Universidad de las Ciencias Informáticas

Facultad 3

Desarrollo de los módulos Administración de memoria y Comunicación entre procesos del subsistema de autoaprendizaje para el Laboratorio Virtual de Sistemas Operativos.

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

Autor(es): Liennys María González Torres.

Lisandra Ramos Acosta.

Tutor: Ing. Carlos Y. Hidalgo García.

La Habana, Cuba

2012

Declaración de Autoría

Se declara ser las únicas autoras de la presente tesis y se reconoce a la Universidad de las Ciencias Informáticas, los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.

_____ _____

Liennys M. Torres González — Firma de la Autora

Lisandra Ramos Acosta — Firma de la Autora

_____

Ing. Carlos Y. Hidalgo

Firma del Tutor

Dedicatoria

Liennys M dedica a:

Esta tesis está dedicada a las personas más importantes de mi vida: mi padre ***Ariel González Montero,*** *mi hermana* ***Liannys Laura González Torres*** *y mi madre* ***Edelma Torres Espinoza****, quienes han sido mis alegrías y mis lágrimas. Por ustedes lo he hecho todo en mi vida y así será eternamente. Gracias por la educación que me han dado. Los amo.*

Lisandra dedica a:

Esta tesis está dedicada a la razón de mi existencia: mi madre ***Camila Acosta Leal*** *y mi hermano* ***Yoandys Ramos Acosta****. A ustedes, mis amores, todo lo que he hecho y haré en la vida va dedicado, mil gracias por ser fieles amigos, sabios maestros y los mejores guías del universo.*

Especialmente y no puede faltar, dedico además esta tesis a quien siempre estuvo ahí presente en todo momento como motor impulsor, para que hoy pudiera decir soy Ingeniera, a la memoria de mi abuela ***Julia María Leal Licourt****.*

Agradecimientos

De Liennys:

A mi padre, que ha sido mi guía y ejemplo a seguir en la vida, de quien estoy muy orgullosa y siempre lo estaré .Gracias por ser el padre más maravilloso del mundo y a quien amo con locura.

A mi hermana, que ha sido el motor de mi corazón en todo momento de mi vida, solo de pensar en ella tengo la fuerza necesaria para seguir adelante, te amo mi ángel.

A mi mamita linda, que ha sido mi amiga y confidente en todo momento de mi vida, la que me ha enseñado el amor más grande y lindo del mundo y de quien estoy muy orgullosa.

A mi abuela, que nunca ha dejado de preocuparse por mí pese a todas sus enfermedades. Ella ha sido mi inspiración para seguir adelante; para un día, al fin, poder tenerla a mi lado.

A mis primas y mi tía, quienes han sido un ejemplo a seguir. Gracias por brindarme su apoyo y ayuda en los momentos que lo he necesitado, nunca las olvidaré.

A mi familia en general, que de una forma u otra han influido en mi formación para ser la persona que soy hoy, gracias a todos, los quiero.

A mi novio Julio Antonio Villaverde Martínez, quien ha sido mi apoyo en todo momento, primero como amigo y de quien no tengo una queja. Te agradezco, mi amor, por todo el tiempo que me has dedicado; te amo mucho.

A Juan Pablo, agradecerle el hecho de estar hoy aquí discutiendo mi tesis; sin ti nunca lo hubiese logrado. Gracias por esos 5 años en los que nunca me dejaste sola, nunca te olvidaré.

A Alejandro Castillo, quien supo soportar mis malacrianzas durante todo un año que estuve en Venezuela. Gracias por ser como un padre para mí, te quiero mucho.

A mi mejor amiga Lisbet Moreno, quien ha estado a mi lado en los momentos más difíciles de mi vida y nunca me ha dejado sola, nunca te olvidaré te quiero demasiado.

A mi mejor amigo Pedro (mi negro), gracias por todos estos años a mi lado que ya suman 13 años junticos y ojalá pasen miles más. Te quiero demasiado, aunque no te lo diga todos los días.

A mis amigos Arturo, Ivian , Raquel, Aliesky, Héctor y el Yordy, agradecerles todo el apoyo que me brindaron durante toda mi carrera. Gracias por estar a mi lado los quiero mucho y los extraño.

A mi compañera de tesis con quien he pasado uno de los momentos más estresantes de mi vida, te quiero mucho Lisi. Gracias por tu apoyo en todo este tiempo.

A José Rolando, Edel, Yunier René, Yoan, Oscar e Ibrahim, quienes de una forma u otra me ayudaron durante todo este tiempo.

De Lisandra:

Toda obra humana por muy humilde que sea necesita el curso de varios, jamás el hombre alcanzó la meta por sí solo, siempre fue de alguna manera asistido en el empeño, y si algo lo enaltece es la gratitud.

Por eso quiero agradecerles a todas aquellas personas que hicieron posible la realización de este trabajo.

En primer lugar agradecerle a Julio Antonio Villaverde, quien siempre estuvo presente ayudándome durante el desarrollo de la tesis. Sin su esfuerzo hoy no me estuviera graduando. De verdad muchas gracias.

A mi mamá, mi hermano y mi tía Chely, gracias por existir sin su apoyo, comprensión y confianza que depositaron en mí no hubiera podido triunfar. Gracias por ser esos fieles guías en mi vida.

A mi tío Israel, miles de gracias pues siempre estuviste ahí presente como el padre que no tengo, te agradezco de todo corazón tu apoyo.

A mis adorados tormentos, mi viejito Pascual y mi amigo Yudel, quienes siempre se preocuparon por mí y por mi familia. Les doy miles de gracias; sin ustedes no hubiera llegado a convertirme en una profesional. Les estoy y estaré eternamente agradecida.

A mi hermanita Leydis Bárbara, gracias mamita por estar siempre desde pequeña ayudándome y apoyándome en todo.

A mis amigas Mairelis Macías Medina, Yunet Estrada Grandales y Elieta Medero, por ser las persona más sinceras y amables que he conocido en la vida. Gracias de verdad.

A mi compañera de tesis Liennys María, por la paciencia y dedicación que me brindaste en todos los momentos cuando necesité que alguien me ayudara y estuviera ahí presente. Gracias, Mita, fue lo mejor que nos pudo pasar.

A Carlos Jorge González, gracias te doy Tito por soportarme en los momentos en los que más necesité de una mano firme para apoyarme, por escucharme y dedicarme unos minutos en tu vida. Gracias amor por saber que puedo contar contigo.

A mis compañeros de aula que aunque todos son ingenieros y muchos no están aquí tengo que agradecerle pues recorrí con ustedes caminos que nunca pensé ni ver, pase momentos tanto de tristeza como de alegrías, les agradezco por todas estas cosas, por citar, aunque todos lo merecen pero estas personas siempre estuvieron cuando más los necesitaba: Pedro Manuel García, Yusmila (La flaca modelo), Dailin (Dayita), Semidaris (BodyShul), Yanetsi (La cachetona) en fin a todos miles de gracias.

En especial quisiera agradecerle a ese pedacito de amor Jorge Yuniel Jorrin por todos los momentos en que siempre estuvo presente como el buen amigo que me guió y ayudó. Gracias por brindarme tus conocimientos.

Por último agradecerle a mi tutor Carlos Yasmany por ayudarme siempre que lo necesite.

Resumen

Con el desarrollo de las Tecnologías de la Informática y las Comunicaciones (TIC) el mundo que se conocía se ha visto en la necesidad de cambiar en todos sus aspectos. El aprendizaje no se ha visto exento de dicha transformación. Así, han ido apareciendo nuevos conceptos relacionados con el **autoaprendizaje;** uno de ellos se refiere al aprendizaje a distancia, solamente supervisado por un conjunto de expertos, que no tienen por qué coincidir en espacio y/o tiempo con los educandos. Para apoyar esta enseñanza han surgido nuevas herramientas, como son los Laboratorios Virtuales, de ahí la importancia de los mismos. En consonancia con estas nuevas tendencias es que se desarrolla un Laboratorio Virtual para la asignatura de Sistemas Operativos en la Universidad de las Ciencias Informáticas, la cual le permitirá a los educandos realizar el estudio por sí mismos de los contenidos Administración de memoria y Comunicación entre procesos. El laboratorio que se ha desarrollado consta con funcionalidades tales como: añadir y modificar contenido de cualquier tema de la asignatura, además les provee a los estudiantes una herramienta de apoyo para una mejor comprensión de la asignatura Sistemas Operativos. Por su concepción, el laboratorio en sí es una herramienta de gran utilidad para los estudiantes de las universidades de cualquier parte del país.

PALABRAS CLAVES

Administración de memoria, Comunicación entre procesos, Laboratorio Virtual y Sistemas de autoaprendizaje.

Índice

Índice de Figuras

Introducción

El año 2002 fue un año crucial para el sector de la informática en Cuba. Es en este año cuando se funda la Universidad de las Ciencias Informáticas (UCI), a partir de una idea del compañero Fidel Castro Ruz, con el propósito de lograr la informatización de la sociedad cubana a través de un ejército de profesionales altamente capacitados en el campo de la ingeniería informática.

Con el pasar de los años, la joven universidad ha transitado por disímiles modificaciones en el plan de estudio de la carrera, orientadas a elevar la formación de sus educandos, en aras de lograr un profesional cada vez más competente ante la necesidad del país de insertarse en el mercado mundial desarrollando sistemas informáticos. En tal sentido, una de las preocupaciones esenciales ha sido la vinculación e incorporación de elementos prácticos en la enseñanza teórica, con el fin de fortalecer la preparación del ingeniero informático. Es entonces cuando el modelo de enseñanza tradicional se ve obligado a sufrir cambios radicales, y el autoaprendizaje tiene una importancia capital. A partir de este momento, el estudiante es el principal responsable de su preparación, la que completará con su vinculación en el sector productivo de la universidad a tiempo completo. Unos de los cambios realizados a partir de la implementación de este modelo es la división del plan de estudio en dos ciclos: un ciclo básico y otro profesional. Este modelo propicia la posibilidad de perfeccionar el plan de estudio y el desarrollo de las actividades que los educandos realizan con el objetivo de lograr una formación integral a través de la incorporación a proyectos de desarrollo de software.

Actualmente, en la UCI se dispone de un conjunto de herramientas y materiales de apoyo que contribuyen a la formación profesional de sus estudiantes, entre los que se encuentran los audiovisuales, como los canales de TV, los cuales transmiten videoconferencias que tributan al plan de estudio; los sitios y portales web como Inter-Nos ,estos incorporan nuevas vías para la preparación integral del estudiantado y el Entorno virtual de aprendizaje (EVA) , plataforma de autoaprendizaje accesible desde todas las áreas de la universidad, como herramienta fundamental de apoyo al proceso docente-educativo en las distintas materias.

Con la puesta en marcha del nuevo modelo, el EVA ha jugado un papel fundamental, pues ha permitido utilizar otros recursos de apoyo como son los Foros, las Wikis, materiales didácticos y entornos de simulación, entre los que se encuentran los Laboratorios Virtuales. Este último tiene gran importancia ya

que se ha convertido en una excelente herramienta, a partir de la integración directa de los conocimientos teóricos con la práctica, toda vez que ayuda al estudiante y al profesor en la consolidación del saber.

Así mismo, en el transcurso de la carrera varias asignaturas técnicas que forman parte de la base de conocimientos de los futuros ingenieros, han sufrido cambios. Una de ellas es Sistemas Operativos (SO), que se cursa en el tercer año de la carrera de Ingeniería en Ciencias Informáticas.

Esta asignatura no ha estado exenta de las modificaciones hechas en el plan de estudio, se ha visto inmersa en un conjunto de cambios que responden al nuevo modelo de formación que se desarrolla en la universidad, cambios que han tenido como principales preocupaciones que los estudiantes de hoy no saben cómo aplicar en la práctica los conocimientos teóricos adquiridos en clase, además se le suma la falta de motivación de los mismos, agravada por la inexperiencia de algunos de los profesores del claustro, provocando que los estudiantes olviden fácilmente los temas tratados en clase, al no contar con un entorno demostrativo que les facilite la fijación de los principales conceptos de la asignatura en cuestión. También la carencia de materiales didácticos en la asignatura, conlleva a que los estudiantes no cuenten con las herramientas necesarias para la comprensión de muchos de los temas tratados, los cuales pueden llegar a ser muy difíciles de entender. Dentro de los que se encuentran, por ser unos de los más complejos, Administración de memoria (AM) y Comunicación entre procesos (CP).

Debido a la complejidad y el carácter práctico de los temas antes mencionado, y que sus actividades se realizan mayormente de forma presencial, para lograr una mejor integración entre formación, producción e investigación, se ha puesto en marcha la creación de nuevos medios didácticos que ayuden y motiven al estudiante a su autopreparación.

Considerando los aspectos planteados durante la investigación, en los cierres semestrales de la asignatura, se detectaron desde la perspectiva informática varios aspectos que obstaculizan su adecuado desempeño, esto se fundamenta en lo siguiente:

- ✓ La falta de motivación en los estudiantes debido a la complejidad de los temas.
- ✓ El trabajo en el EVA aún no es suficiente, pues no existe un seguimiento personalizado de los estudiantes por los profesores de la asignatura a través de la plataforma, quienes solo la visitan para ver la documentación de la asignatura colocada en la misma.
- ✓ La carencia de materiales didácticos en la asignatura trae consigo que los estudiantes no cuenten con los medios necesarios para aumentar la comprensión del contenido de los temas vistos en el

aula, así como tampoco ponen en práctica los contenidos teóricos que se imparten por parte de los profesores.

Debido a las insuficiencias anteriormente planteadas, en el colectivo de profesores se decidió llevar a cabo el desarrollo de un laboratorio virtual para la enseñanza de la asignatura, con el objetivo de darle la posibilidad al estudiante de contar con una herramienta que facilite la comprensión de los contenidos impartidos en el aula además de contribuir a su autopreparación. Para el desarrollo del mismo se establecieron varios subsistemas, dentro de ellos está el subsistema de autoaprendizaje donde se concentra la mayor interacción del estudiante dentro del laboratorio virtual.

Como consecuencia de lo anteriormente expuesto se formula el siguiente **problema de la investigación:** ¿Cómo lograr, a partir de la utilización del Laboratorio Virtual de Sistemas Operativos, que los estudiantes adquieran los conocimientos necesarios en los temas Administración de memoria y Comunicación entre procesos, permitiendo así mejorar la calidad del aprendizaje en función de los resultados en los temas correspondientes?

Teniendo como **objeto de estudio:** proceso de desarrollo de software.

En función de resolver el problema propuesto se ha definido como **objetivo general:** desarrollar los módulos Administración de memoria y Comunicación entre procesos del subsistema de autoaprendizaje para el Laboratorio Virtual de Sistemas Operativos, permitiendo así mejorar la calidad del aprendizaje en función de los resultados en los temas correspondientes, especificado en el **campo de acción:** proceso de desarrollo de software de aplicaciones educativas para la creación de Laboratorios Virtuales.

Se define como **idea a defender:** desarrollando los módulos Administración de memoria y Comunicación entre procesos se facilitará, a partir de la utilización del Laboratorio Virtual de SO, que los estudiantes adquieran los conocimientos necesarios permitiendo así mejorar la calidad del aprendizaje en función de los resultados en los temas correspondientes.

Para darle cumplimiento al objetivo general propuesto se han definido los siguientes **objetivos específicos:**

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Obtener los artefactos correspondientes a las fases de Diseño e Implementación.
- ✓ Desarrollar la solución propuesta.
- ✓ Validar la solución propuesta.

A partir de los objetivos específicos anteriormente propuestos se definieron las siguientes **tareas de investigación:**

- ✓ Elaboración del marco teórico de la investigación.
- ✓ Obtención de los requisitos funcionales y no funcionales.
- ✓ Validación de los requisitos obtenidos.
- ✓ Obtención de los artefactos generados en la fase de diseño.
- ✓ Validación de los artefactos obtenidos en la fase de diseño.
- ✓ Implementación de la aplicación.
- ✓ Validación de los artefactos obtenidos en la fase de implementación

Para lograr el cumplimiento de las tareas planteadas se utilizaron métodos de investigación que permitieron maximizar la validez y confiabilidad de la información, disminuir los errores en los resultados y adquirir la solidez de conocimientos necesarios para el desarrollo de la aplicación propuesta.

Métodos de investigación empleados:

Métodos teóricos:

- ✓ Análisis histórico lógico: para profundizar en los antecedentes de la utilización de las TIC 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 estudiantes que interactúan con las TIC en el proceso de enseñanza aprendizaje presencial o mixto de la asignatura de SO en la Facultad 3 de la UCI.

Este trabajo de diploma se ha estructurado como a continuación se describe: resumen, introducción, cuatro capítulos, conclusiones generales, recomendaciones, referencias bibliográficas, bibliografía y anexos.

A continuación se da una breve reseña del contenido de cada capítulo.

Capítulo 1. Fundamentación teórica: en el presente capítulo se describe el marco teórico en el que se desarrollará el trabajo, el cual ofrece un fundamento sólido a la solución que se propone. Es de vital importancia realizar este estudio preliminar para enmarcar la investigación y arrojar los primeros resultados. Se presenta un estudio de los laboratorios virtuales existentes y de las herramientas utilizadas, además de la metodología y lenguaje de modelado utilizado para desarrollar la solución propuesta.

Capítulo 2. Características del sistema: se realiza una breve descripción de la solución propuesta, sus requisitos funcionales y no funcionales, así como los actores que intervienen en ella. También se presenta el modelo de dominio del sistema, el diagrama de casos de uso del sistema así como la descripción de un caso de uso como ejemplo a tratar durante todo el trabajo.

Capítulo 3. Diseño del sistema: se describe la representación arquitectónica del sistema. Se hace énfasis en los patrones de arquitectura utilizados, así como los patrones de diseño que fueron empleados en el desarrollo de la solución propuesta. Además de la descripción del modelo de datos que se utilizará en dicha solución.

Capítulo 4. Implementación y Prueba: se describe la implementación del sistema, mostrando los diagramas de componentes y todo lo referente a la solución implementada. Además se muestra todo lo relacionado con la validación de la aplicación y demás artefactos obtenidos durante el desarrollo de las funcionalidades en cada una de las iteraciones en el trabajo desarrollado.

Capítulo 1. Fundamentación teórica

Introducción

En el presente capítulo se describe el marco teórico de la investigación el cual ofrece un fundamento sólido a la solución que se propone. Es de vital importancia realizar este estudio preliminar para enmarcar la investigación y arrojar los primeros resultados. Se presenta un estudio de los laboratorios virtuales existentes y las herramientas utilizadas, además de la metodología y lenguaje de modelado utilizado para desarrollar la solución propuesta.

1.1 Laboratorio virtual (LV)

El desarrollo de las tecnologías de la información ha posibilitado la creación de sistemas de apoyo al aprendizaje en materias que carecen de una práctica para mejorar el desarrollo del proceso docente-educativo. Entre estos sistemas se destacan los laboratorios virtuales como medios accesibles y adaptables para poner en práctica los conocimientos, cuyo objetivo principal es proporcionar una experiencia similar a la obtenida en un laboratorio real.

Existen diversas definiciones de *Laboratorio Virtual*, a continuación se citan algunas:

"Un conjunto de experimentos virtuales que tiene como objetivo preparar al usuario para obtener el máximo rendimiento de un laboratorio real. Por ello, se decidió darle a este conjunto de experimentos el nombre de Laboratorio Virtual". [1]

Una de las definiciones de laboratorios virtuales que se ha aplicado a la enseñanza a distancia es la del biólogo Monge-Nájera, que los define como *"simulaciones de prácticas manipulativas que pueden ser hechas por el estudiante lejos de la universidad y el docente"*. [2]

"El laboratorio virtual es un tipo de colaboración centrada en el logro de determinados objetivos creativos o de ayuda a la toma de decisiones. Por lo tanto, un laboratorio virtual puede dedicarse prácticamente a todas las esferas de la actividad intelectual humana." [3]

A partir de las definiciones expuestas, es posible concretar que un *Laboratorio Virtual* constituye un entorno virtual con recursos de simulación que permite realizar prácticas en función de los conocimientos adquiridos en una materia dada.

1.1.1 Tipos de laboratorios virtuales

Los laboratorios virtuales representan una opción creativa, moderna y económica para instituciones universitarias que requieran laboratorios dentro de sus procesos de formación. Existen diversos tipos de laboratorios virtuales, pero generalmente son clasificados en tres tipos.[4]

Laboratorios virtuales de software.

Son laboratorios virtuales desarrollados como un software, destinado a ejecutarse de forma independiente en la máquina del usuario y cuyos servicios no requieren de recursos externos.

Laboratorios virtuales web.

Se basan en un software que depende de recursos ubicados en un servidor al cual se accede mediante un navegador web desde cualquier máquina conectada a la red. [5]

Laboratorios virtuales remotos.

Los laboratorios virtuales remotos son aquellos que permiten operar a distancia cierto equipamiento, bien sea didáctico como maquetas específicas, o industrial, además de poder ofrecer capacidades de laboratorio virtual.

1.1.2 Ejemplos existentes de laboratorios virtuales

Se ha realizado un estudio sobre los laboratorios que existen, a continuación se mencionan algunos de ellos. [6]

- ✓ **Entorno Virtual de Física,** desarrollado en la Universidad de las Ciencias Informáticas (UCI), con el cual se puede trabajar en la asignatura de Física I; consta con 8 prácticas de Laboratorio Virtual para el estudio de algunos temas de la asignatura.
- ✓ **Sistema Interactivo Didáctico de Enseñanza de la Física (SIDEF),** utilizado en el Departamento de Física de la Facultad de Matemática, Física y Computación de la Universidad Central “Marta Abreu” de Las Villas.
- ✓ **Laboratorio Virtual de Química,** utilizado en la Universidad de Villa Clara “Martha Abreu”, el cual a medida que el estudiante va trabajando y avanzando le muestra los conocimientos que ha ido alcanzando.

Posterior al estudio realizado en las bibliografías consultadas sobre los laboratorios virtuales, herramientas de gran importancia de apoyo al aprendizaje, se conoce que no existe un LV vinculado a la materia de SO

en la UCI, por lo que surge la necesidad de desarrollar un LV para la asignatura, que permita al estudiante el acceso al contenido teórico de manera autónoma, realizar las diferentes prácticas de simulación necesarias así como sus evaluaciones correspondientes. Se decide desarrollar este laboratorio virtual pues ninguno de los anteriormente planteados satisface las necesidades que hoy en día tiene la asignatura en cuestión. Asimismo, se concluye utilizar el laboratorio virtual web, ya que el usuario podrá acceder desde cualquier computador al servidor donde esté la práctica virtual, además podrán acceder a él más de un usuario de forma simultánea.

1.1.3 Ventajas de los laboratorios virtuales

Dentro de las ventajas más significativas del uso de laboratorios virtuales se destacan las siguientes:

- ✓ Facilita a un mayor número de estudiantes la adquisición de experiencias, aunque alumno y laboratorio no coincidan en el espacio.
- ✓ Los estudiantes aprenden mediante prueba y error, sin miedo a sufrir o provocar un accidente, sin avergonzarse de realizar varias veces la misma práctica, ya que pueden repetirlas sin límite; sin temor a dañar alguna herramienta o equipo. [7]

A continuación se mencionan algunos de los conceptos más importantes para la investigación en curso, los cuales son necesarios para la comprensión y el desarrollo del tema del trabajo.

1.1.4 Sistema de autoaprendizaje

Con el desarrollo tecnológico aparecen las distintas metodologías y estrategias de aprendizaje, las cuales emplean tecnología digital o informática para el aumento de la calidad en el proceso de aprendizaje. La combinación de la educación con la informática da lugar al surgimiento de la Informática Educativa (e-Learning), que según la definición de la Comisión Europea es “la utilización de las nuevas tecnologías de multimedia e Internet para mejorar la calidad del aprendizaje facilitando el acceso a recursos y servicios”. Vinculando entonces el autoaprendizaje al proceso de formación, se coincide en que Autoaprendizaje es la habilidad que cada persona posee para dirigir y regular la adquisición del conocimiento a través del estudio de diversos contenidos o de la experiencia en actividades de aprendizaje. Es la forma de aprender a aprender por uno mismo.

Al conocer estas realidades globales, se asume el reto de diseñar y poner en marcha programas institucionales dirigidos a la formación del profesional del siglo XXI, capaz de desempeñarse en el nuevo

entorno mediante la acogida de todas las metodologías y herramientas que le permitan llevar a cabo el autoaprendizaje, donde lo más importante ya no es el conocimiento sino la capacidad para adquirirlo, interpretarlo y utilizarlo. [8]

Por estas razones surgen los sistemas de autoaprendizaje, que no son más que programas informáticos, los cuales combinan libros impresos o electrónicos, laboratorios virtuales interactivos, laboratorios reales (entrenador) y sistemas de autoevaluación basados en el computador.

1.1.5 Comunicación entre procesos

La posibilidad de que dos o más procesos puedan comunicarse de alguna manera, se denominan comunicación entre procesos (en inglés IPC, *Inter Process Communication*). [9]

Aunque pueden concebirse procesos que funcionen de manera aislada e independiente, en la práctica ocurre que muchos son cooperantes o interactúan durante su ejecución. Un ejemplo básico lo constituyen las llamadas al sistema, en un entorno multitarea, varios procesos compartirán las mismas bibliotecas y recursos del sistema operativo. Desde las aplicaciones, otro ejemplo lo constituye el *clipboard*, que es un área en la memoria del sistema operativo en la que puede escribir y leer cualquier proceso. Los depuradores son otro buen ejemplo de procesos cooperantes; estos pueden examinar el código de otro proceso sin interrumpir su correcto funcionamiento. Situaciones menos evidentes, como la operación de los antivirus, son similares, ya que deben interceptarse las llamadas al sistema que emita cualquier proceso.

1.1.6 Administración de memoria

La forma más simple de administración de memoria es aquella donde solo se tiene un proceso en cada instante sin la existencia de Sistemas Operativos, esto tiene como desventaja que cada proceso debe contener todos los controladores para cada uno de los dispositivos que utilice. El esquema más común consiste en dividir la memoria en dos partes: una para el programa usuario y la otra para la parte residente del sistema. Generalmente este último se almacena en la parte baja de la memoria y el programa usuario en la parte alta. El esquema fue utilizado en las primeras microcomputadoras y de ahí que los Sistemas Operativos CP/M y MSDOS quedarán condicionados a ella. [10]

1.2 Ingeniería de Requisitos

"Los requisitos de software son la expresión en lenguaje natural de las características y restricciones que debe cumplir un sistema, de forma que identifiquen las necesidades del cliente y que posibiliten el entendimiento entre clientes y desarrolladores. Los requisitos pueden dividirse en requisitos funcionales y no funcionales." [11]

En la Ingeniería de Software al proceso de recopilar, analizar y verificar las necesidades del cliente para un sistema se le conoce como **Ingeniería de Requisitos** (IR). Comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones para satisfacer un software, tomando en cuenta los diversos requisitos de los clientes, así como los conflictos que pueden haber entre ellos. El propósito de la IR es hacer que los mismos alcancen un estado óptimo antes de arribar a la fase de diseño en el proyecto.

A continuación algunos beneficios que se obtienen de la IR son: [11]

- ✓ **Permite gestionar las necesidades del proyecto en forma estructurada:** cada actividad de la IR consiste en una serie de pasos organizados y bien definidos.
- ✓ **Mejora la capacidad de predecir cronogramas de proyectos conjuntamente con sus resultados:** la IR proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como, estimación de costos, tiempo y recursos necesarios.

1.2.1 Actividades de la Ingeniería de Requisitos

La IR implica todas las actividades del ciclo de vida del software dedicadas a la captura, documentación y validación de los requisitos. Dependiendo del tamaño del proyecto y del modelo de proceso de software utilizado para el ciclo de desarrollo, las actividades de la IR varían tanto en número como en nombres.

Desde un punto de vista conceptual las actividades son de cinco clases: [11]

- ✓ **Obtener requisitos:** a través de entrevistas o comunicación con clientes o usuarios para saber cuáles son sus deseos.
- ✓ **Analizar requisitos:** detectar y corregir las falencias comunicativas, transformando los requisitos obtenidos de entrevistas en condiciones apropiadas para ser tratados por el diseño.
- ✓ **Documentar requisitos:** al igual que todas las etapas los requisitos deben estar debidamente documentados.

- ✓ **Verificar los requisitos:** consiste en comprobar el correcto funcionamiento de un requisito en la aplicación.
- ✓ **Validar los requisitos:** comprobar que los requisitos implementados se corresponden con lo pretendido inicialmente.

1.2.2 Técnicas de la Ingeniería de Requisitos

Acorde a las mejores prácticas entre las técnicas para identificar, especificar y validar los requisitos están, entre otras, las siguientes: entrevistas, cuestionarios, tormenta de ideas, sistemas existentes, prototipos, casos de uso y matriz de trazabilidad. Técnicas que pueden ser empleadas en combinación para establecer los requisitos exactos de las personas implicadas, con el objetivo de producir un sistema que resuelva las necesidades del negocio.

- ✓ **Entrevistas:** se entrevista a una selección de personas que represente a todos los sectores críticos de la organización, con el énfasis puesto en los sectores más afectados o que harán un uso más frecuente del nuevo sistema. Los requisitos que surgen de las entrevistas a menudo se contradicen unos a otros o se formulan desde la ignorancia de los detalles del funcionamiento del sistema, sus potencialidades, interdependencias o limitaciones, por lo que se debe trabajar con los mismos para corregir sus fallos. [11]
- ✓ **Tormenta de ideas:** reunión con un grupo reducido, que generalmente no exceden las diez personas, para que cada uno de los participantes exponga sus propias ideas acerca de las funciones que el sistema debe cumplir. [11]
- ✓ **Sistemas existentes:** esta técnica consiste en analizar sistemas ya desarrollados que estén relacionados con el que se quiere construir. Se pueden analizar la interfaz de usuario, observando el tipo de información que se maneja y cómo es manejada. También es útil analizar las salidas que los sistemas producen (listados, consultas, etc.), ya que siempre pueden surgir nuevas ideas sobre la base de estas.
- ✓ **Prototipos: e**s una pequeña muestra, de funcionalidad limitada de cómo sería el producto final una vez terminado. Ayudan a conocer la opinión de los usuarios y rectificar algunos aspectos antes de llegar al producto terminado. Pueden ser: diagramas o aplicaciones operativas con funcionalidades sintetizadas. [11]

- ✓ **Casos de uso:** un caso de uso es una técnica para documentar posibles requisitos, graficando la relación del sistema con los usuarios u otros sistemas. Dado que el propio sistema aparece como una caja negra, y solo se representa su interacción con entidades externas, permite omitir dichos aspectos y determinar los que realmente corresponden a las entidades externas. El objetivo de esta práctica es mejorar la comunicación entre los usuarios y los desarrolladores, mediante la prueba temprana de prototipos para minimizar cambios hacia el final del proyecto y reducir los costos finales. [11]

Como proceso, la administración de requisitos es fundamental en todo proyecto de desarrollo de software, ya que se debe contar con una especificación clara y completa desde las fases iniciales para no tener problemas posteriores que impliquen un retraso en el cronograma, un presupuesto erróneo, o hasta la posible cancelación del proyecto.

Luego del estudio realizado se decide utilizar como técnicas: la entrevista, la revisión de sistemas existentes, así como los prototipos y los casos de uso, para con las mismas poder garantizar una correcta ingeniería de requisitos que garanticen un levantamiento exitoso de los mismos en el sistema a desarrollar.

1.3 Metodología de desarrollo

Las metodologías guían el proceso de desarrollo de un software de manera organizada y planificada, con vista a lograr mayor eficiencia y obtener un resultado satisfactorio. Describen una serie de actividades, herramientas y técnicas a utilizar durante el ciclo de vida del software. [12]

Disponer de diferentes metodologías para aplicarlas a proyectos de desarrollo, resulta imprescindible de acuerdo a las necesidades cambiantes que tiene el entorno de desarrollo actual y el acelerado progreso de la informática a nivel mundial. Estas metodologías se clasifican en tradicionales y ágiles.

Un paradigma de metodología tradicional lo representa el Proceso Unificado de Rational (RUP, acrónimo de *Rational Unify Process*), la cual constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos por la detallada documentación que genera. Es por esta razón principalmente que se decide utilizar RUP como metodología de desarrollo para guiar el proceso de desarrollo del LV, además fue la seleccionada por la dirección del proyecto LV al cual pertenece la investigación.

Para un mejor entendimiento de las ventajas que esta decisión trajo aparejada consigo, a continuación se describen un conjunto de características de la metodología en cuestión.

1.3.1 Proceso Unificado de Rational

RUP es el resultado de varios años de trabajo y uso práctico en el que se han unificado técnicas de desarrollo, a través del trabajo de muchas metodologías utilizadas por los clientes. Fue creado por Jacobson, Rumbaugh y Booch. Se inserta dentro de las metodologías orientadas a objetos y es una opción a tener en cuenta para desarrollar grandes y complejos proyectos.

Es una metodología para el desarrollo de software en la que se realiza el análisis y el diseño orientado a objetos. Incluye además un gran número de técnicas que soportan el ciclo de vida completo de desarrollo de software dando como resultado un proceso basado en componentes, *dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.* [13]

Dirigido por casos de uso: reflejan lo que los usuarios necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los mismos.

Centrado en la arquitectura: muestra la visión común del sistema completo en la que el equipo del proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura.

Iterativo e incremental: propone que en cada fase se desarrollen en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. En cada iteración se realiza un proceso de gestión de riesgos, garantizando así que los costes por algún problema afecten solo a la iteración y no al producto completo.

El proceso unificado se divide en ciclos de trabajo teniendo un producto superior como resultado de cada ciclo. Estos se componen en su interior por varias fases, en las cuales se llevan a cabo un conjunto de flujos para el desarrollo de todo el proyecto.

En RUP, cada ciclo produce una nueva versión del sistema y cada versión es un producto preparado para su entrega. Consta de su cuerpo de código fuente, incluido en componentes que puede compilarse y ejecutarse, además de manuales y otros productos asociados.

1.4 Lenguaje de modelado

Un lenguaje de modelado es un conjunto estandarizado de símbolos y modos de disponerlos para modelar un diseño de software orientado a objetos o parte de él. En la actualidad es muy utilizado el Lenguaje Unificado de Modelado.

1.4.1 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML, acrónimo de *Unify Model Language*) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático, desde el análisis con los casos de uso, el diseño con los diagramas de clases, y objetos, entre otros, hasta la implementación y configuración con los diagramas de despliegue. [14]

Los beneficios que se consiguen al utilizar UML 2.0 son varios, por un lado el uso de lenguajes visuales facilitan su asimilación y entendimiento por parte del equipo de desarrollo; el tiempo invertido en el desarrollo de la arquitectura se minimiza; la trazabilidad y documentación del proyecto se realiza de una forma ordenada y guiada por los casos de uso. Pero sí hay una ventaja que se destaca sobre todas las demás y es la notable efectividad y productividad que se consigue en labores de diseño arquitectónico, haciendo uso de UML frente a la realización de las mismas tareas en ausencia de modelos. [14]

Además es un lenguaje independiente de plataforma y constituye un estándar mundial para el modelado. Permite modelar sistemas utilizando técnicas orientadas a objetos (OO), puede conectarse con lenguajes de programación (Ingeniería directa e inversa) y lo más importante es que permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas y versiones). Al mismo tiempo es el lenguaje de modelado de sistemas de software que más se utiliza en la actualidad durante el proceso de desarrollo de software siendo una de las características fundamentales, además de las mencionadas anteriormente por las que se decide utilizar en el trabajo para modelar los artefactos.

1.5 Herramientas CASE

Las herramientas CASE (acrónimo de *Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) son aplicaciones informáticas que intentan proporcionar ayuda automatizada a las actividades del proceso de software. Permiten a los desarrolladores modelar y documentar sus artefactos, cubriendo el ciclo de vida del proceso de desarrollo de software.[15]

Existen varias herramientas CASE que permiten realizar un correcto modelado, las utilizadas se mencionan a continuación:

1.5.1 ER/Studio Embarcadero

ER/Studio 8.0 es una poderosa herramienta CASE para efectuar el modelado lógico y físico de bases de datos relacionales y multidimensionales. Entre otras características, permite hacer una separación entre los modelos lógicos y físicos, pero manteniendo una total integración entre ellos, así como con la base de datos. Soporta las principales bases de datos del mercado como son: Oracle, Sybase, Microsoft SQL Server, IBM DB2 UDB, tanto en plataforma Linux/Unix/Windows (LUW) como z/OS & OS/390 y iSeries (IBM mainframe), IBM Informix, MySQL, Paradox, Access, etc. Permite efectuar ingeniería inversa de esas bases de datos para documentar en formato .rtf, .xls o bien HTML, las estructuras de tablas, vistas, índices, objetos físicos, definiciones de seguridad, entre otros.

Esta herramienta ofrece a los administradores y desarrolladores de bases de datos la posibilidad de modelado de datos de forma visual, permitiendo el diseño y mantenimiento de bases de datos transaccionales, de soporte a la toma de decisiones y para la Web. Además de conexiones vía ODBC.

1.5.2 Visual Paradigm para UML

Visual Paradigm para UML 5.0 es una herramienta profesional que facilita el modelado del ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML soporta las últimas versiones del mismo y la Notación y Modelado de procesos de negocios. Ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste. [16]

Visual Paradigm para UML 5.0 ofrece distintas funcionalidades como:

- ✓ Entorno de creación de diagramas para UML 2.0.
- ✓ Diseño centrado en casos de uso y enfocado al negocio generando un software de mayor calidad.

- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Disponibilidad en múltiples plataformas (Windows, Linux, etc.)

1.6 Lenguaje de programación

1.6.1 Java

Java es un lenguaje de desarrollo de propósito general, y como tal es válido para realizar todo tipo de aplicaciones profesionales. [17]

A continuación se verán algunas características del lenguaje.

Es orientado a objetos:

Trabaja con sus datos como objetos y con interfaz a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.

Java incorpora funcionalidades como la resolución dinámica de métodos, funcionalidades que han sido heredadas de otros lenguajes de programación. Además, utiliza interfaz específica llamada RTTI (RunTimeTypeIdentification) que define la interacción entre objetos excluyendo variables de instancias o implementación de métodos. Las clases en Java tienen una representación en el tiempo de ejecución que permite a los programadores interrogar por el tipo de clase y enlazar dinámicamente la clase con el resultado de la búsqueda.

Es Seguro:

Las aplicaciones de Java resultan extremadamente seguras, ya que no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción de ciertos virus. Java no posee una semántica específica para modificar la pila de programa, la memoria libre o utilizar objetos y métodos de un programa sin los privilegios del núcleo del sistema operativo. También, para evitar modificaciones por parte de los crackers de la red, implementa un método seguro de autentificación por clave pública. El cargador de clases puede verificar una firma digital antes de realizar una instancia de un objeto. Por tanto, ningún objeto se crea y almacena en memoria sin que se validen los privilegios de acceso. Es decir, la seguridad se integra en el momento de compilación con el nivel de detalle y privilegios necesarios.

Es robusto:

Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda inmediatamente a detectar errores en el ciclo de desarrollo. Obliga a la declaración explícita de métodos reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria. También implementa los arreglos auténticos, en vez de listas enlazadas de punteros con comprobación de límites, para evitar la posibilidad de sobrescribir la memoria. Estas características reducen drásticamente el tiempo de desarrollo de aplicaciones en Java.

1.7 Entorno de desarrollo

1.7.1 Netbeans

El entorno de desarrollo integrado (IDE) NetBeans 7.0.1 es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Existe además un número importante de módulos para extender este IDE.

Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en control de versiones y refactorización.

NetBeans Enterprise Pack soporta el desarrollo de aplicaciones empresariales con JEE 5, incluyendo herramientas de desarrollo visuales de SOA, orientación a servicios web, herramientas de esquemas XML y modelado UML.

Sun Studio, Sun Java Studio Enterprise y Sun Java Studio Creator de Sun Microsystems han sido todos basados en el IDE NetBeans. Desde Julio de 2006 este IDE es licenciado bajo la Common Development and Distribution License (CDDL), una licencia basada en la Mozilla Public License (MPL). [18]

1.8 Ambiente de pruebas

1.8.1 JUnit

JUnit permite realizar la ejecución de clases Java de manera controlada para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado. Si la clase cumple con la especificación,

entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba. En caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. [19]

JUnit también permite controlar pruebas de regresión, es decir, detectar nuevos errores que puedan surgir a consecuencia de haber cambiado el código fuente inicial (por ejemplo, añadiendo nuevas funcionalidades). De esta manera, JUnit permitirá comprobar que la aplicación cumple con los requisitos y que no se ha alterado su funcionalidad inicial.

El framework de JUnit permite visualizar los resultados en modo texto y en modo gráfico. Una de las mejores características de JUnit es que existen plug-ins para Eclipse y Netbeans que generan automáticamente las clases java necesarias para la creación de la pruebas, de manera que el programador solo tenga que centrarse en los resultados, si fueron correctos o no.

1.8.2 JMeter

JMeter es una herramienta Open Source, realizada en Java, que se utiliza para realizar tests de rendimiento normalmente contra aplicaciones web. JMeter permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar su “fuerza” y para analizar el rendimiento ante diferentes tipos de sobrecarga.[20]

A continuación se mencionan alguna de las principales características de JMeter:

- ✓ Permite cargar y realizar tests sobre distintos tipos de servidores: Web (HTTP, HTTPS), SOAP, Bases de Datos (JDBC), LDAP, JMS, Email (POP3 e IMAP).
- ✓ Multiplataforma: Unix (Solaris, Linux, etc), Windows (98, NT, XP, etc), OpenVMSAlpha 7.3+.
- ✓ Permite comprobar cómo se comportará una aplicación web ante multitud de usuarios y la velocidad de carga que tendrá.
- ✓ Es extensible logrando con esto un alto grado de personalización.

Dichas características por si solas fundamentan la decisión de utilizar JMeter, ya que al término de la misma se podrá tener una idea más exacta de cómo se comportará la aplicación desarrollada frente a un conjunto de escenarios posible, como por ejemplo, si la aplicación podría fallar cuando un número significativo de usuarios se conecten al mismo tiempo, etc.

1.9 Framework a utilizar

Los frameworks son arquitecturas de software bien definidas creados tanto para organizar como para integrar diferentes componentes de un proyecto con la idea de facilitar su desarrollo. El desarrollo de frameworks en la actualidad posee gran aceptación entre los desarrolladores, pues el mismo brinda la posibilidad de reutilizar el código del diseño y el código fuente. [21]

1.9.1 Vaadin

Vaadin 6.7.0 no es más que un framework para el desarrollo web permitiéndole a los desarrolladores construir interfaz de usuarios en Java con alta calidad. Además, está provisto de una extensa biblioteca de componentes para el desarrollo de dicha interfaz de usuarios al estar su arquitectura basada en componentes, le permite a los desarrolladores poder desarrollar sus propios componentes. Como características fundamentales se tienen el fácil uso del framework, la posible reutilización del código desarrollado en el mismo, la extensibilidad de sus componentes; además, solo se desarrolla en Java, siendo esto una ventaja para los programadores ya que no necesitarán conocer otros lenguajes como son HTML, CCS, o JavaScript. [22]

1.9.2 JasperReport

Es una herramienta para la generación de informes. Está implementada en Java, es de código abierto y fue desarrollada por Teodor Danciu para facilitar el agregar capacidades de reporte a las aplicaciones Java. No es una herramienta por sí sola ya que no se puede instalar. Para utilizarla es necesario añadirla a las aplicaciones Java por medio de la inclusión de su librería al classpath (Variable de entorno que permite a la JVM conocer dónde localizar sus clases) de la aplicación, la cual no tiene ningún tipo de dependencia con las librerías de Java, lo que posibilita utilizar JasperReport también para aplicaciones de escritorio. Permite incluir datos de texto, imágenes, gráficos, subreportes, tablas, entre otras características básicas en los reportes para que estos tengan un aspecto profesional. Algunas de las características principales que provee son: [23]

- ✓ **Permite una diagramación flexible de los reportes:** los reportes se pueden dividir en secciones opcionales que son: título, encabezado de página, una sección para los detalles del reporte, pie de página y una sección de resumen que aparecen al final del reporte.

- ✓ **Permite que los desarrolladores le surtan datos en varias formas:** los desarrolladores pueden pasar datos a los reportes por medio de parámetros. Estos parámetros pueden ser instancias de cualquier clase de Java.
- ✓ **Pueden generar sub-reportes:** permite la creación de sub-reportes dentro de los reportes lo que facilita el diseño.

Requerimientos de JasperReport:

- ✓ Se requiere tener instalado en el equipo el JDK 1.3 o posterior. No basta con tener instalado el J2RE (Run Time Environment).

1.9.3 JFreeChart

JFreeChart es una librería para gráficos escrita 100% en Java que facilita mostrar gráficos de calidad profesional en la aplicación que se desea desarrollar, ya sean web o de escritorio. Entre las características principales se encuentran: [24]

- ✓ Es una API consistente y bien documentada con soporte para un amplio rango de tipos de gráficas.
- ✓ Tiene un diseño flexible fácilmente extensible, y la posibilidad de ser usado tanto en tecnologías de servidor (aplicaciones Web) como de cliente (Swing, por ejemplo).
- ✓ Consta de soporte para varios tipos de salida, incluyendo componentes Swing, archivos de imagen como PNG y JPEG, formatos gráficos de vectores (incluyendo PDF, EPS y SVG).
- ✓ Es un framework open source, más específicamente, Software Libre. El mismo se distribuye bajo la licencia LGPL, el cual permite el uso en aplicaciones propietarias.

Por las características que se mencionaron con anterioridad de JFreeChart es que se decide utilizar este framework, el cual garantizará que los reportes visualizados en el LV tengan la calidad requerida por cualquier aplicación profesional de hoy en día.

1.9.4 JPA

Java Persistence API, más conocida por sus siglas JPA en su versión 6.7.2, es la API de persistencia desarrollada para la plataforma Java EE. Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE).
Fue originado a partir del trabajo del JSR 220 Expert Group. Ha sido incluido en el estándar EJB3.

Persistencia en este contexto cubre tres áreas: [25]

- ✓ La API en sí misma, definida en javax.persistence.package
- ✓ La Java Persistence Query Language (JPQL)
- ✓ Metadatos objeto/relacional

El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, siendo esta característica la más fundamental y por la que se decide utilizar este framework.

1.10 Herramienta para la gestión de datos

1.10.1 PostgreSQL

PostgreSQL 9.1 es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

A continuación se mencionan alguna de las principales características de este Gestor de Bases de Datos las cuales permitieron hacer uso para el trabajo con la base de datos:

- ✓ **Alta Concurrencia:** mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.
- ✓ **Amplia variedad de tipos nativos:** PostgreSQL provee nativamente soporte para tipos de datos tales como números de precisión arbitraria, texto de largo ilimitado, figuras geométricas, direcciones IP, bloque de direcciones estilo CIDR, direcciones MAC, arreglos, entre otros. Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.
- ✓ **Durabilidad:** es la propiedad que asegura, una vez realizada la operación, que ella persistirá y que no se podrá deshacer aunque falle el sistema. [26]

1.10.2 PgAdmin III

PgAdmin III 1.14.0 es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular bajo la licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como PervasivePostgres, EnterpriseDB, MammothReplicator y SRA PowerGres.

PgAdmin III 1.14.0 está diseñada para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas UNIX), y puede encriptarse mediante SSL para mayor seguridad. [27]

1.11 Servidor para aplicaciones web

1.11.1 Apache

Apache 2.2.3 es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. En 1995, cuando comenzó su desarrollo, se basó inicialmente en el código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf quería que tuviese la connotación de algo que fuera firme y enérgico, pero no agresivo; de ahí el nombre de Apache, quien fuera la última tribu en rendirse ante la conformación de lo que luego se convertiría en el Gobierno de los EEUU .En esos momentos la preocupación del grupo, que luego sería conocido como Apache Software Fundation, era que llegasen las empresas y “civilizaran” el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés, a patchy server (un servidor “parcheado”) suena igual que Apache Server. [28]

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta, entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayudará en su configuración.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan solo pueden ser aprovechadas por usuarios locales y no remotos. Sin embargo, algunas se pueden accionar en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Las características más importantes de este servidor web son:

- ✓ Es un servidor modular.
- ✓ Es un servidor basado en código abierto.
- ✓ Es un servidor de aplicaciones multiplataforma.

Conclusiones parciales

En este capítulo se abordaron temas que permitieron realizar una síntesis de las características y aspectos esenciales del objeto de estudio afrontado, los cuales resultan un importante aporte para proponer soluciones en función de cumplimentar el primero de los objetivos específicos trazados. Una vez que:

- ✓ Con el estudio realizado, acerca de los laboratorios virtuales y su composición, se determinaron los aspectos fundamentales que deberán cumplir cada módulo a desarrollar.
- ✓ La utilización de la metodología RUP, los lenguajes de modelado UML y Visual Paradigm for UML en su versión 5.0, permitirán generar los artefactos bajo los estándares establecidos en el desarrollo de software.

Capítulo 2. Descripción del sistema

Introducción

Se realiza una breve descripción de la solución propuesta, sus requisitos funcionales y no funcionales, así como los actores que intervienen en ella. Además se presenta el modelo de dominio del sistema, el diagrama de casos de uso del sistema y la descripción de un caso de uso como ejemplo a tratar durante todo el trabajo.

2.1 Modelo de dominio

Existen varias alternativas para llevar a cabo el modelamiento de un sistema. RUP, como metodología de desarrollo, propone la realización de un modelo de negocio para el caso en el que los procesos dentro del entorno estén claramente definidos y por otra parte propone la realización de un modelo de dominio para los escenarios en los que no puedan identificarse tales procesos del negocio como es el caso de los módulos a desarrollar por tanto se decide realizar un modelo de dominio.

El modelo de dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Define un modelo de clases común para todos los implicados en el desarrollo, representadas en objetos del dominio, sirve como interlocutor entre clientes y desarrolladores. El propósito fundamental de este modelo es generar una terminología común y sentar las bases del entendimiento del desarrollo y no para definir el sistema completo. [13]

2.1.1 Conceptos del modelo de dominio

Es necesario tener un vasto conocimiento de cómo debe funcionar el proceso en cuestión, para poder capturar correctamente los requisitos y así poder construir un sistema con las características que el cliente desea. Este modelo, posteriormente, contribuirá a identificar algunas clases que se utilizarán en el sistema. Primeramente se identificarán todos los conceptos que se utilizarán en el diagrama.

Conceptos	Descripción
Profesor	Persona con el conocimiento y la metodología necesaria para impartir una asignatura determinada.
Laboratorio virtual	Aplicación en desarrollo para el estudio de un grupo de temas relacionados con la asignatura Sistemas Operativos.

Módulo Autoaprendizaje	Módulo donde el estudiante podrá encontrar todo lo referente a los contenidos de la asignatura.
Contenido	Referente a todo el conocimiento que tiene que ver con la asignatura, dígase texto, medias, etc.
Estudiante	Persona interesada en adquirir un grupo de conocimientos con respecto a la asignatura.
Módulo Ejercicios	Módulo donde el estudiante encontrará todo lo referente a los ejercicios predefinidos por el claustro de profesores con el fin de validar el conocimiento adquirido.
Módulo Reporte	Módulo donde tanto el estudiante como el profesor podrán chequear las notas adquiridas por el estudiante al término de realizar los ejercicios correspondientes a un tema determinado.

Tabla 1. Conceptos del modelo de dominio.

A continuación se muestra el modelo de dominio correspondiente:

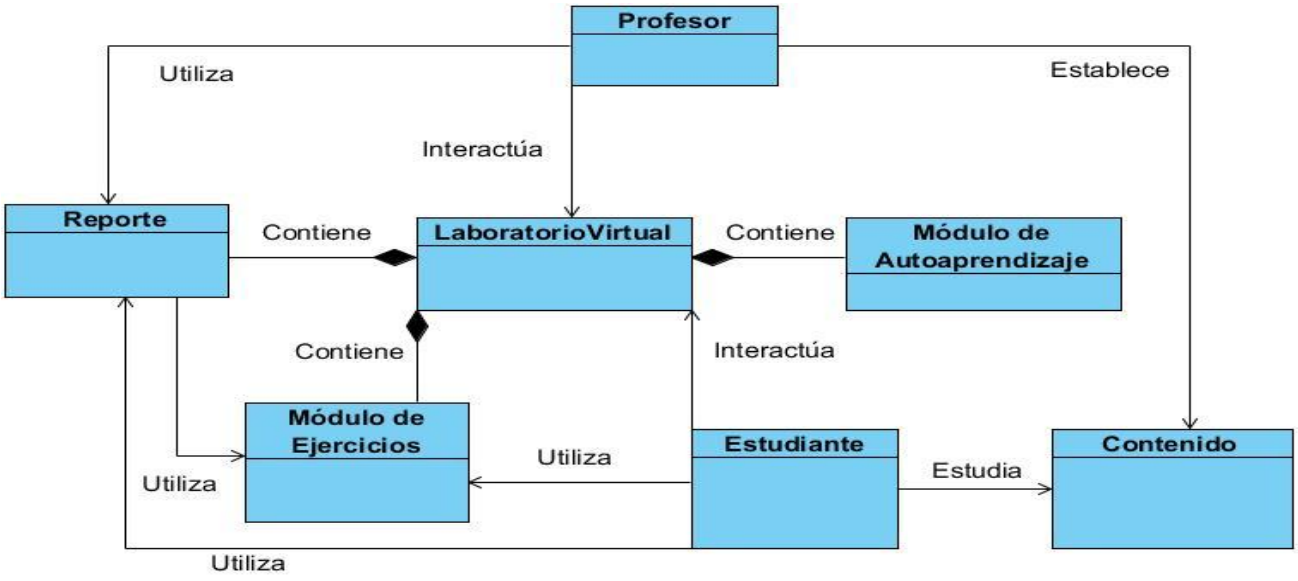

Figura 1. Modelo de dominio.

En el diagrama de clases del modelo de dominio representado anteriormente en la figura 1, se puede apreciar como el profesor, interactúa con el LV donde publicará el contenido de un tema en específico en el subsistema de autoaprendizaje, luego de que el profesor haya publicado el contenido, el estudiante tendrá acceso al contenido referente a los temas de los módulos mencionados con anterioridad. Después del estudiante haberse estudiando el contenido en cuestión, para validar los conocimientos adquiridos, este accederá al módulo de ejercicios contenido en el LV donde realizará un grupo de ejercicios previamente diseñados por el claustro de la asignatura, luego de realizar todas las evaluaciones pertinentes con respecto al tema específico, podrá consultar su evaluación en el módulo de reportes y de no estar conforme con la nota adquirida pues tendrá derecho a revalorizar el contenido estudiando a través del mismo módulo de ejercicios.

2.2 Especificación de requisitos del software

La especificación de requisitos de software es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye los casos de uso que describen las interacciones de los usuarios con el software y las cualidades o restricciones de diseño que el cliente desea para su producto. [11]

2.2.1 Requisitos funcionales

Un requisito funcional (RF) define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. Son condiciones o capacidades que el sistema debe cumplir. [11]
El subsistema de Autoaprendizaje debe ser capaz de:

- ✓ RF1.- Adicionar Contenido.
- ✓ RF2.- Modificar Contenido.
- ✓ RF3.- Revisar Contenido.
- ✓ RF4.- Eliminar Contenido.
- ✓ RF5.- Consultar Contenido.
- ✓ RF6.-Realizar Evaluación.
- ✓ RF7.- Consultar Evaluación.
- ✓ RF8.- Rechazar Evaluación.
- ✓ RF9.- Revalorizar Evaluación.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Así mismo, debe pensarse en estas propiedades, como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. [11]

Interfaz de usuario:

- ✓ RNF1: la aplicación deberá poseer una interfaz amigable e intuitiva.
- ✓ RNF2: los elementos gráficos en la interfaz deberán representar claramente la acción o información al que están asociados, de forma que la interfaz permita la interacción natural del sistema con sus usuarios.
- ✓ RNF3: el usuario debe poder acceder a los documentos de ayuda desde cualquier parte del sistema.

Seguridad:

- ✓ RNF4: garantizar que la información solo sea modificada por las personas que tienen permisos para realizar esta actividad.
- ✓ RNF5: incluir mecanismo de recuperación de datos (salvas automáticas o copias de respaldo) que permitan recuperar la información ante un fallo del sistema.
- ✓ RNF6: los mecanismos de seguridad no afectarán el tiempo de respuesta a las solicitudes de los usuarios.
- ✓ RNF7: el sistema debe estar siempre disponible para sus usuarios.

Soporte:

- ✓ RNF8: el sistema debe ser capaz de asimilar la incorporación de nuevos recursos que enriquezcan su contenido.

Rendimiento:

- ✓ RNF9: el tiempo de respuesta del sistema no deberá exceder los 9 segundos.

Software:

- ✓ RNF10: el sistema podrá ser ejecutado en cualquier navegador web que cuente con interprete JavaScript.

- ✓ RNF11: para el correcto funcionamiento del sistema se tendrá que tener instalado la máquina virtual de Java en su versión 1.5 o superior.
- ✓ RNF12: para la ejecución del sistema desarrollado se propone la utilización de Ubuntu Linux en su versión 11.10.

Hardware:

- ✓ RNF13: Microprocesador Pentium IV a 2.41 GHz o superior.
- ✓ RNF14: 1 GB de memoria RAM como mínimo.

2.3 Actores del sistema

Los actores del sistema representan el rol que juega una o varias personas, un equipo o un sistema automatizado, son parte del sistema y pueden intercambiar información con él o ser recipientes pasivos de información. [13]

En este caso los actores que interactúan con el subsistema se definen en la siguiente tabla.

Actores	Justificación
Estudiante	Representa al estudiante que se encuentra cursando la asignatura SO que se encontrará en el Laboratorio Virtual.
Profesor	Representa al profesor del claustro de la asignatura SO desplegada en el Laboratorio Virtual.
Profesor Editor	Representa al profesor que podrá gestionar todo lo referente al contenido y los temas de la asignatura SO.

Tabla 2. Actores del sistema.

2.4 Diagrama de casos de uso del sistema

Un diagrama de casos de uso del sistema (DCUS) muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. [13]

Las funcionalidades que serán automatizadas como parte del subsistema que se está modelando y su relación con los actores se muestran en el diagrama que aparece a continuación.

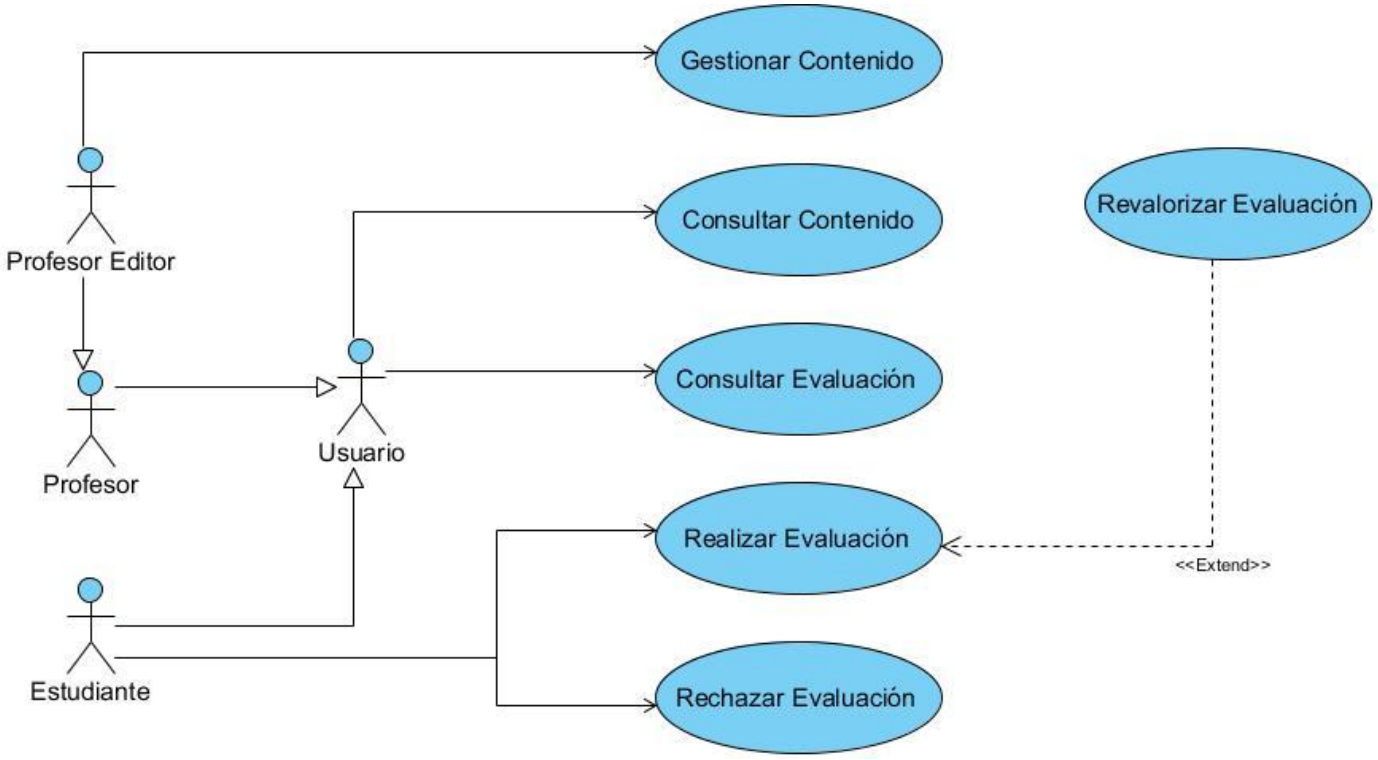

Figura 2. Diagrama de casos de uso del sistema.

2.4.1 Patrones de casos de uso aplicados

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, o sea, describen el uso del sistema y cómo este interactúa con los usuarios. Además, estos patrones son utilizados generalmente como plantillas que describen cómo deberían ser estructurados y organizados los casos de uso del sistema a desarrollar. También son utilizados para capturar mejores prácticas para modelar los casos de uso del sistema que se desea desarrollar.[29]

La aplicación de estos patrones en el desarrollo de una aplicación informática determinada trae múltiples beneficios los cuales son mencionados a continuación:

- ✓ Se logrará aumentar la productividad a la hora de desarrollar una aplicación informática determinada.
- ✓ Permite reutilizar elementos existentes en el desarrollo de una aplicación informática determinada.

- ✓ Se logrará evitar la reelaboración por errores a la hora de la implementación de las funcionalidades de un sistema dado.
- ✓ Permite ganar tiempo y no malgastar el mismo en resolver problemas ya resueltos.
- ✓ Lograr aplicar la teoría desarrollada en previo trabajo de mesa por los ingenieros en sistema, facilitándole así el trabajo a los programadores de un equipo de desarrollo dado.

Existen diferentes tipos de patrones de casos de uso, entre los cuales se pueden encontrar el patrón CRUD, Concordancia, Extensión concreta o Inclusión, Múltiples actores, entre otros.

A continuación se mencionan y se ejemplifican los que fueron utilizados para el desarrollo de este trabajo.

- ✓ Patrón CRUD

El Patrón CRUD completo consiste en un caso de uso para administrar la información (CRUD Información), permitiendo modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar. [29]

En la Figura 3 se puede apreciar un ejemplo donde fue empleado dicho patrón en el desarrollo de la solución propuesta para los módulos desarrollados.

Figura 3.Caso de uso Gestionar Contenido, Patrón CRUD completo.

- ✓ Patrón Extensión

El Patrón Extensión no es más que otra forma de interacción, un caso de uso dado (la extensión) puede extender a otro. Esta relación indica el comportamiento del caso de la extensión el cual puede ser inicializado o no por el caso de uso base. El caso de uso extensión puede ser insertado en el caso de uso extendido bajo ciertas condiciones. La notación, es una flecha de punta abierta con línea discontinua, desde el caso de uso extensión al caso de uso extendido, con la etiqueta «extend». Esto puede ser útil para lidiar con casos de uso especiales, o para acomodar nuevos requisitos durante el mantenimiento del sistema y su extensión. [29]

En la Figura. 4 se puede apreciar un ejemplo donde fue empleado dicho patrón en el desarrollo de la solución propuesta para los módulos desarrollados.

Figura 4.Caso de uso Consultar Evaluación, Patrón Extensión.

- ✓ Patrón Múltiples actores

El Patrón Múltiples actores solo puede suceder cuando dos actores juegan el mismo rol sobre un CU específico. En este caso se representa otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso. [29]

En la Figura 5 se puede apreciar un ejemplo de cómo fue empleado dicho patrón en el desarrollo de la solución propuesta para los módulos desarrollados.

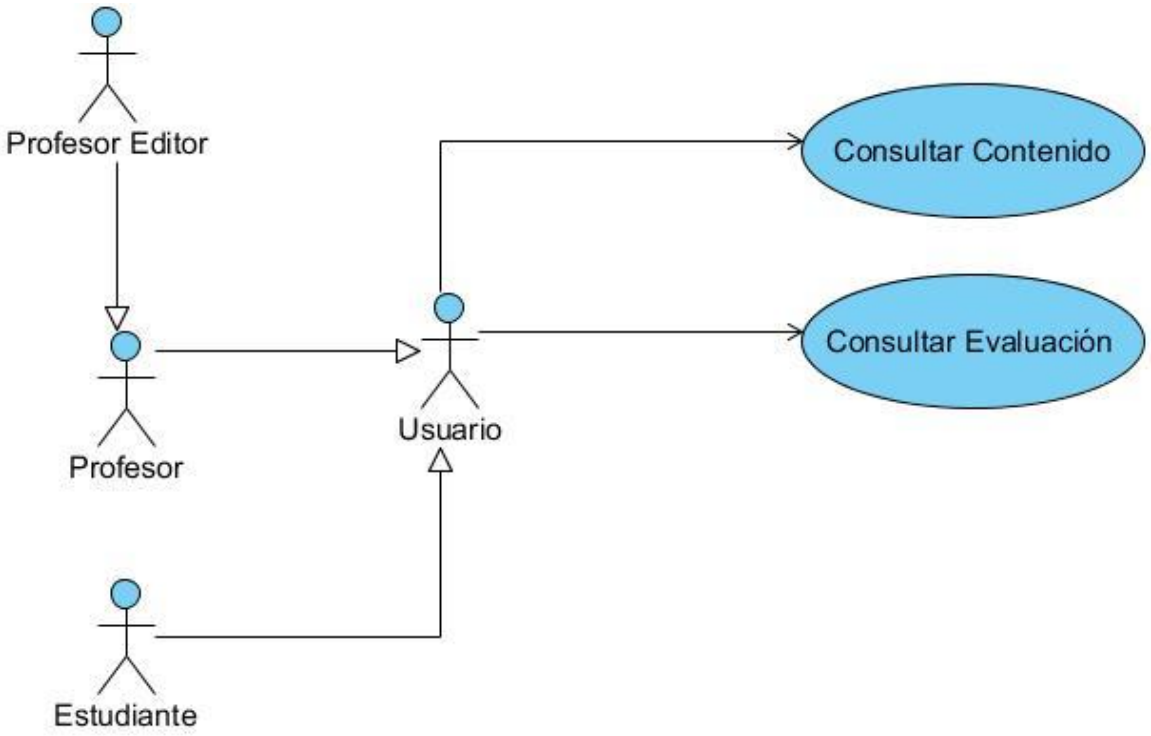

Figura 5.Caso de uso Consultar Evaluación, Patrón Múltiples actores.

2.4.2 Descripción del caso de uso del sistema

Las descripciones de casos de uso son reseñas textuales del caso de uso que explican los procesos o actividades que tienen lugar en el mismo. [13]

A continuación se describe un caso de uso crítico del subsistema en cuestión, los demás se encuentran en el Anexo # 1.

Descripción del Caso de Uso: "Gestionar contenido"

Caso de Uso	Gestionar Contenido
Actores	Profesor Editor
Propósito	Adicionar, Modificar, Revisar, y Eliminar contenido de la asignatura para el posterior uso por los estudiantes en el laboratorio virtual.
Resumen	El caso de uso se inicia cuando el Profesor Editor se dispone a incorporar, revisar, modificar o eliminar todo lo referente al contenido sobre la asignatura en específico, para su posterior estudio por los estudiantes en el laboratorio virtual.
Referencias	RF1, RF2, RF3, RF4.
Precondiciones	Debe existir el contenido de la asignatura que se desea incorporar, revisar, modificar o eliminar.
Prioridad	Crítico
Flujo normal de Eventos	
1.- El Profesor Editor selecciona la opción de gestionar contenido en la vista principal.	1.1.- El sistema muestra en una región las opciones necesarias para la gestión del contenido con el cual se desea trabajar en el laboratorio virtual.
2.- El Profesor Editor podrá seleccionar una de las siguientes opciones que se muestran en la vista gestionar contenido y que se mencionan a continuación: ✓ Adicionar Contenido. ✓ Modificar Contenido.	2.1.- El sistema muestra una de las siguientes secciones: ✓ Si seleccionó Adicionar Contenido ir a la sección "Adicionar Contenido". ✓ Si seleccionó Modificar Contenido ir a la sección "Modificar Contenido". ✓ Si seleccionó Revisar Contenido ir a la sección

✓ Revisar Contenido ✓ Eliminar Contenido.	"Revisar Contenido". ✓ Si seleccionó Eliminar Contenido ir a la sección "Eliminar Contenido".
Sección "Adicionar Contenido" Flujo normal de eventos.	
Actores	**Sistema**
1.- El Profesor Editor selecciona la opción "Adicionar Contenido".	1.1.- El sistema mostrará en la región principal de dicha opción un panel, donde el Profesor Editor podrá escribir el contenido que desee adicionar o subir a partir de un fichero la información que se desee para el estudio posterior del mismo.
2.- El Profesor Editor tecleará el contenido que desee adicionar y presionará el botón "Adicionar Contenido".	2.1.- El sistema mostrará un campo pidiendo la ubicación del tema que se desea incluir.
3.- El Profesor Editor indicará el tema al que se le desea añadir el contenido que se está incorporando.	3.1.- El sistema ubica el contenido en el tema que se indicó anteriormente y muestra cómo quedará el contenido adicionado en el tema señalado.
4.- El Profesor Editor revisará el contenido incorporado y de estar satisfecho presionará el botón "Finalizar".	4.1- El sistema incluye al tema señalado el contenido adicionado dejándolo disponible para su posterior estudio y regresará a la vista principal del Profesor Editor.
Flujo Alterno	
Actores	**Sistema**
4.- El Profesor Editor presiona el botón "Cancelar".	4.1.- El sistema retorna al flujo normal de eventos número 1.
Sección "Modificar Contenido" Flujo normal de eventos.	
Actores	**Sistema**
1.- El Profesor Editor selecciona la opción "Modificar Contenido".	1.1.- El sistema mostrará en la región principal de dicha opción un panel donde el Profesor Editor podrá buscar el

	contenido a modificar.
2.- El Profesor Editor tecleará el nombre del contenido que desea modificar en el cuadro de búsqueda.	2.1.- El sistema hará una búsqueda del contenido que el profesor tecleó en el cuadro de búsqueda, mostrando el resultado de la búsqueda.
3.- El Profesor Editor seleccionará el contenido que desea modificar y presionará el botón “Modificar Contenido”.	3.1.- El sistema mostrará en un editor el contenido que se desea modificar.
4.- El Profesor Editor modificará el contenido que desee y presionará el botón “Aceptar”.	4.1.- El sistema mostrará una advertencia de que se modificará el contenido en cuestión mostrando una vista previa de dicho contenido modificado.
5.- El Profesor Editor revisará si las modificaciones son correctas y presionará el botón “Finalizar”.	5.1- El sistema modificará el contenido en cuestión y regresará a la vista principal del Profesor Editor.
Flujo Alterno	
Actores	**Sistema**
4.- El Profesor Editor presiona el botón “Cancelar”.	4.1- El sistema retorna al flujo normal de eventos número 1.
Sección “Revisar Contenido” Flujo normal de eventos.	
Actores	**Sistema**
1.- El Profesor Editor selecciona la opción “Revisar Contenido”.	1.1.- El sistema mostrará en la región principal de dicha opción un panel donde el Profesor Editor podrá buscar el contenido a revisar.
2.- El Profesor Editor tecleará el nombre del contenido que desea revisar en el cuadro de búsqueda.	2.1.- El sistema hará una búsqueda del contenido que el profesor tecleó en el cuadro de búsqueda, mostrando el resultado de la búsqueda.
3.- El Profesor Editor seleccionará el contenido que desea revisar y presionará el botón “Revisar Contenido”.	3.1- El sistema mostrará en una pantalla el contenido que se desea modificar.

4.- El Profesor Editor revisará que el contenido está completo y presionará el botón “Aceptar”.	4.1.- El sistema regresará a la vista principal del Profesor Editor.
Flujo Alterno	
Actores	**Sistema**
4.- El Profesor Editor presiona el botón “Modificar Contenido”.	4.1.- El sistema regresará a la sección “Modificar Contenido” al flujo normal de eventos número 4.
Sección “Eliminar Contenido” Flujo normal de Eventos.	
Actores	**Sistema**
1.- El Profesor Editor selecciona la opción “Eliminar Contenido”.	1.1.- El sistema mostrará en la región principal de dicha opción un panel donde el Profesor Editor podrá buscar el contenido a eliminar.
2.- El Profesor Editor tecleará el nombre del contenido que desea eliminar en el cuadro de búsqueda.	2.1.- El sistema hará una búsqueda del contenido que el profesor tecleó en el cuadro de búsqueda, mostrando el resultado de la búsqueda.
3.- El Profesor Editor eliminará el contenido que el sistema devolvió en la búsqueda realizada previamente y presionará el botón “Eliminar Contenido”.	3.1.- El sistema eliminará el contenido que el Profesor Editor seleccionó y regresará a la vista principal del Profesor Editor.
Flujo Alterno	
Actores	**Sistema**
3.- El Profesor Editor presiona el botón “Cancelar”	3.1- El sistema retorna al flujo normal de eventos número 1.
Poscondiciones	El Laboratorio Virtual queda listo para su posterior uso tanto por estudiantes como por profesores de la asignatura.

Tabla 3. Descripción del CU Gestionar Contenido.

Conclusiones parciales

Con el desarrollo de este capítulo se llegó a las siguientes conclusiones:

- ✓ La realización del modelo de dominio y parte del diseño facilitan el proceso de comprensión de la solución que se propone, para la cual se utilizaron y tuvieron en cuenta aspectos fundamentales del sistema que se necesita, como los requisitos funcionales y no funcionales descritos en el capítulo.
- ✓ Se logra una correcta modelación de la propuesta de solución, obteniendo y transformando los requisitos en el diseño de los módulos desarrollados.
- ✓ Los artefactos generados en el capítulo, serán la base para su posterior implementación, fase en la cual deberán ser reproducidas cada una de las funcionalidades aquí descritas de forma exacta.

Capítulo 3. Diseño del sistema

Introducción

En el presente capítulo se describe la representación arquitectónica del sistema. Se hace énfasis en el patrón arquitectónico utilizado, así como los patrones de diseño que fueron empleados en el desarrollo de la solución propuesta. Además de la descripción del modelo de datos que se utilizará en dicha solución.

3.1 Representación arquitectónica

Un avance importante dentro de la construcción del software ha sido el desarrollo de la arquitectura de software, toda vez que permite representar la estructura de un sistema, a un nivel mayor que el dado por la programación o incluso por el diseño. [13]

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlos en nuevas situaciones y discusiones sobre sus puntos fuertes y débiles. Existen varios tipos de patrones, ya conocidos, entre los que se encuentran los patrones de requisitos, de arquitectura, de diseño y de programación, entre otros. A continuación se presenta el patrón arquitectónico que fue utilizado para definir la arquitectura del sistema.

3.1.1 Patrón arquitectónico empleado

Los patrones arquitectónicos especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones, para organizar los distintos componentes. [30]

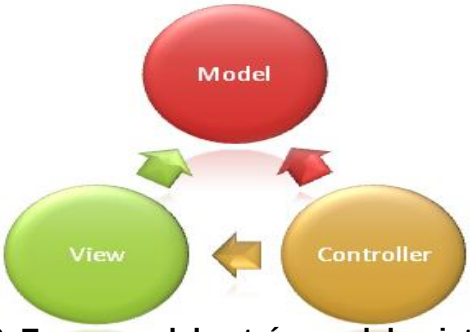

Figura 6. Esquema del patrón modelo-vista-controlador.

El patrón Modelo – Vista – Controlador mostrado en la Figura 6 está catalogado como un patrón de arquitectura de software donde:

Modelo: representa específicamente el dominio de la información sobre la cual funciona la aplicación.

El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos. El modelo encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

Vista: presenta el modelo en un formato adecuado para interactuar, usualmente con un elemento de interfaz de usuario. Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

Controlador: responde a eventos e invoca cambios en el modelo y probablemente en la vista. Los eventos son traducidos a solicitudes de servicio para el modelo o la vista.

A continuación se presenta en la Figura 7 un ejemplo de cómo se aplica este patrón en los módulos desarrollados en este trabajo.

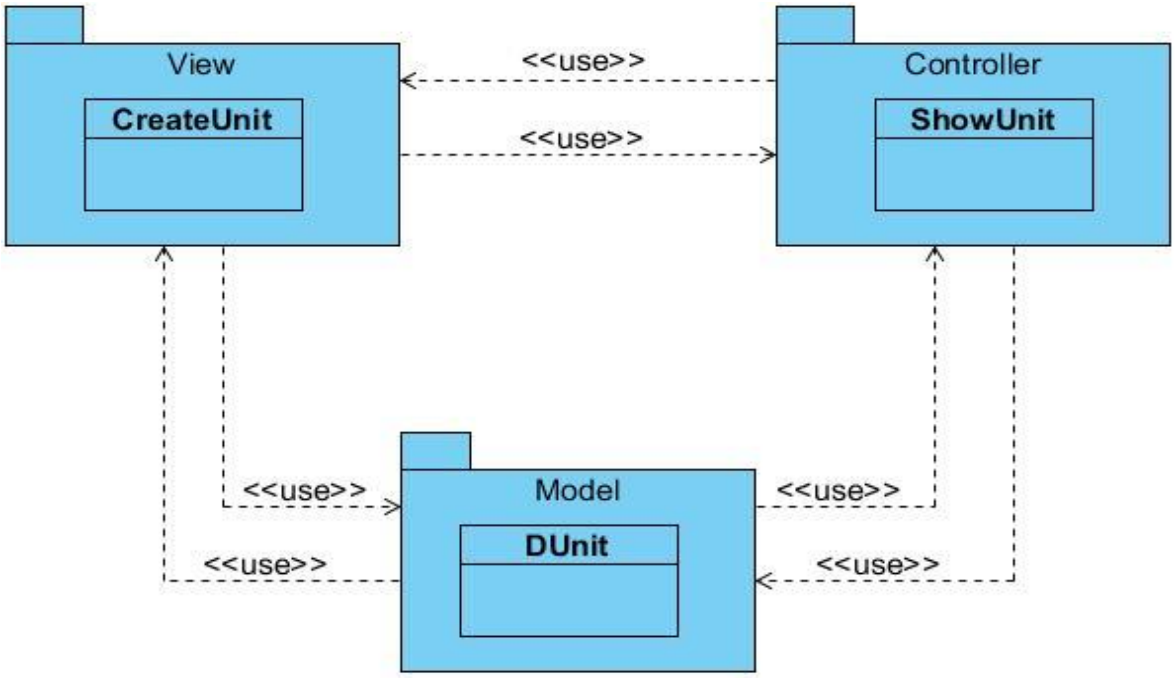

Figura 7. Aplicación del patrón modelo-vista-controlador

3.2 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. El modelo de diseño sirve de abstracción a la implementación y se utiliza como una entrada fundamental de las actividades de implementación. [13]

3.2.1 Diagrama de paquetes

Dada la complejidad del diagrama de clases de los módulos desarrollados, y para su mejor entendimiento, el diagrama se agrupó por paquetes. En la Figura 8 se muestra cómo quedó el diagrama de paquetes de los módulos desarrollados.

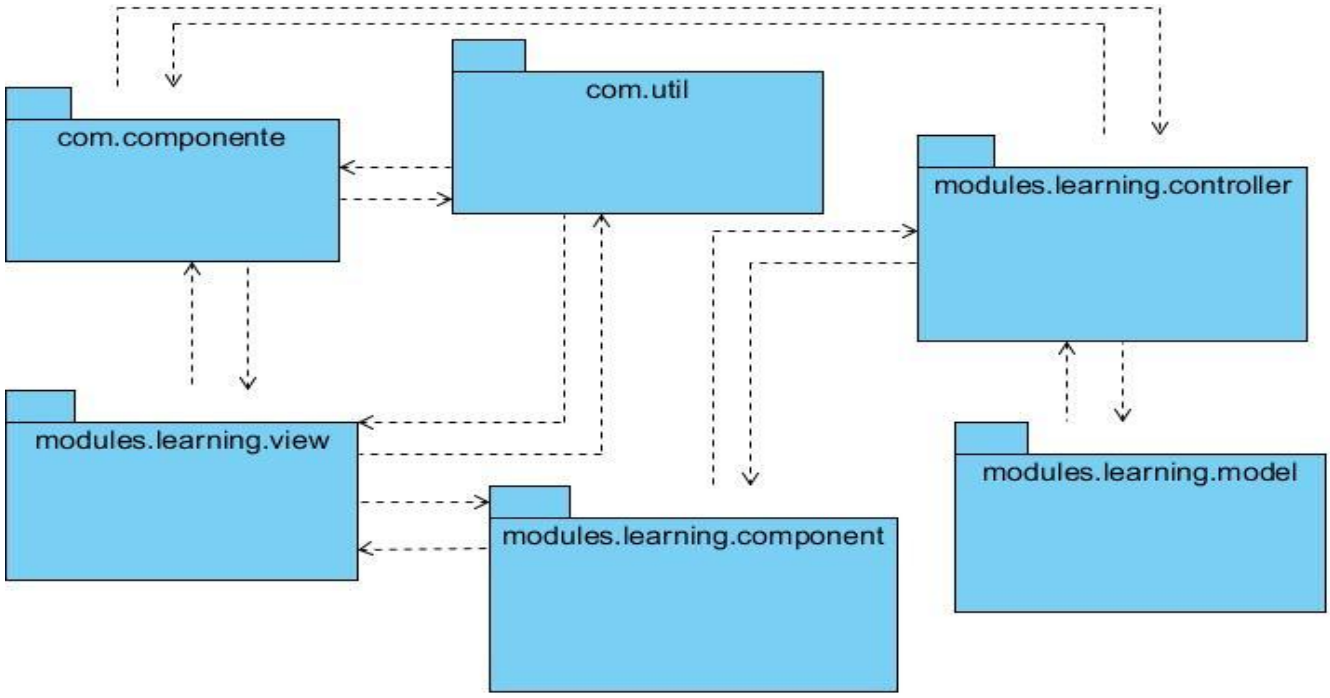

Figura 8. Diagrama de paquetes

A continuación se reflejará una breve explicación sobre el diagrama de paquetes mostrado anteriormente:

En el paquete **modules.learning.view** se encuentran las clases que serán mostradas en los módulos desarrollados, con las cuales el usuario tendrá acceso a las lecciones que se publiquen en dichos módulos. Para la construcción de las mismas, este paquete se apoya en otros de propósito general dentro del Laboratorio Virtual como son **com.component,** donde se encontrarán componentes de uso general para todo el laboratorio y **com.util**, es donde se encontrarán las clases de apoyo general para cualquier módulo del laboratorio en cuestión. También, las vistas de los módulos desarrollados harán uso de componentes propios, los cuales se encuentran dentro del paquete **modules.learning.component**. Cuando el usuario necesite alguna información de los componentes, estos accederán al paquete **modules.learning.controller,** los cuales contendrán todos los métodos necesarios para acceder a la información guardada en la base de datos a través del paquete **modules.learning.model**. Una vez que ya las clases controladoras obtengan la información de la base de datos, estas actualizarán los componentes que lo requieran y a su vez estos se actualizarán con la nueva información en la vista donde se encuentren.

3.2.2 Diagrama de clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema; también muestra sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, cuando se crea el diseño conceptual de la información que se manejará en el sistema, conjuntamente con los componentes que se encargarán del funcionamiento y la relaciones entre unos y otros. [13]

En el epígrafe siguiente se muestra el diagrama de clases de los módulos desarrollados para el caso de uso Gestionar Contenido el cual servirá como caso base de estudio a lo largo de este trabajo.

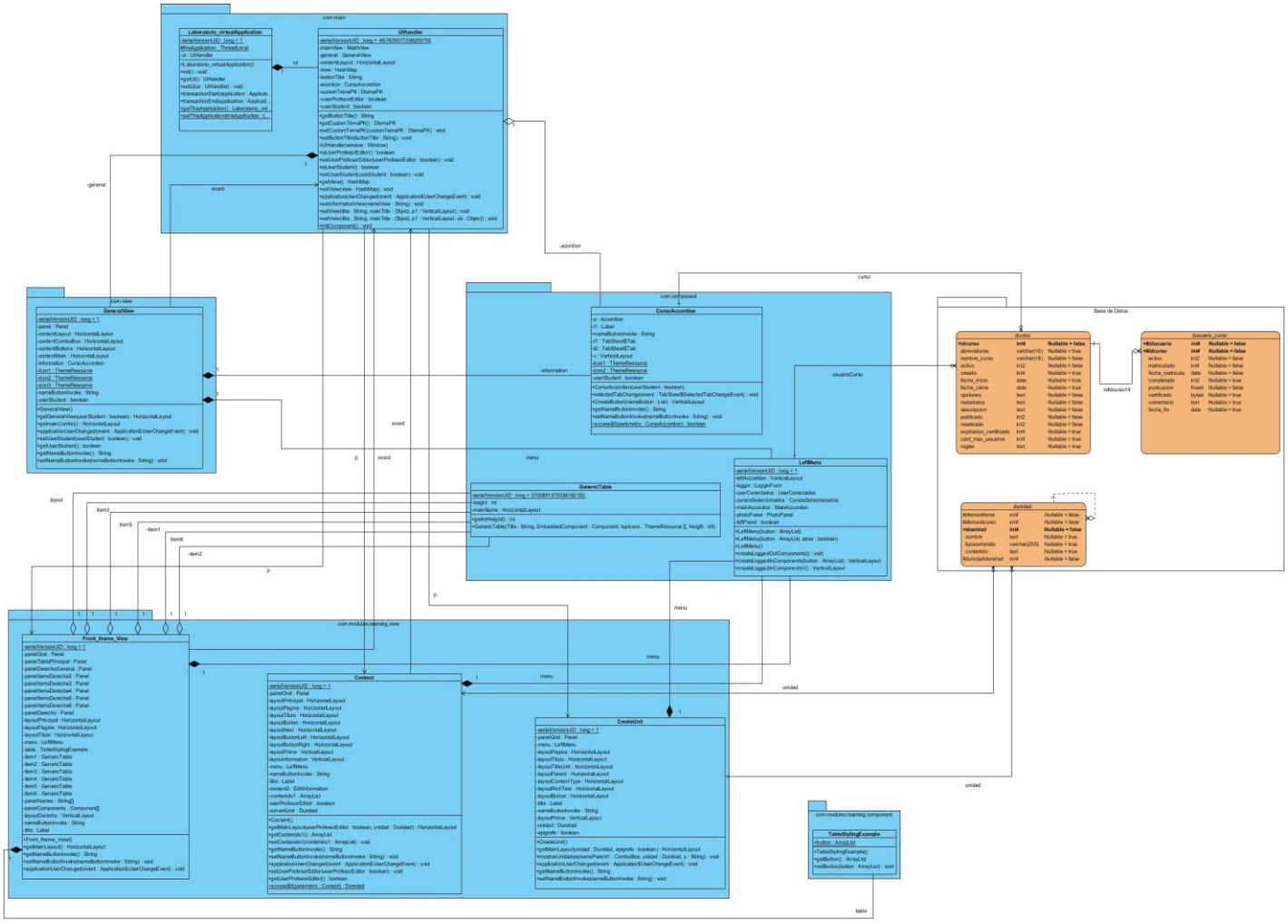

Figura 9. Diagrama de clases del diseño para el caso de uso Gestionar Contenido

3.2.3 Patrones de diseño propuestos

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. [30]

A continuación se abordarán los patrones de diseño utilizados en el sistema desarrollado para asegurar una solución mucho más confiable.

Patrones creacionales

- ✓ **Singleton (Instancia única):** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. [30]

En la figura 19 se muestra cómo fue utilizado dicho patrón en la solución propuesta.

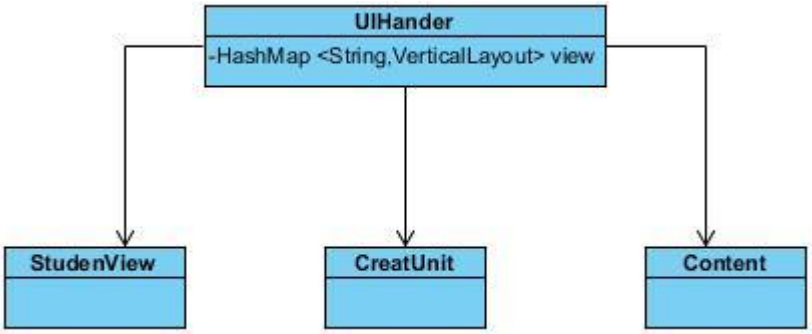

Figura 10. Aplicación del Patrón Singleton.

Patrones estructurales

- ✓ **Facade (Fachada):** Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema. [30]

En la figura 11 se muestra cómo fue aplicado dicho patrón en la solución propuesta.

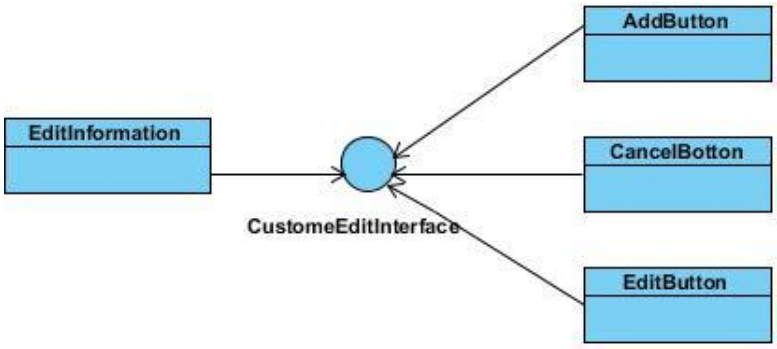

Figura 11. Aplicación del Patrón Facade

Patrón Creador

Este patrón es el encargado de que una clase B cree una instancia de una clase A, siempre que:

La clase B contenga a la clase A.

B sea una agregación (o composición) de A.

B almacene a A.

B tenga los datos de inicialización de A (datos que requiere su constructor).

B use a A.

Este patrón se utilizó en la implementación de las clases del Subsistema de Autoaprendizaje, ejemplo, la clase StudentView.

Alta Cohesión

La información que almacena una clase debe de ser coherente y estar, en la medida de lo posible, relacionada con la clase.

Este patrón fue utilizado en los módulos desarrollados cuando, por ejemplo, se implementó la clase controladora EditInformation, la cual se encarga de gestionar los datos de la información de una lección determinada, auxiliándose de la clase DtemaJpaController.

Bajo Acoplamiento

Es la idea de tener las clases lo menos ligadas entre sí posible de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización y disminuyendo la dependencia entre las clases. [30]

A continuación se muestra cómo fue utilizado dicho patrón en la solución propuesta.

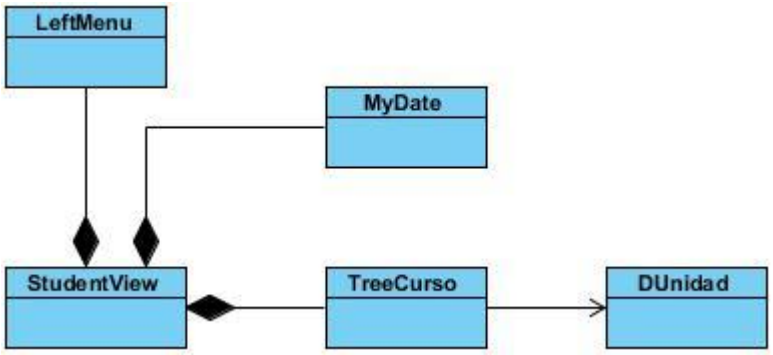

Figura 12. Aplicación del Patrón Bajo Acoplamiento

3.2.4 Diagramas de interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva modelar instancias concretas o prototípicas de clases interfaz, componentes y nodos, junto con los mensajes enviados entre ellos. Todo en el contexto de un escenario que ilustra un comportamiento. En el contexto de las clases se describe la forma en que grupos de objetos colaboran para proveer un comportamiento. Mientras que un diagrama de casos de uso presenta una visión externa del sistema, las funcionalidades de dichos casos de uso se recoge como un flujo de eventos, y se utilizan para ello, interacciones entre sociedades de objetos. [13]

A continuación se muestra una primera sección del diagrama de secuencia del caso de uso Gestionar Contenido, por lo complejo de este caso de uso el resto de las secciones se encuentran en el Anexo # 2.

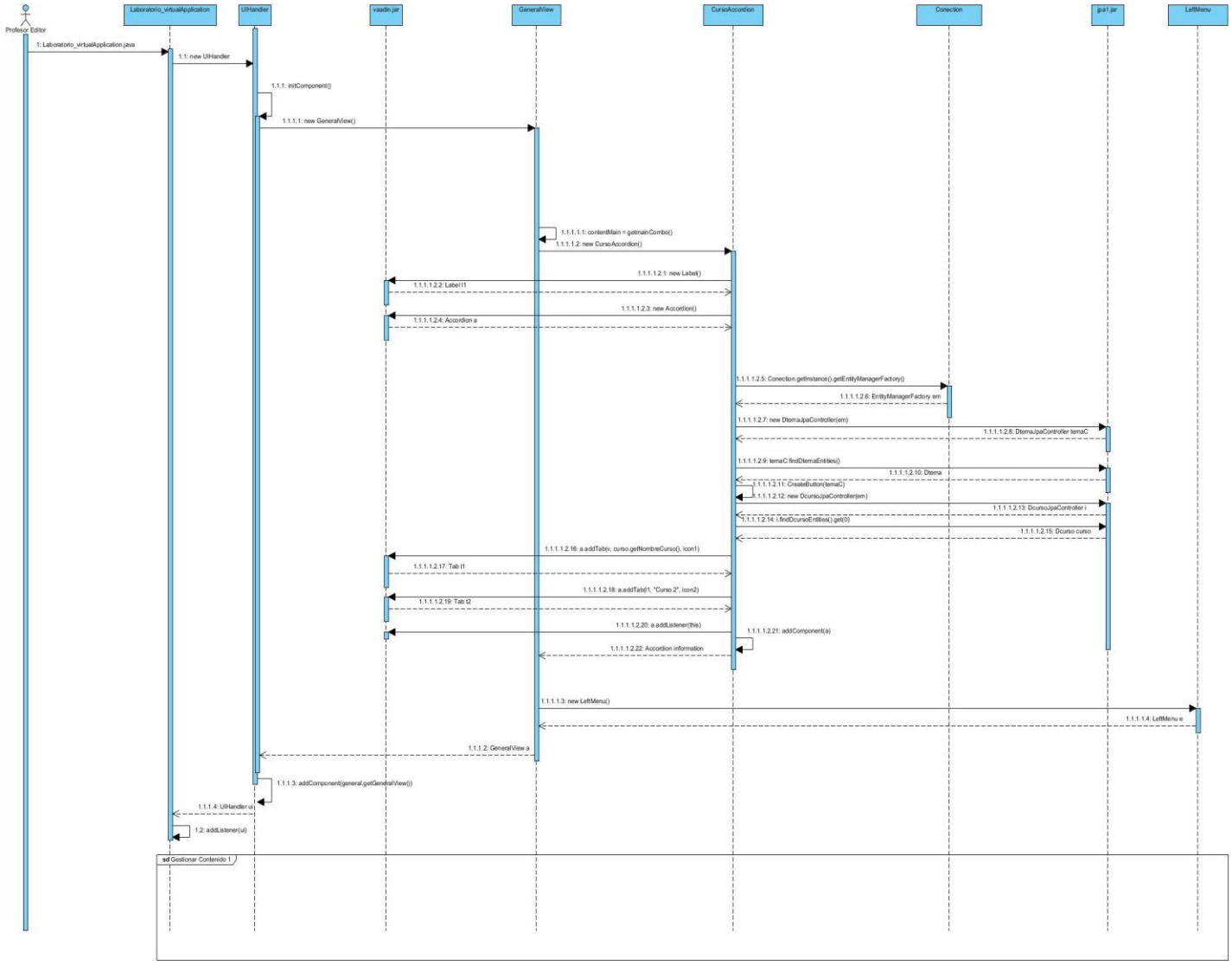

Figura 13. Diagrama de secuencia CU Gestionar Contenido

3.2.5 Modelo de datos

A continuación se muestra el modelo de datos modelado donde los módulos desarrollados guardarán los datos que gestionen dichos módulos.

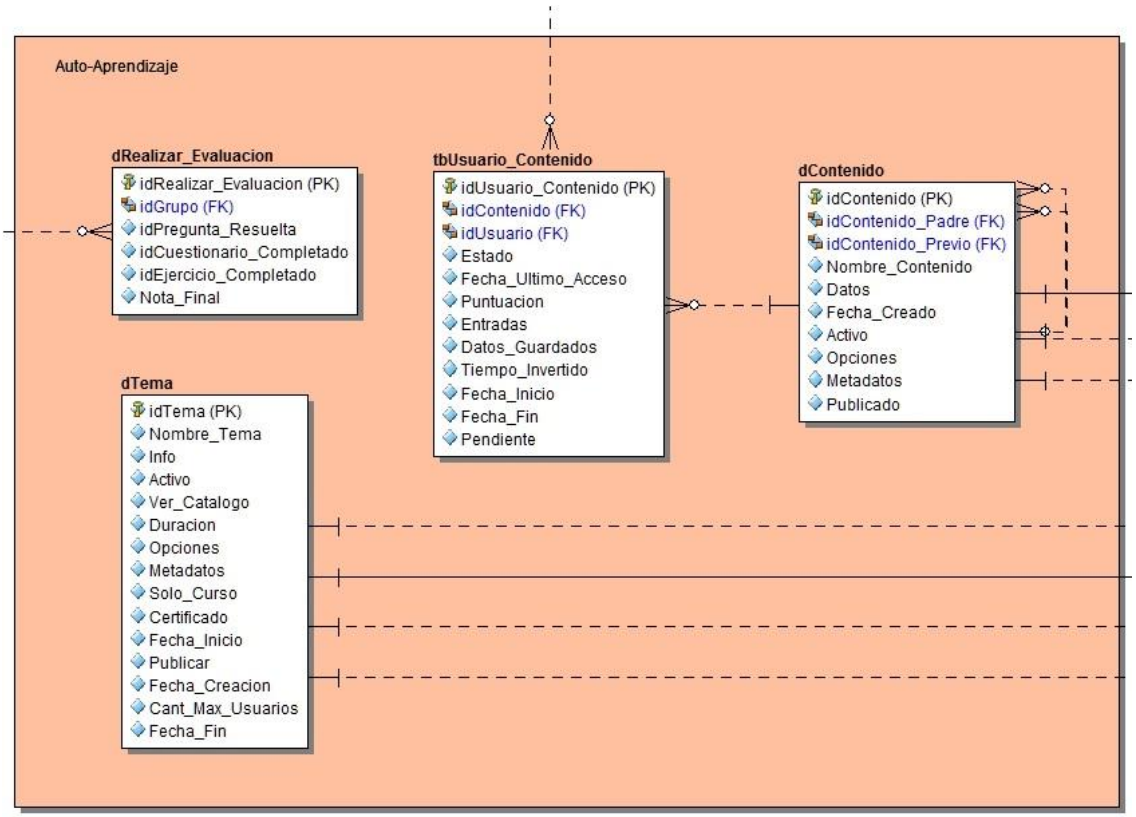

Figura 14. Modelo de datos.

La información que se gestiona en los módulos desarrollados en el presente trabajo comienza a partir de la tabla dTema, la cual le proveerá a dichos módulos los temas de la asignatura SO que con anterioridad se encuentra registrada en el sistema; a partir de ahí el usuario del Laboratorio Virtual, según su rol, podrá consultar el contenido o editarlo. Dicho contenido se encontrará almacenado en la tabla dContenido, además de que se guardará el estado del aprendizaje del usuario en cuestión, ya que a partir de que el usuario haga solicitud de un contenido en específico, se irá guardando en la tabla tbUsuario_Contenido y le servirá al profesor para saber en qué estado se encuentra un estudiante determinado y así poder realizar un trabajo más profundo en lo que a un tema en específico se refiere, o para saber cuándo un estudiante determinado necesita una atención diferenciada.

Después que un estudiante termine de estudiar un contenido en específico, terminará la lección realizando la evaluación correspondiente a la misma, la cual se guardará en la tabla dRealizar_Evaluación y podrá

ser consultada por él, en cualquier momento en la plataforma, o por cualquier profesor que tenga asignado el grupo del estudiante en cuestión.

3.2.6 Diagrama de despliegue de los módulos desarrollados

En la figura 15 se muestra el diagrama de despliegue que presenta el Laboratorio Virtual para la asignatura de Sistemas Operativos. Los módulos desarrollados pertenecientes al subsistema de autoaprendizaje serán desplegados en las PC clientes y desde ahí los disímiles usuarios del laboratorio podrán acceder a los mismos.

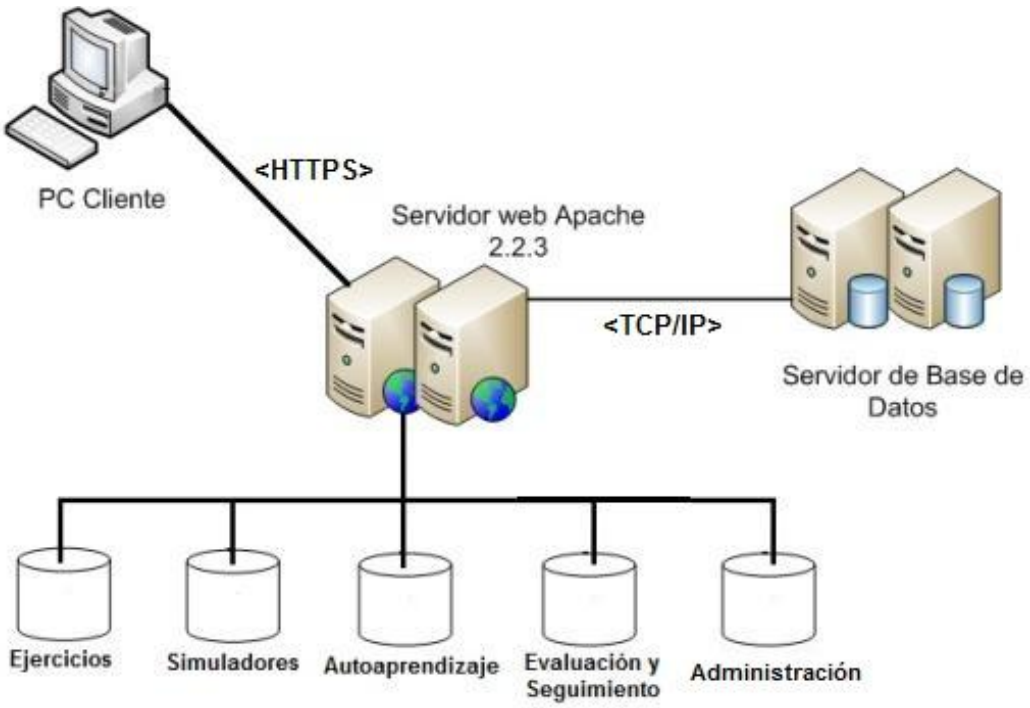

Figura 15. Diagrama de Despliegue

Conclusiones parciales

Mediante el desarrollo de este capítulo:

- ✓ Se obtuvo una primera aproximación de la solución para el desarrollo de los módulos de Administración de memoria y Comunicación entre procesos con el objetivo de apoyar la visualización de los contenidos referentes a estos temas.
- ✓ Con la revisión del diseño propuesto y de todos los artefactos generados se puede concluir que dicho diseño facilita y establece las bases para la implementación gracias a la buena estructura y organización que posee.
- ✓ Se logró un modelo de cómo quedará desplegada la solución propuesta para los módulos Administración de memoria y Comunicación entre procesos del Laboratorio Virtual.

Capítulo 4. Implementación y Prueba

Introducción

Este capítulo describe la implementación del sistema, mostrando los diagramas de componentes y todo lo referente a la solución desarrollada. Además, se muestra todo lo relacionado con la validación de la aplicación y demás artefactos obtenidos durante el desarrollo de las funcionalidades en cada una de las fases de desarrollo del trabajo.

4.1 Implementación

4.1.1. Diagramas de componentes

Los diagramas de componentes muestran las organizaciones y dependencias lógicas entre componentes de software, sean estos de código fuente, binarios, archivos, bibliotecas cargadas dinámicamente o ejecutables. [13]

Por la complejidad del sistema a elaborar, se consideró que no era factible una representación del diagrama de componentes de forma explícita. Así, se tomó la determinación de que éste fuera agrupado por paquetes, en concordancia con lo ya planteado anteriormente, al representar el diagrama de clases del diseño del sistema.

En la figura 16 se muestra el diagrama de componentes de los módulos Administración de memoria y Comunicación entre procesos.

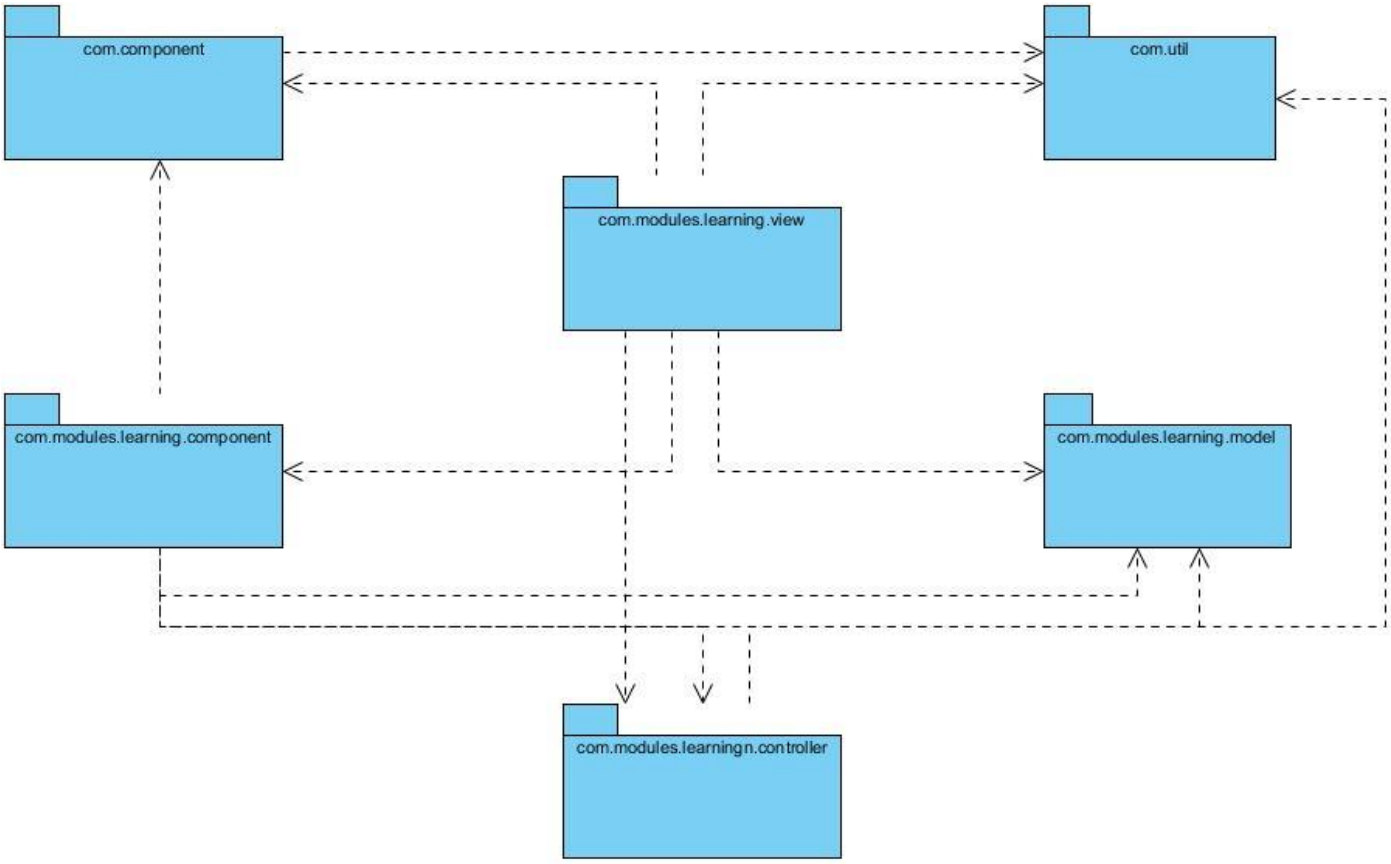

Figura 16. Diagrama de paquetes de los módulos desarrollados.

En la figura 16, se muestra el diagrama de paquetes de los módulos AM y CP.

Para la comprensión del mismo se explicará algunos paquetes que se consideran fundamentales para la implementación del sistema desarrollado. Se irán mostrando las partes del diagrama donde se evidencia el uso de dichos paquetes.

Jpa1: componente que le permite al sistema poder mapear las tablas de la base de datos, y así poder tratar más fácilmente con la gestión de los datos del laboratorio desarrollado. La utilización de este componente se puede apreciar en la figura 17, donde se muestra el diagrama de componentes del paquete com.modules.learning.component.

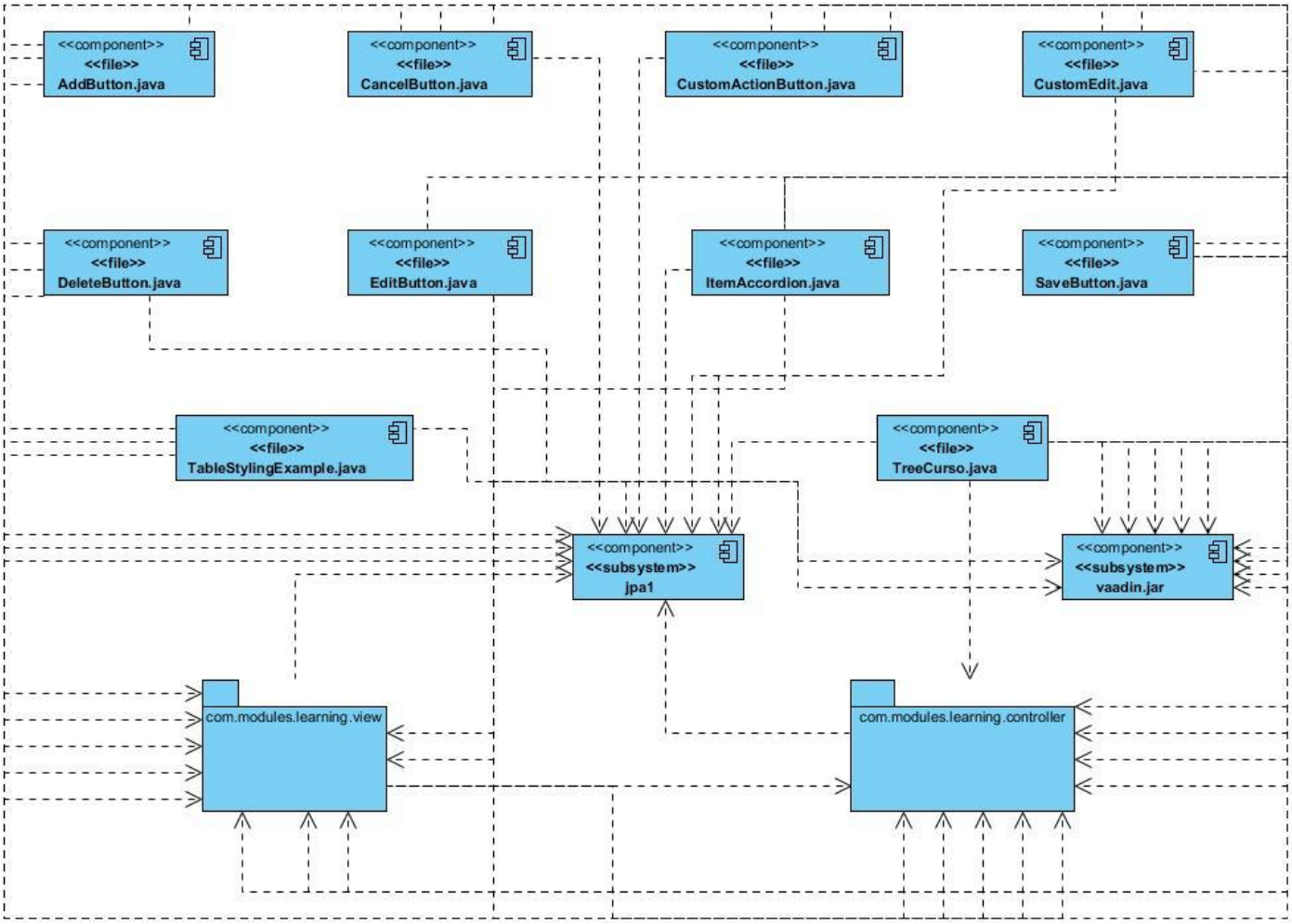

Figura 17. Diagrama de componentes del paquete com.modules.learning.component.

Vaadin: componente que le permite al sistema lograr una mejora en cuanto a interfaz de usuario se refiere, haciéndola atractiva a los ojos de los usuarios finales del laboratorio. La utilización de este componente se puede apreciar en la figura 18, donde se muestra el diagrama de componentes del paquete com.modules.learning.component.

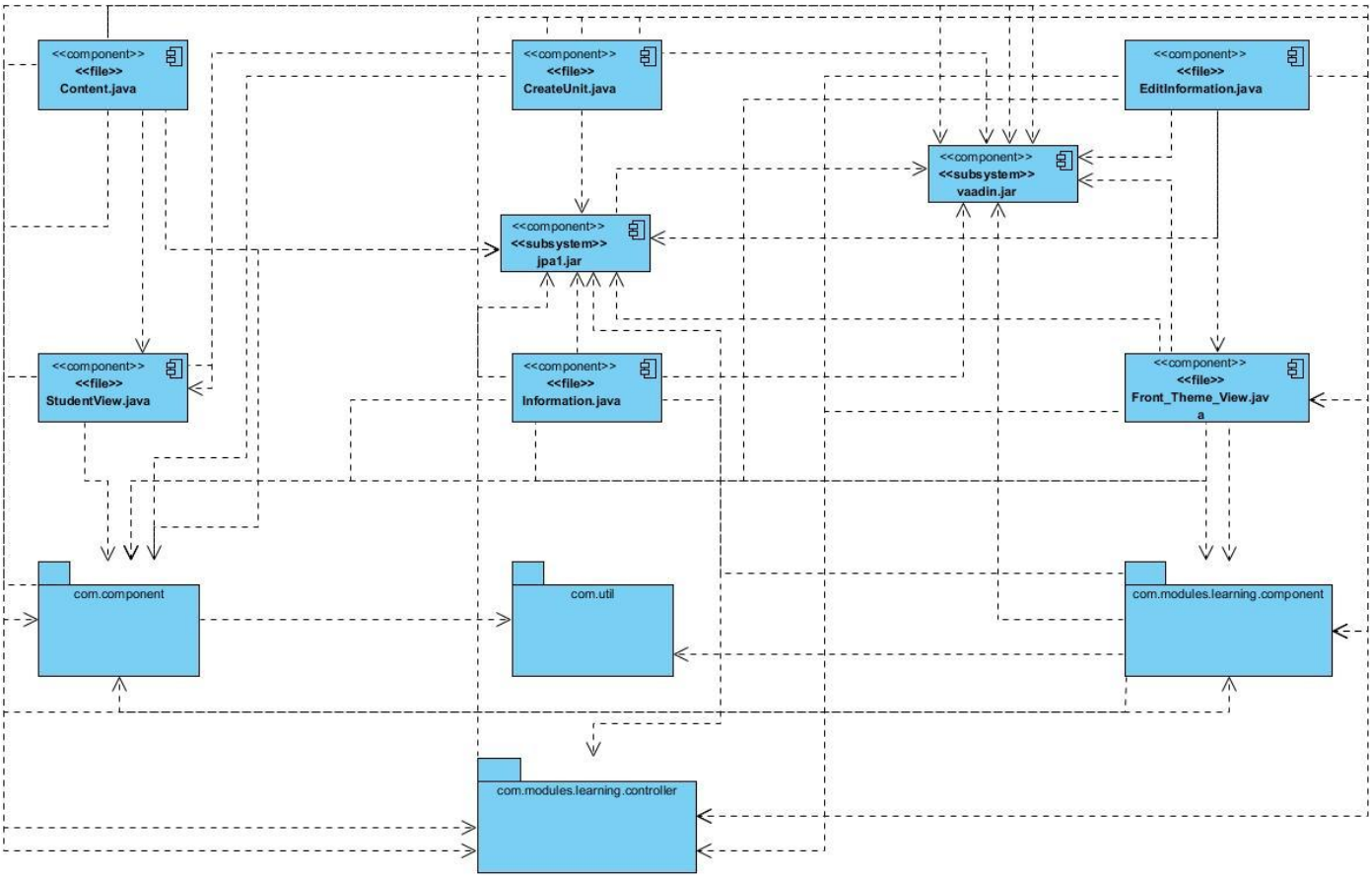

Figura 18. Diagrama de componentes del paquete com.modules.learning.view

4.2 Técnicas de validación de artefactos y pruebas al sistema

La validación conlleva una singular importancia en el proceso de desarrollo de software, logrando de manera paulatina que los desarrolladores tengan una idea de la calidad del trabajo realizado. Por lo general, para analistas y diseñadores es de vital importancia este proceso, ya que de ellos depende que a fases de desarrollo se llegue con las funcionalidades específicas que desea el cliente, así como de un diseño que haga posible que el sistema a implementar sea consistente, óptimo y sencillo. La ingeniería de software tiene definida varias métricas que permiten tanto la validación de los modelos del análisis como del diseño, entre ellas es válido destacar las métricas de la calidad de la especificación, y las métricas del modelo de diseño, que se desglosan en métricas de diseño de alto nivel, de interfaz, entre otras. El diseño también puede ser validado con la utilización de prototipos no funcionales de interfaz, haciéndole una

aproximación al cliente de cómo puede ser la aplicación una vez terminada, esclareciendo que estos pueden tener cambios en la versión final del producto [13]

Para la validación se tuvo en cuenta la opinión de los expertos:

1. Ing. Carlos Yasmany Hidalgo García.
2. Ing. Dailín Fernández Romero
3. Ing. Semidaris Batistas Verdugo.
4. Ing. Yordanys García Leyva
5. Ing. Pedro Manuel García Rodríguez

4.2.1 Métrica de la calidad de la especificación

Las métricas se aplican para valorar la calidad de los productos de ingeniería o los sistemas que se construyen, comprobar la efectividad y correspondencia de los artefactos generados atendiendo a las exigencias del cliente. Proporcionan una manera sistemática de valorar la calidad basándose en un conjunto de reglas claramente definidas y se aplican durante todo el ciclo de vida, permitiendo descubrir y corregir problemas potenciales. [31]

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

23 = 9 + 14

Dónde:

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 aplica la 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

0.96 = 22/23

Dónde:

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 = 0.96, esto demuestra que los requisitos se encuentran con un alto nivel de especificidad.

La estabilidad fue determinada con la fórmula donde Rm son los requisitos modificados, que es equivalente al número de requisitos. Se considera como valor óptimo para esta métrica el valor más cerca de 1.

Se sustituyen los valores:

E = (Rt – Rm) / Rt

E = (23 – 0) / 23

E = 1

El resultado obtenido muestra que los requisitos son estables, ya que se obtuvo el valor máximo de estabilidad basado en el siguiente rango:

Alta (0.90 <= E <= 1).

Media (0.80 <= E < 0.90).

Baja (0.7 <= E < 0.80).

4.2.2 Métrica 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. [31]

***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: *g*rado de claridad en la presentación de los elementos que describen el contexto y la claridad del sistema.

Para clasificar los resultados obtenidos se establecieron las siguientes reglas:

Alto (90% <= E <= 100%).

Medio (80%<= E < 90%).

Bajo (70%<= E < 80%).

Para conseguir una certera validación de los casos de uso del sistema se realizaron dos revisiones, a continuación se presentan los resultados obtenidos:

Para una primera iteración se obtuvo un 100% para la completitud, un 100% de consistencia, 75% de correctitud y un 75% de complejidad, mostrando un bajo nivel en los resultados de la correctitud y la complejidad por lo que fue necesario realizar una segunda iteración buscando elevar el grado de funcionalidad del diagrama de casos de uso.

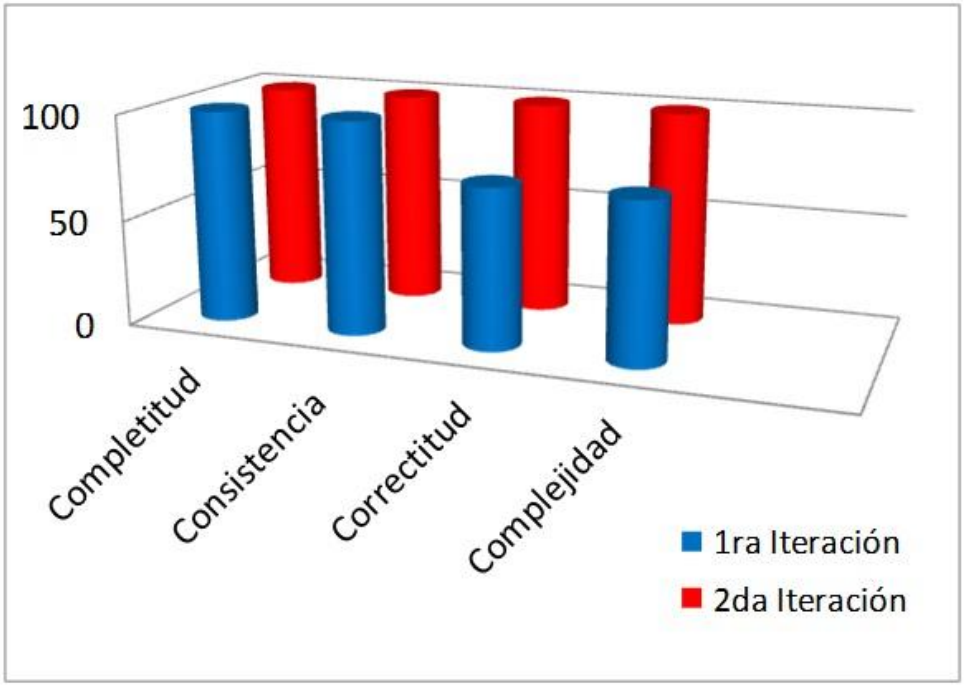

Figura 19. Gráfica de factores de la métrica para validar los casos de uso.

Resultados: como se puede apreciar en el gráfico anterior los resultados obtenidos en la segunda revisión fueron relevantes y con la aplicación de estas métricas se pudieron evaluar los factores

completitud, consistencia, correctitud y complejidad del diagrama de casos de uso del sistema lo que permitió demostrar que los requisitos identificados son implementados en al menos un caso de uso, abarcando de esta forma todas las necesidades expresadas por el cliente, donde los casos de uso fueron descritos detalladamente mostrando el flujo alterno separado del flujo básico lo que da una mayor legibilidad de los mismos, evidenciándose además que estos son iniciados por la interacción del usuario con el sistema.

Todo esto expuesto anteriormente permitió comprobar que el artefacto casos de uso del sistema se encuentra con la calidad requerida para entrar al flujo de trabajo implementación donde se comenzaría la realización del software.

4.2.3 Métrica para validar el diseño

- ✓ **Tamaño operacional de clase (TOC)**

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad: [32]

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: *c*onsiste en el grado de dificultad que tiene que 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.

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.

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

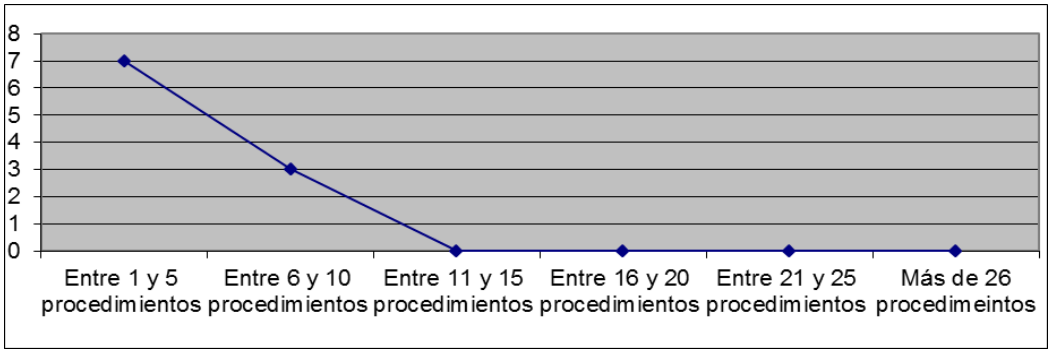

Figura 20. Representación de los resultados por intervalos definidos por procedimientos

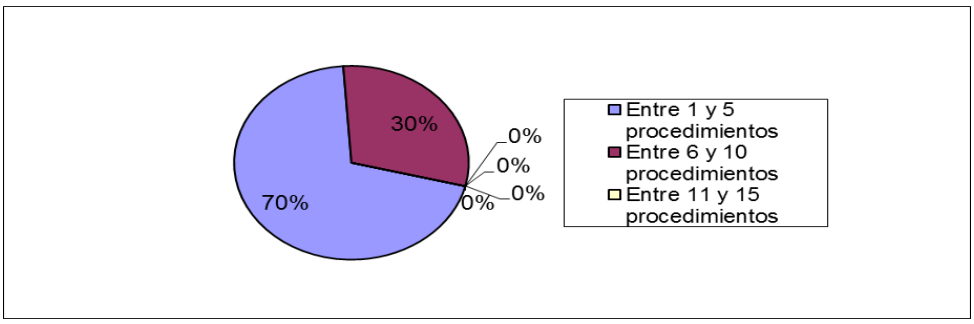

Figura 21. Representación de los resultados por intervalos definidos por porciento

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en cada uno de los atributos:

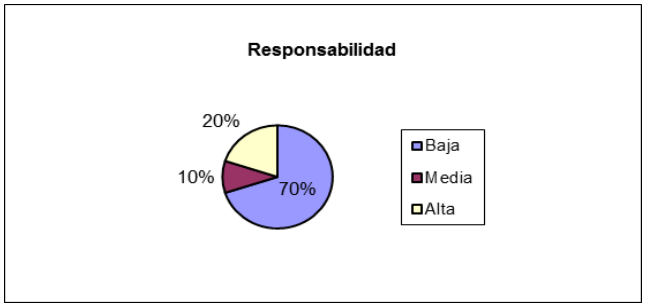

Figura 22. Resultados de la métrica TOC en el atributo Responsabilidad.

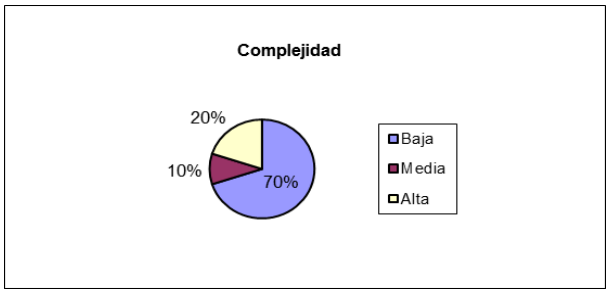

Figura 23. Resultados de la métrica TOC en el atributo Complejidad de Implementación.

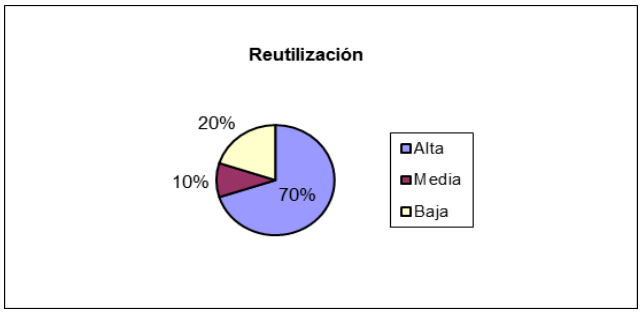

Figura 24. Resultados de la métrica TOC en el atributo Reutilización.

Al analizar los resultados obtenidos, luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases posee una menor cantidad de operaciones que la media registrada en las mediciones. Los atributos de calidad se encuentran en un nivel medio satisfactorio 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 lo que significa que Responsabilidad está representada a un: 70% (Baja), Complejidad de implementación a un: 70% (Baja) y la Reutilización a un: 70% (Alta).

4.3 Pruebas del sistema

La prueba de software es un proceso que corre en paralelo al proceso de desarrollo del software, usado para identificar posibles fallos de implementación, calidad, o usabilidad de un software. Se realiza por convencimiento de que todo sistema debe ser revisado con el objetivo de establecer el nivel de calidad requerido. Básicamente, es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas. Ellas se integran a las diferentes fases del ciclo del desarrollo del software y dentro de la propia Ingeniería. [13]

4.3.1 Plan de prueba

Un plan de pruebas está constituido por un conjunto de pruebas. Cada prueba debe dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad); cómo se mide el resultado; especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta) y definir cuál es el resultado que se espera. En lo adelante se abordará todo lo relacionado con las pruebas realizadas al sistema. [13]

4.3.2 Configuración del entorno de prueba

La configuración del entorno donde se vayan a ejecutar las diferentes pruebas que se realizan a un software es un aspecto muy importante dentro del proceso de pruebas, pues si no se analizan bien los recursos de software y hardware que necesita el producto que se está construyendo, a la hora de probarlo se prescindirá de los elementos necesarios para la ejecución de un proceso de pruebas exitoso. [32]
Por ello se tuvo en cuenta, a la hora de llevar a cabo todo el proceso de pruebas, algunos requerimientos de hardware y de software que hicieron posible un mejor desarrollo de las mismas, en aras de que se lograran minimizar los errores de la aplicación en desarrollo. Algunos de los requerimientos que se consideraron necesarios para las pruebas son los referenciados a continuación:

Requerimientos de software:

- ✓ Una PC con Windows XP o superior.
- ✓ Una PC con Linux (Ubuntu).

- ✓ Máquina Virtual de Java 7.

Requerimientos de hardware:

- ✓ PC con Microprocesador Pentium IV a 2.41 GHz o superior.
- ✓ Con 160 GB de Disco Duro.
- ✓ 512 MB de memoria RAM o superior.

Fecha de Inicio	Fecha de Fin	Actividades	Responsables
14/05/2012	14/05/2012	Aceptación y firma del plan de pruebas.	Liennys María González Torres, Lisandra Ramos Acosta, Carlos Yasmany Hidalgo García.
Primera iteración para los módulos desarrollados del Laboratorio Virtual			
23/05/2012	25/05/2012	Ejecución de las pruebas de caja negra al caso de uso Gestionar Contenido.	Liennys María González Torres.
27/05/2012	28/05/2012	Ejecución de las pruebas de caja negra al caso de uso Estudiar Contenido**.**	Lisandra Ramos Acosta.
29/05/2012	29/05/2012	Ejecución de las pruebas de caja negra al caso de uso Realizar Evaluación.	Liennys María González Torres.
30/05/2012	30/05/2012	Ejecución de las pruebas de caja negra al caso de uso Consultar Evaluación.	Lisandra Ramos Acosta.

30/05/2012	30/05/2012	Ejecución de las pruebas de caja negra al caso de uso Revalorizar Evaluación.	Liennys María González Torres, Lisandra Ramos Acosta, Carlos Yasmany Hidalgo García.

Tabla 4. Plan de pruebas para los casos de uso desarrollados en los módulos de Administración de memoria y Comunicación entre procesos para el Laboratorio Virtual.

4.3.3 Diseño de las pruebas de unidad (Caja Negra)

Caso de Uso "Gestionar Contenido"

Secciones a probar en el caso de uso.

Nombre de la sección	Escenario de la sección	Descripción de la funcionalidad	Flujo Central
SC1: Adicionar Contenido.	EC1.1: Adicionar Contenido.	El Profesor Editor escoge la opción crear unidad en la pantalla de contenido, luego de presionar el botón para crear la unidad le aparece la pantalla para adicionar la nueva unidad, en donde le aparecerán las opciones necesarias para crear una nueva unidad. Al terminar de llenar los campos necesarios el Profesor Editor presiona el botón salvar, y se regresa a la pantalla contenido donde se muestra la unidad creada con satisfacción.	Escoge el botón crear unidad en la pantalla de contenido.
SC2: Adicionar Epígrafe.	EC2.1: Adicionar Epígrafe.	El Profesor Editor escoge la opción crear epígrafe en la pantalla de contenido, luego de presionar el botón para crear el epígrafe le aparece la pantalla para adicionar el nuevo epígrafe, en donde le aparecerán las opciones necesarias para crear un nuevo epígrafe. Al terminar de llenar los campos necesarios el Profesor Editor presiona el botón salvar, y se	Escoge el botón crear epígrafe en la pantalla de contenido.

		regresa a la pantalla contenido donde se muestra el epígrafe creado con satisfacción.	
SC3: Editar Contenido.	EC3.1: Adicionar Epígrafe.	El Profesor Editor escoge la opción editar contenido en la pantalla de contenido, luego de presionar el botón para editar el contenido le aparece la pantalla para editar el contenido en cuestión, en donde le aparecerán las opciones necesarias para editar el contenido. Al terminar de editar los campos que desee, el Profesor Editor presiona el botón salvar, y se regresa a la pantalla contenido donde se muestra el contenido modificado con satisfacción.	Escoge el botón editar contenido en la pantalla de contenido.

Tabla 5. Diseño de las pruebas para los casos de uso desarrollados en los módulos de Administración de memoria y Comunicación entre procesos para el Laboratorio Virtual.

El diseño de las pruebas por cada sección definida anteriormente se encuentra en el Anexo # 3.

4.3.4 No conformidades detectadas

Luego de concluir las pruebas realizadas se detectaron las siguientes no conformidades.

Elemento	#	No conformidad	Aspecto Correspondiente	Etapa de detección	Signi fi.	No Signif.
Interfaz	1	Al presionar el botón para crear una nueva unidad el sistema no era capaz de mostrar la ventana correspondiente para poder crear la nueva unidad.	Cuando se presiona el botón crear unidad en la pantalla contenido.	Etapa de diseño y Aplicación de pruebas a los módulos desarrollados para el Laboratorio Virtual.	X	

Interfaz	2	Al presionar el botón para crear la una nueva unidad el sistema no era capaz de mostrar todos los componentes visuales necesarios para poder llenar todos los campos necesarios para la creación de la nueva unidad.	Cuando se presiona el botón crear unidad en la pantalla contenido.	Etapa de diseño y Aplicación de pruebas a los módulos desarrollados para el Laboratorio Virtual.	X	
Programación	3	Al presionar el botón salvar luego de haber creado una nueva unidad la aplicación reportaba que había salvado la nueva unidad satisfactoriamente pero no se guardaban los datos nuevos en la base de datos.	Cuando se presiona el botón salvar en la pantalla crear unidad nueva.	Etapa de diseño y Aplicación de pruebas a los módulos desarrollados para el Laboratorio Virtual.	X	
Interfaz	4	Al presionar el botón salvar luego de haber creado una nueva unidad la aplicación reportaba que había salvado la nueva	Cuando se presiona el botón salvar en la pantalla crear unidad nueva.	Etapa de diseño y Aplicación de pruebas a los módulos desarrollados para el	X	

		unidad satisfactoriamente pero no se visualizaban los datos nuevos en la pantalla correspondiente a la nueva unidad creada.		Laboratorio Virtual.		

Tabla 6. No conformidades detectadas para los casos de uso desarrollados en los módulos de Administración memoria y Comunicación entre procesos para el Laboratorio Virtual.

Luego de haber recogido las no conformidades mostradas en la tabla anterior, se realizó una segunda iteración donde se resolvieron las no conformidades anteriormente mencionadas.

Conclusiones parciales

En el capítulo que concluye se evaluaron y analizaron los resultados obtenidos, a partir de las pruebas realizadas al sistema, por lo que se puede decir que:

- ✓ Se logró implementar con éxito los módulos de Administración de memoria y Comunicación entre procesos para el subsistema de autoaprendizaje de SO para el Laboratorio Virtual.
- ✓ Con la aplicación de métrica de la especificación se logró comprobar que cada uno de los requisitos identificados en el proceso de levantamiento de requisitos da respuesta a un único caso de uso.
- ✓ Se analizó y validó el diagrama de casos de uso mediante la métrica orientada a objetos, lo que arrojó como resultado que el diagrama de casos de uso cuenta con 100% de funcionalidad.

Conclusiones generales

1. El estudio realizado sobre los Laboratorios Virtuales y todo lo concerniente al concepto del autoaprendizaje, permitió valorar la importancia de los mismos en el apoyo a la formación y consolidación de conocimientos y habilidades de los estudiantes.
2. Con la utilización de las técnicas de ingeniería de requisitos se capturaron, especificaron y validaron los requisitos del sistema, obteniendo la calidad necesaria en ellos, así como un elevado nivel de aceptación. Además, se logró comprender el sistema a desarrollar quedando claro todo lo referente de los procesos a automatizar.
3. El modelo de clases del diseño generado para la implementación de los módulos Administración de memoria y Comunicación entre procesos, permitió a los autores de este trabajo hacerse una idea precisa de cómo serían implementados los módulos concernientes para lograr así un funcionamiento óptimo de los mismos, lo cual se demuestra a través de la validación realizada a dicho modelo, dando paso así a la implementación de los módulos anteriormente mencionados.
4. Se logró implementar una primera versión de los módulos de Administración de memoria y Comunicación entre procesos para el Laboratorio Virtual que se desarrolla hoy en la Universidad de las Ciencias Informáticas. Además se validó la corrección de la implementación a través de los estándares de calidad existentes para la industria del software a nivel mundial.

Recomendaciones

1. Continuar la incorporación de nuevas funcionalidades al Laboratorio Virtual para la Asignatura de Sistemas Operativos para lograr una mejor forma de guiar el autoaprendizaje en los educandos de la Universidad de las Ciencias Informáticas (UCI).
2. Desarrollar nuevos métodos de enseñanza como la reproducción de audio-visuales, simuladores y otras vías para lograr una mejor comprensión del contenido de la asignatura para los módulos de Administración de memoria y Comunicación entre procesos, ya que el mismo puede resultar muy abstracto en ocasiones.
3. Implementar los módulos restantes en el Laboratorio Virtual que servirán de apoyo a los módulos de Administración de memoria y Comunicación entre procesos, en aras de lograr informatizar por completo los temas concernientes a este trabajo y así poder dar paso al desarrollo de nuevos módulos.

Referencias bibliográficas

[1] Á. Salaverría, L. F. Ferreira, J. Martínez, J.G. Dacosta y E. Mandado. Laboratorio virtual para el autoaprendizaje de la Electrónica Aplicada. [En línea]. Tecnologías Aplicadas a la Enseñanza de la Electrónica, 2006. [Consultado en: marzo, 2011]. Disponible en:

http://taee.euitt.upm.es/Congresosv2/2006/papers/2006SD108.pdf

[2]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: 16 de febrero de 2010.] http://www.latindex.ucr.ac.cr/edu31-1/edu-31-1-05.pdf.

[3] Gutiérrez Atoche, Egberto Serafín. Laboratorio Virtual de Física en el área de circuitos de Corriente Continua y Corriente Alterna. [En línea]. [Consultado en: marzo, 2011]. Disponible en: http://www.monografias.com/trabajos46/laboratorio-virtual-fisica/laboratorio-virtual-fisica.shtml

[4] Vary, James P. 2000. Informe de la reunión de expertos. Informe de la reunión de expertos. [En línea] 2000. [Citado el: 10 de febrero de 2010.]

http://unesdoc.unesco.org/images/0011/001191/119102s.pdf.

[5] Herreros, L. Rosado y J. R. 2005. Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física. [En línea] 2005. [Citado el: 5 de diciembre de 2010.] http://www.formatex.org/micte2005/286.pdf.

[6] Herías, Francisco Andrés Candelas. 2003. Propuesta de Portal de la Red de Laboratorios Virtuales y Remotos de CEA. [En línea] 27 de noviembre de 2003. [Citado el: 16 de febrero de 2011.] http://www.disc.ua.es/docenweb/Docs/PropuestaDePortal.pdf.

[7] 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. [En línea] 2007. [Citado el: 16 de febrero de 2010.] http://www.latindex.ucr.ac.cr/edu31-1/edu-31-1-05.pdf

http://www.latindex.ucr.ac.cr/edu31-1/edu-31-1-05.pdf.

[8] Novak, J. D. 1988. *Aprendiendo a Aprender.* s.l. : Ediciones Martínez Roca, S.A, 1988. Disponible en: http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=Evaluaci%C3%B3n

[9] S. Tanembaum Andrew. Sistemas Operativos modernos.

[10] León, Ramos Lianyi y Pulido, Yanelis. 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.

[11] Pérez González, Dairelis. Hernández Pérez, Grenia. Guía para realizar Ingeniería de Requisitos a la línea de productos informáticos que utilizan tecnología multimedia. Universidad de las Ciencias Informáticas. 2010

[12] Mendoza, María y Sánchez. Metodologías De Desarrollo De Software. Perú: s.n., 2004.

[13]Sanchez, María A. Mendoza. 2004. Metodologías De Desarrollo De Software. Metodologías De Desarrollo De Software - RUP. [En línea] 7 de junio de 2004. [Citado el: 11 de febrero de 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.htm.

[14] Orallo, Enrique Hernández. 2007. El lenguaje Unificado de Modelado (UML). El lenguaje Unificado de Modelado (UML). [En línea] 2007. [Citado el: 12 de febrero de 2010.]

[15] Jacobson Ivar, Booch Grady and Rumbaugh James. El Proceso Unificado de Desarrollo de Software. 2000.

[16] Visual Paradigm for UML. [En línea.] 2007. [Consultado en: marzo, 2011.]. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(MÍ)_14720_p/

[17] González, Jesús Alberto Silva. Tarea de Java. [En línea] 2009. [Citado el: 2 de 1 de 2011.]

[18] Tim Boudreau, Jesse Glick, Simeon Greene, Vaughn Spurlin, Jack Woehr. NetBeans IDE Release Information. *http://netbeans.org.* [En línea] 2010. [Citado el: 22 de 4 de 2011.]

[19] Beck Kent, GammaErich, Saff David [En línea.] [Citada 29 de septiembre 2011.] Disponible en: http://junit.sourceforge.org

[20] The Apache JMeter, [En Línea.] [Citada 7 de noviembre 2011.] Disponible en:

http://jmeter.apache.org/

[21] The SOA agenda, [En Línea.] [Citada 25 de septiembre 2011] Disponible en: www.alegsa.com.ar/Dic/framework.php

[22] Vaadin Ltd, [En línea.] [Citada 23 de abril 2012.] Disponible en:

http://vaadin.com

[23] SoftJasper, [En línea.] [Citada 12 de septiembre 2011.] Disponible en: http://jasperforge.org/projects/jasperreports

[24] Object Refinery Limited, [En Línea.] [Citada 21 de noviembre 2011.] Disponible en: http://www.jfree.org/jfreechart/

[25] Microsystems Sun [En línea]. Disponible en: http://java.sun.com/javaee/

[26] PostgreSQL Global Development Group, [En línea.] [Citada 4 de junio 2012.] Disponible en:www.postgresql.org

[27] IGP SQLServer. Información General del Producto SQLServer 2005. [En línea] http://www.microsoft.com/spain/sql/productinfo/overview/default.mspx.

[28] Apache Software Foundation, [En línea]. [Citada 22 de febrero de 2012;] Disponible en: httpd.apache.org

[29] "Applying UML and Patterns" Craig Larman. Ed Prentice Hall. Viernes, 14 de Diciembre de 2007

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

[31] Martinez, Villaverde, Julio Antonio. 2009. Un nuevo Front-End para la plataforma alasGRATO. Ciudad de la Habana : s.n., 2009.

[32] Urrutia, Rodríguez, Misael y Jiménez, Figueredo Lianni Yadira. 2010. Simulador para la asignatura Sistemas Operativos. Ciudad de la Habana : s.n., 2010.

Bibliografía

1. Ángel García-Beltrán, Raquel Martínez, José-Alberto Jaén, Santiago Tapia. La autoevaluación como actividad docente en entornos virtuales de aprendizaje/enseñanza. [En línea]. [Consultado en: mayo, 2011]. Disponible en: http://www.um.es/ead/red/M6/garcia_beltran.pdf

2. Ángel G. y Raquel M. AulaWeb: un sistema para la gestión, evaluación y seguimiento de asignaturas. [En línea]. [Consultado en: marzo, 2011]. Disponible en: http://www.dii.etsii.upm.es/documents/AulaWebIndustriaConFiguras.pdf

3. Espinoza Robles. Clase Flujo de Análisis. [En línea]. [Consultado en: junio, 2011]. Disponible en: http://www.slideshare.net/juliopari/13-clase-flujo-de-analisis

4. GRASP. [En línea]. Wikipedia.org, 2010. [Consultado en: marzo, 2011]. Disponible en: http://es.wikipedia.org/wiki/Grasp

5. Ingeniería de Software 1. Fase de Elaboración. Flujo de trabajo de Análisis y Diseño. 2008 - 2009.

6. Ortega González, Dainerys. Rodríguez Veitia, Reinaldo. Análisis y Diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas. Universidad de las Ciencias Informáticas. 2010.

7. Övergaard, Gunnar; Palmkvist, Karin, “Use Cases: Patterns and Blueprints”. 2004. Addison Wesley.

8. P. Fernández, A. Salaverria, A. Valdés y E. Mandado. Laboratorio virtual como conjunto de objetos de aprendizaje de los dispositivos electrónicos. Utilización como herramienta de autoevaluación. [En línea]. [Consultado en: marzo, 2011]. Disponible en:http://e-spacio.uned.es/fez/eserv.php?pid=taee:congreso-2008-1048&dsID=S2D03.pdf

9. Ramos León, Lianyi. Pulido Piña, Yanelis. Análisis de los módulos Planificación de Disco y Administración de Memoria de un Laboratorio Virtual de apoyo a la asignatura de Sistemas Operativos. Universidad de las Ciencias Informáticas. 2008.

10. Vary, James P. 2000. Informe de la reunión de expertos. Informe de la reunión de expertos. 2000.