Universidad de las Ciencias Informáticas

Facultad 3

Módulo de Administración para el Laboratorio Virtual de Sistemas Operativos

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Arianne Soriano Betancourt

Tutor: Ing. Carlos Y. Hidalgo García

La Habana, 2011- 2012

Declaración de autoría

Declaro ser la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Arianne Soriano Betancourt	Carlos Y. Hidalgo García
_____	_____
(Autor)	(Tutor)

Agradecimientos

Agradezco a todas las personas que de alguna manera han contribuid a mi formación como ingeniera y como persona, en especial a aquellas que han puesto su máximo esfuerzo en darme aliento y seguridad cuando más lo he necesitado.

Agradezco muy especialmente a mis padres bellos, Virgen y Cheo, que tanto empeño han puesto en ver estos resultados y que tanto se han esforzado por darme un futuro mejor. Me inculcaron tantos valores lindos que gracias a ellos hoy soy una mejor persona. Gracias papis, a ustedes los amo.

A mi hermano por ser mi compañero de la vida y por darme una sobrina tan linda, que tanto quiero y amo.

A mi cuñada Deyris que me alivia las preocupaciones al estar siempre pendiente de la salud de mis padres.

A mis amigos de la vocacional y de la UCI que tanto apoyo me han dado en los momentos más difíciles de mi vida estudiantil, a Yaimit, Sonia, Yordialis, a Diamy, Katy, Yami, Daniel y toda la familia de mi familia de amigos que me han recibido con el mismo amor que yo a ellos. Lisi, Vilma, Irma, Maribel. Los llevo en mi corazón y jamás los olvidaré.

A mi novio lindo por ser incondicional en todo momento, por darme la mano cuando creo que el mundo se me va acabar, por darme tanto aliento y tantas fuerzas. Te amo mi tonton.

A la familia de mi novio que siempre están llenos de buenas energías y esperanzas, a Laude, Martin, y Nana.

A mis familiares y amigos que siempre están pendientes de los problemas, dispuestos ayudar en lo que sea necesario y a los que han contribuido con mi formación.

Gracias a todos. Gracias por todo.

Dedicatoria

Dedico el presente trabajo a mis padres queridos que tanto empeño han puesto en ver estos resultados y que tanto se han esforzado por darme un futuro mejor.

A mi hermano por ser mi compañero de la vida y molestarse tanto conmigo.

A mi sobrina linda que tanto quiero y amo.

A mis amigos de la vocacional y de la UCI que tanto me han apoyado.

A mi novio lindo por ser incondicional en todo momento.

A mis familiares y amigos.

Resumen

La Universidad de las Ciencias Informáticas (UCI) fue creada en el año 2002 a partir de la idea futurista del Comandante en Jefe Fidel Castro Ruz de preparar un ejército de profesionales altamente capacitados, capaces de ayudar a la economía del país con el desarrollo de software.

A partir de la necesidad de incorporar a la formación de los estudiantes elementos prácticos que fortalezcan su preparación como futuros ingenieros informáticos y la necesidad del país de informatizar las diferentes ramas de la sociedad cubana, se decidió cambiar el modelo tradicional de enseñanza que presentaba la universidad por un modelo centrado en el aprendizaje, propiciando un aumento en la calidad de este, donde la semipresencialidad juega un papel fundamental y por ende, es el estudiante el principal responsable de su propio aprendizaje.

Una de las materias que se imparte en tercer año es la asignatura de Sistemas Operativos (SO), a la cual se le han hecho un conjunto de modificaciones, como es la creación de nuevos medios didácticos que ayuden y motiven al estudiante a su auto preparación, pues en el informe realizado por el colectivo de profesores de la asignatura de Sistemas Operativos de la Facultad III al concluir el primer semestre del curso 2011-2012, se refiere que los estudiantes se encuentran desmotivados por la complejidad de los temas que se imparten, además no se cuenta con los suficientes medios didácticos que les permita poner en práctica los conocimientos teóricos adquiridos en clases, por lo que surge la idea de crear un Laboratorio Virtual para la asignatura de Sistemas Operativos (LAVSO). Este laboratorio virtual debe estar compuesto por cuatro subsistemas y un módulo de Administración que es el encargado de configurar y administrar todo el sistema para un correcto uso por parte de los usuarios finales.

En el presente trabajo se obtuvo el módulo de Administración que será el encargado de gestionar los cursos, controlar el tiempo de apertura y cierre de los mismos, gestión de los grupos, roles, usuarios, perfiles y reglas de acceso. Para ello se identificaron, analizaron y especificaron los requerimientos y casos de uso del sistema. Además se generaron los artefactos correspondientes al diseño como son el diagrama de paquetes, modelo de datos, diagramas de clases y diagramas de secuencia. También se generaron los artefactos referentes a la implementación como son los diagramas de componentes y diagrama de despliegue, tributando a la obtención final del sistema. Finalmente se validaron los artefactos obtenidos con el objetivo de evaluar la calidad presentada por cada uno de ellos.

Índice de Contenido

Índice de Contenido

Índice de Figuras

Introducción

La Universidad de las Ciencias Informáticas (UCI) fue creada en el año 2002 a partir de la idea futurista del Comandante en Jefe Fidel Castro Ruz, de preparar un ejército de profesionales altamente capacitados, comprometidos con la Patria y capaces de ayudar a la economía del país con el desarrollo de software.

A partir de la necesidad de incorporar a la formación de los estudiantes elementos prácticos que fortalezcan su preparación como futuros ingenieros informáticos y la necesidad del país de informatizar las diferentes ramas de la sociedad cubana, se decidió cambiar el modelo tradicional de enseñanza por un modelo centrado en el aprendizaje, propiciando un aumento en la calidad de este, donde la semipresencialidad juega un papel fundamental y por ende, es el estudiante el principal responsable de su propio aprendizaje, además es vinculado a tiempo completo a la producción, contribuyendo de manera directa a la creación de software tanto para el mercado nacional como internacional. Este modelo brinda la posibilidad de perfeccionar el plan de estudio de la carrera de Ingeniería en Ciencias Informáticas con el objetivo de lograr una formación integral a través de la instrucción.

En la UCI actualmente, se dispone de un conjunto de herramientas y materiales de apoyo que contribuyen a la formación profesional de los estudiantes, entre los que se encuentra los medios audiovisuales como los canales de TV, que transmiten durante 24 horas video-conferencias que tributan al plan de estudio, los sitios y portales web como Inter-Nos que incorporan nuevas vías para la preparación integral del estudiantado, y el Entorno Virtual de Aprendizaje (EVA) como plataforma de autoaprendizaje que se encuentra 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 pues se ha convertido en una excelente herramienta de aprendizaje para el estudiante y de ayuda para el profesor, permitiendo la consolidación de conocimientos a partir de integración directa de los conocimientos teóricos con la práctica.

Una de las materias que se imparte en tercer año es la asignatura de Sistemas Operativos (SO), cuyo principal objetivo es brindar los elementos básicos a los estudiantes sobre el funcionamiento de los Sistemas Operativos, su arquitectura y la relación con los procesos que lo componen, la gestión de memoria y la gestión de entrada/salida.

Introducción

Esta asignatura no se ha quedado exenta de las modificaciones hechas en el plan de estudio, sino que se ha visto inmersa en un conjunto de cambios que responden al nuevo modelo de formación que se desarrolla en la universidad.

Además de materiales como las conferencias, guías de ejercicios y otros, se ha propuesto la creación de nuevos medios didácticos que ayuden y motiven al estudiante a su auto preparación, pues en el informe realizado por el colectivo de profesores de la asignatura de Sistemas Operativos de la Facultad III al concluir el primer semestre del curso 2011-2012, se refiere que:

"…*La falta de motivación en los estudiantes por lo difícil de algunos de los temas,…provoca que los estudiantes olviden fácilmente el contenido una vez pasado el mismo además de que el autoestudio les resulte muy complejo al contar solamente con los documentos de clases*…" También refieren que "…*El trabajo en el EVA, aun no es suficiente, pues no existe un seguimiento personalizado de los estudiantes por los profesores de la asignatura, a través de la plataforma*…" y expresan además que "…*La carencia de materiales didácticos en la asignatura conlleva a que los estudiantes no cuenten con los medios necesarios para aumentar la comprensión de los contenidos vistos en el aula así como que no cuenten con una forma de poner en práctica los contenidos teóricos que se imparten por parte de los profesores*." (1)

La situación anteriormente descrita ha permitido que surja la idea de crear un Laboratorio Virtual para la asignatura de Sistemas Operativos (LAVSO) que contribuya a la comprensión de sus temas y a la puesta en práctica de los contenidos teóricos por parte de los estudiantes. Este laboratorio virtual incorporará a su entorno un subsistema de Autoaprendizaje, uno de Ejercicios y uno de Simuladores que se integrarán de acuerdo a los temas por los que está compuesto un curso. Contará además con un subsistema de Evaluación y Seguimiento y un módulo de Administración. Este último deberá ser el encargado de configurar y administrar todo el sistema para un correcto uso por parte de los usuarios finales. En estos momentos ese módulo no está presente por lo que se necesita un mecanismo que permita gestionar el curso, controlar el tiempo de apertura y cierre del mismo, gestión de los grupos, roles, usuarios, perfiles y reglas de acceso.

Teniendo en cuenta toda la problemática descrita, se identificó el siguiente **problema de investigación**: ¿Cómo administrar los subsistemas de Evaluación y Seguimiento, Ejercicios y Autoaprendizaje del LAVSO de manera que permita dar seguimiento y control a las acciones que se realicen en los mismos?

Objetivo General: Desarrollar el módulo de Administración del LAVSO, contribuyendo que se le dé

seguimiento y control a las acciones que se realicen en los subsistemas de Evaluación y Seguimiento, Ejercicios y Autoaprendizaje del mismo.

Objeto de estudio: Proceso de desarrollo de software

Campo de acción: Análisis, Diseño e Implementación del módulo de Administración para el LAVSO.

Idea a Defender: Obteniendo el módulo de Administración del LAVSO se contribuirá a dar seguimiento y control a las acciones que se realicen en los subsistemas de Evaluación y Seguimiento, Ejercicios y Autoaprendizaje que conforman el mismo.

Objetivos específicos:

- Elaborar el marco teórico de la investigación.
- Describir las características del negocio y los requisitos de software.
- Realizar el diseño y la implementación del sistema.
- Validar la solución propuesta.

Para el cumplimiento de los objetivos específicos se plantean un conjunto de **tareas de la investigación**, dentro de las cuales se encuentran:

- Estudio de los principales laboratorios virtuales a nivel mundial y en Cuba.
- Estudio de la metodología de desarrollo, lenguaje de modelado, herramientas CASE, lenguajes de programación, framework e IDE de desarrollo a utilizar.
- Estudio de las herramientas a utilizar para el modelado, administración y manejo de la base de datos.
- Estudio de las etapas de Ingeniería de Requisitos.
- Estudio de los patrones de casos de uso y patrones de diseño.
- Elaboración del documento Especificación de Requisitos.
- Elaboración del documento Modelo de Sistema.
- Realización de los diagramas de secuencia, clases del diseño y modelo de datos.
- Implementar los requisitos de software identificados.
- Validar los artefactos obtenidos a través de la matriz de trazabilidad de requisitos a casos de uso, listas de chequeo a los documentos de Especificación de Requisitos y Modelo de Sistema, métricas orientadas a objetos para el tamaño de clase y casos de pruebas para el sistema.

Introducción

Métodos de la Investigación Científica empleados

Métodos Teóricos

Analítico –Sintético: utilizado para entender y resumir la situación que conlleva a la investigación, además para interiorizar y sintetizar el estudio que se realiza a lo largo del capítulo 1.

Histórico–Lógico: se emplea para realizar un estudio sobre el tema de los laboratorios virtuales en el mundial y en Cuba y valorar la necesidad de crear un nuevo sistema con estas características.

Método genético: se utiliza para determinar el campo de acción sobre el que se hará la investigación y los elementos que rigen su comportamiento.

Método sistémico: se emplea para determinar los componentes que conforman el objeto.

Modelación: utilizado para obtener los principales diagramas que se generan en el desarrollo de la investigación.

Métodos Empíricos

Entrevista: empleado para conversar con el cliente y lograr un entendimiento con este.

Análisis Documental: se emplea para estudiar y comprender las bibliografías consultadas a lo largo de la investigación.

Criterios de Expertos: se utiliza para intercambiar con los especialistas en el tema, esencialmente profesores de la asignatura de Sistemas Operativos y tomar datos relevantes.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

El presente capítulo tiene como objetivo abordar los aspectos teóricos fundamentales que facilitan la comprensión y el desarrollo de la solución que se desea proponer. Se mencionan los principales conceptos asociados a la investigación. De estos, se enuncian sus acepciones según diferentes autores, escogiendo la más conveniente para esta investigación. Se realiza un estudio de los principales laboratorios virtuales que existen en el mundo y en Cuba así como sus aportes a la pedagogía. Se proporciona un resumen de las herramientas a utilizar en la realización del sistema así como la metodología de desarrollo y el lenguaje de modelado escogido en la arquitectura.

1.2 Conceptos asociados

A continuación se enuncian varios conceptos que serán útiles para entender en qué consiste la investigación, de ellos se brindan diferentes acepciones según varios autores, tomando de ellas la que más se ajuste a las necesidades de la investigación.

1.2.1 Sistema operativo

Un **sistema operativo** es un programa que controla la ejecución de las aplicaciones en un computador y que actúa como mediador entre el usuario y el hardware de este. (2)

Un **sistema operativo** es el conjunto de programas que se encarga de gestionar el hardware de una computadora y establecer la relación que tiene con el usuario. Es considerado como uno de los elementos de software a bajo nivel más importante que funciona de puente entre el hardware, software y el usuario. (3)

Sistema operativo está definido como el programa que se encarga de manejar y distribuir el hardware que es usado por los diferentes programas que se ejecutan en una computadora, teniendo en cuenta las peticiones del usuario y las necesidades del propio computador, o sea, administra los recursos de hardware del sistema. Se encarga de controlar ordenadamente y por prioridad los recursos con que cuenta la PC y distribuirlos a los diferentes programas que compiten por ellos, facilitándole una interfaz al usuario para su uso y utilización. (4)

Se puede definir un sistema operativo como ese conjunto de software que controla el hardware con que dispone una PC. Proporciona al usuario una interfaz para interactuar con esta y a su vez se encarga de darle una adecuada distribución por prioridades a las peticiones que se realizan de acuerdo a sus propias necesidades o a partir de las peticiones realizadas por el usuario. Es el puente o intermediario entre hardware, software y usuario final.

Capítulo 1: Fundamentación Teórica

Además **Sistemas Operativos** es el nombre que recibe una de las asignaturas que se imparte en el quinto semestre de la carrera de Ingeniería en Ciencias Informáticas en la UCI, para la cual se desarrolla la propuesta de solución de la presente investigación.

1.2.2 Administración

La **administración** se define como la disciplina que define un conjunto de principios, ideas y acciones que se tienen en cuenta y se ejecutan para guiar y controlar las acciones de un grupo de personas. También se considera a la administración como el proceso que se encarga de planificar, organizar, dirigir y controlar las actividades utilizando para ello las técnicas necesarias. (5)

En los sistemas informáticos de gestión existe un concepto bien similar conocido como **módulo de administración**, el cual le permite a un sistema informático que funcione de forma ordenada, y segura. Dentro de sus funciones principales está controlar las operaciones que se realicen en el sistema como garantía a su seguridad, se encarga de definir a grandes rasgos las configuraciones generales del sistema y la integración entre sus partes, además de definir cómo estará estructurado. (6)

1.2.3 Laboratorio virtual

Un **laboratorio virtual** es considerado como el espacio digitalizado que ha sido concebido para desarrollar trabajos de investigación, estudiar y difundir información haciendo uso de las tecnologías de la información y las comunicaciones. (7)

Está definido un **laboratorio virtual** como un simulador de un laboratorio real donde los estudiantes mediante el uso de las tecnologías, podrán realizar actividades interactivas sobre un tema específico. Estos permiten que los estudiantes puedan desarrollar habilidades y ejercitarlas como forma de consolidación de conocimientos. (8)

Los **laboratorios virtuales** facilitan un modelo de enseñanza de forma virtual que propone desarrollar el trabajo independiente, ayuda a la creación de un espacio científico en el que se pueden realizar estudios de acuerdo al tema en cuestión permitiendo una mayor disponibilidad de la información tratada para lograr una mejor comprensión de esta. Son de gran importancia porque mediante la realización de simulaciones optimizan el proceso de aprendizaje del alumno. (9)

Un laboratorio virtual como bien se define en los conceptos anteriores es un espacio virtual que simula o representa el comportamiento de determinados eventos cotidianos, de la ciencia o fenómeno natural que facilita el aprendizaje, la ejercitación y la comprensión del tema que se estudia.

Capítulo 1: Fundamentación Teórica

1.3 Laboratorios virtuales en el mundo y en Cuba

Actualmente existen en el mundo muchos laboratorios virtuales de diferentes temas, de los cuales varios de ellos están publicados en la red de redes para que sean accedidos por todos los interesados. A continuación se mencionan algunos de los laboratorios virtuales más nombrados y utilizados en el mundo.

Laboratorio Virtual Ibercaja o bien nombrado (LAV) es un proyecto de obra social construido por la entidad con el mismo nombre, el cual brinda un espacio web con diferentes aspectos científicos. Da la posibilidad de acceder a aplicaciones didácticas como simuladores, en los que se abordan temas relacionados con el conocimiento del medio (Sistema Solar, Los Minerales), química (Formulación de Química Inorgánica, Moles y Disoluciones) y física (Movimientos, Leyes de Newton) etc. Cuenta con un espacio destinado a la investigación juvenil, donde se publican los trabajos realizados por los estudiantes. Divulga convocatorias de concursos en aras de fomentar el interés por la ciencia y la tecnología, y brinda cursos dirigidos a profesionales de la educación a diferentes niveles. (10)

Laboratorio Virtual de Física que también se encuentra publicado en la red de redes con espacios virtuales para el tema de Proyectiles, Teoría cinética de los gases, Oscilaciones y ondas, Óptica y Teoría de la Relatividad Especial, con una sección destinada a la práctica y experimentos sobre los temas mencionados anteriormente, y en los cuales se ejemplifica y se describe el comportamiento de determinados fenómenos físicos. (11)

Otro de los laboratorios virtuales existentes sobre el estudio de la física y la química es el **Laboratorio Virtual de Fisquiweb** que al igual que los mencionados anteriormente trata temas como la cinemática, la dinámica, ondas, energía entre otros. (12) También existen laboratorios de biología dirigidos a alumnos y profesores de Biología, tanto de bachillerato como universitarios, y constituye un instrumento de estudio y aprendizaje en el tema. Su objetivo es realizar prácticas y experimentos sobre el tema de biología molecular de forma simulada en el ordenador. (13)

En Cuba también se utilizan los laboratorios virtuales para mejorar el proceso docente educativo, ejemplificándose su utilización en el **Sistema de Laboratorios a Distancia en Asignaturas de Regulación Automática** en la Universidad Central de Las Villas Martha Abreu, que permite el ensayo de algoritmos de control de forma remota vía internet, y permite la realización de prácticas simuladas en un entorno web sin la necesidad de instalar otros software. Las prácticas que brinda el sistema han sido de gran utilidad en las asignaturas de pregrado como es el caso de Sistema de Control II, Modelado y Simulación. También ha tenido su utilidad en países como Brasil, México, España, entre otros. (14)

Otro ejemplo es el **Laboratorio Virtual de Anestesiología** aplicado a la especialidad de cirugía oral y maxilofacial que de una manera muy dinámica contribuye a la preparación de los estudiantes y

profesores interesados en el tema. Este software permite clasificar imágenes tomadas a partir de otros dispositivos como cámaras digítales y scanner. En estas imágenes se describen aspectos de interés para la preparación y entrenamiento de las personas que se encuentran en formación de las diferentes técnicas de anestesiología. Le permite a los especialistas del tema representar, editar, estudiar, procesar imágenes y gráficos sin necesidad de hacer inversiones mayores. (15)

Para la enseñanza de la informática también se emplean laboratorios virtuales tanto en el mundo como en Cuba, que dan muestra de la utilidad e importancia que tienen estos medios didácticos en la formación del hombre como futuros informáticos. Se pueden mencionar dentro de los más utilizados a nivel mundial el **Laboratorio Virtual para la Docencia de Redes de Computación** que fue una herramienta creada para elevar el rendimiento académico de los alumnos que reciben la asignatura de Redes de Computación de la carrera de Ingeniería en informática de la Universidad de Alicante, accediendo a este para realizar prácticas y experimentos que le permitan una mayor comprensión de la materia. (16) En Cuba un ejemplo de la aplicación de los laboratorios virtuales es el caso del **Laboratorio Virtual para la Enseñanza de la Arquitectura de Máquinas Computadoras** el cual se creó para que los estudiantes de la universidad de Camagüey desarrollen habilidades de forma continua en la simulación de microprocesador, con el objetivo de beneficiar el proceso docente-educativo, puesto que los estudiantes podrán observar diferentes procesos en la máquina en lugar de imaginarlos en el aula solo con la explicación del profesor. (9)

Después de hacer un estudio de la utilización de los laboratorios virtuales en el mundo y en Cuba se puede decir que no se identificó un sistema para la enseñanza de los Sistemas Operativos. Tampoco se identificó una herramienta lo suficientemente flexible que pueda ser adaptada al negocio del proyecto LAVSO.

1.4 Ventajas y desventajas del uso de los laboratorios virtuales

En el proceso de enseñanza-aprendizaje los laboratorios virtuales son muy importantes, pues facilitan el aprendizaje a los estudiantes. Dentro de las principales **ventajas** del uso de los laboratorios virtuales se encuentran:

- Sirven de apoyo al aprendizaje y la investigación de las personas que hacen uso de estos.
- Ayudan a atraer un mayor número de interesados a la experimentación.
- Los horarios de disponibilidad son más asequibles y más flexibles pues tienden a estar disponibles a tiempo completo y con capacidad para un elevado número de personas.
- Reduce enormemente el costo del ensamblaje y montaje de los laboratorios reales, siendo una alternativa al estudiante para practicar fenómenos como si fueran reales.

- Permite que el estudiante interactúe con este, modificando las variables de entrada y comparando los resultados obtenidos.
- No se corre el riesgo de sufrir algún daño con la puesta en marcha de los experimentos sino que se puede reiterar cuantas veces se necesite.

A su vez los laboratorios virtuales presentan desventajas que si no se les da un correcto tratamiento pueden convertirse en debilidades para el proceso enseñanza-aprendizaje. Dentro de las principales **desventajas** están:

- Nunca suplantará la interacción real de los estudiantes con los medios, objetos y dispositivos con que se desarrollan los laboratorios reales.
- En los laboratorios virtuales no se emplean elementos reales lo que trae como consecuencia que el interesado no le preste el suficiente interés y/o pierda la visión objetiva de los experimentos.
- Se recomienda que se realicen tareas ordenadas y escalonadas para que al finalizar la actividad puedan haberse vencido los objetivos trazados.
- Debe tener algún espacio para la descripción de los fenómenos que se manejan, pues el estudiante debe sentirse identificado con la actividad que se realizan, permitiendo su comprensión adecuadamente. (17)

1.5 Metodología de desarrollo

Las metodologías de desarrollo de software son consideradas un conjunto de pasos que siguen un orden lógico para obtener un sistema. Las metodologías tienen un fin común, que es lograr que se convierta el proceso de desarrollo de software en un proceso formal, con resultados claramente predecibles y que permitan obtener un producto con la calidad requerida en un tiempo establecido.

1.5.1 RUP

La metodología Rational Unified Process (RUP) es una metodología pesada pero muy flexible, de modo que permite adaptarla de acuerdo a las características del proyecto en que se emplee. Genera una amplia documentación que facilita el trabajo a los que la practican. Considerada como metodología que mejora el rendimiento del equipo de trabajo en cuanto a productividad, define claramente los roles, responsabilidades y actividades de todos los que intervienen en este proceso.

Es una metodología **dirigida por casos de usos** ya que todo lo que se desea construir en el sistema, o sea, los requisitos, son agrupados de acuerdo a sus características en casos de usos que son los que guían el proceso.

Es **centrada en la arquitectura** porque no existe un modelo único que abarque todos los elementos del sistema sino que existen múltiples modelos y vistas que definen la arquitectura de software de un sistema.

Es **iterativo e incremental** porque cuenta con cuatro fases que son: inicio, elaboración, construcción y transición, que a su vez están divididas en grupos de iteraciones que son nombradas flujos de trabajo que van añadiendo completitud al producto que se quiere obtener.

Los flujos de trabajos son: **modelo del negocio, requisitos, análisis y diseño, implementación, prueba, despliegue, gestión de configuración y cambios, gestión de proyectos y ambiente**.

La fase de **inicio** se encarga de lograr un entendimiento entre todos los interesados, o sea, clientes, usuarios y desarrolladores del software. El hito de esta fase son los objetivos del sistema. En ella se delimitan las necesidades del software, los elementos que van incluidos en el producto final y los que no, se estima el coste total, se planifica el proyecto y se analizan los principales riesgos.

La fase de **elaboración** se encarga de obtener una línea base de la arquitectura lo suficientemente estable para el desarrollo del sistema, se capturan los requisitos y se tratar los riesgos arquitectónicamente significativos.

La fase de **construcción** es la que se encarga de desarrollar el sistema. Su hito es la capacidad operativa del sistema. En ella se obtienen las funcionalidades que se desean y se trata de lograr la máxima calidad en cada versión completada en las diferentes iteraciones.

La fase de **transición** es la última que se realiza y en ella se tiene el producto listo para la entrega al cliente, o sea, el release del producto. También se enseña al usuario a utilizar el software (18)

1.6 Lenguaje de modelado

Los Lenguajes de modelados como su nombre lo indica son lenguajes que transmiten de forma gráfica un mensaje o una idea a través de gráficos, tablas y documentaciones. Estos generan artefactos que pueden ser representaciones gráficas o documentos explicativos. En la siguiente sección se hará referencia a UML como lenguaje de modelado.

1.6.1 UML

Unified Modeling Language o Lenguaje de Modelado Unificado (UML) es uno de los lenguajes de modelado visual que más se emplea en el mundo. Se utiliza para representar, visualizar, construir y documentar los diversos artefactos que se generan en la construcción de los sistemas informáticos. Ayuda a mantener un estricto control sobre los artefactos y permite organizar la información que surge a lo largo del ciclo de vida de este. Define un vocabulario y un conjunto de reglas que facilitan la comunicación. Brinda apoyo a los procesos orientados a objetos. (19)

Las herramientas que lo soportan pueden generar códigos para diferentes lenguajes pero en sentido general es un lenguaje de modelado y no de codificación. Es suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno.

Un modelo UML indica qué es lo que supuestamente hará el sistema, no precisamente cómo lo hará, pero constituye una guía y una representación de las especificaciones del sistema, ayudando a tomar decisiones para lograr un sistema que cumpla con todas las expectativas del cliente y los usuarios. Los diagramas que lo componen son los siguientes:

Diagramas de estructura estática: diagrama de casos de uso, diagrama de clases y diagrama de objetos.

Diagramas de comportamiento: diagrama de interacción, diagrama de secuencia, diagrama de colaboración, diagrama de actividad y diagrama de estados.

Diagramas de implementación: diagrama de componentes y diagrama de despliegue.

1.7 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering) son herramientas informáticas que apoyan y facilitan el proceso de desarrollo de software con el objetivo de aumentar la productividad, reducir el coste en tiempo de realización y costo de inversión. Se pueden utilizar de acuerdo a las características del proyecto como es el caso de Visual Paradigm.

1.7.1 Visual Paradigm

Visual Paradigm 5.0 es una herramienta CASE que está diseñada para modelar principalmente en lenguaje UML, siendo muy empleada en la construcción de software. Dentro de sus características se encuentra la interfaz de recursos céntricos, que permite el acceso a las opciones de modelado fácilmente, sin necesidad de desplegar el menú o ir a la barra de herramientas sino que permite a través de una interfaz cómoda y amigable construir diagramas de acuerdo a las necesidades que exija el producto. (20)

VP-UML se caracteriza por permitir la integración con entornos de desarrollo como Eclipse, que facilita la generación de código y la ingeniería inversa. Soporta un gran número de lenguajes de modelado como son UML, BPMN, SysML, ERD y más. Para representar diagramas complejos, VP-UML posee una funcionalidad que permite darle forma al diagrama solo con seleccionar la forma que se desee, se denomina disposición automática (auto layout). Facilita la generación de reportes a formatos como HTML, PDF, Word y otros. Permite importar diagramas que hayan sido creados por IBM Rational Rose y ERWin. Permite la generación de código a partir de los diagramas, soportando lenguajes como Java, C#, PHP, C++ y Python. Es una herramienta con licencia comercial y en la UCI se puede hacer uso de ella. (21)

Capítulo 1: Fundamentación Teórica

1.8 Lenguajes de programación

Los lenguajes de programación son considerados como un idioma artificial definido para ser interpretados y ejecutados por las máquinas computadoras, o sea, conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones para que puedan ser entendidas por los ordenadores. Dentro de los lenguajes de programación definidos en la arquitectura de software del LAVSO se encuentran Java Script como lenguaje del lado del cliente y Java como lenguaje del lado del servidor.

1.8.1 Java

Java es un lenguaje de programación muy utilizado en el mundo que permite desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra, permite crear programas que funcionan en un navegador web y en servicios web, desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, entre otros. Facilitan la creación de aplicaciones distribuidas ya que proporcionan un conjunto de clases para su práctica en software para red, que proporcionan la conexión con servidores. Es un lenguaje Orientado a Objeto. (22)

Este lenguaje fue diseñado para crear software altamente confiable pues realiza variadas comprobaciones en compilación y en tiempo de ejecución. Le facilita al programador el manejo del código ya que en este lenguaje se obvian los punteros. Ya que es un lenguaje multiplataforma Java genera con su compilador un formato diseñado específicamente para trasladar el código correctamente a múltiples plataformas. Java soporta la sincronización de varios hilos a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas y vale reconocer lo bien integrado que tiene el protocolo TCP/IP. (23)

1.9 Framework de desarrollo

Los framework de desarrollo se definen como una estructura de software que fue previamente definida genéricamente y que persigue la idea de agilizar el proceso de desarrollo de software, reutilizar un conjunto de código y aplicar patrones de diseño que están implementados. A continuación se describen algunos framework como JPA y Vaadin.

1.9.1 JPA (Java Persistence API)

La API (Application Programming Interface) de persistencia de Java (JPA) es un estándar de Java para el mapeo de objetos a una base de datos relacional. Muchos ORM (Object Relational Mapping) de Java como son Hibernate y TopLink utilizan la API de JPA para la prestación de sus funcionalidades. Cubre las entidades, utilizando un gestor de entidad, además permite crear y ejecutar consultas. La unidad básica de persistencia en JPA es la entidad, que no es más que una clase de Java con los

metadatos para describir cómo son los mapas de las tablas de bases de datos. Los metadatos pueden ser en forma de anotaciones en la clase de entidad en sí, o puede ser un archivo XML de acompañamiento, pero son más utilizadas las anotaciones, ya que son más fáciles de especificar y entender. Cada clase de entidad debe tener un marcador @ Entity y un campo de identificador, indicada por @ Id, que se asigna a la columna de clave principal en la base de datos.

Existe solo una unidad de configuración de JPA necesaria para conseguir que la aplicación se encuentre en marcha. Se basa en la noción de una unidad de persistencia, y se configura en un archivo llamado persistence.xml, que siempre se debe colocar en el directorio META-INF de su unidad de despliegue. Es un elemento de persistencia y puede contener una o más unidades de la persistencia de los elementos que representan diferentes configuraciones de ejecución. (24)

1.9.2 Vaadin

Vaadin es considerado una biblioteca de Java que está diseñado especialmente para la creación y mantenimiento de las interfaces de usuarios de alta calidad para la web. Se encarga de la gestión de la interfaz de usuario en el navegador y la comunicación AJAX entre el propio navegador web y el servidor. Con Vaadin no es necesario aprender ni depurar tecnologías web como son HTML y Java Script pues consiste en un marco de trabajo del lado del servidor, que tiene un motor del lado del cliente que se ejecuta en el navegador como un programa Java Script, permitiendo la interacción entre la interfaz de usuario y el servidor. Con aplicaciones hechas en Vaadin no se necesita de los plugins del navegador, facilitando el trabajo con marcos basados en Flash, applets de Java y otros plugins.

También cuenta con el apoyo de GWT (Google Web Toolkit) para una amplia gama de navegadores, por lo que el desarrollador no tiene que preocuparse acerca de la compatibilidad del navegador. Ofrece excelentes temas por defecto que por lo general no necesita hacer muchas modificaciones de personalización. Provee al desarrollador de un conjunto de librerías listas para usar de compontes de interfaz y un marco limpio para la creación de sus propios componentes. (25)

1.10 IDE de desarrollo

Los IDE son entornos de desarrollo integrados o bien conocidos también como Integrated Development Environment. Se clasifican como programas informáticos que se instalan y ejecutan en una PC para desarrollo y que permite la obtención de otras aplicaciones ya sean web como de escritorios. En la actualidad existen disímiles de IDEs de desarrollo pero en este acápite solo se hará referencia al Netbeans 7.1 RC1.

Capítulo 1: Fundamentación Teórica

1.10.1 Netbeans

Netbeans 7.1 RC1 es un IDE de desarrollo muy reconocido a nivel mundial por ser un proyecto de código abierto que le ha permitido mantener un constante crecimiento de la comunidad de usuarios que se sienten identificados con ella. Este importante entorno de desarrollo les permite a los programadores escribir sus códigos, compilarlos y ejecutar los programas que se obtengan como la integración de dichos códigos. Es un producto libre y gratuito que brinda la facilidad de hacer uso de sus funcionalidades sin restricción alguna. Es multiplataforma y presenta una interfaz muy cómoda e intuitiva que proporciona facilidad a los usuarios. Está diseñado para un gran número de lenguajes y se pueden obtener tanto aplicaciones web como de escritorio. (26)

En sus versiones más recientes Netbeans tiene integrado un sistema para examinar los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos, para acelerar la programación, propiciando un esqueleto para organizar el código fuente. Este editor conjuntamente integra los lenguajes como HTML, Java Script y CSS con completamiento de código para este último. Es importante destacar que desde este IDE de desarrollo se pueden llevar un correcto control de las versiones del proyecto gracias a su excelente integración con el SVN. (27)

1.11 Servidor Web

Un servidor Web según el término informático es un software que se encuentra escuchando las peticiones hechas por los clientes o usuarios y que son realizadas a través de un navegador web. Emplean el protocolo HTTP o el protocolo HTTPS para lograr la comunicación entre clientes y servidores mediante una conexión bidireccional y/o unidireccional así como síncronas o asíncronas, o sea, se encuentra en espera de una petición para dar respuesta a dicha solicitud. A continuación se relacionan las principales características del servidor web Apache 2.2.3.

1.11.1 Apache

El servidor Apache 2.2.3 es un servidor web HTTP de código abierto para diversas plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Tiene la característica de ser altamente configurable con bases de datos de autenticación. Es de código abierto y es multiplataforma, es muy fácil de conseguir y más aun de utilizar pues cuenta con la más variada documentación. Está soportado para varios lenguajes de programación como son PHP, Java, Perl y librerías ASP. Tiene soporte de host virtuales y servidor proxy integrado como manejo a la seguridad de los sistemas que maneja. Es libre.

Apache permite personalizar la respuesta ante los posibles errores que puedan suceder en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en el sistema, facilitándole al administrador llevar un mejor control de todo lo que sucede en el servidor. (21)

1.12 Herramienta de modelado de Base de Datos

Las herramientas de modelado de bases de datos como su nombre lo indica son herramientas que se emplean para diseñar las entidades de una base de datos y la relación entre ellas, permitiendo la representación gráfica de como quedarían relacionadas entre sí. La herramienta seleccionada para esta función en el LAVSO es ERStudio a la que a continuación se hace referencia.

1.12.1 ERStudio

El embarcadero ER / Studio en su versión 8.0.0.5865, es considerado una herramienta que se emplea en el modelado de los datos tanto lógico como físico, que brinda grandes facilidades para el diseño de esta, con interfaz cómoda y sugerente, ayudando a las empresas a realizar sus soluciones de diseño de bases de datos, también les permite documentarlas fácilmente. Brinda la facilidad a los arquitectos de datos de estudiar a profundidad las fuentes de datos existentes, así como diseñar nuevas bases de datos e implementarlas con alta calidad. El formato visual es altamente legible direccionando las necesidades comunes de los que administran las bases de datos además de facilitar el manejo a aquellos que desarrollan y dan soporte a aplicaciones con bases de datos muy grandes.

Esta herramienta presenta excelentes medios para un diseño lógico, sincronización directa de los diseños físicos y lógicos, creación automática de bases de datos y fácil creación de reportes. Permite la reingeniería inversa obteniendo la documentación precisa del modelado. Genera una amplia documentación soportada en HTML y un repositorio para el modelado realizado. Es muy útil en la toma de decisiones en caso de embotellamientos de datos. Genera diversos objetos de bases de datos como reglas, procedimientos y genera código fuente para la creación de bases de datos. (28)

1.13 Gestor de base de datos

Los sistemas gestores de bases de datos o también conocido como ODBC son unas herramientas informáticas que permiten la creación y mantenimiento de las bases de datos. Funciona como intermediario entre la base de datos, la aplicación y el usuario. Proporcionan mayor seguridad al establecer claves para identificar y asignar permisos según el tipo de usuario. A continuación se referencia como gestor de base de datos a PostgreSQL.

Capítulo 1: Fundamentación Teórica

1.13.1 PostgreSQL

PostgreSQL 9.1 es un gestor de Bases de Datos objeto –relacional que es muy conocido a nivel mundial en entornos de software libre por el conjunto de funcionalidades avanzadas que brinda y por soportar los estándares de SQL92 y SQL99, convirtiéndolo en tan bueno o mejor que otros gestores de bases de datos. Tiene licencia BSD lo que permite su uso, redistribución y modificación, bajo las condición de mantener el copyright. Es multiplataforma y presenta una API muy cómoda y amigable, disponible en varios lenguajes de programación.

Los mensajes de error pueden estar disponible en español y tiene soporte para vistas y procedimientos almacenados, claves foráneas, disparadores y subconsultas. Permite la definición de un nuevo tipo de tabla a partir de una previamente definida. Además cuenta con un rico conjunto de tipos de datos y su administración se basa en usuarios y privilegios. Cuenta con una comunidad muy bien organizada y estructurada que le facilita gran variedad de documentación. (29)

1.14 Cliente de base de datos

Los clientes de bases de datos son herramientas que permiten una mejor interacción con los gestores de bases de datos con el objetivo de minimizar errores durante la creación y administración de bases de datos así como la administración de usuarios y roles para el acceso a esta. A continuación se describe el PgAdmin III como cliente de bases de datos.

1.14.1 PgAdmin III

PgAdmin III 1.14 es considerada una herramienta de administración de datos para PostgreSQL de código abierto, que permite al programador la edición rápida de consultas, dando además soporte para todo tipo de objetos en PostgreSQL. Tiene la facilidad de ser empaquetado con el instalador del sistema operativo Windows y puede ser usado como cliente para administrar un servidor remoto en otro sistema operativo como Linux. Incluye en su centro de administración una interfaz gráfica, una herramienta que permite el trabajo y manejo del SQL además de un editor que permite la generación de códigos de procedimientos y funciones SQL. Puede ser empleada en las más simples bases de datos hasta las más complejas y grandes aplicaciones. La interfaz gráfica que presenta da soporte a las características que presenta el PostgreSQL y se facilita una administración cómoda y fácil. Está diseñado para más de 30 lenguajes y diversos Sistemas Operativos. (30)

1.15 Proceso de desarrollo de software

El proceso de desarrollo de software define a través de un proceso, las personas que están involucradas en un proyecto, utilizando un conjunto de herramientas que permite obtener como resultado final un producto de software.

Capítulo 1: Fundamentación Teórica

1.15.1 Ingeniería de Requisitos

Para lograr comprender qué es la Ingeniera de Requisitos (IR), es necesario definir qué es un requisito y cuál es su función. A continuación se mencionan algunos de los conceptos existentes actualmente:
Es una condición o una capacidad que debe encontrarse presente en un sistema o parte de sistema para cumplir con un contrato, estándar, especificación u otro documento formal. (31)
Es una descripción de las prestaciones que debe brindar el sistema y las restricciones que este debe cumplir. Representa la necesidad que tienen los clientes en solucionar determinado problema. (32)
Los requisitos de software son imprescindibles en la construcción de un software pues definen las características que deben incluirse en un software y las restricciones que lo rigen. Dentro de las etapas que conforman la disciplina de la IR están:

Elicitación de requisitos: es la primera de las etapas que conforman la Ingeniería de requisitos y es donde se extrae de todas las fuentes existentes la información que se necesita para identificar cual es el problema a resolver. Se pueden aplicar las técnicas de **Entrevista** y **Tormenta de Ideas**.
Análisis de requisitos: se realiza un análisis de cada uno de los requisitos a ver si tienen ambigüedades y si son consistentes. En esta etapa se agrupan los requisitos en correspondencia con las características de las funcionalidades. Se puede aplicar la técnica de **Casos de Uso**.
Especificación de requisitos: en esta etapa es donde se describen las funcionalidades que fueron identificadas y analizadas anteriormente. Puede hacerse mediante un documento o una representación gráfica que explique cómo se comporta el sistema.
Validación de requisitos: es donde el analista se encarga de verificar la calidad de cada uno de ellos, asegurándose que estén sin ambigüedades, sin inconsistencias, sin omisiones y que los errores detectados anteriormente hayan sido corregidos. Se puede aplicar la técnica de **Prototipos**.
Gestión de requisitos: se encarga de darle una correcta identificación, seguimiento y control a los requisitos a lo largo del ciclo de vida del sistema. (33)

1.15.2 Patrones

Un patrón es un problema/solución que estandariza principios y sugerencias relacionadas frecuentemente con la asignación de responsabilidades.

Patrones de Casos de Uso

Para desarrollar los diagramas de casos de uso se emplean una serie de patrones que facilitan la representación de estos. Dentro de los patrones de casos de uso más importantes están:

Regla del negocio: se centra fundamentalmente en la extracción de información relacionada con el cliente (políticas, reglas y regulaciones del negocio) que luego son tratadas como reglas del negocio.
Concordancia:
Reusabilidad: es separar en un caso de uso la sub-secuencia de acciones que son obligatorias para dos o más de un caso de uso.
Adición: es separar en un caso de uso la sub-secuencia de acciones que son alternativas a las funcionalidades de otros casos de uso.
Especialización: es la representación de una herencia pero enmarcada en casos de uso. Varios casos de uso (hijos) son del mismo tipo de un (padre) con funcionalidades genéricas.

CRUD:
Completo: representa a un conjunto de funcionalidades que engloban relación. Maneja operaciones como creación, lectura, actualización y eliminación. Por lo general son llamados CRUD o Gestionar.
Parcial: son similares a los CRUD Completos, pero separan una funcionalidad por determinadas características que la hacen compleja o tediosas.

Múltiples actores:
Roles diferente: consiste en un caso de uso y por lo menos dos actores. Es utilizado cuando dos actores juegan diferentes roles en un caso de uso, o sea, interactúan de forma diferente con el caso de uso.
Rol común: se aplica cuando dos actores juegan un mismo rol sobre un caso de uso. Representa la herencia entre actores. (34)

Patrones de diseño

Los patrones de diseño son propuestas para solucionar problemas surgidos con frecuencia en el proceso de desarrollo de software, especialmente en la etapa de implementación.
En la actualidad se emplean diversos patrones de diseño en la implementación de un sistema, que facilitan el desarrollo de estos. A continuación se describen algunos de ellos como son los patrones GoF **(Gant of Four)** y GRASP **(General Responsibility Assignment Software Patterns)**.

Patrones GRASP
Experto: es el encargado de asignar la responsabilidad de crear un objeto o la implementación de un método a la clase que conoce toda la información necesaria para crearla.
Creador: es el encargado de identificar quien debe ser el encargado de la creación de un nuevo objeto o clases. Debe tener toda la información necesaria para la creación del objeto.

Controlador: este patrón sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de manera que es la que recibe los datos del usuario y los envía a las distintas clases según el método que se esté llamando.

Alta cohesión: este patrón plantea que lo que se debe almacenar en una clase debe ser lo más coherente y debe estar bien relacionada con la propia clase.

Bajo acoplamiento: sugiere que se debe tener la menor cantidad posible de dependencia entre clases de manera que al producirse un cambio no se vean afectada las demás.

Patrones GoF

Patrones Creacionales: se encargan de la inicialización y configuración de objetos. Pueden ser: Abstract Factory, Builder Factory Method, Prototype, Singleton.

Patrones Estructurales: separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes. Pueden ser: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy.

Patrones de Comportamiento: más que describir objetos o clases, describen la comunicación entre ellos. Pueden ser: Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor. (35)

Se recomienda de manera general utilizar los patrones de diseño en caso que se tenga total dominio del problema pues un mal uso de estos puede ocasionar disminución de la capacidad de entendimiento de un diseño o de la implementación, al igual que con la cantidad de código que se genere, todo depende del uso que se le dé. Una vez dominados los patrones de diseño, pueden servir de gran ayuda para incorporar calidad al producto que se desarrolla, al igual que valor agregado al mismo.

Modelo Vista Controlador (MVC)

El MVC es un patrón de arquitectura de software que separa las diferentes capas en la que estará organizada un software como los datos, la interfaz de usuario y la lógica del negocio en tres componentes diferentes. Su uso es muy frecuente en aplicaciones web donde la vista es la página mostrada al usuario en HTML, el modelo es el sistema que integra con la Base de Datos y la lógica del negocio que lo soporta; y el controlador es el encargado de recibir los eventos y datos que provienen de la vista. (36)

Arquitectura en tres capas

La programación por capas es una arquitectura cliente-servidor en el que se separa la lógica del negocio de la lógica del diseño. Tiene gran utilidad en el mundo del desarrollo web pues permite llevar

varios niveles, facilitando el trabajo una vez que sea necesario realizar algún cambio mediante la revisión del nivel requerido. También permite la distribución del trabajo por capas, siendo suficiente con el conocimiento de la API por capas para relacionarlas entre ellas. En la arquitectura en tres capas se le confiere a cada nivel una misión diferente, lo que permite un diseño de la arquitectura escalable, pudiéndose ampliar en caso que sea necesario. (35)

1.15.3 Estándares de codificación

Los estándares de codificación son de gran utilidad para el desarrollo de una aplicación pues facilitan el mantenimiento del software una vez que se requiera de algún cambio. Permite que cualquier programador entienda las funcionalidades de la aplicación sin necesidad de contactar con su autor original ya que frecuentemente el software no es usado por su creador. Además los estándares permiten mejorar la legibilidad del código y su comprensión rápidamente. Algunos de los estándares de codificación en java son: deben evitarse ficheros con más de a 1000 líneas de código, para evitar un tamaño exagerado de estos. Cada fichero fuente debe contener una única clase o interfaz pública. La longitud de línea no debe superar los 80 caracteres. Las variables locales se inicializan en el momento de su declaración. Los métodos se separan entre ellos por una línea en blanco. Los nombres de paquetes se escriben con minúscula para evitar conflictos con los nombres de clases, etc. (37)

1.16 Conclusiones parciales

En este capítulo se compararon los conceptos dados por los diferentes autores sobre los términos más importantes para el desarrollo de la investigación. Se relacionan los principales laboratorios virtuales usados a nivel mundial y en Cuba y las ventajas y desventajas de aplicarlos. También se realizó un estudio de la metodología y las herramientas necesarias para obtener el software. Como resultado del estudio hecho de los elementos anteriormente mencionados se llegó a las siguientes conclusiones:

- Partiendo de los principales conceptos asociados a la investigación se plantearon las definiciones teóricas que se utilizarán.
- Como resultado del estudio realizado sobre los laboratorios virtuales más utilizados y sus principales características se concluyó que son de gran utilidad para el proceso docente educativo.
- No se identificó un laboratorio virtual en Cuba ni en el mundo para el tema de los Sistemas Operativos.
- Las potencialidades con la que cuentan las herramientas, lenguajes y metodologías a utilizar facilitará el desarrollo del módulo.

Capítulo 2: Características del Sistema

2.1 Introducción

El presente capítulo muestra una parte de la solución que se propone a través de la realización del Modelo de Dominio y el levantamiento de requisitos del módulo de Administración del LAVSO. Se hace una representación gráfica de los conceptos que se manejan en el módulo y la relación entre ellos, evidenciados en el modelo de clases del dominio. Posteriormente se mostrará las funcionalidades y restricciones del sistema que se obtuvieron a partir del modelo del negocio previamente definido. Por último se obtienen los artefactos necesarios para conformar el modelo del sistema (Especificación de Requisitos de Software, Diagrama de Casos de Uso del Sistema y su descripción).

2.2 Modelo del Dominio

El modelo de dominio es un artefacto generado en el flujo de trabajo de modelamiento del negocio y que se obtiene cuando el negocio no se encuentra claramente definido. Es una representación de los diferentes términos que se manejan en la realidad y que se encuentran estrechamente relacionados. Tiene como objetivo entender los conceptos que utilizan los usuarios o clientes y que debe manejar el sistema.

2.2.1 Diagrama de Clases del Dominio

A continuación se muestra el diagrama de Clases del Dominio del módulo de Administración.

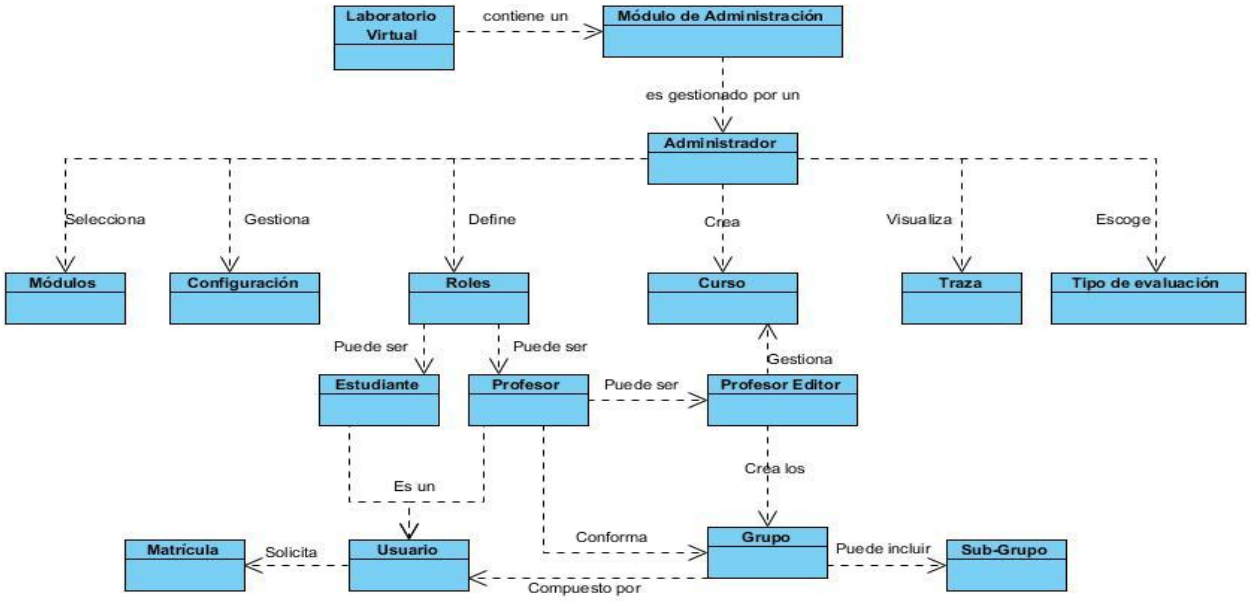

Figura 1:Diagrama de clases del dominio

Capítulo 2: Características del Sistema

Ilustración 1: Diagrama de clases del dominio

El presente diagrama de Clases del Dominio representa cómo el LAVSO contiene un módulo de Administración que es gestionado por el Administrador. Este es la persona que se encarga en el sistema de gestionar los principales aspectos que son necesarios para su correcto funcionamiento, tal es el caso de: la gestión de los módulos, cursos, roles, usuarios, traza, configuración general y tipo de evaluación. Los roles pueden ser profesor o estudiante, que son además los tipos de usuarios que maneja el sistema. Los profesores pueden ser profesores editores, que son los encargados de crear los grupos. Los grupos están compuestos por usuarios y pueden incluir subgrupos. Para ver detalladamente el diagrama de Clases del Dominio y su descripción ver el Modelo de Dominio del módulo de Administración del LAVSO. (38)

2.2.2 Definición de clases del dominio

Los principales conceptos que se manejan en el Modelo de Dominio son:

- **Laboratorio Virtual :** es el espacio virtual de experimentación para realizar las prácticas de laboratorio de los temas relacionados con la asignatura de Sistemas Operativos.
- **Módulo Administración:** en él se realizará un conjunto de funcionalidades necesarias para configurar y administrar correctamente del laboratorio virtual y que son necesarias para su adecuado funcionamiento.
- **Administrador:** es el usuario con máxima autoridad en el sistema para la gestión de la configuración general. Es el encargado de definir roles, crear el curso, y ver las trazas. Además gestiona la configuración general del sistema.
- **Módulos:** son las partes que componen un sistema y que le agrupan funcionalidades específicas.
- **Configuración:** es donde se gestiona la configuración general del sistema como son la longitud mínima de las contraseñas, el nombre del sitio, la abreviatura, el logo , etc.
- **Roles:** es donde se le dan los permisos al usuario de crear, modificar, eliminar, consultar y buscar. Estos usuarios pueden ser profesores, profesores editores y estudiantes.
- **Curso:** es donde estará publicado toda la documentación relacionada con los diferentes temas de Sistemas Operativos.
- **Traza:** es el registro de todas las acciones que se realizan en el sistema.
- **Tipo de Evaluación:** permitirá escoger la forma en la que el sistema mostrará las evaluaciones de los estudiantes, puede ser cualitativa o cuantitativa.
- **Estudiante:** se nutrirá de los contenidos y podrá realizar los ejercicios publicados en el sistema. Podrá tener acceso a los simuladores de los distintos temas.

Capítulo 2: Características del Sistema

- **Profesor:** es el encargado de conformar los grupos y sus subgrupos.
- **Profesor Editor:** es el tipo de profesor que tendrá permisos para gestionar todo lo referente al curso y manejo de sus contenidos y temas.
- **Matrícula:** es donde el personal docente se puede matricular en el curso.
- **Usuario**: son todas las personas que interactúan con el sistema. Conforman los grupos y subgrupos.
- **Grupo:** es donde se agrupan un conjunto de estudiantes que tienen características comunes. Además puede incluir subgrupos.
- **Subgrupos:** son los pequeños grupos por los que está conformado un grupo.

2.3 Requisitos de Software

Partiendo de la interacción con el cliente y la aplicación de las técnicas anteriormente estudiadas se obtuvieron los requisitos del sistema, agrupados en requisitos funcionales y no funcionales. Dentro de las técnicas empleadas para el levantamiento de requisitos se encuentra la tormenta de ideas para entender en un primer momento que es lo que se quiere hacer y posteriormente la técnica de la entrevista para profundizar en las características que debe tener el sistema. También se empleó la técnica de Casos de Uso para agrupar las funcionalidades y ver su relación con los usuarios. Para ver la descripción textual de cada uno de los requisitos del sistema, ver el documento de Especificación de Requisitos del módulo de Administración del LAVSO. (39)

Requisitos Funcionales

Se obtuvo un total de 50 requisitos funcionales, de ellos 17 de complejidad alta, 15 de complejidad media y 18 de complejidad baja.Las condiciones o capacidades que debe tener el sistema son:

RF_001 Crear grupo

RF_002 Modificar grupo

RF_003 Eliminar grupo

RF_004 Buscar grupo

RF_005 Mostrar detalles del grupo

RF_006 Listar grupos

RF_007 Añadir miembro a un grupo

RF_008 Eliminar miembro de un grupo

RF_009 Listar miembros de un grupo

RF_010 Crear subgrupo

RF_011 Modificar subgrupo

RF_012 Eliminar subgrupo

RF_013 Listar subgrupos

RF_014 Mostrar detalles del subgrupo

RF_015 Añadir miembro a un subgrupo

RF_016 Eliminar miembro de un subgrupo

RF_017 Listar miembros de un subgrupo

RF_018 Adicionar usuario

Capítulo 2: Características del Sistema

RF_019 Modificar usuario

RF_020 Eliminar usuario

RF_021 Buscar usuario

RF_022 Listar usuarios online

RF_023 Listar usuarios

RF_024 Mostrar detalles del usuario

RF_025 Crear rol

RF_026 Modificar rol

RF_027 Eliminar rol

RF_028 Listar roles

RF_029 Crear curso

RF_030 Modificar curso

RF_031 Eliminar curso

RF_032 Mostrar datos del curso

RF_033 Matricular personal docente

RF_034 Desmatricular personal docente

RF_035 Listar trazas

RF_036 Buscar traza

RF_037 Habilitar módulo

RF_038 Ajustar configuración general

RF_039 Escoger tipo de evaluación

RF_040 Adicionar rango de evaluación

RF_041 Modificar rango de evaluación

RF_042 Eliminar rango de evaluación

RF_043 Listar rangos de evaluación

RF_044 Ver sistema como profesor

RF_045 Ver sistema como estudiante

RF_046 Autenticar usuario

RF_047 Cerrar sesión

RF_048 Crear tema

RF_049 Modificar tema

RF_050 Eliminar tema

Requisitos No Funcionales

Las cualidades o propiedades que debe cumplir el sistema son:

Seguridad:

RNF_001 Autenticación obligatoria y segura.

RNF_002 Acceso a la información según el rol.

RNF_003 Manejo de sesiones del usuario, expira la sesión en 10 minutos.

RNF_004 Realizar salvas cada 5 días de la información contenida en la base de datos.

Usabilidad:

RNF_005 Permitir uso del teclado para realizar operaciones sobre el sistema.

RNF_006 Debe poseer una interfaz agradable para el cliente.

RNF_007 Mostrar la información de forma lógica y correctamente estructurada.

Capítulo 2: Características del Sistema

Disponibilidad:

RNF_008 El sistema debe estar accesible en todo momento.

RNF_009 El sistema debe ofrecer tiempos de respuesta relativamente bajos, sin sobrepasar los 3 minutos.

RNF_010 Debe garantizarse como mínimo 60 usuarios conectados concurrentemente sin que disminuya el rendimiento y rapidez de la aplicación.

Interfaz:

RNF_011 El sistema debe garantizar la configuración y cambio de sus parámetros de forma fácil y rápida.

Restricciones de diseño e Implementación:

RNF_012 Debe garantizarse el uso de patrones de diseño y estándares de codificación.

Requisitos para usuarios

Hardware:

RNF_013 CPU Intel Pentium 4, 1.7 HGz o AMD con características similares.

RNF_014 Memoria RAM mínimo 256 MB

RNF_015 Capacidad de disco duro 20 GB.

Software:

RNF_016 Debe permitirse su uso en cualquier plataforma.

Requisitos para servidores:

Hardware:

RNF_017 CPU Intel Pentium 4 1.7 HHz o AMD con características similares.

RNF_018 Memoria RAM mínimo 512 MB.

RNF_019 Capacidad de disco duro 40 GB.

Software:

RNF_020 Sistema Operativo: GNU/Linux Debian o Ubuntu.

RNF_021 Servidor Web Apache 2.2.3.

2.4 Diagrama de CU del Sistema

El diagrama de casos de uso (CU) referente al módulo de Administración quedó compuesto por un total de diecisiete CU, clasificados en seis críticos, ocho secundarios, dos auxiliares y uno opcional, que agrupan los requisitos funcionales identificados en etapas anteriores.

Dentro de los principales patrones de CU utilizados en el diagrama de CU del sistema se encuentran: **Reusabilidad** para separar funcionalidades que deben ser ejecutadas por varios CU; el **CRUD Completo** para los CU que incluyen todas las funcionalidades básicas de un Gestionar (Adicionar,

Modificar, Eliminar y Mostrar); y el patrón **Roles Comunes** para unir roles que hacen las mismas funcionalidades sobre un CU.

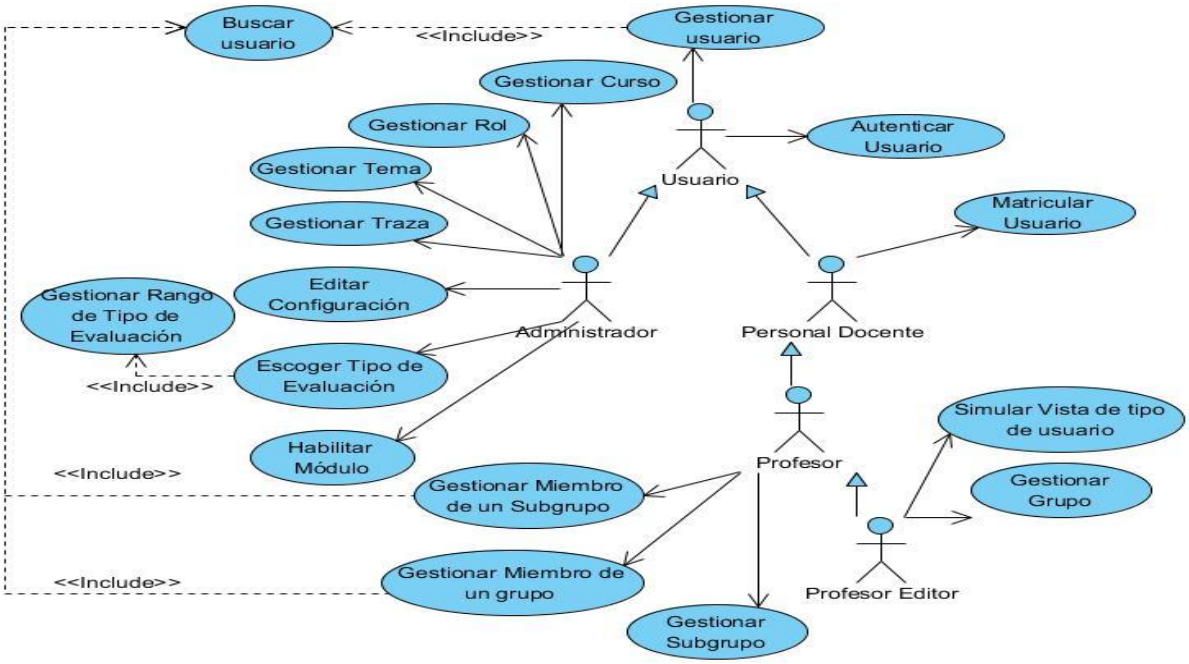

Figura 2: Diagrama de CU del sistema

2.4.1 Actores

Un actor es alguien o algo, ya sea un usuario o una aplicación existente, que interactúa con el sistema; es quien lo utiliza. A continuación se mencionan y se describen los actores relacionados con el módulo de Administración:

Usuario: son todas las personas que interactúan con el sistema, que pueden ser el administrador, un profesor o un estudiante.

Administrador: es la persona con máxima autoridad en el sistema para la gestión de este. Se encarga de la gestión de usuario, rol, permiso, configuración general y curso.

Personal Docente: agrupa los usuarios que tienen acceso al sistema con fines educativos, puede ser profesor o estudiante.

Profesor: Es el trabajador docente que enseña una materia. Pueden estar asignados a un grupo específico y puede ser profesor común o un profesor editor.

Profesor Editor: Es el tipo de profesor que es el responsable de la gestión de la asignatura en el sistema. Gestiona los temas, contenidos del curso y los grupos.

2.4.2 Casos de Uso (CU)

Un caso de uso es una o varias funcionalidades agrupadas por características comunes. Describe paso a paso las acciones que deberán realizarse tanto por el usuario como por el sistema para llevar a cabo algún proceso. En esta sección se hará referencia solo a los CU críticos del sistema:

Caso de Uso:	**Gestionar Grupo**
Actores:	Profesor Editor
Resumen:	Este caso de uso se inicia cuando el Profesor Editor elige gestionar un grupo. Debe haberse autentificado en el sistema. Consiste en crear, modificar, eliminar, buscar, mostrar detalles y listar los grupos. Finaliza cuando se ha adicionado, modificado, eliminado, buscado, mostrado los detalles o listado los grupos.
Referencias:	RF_001, RF_002, RF_003, RF_004, RF_005, RF_006
Caso de Uso:	**Gestionar Miembros de un Grupo**
Actores:	Profesor
Resumen:	Este caso de uso se inicia cuando el Profesor elige gestionar los miembros de un grupo. Debe haberse creado previamente el grupo. El caso de uso consiste en añadir un miembro al grupo, eliminar un miembro del grupo y listar los miembros del grupo. Finaliza cuando se ha añadido, eliminado o listado los miembros que integran un grupo.
Referencias:	RF_007, RF_008, RF_009
Caso de Uso:	**Gestionar Curso**
Actores:	Administrador
Resumen:	Este caso de uso se inicia cuando el Administrador elige Gestionar Curso. Debe haberse autentificado previamente. Este caso de uso consiste en crear, modificar y eliminar un curso. Finaliza cuando se ha adicionado, modificado o eliminado un curso.
Referencias:	RF_029, RF_030, RF_031, RF_032
Caso de Uso:	**Gestionar Rol**
Actores:	Administrador
Resumen:	Inicia este caso de uso cuando el Administrador elige Gestionar Rol. Este caso de uso consiste en crear, modificar, eliminar y listar los roles existentes. Finaliza cuando se ha adicionado, modificado, eliminado o listado los roles.
Referencias:	RF_025, RF_026, RF_027, RF_028
Caso de Uso:	**Gestionar Usuario**
Actores:	Administrador
Resumen:	Se inicia el caso de uso cuando el Administrador elige Gestionar Usuario. Se debe haber autentificado previamente. Este caso de uso consiste en adicionar, modificar, eliminar, listar usuarios, listar usuarios online y mostrar

	detalles del usuario. Finaliza cuando se ha adicionado, modificado, eliminado o mostrado los detalles del usuario. También se deben haber listado los usuarios del sistema y los usuarios conectados.
Referencias:	RF_018, RF_019, RF_020, RF_022, RF_023, RF_024
Caso de Uso:	**Gestionar Tema**
Actores:	Administrador
Resumen:	Se inicia este caso de uso cuando el Administrador elige Gestionar Tema. Debe haberse creado previamente el curso para poder asignar los temas que lo componen. Este caso de uso consiste en adicionar, modificar y eliminar un tema. Finaliza cuando se ha adicionado, modificado o eliminado un tema.
Referencias:	RF_018, RF_019, RF_020, RF_022, RF_023, RF_024

2.4.3 Descripción de CU

Luego de definir los casos de uso del sistema, se procede a la descripción textual de los mismos con la intención de especificar cada una de las funcionalidades que deben ser implementadas. La descripción de los casos de uso constituye una guía para los desarrolladores y un documento de obligatorio cumplimiento en cuanto a desarrollo de funcionalidades en el sistema. Se realiza para describir las acciones que realizan los usuarios y la respuesta que debe dar el sistema ante cada una de ellas. Seguidamente se muestra la descripción de uno de los principales casos de uso del módulo de Administración del LAVSO. Para acceder a la descripción íntegra de todos los CU, ver el documento Modelo de Sistema del módulo de Administración del LAVSO. (40)

Caso de Uso: Gestionar Curso

Caso de Uso:	Gestionar Curso
Actores:	Administrador
Resumen:	Este caso de uso se inicia cuando el Administrador elige Gestionar Curso. Debe haberse autentificado previamente. Este caso de uso consiste en crear, modificar y eliminar un curso. Finaliza cuando se ha adicionado, modificado o eliminado un curso.
Precondiciones:	Que el Administrador esté registrado en el sistema.
Referencias	RF_029, RF_030 , RF_031, RF_032
Prioridad	Crítico
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El Administrador decide gestionar el curso.	2. El sistema muestra la interfaz de gestionar curso con los siguientes datos: • Nombre • Descripción • Profesor editor
3. Si el Administrador selecciona la opción Crear Curso ver la sección **"Crear Curso".** Si selecciona la opción modificar Grupo ver la sección **"Modificar Grupo".** Si selecciona la opción Eliminar Curso ver la sección **"Eliminar Curso".**	4. El sistema cierra la interfaz. Se termina así el caso de uso.

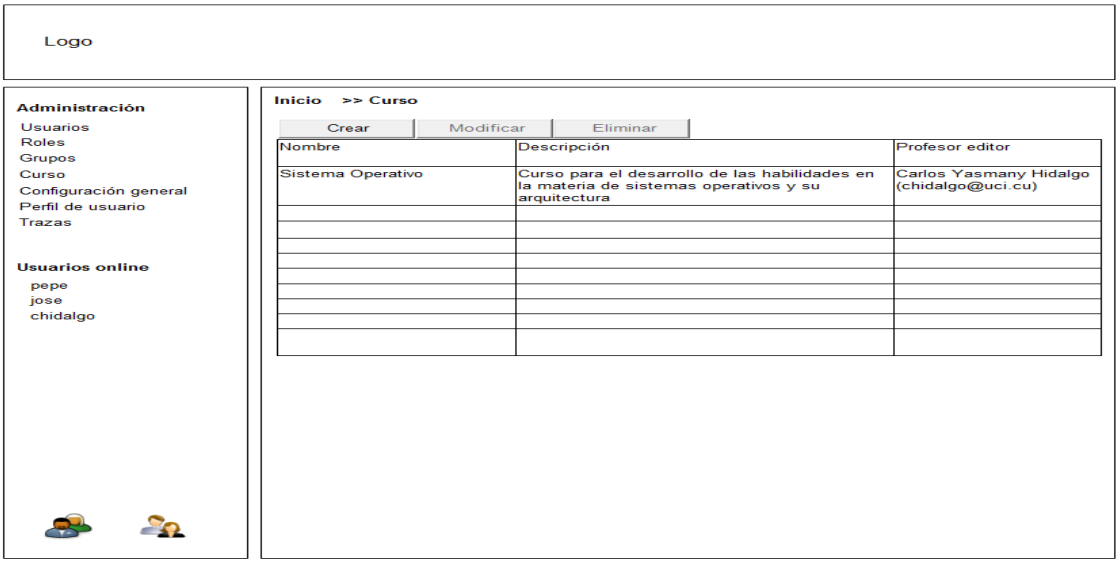

Figura 3: Prototipo no funcional CU Gestionar Curso. Escenario Listar Curso

Sección "Crear Curso"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador elige la opción Crear Curso.	2. El sistema muestra la interfaz crear curso con los siguientes campos: • Nombre • Descripción • Fecha de inicio • Fecha de cierre • Activo • Profesores editores

3. El Administrador introduce los datos correspondientes. 4. El Administrador elige la opción aceptar. Si decide cancelar ver flujo alterno 1 **"Cancelar".**	5. El sistema guarda los datos. Termina así el caso de uso.

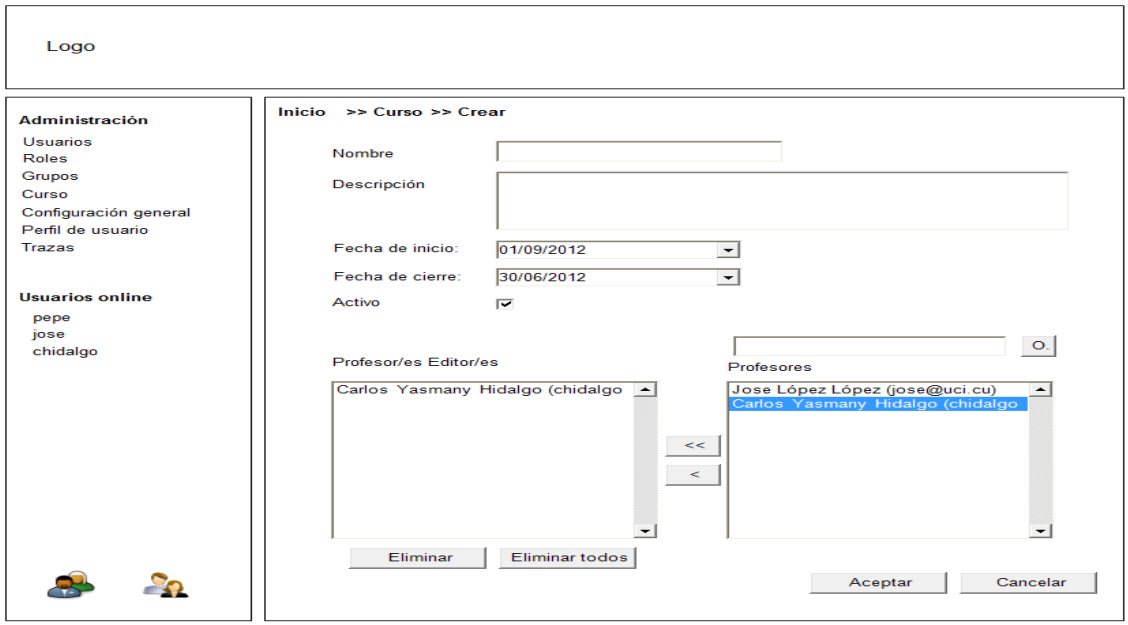

Figura 4: Prototipo no funcional CU Gestionar Curso. Escenario Crear Curso

Sección "Modificar Curso"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador elige el curso que desea modificar. 2. El Administrador elige la opción Modificar Curso.	3. El sistema muestra la interfaz modificar curso con los siguientes datos cargados: • Nombre • Descripción • Fecha de inicio • Fecha de cierre • Activo • Profesores editores
4. El Administrador modifica los datos deseados. 5. El Administrador elige la opción aceptar. Si decide cancelar ver flujo alterno 1 **"Cancelar".**	6. El sistema guarda los datos. Termina así el caso de uso.

Capítulo 2: Características del Sistema

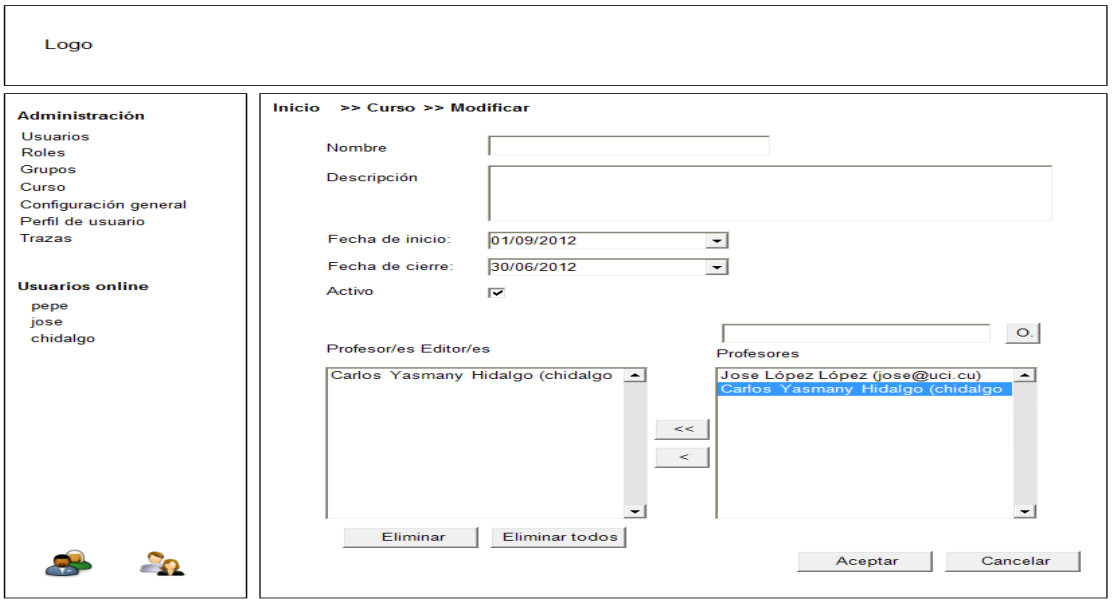

Figura 5: Prototipo no funcional CU Gestionar Curso. Escenario Modificar Curso

Sección "Eliminar Curso"

Acción del Actor	**Respuesta del Sistema**
1. El Administrador elige el curso que desea eliminar. 2. El Administrador elige la opción Eliminar Curso.	3. El sistema muestra un mensaje de confirmación.
4. El Administrador elige la opción aceptar. Si decide cancelar ver flujo alterno 1 **"Cancelar".**	5. El sistema elimina el curso. Termina así el caso de uso.

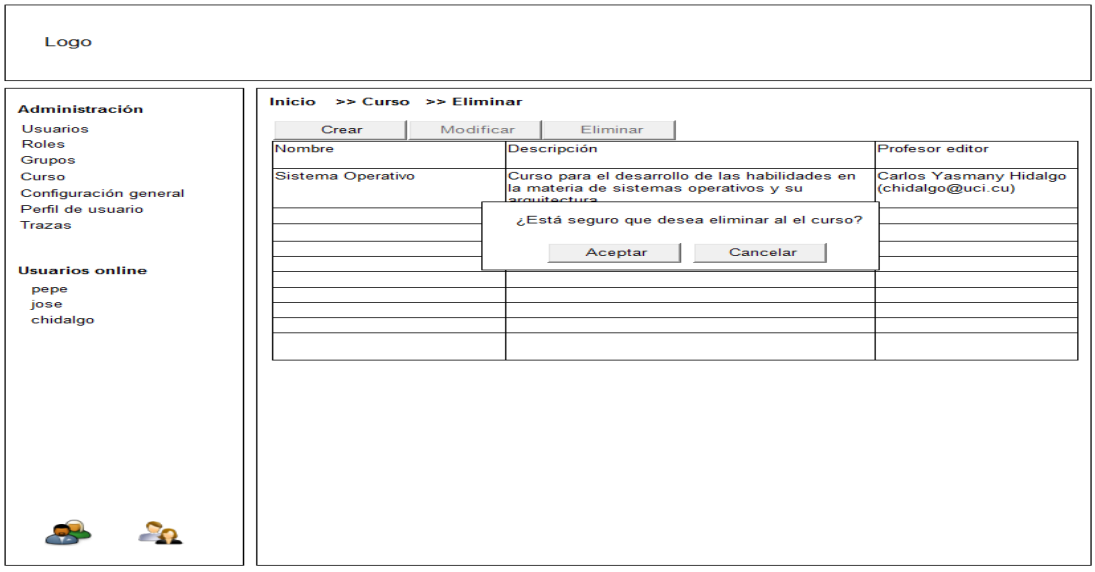

Figura 6: Prototipo no funcional CU Gestionar Curso. Escenario Eliminar Curso

Flujos Alternos 1 "Cancelar"

Acción del Actor	Respuesta del Sistema
1. El Administrador elige cancelar	2. El sistema cancela la operación. Termina así el caso de uso.
Poscondiciones:	Queda adicionado, modificado o eliminado un curso.

Tabla 1: Descripción de CU Gestionar Grupo

2.5 Conclusiones parciales

En este capítulo se analizaron las características que debe presentar el módulo de Administración del LAVSO arribándose a las siguientes conclusiones:

- Se obtuvo el modelo de dominio por no existir un negocio claramente definido, propiciando un mejor entendimiento sobre los principales conceptos que maneja el módulo.
- El análisis del Modelo de Dominio, junto a la interacción con los clientes del sistema y la aplicación de técnicas para la Elicitación de requisitos, propició que se obtuvieran resultados satisfactorios en la identificación de estos, tanto funcionales como no funcionales que debe cumplir el sistema.
- La identificación y especificación de los artefactos del modelo del sistema facilitó un mayor entendimiento y un acuerdo común entre los clientes y los desarrolladores, en cuanto a la concepción de las funcionalidades que el sistema debe cumplir.
- La aplicación de patrones de casos de uso propiciaron obtener artefactos aceptables y de fácil entendimiento para las posteriores etapas de desarrollo de software.

Capítulo 3: Diseño e Implementación

3.1 Introducción

En el presente capítulo se tratan los elementos relacionados con el diseño y la implementación del módulo de Administración del LAVSO. Se muestra cómo está concebida la arquitectura y las principales ventajas que brinda el marco de trabajo Vaadin. Se exponen los principales diagramas generados en la fase de diseño como son Diagramas de Paquetes, Diagrama de Clases del Diseño y Diagramas de Secuencia. También se describe los estándares de codificación a seguir y los patrones de diseño empleados durante el desarrollo, además se exponen los elementos que describen la implementación del sistema como son: el diagrama de componentes, el modelo de datos y el diagrama de despliegue.

3.2 Arquitectura del LAVSO

Para el LAVSO se definió la arquitectura en tres capas que como se menciona en el capítulo anterior proporciona la escalabilidad necesaria para dar organización a las diferentes partes que componen el sistema, además define una organización jerárquica de manera que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediata inferior, además se emplea el patrón arquitectónico Modelo–Vista–Controlador (MVC) el cual separa las clases que componen el sistema de acuerdo a sus funcionalidades, agrupando las clases del acceso a datos , las clases controladoras y las vistas del sistema, definiendo claramente los lenguajes de programación utilizados.

Vaadin es un framework web creado para desarrollar aplicaciones RIA (Rich Internet Applications), brinda la facilidad de crear aplicaciones de escritorios embebidas en la web. Presenta un modelo orientado a componentes, donde ofrece la facilidad de utilizar el editor visual, con funcionalidades para crear Componentes y Wizards, con ayuda integrada. Fue hecho pensando en simplificar la programación, donde no es necesario conocer ni HTML, ni XML, ni Javascript. Este framework utiliza GWT como motor de renderizado, de modo que aunque genera aplicaciones de escritorio permite combinarlas en la web gracias a la facilidad de generar el código del lado del cliente.

La estructura que define el marco de trabajo Vaadin para la organización por capas deja bien delimitado donde quedan agrupadas las clases del modelo, las controladoras y las vista, existiendo una relación directa entre cada una de estas capas y los paquetes que organizan las clases existentes, quedando estructurada como sigue:

Capítulo 3: Diseño e Implementación

Figura 7: Capa de acceso a datos y capa controladora

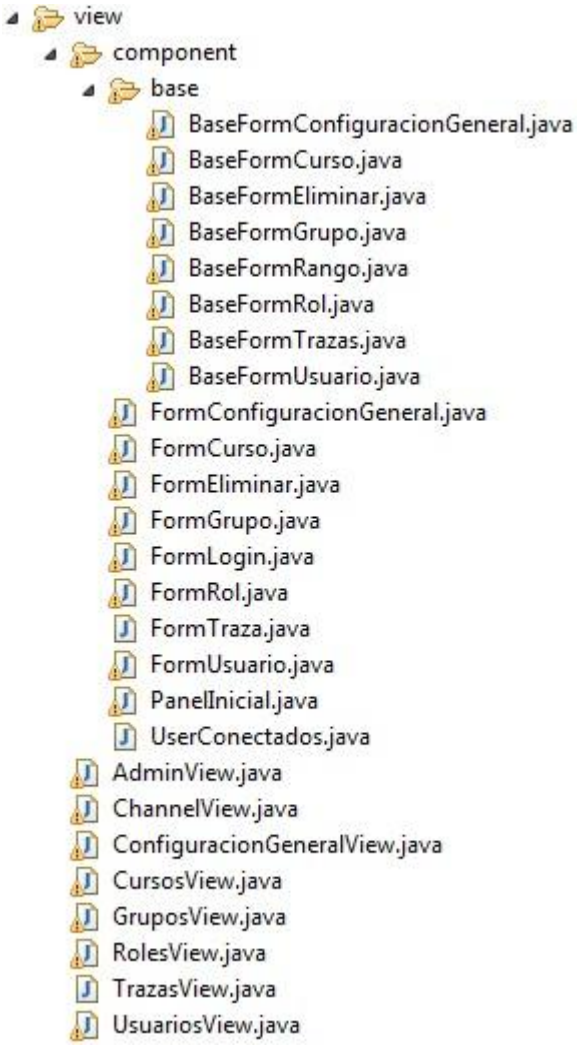

Figura 8: Capa de las vistas

Capítulo 3: Diseño e Implementación

3.3 Diagrama de Paquetes

La organización por paquetes se realiza puesto que en un punto del ciclo de desarrollo, el software va adquiriendo un elevado número de clases y componentes, y con ello un tamaño excesivo. Es entonces cuando surge la necesidad de dividir sus elementos en subconjuntos más pequeños para organizarlos de manera más detallada.

El LAVSO, en su concepción como sistema, quedó estructurado con un total de cinco paquetes, de ellos un paquete *Main* para agrupar las clases principales del sistema; un paquete *View* para agrupar las vistas generales de la aplicación, un paquete *Component* para agrupar los componentes, como formularios y secciones de vistas; un paquete *Util* para englobar las clases auxiliares que pueden ser de utilidad para el sistema y un paquete *Modules*, que agrupa en sí los módulos o subsistemas que integran el sistema LAVSO (Autoaprendizaje, Ejercicios, Evaluación y Seguimiento, Simuladores y Administración). La representación de los paquetes y su relación es evidenciada en el siguiente diagrama:

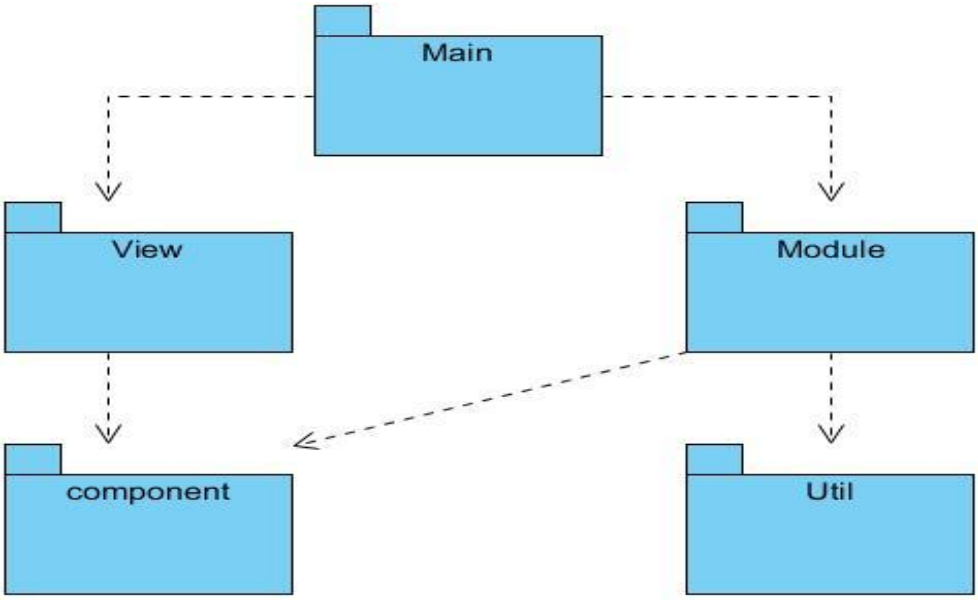

Figura 9: Diagrama de paquetes del LAVSO

A su vez el módulo de Administración quedó estructurado de la siguiente manera: un paquete *Controller* para agrupar las clases controladoras del módulo; un paquete *View* para organizar las interfaces de usuario, un paquete *Component* para agrupar los componentes; como son los formularios y las secciones de vistas, y finalmente un paquete *Model* para agrupar las clases del negocio y todo el acceso a datos. El paquete *Component* forma parte del paquete *View*.

Capítulo 3: Diseño e Implementación

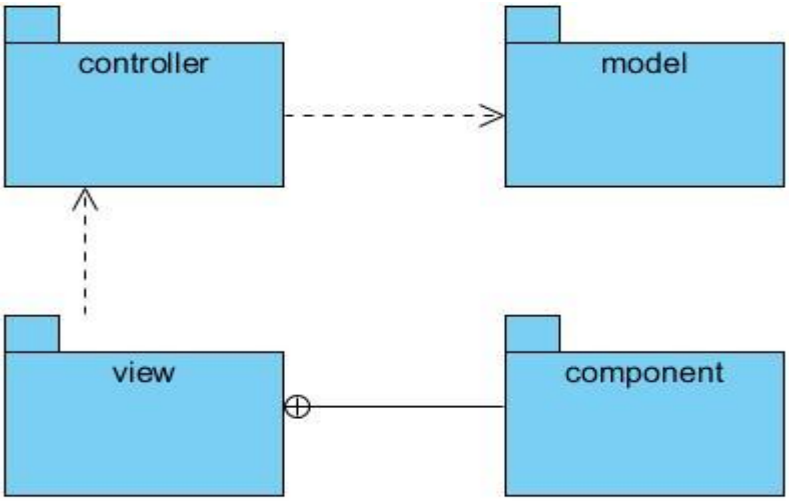

Figura 10: Diagrama de paquetes del módulo de Administración

3.4 Diagrama de Clases del Diseño

Los diagramas de Clases del Diseño son considerados diagramas de estructura estática que representan los elementos que componen la solución del sistema. Especifican las clases de software de una aplicación, y las relaciones que existen entre ellas a través de asociaciones entre clases. Las Clases del Diseño están compuestas por: nombre, atributos y operaciones (métodos). Se pueden emplear estereotipos web para la representación de algunas clases como son: los formularios, los archivos Java Script, los archivos CSS, las páginas clientes y las páginas servidoras, pero en el caso de LAVSO no se utilizará porque está diseñada para ser una aplicación de escritorio que funciona sobre la web. El marco de trabajo Vaadin proporciona como una de sus ventajas la generación de aplicaciones de este tipo.

A continuación se muestra la representación del diagrama de Clases del Diseño del CU Gestionar Curso, donde la aplicación se inicia cuando la clase Laboratorio_virtualAplication ejecuta la instancia de la clase UIHandler, que es la encargada de decidir qué vista es la que se debe mostrar de acuerdo a los permisos del usuario. Estas dos clases forman parte del paquete general del sistema “Main”. Una vez que se decide qué clase de la vista del módulo se va a ejecutar, esta se encarga de mostrar la interfaz con el formulario que la compone. Tanto la clase CursoView como la FormCurso tienen relación directa con la clase controladora, ya que cuando se llama a los métodos getCursoView, getCursoAddView y getCursoModView la relación con la controladora es a través de la clase FormCurso, y cuando se llama a los métodos getCursoDelView y getCursoDetView es mediante la clase CursoView. La clase controladora CursoController tiene una instancia de la clase CursoDAO que hereda de la clase CursoJPAController que se encuentra dentro de un paquete ya existente de

framework JPA. La clase CursoDAO se relaciona directamente con la clase persistente dCurso. Para ver los restantes diagramas de clases ver el Modelo de Diseño del módulo de Administración del LAVSO. (41)

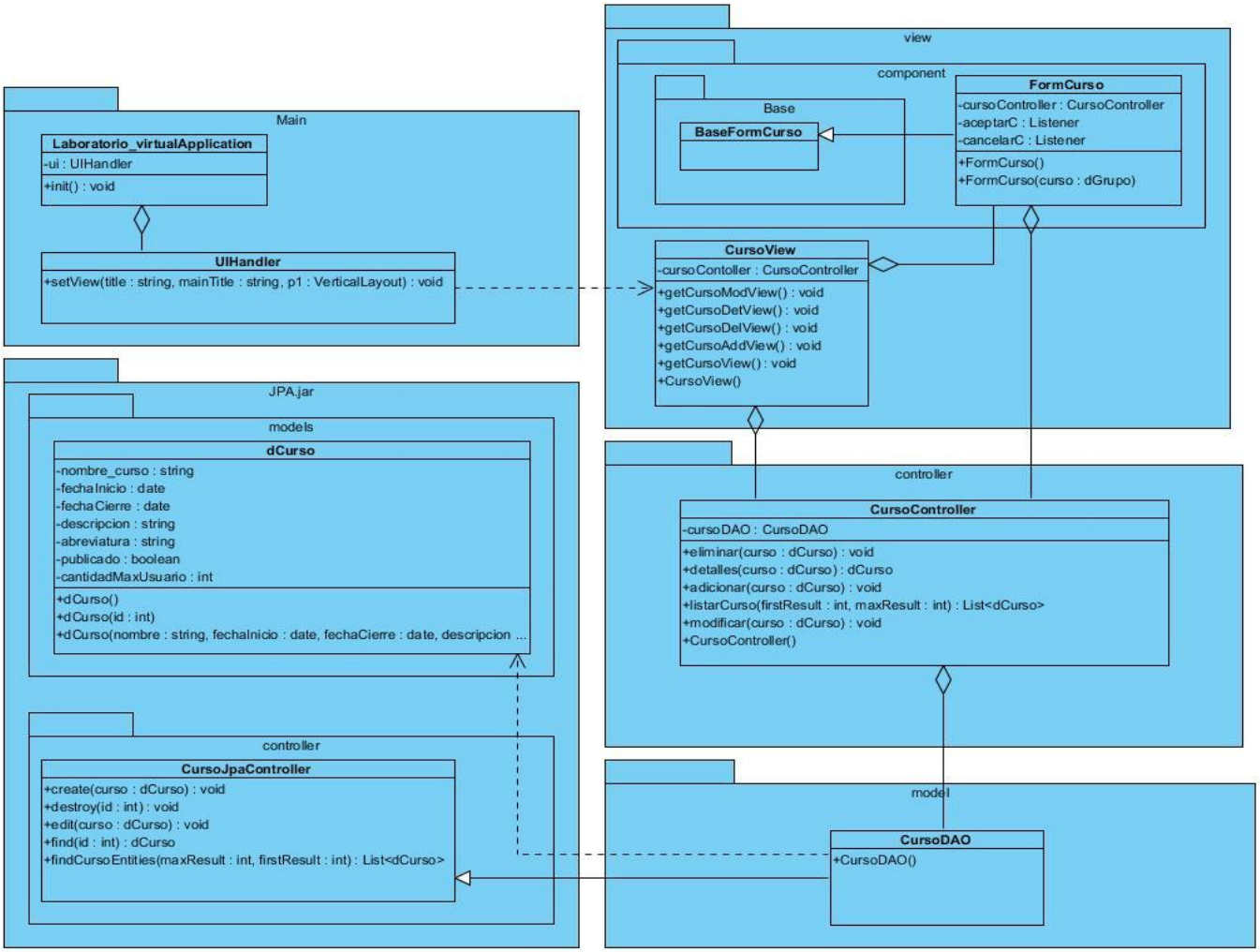

Figura 11: Diagrama de clases. CU Gestionar Curso

3.5 Diagrama de Secuencia

Los diagramas de Interacción explican gráficamente la interacción existente entre las instancias de las clases y la secuencia de pasos que deben seguir. Para una mejor representación y entendimiento de los flujos de los CU definidos, se elaboraron los diagramas de Secuencia como parte de la realización de cada uno de estos. Los diagramas de Secuencia son diagramas de Interacción que muestran las interacciones entre los objetos que estarán presentes en la aplicación, ordenados por una secuencia temporal de pasos que deben seguirse para la realización de cada uno de los escenario que

componen el CU. A los CU que tienen varias secciones se les realizó un diagrama de secuencia por cada flujo para un mejor entendimiento de los mismos.

A continuación se muestran los diagramas de secuencia elaborados para los escenarios del CU Gestionar Curso, los restantes diagramas de Secuencia del diseño se pueden consultar en el documento referente al Modelo de Diseño del módulo de Administración del LAVSO. (41)

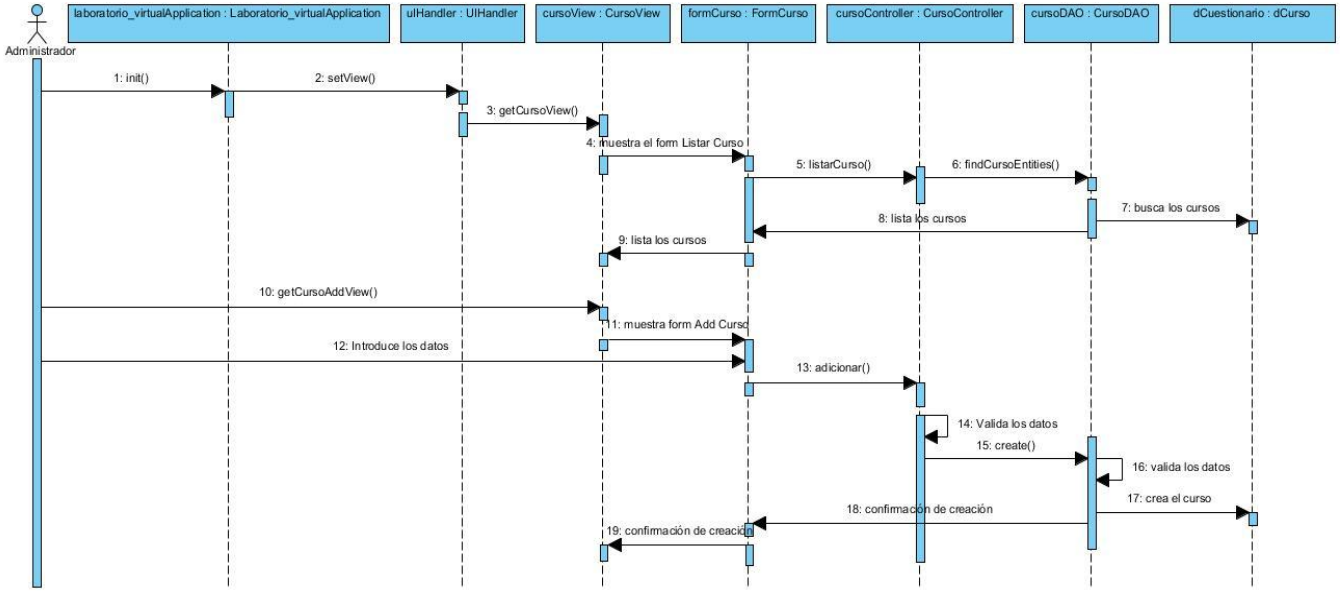

Figura 12: Diagrama de secuencia. Sección Crear Curso

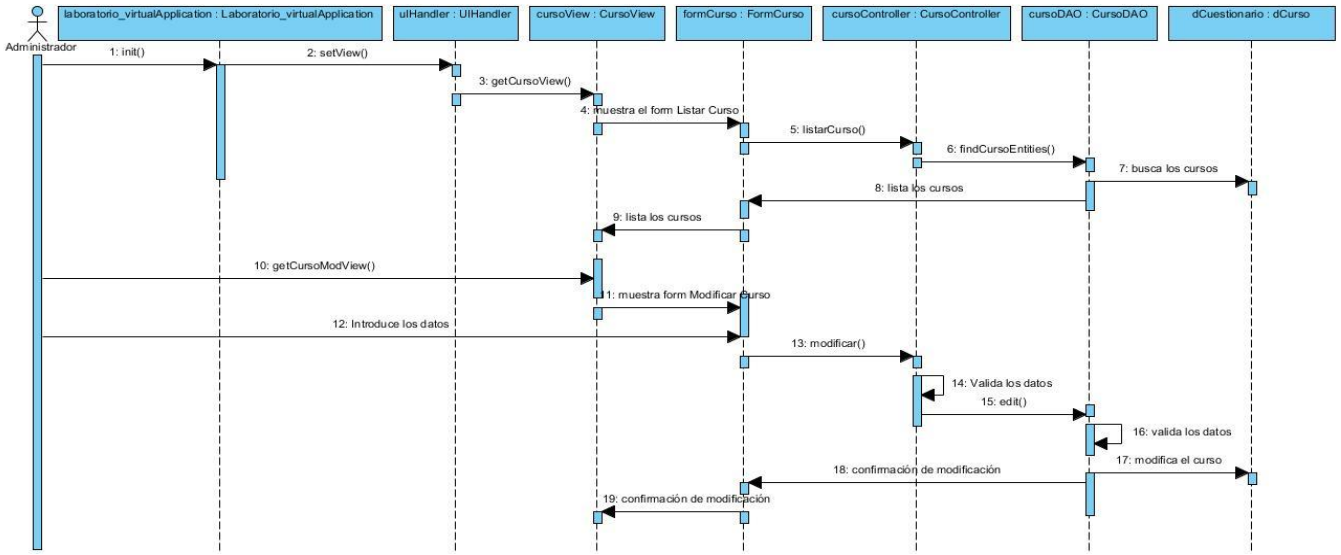

Figura 13: Diagrama de secuencia. Sección Modificar Curso

Capítulo 3: Diseño e Implementación

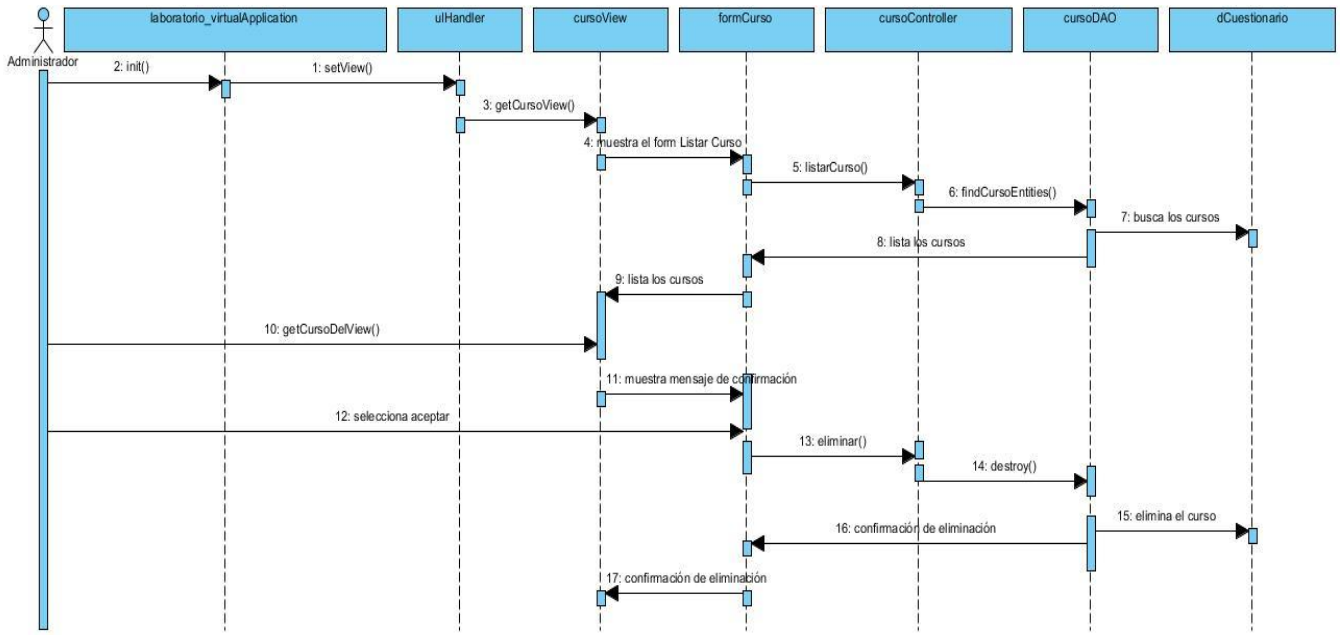

Figura 14: Diagrama de secuencia. Sección Eliminar Curso

3.6 Modelo de Datos

El modelo de datos es la representación lógica que se hace de todos los datos que deben guardarse en el sistema. Básicamente está formado por tres elementos fundamentales: los objetos, que son todas las entidades que manipulan los datos a persistir; los atributos, que son las características básicas de los objetos antes mencionados; y las relaciones, que son las que enlazan a dichos objetos entre sí.

El modelo de datos correspondiente al módulo de Administración del LAVSO consta de un total de dieciséis tablas. De ellas un nomenclador y las quince restantes para el almacenamiento de la información correspondiente al flujo de trabajo del módulo. La nomenclatura de dichas entidades es de la siguiente forma: para las tablas de datos d<Nombre>, para la tabla nomencladora n<Nombre> y para las que almacenan relaciones entre entidades d<Nombre1><Nombre2>. A continuación se muestra el modelo de datos del módulo de Administración del LAVSO.

Capítulo 3: Diseño e Implementación

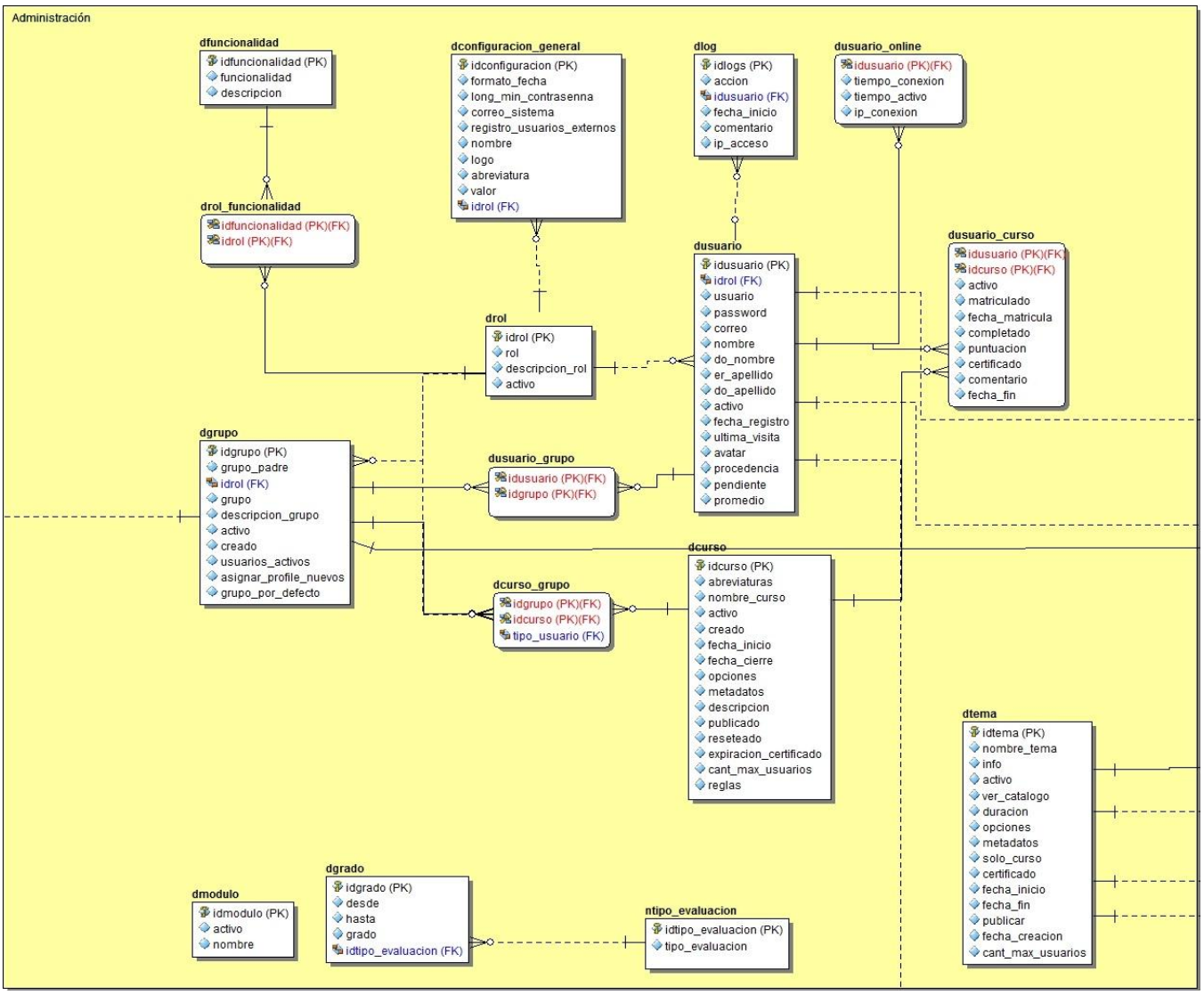

Figura 15: Modelo de datos del módulo de Administración

3.7 Patrones empleados

Teniendo en cuenta las características del sistema en términos de programación y como posible solución a un problema de este tipo se decide emplear un conjunto de patrones de diseño que se aplican como sigue:

Bajo Acoplamiento: el bajo acoplamiento consiste en que exista la menor dependencia posible entre clases, evitando que al hacer una modificación sobre una de ellas se deban modificar las otras. Vaadin como framework de desarrollo presenta interfaz gráfica y genera el código java correspondiente a la misma. La clase BaseFormCurso no es más que la clase que engloba dicho código, por lo que al hacer una modificación en la interfaz del Gestionar Curso, Vaadin sobrescribe el código existente hasta ese

momento. Es aquí cuando surge la necesidad de heredar de esta clase para poder insertar cualquier código diferente del que genera el marco de trabajo, permitiendo que al modificar la clase BaseFormCurso no se vea afectada la clase FormCurso.

Controlador: el patrón controlador se evidencia en la clase CursoController que es la clase encargada de manejar el intercambio de datos entre la clase de la vista y el acceso a datos.

Experto: el patrón experto se ve reflejado en la clase UIHandler, que es la clase que decide qué interfaz es la que se debe mostrar de acuerdo a los privilegios que tenga el usuario.

Creador: el patrón creador es evidenciado en diferentes clases, pues se emplea siempre que se necesite asignar a una clase la responsabilidad de crear una instancia de otra, partiendo que esta es la que maneja la información necesaria para invocar dicha clase. Se pone en práctica en la clase CursoView que tiene una instancia de la clase CursoController, pues en muchos de los escenarios del CU Gestionar Curso, esta clase es la que contiene la información necesaria para instanciar la clase controladora.

Decorador: en cada una de las vistas que conforman el módulo de Administración se utilizan componentes comunes del sistema que agrupan funcionalidades y diseños para las interfaces, permitiendo así la reutilización del código para las vistas.

3.8 Estándares de codificación

Los estándares de codificación como se hace referencia en el capítulo 1 son de gran importancia para la implementación del sistema, pues dan uniformidad al código además de que permiten entenderlo fácilmente. El estándar de codificación definido en el desarrollo del sistema LAVSO ha sido aplicado por todos los módulos y subsistemas que lo componen y es evidenciado en los siguientes ejemplos:

- Los nombres a las clases han sido establecidos de acuerdo a su función y todas las palabras que la componen se inician con mayúscula. Ejemplo: "`FormCurso.java`".

- Los nombres de las variables y de las funciones pueden iniciar con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula o pueden estar separadas por guión. Ejemplo: `Front_theme_View themeView = new Front_theme_View();`

- La llave de apertura "{" de los métodos deben aparecer al final de la misma línea de la sentencia de declaración. La llave de cierre "}" debe colocarse en un nuevo renglón.

```
public void MostrarVista() {

    Front_theme_View themeView = new Front_theme_View();
    addComponent(themeView.getMainLayout());

}
```

- Los nombres de los paquetes son escritos completamente con minúscula.
- Cada clase y cada método se le debe incluir un comentario con su descripción. A las clases se le debe incluir su autor y última fecha de modificación.
- Evitar las líneas de más de 80 caracteres, en caso de existir romperlas después de una coma o antes de un operador.
- Las variables locales se deben inicializar donde se declaren.
- Los métodos se separan con una línea en blanco.

3.9 Diagrama de Componentes

Los diagramas de componentes modelan la vista estática de un sistema. Se representa como un grafo de componentes de software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten.

En la siguiente figura se muestra el diagrama de componentes correspondiente al caso de uso Gestionar Curso. Cada paquete representa una división física del sistema que agrupa los componentes que integran el CU al que se está haciendo referencia. Para ver los restantes diagramas de paquetes ver el Modelo de Implementación del módulo de Administración del LAVSO. (42)

Capítulo 3: Diseño e Implementación

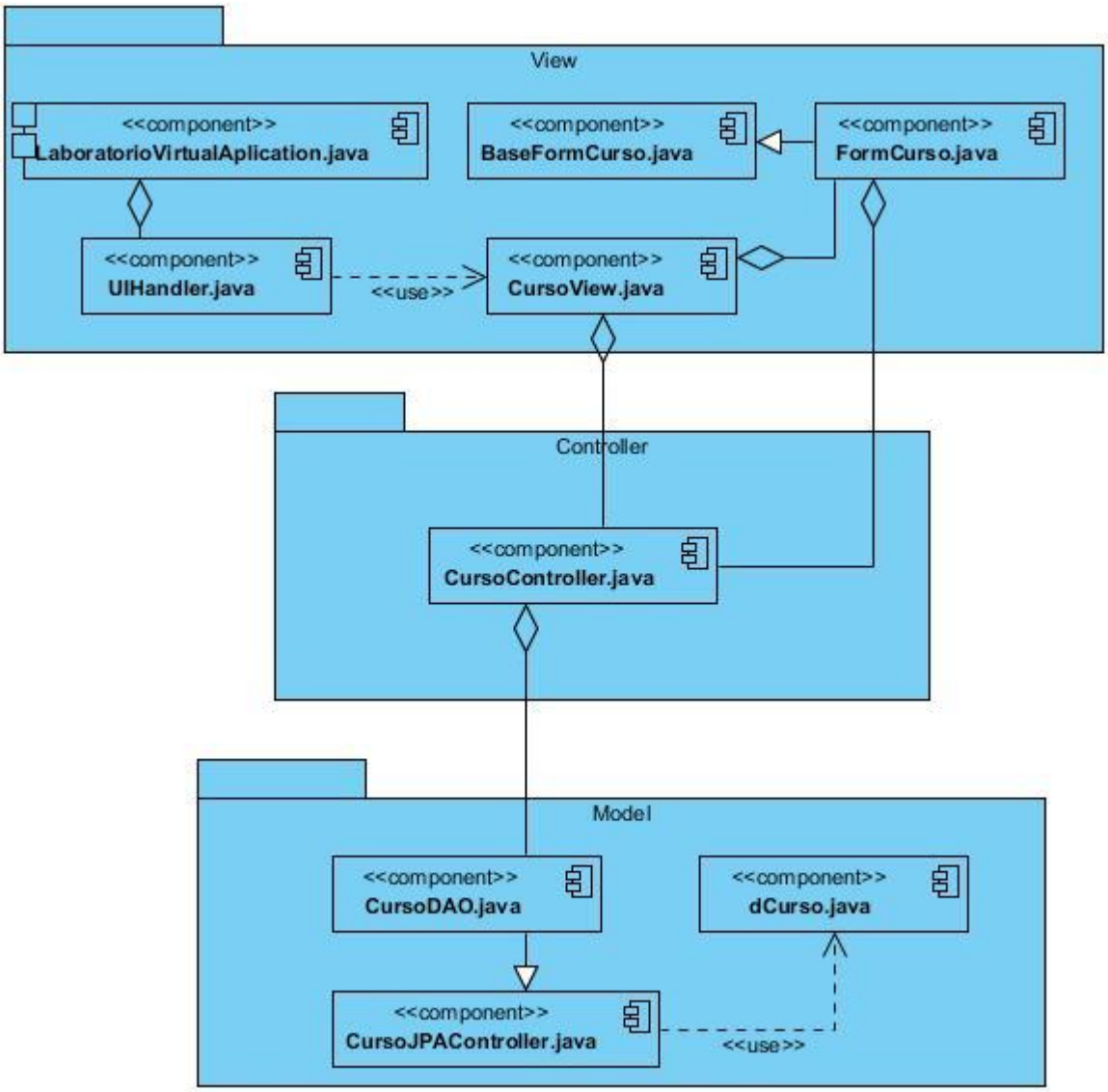

Figura 16: Modelo de componentes. CU Gestionar Curso

3.10 Diagrama de Despliegue

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento, y las conexiones entre estos elementos en el sistema. El modelo consiste en uno o más nodos (elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos), dispositivos (nodos estereotipados con una capacidad de procesamiento en el nivel modelado de abstracción), y conectores, entre nodos, y entre nodos y dispositivos. A continuación se

muestra el diagrama de despliegue correspondiente al LAVSO como sistema integrado por varios módulos, dentro de ellos el módulo de Administración.

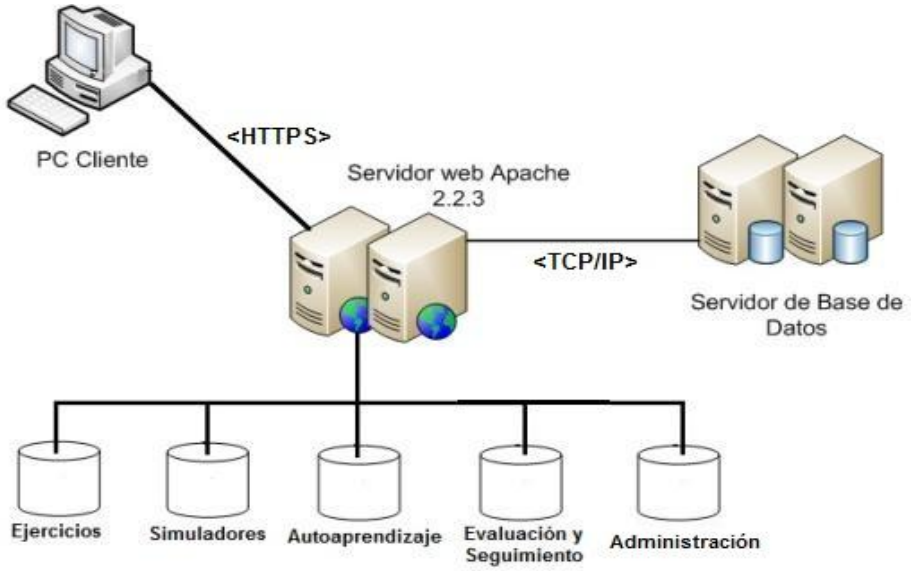

Figura 17: Modelo de despliegue. Sistema LAVSO

El presente diagrama muestra como el LAVSO compuesto por los cuatro subsistemas (Ejercicios, Simuladores, Autoaprendizaje, Evaluación y Seguimiento) y el módulo de Administración, se encuentran estrechamente relacionados, montados sobre un servidor web Apache 2.2.3 que mantiene comunicación con el servidor de base de datos a través del protocolo TCP/IP, y donde las PCs clientes se mantiene realizando peticiones al sistema a través del protocolo segura HTTPS.

3.11 Prototipos Funcionales

Un prototipo funcional es un modelo, representación, demostración o simulación de un sistema, que incluye su funcionalidad de entradas y salidas. Está basado en imágenes reales del funcionamiento del sistema. A continuación se muestran los prototipos funcionales del CU Gestionar Curso.

Capítulo 3: Diseño e Implementación

Figura 18: Prototipo funcional CU Gestionar Curso. Escenario Listar Cursos.

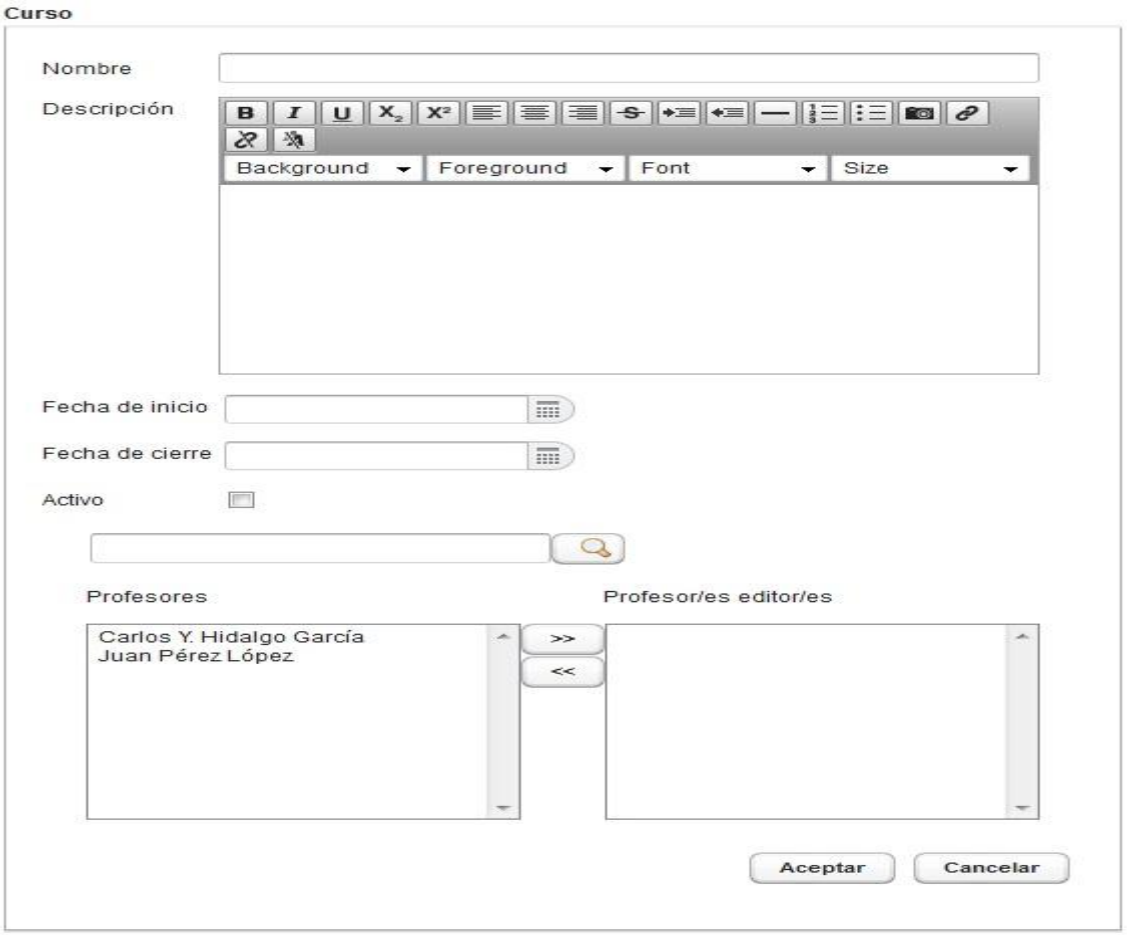

Figura 19: Prototipo funcional CU Gestionar Curso. Escenario Crear Curso.

Capítulo 3: Diseño e Implementación

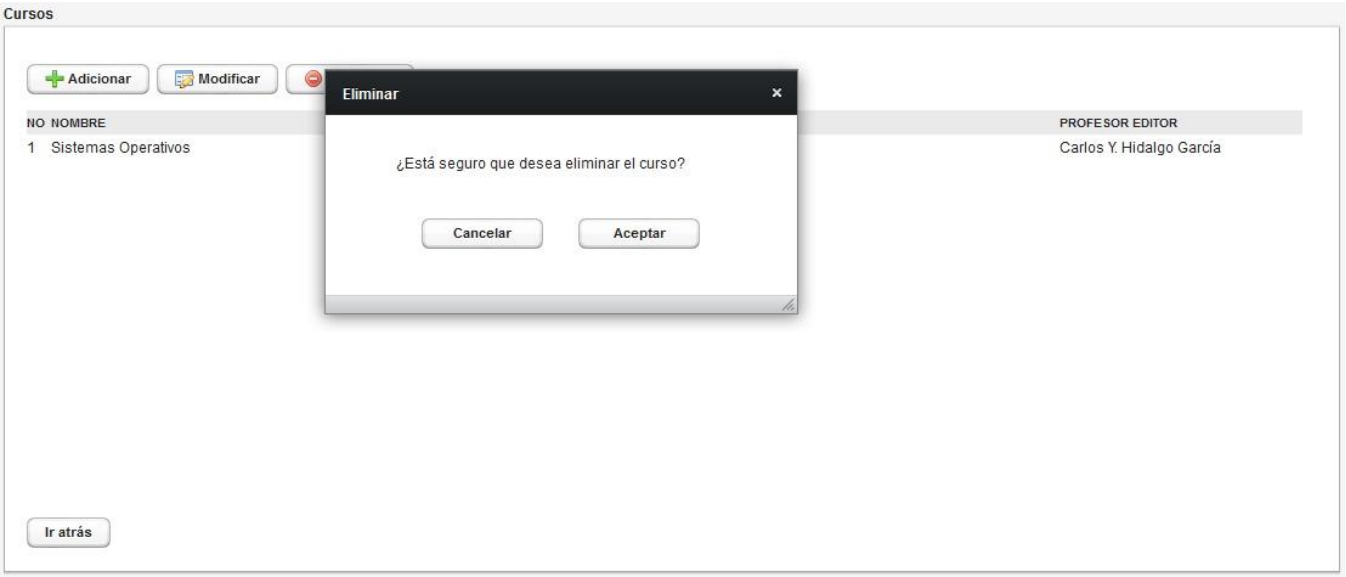

Figura 20: Prototipo funcional CU Gestionar Curso. Escenario Eliminar Curso.

3.12 Conclusiones parciales

Con la realización de este capítulo se arribó a las siguientes conclusiones:

- La construcción de los artefactos de flujo de trabajo de diseño permitió obtener cada uno de los elementos que describen el comportamiento del sistema en términos de diseño, siendo estos la entrada fundamental a las actividades de implementación.
- La aplicación de patrones de diseño propició obtener diagramas de clases de fácil entendimiento para la implementación del sistema.
- La definición de los estándares de codificación para la implementación del LAVSO, específicamente el módulo de Administración, facilitó la comunicación entre desarrolladores y la obtención de un código más legible.
- Se definieron los diagramas que describen la implementación, facilitando un entendimiento de cómo es la integración del sistema en términos de componentes y su futuro despliegue.

Capítulo 4: Validación de la Solución Propuesta

4.1 Introducción

El presente capítulo muestra los procedimientos y métodos utilizados para medir la factibilidad de los artefactos obtenidos en todas las etapas del proceso de desarrollo de software por las que se transitó en el módulo de Administración del LAVSO. Para la validación de los artefactos del modelo del sistema se muestran las listas de chequeo que le fueron aplicados, además se muestra la matriz de trazabilidad obtenida para dar seguimiento a los requisitos. También se mostrarán los principales prototipos funcionales que responden a las funcionalidades del sistema. Para la validación de los artefactos del modelo del diseño se aplicará la métrica del tamaño de clases. Por último se referenciarán las pruebas de caja negra realizadas al software.

4.2 Técnicas de validación de artefactos

La validación de artefacto es una tarea sumamente importante a lo largo de todo el proceso de desarrollo de software, puesto que ayuda a los trabajadores que intervienen en la obtención de un sistema a tener una idea de la calidad de los artefactos que han sido generados. RUP propone para las diferentes etapas, técnicas de validación, que van acorde al flujo de trabajo, específicamente con los artefactos obtenidos. A continuación se describen las principales validaciones realizadas para los artefactos obtenidos en el módulo de Administración del LAVSO.

4.3 Matriz de trazabilidad

En la etapa de Ingeniería de Requisitos es muy importante la gestión de requisitos, puesto que es el conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y sus cambios en cualquier momento. Esta actividad asegura la consistencia entre los requisitos y el sistema construido. Existen varias matrices de trazabilidad que permiten dar seguimiento a los requisitos, dentro de ellas la matriz de requisitos a Casos de Uso **(ver anexo 1).**

La aplicación de esta matriz de trazabilidad permitió a los analistas del sistema identificar qué requisito no se encontraba identificado en algún CU, conociendo que cada requisito debe al menos quedar reflejado en uno de los CU propuesto para el sistema. A continuación se muestra la leyenda utilizada para la identificación de los CU del módulo de Administración. La leyenda de los requisitos puede encontrarse en la sección 3 del capítulo 2 (2.3).

CU01 Gestionar Grupo
CU02 Simular Vista de Tipo de Usuario
CU03 Gestionar Miembros de un Grupo
CU04 Gestionar Subgrupo
CU05 Gestionar Miembros de un Subgrupo
CU06 Habilitar Módulos

Capítulo 4: Validación de la Solución Propuesta

CU07 Escoger Tipo de Evaluación
CU08 Gestionar Rango de Tipo de Evaluación
CU09 Editar Configuración
CU10 Visualizar Traza
CU11 Gestionar Rol
CU12 Gestionar Curso
CU13 Buscar Usuario
CU14 Gestionar Usuario
CU15 Autenticar Usuario
CU16 Matricular Usuario
CU17 Gestionar Tema

4.4 Prototipos de interfaz no funcional

Los prototipos no funcionales de interfaz de usuario son considerados una técnica de validación de requisitos, que permiten mostrarle al cliente y al desarrollador cómo debe quedar la aplicación. No siempre los prototipos funcionales deben ser fieles a los no funcionales, pero sí deben ser similares, pues con estos el cliente es capaz de analizar si la propuesta que se le hace del sistema cumple con sus expectativas. Para el módulo de Administración se obtuvieron los prototipos no funcionales que responden al 100% de los requisitos funcionales del sistema. Los prototipos no funcionales se encuentran referenciados en el documento Modelo del Sistema (40). A continuación se muestra el prototipo referente a la interfaz principal del módulo de Administración.

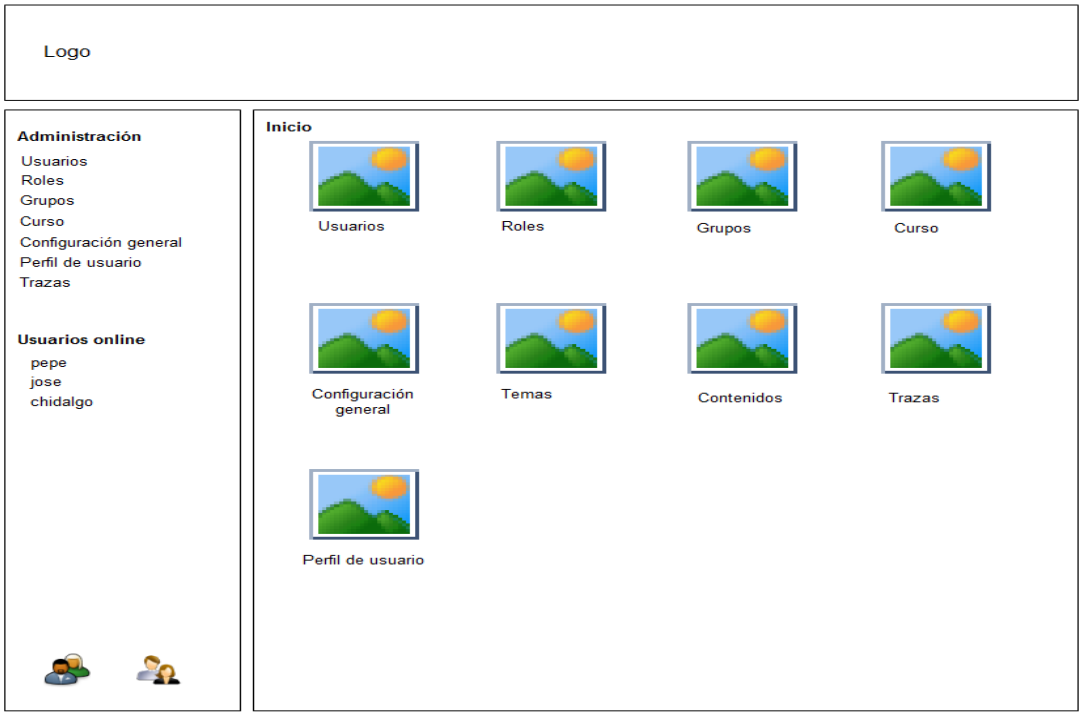

Figura 21: Prototipo no funcional. Interfaz principal de Administración

Capítulo 4: Validación de la Solución Propuesta

4.5 Lista de chequeo

Según Diana Susana Bichachi "S*e entiende por lista de chequeo (cheks-list) a un listado de preguntas, en forma de cuestionario que sirve para verificar el grado de cumplimiento de determinadas reglas establecidas a priori con un fin determinado. La lista de chequeo enumera una serie de ítems que deberían verificarse uno a uno para asegurarse de lograr el producto final con un nivel de calidad previamente aceptado*". (43)

Los requerimientos identificados con el cliente se especificaron en el artefacto llamado "Especificación de Requisitos", el cual con el objetivo de garantizar su factibilidad, fue sometido a varias iteraciones de revisiones por parte de calidad a nivel de proyecto. Para la prueba de este artefacto se aplicó la lista de chequeo definida por Calisoft para los requerimientos **(ver Anexo 2).** Durante la primera iteración de la revisión se identificó un total de dos no conformidades, a las cuales se les dio respuesta por parte del equipo de desarrollo en la siguiente y última iteración. Del total de las no conformidades, el 50% fue solucionado y el 50% no procede, por no concernir dicha no conformidad (NC) al módulo de Administración, sino al LAVSO como sistema.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Restricciones del diseño	1	No se propone ningún requisito para el diseño	Requisitos no funcionales, de restricción de diseño	Revisión		X	Proponer algún requisito para el diseño siempre que cumpla objetivo	PD 01/05/2012	Se agregó un requisito para el diseño y la implementación.
Identificación de requisitos de soporte	2	No se propone ningún requisito para el soporte	Requisitos no funcionales, de soporte	Revisión		X	Proponer algún requisito para el soporte siempre que cumpla objetivo	PD 01/05/2012	NP el soporte es para el LAVSO en general, no para el módulo de Administración

Tabla 2: Registro de dificultades detectadas en el documento de Especificación de Requisitos

Capítulo 4: Validación de la Solución Propuesta

También para verificar la calidad del documento Modelo del Sistema se aplicó la lista de chequeo correspondiente a este artefacto. En la primera iteración se detectaron cuatro NC que fueron solucionadas en la segunda iteración. Las dificultades detectadas fueron:

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Resumen	1	No se describe en el resumen como se inicia y como terminan las operaciones principales del CU.	Solo se le aplicó a los CU críticos.	Revisión	X		Describir correctamente el resumen	PD 10/05/2012	
Inicio de la descripción del CU	2	No se comienza diciendo “El caso de uso se inicia cuando el actor…”	Ningún CU cumple con este aspecto.	Revisión	X		Describir correctamente el inicio de la descripción textual del CU	PD 10/05/2012	
flujo alterno	3	En los CU que es necesario entrar datos no se hace un flujo alterno para la validación de estos.	Ningún CU cumple con este aspecto.	Revisión	X		Realizar el flujo alterno para todos los CU que lo requieran.	PD 10/05/2012	
CU extendidos e incluidos	4	No se mencionan ni se describen los CU incluidos y	Ningún CU cumple con este aspecto.	Revisión	X		Describir los CU incluidos y extendidos.	PD 10/05/2012	

		extendidos del CU base.							

Tabla 3: Registro de dificultades detectadas en el documento de Modelo de Sistema

4.6 Métrica de Tamaño de Clases (TC)

Con el objetivo de comprobar el adecuado diseño de las clases y el nivel de reutilización de las mismas se aplicó la métrica del TC. El tamaño general de una clase se puede determinar siguiendo los planteamientos a continuación:

- El número total de operaciones (tanto operaciones heredadas como operaciones privadas de la instancia) que están encapsuladas dentro de la clase.
- El número de atributos (tanto atributos heredados como atributos privados de la instancia) que están encapsulados en la clase.

Si existen valores grandes de TC, se estará demostrando que una clase puede tener demasiada responsabilidad, lo cual reduciría la reutilización de la clase y hará complicada la implementación y la prueba. De forma contraria sucede si los valores TC son de menor valor. Por otra parte es necesaria una evaluación concreta de las métricas mediante los umbrales. Algunos especialistas plantean la clasificación de la siguiente manera:

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

Tabla 4: Clasificación de los umbrales

Para verificar el TC de las clases se plantea la siguiente tabla con el nombre de las clases, cantidad de atributos y cantidad de métodos.

No	Clase	Cantidad de atributos	Cantidad de métodos	Umbral	Tamaño
1	CursoView.java	1	6	7	pequeño
2	FormCurso.java	3	2	5	pequeño
3	CursoController.java	1	6	7	pequeño
4	CursoDAO.java	0	6	6	pequeño

5	dCurso.java	7	3	10	pequeño

Tabla 5: Relación de clases del CU Gestionar Curso

Resultados obtenidos en la aplicación de la métrica:

Se le aplicó la métrica a un total de 5 clases. Todas correspondientes al CU Gestionar Curso, para un promedio de atributos de 2.4 y un promedio de operaciones de 4.6.

Umbral	Tamaño	Cantidad de Clases
<=20	**Pequeño**	**5**
>20 y <=30	**Medio**	**0**
>30	**Grande**	**0**

Tabla 6: Resultados obtenidos en la aplicación de la métrica TC

Después de mostrados los datos en las tablas anteriores se puede apreciar que todas las clases se clasifican en pequeñas, ninguna mediana, y ninguna grande, lo que implica un resultado positivo según los parámetros de calidad propuestos para esta métrica.

4.7 Casos de Prueba

Los casos de prueba son un conjunto de condiciones bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. (44) Para la validación de los casos de uso del módulo de Administración se obtuvo un total de diecisiete casos de pruebas, los cuales responden al 100 % de los casos de usos identificados. Seguidamente se describe el caso de prueba correspondiente al caso de uso Gestionar Curso.

Sección Crear Curso									
Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de Inicio**	**Fecha de Cierre**	**Activo**	**Profesores Editores**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Crear Curso	Se debe crear el curso.	V	V	V	V	V	V	El sistema crea correctamente el curso.	1. El administrador decide gestionar el curso. 2. El sistema muestra la

		V	V	V	I	V	V	El sistema muestra un mensaje de error.	interfaz de gestionar curso. 3. El administrador selecciona la opción Crear Curso. 4. El sistema muestra la interfaz crear curso. 5. El administrador introduce los datos correspondientes. 6. El administrador elige la opción aceptar. 7. El sistema guarda los datos. Termina así el caso de prueba.
		V	V	I	V	V	V	El sistema muestra un mensaje de error.	
		V	I	V	V	V	V	El sistema muestra un mensaje de error.	
		I	V	V	V	V	V	El sistema muestra un mensaje de error.	
EC 1.2 Datos incompletos	Debe mostrar un mensaje de error	V	V	V	V	V	I	El sistema muestra un mensaje de error.	1. El administrador decide gestionar el curso. 2. El sistema muestra la interfaz de gestionar curso. 3. El administrador selecciona la opción Crear Curso. 4. El sistema muestra la interfaz crear curso. 5. El administrador introduce los datos correspondientes. 6. El administrador elige la opción aceptar. 7. El sistema muestra un mensaje de error. Termina así el caso de prueba.
		V	V	V	I	V	V	El sistema muestra un mensaje de error.	
		V	V	I	V	V	V	El sistema muestra un mensaje de error.	
		V	I	V	V	V	V	El sistema muestra un mensaje de error.	
		I	V	V	V	V	V	El sistema muestra un mensaje de error.	

EC 1.3: Cancelar	Se debe cancelar la operación	NA	NA	NA	NA	NA	NA	El sistema cancela la operación.	1. El administrador decide gestionar el curso. 2. El sistema muestra la interfaz de gestionar curso. 3. El administrador selecciona la opción Crear Curso. 4. El sistema muestra la interfaz crear curso. 8. El administrador elige la opción cancelar. El sistema cancela la interfaz. Termina así el caso de prueba.

Tabla 7: Caso de Prueba Gestionar Curso. Sección Crear Curso

Sección Modificar Curso									
Escenario	**Descripción**	**Nombre**	**Descripción**	**Fecha de Inicio**	**Fecha de Cierre**	**Activo**	**Profesores Editores**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Modificar Curso	Se debe modificar el curso.	V	V	V	V	V	V	El sistema modifica correctamente el curso.	8. El administrador decide gestionar el curso. 9. El sistema muestra la interfaz de gestionar curso. 10. El administrador selecciona la opción Modificar Curso. 11. El sistema muestra la interfaz modificar curso. 12. El administrador introduce los datos correspondientes. 13. El administrador elige la opción aceptar. 14. El sistema guarda los datos. Termina así el caso de prueba.
		V	V	V	I	V	V	El sistema muestra un mensaje de error.	
		V	V	I	V	V	V	El sistema muestra un mensaje de error.	
		V	I	V	V	V	V	El sistema muestra un mensaje de error.	

		I	V	V	V	V	V	El sistema muestra un mensaje de error.	
EC 1.2 Datos incompletos	Debe mostrar un mensaje de error	V	V	V	V	V	I	El sistema muestra un mensaje de error.	1. El administrador decide gestionar el curso. 2. El sistema muestra la interfaz de gestionar curso. 3. El administrador selecciona la opción Modificar Curso. 4. El sistema muestra la interfaz modificar curso. 9. El administrador introduce los datos correspondientes. 10. El administrador elige la opción aceptar. 11. El sistema muestra un mensaje de error. Termina así el caso de prueba.
		V	V	V	I	V	V	El sistema muestra un mensaje de error.	
		V	V	I	V	V	V	El sistema muestra un mensaje de error.	
		V	I	V	V	V	V	El sistema muestra un mensaje de error.	
		I	V	V	V	V	V	El sistema muestra un mensaje de error.	
EC 1.3: Cancelar	Se debe cancelar la operación	NA	NA	NA	NA	NA	NA	El sistema cancela la operación.	1. El administrador decide gestionar el curso. 2. El sistema muestra la interfaz de gestionar curso. 3. El administrador selecciona la opción Crear Curso. 4. El sistema muestra la interfaz crear curso. 12. El administrador elige la opción cancelar. El sistema cancela la interfaz. Termina así el caso de prueba.

Capítulo 4: Validación de la Solución Propuesta

Tabla 8: Caso de Prueba Gestionar Curso. Sección Modificar Curso

Sección Eliminar Curso						
Escenario	**Descripción**	**Nombre**	**Descripción**	**Profesores Editores**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Eliminar Curso	Se debe eliminar el curso.	NA	NA	NA	El sistema elimina correctamente el curso.	15. El administrador decide gestionar el curso. 16. El sistema muestra la interfaz de gestionar curso. 6. El administrador elige el curso que desea eliminar. 17. El administrador elige la opción Eliminar Curso. 18. El sistema muestra un mensaje de confirmación. 19. El administrador elige la opción aceptar. 20. El sistema elimina el curso. Termina así el caso de prueba.
EC 1.2 Datos incompletos	Debe mostrar un mensaje de error	V	V	I	El sistema muestra un mensaje de error.	1. El administrador decide gestionar el curso. 2. El sistema muestra la interfaz de gestionar curso. 3. El administrador selecciona la opción Modificar Curso. 4. El sistema muestra la interfaz modificar curso. 13. El administrador introduce los datos correspondientes. 14. El administrador elige la opción aceptar. 15. El sistema muestra un mensaje de error. Termina así el caso de
		V	V	V	El sistema muestra un mensaje de error.	
		V	V	V	El sistema muestra un mensaje de error.	
		V	I	V	El sistema muestra un mensaje de	

Capítulo 4: Validación de la Solución Propuesta

					error.	prueba.
		I	V	V	El sistema muestra un mensaje de error.	
EC 1.3: Cancelar	Se debe cancelar la operación	NA	NA	NA	El sistema cancela la operación.	1. El administrador decide gestionar el curso. 2. El sistema muestra la interfaz de gestionar curso. 3. El administrador selecciona la opción Crear Curso. 4. El sistema muestra la interfaz crear curso. 5. El administrador elige la opción cancelar. 6. El sistema cancela la interfaz. Termina así el caso de prueba.

Tabla 9: Caso de Prueba Gestionar Curso. Sección Eliminar Curso

Se aplicaron los casos de prueba para la verificación de la calidad del software obteniéndose un total de 18 no conformidades (NC). Algunos ejemplos de NC son mostrados a continuación:

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Error ortográfico	1	La palabra Descripción no presenta tilde	Inicio/Grupos/ campo Descripción	Revisión	x		Agregarle la tilde a la palabra Descripción	PD 29/05/2012	
Error de funcionalidad	2	La funcionalidad añadir todos no funciona correctamente	Inicio/Grupos/ campo /Estudiantes/ funcionalidad Añadir todos	Revisión	x		Implementar la funcionalidad añadir todos	PD 29/05/2012	

Error al cargar la imagen	3	El avatar cargado no se muestra en su totalidad, solo la mitad de la imagen	Inicio/Usuario/Crear	Revisión	x		Lograr que la imagen se muestre completa	PD 29/05/2012	
Error de campo	4	El campo cantidad máxima de usuario no se encuentra validado	Inicio/Temas/adicionar/campo cantidad máxima de usuario	Revisión	x		Lograr que el sistema solo permita enteros	PD 29/05/2012	
Error al mostrar los estudiantes del subgrupo	5	En los detalles de un grupo muestra los nombres de los subgrupos que lo componen pero no los miembros de estos.	Inicio/Grupos/Detalles	Revisión	x		Agregar a la interfaz los miembros de los subgrupos de un curso.	PD 29/05/2012	

Tabla 10: Registro de dificultades detectadas en la aplicación de los casos de pruebas

4.8 Conclusiones parciales

En este capítulo se evidenció la importancia que tiene la realización de diferentes tipos de pruebas de software, pues contribuyeron a detectar errores en los diferentes artefactos generados. Al validar los artefactos obtenidos se arrojaron las siguientes conclusiones:

- Las listas de chequeo aplicadas al documento de Especificación de Requisitos y Modelo de Sistema, permitieron identificar las irregularidades que presentaban los mismos, obteniendo como resultado un total de dos y cuatro no conformidades, que en próximas iteraciones fueron solucionadas.
- Se obtuvo la matriz de trazabilidad como artefacto que permite dar seguimiento a los requisitos del software, contribuyendo a la gestión adecuada de estos, y verificando que la totalidad de los requisitos funcionales quedaran agrupados en al menos un CU.

Capítulo 4: Validación de la Solución Propuesta

- Los prototipos no funcionales de interfaz de usuario permitieron validar con el cliente la conformidad existente acerca de las funcionalidades que debe cumplir el sistema.
- La aplicación de métricas de calidad a las diferentes Clases del Diseño permitió evaluar algunos atributos de calidad como el tamaño de las clases, arrojando como resultado que todas las clases existentes en la implementación del sistema tiene un tamaño pequeño, demostrando así que fueron fácilmente implementadas.
- La aplicación de los casos de prueba contribuyó a detectar un total de 18 no conformidades, que fueron erradicadas en una segunda iteración del proceso de calidad.

.

Conclusiones Generales

Al finalizar el presente trabajo se arribó a las siguientes conclusiones:

- El análisis del modelo de dominio, junto a la interacción con los clientes del sistema y la aplicación de técnicas para la Elicitación de requisitos, propició que se obtuvieran resultados satisfactorios en la identificación de estos, tanto funcionales como no funcionales que debe cumplir el sistema.
- La especificación de los artefactos del modelo del sistema facilitó un mayor entendimiento y un acuerdo común entre los clientes y los desarrolladores, con respecto a las funcionalidades que el sistema debe brindar.
- La construcción de los artefactos del modelo de diseño permitió obtener los elementos que deberán ser implementados para obtener un sistema que cumpla con los requerimientos de software.
- La implementación de los requisitos identificados permitió obtener un sistema que contribuye a dar seguimiento y control a las acciones que se realizan en los restantes módulos y subsistemas de LAVSO.
- El análisis y validación de los artefactos obtenidos a lo largo del proceso de desarrollo de software permitió medir la calidad de cada uno de ellos.

Recomendaciones

Una vez culminada la aplicación que administra los subsistemas que componen el LAVSO, se recomienda:

- Realizar la integración del módulo de Administración con los restantes subsistemas del LAVSO, con el objetivo de que pueda ser utilizado como medio de enseñanza en la Universidad de las Ciencias Informáticas.
- Realizar una prueba piloto en un ambiente real de ejecución y con un mayor volumen de datos.
- Profundizar en el análisis realizado con el objetivo de encontrar nuevas funcionalidades que puedan ser incorporadas a la aplicación.

Bibliografía

1. **Colectivo de profesores de la asignatura de Sistemas Operativos, Facultad 3.** *Informe Semestral de Sistemas Operativos.* Habana : s.n., 2011-2012.

2. **Stallings, William.** *Sistemas Operativos.* Madrid : Prentice Hall.

3. **Fernández Rivera, Francisco.** *Guía Docente. Sistemas Operativos.* 2006.

4. **Euroamericana S, A.** Paginas Verdes. [En línea] 11. http://www.euram.com.ni/pverdes/verdes_informatica/informatica_al_dia/que_es_un_so_144.htm.

5. **Definicion.de.** Definición.de. [En línea] 2008-2012. http://definicion.de/administracion/.

6. **Geothesis.** Geothesis. [En línea] Geothesis, 2009. http://www.geothesis.com/index.php?option=com_content&view=article&id=479:-mo-de-administracin-un-sistema-informco&catid=21:artulos&Itemid=100.

7. **Instituto Internacional de Física Teórica y Aplicada Iowa Ames.** *Informe de la Reunión de Expertos sobre Laboratorios Virtuales.* EEUU : s.n., 2009.

8. *Laboratorios Virtuales.* **Vázquez Salas, Carlos.** 2009. 1988-6047.

9. *Laboratorios VIrtuales para la Enseñanza de Máquinas Computadoras.* **Corrales Barrios , Luis , Machado López, Libbis y Puentes Gómez , Antonio .** Camaguey : s.n.

10. **Compañía Ibercaja.** Laboratorio Virtual de Ibercaja. [En línea] Ibercaja, 2012. http://www.ibercajalav.net/.

11. **Asociación de Ciencias de Galicia.** Laboratorio Virtual de Física. [En línea] http://www.enciga.org/taylor/lv.htm.

12. **Gobierno del Principado de Asturias.** Fisquiweb. [En línea] http://web.educastur.princast.es/proyectos/fisquiweb/Laboratorio/AccesoZV.htm.

13. **Universidad Internacional de Catalunya.** Laboratorio Virtual. [En línea] UIC, 2012. http://www.uic.es/es/laboratorio-virtual.

14. *Aplicación del Sistema de Laboratorios a Distancia en Asignaturas de Regulación Automática.* **Santana, I., y otros.** 1697-7912, Santa Clara : s.n., 2010, Vol. 7.

15. *Laboratorio Virtual en Anestesiología.* **Rodríguez, Orlando L. y Iriarte, Leonel.** La Habana : s.n.

Bibliografía

16. *Laboratorio Virtual para la Docencia de Redes de Computadora.* **Berná, José A., y otros.** Alicante : s.n.

17. *Ventajas y desventajas de usar laboratorios virtuales en educación a distancia.* **Monge, Julián y Méndez, Víctor H.** 0379-7082, Costa Rica : Educación, 2007.

18. **IBM, Corp.** *Rational Unified Process.* s.l. : IBM, 2006.

19. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El lenguaje Unificado de Modelado. Manual de Referencia.2da Edición.* s.l. : Addison-Wesley, 2000.

20. **Visual Paradigm.** Visual Paradigm. [En línea] 2011. http://www.visual-paradigm.com/.

21. **The Apache Software Foundation.** Apache. [En línea] 2012. http://www.apache.org/.

22. **García, Javier, y otros.** *Aprenda Java.* San Sebastián : s.n., 2007.

23. **Roberts, Simon, Heller, Phillip y Ernest, Michael.** *Complete Java 2 Certification. Study Guide.* s.l. : SYBEX.

24. **Keith, Mike.** *Getting Started with JPA.* 2008. 978-1-934238-24-0.

25. **Grönroos, Marko.** *Book of Vaadin 4th edicion.* 2011.

26. **MicroSystems.** NetBeans. [En línea] Oracle, 2012. http://netbeans.org/index_es.html.

27. **Oracle.** *Ayuda del NetBeans 7.1 RC1.* s.l. : Oracle, 2011.

28. **ECURED.** Ecured. [En línea] 2012. http://www.ecured.cu/index.php/ER/Studio#Funcionalidades.

29. **The PostgreSQL Global Development Group.** *Ayuda del PgAdmin III, PostgreSQL.* 2012.

30. **Institute of Technology (M.I.T.).** *Ayuda del PgAdmin III.* 2009.

31. **IEEE.** *IEEE Computer Dictionary-Compilation of IEEE Standard Computer Glossaries.* s.l. : IEEE, 1990.

32. **Ian, Sommerville.** *Ingeniería de Software. 7ma edición.* Madrid : s.n.

33. **Silva, Andrés.** *Ingeniería de requisitos.*

Bibliografía

34. **Berrocal, Javier, García Alonso, José Manuel y Murillo Rodríguez, Juan Manuel.** *Patrones para la Extracción de Casos de Uso a partir de Procesos de Negocio.* Extremadura : s.n., 2009. 1988-3455.

35. *Patrones de Diseño, Refactorización y Antipatrones.* **Campo, Gustavo Damián.** Buenos Aires : s.n., 2009.

36. **Microsoft.** Microsoft ASP.net. [En línea] Microsoft . http://www.asp.net/mvc.

37. **Hommel, Scott.** Convenciones de código para el lenguaje de programación java. [En línea] Programación en Castellano. http://www.programacion.com/articulo/convenciones_de_codificacion_en_java_105.

38. **Soriano Betancourt, Arianne.** Modelo de Domino. La Habana : s.n., 2012.

39. **Soriano Betancourt, Arianne.** Especificación de Requisitos. La Habana : s.n., 2012.

40. **Soriano Betancourt, Arianne.** *Modelo de Sistema .* Habana : s.n., 2012.

41. **Soriano Betancourt, Arianne.** *Modelo de Diseño.* Habana : s.n., 2012.

42. **Soriano Betancourt, Arianne.** *Modelo de Implementación.* Habana : s.n., 2012.

43. *Cheks-list.* **Susana Bichachi, Diana.** El Salvador : s.n.

44. **Aristegui, José Luis.** *Test cases in software test.* Lámpsakos : s.n., 2010. 2145-4086.

Anexos

Anexo 1: Matriz de trazabilidad de Requisitos a Casos de uso.

RF/CU	CU01	CU02	CU03	CU04	CU05	CU06	CU07	CU08	CU09	CU10	CU11	CU12	CU13	CU14	CU15	CU16	CU17
RF_001	x																
RF_002	x																
RF_003	x																
RF_004	x																
RF_005	x																
RF_006	x																
RF_007			x														
RF_008			x														
RF_009			x														
RF_010				x													
RF_011				x													
RF_012				x													
RF_013				x													
RF_014				x													
RF_015					x												
RF_016					x												
RF_017					x												
RF_018														x			
RF_019														x			
RF_020														x			
RF_021														x			
RF_022													x				
RF_023														x			
RF_024														x			
RF_025											x						
RF_026											x						
RF_027											x						
RF_028											x						
RF_029												x					
RF_030												x					
RF_031												x					

Anexos

RF_032												x					
RF_033																x	
RF_034																x	
RF_035										x							
RF_036										x							
RF_037						x											
RF_038									x								
RF_039							x										
RF_040								x									
RF_041								x									
RF_042								x									
RF_043								x									
RF_044		x															
RF_045		x															
RF_046															x		
RF_047															x		
RF_048																	x
RF_049																	x
RF_050																	x

Tabla 11: Matriz de trazabilidad de Requisitos a Casos de uso

Anexo 2: Lista de chequeo aplicada al documento de Especificación de Requisitos.

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Evaluación**	**(NP)**	**Cantidad elementos afectados**	**Comentarios**
crítico	¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?	0			
crítico	¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)	0			

Anexos

Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Evaluación**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	¿Están todos los requisitos redactados de forma simple y clara para aquellos que vayan a consultarlo en un futuro?	0			
	¿Debería especificarse algún requisito con más detalle?	0			
	¿Debería especificarse algún requisito con menos detalles?		x		No es necesario porque todos los requisitos tienen un adecuado nivel de detalles.
	¿Todos los requisitos identificados se centran en lo que el sistema debe hacer y no cómo el sistema debe hacerlo?	0			
crítico	¿Han sido abordadas e identificadas los valores de entradas y salidas?	0			
	¿Se han identificado los requerimientos de software y de hardware?	0			
	¿Han sido identificadas las restricciones de diseño e implementación?	1			Solo se restringe la implementación, no así con el diseño.

Anexos

	¿Han sido identificadas las restricciones de interfaz externa?	0			
	¿Los requerimientos de soporte y usabilidad se han identificado?	1			No se hace referencia a los requisitos de soporte, solo a los requisitos que tributan a la usabilidad.
	¿Se han identificado los requerimientos de seguridad (confidencialidad, integridad, disponibilidad)?	0			
	¿Se puede verificar cada requisito? (Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina pueda chequear que el software satisface dicho requerimiento, ejemplo la especificación del caso de uso).	0			Todos los requisitos han sido asociados a un caso de uso y descrito correctamente.
	¿Se han enumerado los requisitos incluso los que se derivan de otros requisitos?	0			
	¿Se puede trazar cada requisito al origen en el entorno del problema, (caso de uso del negocio)?	0			Todos los requisitos tienen relación con el modelo de dominio descrito.
	¿Se han especificado todos los posibles	0			

	cambios en los requisitos, incluyendo la probabilidad de cambio?				
	¿No aparece un mismo requisito en más de un lugar del documento de especificación?	0			
crítico	¿No existe contradicción entre lo especificado por un requisito y lo especificado por otro?	0			
	¿Existe correspondencia entre el modelo de caso de uso, las Especificaciones Suplementarias y las especificaciones de requerimientos?	0			
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Evaluación**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	¿Ha identificado errores ortográficos?	0			
Crítico	¿Se entiende claramente lo que se ha especificado en el documento?	0			
	¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?	0			
	¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?	0			

Tabla 12: Lista de chequeo aplicada al documento de Especificación de Requisitos

Glosario de Términos

- **LAVSO**: Laboratorio Virtual de Sistemas Operativos.
- **CASE**: (Computer Aided Software Engineering), Ingeniería de Software Asistida por Ordenador.
- **UML**: (Unified Modeling Language), Lenguaje de Modelado Unificado.
- **RUP**: (Rational Unified Process), Proceso Unificado de Rational.
- **Caso de uso (CU)**: Representación de la agrupación de funcionalidades comunes.
- **RF**: Requisito Funcional.
- **RNF**: Requisito no Funcional.
- **API**: (Application Programming Interface), Interfaz de Programación de Aplicaciones.
- **JPA**: (Java Persistence API), API de Persistencia de Java.
- **ORM:** (Object Relational Mapping), Mapeo Objeto Relacional.
- **GOF**: (Gant of Four).
- **GRASP**: (General Responsibility Assignment Software Patterns), Patrones de Software para la Asignación General de Responsabilidades.
- **MVC**: Modelo Vista Controlador.