unified Process, (RUP, por sus siglas en inglés) es una metodología para la ingeniería de software que va más allá del simple análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software, se caracteriza por ser:

- ✓ Guiado por casos de uso donde los mismos son el instrumento para validar la arquitectura del software y extraer los casos de prueba.
- ✓ Centrado en la arquitectura, es donde los modelos son proyecciones del análisis y el diseño que constituyen la arquitectura del producto a desarrollar.
- ✓ Iterativo e incremental esto es durante todo el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.

Esta metodología se divide en cuatro fases el proceso de desarrollo.

Inicio que es donde se hace mayor énfasis en actividades de modelado del negocio y de requerimientos, **elaboración** donde el objetivo es determinar la arquitectura óptima, le sigue **construcción** que es donde se lleva a cabo la construcción del producto por medio de una serie de iteraciones y por último **transición** que su objetivo es obtener el producto preparado para su entrega a la comunidad de usuarios.

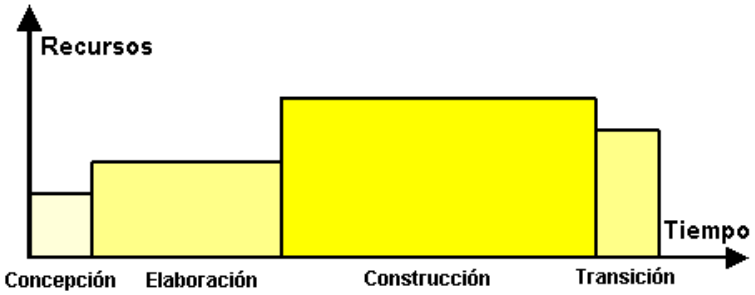


Figura 1: Ciclo de vida de la Metodología RUP.

Tomado de: *(Sánchez, 2004)*

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. (Sánchez, 2004)

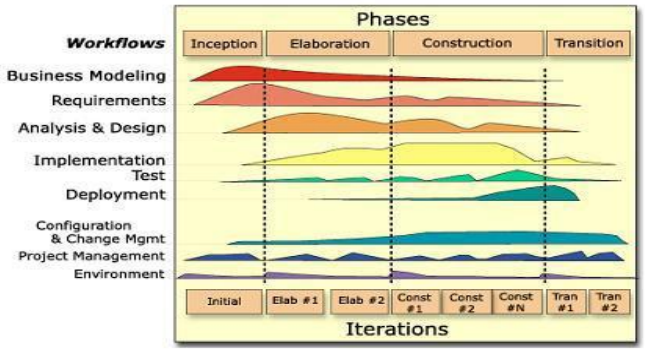


Figura 2: Fases e Iteraciones de la Metodología RUP.

Tomado de: (Sánchez, 2004)

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, para luego convertirse en un producto entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entrega o en cada iteración.
En RUP están presente elementos como son **actividades**; que no son más que los procesos que se llegan a determinar en cada iteración, están los **trabajadores**; que vienen siendo las personas involucradas en cada proceso, los **artefactos**; que no son más que documentos, modelos, o un elemento de modelo y el **flujo de actividades**; que son la secuencia de actividades realizadas por trabajadores y que producen un resultado de valor observable.
Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (Sánchez, 2004)

1.6.2 Microsoft Solution Framework (MSF).

Esta es una metodología manejable e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Figura 3: Metodología MSF.

Tomado de: (Sánchez, 2004)

La metodología MSF tiene las características de ser **adaptable**; es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar, **escalable**; donde puede organizar equipos tan pequeños entre 3 ó 4 personas, así como también, proyectos que requieren 50 personas o más, **flexible**; es cuando es utilizada en el ambiente de desarrollo de cualquier cliente y **tecnología agnóstica**; porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología. (Sánchez, 2004)
Concretamente MSF se compone de principios, modelos y disciplinas donde los principios no son más que promover comunicación abierta, trabajar para una visión compartida, fortalecer los miembros del equipo, establecer responsabilidades claras y compartidas, focalizarse en agregar valor al negocio e invertir en la calidad. Las disciplinas de MSF van a ser la Gestión de proyecto, el Control de riesgo y el Control de cambios. MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

1.6.3 Programación Extrema (XP).

La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica. Es utilizada para proyectos de corto plazo.

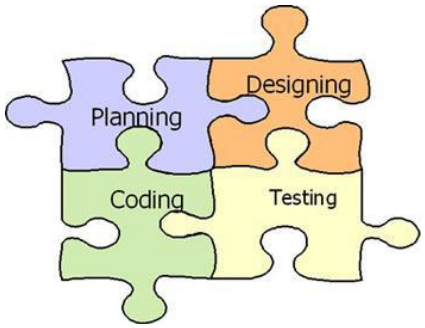


Figura 4: Metodología Extreme Programing.

Tomado de: (Sánchez, 2004)

Esta metodología está basada en **pruebas unitarias**; estas son las pruebas realizadas a los principales procesos, de tal manera que si adelanta hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores, otra característica es la **refabricación**; que se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio y la **programación en pares**; la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

A partir del estudio previo de las distintas metodologías se escogió el Proceso Unificado de Desarrollo de Software (RUP), ya que es una metodología de desarrollo de software muy bien organizada, en fases y flujos, que se ajusta perfectamente a lo que exige el cliente pues tiene como base fundamental del desarrollo: generar los artefactos completamente documentados, lo cual permite sin dudas mitigar los riesgos en forma temprana y continua, con un progreso demostrable, además los analistas del sistema no son los que van a implementar finalmente el software, lo que conlleva que los clientes sean un poco más exigentes con la claridad de todos los artefactos antes de entregarlos a los desarrolladores. Establece tempranamente una arquitectura que no se verá fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento; su enfoque basado en modelos permite un buen entendimiento entre clientes y desarrolladores facilitando la obtención de un producto con altos niveles de calidad; otra ventaja es su enfoque iterativo: la metodología parte de que se trabajará en iteraciones cortas en tiempo y con metas muy claras.

Capítulo 1: Fundamentación Teórica

1.7 Lenguajes de Modelado.

Un sistema, tanto del mundo real como en el mundo del software, es bastante complejo, por ello es necesario dividir el sistema en partes o fragmentos si se quiere entender y administrar su complejidad. Estas partes se pueden representar como modelos que describan sus aspectos esenciales. Por tanto, un paso útil en la construcción de un sistema de software es el de crear modelos que organicen y comuniquen los detalles más importante de la vida real con que se relacionan y del sistema a construir.

1.7.1 Integration Definition for Function Modeling (IDEF0).

Integration Definition for Function Modeling, es una técnica de modelación concebida para representar de manera estructurada y jerárquica las actividades que conforman un sistema o empresa, y los objetos o datos que soportan la interacción de esas actividades. Un modelo IDEF0 se compone de una serie jerárquica de diagramas que permiten mediante niveles de detalle, describir las funciones especificadas en el nivel superior. En las vistas superiores del modelo la interacción entre las actividades representadas permite visualizar los procesos fundamentales que sustentan la organización. Los elementos gráficos utilizados para la construcción de los diagramas IDEF0 son cuadros y flechas.
Permite representar el proceso cronológicamente. Se describe el flujo orientado al cliente final de ese negocio, cruzando todas las actividades de la organización que dan cumplimiento a la solicitud de producto o servicio que realiza el cliente, representando así la "cadena de valor" de la empresa.
Permite incorporar en el flujo los datos que entran y salen de las actividades, así como las reglas del negocio y los actores, todo en la misma vista. (Estrada, 2009)

1.7.2 Business Process Modeling Notation.

Business Process Modeling Notation, (BPMN, por sus siglas en inglés) define un Diagrama de Procesos de Negocio (BPD, del inglés Business Process Diagram), que se basa en una técnica de grafos de flujo para crear modelos gráficos de operaciones de procesos de negocio. Un modelo de procesos de negocio, es una red de objetos gráficos, que son actividades (trabajo) y controles de flujo que definen su orden de rendimiento.

El BPMN se compone de varios conjuntos de elementos que abarcan la representación, tanto de los Objetos del flujo y sus conexiones como los instrumentos de ayuda que son las Bandas (Swimlanes) y los Artefactos.
Además está diseñado para cubrir muchos tipos de modelados y para permitir la creación de segmentos de proceso así como procesos de negocio *end-to-end*, con diferentes niveles de fidelidad. (Barriento, 2008)

1.7.3 Lenguaje de Modelado Unificado (UML).

El Lenguaje de Modelado Unificado, (UML, por sus siglas en inglés) es un lenguaje estándar para escribir planos de software. Puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software. Se utiliza para definir un sistema, detallar los artefactos en el mismo y para documentar y construir.

UML, permite expresar un sistema de una forma gráfica de manera que otro lo pueda entender, además de especificar cuáles son las características de un sistema antes de su construcción.
Este es un lenguaje que ayuda a interpretar grandes sistemas mediante gráficos o texto, obteniendo modelos explícitos que contribuyen a la comunicación durante el desarrollo, ya que al ser estándar, pueden ser interpretados por personas que no participaron en su diseño. En este contexto, UML sirve para especificar modelos concretos, no ambiguos y completos.
UML, aporta las siguientes ventajas:

- Mayor rigor en la especificación.
- Permite realizar una verificación y validación del modelo realizado.
- Se puede automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa. Esto hace que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto. (Orallo, 2007).

UML propone diagramas con la finalidad de presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.
Luego de analizadas estas notaciones de modelado se decide utilizar el lenguaje UML, pues proporciona la posibilidad de construir todos los modelos necesarios y todos los flujos, estos permiten expresar requisitos y representar todos sus detalles, además hace que se obtenga una documentación que es válida durante todo el ciclo de vida de un proyecto. Además es el Lenguaje de Modelado de sistemas de software más conocido y utilizado en la actualidad para modelar los artefactos creados durante el proceso de desarrollo de software. Al mismo tiempo teniendo en cuenta que fue desarrollado junto con la metodología RUP, responde a todas sus necesidades y se combinan como la elección correcta del equipo de desarrollo.

1.8 Herramientas CASE.

Las herramientas de desarrollo de software (HDS) han desempeñado un importante papel en el desarrollo de aplicaciones. Como consecuencia del avance tecnológico estas han experimentado también continuos cambios.
Estas herramientas favorecen el apoyo al desarrollo de software, proporcionando un conjunto de programas de asistencia a los analistas para la Ingeniería de Software durante todo el ciclo de vida del desarrollo del sistema.

1.8.1 Rational Rose Enterprise Edition.

Rational Rose, es la herramienta CASE desarrollada por los creadores de UML, que cubre todo el ciclo de vida de un proyecto, desde la fase de inicio, formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases. Permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable.
Entre sus características principales se puede encontrar un avanzado modelado de UML para trabajar en diseños de bases de datos, con capacidad de representar la integración de los datos y los requisitos de aplicación a través de diseños lógicos y físicos, posee además una fuerte capacidad para integrarse con cualquier sistema de control de versiones.

Es importante resaltar que a pesar de esto Rational Rose presenta una necesidad de alta capacidad de procesamiento y es una herramienta poco amigable.

1.8.2 Visual Paradigm.

Visual Paradigm para UML, es una herramienta multiplataforma de modelado visual UML y una herramienta CASE muy potente y fácil de utilizar. Aporta a los desarrolladores de software una plataforma de desarrollo puntera para construir aplicaciones de calidad. Tributa una excelente interoperabilidad con otras herramientas CASE y muchos de los entornos IDE líderes del mercado. Permite la captura de requisitos, análisis, diseño e implementación, también proporciona características tales como generación del código, ingeniería inversa y generación de informes. Permite dibujar todos los tipos de diagramas de clases. Apoya los estándares más recientes de las notaciones de UML. Incorpora el soporte para trabajo en equipo, permitiendo que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros.

Visual Paradigm presenta características esenciales como son:

- Modelado colaborativo con CVS (Concurrent Versions System) y Subversión.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso, entorno todo en uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Diagramas de flujo de datos.
- Generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos, desde Sistemas Gestores de Bases de Datos existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas. Reorganización de las figuras y conectores de los diagramas UML.
- Modelo para realizar prototipos de interfaz.

Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como obtener diagramas a partir de código. (Giraldo, 2005).

1.8.3 Enterprise Architect.

Enterprise Architect (EA), es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad.
Enterprise Architect, provee trazabilidad completa desde el análisis de requerimientos hasta los artefactos de análisis y diseño, a través de la implementación y el despliegue. Combinados con la ubicación de

recursos y tareas incorporados, los equipos de Administradores de Proyectos y Calidad están equipados con la información que ellos necesitan para ayudarles a entregar proyectos en tiempo.
Las bases de Enterprise Architect están construidas sobre la especificación de UML 2.0 - pero no se detiene ahí. Usa Perfiles UML para extender el dominio de modelado, mientras que la Validación del Modelo asegura integridad. Combina Procesos de Negocio, Información y Flujos de trabajo en un modelo usando nuestras extensiones gratuitas para BPMN y el perfil Eriksson-Penker.

Después de investigadas estas herramientas se decide utilizar Visual Paradigm ya que genera toda la documentación de lo que se hace cumpliendo con los estándares establecidos, es una de las pocas herramientas CASE que soporta el análisis textual, siendo esta una técnica útil y práctica para la captura de los requisitos del sistema. Otra de las características por la que se decide usar Visual Paradigm es su disponibilidad en múltiples plataformas, ya que no obliga al usuario a desarrollar solo en el sistema operativo Windows, sino que está disponible en sistemas operativos como Windows, Linux, Unix. Además de todo lo anterior expuesto, es un requisito no funcional del cliente.

1.9 Axure RP Pro 5.5.

AXURE RP, es una herramienta de generación de prototipos de usabilidad, diagramas de flujo, así como la creación de mapas y especificaciones técnicas, que permiten el prediseño y documentación de aplicaciones web. Los prototipos generados por Axure son altamente interactivos; se puede definir las reglas de comportamiento, añadir notas y variables. Además se puede simular completamente el funcionamiento de la aplicación cuando la estás diseñando. El programa incluye decenas de widgets: formularios, tablas, checkbox, imágenes, combobox y listas. También se puede controlar eventos y definir reglas de comportamiento.
Se trata de una herramienta especializada en la tarea, así que cuenta con todo lo que se puede necesitar para crear los prototipos de forma más eficiente. Axure RP permite componer la página web visualmente, añadiendo, quitando y modificando los elementos con suma facilidad.
Donde Axure RP demuestra su grado de especialización es en las anotaciones. En este punto, permite especificar el estado de cada elemento (Propuesto, Aceptado, Incorporado), el beneficio esperado (Crítico, Importante, Útil), el riesgo, la estabilidad, a quién va dirigido y a quién se le asignará la tarea.
[Gómez, 2011] .

1.10 Patrones de Caso de Uso.

Los patrones de Casos de Uso son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios. Estos patrones son utilizados generalmente como plantillas que describen cómo deberían ser estructurados y organizados los casos de uso, a continuación se describen algunos de los patrones a utilizar en el desarrollo del sistema.

1.10.1 Concordancia.

Extrae una sub secuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

Adición
En el caso de este patrón alternativo, la sub secuencia común de casos de uso, extiende los casos de uso compartiendo la sub secuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la sub secuencia. Este patrón es preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una sub secuencia común de acciones para modelar los usos completos del sistema. [Vallejo,2009].

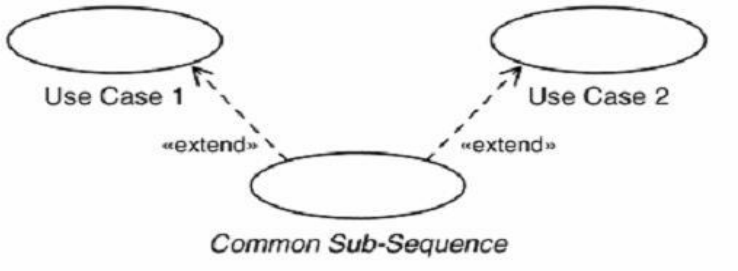


Figura 5: Modelo de Adición.

1.10.2 CRUD (Creating, Reading, Updating, Deleting).

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.
Completo
Consta de un caso de uso, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples. [Gracia, 2009]

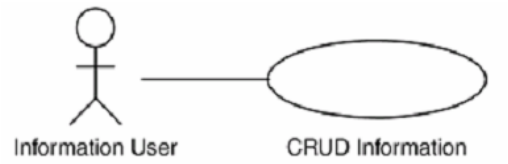


Figura 6: Modelo de CRUD Completo.

1.10.3 Inclusión Concreta.

En él, se incluye una relación del caso de uso base al caso de uso de inclusión. El último puede ser instalado en sí mismo. El caso de uso base puede ser concreto o abstracto.

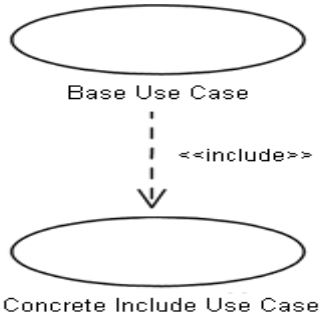


Figura 7: Modelo de Inclusión Concreta.

1.10.4 Múltiples Actores.

Roles Comunes.

Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, sólo exista una entidad externa interactuando con cada una de las instancias del caso de uso. [Gracia, 2009]

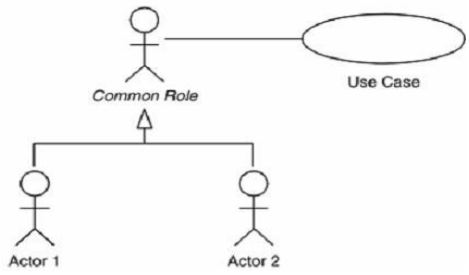


Figura 8: Modelo de Roles comunes.

1.11 Conclusiones Parciales.

Haciendo un estudio de lo antes expuesto se puede llegar a la conclusión que la UCI no cuenta con una herramienta que facilite la realización de ejercicios prácticos del tema referente a la criptografía. Se definieron las herramientas a ser usados para la elaboración del Análisis y Diseño del Laboratorio Virtual de Criptografía. Para guiar el proceso de desarrollo de la aplicación se determinó el uso de la metodología RUP, utilizando como lenguajes de modelado UML y herramienta CASE a Visual Paradigm.

Capítulo2: Solución propuesta. Análisis y diseño del sistema

CAPÍTULO:2 SOLUCIÓN PROPUESTA. ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

El presente capítulo muestra la solución que se propone para la realización del Análisis y Diseño del Laboratorio Virtual de Criptografía. Se trabajará en la modelación del sistema, utilizando la metodología, el lenguaje de modelado y la herramienta seleccionada en el capítulo anterior, es decir RUP como metodología, UML como lenguaje de modelado y Visual Paradigm como herramienta CASE. Además se determinan los actores y los casos de usos del sistema junto con sus breves descripciones. También los requisitos funcionales y no funcionales que deberá de cumplir el sistema a desarrollar.

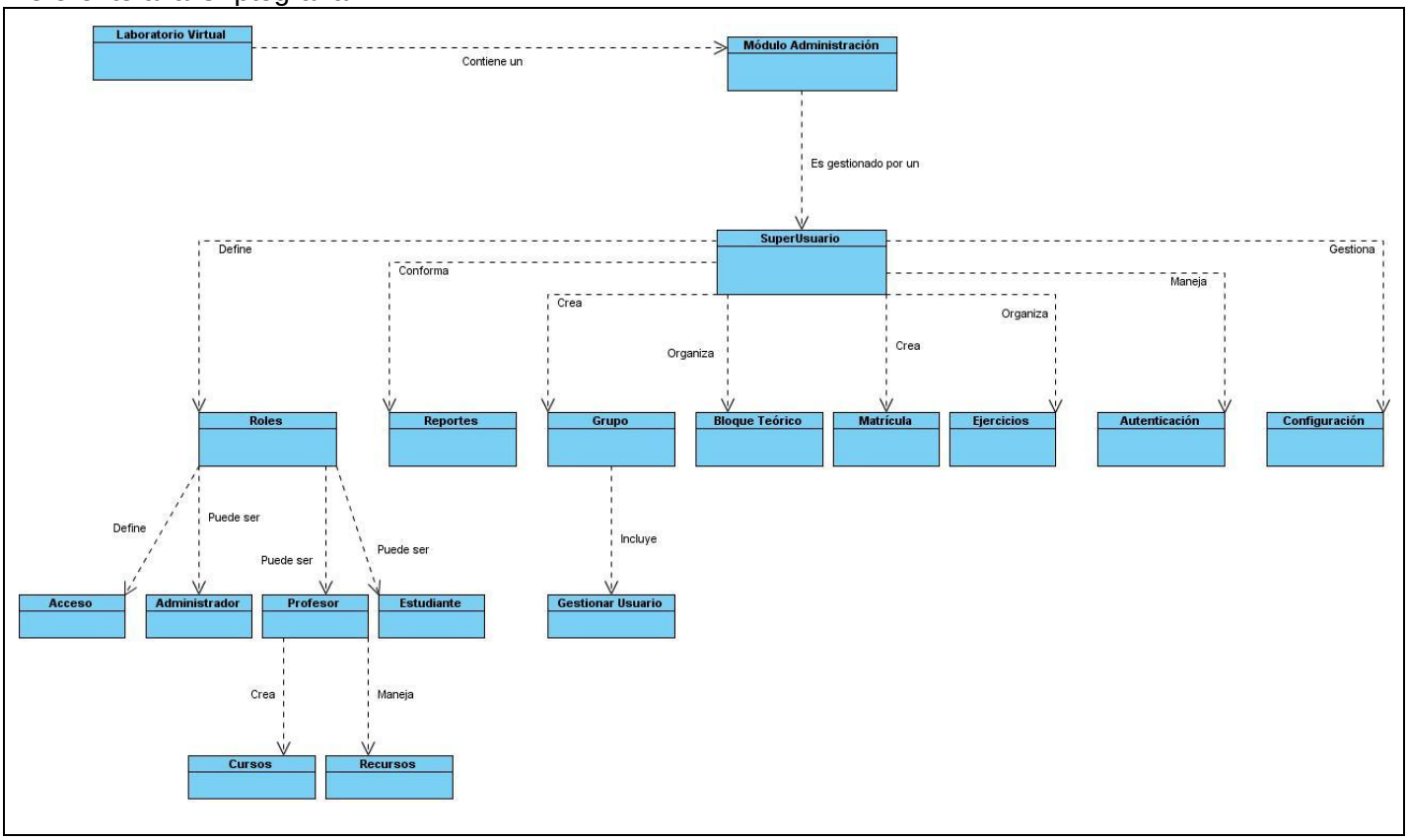
2.1 Modelo de Dominio.

Un Modelo de Dominio, es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea de construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, conceptos no propios de un sistema de software sino de la propia realidad física.
Este tiene como objetivo ayudar a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación. Es la herramienta más importante del análisis orientado a objetos.

2.1.1 Conceptos Fundamentales del Dominio.

Laboratorio Virtual : No es más que el entorno de experimentación para realizar el conjunto de prácticas de laboratorio del tema Criptografía, como apoyo a la asignatura de Seguridad Informática.
Módulo Administración: En él se realizará un conjunto de funcionalidades necesarias del Laboratorio Virtual.
Superusuario: Es el encargado de definir roles, de conformar reportes, de crear grupos, de organizar el bloque teórico y los ejercicios. Además maneja la autenticación y gestiona la configuración.
Roles: Es donde se le dan los permisos al usuario de crear, modificar, eliminar y consultar. Estos usuarios pueden ser administradores, profesores y estudiantes. También puede definir acceso.
Reportes: Es donde el sistema genera las soluciones propuestas por el usuario y son guardadas.
Grupo: Es donde se agrupan un conjunto de usuarios que van a realizar las mismas actividades. En él se puede gestionar un usuario.
Bloque Teórico: Es donde se organizará toda la teoría referente a Criptografía.
Ejercicios: Es donde se organirzarán los ejercicios referente al tema Criptografía.
Autenticación: Es donde el usuario se podrá registrar para tener acceso al sistema.
Configuración: Es donde se gestiona la configuración del sistema.
Acceso: Es donde se gestiona el acceso al sistema, este puede incluir una matrícula.
Administrador: Puede conformar reportes, crear grupos, organizar el bloque teórico y los ejercicios. Además maneja la autenticación y gestiona la configuración.
Profesor: Tendrá permisos para crear cursos y manejar recursos.
Estudiante: Se nutrirá de conocimientos con las teorías y podrá realizar ejercicios mediante los distintos recursos que existan. También podrá participar en cursos.
Gestionar Usuario: Es donde se podrá crear, modificar, eliminar y consultar usuarios.

Matrícula: Es donde el usuario, que sería un estudiante en este caso, se puede matricular en los distintos cursos.
Cursos: Es donde estarán publicados los diferentes cursos referentes al tema de la Criptografía.
Recursos: Es donde estarán publicados los distintos recursos que apoyen la realización de ejercicios referente a la Criptografía.



2.2 Requerimientos de Software.

Teniendo como entrada a la interacción con los clientes del sistema y aplicando técnicas para la elicitación de requisitos, se obtuvieron los requerimientos que debe cumplir el sistema, así como las restricciones necesarias.

Los requisitos obtenidos persiguen llegar a un entendimiento entre el cliente y el equipo de desarrollo de las condiciones que debe presentar el producto desde el punto de vista funcional. Los mismos se agruparon en dos categorías, funcionales y no funcionales. Además de llevar un proceso de control de la calidad de la especificación de los mismos.

Capítulo2: Solución propuesta. Análisis y diseño del sistema

2.2.1 Técnicas de obtención de Requisitos Utilizadas.

Para efectuar el levantamiento de requisitos, se utilizaron dos técnicas de obtención de requisitos: Primeramente se llevó a cabo con el cliente la técnica de tormenta de ideas, para tener una visión general de las características del sistema y de los procesos que se deseaban automatizar. La técnica de entrevista facilitó realizar satisfactoriamente la captura de estos requisitos pues se tomaron en cuenta las principales necesidades del usuario.

2.2.2 Requisitos Funcionales.

Como resultado de las técnicas de identificación de requisitos aplicadas se obtuvieron las siguientes condiciones o capacidades que el sistema debe cumplir:

RF01.01 Autenticar Usuario.
RF01.02 Crear Reportes.
RF01.03 Modificar Reportes.
RF01.04 Eliminar Reportes.
RF01.05 Consultar Reportes.
RF01.06 Crear Configuración.
RF01.07 Modificar Configuración.
RF01.08 Eliminar Configuración.
RF01.09 Consultar Configuración.
RF01.10 Crear Usuario.
RF01.11 Modificar Usuario.
RF01.12 Eliminar Usuario.
RF01.13 Consultar Usuario.
RF01.14 Crear Rol.
RF01.15 Modificar Rol.
RF01.16 Eliminar Rol.
RF01.17 Consultar Rol.
RF01.18 Crear Grupo.
RF01.19 Modificar Grupo.
RF01.20 Eliminar Grupo.
RF01.21 Consultar Grupo.
RF01.22 Crear Curso.
RF01.23 Modificar Curso.
RF01.24 Eliminar Curso.
RF01.25 Consultar Curso.
RF01.26 Crear Ejercicio.
RF01.27 Modificar Ejercicio.
RF01.28 Eliminar Ejercicio.
RF01.29 Consultar Ejercicio.
RF01.30 Crear Recurso.
RF01.31 Modificar Recurso.
RF01.32 Eliminar Recurso.
RF01.33 Consultar Recurso.
RF01.34 Crear Teoría.
RF01.35 Modificar Teoría.
RF01.36 Eliminar Teoría.
RF01.37 Consultar Teoría.
RF01.38 Crear Acceso.
RF01.39 Modificar Acceso.
RF01.40 Eliminar Acceso.
RF01.41 Consultar Acceso.
RF01.42 Crear Matrícula.
RF01.43 Modificar Matrícula.
RF01.44 Eliminar Matrícula.
RF01.45 Consultar Matrícula.

2.2.3 Requisitos No Funcionales.

Las cualidades o propiedades que debe cumplir el sistema son:

Orientados al usuario
Seguridad:

RNF01.01	Autenticación obligatoria y segura.
RNF01.02	Los datos que circulan por la red no viajan en texto plano.

RNF01.03 Acceso a la información según el rol.
RNF01.04 Manejo de sesiones del usuario, expira la sesión en 10 minutos.
RNF01.05 Debe permitir ocultar la información que aparece en la URL.
RNF01.06 Realizar salvas cada 5 días de la información contenida en la base de datos.
RNF01.07 El sistema debe recuperarse ante fallos.

Usabilidad:
RNF01.08 Permitir uso del teclado para realizar operaciones sobre el sistema.
RNF01.09 Debe poseer una interfaz agradable para el cliente.
RNF01.010 Mostrar la información de forma lógica y correctamente estructurada.

Robustez:
RNF01.011 El sistema debe estar accesible en todo momento.
RNF01.012 El sistema debe recuperarse en una hora tras cualquier anormalidad.
RNF01.013 Se debe hacer salvas automáticas de la Base de Datos, para no perder la información.

Disponibilidad:
RNF01.014 El sistema debe estar accesible desde EVA.
RNF01.015 Las páginas de la aplicación deben cargar en un tiempo inferior a 15 segundos.
RNF01.016 Debe garantizarse que con 300 usuarios conectados concurrentemente no disminuya el rendimiento y rapidez de la aplicación.
RNF01.017 El tiempo de carga de la aplicación debe ser de 10 a 25 segundos.

Orientados al desarrollador:

Disponibilidad:
RNF01.018 Tener una correcta y completa configuración del entorno de trabajo.

Portabilidad:
RNF01.019 Debe permitirse ser usado en cualquier plataforma.

Adaptabilidad:
RNF01.020 El sistema debe garantizar la configuración y cambio de sus parámetros de forma fácil y rápida.

Comprensibilidad:
RNF01.021 Debe garantizarse el uso de estándares de codificación.

Requisitos de Hardware

Hardware:
RNF01.022 CPU Intel Pentium 4.
RNF01.023 Memoria RAM Mínimo 768 Mb, Memoria RAM Máximo 1 GB.
RNF01.024 Capacidad Disco Duro 160 GB.
Requisitos para Servidores:

Hardware:

RNF01.025 CPU Intel Pentium 4.
RNF01.026 Memoria RAM Mínimo 1 GB, Memoria RAM Máximo 2 GB.
RNF01.027 Capacidad Disco Duro 160 GB.

Software:

RNF01.028 Sistema Operativo: GNU/Linux Debian Etch 4.0.
RNF01.029 Servidor Web Apache 2.0.
RNF01.030 Servidor Web Tomcat 5.5.

2.3 Actores del Sistema.

Un actor se define como una persona (identificada por un rol), un sistema informatizado u organización, y que realiza algún tipo de interacción con el sistema. El actor determinado fue: **Administrador.**

Descripción de los actores del sistema

Actor	Descripción
Administrador	Este actor puede insertar, modificar y eliminar un usuario, un grupo, un curso, un ejercicio, un recurso. También puede realizar reportes y asignar roles a los usuarios. Además es el encargado de gestionar la teoría, las matrículas y toda la configuración del sistema.

2.3.1 Casos de Uso.

Cada forma en que los actores usan el sistema constituye un caso de uso (CU). Luego de definidos los requerimientos del sistema y aplicando patrones como es el caso de CRUD y Concordancia, se obtuvieron las agrupaciones de funcionalidades que aportan resultados de valor para los actores del sistema. A continuación se enuncian los CU determinados para el sistema.

CU: Autenticar Usuario.

Caso de Uso:	**Autenticar Usuario.**
Actores:	Usuario
Resumen:	El caso de uso consiste en que el administrador puede autenticarse en la aplicación.
Referencias:	RF01.01

CU: Gestionar Reporte.

Caso de Uso:	**Gestionar Reporte.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear, modificar, eliminar y consultar reportes que posibilite obtener información.
Referencias:	RF01.01, RF01.02, RF01.03, RF01.04, RF01.05

CU: Gestionar Configuración.

Caso de Uso:	**Gestionar Configuración.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear, modificar, eliminar o consultar la configuración del sistema.
Referencias:	RF01.01, RF01.06, RF01.07, RF01.08, RF01.09

CU: Gestionar Usuario.

Caso de Uso:	**Gestionar Usuario.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear, modificar, eliminar y consultar usuarios.
Referencias:	RF01.01, RF01.10, RF01.11, RF01.12, RF01.13

CU: Gestionar Rol.

Caso de Uso:	**Gestionar Rol.**
Actores:	Administrador
Resumen:	El caso de uso consiste en que el administrador podrá crear, modificar y eliminar un rol.
Referencias:	RF01.01, RF01.10, RF01.11, RF01.14, RF01.15, RF01.16, RF01.17

CU: Gestionar Grupo.

Caso de Uso:	**Gestionar Grupo.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear un grupo de estudiantes, donde este grupo se podrá modificar, eliminar y consultar.
Referencias:	RF01.01, RF01.18, RF01.19, RF01.20, RF01.21

CU: Gestionar Curso.

Caso de Uso:	**Gestionar Curso.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear un curso, donde este curso podrá ser modificado, eliminado y consultado.
Referencias:	RF01.01, RF01.22, RF01.23, RF01.24, RF01.25

CU: Gestionar Ejercicio.

Caso de Uso:	**Gestionar Ejercicio.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear un ejercicio, donde este ejercicio podrá ser modificado, eliminado y consultado.
Referencias:	RF01.01, RF01.26, RF01.27, RF01.28, RF01.29

CU: Gestionar Recurso.

Caso de Uso:	**Gestionar Recurso.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear un recurso, donde este curso podrá ser modificado, eliminado y consultado.
Referencias:	RF01.01, RF01.30, RF01.31, RF01.32, RF01.33

CU: Gestionar Teoría.

Caso de Uso:	**Gestionar Teoría.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear una teoría, donde esta podrá ser modificada, eliminada y consultada.
Referencias:	RF01.01, RF01.34, RF01.35, RF01.36, RF01.37

CU: Gestionar Acceso.

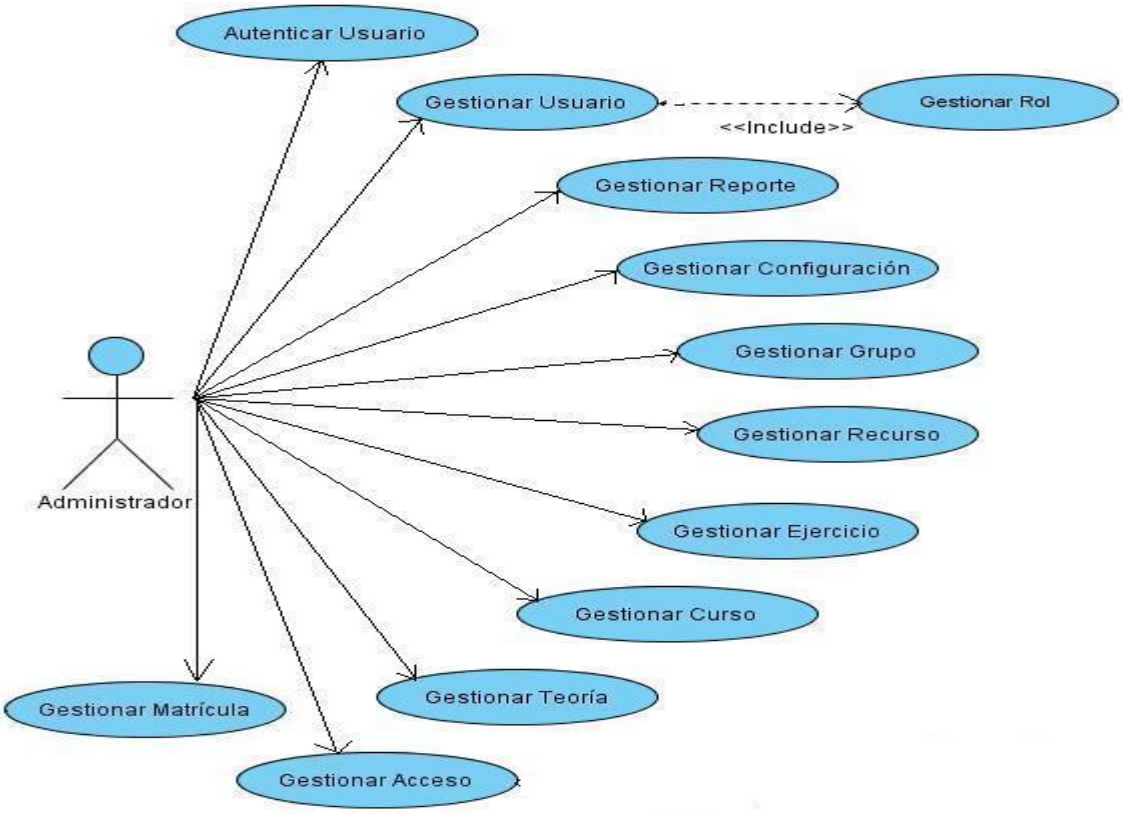
Caso de Uso:	**Gestionar Acceso.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear un acceso, donde este podrá ser modificado, eliminado y consultado.
Referencias:	RF01.01, RF01.38, RF01.39, RF01.40, RF01.41

CU: Gestionar Matrícula.

Caso de Uso:	**Gestionar Matrícula.**
Actores:	Administrador
Resumen:	El caso de uso consiste en crear una matrícula, la cual podrá ser modificada, eliminada y consultada.
Referencias:	RF01.01, RF01.42, RF01.43, RF01.44, RF01.45

Luego del estudio a los diferentes patrones de caso de uso realizado, se decide seleccionar los patrones: Conordancia para inluir el CU Gestionar Rol dentro del CU Gestionar Usuario y CRUD para todos los CU gestionar.

2.3.2 Diagrama de Casos de Uso del Sistema.



2.3.3 Descripción de Casos de Uso del Sistema.

Luego de definir los casos de uso del sistema, se procede a la descripción textual de los mismos con la intención de especificar en detalles cada una de las funcionalidades que deben ser implementadas. La descripción de los casos de uso constituye una guía para los desarrolladores y un documento de obligatorio cumplimiento en cuanto a desarrollo de funcionalidades en el sistema. Seguidamente se muestra la descripción de los casos de uso del módulo Administración del Laboratorio Virtual de Criptografía.

2.3.4 Descripción del Caso de Uso Gestionar Usuario.

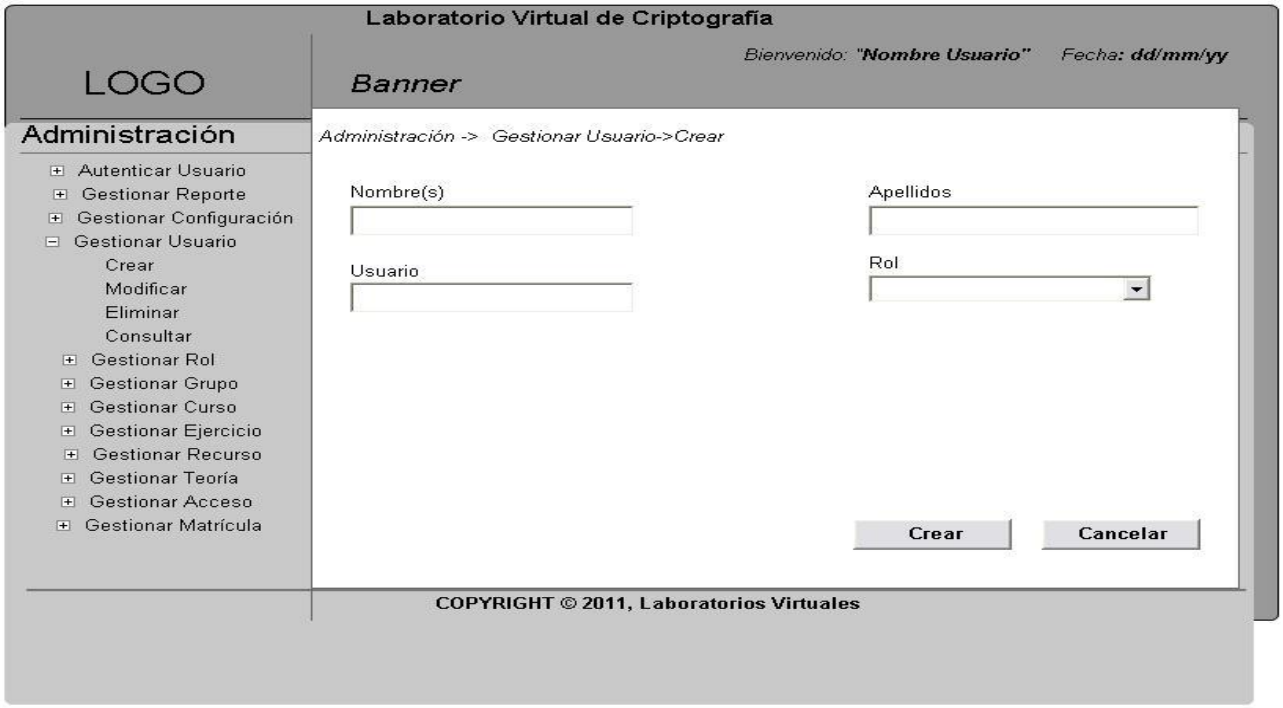
Caso de Uso:	Gestionar Usuario.
Actores:	Administrador.
Resumen:	El caso de uso consiste en que el administrador podrá crear, modificar,

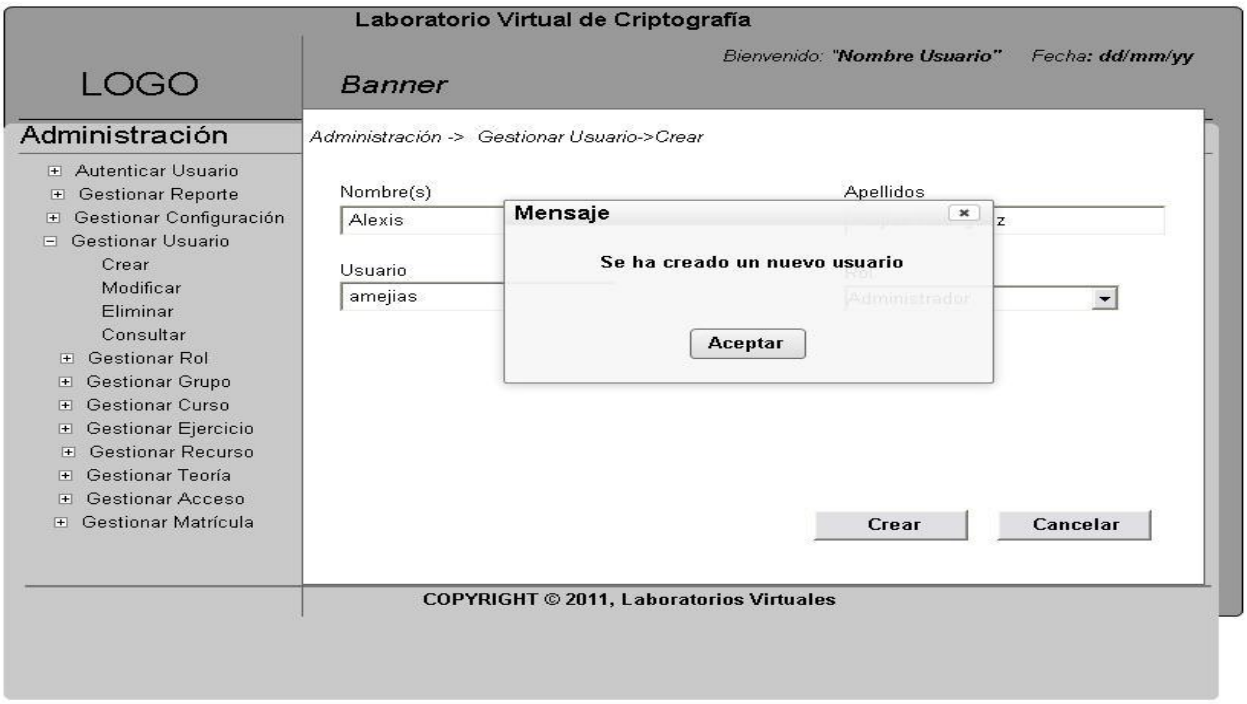
	eliminar y consultar usuarios en la aplicación.
Precondiciones:	El administrador debe estar autentificado en la aplicación.
Referencias	**RF01.01, RF01.10, RF01.11, RF01.12, RF01.13**
Prioridad	Alta
Nivel	Administrador

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el Administrador desea Crear, Modificar, Eliminar o Consultar un usuario.	2. La aplicación muestra la interfaz **"Gestionar Usuario"** con los siguientes datos: • **Crear Usuario.** • **Modificar Usuario.** • **Eliminar Usuario.** • **Consultar Usuario.**
3. Si el Administrador selecciona la opción Crear Usuario, ir a la sección **"Crear Usuario"**. 4. Si el Administrador selecciona la opción Modificar Usuario, ir a la sección **"Modificar Usuario**. 5. Si el Administrador selecciona la opción Consultar Usuario, ir a la sección **"Consultar Usuario"**. 6. Si el Administrador selecciona la opción Eliminar Usuario, ir a la sección **"Eliminar Usuario"**.	
Sección "Crear Usuario"	
Acción del Actor	**Respuesta del Sistema**
	1. La aplicación muestra la interfaz **"Crear Usuario"** con los siguientes datos: • **Nombre.** • **Apellidos.** • **Usuario.** • **Rol.** Y las opciones **"Crear"** y **"Cancelar"**.
2. El Administrador inserta los datos solicitados por la interfaz. 3. El Administrador selecciona la opción	4. La aplicación valida que el usuario no existe ya en la base de datos, y muestra un mensaje **"Se ha creado un nuevo**

"Crear".	**usuario"**. 5. En caso que el usuario exista se debe ir a la sección **"Usuario Existente".** 6. En caso que los datos estén incompletos, ir a la sección **"Datos Incompletos"**.
7. El Administrador selecciona la opción ¨**Aceptar**¨. Termina el caso de uso.	

Prototipo de Interfaz

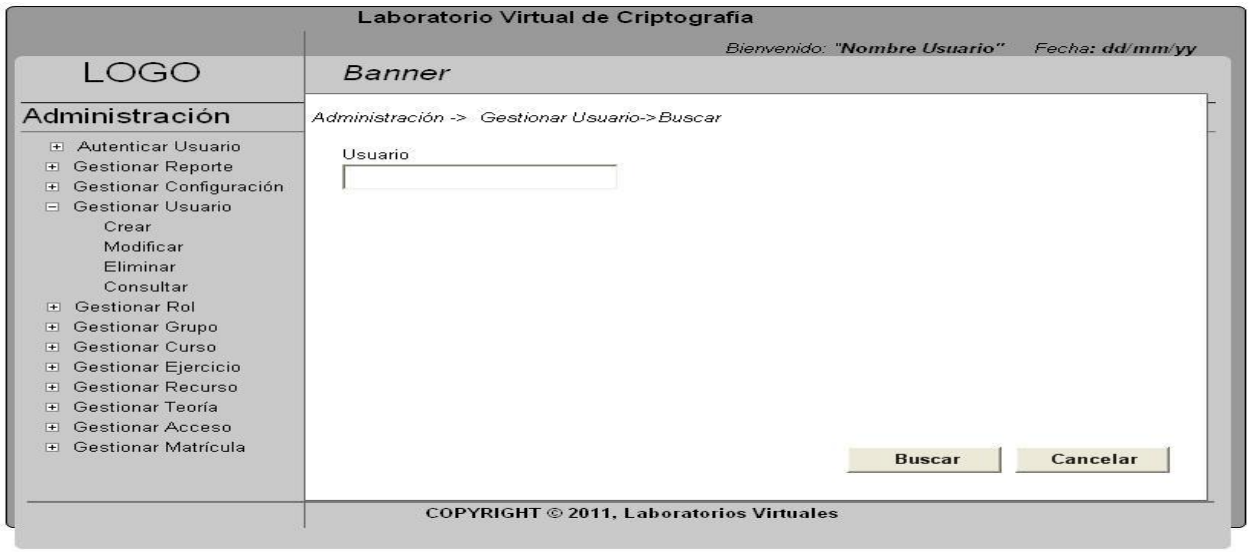


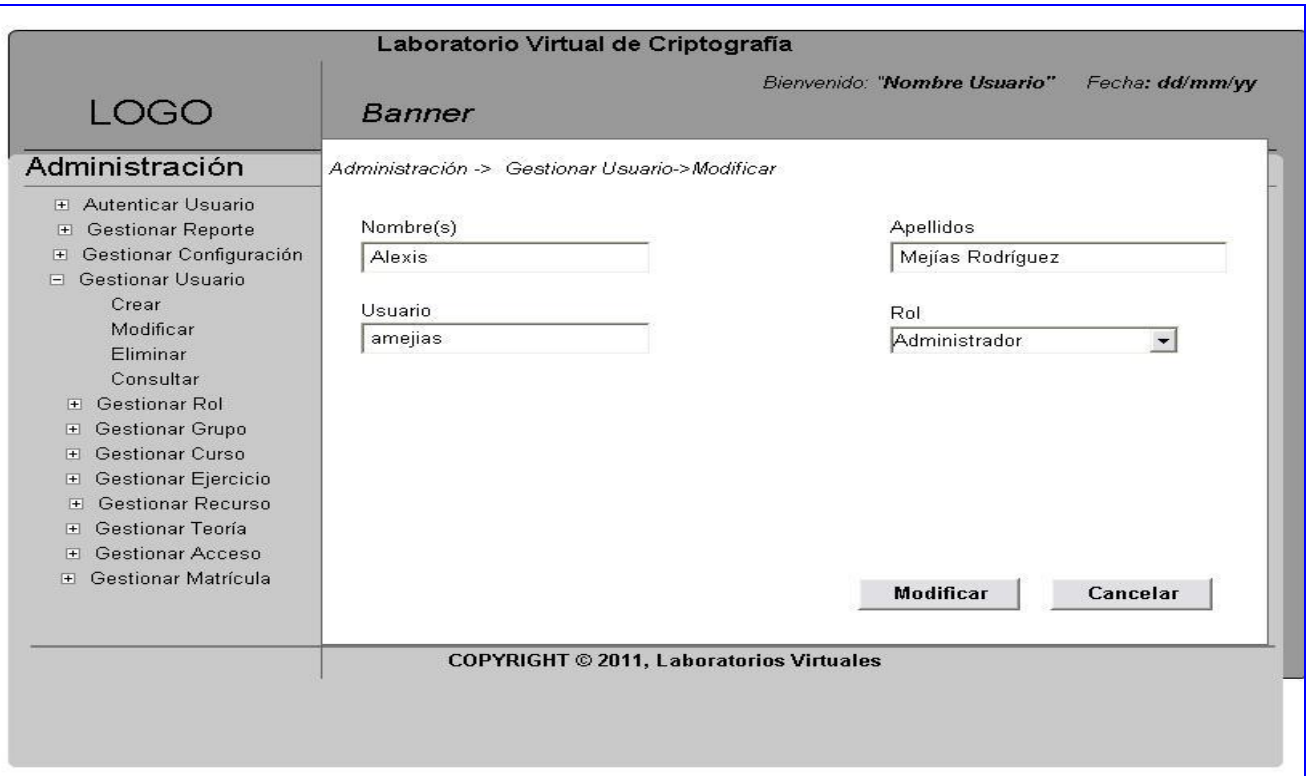


Sección "Modificar Usuario"	
Acción del Actor	**Respuesta del Sistema**
	1. La aplicación muestra la interfaz **"Buscar Usuario"** con los siguientes datos: • **Usuario.** Y las opciones **"Buscar"** y **"Cancelar"**.
2. El Administrador introduce los datos solicitados por la interfaz, y selecciona la opción **"Buscar"**.	3. La aplicación muestra la interfaz **"Modificar Usuario"** con los siguientes datos: • **Nombre.** • **Apellidos.** • **Usuario.** • **Rol.** Y las opciones **"Modificar"** y **"Cancelar"**.
4. El Administrador selecciona la opción **"Modificar"**.	5. Si la acción fue correcta la aplicación muestra un mensaje **"El usuario ha sido modificado correctamente."**. 6. En caso que no se inserte algún dato solicitado por la aplicación, ir a la sección

	“Datos Incompletos”.
7. El Administrador selecciona la opción **“Aceptar”**. Termina el caso de uso.	

Prototipo de Interfaz

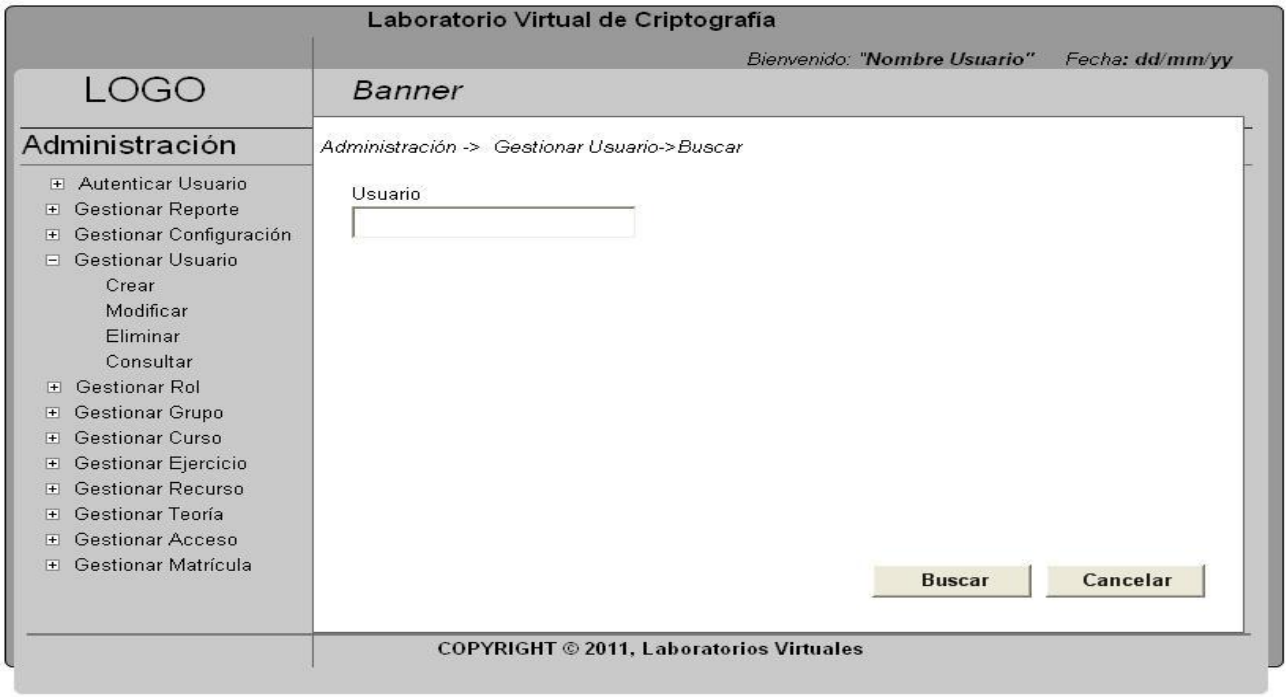


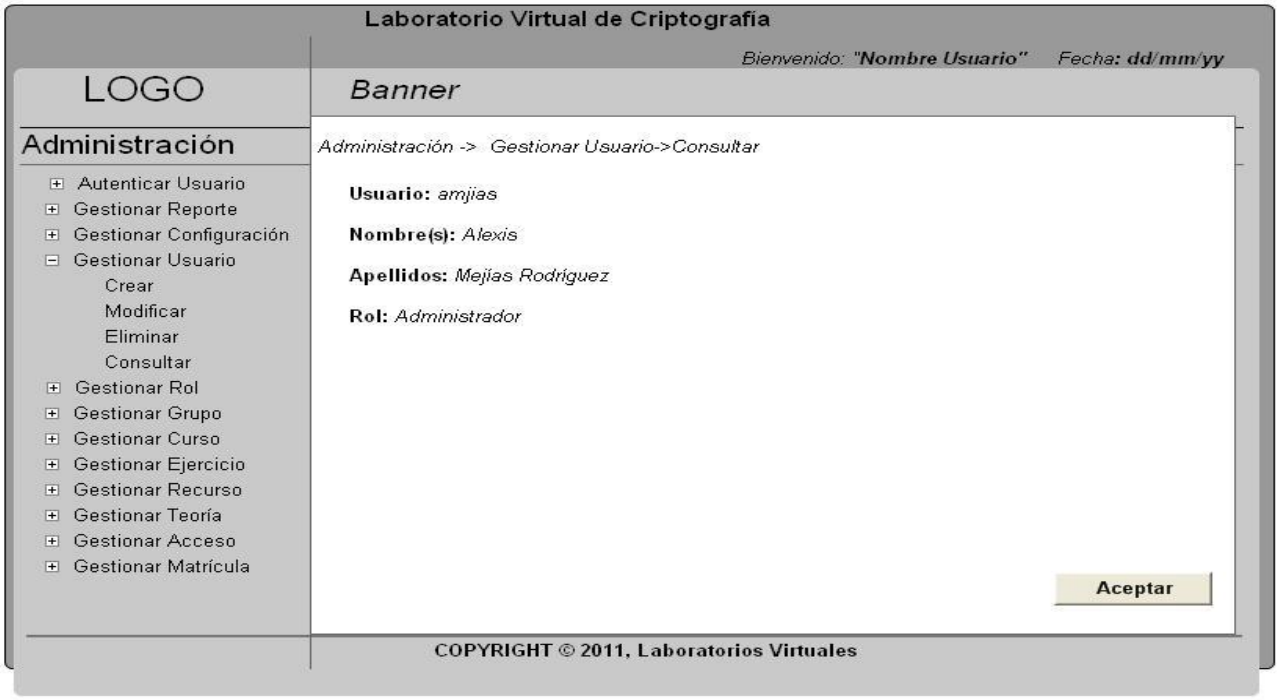


Sección "Consultar Usuario"

Acción del Actor	Respuesta del Sistema
	1. La aplicación muestra la interfaz **"Buscar Usuario"** con los siguientes datos: • **Usuario.** Y las opciones **"Buscar"** y **"Cancelar"**.
2. El Administrador inserta los datos solicitados por la interfaz, y selecciona la opción **"Buscar".**	3. La aplicación muestra la interfaz **"Consultar Usuario"** con los siguientes datos: • **Nombre.** • **Apellidos.** • **Usuario.** • **Rol.** Y la opción **"Aceptar"**. 4. En caso que no se inserte algún dato solicitado por la aplicación**."**, ir a la sección **"Datos Incompletos"**.
5. El Administrador selecciona la opción **"Aceptar".** Termina el caso de uso.	

Prototipo de Interfaz

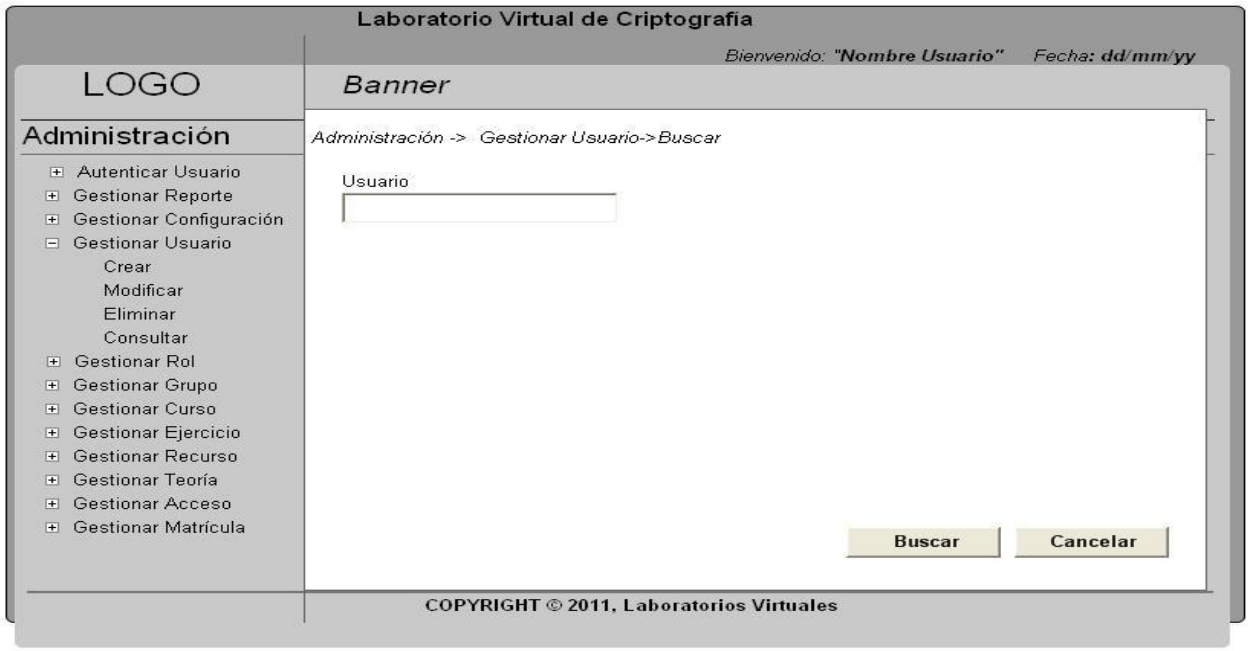


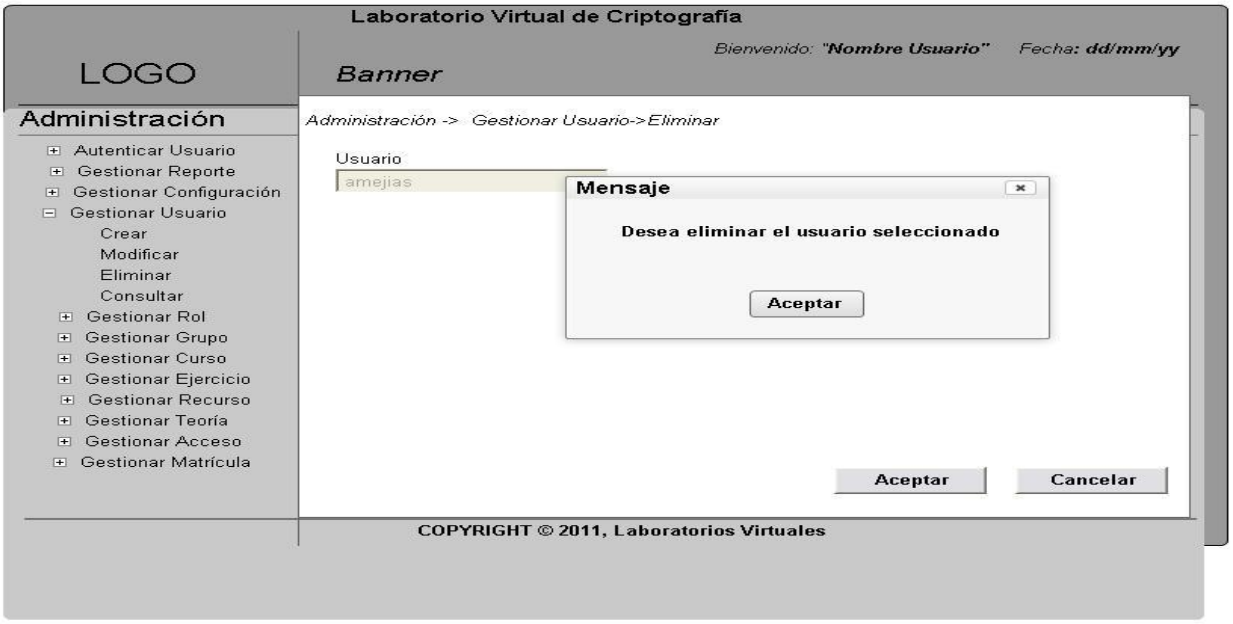


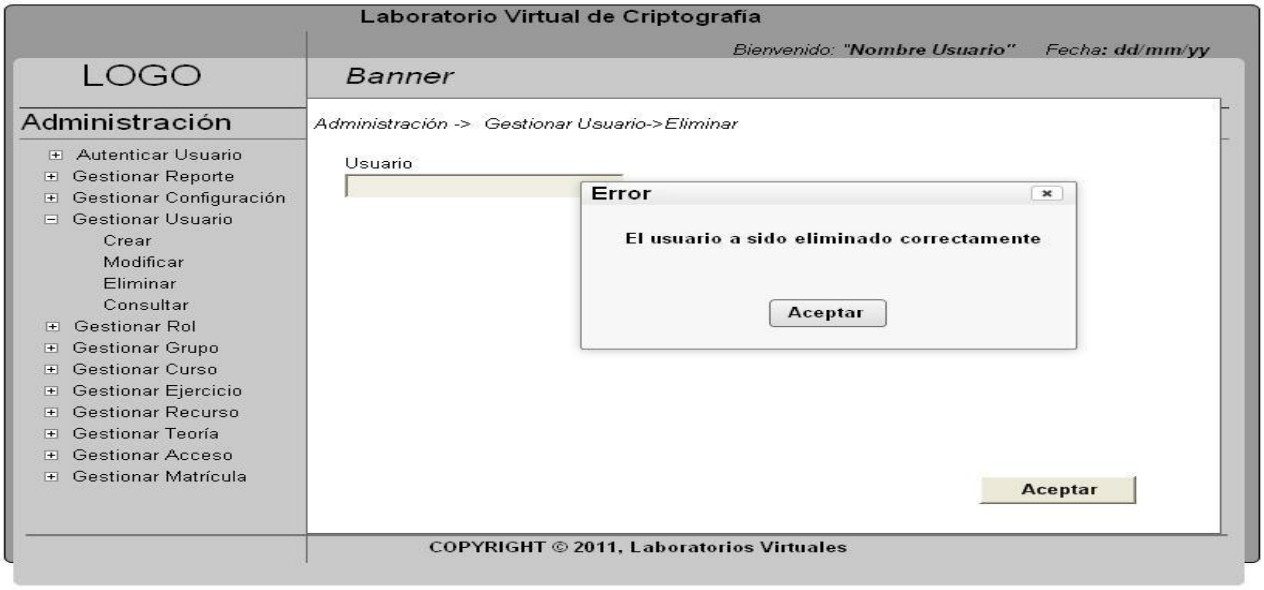
Sección "Eliminar Usuario"	
Acción del Actor	**Respuesta del Sistema**
	1. La aplicación muestra la interfaz **"Buscar Usuario"** con los siguientes datos: • **Usuario.** Y las opciones **"Buscar"** y **"Cancelar"**.
2. El Administrador inserta los datos solicitados por la interfaz, y selecciona la opción **"Buscar".**	3. La aplicación muestra la interfaz **"Eliminar Usuario"** con los siguientes datos: • **Usuario.** Y la opción **"Eliminar"**.
4. El Administrador selecciona el usuario que desea eliminar, y selecciona la opción **"Eliminar".**	5. La aplicación muestra un mensaje **"Desea eliminar el usuario seleccionado."** Y las opciones **"Aceptar"** y **"Cancelar"**.
6. El Administrador selecciona la opción **"Aceptar".**	7. La aplicación muestra un mensaje **"El usuario ha sido eliminado correctamente".** 8. En caso que no se inserte algún dato solicitado por la aplicación, ir a la sección **"Datos Incompletos"**.
9. El Administrador selecciona la opción	

"Aceptar". Termina el caso de uso.	

Prototipo de Interfaz



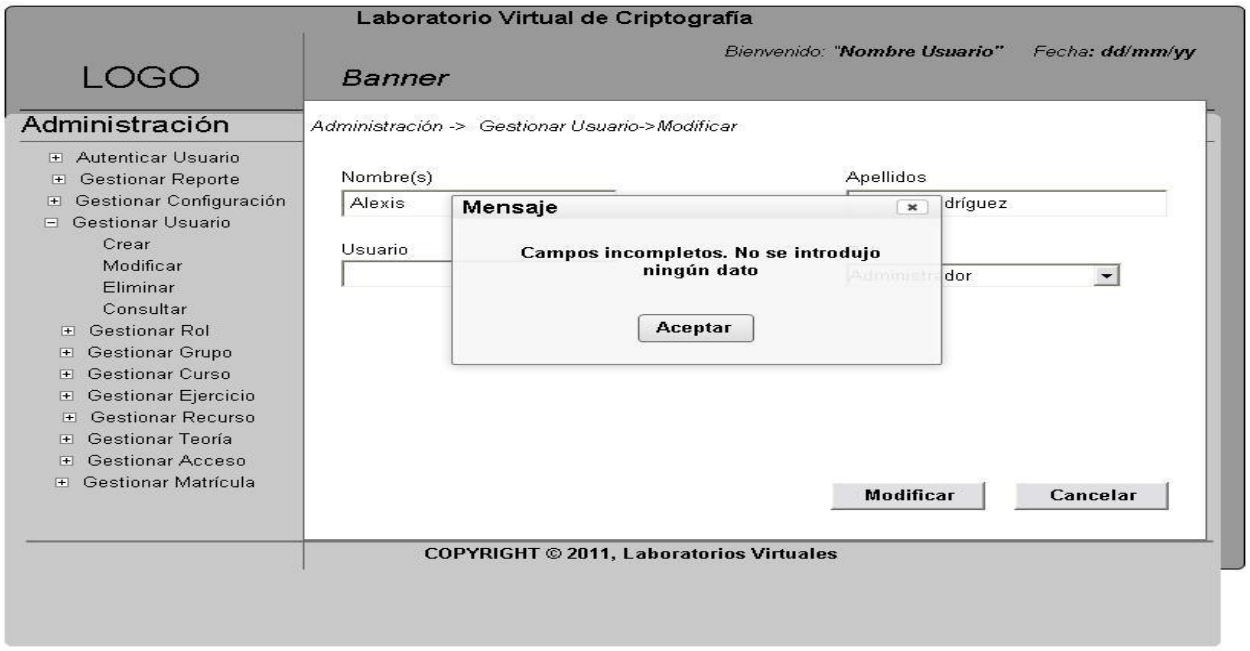


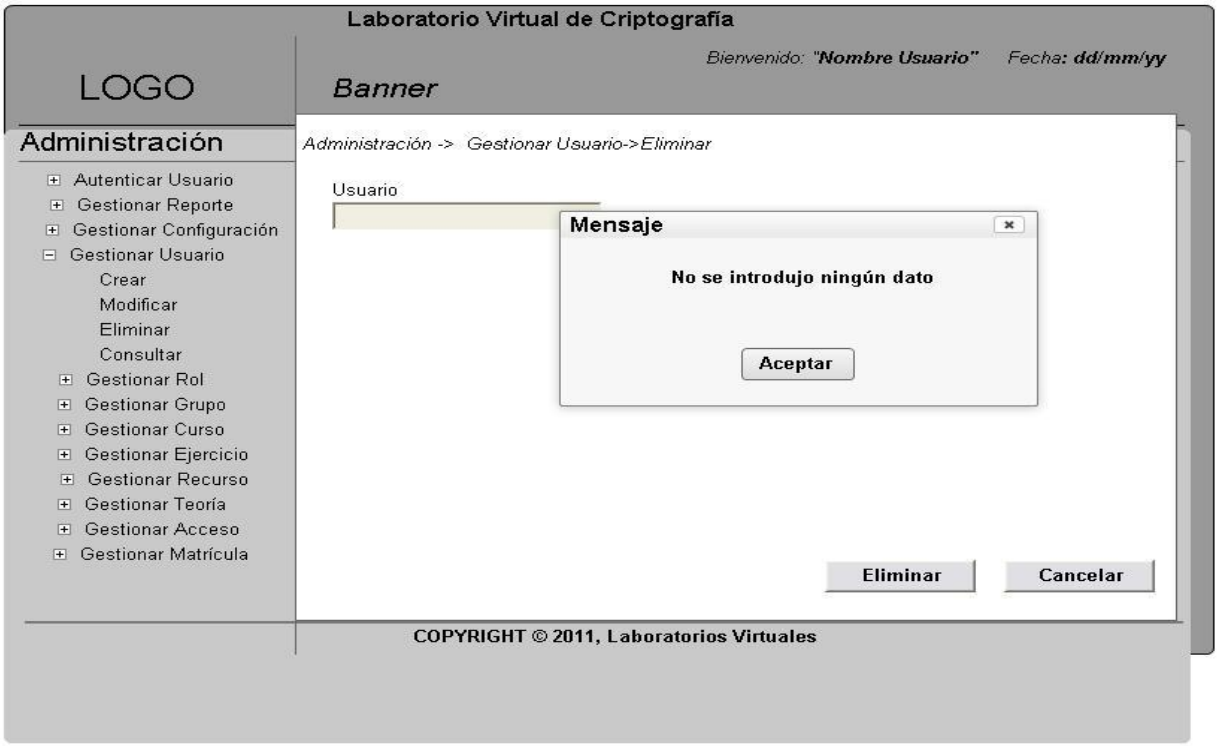


Flujo Alternos	
Sección "Datos Incompletos"	
Acción del Actor	**Respuesta del Sistema**
	1. La aplicación muestra el mensaje **"Existen campos incompletos o vacíos. Por**

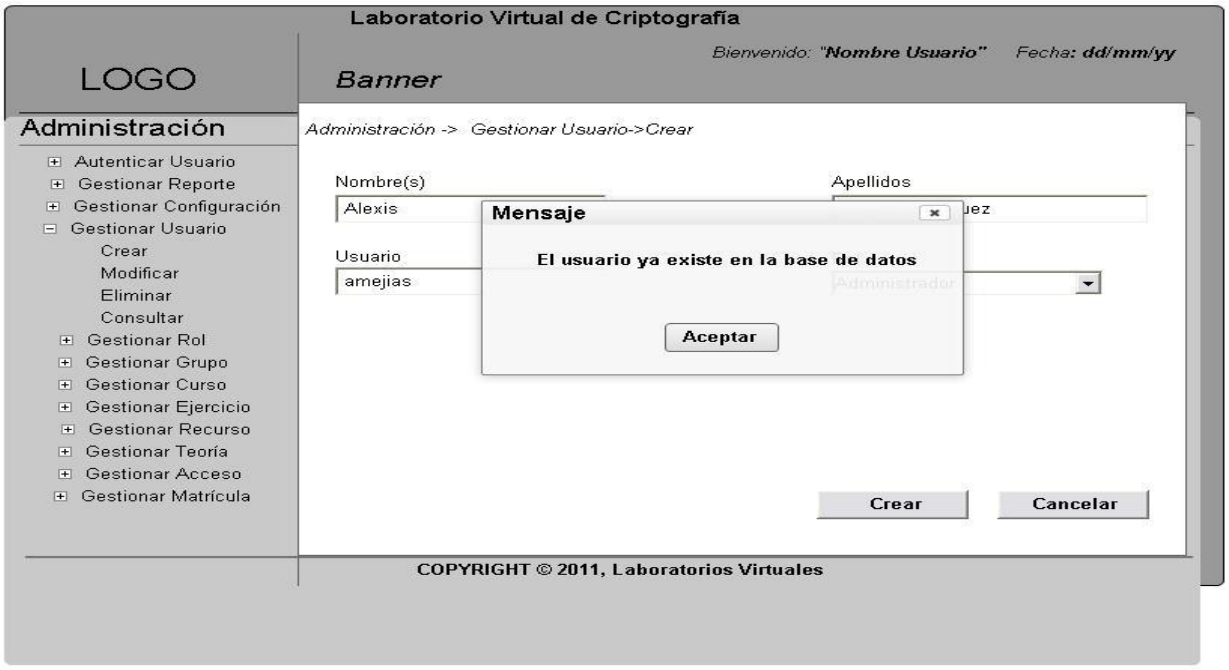
	favor, complete los datos".
2. El usuario selecciona la opción **"Aceptar"**.	

Prototipo de Interfaz





Sección "Usuario Existente"	
Acción del Actor	**Respuesta del Sistema**
	1. La aplicación muestra el siguiente mensaje **"El usuario ya existe en la base de datos".**
1. El Administrador selecciona la opción **"Aceptar"**.	
Prototipo de Interfaz	



Pos condiciones	2. El usuario debe quedar autentificado.

2.4 Modelo de Análisis.

En el análisis, se analizaron los requisitos con mayor profundidad, utilizando el lenguaje de los desarrolladores. El objetivo es conseguir una compresión más precisa de estos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero. El análisis prepara y simplifica la actividad de diseño e implementación, delimitando los temas que debe resolverse y las decisiones que deben tomarse en esas actividades.

2.4.1 Clases de Análisis.

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

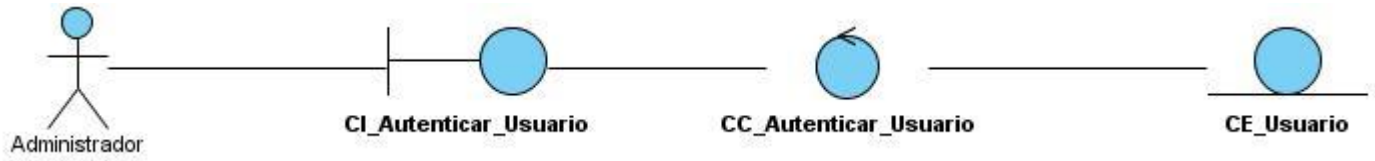
Clase Interfaz: Se utilizan para modelar la interacción entre el sistema y sus actores. Esta interacción a menudo implica recibir información y peticiones de los usuarios y sistemas externos. Cada clase de interfaz debe asociarse con al menos un actor.

Clase Control: Se usa para coordinar la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso, por lo que definen el flujo de control y las transacciones dentro de un caso de uso delegando el trabajo a otros objetos.

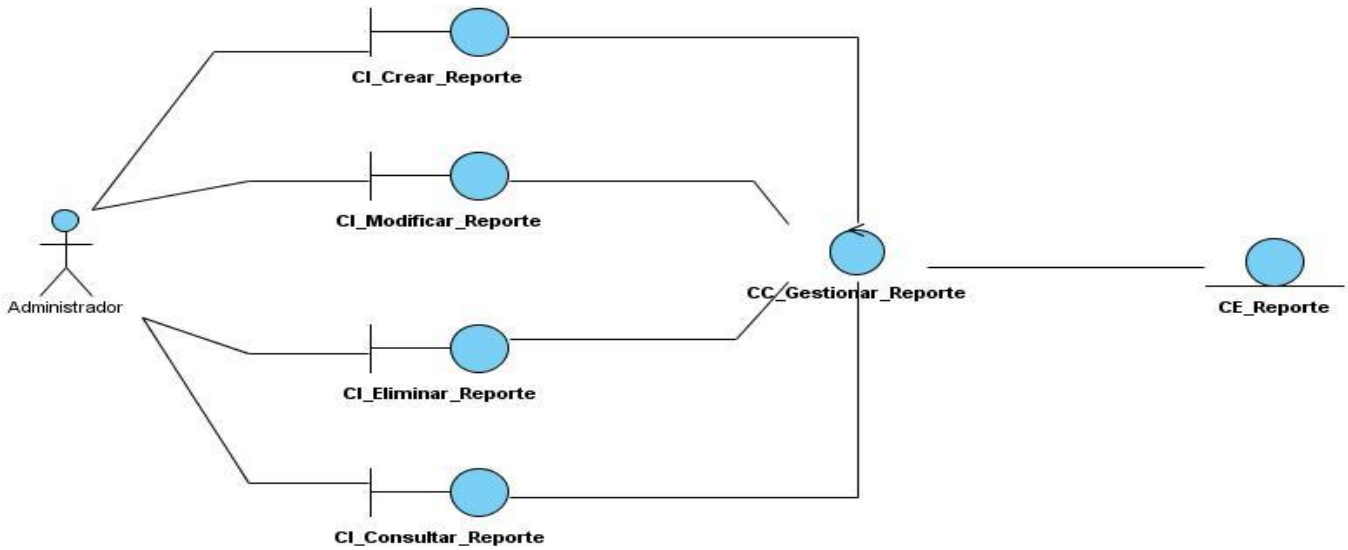
Clase Entidad: Estas modelan información que posee una larga vida y que a menudo es persistente. Explican fenómenos, conceptos y sucesos que ocurren en el mundo real. La fuente principal de obtención son las clases entidades del negocio y el glosario de términos que se ha ido elaborando.

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa los objetos del mundo real, no de la implementación automatizada de estos.

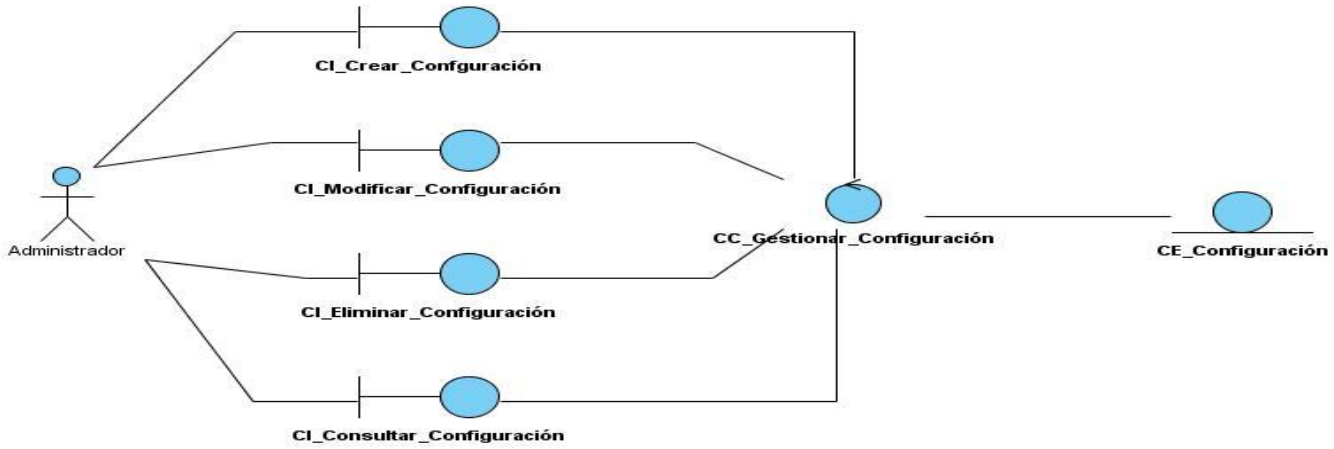
DCA Autenticar Usuario



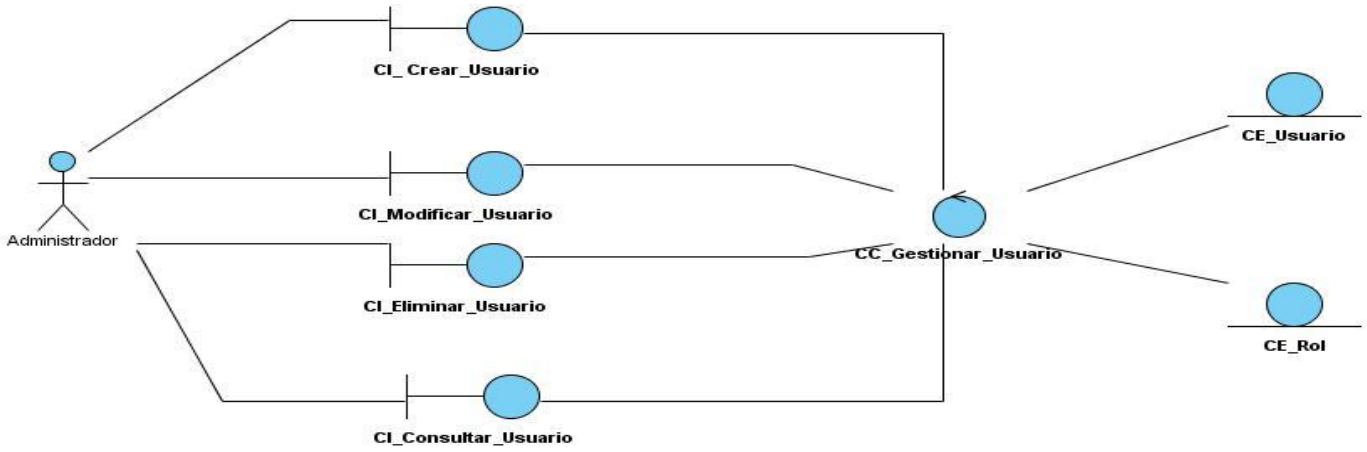
DCA Gestionar Reporte



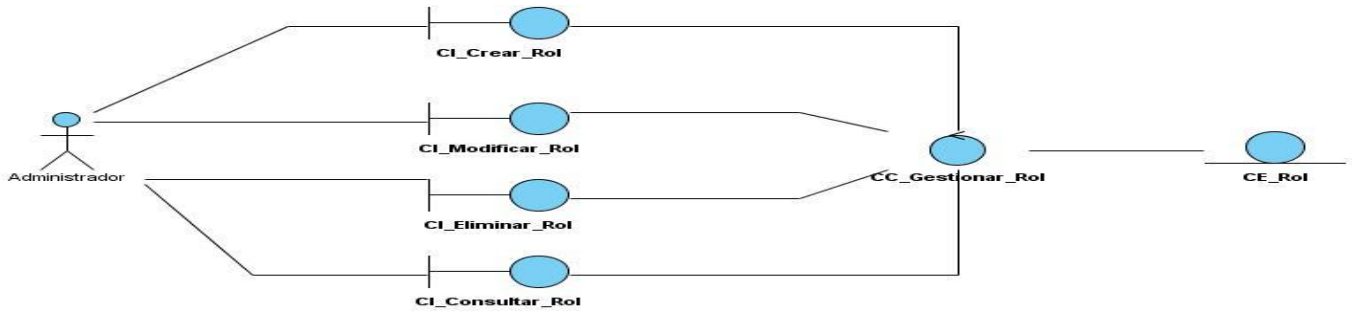
DCA Gestionar Configuración



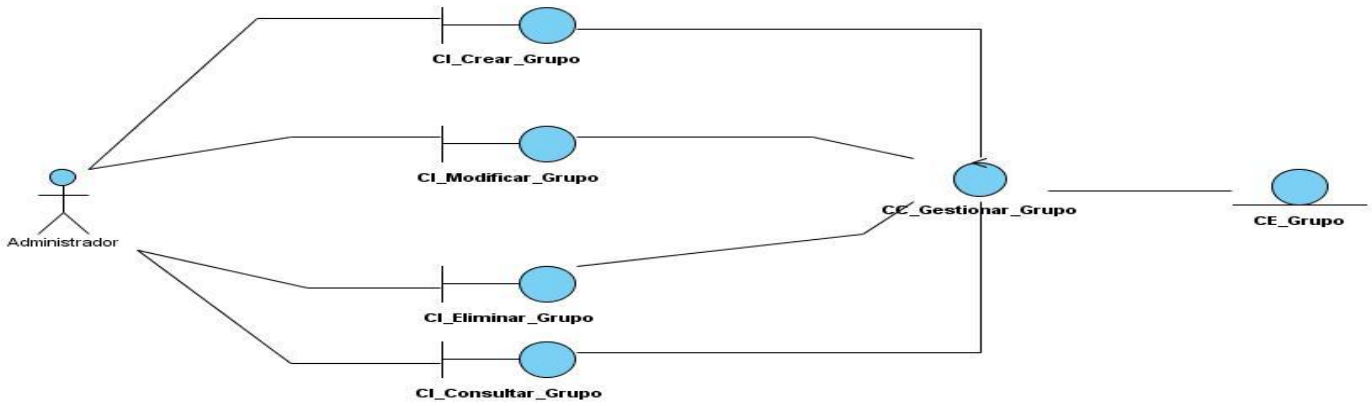
DCA Gestionar Usuario



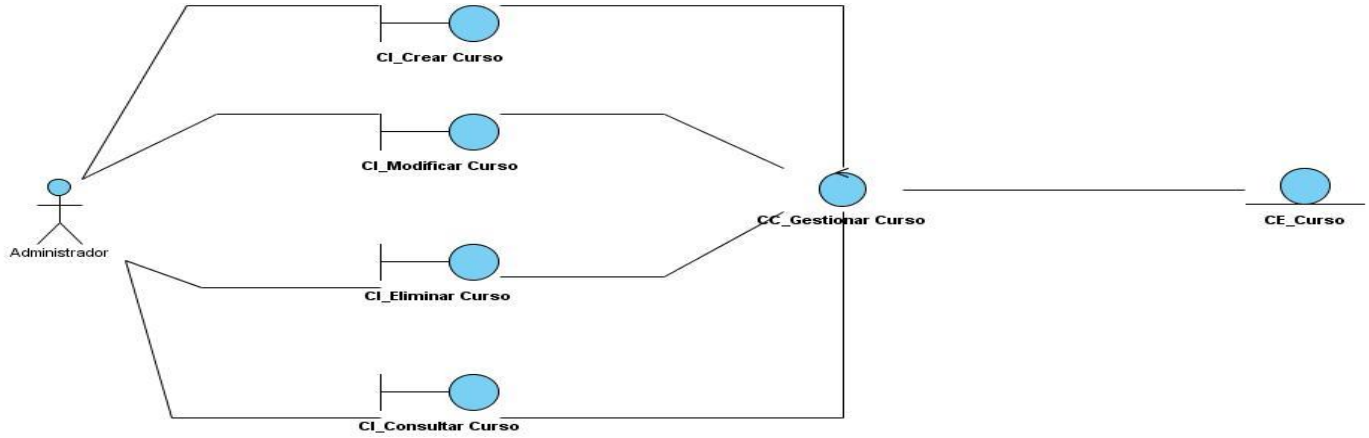
DCA Gestionar Rol



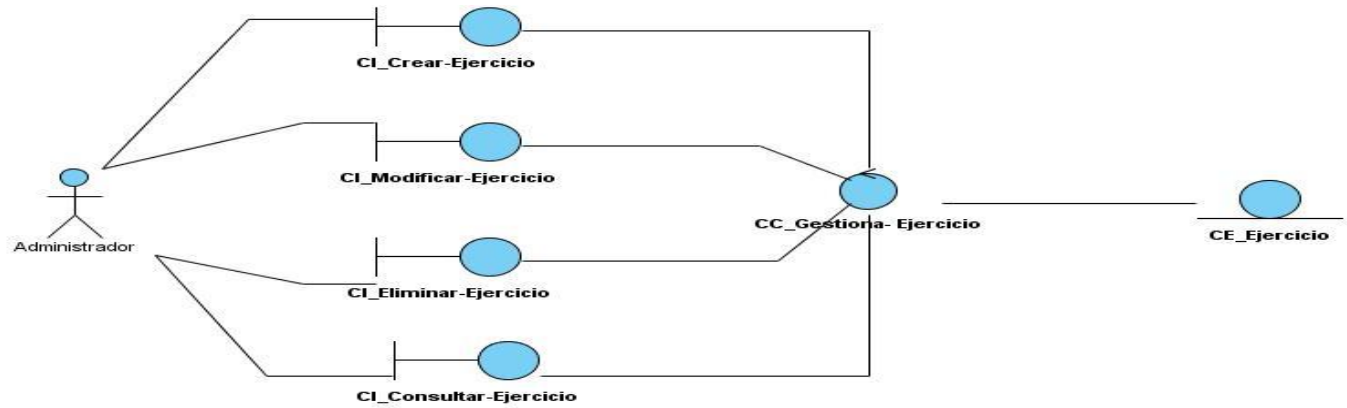
DCA Gestionar Grupo



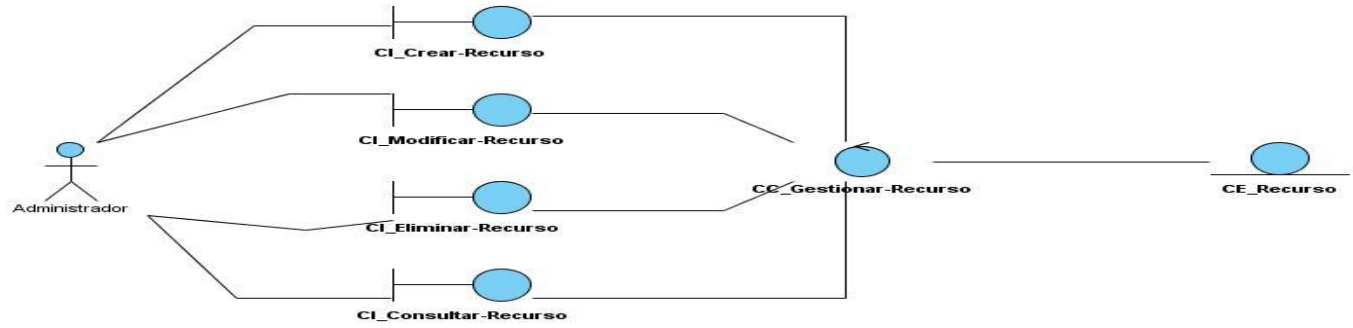
DCA Gestionar Curso



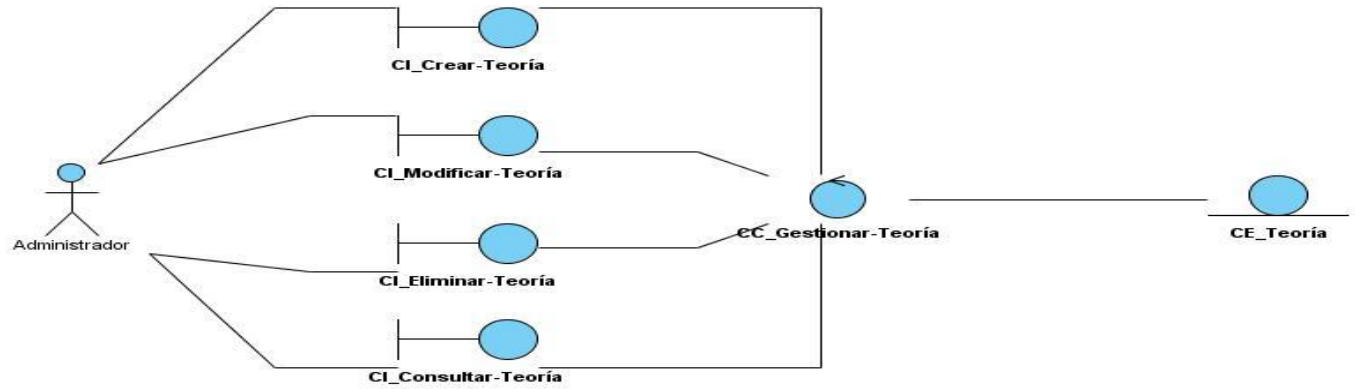
DCA Gestionar Ejercicio



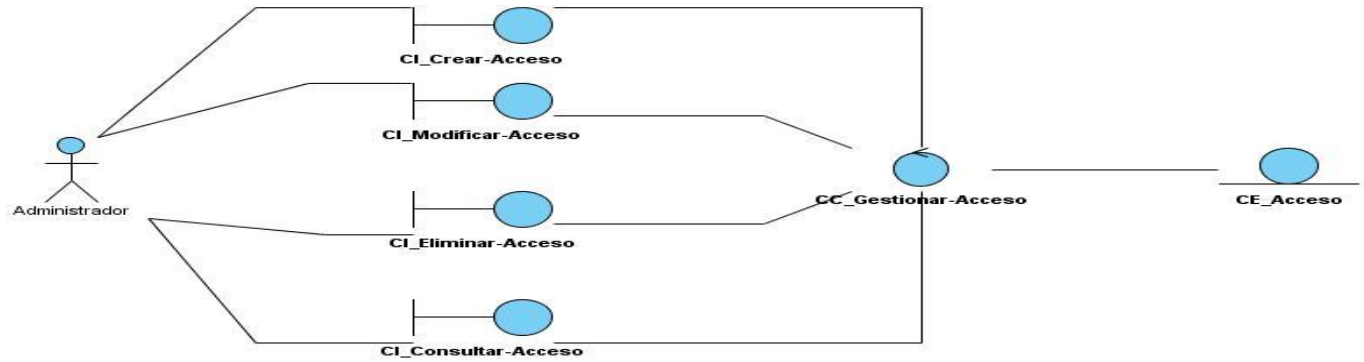
DCA Gestionar Recurso



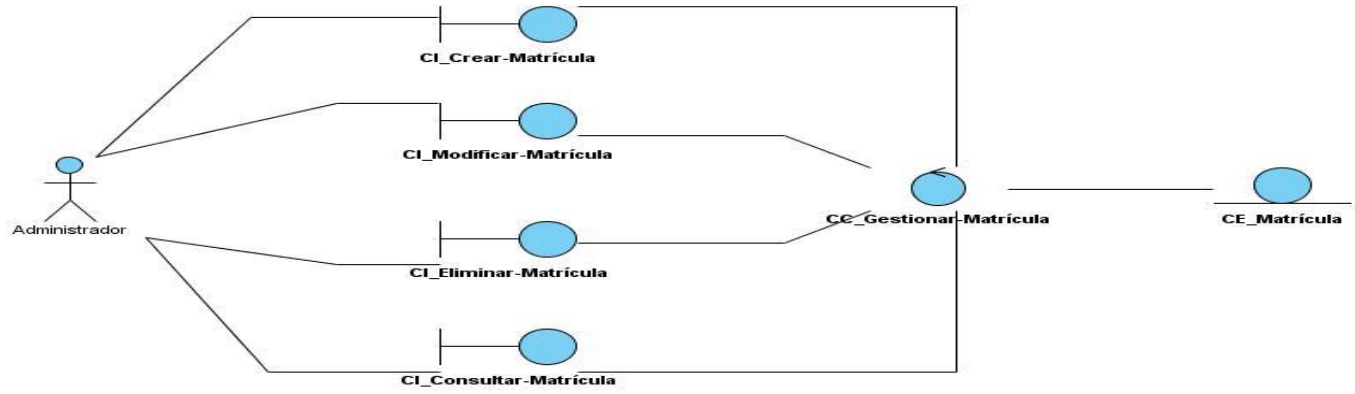
DCA Gestionar Teoría



DCA Gestionar Acceso



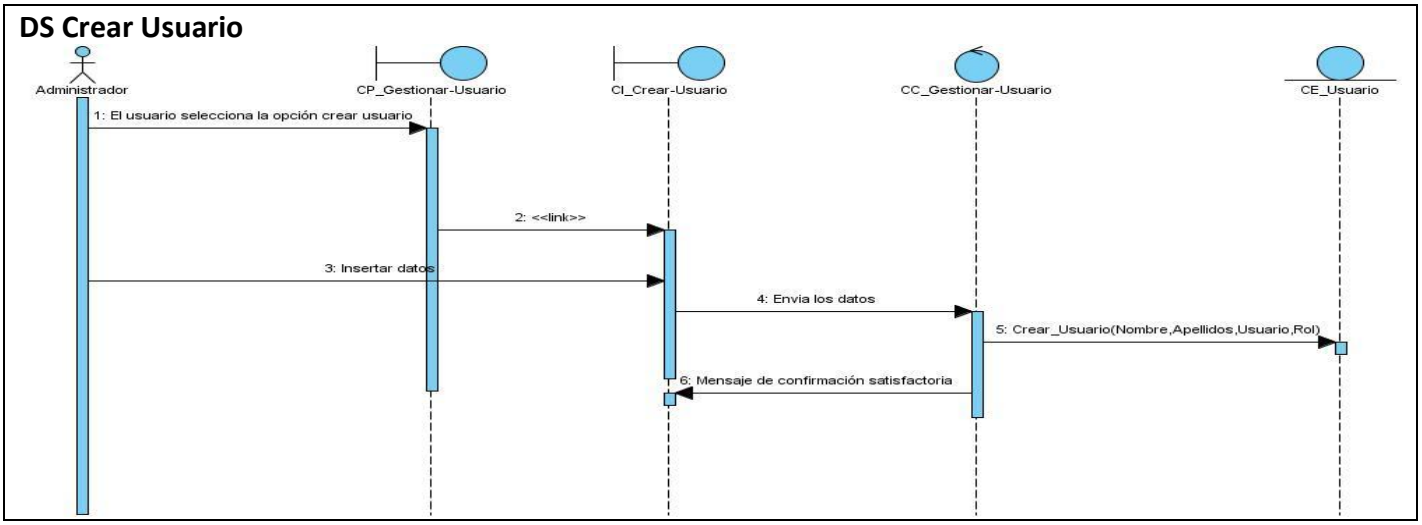
DCA Gestionar Matrícula



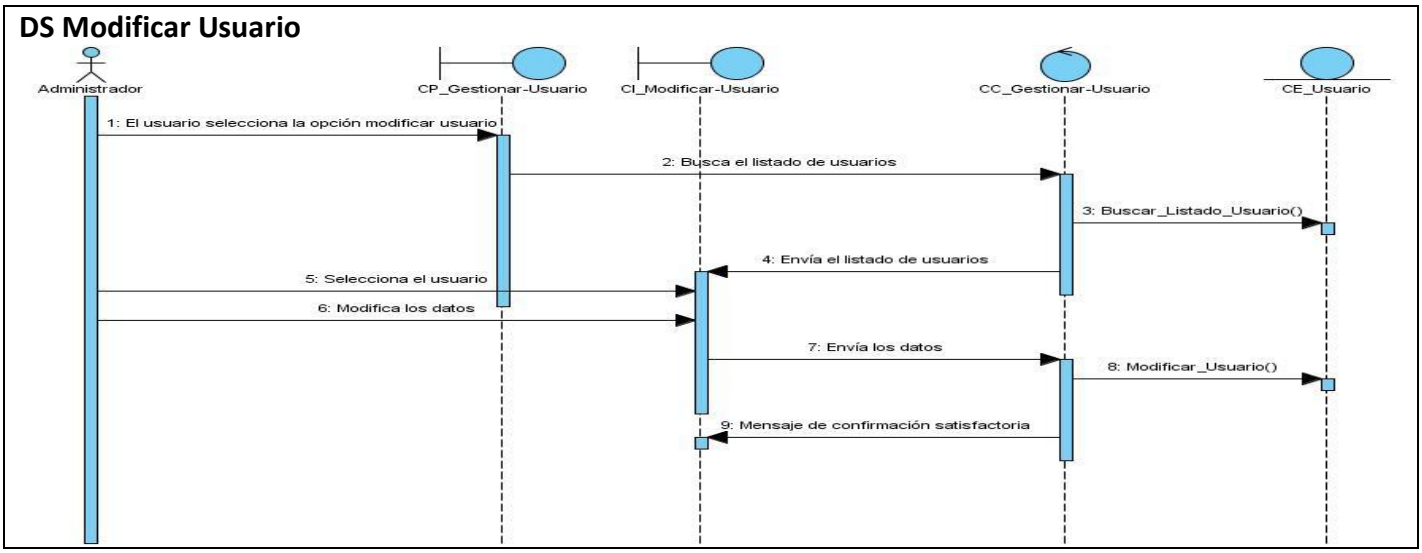
2.4.2 Diagramas de Interacción (Secuencia).

Un diagrama de interacción es un artefacto que explica gráficamente cómo los objetos interactúan a través de mensajes para realizar las tareas, modelando el comportamiento dinámico del sistema. Estos constituyen una de las herramientas más importantes para el análisis y diseño orientado a objetos pues describen la interacción entre los objetos. Estos diagramas son muy útiles para visualizar, especificar, construir y documentar la dinámica entre dos objetos. Hay dos tipos de diagramas de interacción: Diagramas de Secuencia y Diagramas de Colaboración. Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. Representa los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo. Se pueden colocar etiquetas (como restricciones de tiempo y descripciones de acciones) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

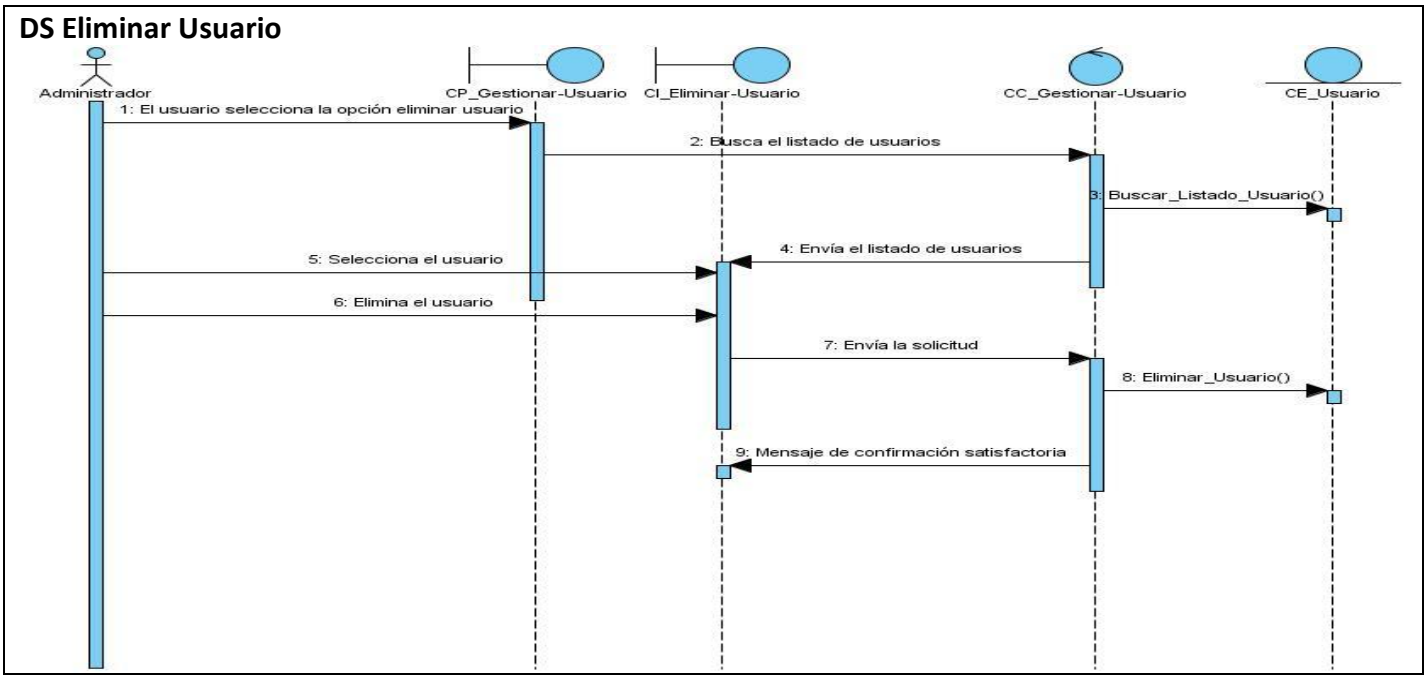
Diagrama de Secuencia Gestionar Usuario.

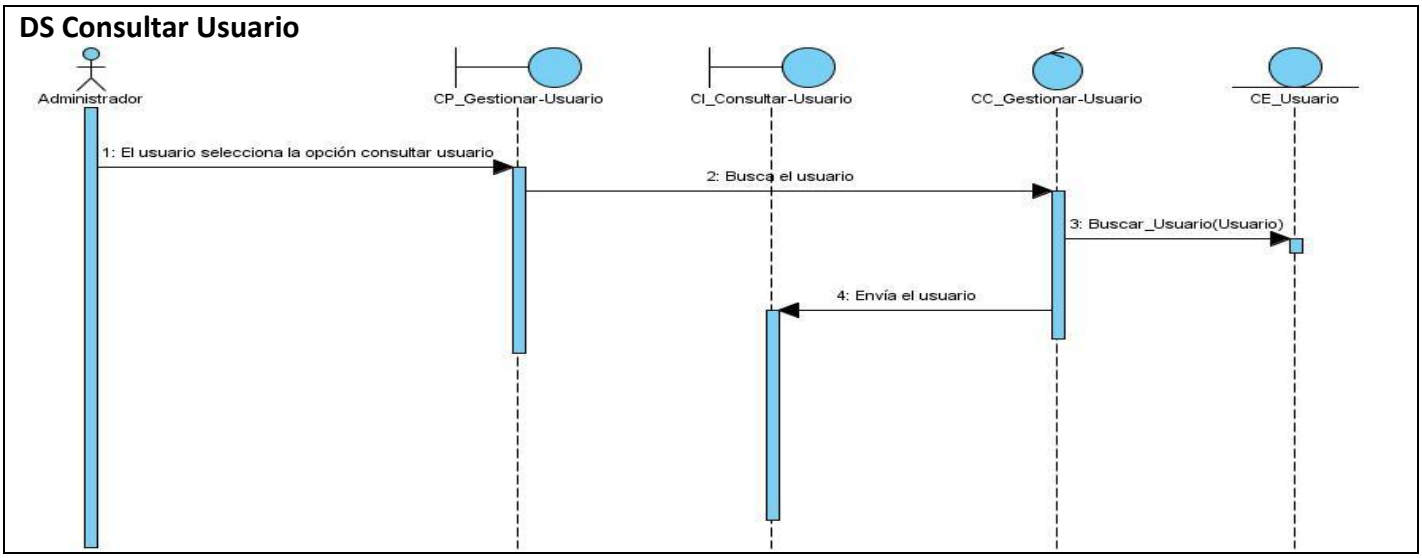


DS Modificar Usuario



DS Eliminar Usuario





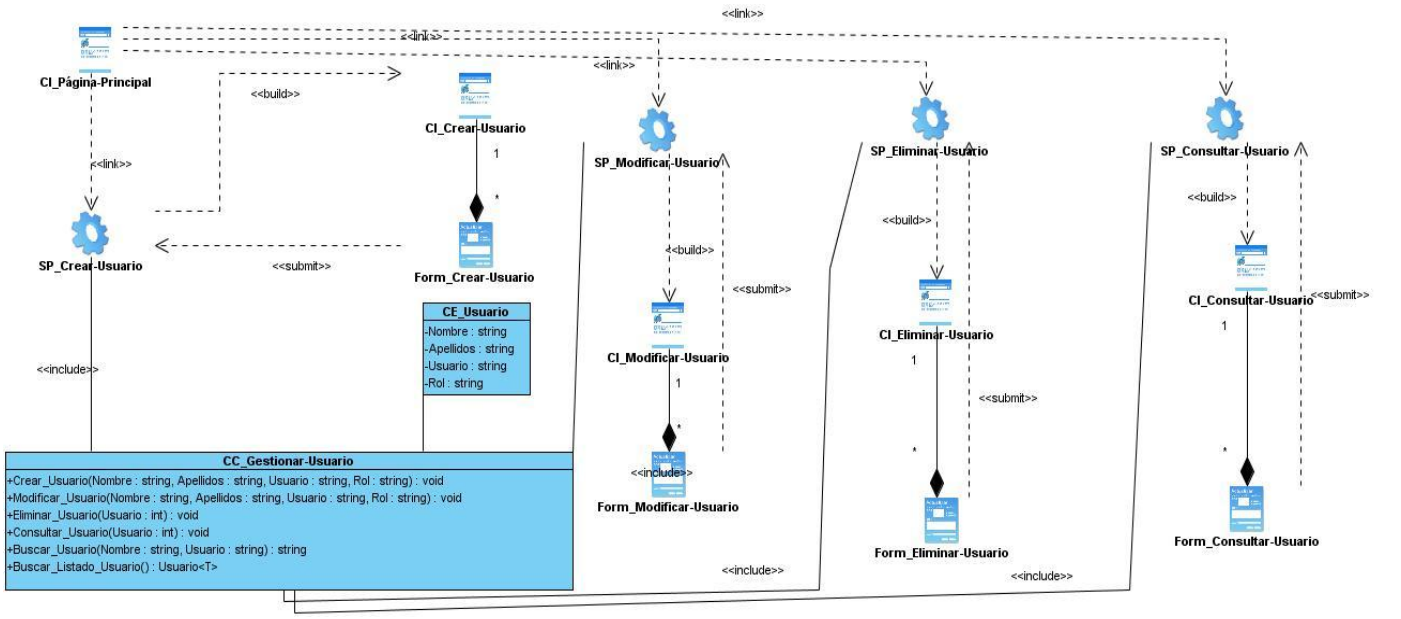
2.5 Modelo de Diseño.

El modelo de diseño, tiene como objetivo principal modelar, comprender y documentar todos los aspectos referentes a los requerimientos funcionales y no funcionales del sistema, además de lograr una entrada adecuada a las actividades de implementación. Para su elaboración se procede a identificar los subsistemas, paquetes y clases de diseño, además de distribuir el comportamiento de los casos de uso entre los elementos de diseño identificados.

2.5.1 Diagrama de Clases del Diseño.

Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación para satisfacer los detalles de la implementación. Estos se utilizan para modelar la vista de diseño estática de un sistema y contienen información como clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información sobre los tipos de los atributos, navegabilidad y dependencias.

DCD Gestionar Usuario



2.5.2 Patrones de Diseño.

Los patrones de diseño son soluciones que se pueden aplicar a problemas recurrentes en el diseño de software, los mismos cubren aspectos como la creación e iteración de objetos así como la comunicación entre ellos, además estos se hacen mas fácil el reutilizar componentes de software basándose en técnicas ya probadas una y otra vez en distintas aplicaciones, pudiendo realizar diseños de una manera más rápida y simplificada.
Los patrones de diseño se dividen en tres grupos: creacionales, estructurales, y de comportamiento. El objetivo principal de un patrón de diseño es la Protección Frente al Cambio. Esta protección se consigue "facilitando los cambios" y no "evitándolos" porque los cambios son inevitables.

2.5.2.1 Patrón Singleton.

El patrón singleton es quizás el patrón más conocido, y a la vez el más sencillo de diseñar. El contexto de este patrón es cuando se diseña una aplicación donde se tiene una clase, de la cual sólo puede existir una instancia. El problema está en controlar este aspecto, ya que por defecto, la programación orientada objetos da libertad para instanciar tantos objetos de una clase como se desee.
La solución que se aplicaría usando el patrón Singleton, es la de realizar unas modificaciones en la clase para bloquear el método constructor donde no se pueda crear mas de un objeto, y también crear un método alternativo para conseguir la instancia única que se desea. Detallando en forma de lista las modificaciones que hay que hacer en una clase para aplicar el patrón Singleton, quedaría así:

- ✓ **Convertir en privado** el método constructor.

- ✓ **Crear un atributo**, privado y estático, que almacenará la única instancia.
- ✓ **Crear un método**, público y estático, que devolverá la única instancia. Su función será:
- La **primera vez** que se llame a este método, creará una instancia, la almacenará en el atributo estático, y la devolverá.
- Las **siguiente veces** que se llame, devolverá la instancia almacenada.[Monge, 2010].

2.5.2.2 Patrón Bajo Acoplamiento.

El acoplamiento de una clase es el conjunto de dependencias que tiene con otras clases. Así, una clase A que deriva de B, tendrá un fuerte acoplamiento con ella. Y otra clase C, que utilice una clase D, tendrá también una relación de dependencia. Resulta evidente que, cuanto menor sea el acoplamiento entre clases, menor influencia tendrán los cambios. Así, un cambio en una clase A, que depende de las clases , C y D será mucho más complejo que si la misma clase A no dependiera de ninguna otra y, por lo tanto, el cambio sólo le afectara a ella.
Mantener Bajo el Acoplamiento entre Clases, más que un patrón que se pueda implementar, es un principio que servirá para elegir entre alternativas de diseño: un diseño menos acoplado siempre será mejor que uno más acoplado**,** porque en el primero será más fácil realizar cambios.

2.5.2.3 Patrón Abstract Factory.

Este patrón provee una interfaz para crear familias de objetos relacionados o dependientes entre ellos sin especificar una clase en concreto. Su objetivo es independizar un sistema de la manera en que se crean, componen y representan sus objetos. Estos adquiren mayor importancia cuando el sistema se apoya más en la composición de objetos que en la herencia. Encapsula el conocimiento acerca de qué clase concreta se instancia, encapsula el conocimiento acerca de cómo se crean las instancias y cómo se ensamblan y proporciona una gran flexibilidad acerca de qué se crea, quién lo crea, cómo se crea, y cuándo se crea.
Entre los problemas más comunes resueltos por el patrón Abstract Factory se encuentra:

- ✓ La creación de un objeto creando su tipo de manera dinámica de forma que no se vean atados a una implementación sino a una interfaz haciendo mas fácil los cambios futuros.
- ✓ Permite jugar con la manera de satisfacer las peticiones hacia una operación determinada, de esta manera es más sencillo cambiar la forma en que se atienden las peticiones desde una operación en concreto.
- ✓ Permite poder ser manejados entre distintas plataformas de hardware y software al poder ser capaces de realizar diferentes manejos de las peticiones que se basan en las plataformas.

Luego del estudio a diferetenes patrones de diseño, se decide utilizar los siguientes patrones: *Singleton* para crear una instancia en las Clases Controladoras, de dicha instancia se podrá acceder a todas las funcionalidades de las demás clases y *Bajo Acoplamiento* para evitar dependencia entre las clases existentes.

2.6 Conclusiones Parciales.

El desarrollo de la solución consistió en la realización de las fases de Modelado y Levantamiento de requisitos propuestas por RUP donde se obtuvieron los artefactos más importantes de estos flujos para el

Capítulo2: Solución propuesta. Análisis y diseño del sistema

Módulo Administración del Laboratorio Virtual de Criptografía. En el Levantamiento de requisitos se identificaron mediante el empleo de las técnicas de Ingeniería de Requisitos los requisitos no funcionales y funcionales del sistema a desarrollar y se estructuró el Diagrama de Caso de Uso del Sistema que engloba las funcionalidades que el sistema debe cumplir y a partir de las cuales se realizó el diseño del sistema. Además se realizaron los Diagramas de Clases de Análisis, Diagramas de Secuencias, Diagramas de Clases del Diseño y se construyeron los Prototipos de Interfaz.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

Introducción.

En este capítulo se realizará la validación de la solución propuesta, específicamente se validará la Especificación de requisitos, el Diagrama de Caso de Uso del Sistema y se validará el diseño, para esto se utilizarán técnicas de Ingeniería de Requisitos y métricas de calidad.

3.1 Validación de la Solución por Métricas.

Las métricas son el término que describen muchos y muy variados casos de medición. Siendo una medida estadística (no cuantitativa como en otras disciplinas) que se aplica a todos los aspectos de calidad de software, los cuales deben ser medidos desde diferentes puntos de vista como el análisis, construcción, funcional, documentación, métodos, proceso, usuario, entre otros. Actualmente en el proceso de desarrollo de software están presentes un conjunto de métricas, las cuales se utilizan para la validación de los requisitos identificados en la realización de un software, estas métricas permiten validar de una manera correcta que los requisitos identificados durante el proceso de desarrollo tienen la calidad requerida y cumplen con las normas internacionales. También existen métricas que permiten validar el modelo de casos de uso. (Piña, 2008)

Seguidamente se hará referencia a algunas de las métricas que fueron aplicadas a los requisitos de software y casos de uso.

3.1.1 Métrica de la Calidad de la Especificación.

La validación de requisitos se realiza aplicando la métrica de la calidad de la especificación, examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad. Para esto es necesario conocer el total de los requisitos **Rt** dado por:

Rt = Rf + Rnf
75 = 45 + 30 donde:
Rt: Total de requisitos.
Rf: Cantidad de requisitos funcionales.
Rnf: Cantidad de requisitos no funcionales.
Para determinar la especificidad (ausencia de ambigüedad) de los requisitos. Se explica una métrica basada en la consistencia de la interpretación de los revisores para cada requisito: Se calcula
Q1 para determinar la especificidad de los requisitos.
Q1 = Rui / Rt
0.94= 71/75 donde:
Rui: Número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.
Q1: Ausencia de ambigüedad.
Cuanto más cerca de 1 esté el valor de **Q1**, menor será la ambigüedad de la especificación.
El valor de **Q1** = 0.94, esto demuestra que los requisitos se encuentran con un alto nivel de especificidad.

La ***estabilidad*** fue determinada con la fórmula $E = \frac{Rt-Rm}{Rt}$ donde R_m son los requisitos modificados, que es equivalente al número de requisitos. Se considera como valor óptimo para esta métrica el valor más próximo a 1.

Se sustituyen los valores:

E = (Rt – Rm) / Rt
E = (75 – 4) / 75
E = 0,94

El resultado obtenido indica que los requisitos son estables, ya que 0,94 se considera un valor de estabilidad alto, basado en el siguiente rango:

Alta (0.90 <= **E** <= 1).
Media (0.80 <= **E** < 0.90).
Baja (0.7 <= **E** < 0.80).

Para la revisión de la métrica aplicada a los requisitos se escogió una representación de integrantes del grupo de trabajo de calidad de la facultad, puesto que por el rol que desempeñan poseen conocimiento y experiencia en el tema.

3.1.2 Métricas para Validar los Casos de Uso del Sistema.

En este acápite se aplican un conjunto de métricas orientadas a objetos para evaluar las siguientes propiedades de calidad: consistencia, correctitud, completitud y complejidad. Cada uno de estos factores tendrá asociada una o más métricas, que establecen una medida cuantitativa del grado en que los factores presentan una pobre calidad.

***Completitud*:** Grado en que se ha logrado detallar todos los casos de uso relevantes.

Consistencia: Grado en que los casos de uso del sistema describen las interacciones adecuadas entre el usuario y el sistema.

Correctitud: Grado en que las interacciones actor / sistema soportan adecuadamente el proceso del negocio.

Complejidad: Grado de claridad en la presentación de los elementos que describen el contexto y la claridad del sistema. Para conseguir una certera validación de los casos de uso del sistema se realizaron dos revisiones, a continuación se presentan los resultados obtenidos:

Primera Revisión:

Atributo	Factor	Métrica Asociada	Valor
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar/ modificar o consultar información?	Métrica 1: número de roles relevantes omitidos.	Número de roles relevantes omitidos: 0 Se presenta un 100%.
	Factor 2. ¿Se presenta una descripción	Métrica 2: número de casos de uso que no	Número de casos de uso que no tienen

	resumida de todos los casos de uso?	tienen descripción resumida.	descripción resumida: 0 Se presenta un 100 %.
	Factor 3. ¿Están definidos todos los requisitos que justifican toda la funcionalidad del caso de uso?	Métrica 3: número de requisitos omitidos por caso de uso. Métrica 4: número de casos de uso que tienen requisitos omitidos.	Número de requisitos omitidos por caso de uso: 4 Se presenta un 5.12%. Número de casos de uso que tienen requisitos omitidos: 4 Se presenta un 5.12%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en crítico, secundario y auxiliar, opcional?	Métrica 5: número de casos de uso que han sido clasificados.	Número de casos de uso que han sido clasificados: 12 Se presenta un 100%.
			Se presenta un 62.04%.
Consistencia	Factor 5. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	Métrica 6: número de casos de uso que tienen un nombre incorrecto.	Número de casos de uso que tienen un nombre incorrecto: 0 Se presenta un 100%.
	Factor 6. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos?	Métrica 7: grado de adecuación de la descripción del flujo de eventos para un caso de uso	La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción: 7 Se presenta un 8.97%.
	Factor 7. ¿La	Métrica 8: número de	Número de casos de

	descripción del flujo de eventos se inicia con la descripción de una acción externa originada por un actor o por una condición interna del sistema claramente identificable?	casos de uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema.	uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema: 0 Se presenta un 100%.
	Factor 8. ¿Existe una adecuada separación entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados?	Métrica 9: número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos.	Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos: 11 Se presenta un 91.66%.
			Se presenta un 75.15%.
Correctitud	Factor 9. ¿Representa el caso de uso requisitos comprensibles por el usuario?	Métrica 10: números de casos de uso en que los requisitos representados no son comprensibles por el usuario.	Números de casos de uso en que los requisitos representados no son comprensibles por el usuario: 1 Se presenta un 8.33%.
	Factor 10. ¿Las iteraciones definidas describen la funcionalidad requerida del sistema?	Métrica 11: número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema.	Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema: 0 Se presenta un 100%.
	Factor 11. ¿Se ajusta la representación del diagrama de caso de uso de acuerdo a lo normado en la metodología?	Métrica 12: grado en que se ajusta el diagrama del caso de uso a la metodología.	Grado en que se ajusta el diagrama del caso de uso a la metodología: 1 Se presenta un 8.33%.

	Factor 12. ¿Las interacciones definidas introducen mejores al proceso actual?	Métrica 13: número de casos de uso que deben ser modificados para mejorar el proceso actual.	Número de casos de uso que deben ser modificados para mejorar el proceso actual: 0 Se presenta un 100%.
			Se presenta un 54.16 %.
Complejidad	Factor 13. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	Métrica 14: número de elementos del diagrama que requieren reubicación.	Número de elementos del diagrama que requieren reubicación: 0 Se presenta un 100%.
			Se presenta un 100%.

Métricas para validar diagrama de caso de uso del sistema.

Segunda Revisión:

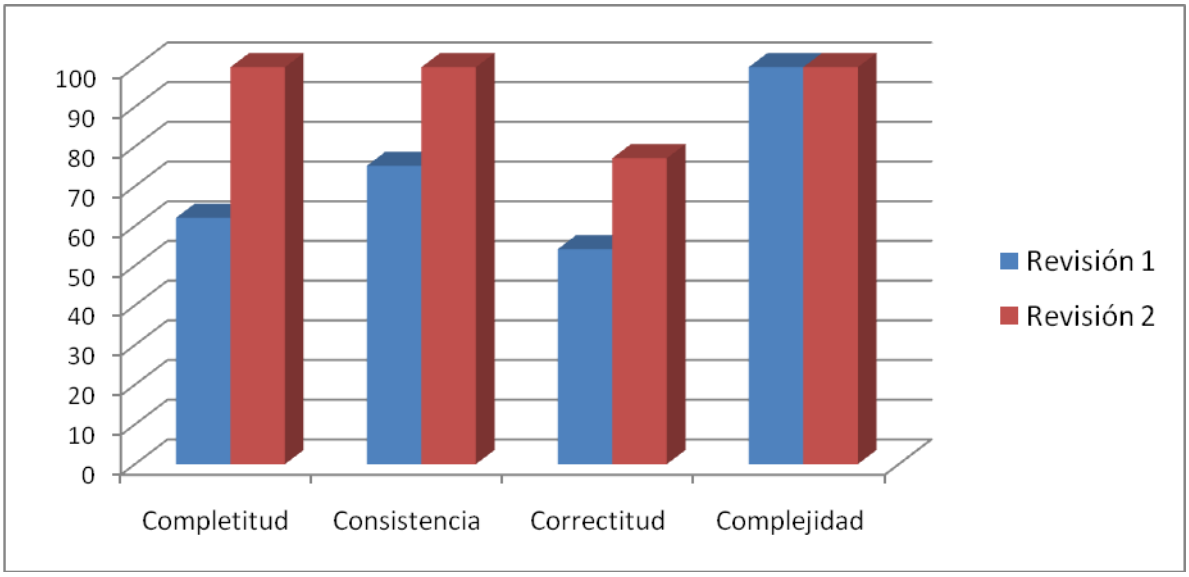
Atributo	**Factor**	**Métrica Asociada**	**Valor**
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar/ modificar o consultar información?	Métrica 1: número de roles relevantes omitidos.	Número de roles relevantes omitidos: 0 Se presenta un 100%.
	Factor 2. ¿Se presenta una descripción resumida de todos los casos de uso?	Métrica 2: número de casos de uso que no tienen descripción resumida.	Número de casos de uso que no tienen descripción resumida: 0 Se presenta un 100 %.
	Factor 3. ¿Están definidos todos los requisitos que justifican toda la funcionalidad del caso de uso?	Métrica 3: número de requisitos omitidos por caso de uso. Métrica 4: número de casos de uso que tienen	Número de requisitos omitidos por caso de uso: 0 Se presenta un 100%.

		requisitos omitidos.	Número de casos de uso que tienen requisitos omitidos: 0 Se presenta un 100%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en crítico, secundario y auxiliar, opcional?	Métrica 5: número de casos de uso que han sido clasificados.	Número de casos de uso que han sido clasificados: 0 Se presenta un 100%.
			Se presenta un 100%.
Consistencia	Factor 5. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	Métrica 6: número de casos de uso que tienen un nombre incorrecto.	Número de casos de uso que tienen un nombre incorrecto: 0 Se presenta un 100%.
	Factor 6. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos?	Métrica 7: grado de adecuación de la descripción del flujo de eventos para un caso de uso	La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción: 0 Se presenta un 100%.
	Factor 7. ¿La descripción del flujo de eventos se inicia con la descripción de una acción externa originada por un actor o por una condición interna del sistema claramente identificable?	Métrica 8: número de casos de uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema.	Número de casos de uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema: 0 Se presenta un 100%.
	Factor 8. ¿Existe una adecuada separación entre el flujo básico de eventos y los flujos alternos y/o flujos	Métrica 9: número de casos de uso complejos que no tienen separación del flujo básico y de flujos	Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos: 0

	subordinados?	alternos.	Se presenta un 100%.
			Se presenta un 100%.
Correctitud	Factor 9. ¿Representa el caso de uso requisitos comprensibles por el usuario?	Métrica 10: números de casos de uso en que los requisitos representados no son comprensibles por el usuario.	Números de casos de uso en que los requisitos representados no son comprensibles por el usuario: 0 Se presenta un 100%.
	Factor 10. ¿Las iteraciones definidas describen la funcionalidad requerida del sistema?	Métrica 11: número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema.	Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema: 0 Se presenta un 100%.
	Factor 11. ¿Se ajusta la representación del diagrama de caso de uso de acuerdo a lo normado en la metodología?	Métrica 12: grado en que se ajusta el diagrama del caso de uso a la metodología.	Grado en que se ajusta el diagrama del caso de uso a la metodología: 1 Se presenta un 8.33%.
	Factor 12. ¿Las interacciones definidas introducen mejores al proceso actual?	Métrica 13: número de casos de uso que deben ser modificados para mejorar el proceso actual.	Número de casos de uso que deben ser modificados para mejorar el proceso actual: 0 Se presenta un 100%.
			Se presenta un 77.08%.
Complejidad	Factor 13. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que	Métrica 14: número de elementos del diagrama que requieren reubicación.	Número de elementos del diagrama que requieren reubicación: 0 Se presenta un 100%.

	facilitan su interpretación?		
			Se presenta un 100%.

Gráfica de Factores de Métrica



Para clasificar los resultados obtenidos se establecieron las siguientes reglas:

Alto (90% <= **E** <= 100%).
Medio (80% <= **E** < 90%).
Bajo (70% <= **E** < 80%).

El resultado obtenido en la primera iteración, demostró un bajo grado de funcionalidad del DCUS con un valor de **72,83%**, para una contribución independiente por atributo a la calidad total de 62.04% en Completitud, 75.15% de Consistencia, 54,16% de Correctitud y 100% de Complejidad.
En una segunda iteración, se obtuvieron los valores de Completitud 100%, Consistencia 100%, Correctitud 77.08% y Complejidad 100%, para un porciento de funcionalidad del DCUS de **94,27**%. Estos resultados reflejan una mejora en la calidad del artefacto luego de corregir las no conformidades detectadas en la iteración anterior y se considera que el mismo posee un alto grado de funcionalidad.

3.2 Tamaño Operacional de Clase (TOC).

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:
Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

Atributos de calidad evaluados por la métrica TOC.

Atributo de Calidad.	Modo en que lo afecta.
Responsabilidad.	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de Implementación.	Un aumento del TOC implica un aumento en la complejidad de implementación de la clase.
Reutilización.	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

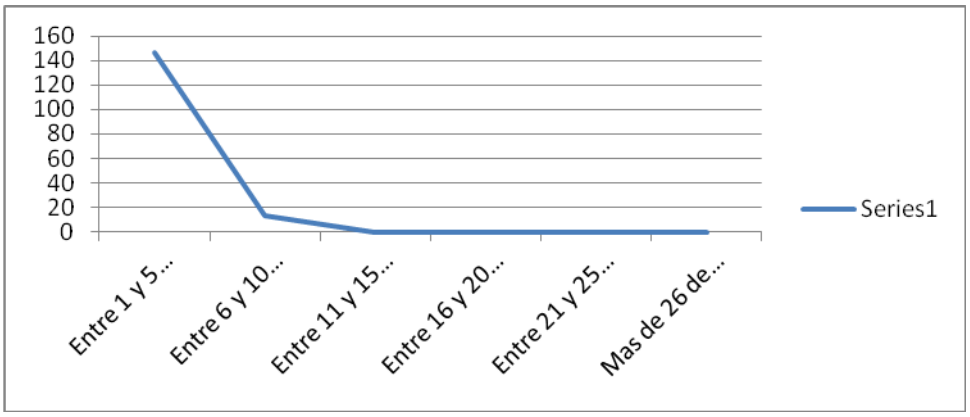
Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Criterios de evaluación para la métrica TOC.

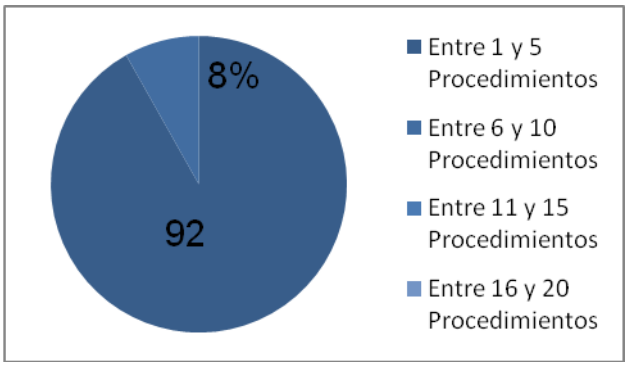
Atributo	Categoría	Criterio
Responsabilidad.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio
Complejidad de Implementación.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio
Reutilización.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio

Total de Clases: 159
Procedimientos: 278
Promedio de Procedimientos: 1.74

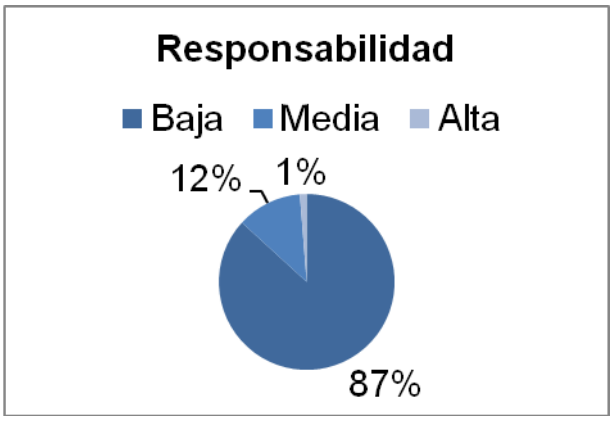
Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.



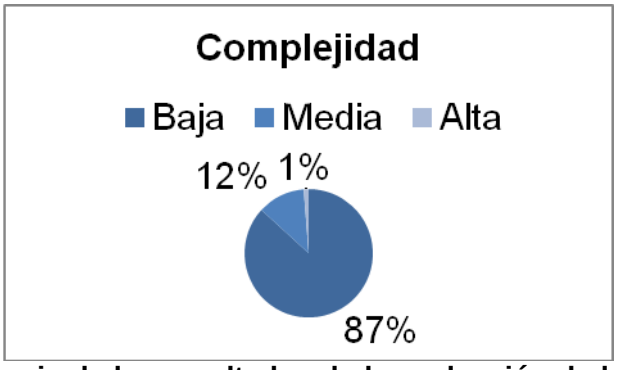
Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.



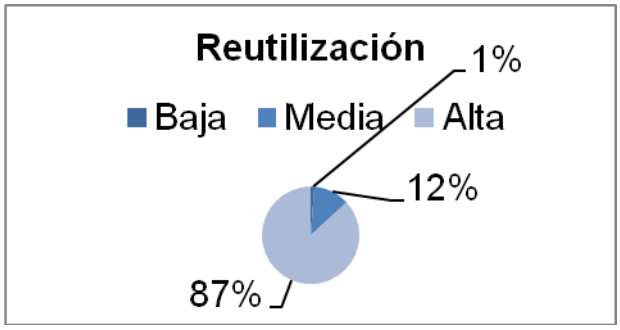
Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.



Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.



Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.



Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para el módulo de Administración del Laboratorio Virtual de Criptografía está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (87%) posee menos cantidad de operaciones que la media registrada en las mediciones. Los atributos de calidad se encuentran en un nivel medio satisfactorio en el 87% de las clases; de manera que se puede observar cómo se fomenta la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

3.3 Relaciones entre clases (RC).

Con la presente métrica se evalúan los siguientes atributos de calidad:

Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, modulo, clase, conjunto de clases, etc.) diseñado.

Atributos de calidad evaluados por la métrica RC.

Atributo de Calidad.	Modo en que lo afecta.
Responsabilidad.	Un aumento del RC implica un aumento de la responsabilidad asignada a la clase.
Complejidad del mantenimiento.	Un aumento del RC implica un aumento en la complejidad del mantenimiento de la clase.
Reutilización.	Un aumento del RC implica una disminución del grado de reutilización de la clase.
Cantidad de pruebas.	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

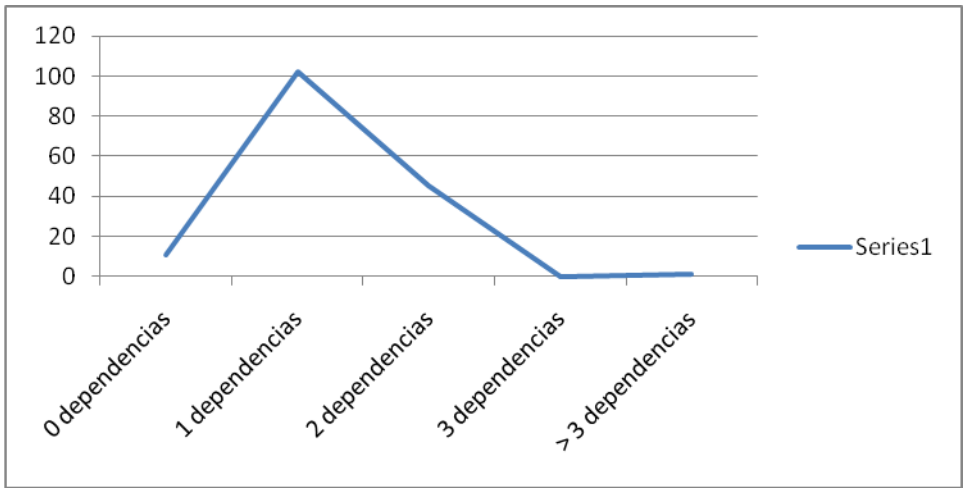
Criterios de evaluación de la métrica RC.

Atributo.	Categoría.	Criterio.
Acoplamiento.	Ninguno.	0
	Bajo.	1
	Medio.	2
	Alto.	>2
Complejidad de mantenimiento.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio
Reutilización.	Baja.	>2*Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	<=Promedio
Cantidad de pruebas.	Baja.	<=Promedio

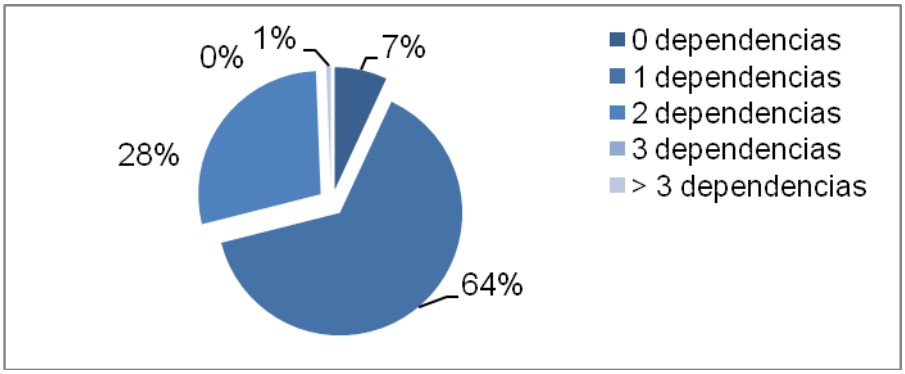
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio

Total de clases: 159
Cantidad de Relaciones de uso: 236
Promedio de Asociaciones de Uso: 1.48

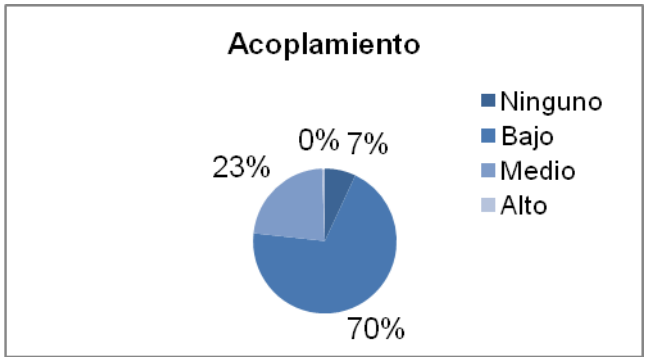
Gráfica de los resultados de la evaluación de la métrica RC agrupados por la tendencia de los Valores.



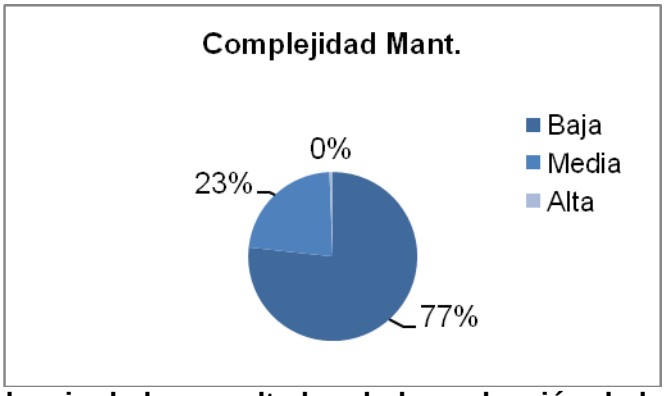
Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.



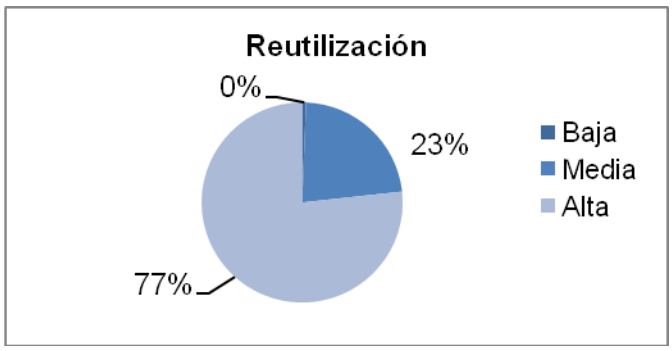
Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.



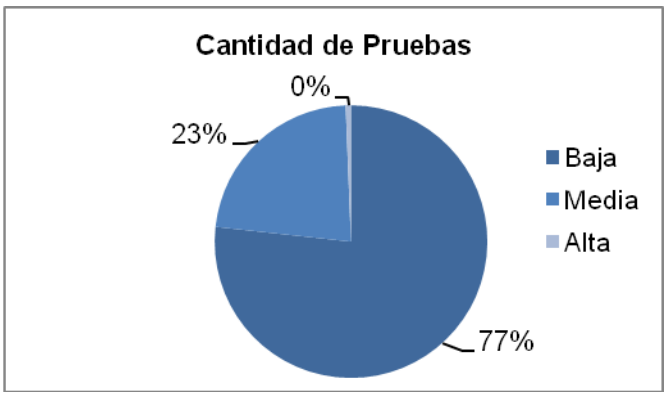
Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.



Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.



Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.



Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el módulo Administración del Laboratorio Virtual de criptografía entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (99%) poseen menos de 3 dependencias respecto a otras. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 70% de las clases el grado de dependencia o acoplamiento es mínimo, la Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 77%, 77% y 77% de las clases respectivamente.

3.4 Conclusiones Parciales.

En este capítulo se realizó la etapa de validación de requisitos donde se emplearon métricas para la validación, estas fueron: Métrica de la Calidad de la Especificación y Métricas para Validar los Casos de Uso del Sistema. Luego de validados los resultados se garantiza que la solución posea la calidad requerida y cumpla con las necesidades del cliente.

Conclusiones Generales

CONCLUSIONES GENERALES

La realización de la investigación permite arribar a las siguientes conclusiones:

- El estudio realizado sobre las metodologías de desarrollo de software, lenguajes de modelado y herramientas CASE posibilitó seleccionar las variantes más óptimas para realizar el Análisis y Diseño del Módulo Administración para el Laboratorio Virtual de Criptografía.

- Mediante el desarrollo de las técnicas y etapas más importantes de la Ingeniería de Requisitos se logró obtener los requisitos del sistema con la calidad requerida.

- La validación de los resultados obtenidos mediante métricas de calidad y técnicas de validación de requisitos permitió elevar la calidad del Diagrama de Caso de Uso del Sistema y la especificación de requisitos.

- Se cumplió el objetivo general trazado para este Trabajo de Diploma: realizar el análisis y diseño del módulo administración para el laboratorio virtual de criptografía que apoye la visualización de los contenidos del tema Criptografía en la asignatura Seguridad Informática.

Recomendaciones

RECOMENDACIONES

Al concluir la presente tesis se realizan una serie de recomendaciones que podrán tenerse en cuenta para el desarrollo futuro del sistema:

- Se propone que se le de seguimiento a este trabajo para lograr un producto de mayor calidad.
- Se recomienda realizar la implementación y pruebas de la propuesta que se presenta en este trabajo, con el fin de obtener al menos una versión del producto.
- Proseguir con el estudio realizado con el propósito de añadir nuevas funcionalidades al sistema.

Bibliografía

BIBLIOGRAFÍA

Barriento, Manuel Sánchez. 2008. BPMN desventajas. *BPMN desventajas.* [En línea] 2 de noviembre de 2008. [Citado el: 20 de noviembre de 2010.] http://www.aprendergratis.com/stag/bpmn-desventajas.html.

Estrada, Julián Monge Nájera y Víctor Hugo Méndez. 2007. Ventajas y desventajas de usar laboratorios virtuales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración. *Ventajas y desventajas de usar laboratorios virtuales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración.* [En línea] 2007. [Citado el: 20 de noviembre de 2010.] http://www.latindex.ucr.ac.cr/edu31-1/edu-31-1-05.pdf.

Estrada, Yelena Hernadez. 2009. Análisis del Módulo Proceso Confiscatorio de Bienes del proyecto Sistema Gestión Fiscal. Ciudad de la Habana: s.n., 2009.

Giraldo, L. &. (2005). Herramientas de Desarrollo de Ingeniería de SW para Linux.
Herías, Francisco Andrés Candelas. 2003. Propuesta de Portal de la Red de. *Propuesta de Portal de la Red de Laboratorios Virtuales y Remotos de CEA.* [En línea] 27 de noviembre de 2003. [Citado el: 16 de enero de 2011.] http://www.disc.ua.es/docenweb/Docs/PropuestaDePortal.pdf.

Herreros, L. Rosado y J. R. 2005. Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física. *Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física.* [En línea] 2005. [Citado el: 21 de noviembre de 2010.] http://www.formatex.org/micte2005/286.pdf.

James P. Vary. [En línea] [Citado el: 10 de febrero de 2011.] http://unesdoc.unesco.org/images/0011/001191/119102s.pdf.

Larman, C. (1999).*UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México: Primera Edición.

Orallo, Enrique Hernández. 2007. El lenguaje Unificado de Modelado (UML). *El lenguaje Unificado de Modelado (UML).* [En línea] 2007. [Citado el: 25 de febrero de 2011.]

Pressman, R. S. 2005. *Ingeniería de software. Un enfoque práctico. Parte ll.* Ciudad de la Habana: Félix Varela.

Pressman, R. *Ingeniería del Software. Un enfoque práctico.* La Habana: Félix Varela, 2005
Sanchez, María A. Mendoza. 2004. Metodologías De Desarrollo De Software. *Metodologías De Desarrollo De Software.* [En línea] 7 de junio de 2004. [Citado el: 26 de febrero de 2011.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.htm.

Bibliografía

Vary, James P. 2000. Informe de la reunión de expertos. *Informe de la reunión de expertos.* [En línea] 2000. [Citado el: 10 de enero de 2011.] http://unesdoc.unesco.org/images/0011/001191/119102s.pdf.

Ventajas y desventajas de usar laboratorios Virtuales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración. . *Ventajas y desventajas de usar laboratorios Virtuales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración.* [En línea] 2007. [Citado el: 26 de noviembre de 2010.] http://www.latindex.ucr.ac.cr/edu31-1/edu-31-1-05.pdf.

Visual Paradigm for UML (ME). [En línea] 5 de marzo de 2007. [Citado el: 18 de marzo de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

Tecnología y Synergix. [En línea]. [Citado el 15 de enero de 2011.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

Duram, Amador. 2000. Un entorno metodológico para la ingeniera de requisitos para sistemas de información. 2000.

Escalona, Maria José y Koch, Nora. 2002. Ingeniería de Requisitos en Aplicaciones para la Web. Un estudio comparativo. España: s.n., 2002.

Pressman, Roger S. 2002. Ingeniería de Software (Un enfoque práctico). 2002.

Torres, Jose Luis. 2008. Especificación de requisitos en Ingeniería de Software. [En línea] 2008. http://www.uag.mx/ieee/contsep01/requerimientos.htm.

Díaz Vallejo, L.2009. "Propuesta de arquitectura para el sistema de gestión automatizado de recursos humanos GESTAPro.," Universidad de las Ciencias Informáticas, 2009.

Gracia, A. 2009. "Análisis y diseño de un sistema automatizado para el control de los recursos humanos en los polos productivos de la facultad 9.," Universidad de las Ciencias Informáticas, 2009.

Torres Peña, D. 2009. "Análisis y Diseño del Sistema para la Gestión del Despliegue de los proyectos productivos de la Facultad 4.," Universidad de las Ciencias Informáticas, 2009.

Monge, Ian (2010). Patrones de diseño: patrón Singleton. [En línea], Otro Blog Más.[Citado el 25 de abril de 2011] en: http://otroblogmas.com/patrones-de-diseno-patron-singleton/

Apoyo a la Cultura Libre (2008). Bajo Acoplamiento. [En línea]. [Citado el 25 de abril de 2011] en: http://migueljaque.com/index.php/patrones/grasp/38-grasp/72-bajoacoplamiento

Monge, Ian. (2010). Patrones de diseño: patrón Observador.[En línea],Otro Blog Más.[Citado el 25 de abril de 2011] en: http://otroblogmas.com/patrones-de-diseno-patron-observador/

Bibliografía

High Scalability (2010). Patrones de diseño: Introducción y el patrón Abstract Factory.[En línea].[Citado el 25 de abril de 2011] en: http://highscalability.wordpress.com/2010/03/08/patrones-de-diseno-introduccion-y-el-patron-abstract-factory/

Gómez, Julián. (2011). Diseña prototipos navegables de páginas web. Axure RP Pro 5.6.0.2089.[Citado el 3 de abril de 2011] en: http://axure-rp.softonic.com/

Piña., Lianyi Ramos León y Yanelis Pulido. 2008. Análisis de los módulos Planificación de Disco y Administración de Memoria de un Laboratorio Virtual de apoyo a la asignatura de Sistemas Operativos. Habana, Cuba: s.n., 2008.

Slideshare. Clase Flujo De Análisis. [En línea] [Citado el: 17 de Febrero de 2011.] http://www.slideshare.net/juliopari/13-clase-flujo-de-analisis.

Scribd. Diagramas de Colaboración. [En línea] [Citado el: 18 de Febrero de 2011.] http://www.scribd.com/doc/11802367/diagramas-de-colaboracion.

Clikear.com. Manual. [En línea] [Citado el: 18 de Febrero de 2011.] http://www.clikear.com/manuales/uml/diagramasinteraccion.aspx

Glosario de Términos

GLOSARIO DE TÉRMINOS

Laboratorio Virtual: No es más que un entorno de experimentación para realizar el conjunto de prácticas de laboratorio que son necesarias para poder superar las distintas asignaturas.
Seguridad Informática: es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático.
Tecnología: es el conjunto de conocimientos técnicos, ordenados científicamente, que permiten diseñar y crear bienes y servicios que facilitan la adaptación al medio ambiente y satisfacer tanto las necesidades esenciales como los deseos de las personas.
Actores: Roles pertenecientes a los usuarios, agrupados según sus iteraciones con las funcionalidades del sistema.
Caso de uso (CU): Representación de la agrupación de funcionalidades comunes. Representan un conjunto de iteraciones entre el sistema y sus actores.
Reporte: Se refiere a la información lógica, relevante, y organizada, obtenida a partir de la recuperación de datos incluidos en el sistema.
Hardware: La parte física de una computadora y más ampliamente de cualquier dispositivo electrónico. Se refiere a todos los componentes físicos.
Software: conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.
Modelado: Modelar es desarrollar una descripción lo más exacta posible de un sistema y de las actividades llevadas a cabo en él.
Patrón: solución común a un problema común de un determinado contexto.
RUP (Rational Unified Process)**:** Proceso unificado de desarrollo de software.
UML (Lenguaje Unificado de Modelado): Es un lenguaje de modelado visual para especificar, visualizar, construir y documentar artefactos de un sistema de software.
Diagrama: Presentación gráfica de un conjunto de elementos y sus relaciones.