Universidad de las Ciencias Informáticas

Facultad 3

Sistema para la gestión del proceso de trabajos de diplomas en la Facultad 3

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

Autores:

Laura Romero Romero

Danet Nasco Castrillón

Tutor(es):

Dr. C Yoan Martínez Márquez

Ing. José Miguel Fabra Gallo

Cotutor:

Ing. Pedro Arango Astorga

La Habana, Cuba

Curso: 2018-2019

DECLARACIÓN DE AUTORÍA

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

Para que así conste firman la presente a los ___ días del mes de _____ del año _____.

_____ _____

Laura Romero Romero
(Autor)

Danet Nasco Castrillón
(Autor)

_____ _____

DrC. Yoan Martínez Márquez
(Tutor)

Ing. José Miguel Fabra Gallo
(Tutor)

_____

Ing. Pedro Arango Astorga

(Co-Tutor)

AGRADECIMIENTOS

Hoy es un día especial para mí, ya han pasado cinco cursos de sacrificios y alegrías, ha llegado el momento de enfrentarme al mundo como una profesional, son innumerables a las personas a las que tengo que agradecerles por ayudarme a llegar aquí.

A mi mami gracias por estar siempre en los momentos más difíciles, por el apoyo incondicional para seguir adelante, gracias por confiar en mí, gracias por tu amor.

A mi papi que ha sido un ejemplo a seguir, gracias por tu apoyo, por esas palabras alentadoras en los momentos difíciles, gracias por estar cuando te necesité, gracias por tu amor.

Gracias a ambos por existir, este es mi regalo para ustedes.

A mi hermana gracias por el amor que me has dado y por la confianza que has tenido en mí, aunque a veces me cuestiones si de verdad estudié informática.

Gracias a mis abuelas que, aunque ya no estén conmigo se sentirían orgullosas de mí, en especial a mi abuela Mariana por la fuerza y el aliento que me dio siempre.

A mi novio gracias por quererme tanto, gracias por la paciencia y la fuerza que me has dado en esta etapa final, te amo leoncito.

A toda mi familia gracias por estar siempre pendiente de mí.

A Alejandro gracias por apoyarme y ayudarme en estos años.

A la mejor compañera de tesis que puede haber tenido gracias por la paciencia que me has tenido.

A mis amigas Maite, Patry, Haraid, Linda Liz, Arlet gracias por estar a mi lado en los buenos y malos momentos.

A Yoandry gracias por tu apoyo y ayuda hasta el último momento.

A mis tutores gracias por todo el tiempo dedicado, en especial Fabra que siempre estuvo para apoyarnos sin importar el momento, gracias por confiar en nosotras.

A mis profesores que fueron mi guía en la carrera, y en especial a la profe Dariela que apoyó y confió en mí en todo momento, gracias por guiarme al mundo profesional porque sin su ayuda no hubiera llegado hasta aquí.

A mis compañeros de estudio, a los viejos y a los nuevos, quienes me ayudaron y a poyaron en algunos momentos, con los que compartí una magnífica etapa de mi vida.

Laura Romero Romero

AGRADECIMIENTOS

A mi mami por apoyarme todos estos años.

A mis tías por nunca haber perdido la fe en sus sobrinas.

A mi compañera de tesis Laura, por no permitir caer durante la batalla.

A Yoandry, por su comprensión, dedicación y paciencia en todo el proceso de tesis.

A mis amigos Selianne, Patry, Maite, Haraid, Ary, Sahyli, Cary, Carola y Hansel por su apoyo y amistad inquebrantable.

Agradezco inmensamente al Ing. Samuel por haberme guiado y apoyado durante varios años, en mi desarrollo académico e investigativo.

A todos los profesores que de una forma u otra formaron parte de mi formación como profesional. Especialmente a la profesora Dariela y a mis tutores: Fabra, Yoan y Pedro.

Danet Nasco Castrillón

DEDICATORIA

A mis padres y a mi hermana que son
las personas más importantes de mi vida.

Laura Romero Romero

A mi padre y a mi madre que
son mi inspiración y fortaleza.

Danet Nasco Castrillón

RESUMEN

En la carrera de Ingeniería en Ciencias Informáticas, el ejercicio final de culminación de estudios es un Trabajo de Diploma y está concebido, como un trabajo de ingeniería en algunos de los perfiles del ejercicio de la profesión del Ingeniero en Ciencias Informáticas Este ejercicio es el resultado de un proceso que transita por varias fases donde se genera gran cantidad de información relevante para el seguimiento y control de las tesis. En la actualidad, este proceso presenta un conjunto de carencias en la Facultad 3 de la Universidad de las Ciencias Informáticas; entre las más significativas se pueden relacionar: la diversidad de formatos en los que se emiten los reportes, y la dispersión y desactualización de la información. Es por ello que el objetivo de la presente investigación es desarrollar un sistema informático que contribuya a la estructuración, consistencia y persistencia de la información que se gestiona en el proceso de trabajos de diploma en esta área. Para el desarrollo del sistema se emplearon las siguientes tecnologías y herramientas: Visual Paradigm, NetBeans, PostgreSQL, UML, Symfony, PHP, Bootstrap, Apache, entre otras. La solución implementada favorece la centralización de los datos asociados al proceso de culminación de estudios de pregrado en la Facultad 3.

Palabras clave: consistencia, estructuración, persistencia, sistemas de gestión universitaria, trabajo de diploma.

ABSTRACT

In the Engineering degree in Computer Science, the final year of completion of studies is a Diploma Work and is conceived, as an engineering job in some of the profiles the exercise of the Computer Science Engineer's profession This exercise is the result of a process that goes through several phases where a large amount of relevant information is generated for the follow-up and control of the thesis. Currently, this process presents a set of deficiencies in the Faculty 3 of the University of Information Sciences; Among the most significant can be related: the diversity of formats in which the reports are issued, the dispersion and the outdated information. That is why the objective of this research is the development of a computer system that contributes to the structuring, consistency and persistence of the information that is managed in the process of diploma work in this area. For the development of the system, the following technologies and tools are used: Visual Paradigm, NetBeans, PostgreSQL, UML, Symfony, PHP, Bootstrap, Apache, among others. The implemented solution favors the centralization of the data associated with the completion process of the undergraduate studies in the Faculty 3.

Keywords: consistency, structuring, persistence, university management systems, diploma work.

Índice

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

Una de las características distintivas del siglo XXI es el desarrollo acelerado de las Tecnologías de la Información y las Comunicaciones (TIC); las cuales impactan en todas las esferas de la sociedad, cambiando los paradigmas sociales, económicos, políticos y culturales. En este proceso de cambios, las Instituciones de la Educación Superior (IES) juegan un papel importante debido a la responsabilidad que les ha correspondido de manera histórica: “anticiparse a las tendencias del mundo contemporáneo, y ser capaz de formar profesionales comprometidos con la sociedad y preparados para enfrentar los desafíos que la vida laboral les impone”. (1) En este contexto, adquiere relevancia la formación de profesionales, para utilizar las tecnologías de forma eficiente, así como su desarrollo y soporte.

Desde la década de 1970 se identifica en Cuba la necesidad de la participación activa y coherente de las IES en el desarrollo de las tecnologías, lo que se materializa con la creación de carreras afines a la Computación y la Electrónica.(2) A partir de entonces, el número y diversidad de carreras en la rama de las TIC ha experimentado un aumento significativo. Un hito importante en este proceso fue la creación en 2002 de la Universidad de las Ciencias Informáticas (UCI), la cual tiene la misión de formar profesionales comprometidos y altamente calificados en el área de las TIC; cuya función está asociada al desarrollo de la informatización de la sociedad desde tres aristas fundamentales: el desarrollo de la industria de software nacional, las transformaciones de procesos en las entidades para asumir su informatización y el soporte necesario para su mantenimiento. (2)

Una de las características distintivas de la UCI es su carácter de “universidad productiva” y la estrecha relación entre los procesos de formación, investigación y desarrollo de software. El currículo del Ingeniero en Ciencias Informáticas se divide en dos grandes ciclos, denominados: ciclo de integración básico (los cinco primeros semestres) y ciclo de integración profesional (los cinco últimos semestres). En el ciclo de integración básico, el énfasis se hace en la formación académica y tiene como objetivo fundamental la preparación para la incorporación al ciclo profesional. Por otro lado, en el ciclo de integración profesional el peso fundamental lo tiene la formación desde lo laboral y su objetivo fundamental es concretar el desempeño de los estudiantes en situaciones profesionales reales. (2) El ejercicio final de culminación de estudios para la carrera, es un Trabajo de Diploma que está concebido, como un trabajo de ingeniería en

alguno de los perfiles del ejercicio de la profesión del ingeniero informático. Este ejercicio es el resultado de un proceso, que transita por varias fases, las cuales se describen a continuación:

- **Elaboración del perfil de tesis**: en esta fase los aspirantes y tutores participan en la elaboración de los perfiles de tesis siguiendo las pautas establecidas en cada Facultad/Área.

- **Constitución del Comité o Comisión de Tesis (CT)**: se conforma en las facultades con la aprobación del Decano(a) y es el órgano docente encargado de la toma de decisiones sobre el proceso de desarrollo y seguimiento de los trabajos de diploma. Este órgano es el encargado de la clasificación de los proyectos de tesis en correspondencia con su área del conocimiento, donde se genera un acta de constitución de los CT.

- **Revisión y aprobación de los perfiles de tesis**: en esta fase juegan un papel activo los miembros del CT y los tutores. Una vez aprobados los perfiles, no podrán modificarse el objetivo general o el problema de la investigación científica definidos inicialmente en el perfil del Trabajo de Diploma, exceptuando los casos avalados por el CT, de esta fase se genera un acta de aprobación, siguiendo las pautas establecidas en la Facultad/Universidad.

- **Realización de los seminarios de tesis**: este es un acto de carácter evaluativo dentro del desarrollo del Trabajo de Diploma. En cada seminario debe dejar constancia escrita de las valoraciones realizadas por los miembros del CT, el tutor y el aspirante. De igual manera, se debe reportar al Vicedecano(a) de Formación un resumen del estado de los trabajos de diploma, incidencias y acciones correctivas en caso de ser necesario.

- **Constitución de los Tribunales de Tesis**: los tribunales y oponentes de los trabajos de diploma son definidos por el CT y aprobados por el Decano(a) de la Facultad, donde se genera una resolución decanal, siguiendo las pautas establecidas en la Facultad/Universidad.

- **Defensa de trabajos de diploma**: es un ejercicio académico donde se evalúa la calidad de la investigación desarrollada, el documento de tesis entregado, la defensa que se realiza en acto público ante el tribunal, y la expresión de un comportamiento ético y moral conforme a la culminación de estudios de un Ingeniero en Ciencias Informáticas.

Como se evidencia en lo antes descrito, la gestión de los trabajos de diploma genera gran cantidad de información relevante para su seguimiento y control. En la actualidad, este proceso presenta un conjunto de carencias en la Facultad 3, que inciden negativamente en la gestión de la información asociada a las tesis. A continuación, se relacionan las que se consideran, en mayor medida, impactan negativamente:

- Los documentos que se gestionan (perfiles, actas, informes, etc.) no tienen una estructura estándar que facilite la compresión y el intercambio de información entre los numerosos actores que intervienen en este proceso (tutores, miembros de los CT y directivos docentes)
- Los reportes de seguimiento se recogen en varios formatos (csv, ods, xlsx, documentos word, pdf, txt) y el control de versiones depende de la persona o el nivel al que se gestiona, lo cual puede traer consigo inconsistencias en la información que se reporta.
- La información se almacena en varios formatos y soportes, pudiendo darse la pérdida de la información y dificultad para consultar datos históricos del proceso.

Los elementos antes referidos inciden negativamente, afectando el manejo de estadísticas, la toma de decisiones y, por ende, la planificación y calidad del proceso.

A partir de la problemática antes descrita se identifica como **problema a resolver:** ¿Cómo gestionar la información asociada al proceso de trabajos de diploma en la Facultad 3, de manera que contribuya a su estructuración, consistencia y persistencia?

Este problema se enmarca en el **objeto de estudio**: Proceso de desarrollo de sistemas de gestión universitaria.

Su **campo de acción** se enfoca en la: Gestión de trabajos de diploma de la Facultad 3 en sistemas de gestión universitaria.

Para darle solución al problema, se plantea como **objetivo general**: Desarrollar un sistema informático que contribuya a la estructuración, consistencia y persistencia de la información que se gestiona en el proceso de trabajos de diploma en la Facultad 3, desglosado en los siguientes **objetivos específicos**:

- Establecer los referentes teórico-metodológicos de sistema de gestión universitario particularizando la gestión de trabajos de diploma.

- Realizar el análisis y diseño de la propuesta de solución como aproximación a la implementación.
- Implementar el sistema para la gestión de los trabajos de diploma en la Facultad 3.
- Valorar la viabilidad de la propuesta de solución.

Se plantea como **idea a defender** que si se desarrolla un sistema informático para la gestión de los trabajos de diploma en la Facultad 3 se contribuirá a la estructuración, consistencia y persistencia de la información asociada a este proceso.

El **método Científico de Investigación** es la forma de estudiar los fenómenos de la naturaleza y la sociedad para descubrir sus relaciones y su esencia, el mismo se puede clasificar en métodos de nivel teórico o empírico los cuales están relacionados entre sí de forma dialéctica. A continuación, se muestran los métodos utilizados para el desarrollo de esta investigación.

Métodos del nivel teórico

- Método Analítico-Sintético: Posibilitó realizar un análisis de las distintas partes que afectan el objeto de estudio y sintetizar los elementos más significativos.
- Método Histórico-Lógico: Para la realización de la investigación se hizo necesario estudiar la evolución del problema y la existencia de metodologías, procedimientos y sistemas informáticos similares al que se pretende elaborar; determinando cuáles son las tendencias actuales para el desarrollo de sistemas de gestión académica.

Métodos del nivel empírico

- Observación: Se empleó para identificar características en el proceso de gestión de tesis como la forma de realización, quiénes intervienen, qué utilizan. De igual manera se empleó para identificar las carencias más significativas en este proceso.
- Entrevista: Se utilizó para obtener información acerca de cómo se desarrolla el proceso de gestión de trabajos de diploma y las limitaciones que se detectaron por parte de profesores, estudiantes y directivos docentes de la Facultad 3.

El documento de presentación de los resultados de la investigación está estructurado en 3 capítulos:

Capítulo 1 Fundamentos teórico-metodológicos: Se presentan los elementos teóricos referente al proceso de gestión de trabajos de diploma, que sirven de base a la investigación del problema planteado. En este capítulo se analizarán los principales conceptos relacionados con el objeto de estudio, se realiza una descripción de los sistemas existentes de gestión académica; y una explicación de herramientas y tecnologías potenciales a utilizar en el desarrollo de la propuesta de solución.

Capítulo 2 Descripción de la propuesta de solución: Se describe la propuesta de solución y los artefactos generados por el escenario número 4 en la metodología Agile Iniciad Procesos (AUP) en su variación para la UCI en las etapas de análisis, diseño e implementación.

Capítulo 3 Valoración de la viabilidad de la propuesta de solución: Se valora la viabilidad de la propuesta de solución mediante la validación de los requisitos y el diseño propuesto, además de la realización de pruebas funcionales, unitarias y de aceptación al sistema informático SIGETESIS.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO-METODOLÓGICOS

En este capítulo se presentan los elementos teóricos referente al proceso de gestión de trabajos de diploma, que sirven de base a la investigación del problema planteado. En este capítulo se analizarán los principales conceptos relacionados con el objeto de estudio, se realiza una descripción de los sistemas existentes de gestión académica; y una explicación de herramientas y tecnologías potenciales a utilizar en el desarrollo de la propuesta de solución.

1.1 Referentes teóricos asociados al dominio del problema.

Tesis de grado o Trabajo de Diploma

Una tesis de grado es un trabajo de investigación que se realiza al término de una carrera universitaria. Amplía o profundiza en un área del conocimiento humano aportando una novedad o una revisión crítica aplicando lo aprendido en la carrera, utilizando métodos científicos.(3) Una vez que el estudiante culmina, o está a punto, la carga académica correspondiente al plan de estudio de la carrera que cursa, se le exige como requisito indispensable para optar al título que realice una tesis o trabajo de grado. Esto no es más que la presentación de los resultados logrados por él, al llevar a cabo una investigación destinada a ampliar el acervo de conocimientos existentes, hasta ese momento, en el ámbito de la carrera o especialidad que estudia o una aplicación de ese conocimiento en la solución de algún problema. El proceso desarrollado para su realización es evaluado en distintas oportunidades por parte de representantes de la institución o universidad y, en la mayoría de las veces, debe exponerse (defenderse) de manera oral ante un jurado que se designa especialmente para ello. (3)

En Cuba el Trabajo de Diploma o Tesis de grado es la forma que adopta el ejercicio final de varias carreras y es reconocido por el Reglamento del Trabajo Docente y Metodológico de la Educación Superior. En este Reglamento, se define como un tipo de evaluación de la culminación de los estudios cuyo objetivo es comprobar el grado de dominio de los estudiantes de los objetivos generales de la carrera, mediante la solución, con independencia y creatividad, de un problema propio de la profesión, utilizando la metodología de la investigación científica. (3)

En el desarrollo de los trabajos de diploma, los profesores deben promover la iniciativa, la independencia y la creatividad de los estudiantes, garantizando el trabajo individual y estimulando el análisis interdisciplinario en la resolución del problema objeto de la investigación.

Los profesores que dirigen los trabajos de curso y los trabajos de diploma son designados por el decano de la facultad-carrera, por el jefe del departamento-carrera o por los directores de los centros universitarios municipales y filiales, según el caso, garantizando que tengan la preparación adecuada en correspondencia con los objetivos y la complejidad del tema que desarrollarán los estudiantes. Pueden realizar esta labor también, especialistas reconocidos de las entidades laborales de base, las unidades docentes y centros de investigación. (3)

1.2 Uso de las TIC en la gestión de los trabajos de diploma.

El creciente papel de las Tecnologías de la Información y las Comunicaciones (TIC) en las diferentes esferas de la sociedad, es tema de estudio y reflexión de múltiples investigadores, organismos, gobiernos y organizaciones de todo el mundo. Su auge ha estado acompañado por un gran avance en la gestión de la información y el conocimiento. (2) La incorporación de las TIC en la sociedad y en especial en el ámbito de la educación ha adquirido relevancia y ha evolucionado exponencialmente en los últimos años, ejemplo de ello es el uso de sistemas informáticos para la gestión de los procesos universitarios como los que se describen a continuación.

1.2.1 Sistemas Internacionales

TESEO: Es un sistema de gestión académica, que se encarga de gestionar los resúmenes de las tesis doctorales, posee una base de datos que mantiene el registro de las tesis doctorales declaradas aptas. Todo el resultado del módulo de graduación es un fichero de texto que se envía por correo y que es procesado por el Consejo de Universidades para incorporar los datos a una base de datos temporal, y una ficha impresa que se envía por correo. El Consejo coordina los datos de la ficha con los del fichero, los pasa de la base de datos temporal a la base de datos definitiva, momento en que estarán disponibles para su consulta. El sistema TESEO cuenta con las siguientes funcionalidades:

- Permite añadir, modificar y consultar los resúmenes de las tesis doctorales.
- Genera un fichero con los datos de la universidad a la que pertenece el estudiante y los datos de la dirección de la universidad.
- Imprime la ficha generada con los datos de los estudiantes.(4)

TESIS: Es un sistema que fue desarrollado por los Servicios Informáticos de la Universidad de Castilla-La Mancha en el año 1998, a propuesta del Archivo General Universitario, en colaboración con la Secretaría General y el Vicerrectorado de Ordenación Académica. La aplicación permite realizar, de forma compartida y automatizada, la gestión administrativa de las tesis doctorales presentadas en la institución. Mediante TESIS se controla el trámite y la localización física de la tesis doctoral a lo largo de todo el procedimiento para su lectura, desde el trámite de presentación al de archivo, incluyendo además otras posibilidades como es el análisis del contenido o la información relativa a la posterior publicación de estos trabajos de investigación. (5)

1.2.2 Sistemas Nacionales

SIGENU (Sistema de Gestión de la Nueva Universidad): Es un sistema nacional de gestión de información para centros de educación superior creado en el ISPJAE (Instituto Superior Politécnico José Antonio Echevarría) en abril del 2007, que tiene el fin de gestionar toda la información académica vinculada a la educación superior en Cuba. Este sistema es una aplicación Desktop que cuenta con seis módulos que gestionan el proceso de matrícula en las universidades, toda la información con que debe contar el sistema (datos personales, académicos, datos de servicios militar y laborales) y modificar datos de estudiantes existentes en el sistema. El sistema gestiona lo referente a los planes de estudio de cada carrera y permite gestionar los trabajos de diploma de cada estudiante en cuanto a nota y tema de este. (6)

Sistema Gestor de Trabajos de Diploma(SGTD): Es un intento de informatizar el proceso de trabajos de diploma en la antigua Facultad 9 de la Universidad de Ciencias Informáticas en el año 2009. Este sistema compuesto por un módulo Web y un módulo escritorio muestran información referente a todo el proceso y permiten la gestión de usuarios, la gestión de solicitudes de perfiles, la gestión de tribunales, comité de tesis; así como la planificación de evaluaciones y la generación de reportes. (7)

A continuación, se presenta una tabla comparativa entre los sistemas antes descritos, tomando como referencia los siguientes indicadores: el tipo de tesis que gestiona (doctoral, de maestría o de pregrado), la plataforma sobre la que fue desarrollado (Web o Desktop), la licencia bajo la cual se distribuye, si el sistema se utiliza aún y el tipo de software (privativo o libre). Estos indicadores responden a las variables de la investigación y las principales demandas del cliente: un sistema que gestione la información asociada a las tesis de pregrado, que sea web, para

facilitar el acceso de los actores del proceso, y que esté en correspondencia con los principios de soberanía tecnológica.

	TESEO	TESIS	SINEGU	SGTD
Tipo de tesis	Doctoral	Doctoral	Pregrado	Pregrado
Plataforma	BD_Web + Aplicación Desktop	BD_Web + Aplicación Desktop	Desktop	Web + Desktop
Licencia	La universidad se reserva el derecho de la licencia de la aplicación			
Vigencia	Vigente	No vigente	Vigente	No se implementó
Tipo de Software	Privativo	privativo	libre	privativo

Tabla 1.Comparación entre sistemas de gestión de trabajos de diploma

Un análisis global de los sistemas de gestión estudiados, evidencia que tanto los sistemas internacionales (Tesis y TESEO), como los sistemas nacionales (SIGENU Y SGTD) no cumplen las demandas del cliente. En el caso de Tesis, TESEO y SGTD están desarrolladas con tecnologías privativas, lo que va en contra del proceso de migración al software libre que se lleva a cabo en la Facultad 3 y en la UCI. Por su parte, SIGENU solo gestiona las notas y nombres de los estudiantes. Por lo antes mencionado, se decide desarrollar un sistema que cumpla con las demandas del cliente y el dominio del problema.

1.3 Metodología de desarrollo de software

Una metodología de desarrollo de software se puede definir como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda en la construcción de un software. (8) Actualmente, existen diversas metodologías recogidas en dos grandes grupos, las denominadas metodologías tradicionales o formales y las ágiles o ligeras. El primer grupo se caracteriza por centrarse en llevar una documentación exhaustiva de todo el desarrollo del software, cumplir con la planificación realizada en la fase inicial del proyecto. y mostrar cierta resistencia a los cambios. Se caracteriza por necesitar un proyecto de gran

cantidad de participantes, pues requiere de un equipo de trabajo capaz de administrar un proceso complejo en varias etapas.

Por su parte, las metodologías ágiles, se caracterizan por ser más orientadas al desarrollo de software, con bajos niveles de formalización en la documentación requerida y por ser, a diferencia de las tradicionales, más adaptables a los cambios, requerir de pequeños grupos de trabajo y por ser apropiadas para entornos volátiles. (8)

Teniendo en cuenta las características del equipo de desarrollo (2 miembros + cliente), la participación activa de los clientes en la solución y el carácter cambiante de los requisitos, se decide utilizar una metodología ágil; de ellas, se resuelve utilizar AUP (variación para la UCI) para estar en correspondencia con las buenas prácticas definidas en la universidad para el desarrollo de software.

El Proceso Unificado Ágil de Scott Amblar o Agile Iniciad Procesos (AUP) es una versión simplificada del Proceso Unificado de Racional (RUP). Esta describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva. (9)

En el caso de los proyectos de desarrollo de software de la UCI, de las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide mantener la fase de Inicio, pero modificando el objetivo de la misma y se unifican las restantes 3 fases de AUP en una sola, a la que se denomina Ejecución y se agrega la fase de Cierre. Los elementos antes descritos, se detallan la siguiente tabla: (9)

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración		

Construcción **Transición**	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Tabla 2. Fases de la metodología AUP

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso del Negocio (CUN), Descripción de Proceso de Negocio (DPN) o Mapa Conceptual (MC))y existen tres formas de encapsular los requisitos (Casos de Uso del Sistema (CUS), Historias de Usuario (HU), Descripción de Requisito por Proceso (DRP)), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma: (9)

Escenario No 1:

Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

Escenario No 2:

Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

Escenario No 3:

Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

Escenario No 4:

Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

Teniendo en cuenta lo antes expuesto, referidos a los escenarios de la disciplina de requisitos de software de la metodología AUP variación UCI, se selecciona el escenario No.4 ya que aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos.

1.4 Arquitectura de software

La arquitectura de software de un sistema informático comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos. La arquitectura no es el software operacional. Más bien, es la representación que capacita al ingeniero del software para: (8)

- Analizar la efectividad del desafío para la consecución de los requisitos fijados.
- Considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el desafío es relativamente fácil.
- Reducir los riesgos asociados a la construcción del software.

La arquitectura de software es importante por las 3 razones siguientes:

- Facilitan la comunicación entre todas las partes (partícipes) interesadas en el desarrollo de un sistema basado en computadora.
- Destaca decisiones tempranas de desafío que tendrán un profundo impacto en todo el trabajo de ingeniería del software que sigue, y es tan importante en el éxito final del sistema como una entidad operacional.

- Constituye un modelo relativamente pequeño e intelectualmente comprensible de cómo debe estar estructurado el sistema y cómo trabajan juntos sus componentes. (8)

El modelo arquitectónico y los patrones arquitectónicos contenidos dentro son transferibles. Esto es, los estilos y patrones de arquitectura pueden ser aplicados en el desafío de otros sistemas y representados a través de un conjunto de abstracciones que facilitan a los ingenieros del software la descripción de la arquitectura de un modo predecible. (8)
Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

- **Descomposición Modular**: El software se estructura en grupos funcionales muy acoplados.
- **Cliente-servidor**: El software reparte su carga de cómputo en dos partes independientes, pero sin reparto claro de funciones.
- **Arquitectura de tres niveles**: Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente. (8)

Otras arquitecturas menos conocidas son:

- Modelo Vista Controlador.
- En pipeline.
- Entre pares.
- Orientada a servicios (SOA del inglés Service-Oriented Architecture).
- Arquitectura de microservicios (MSA del inglés MicroServices Architecture). Algunos consideran que es una especialización de una forma de implementar SOA.
- Dirigida por eventos.
- Máquinas virtuales. (8)

1.4.1 Estilos arquitectónicos

Cada estilo arquitectónico describe una categoría del sistema que contiene: un conjunto de componentes, que realiza una función requerida por el sistema, un conjunto de conectores que posibilitan la comunicación, la coordinación y la cooperación entre los componentes; restricciones que definen como se puede integrar los componentes que forman el sistema; y modelos semánticos que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes. De estos, se puede mencionar que: (10)

- Sirven para sintetizar estructuras de soluciones.
- Pocos estilos abstractos encapsulan una enorme variedad de configuraciones concretas.
- Definen los patrones posibles de las aplicaciones.
- Permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales.

Existen diferentes tipos de estilos arquitectónicos entre los que se encuentran:

- Centrada en Datos.
- Llamada y retorno.
- Orientada a Objetos.
- Estratificada.

1.5 Tecnologías y herramientas para el desarrollo de la solución

En el desarrollo de software, un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio. (11)

Symfony 3.0

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones Web mediante algunas de sus principales características: (12)

- Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web.
- Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja.
- Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.
- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las librerías de otros fabricantes

Motor de Plantillas Twig

Twig es un moderno motor de plantillas para PHP, el cual posee las características siguientes: (13)

- Rápido: Twig compila las plantillas hasta el código PHP optimizado. La sobrecarga en comparación con el código PHP normal se redujo al mínimo.
- Seguro: Twig tiene un modo para evaluar el código de la plantilla no confiable. Esto permite que Twig se utilice como lenguaje de plantilla para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla.
- Flexible: Twig es impulsado por un analizador flexible. Esto permite al desarrollador definir sus propias etiquetas y filtros personalizados, y crear su propio DSL (La línea de abonado digital o línea de suscriptor digital).

Doctrine 2.4

Doctrine es el proyecto de varias bibliotecas PHP enfocadas principalmente en el almacenamiento de bases de datos y el mapeo de objetos. Los proyectos centrales son el Asignador Relacional de Objetos (ORM) y la Capa de Abstracción de Base de Datos (DBAL) sobre la que se construye. (14)

Bootstrap 4.1

Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end. La información básica de compatibilidad de sitios web o aplicaciones está disponible para todos los dispositivos y navegadores. Desde la versión 2.0 también soporta diseños web adaptables. Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado. (15)

Visual Paradigm para UML 15.1

Suite de productos para desarrollar software de manera eficiente, rápida y de forma colaborativa. Permite realizar Diagramas de procesos de negocios, Modelado UML, Diagramas de casos de usos, Diagramas de actividad, de interacción, de bases de datos, de entidad-relación. Además, posee integración para varios IDE, puede realizar Ingeniería de Código y también generar documentación, entre otras cosas; todo bajo un modelo colaborativo. Visual Paradigm Suite soporta todas las necesidades de diseño y modelado a lo largo del ciclo de vida de desarrollo de software,

es una herramienta que ayuda a construir aplicaciones de calidad, de manera más rápida, óptima y más barata.(16)

Modelado de Procesos de Negocio 2.0

La Notación para el Modelado de Procesos de Negocio (Business Process Modeling Notation, BPMN, por sus siglas en inglés) ha sido creada para proporcionar un lenguaje unificado que sea comprensible tanto para los analistas de negocio como para los expertos del área de tecnología. Es una notación gráfica que describe la lógica de los pasos en un proceso de negocio, define un modelo para este proceso basándose en diagramas de flujo. Provee una notación común para que las personas relacionadas con los procesos puedan expresarlos gráficamente en una forma más clara, estandarizada y completa. Ha sido diseñada especialmente para coordinar la secuencia de procesos y mensajes que fluyen entre participantes de actividades distintas. Además, facilita no solo la estandarización de los procesos dentro de la organización, sino que amplía el campo de acción para que estos puedan ser compartidos y entendidos entre los diferentes socios del negocio. (17)

Lenguajes de Programación (PHP 7.0)

PHP (acrónimo de "Hypertext Preprocessor") es un lenguaje interpretado de "código abierto", de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Está diseñado especialmente para desarrollo web. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. (18)

JavaScript 1.8

Es un lenguaje de programación interpretado, es decir, que no requiere compilación, es utilizado principalmente en páginas web, con una sintaxis semejante a la de los lenguajes Java y C. Es un lenguaje orientado a objetos el cual se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. JavaScript se ejecuta en el cliente al mismo tiempo que las sentencias van descargándose junto con el código HTML. (19)

JQuery 1.8

Es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. jQuery es la biblioteca de JavaScript más utilizada. Es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. (20)

Netbeans 8.2

NetBeans es un IDE escrito en Java, de código abierto (OpenSource), gratuito para desarrolladores de software. Ofrece todas las herramientas necesarias para crear aplicaciones profesionales, empresariales, web y móviles con los lenguajes Java, JavaFX, C/C++ y lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby. NetBeans es fácil de instalar y se puede ejecutar en los sistemas operativos Windows, Linux, Mac OS X y Solaris. La plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio de gran tamaño. Ofrece servicios comunes permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. (21)

PostgreSQL 9.5

Es un sistema de gestión de base de datos relacional de objetos (ORDBMS) basado en POSTGRES, Versión 4.2, desarrollado en el Departamento de Ciencias de la Computación de la Universidad de California en Berkeley. POSTGRES fue pionero en muchos conceptos que solo estuvieron disponibles en algunos sistemas de bases de datos comerciales mucho más tarde. Es compatible con una gran parte del estándar SQL y ofrece muchas características modernas como consultas complejas, llaves extranjeras, vistas actualizables, integridad transaccional, control de concurrencia multiversión. (22)

PgAdmin III 1.22

Es una aplicación de diseño y manejo de bases de datos para su uso con PostgreSQL. La aplicación se puede utilizar para manejar PostgreSQL 7.3 y superiores. Este software fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas

SQL a la elaboración de bases de datos complejas. La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor de la sintaxis SQL, un editor de código del lado del servidor, un agente para la programación de tareas «SQL/batch/shell» y soporte para el motor de replicación Slony-I (sistema de replicación maestro-esclavo asíncrono para el DBMS PostgreSQL, que brinda soporte para la conexión en cascada y la conmutación por error). La conexión del servidor se puede realizar mediante TCP/IP o Unix Domain Sockets (socket de comunicación entre procesos), y puede ser cifrado mediante SSL por seguridad. No se requieren controladores adicionales para comunicarse con la base de datos del servidor. (23)

Apache 2.2

Apache un servidor web caracterizado por su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. La licencia Apache es una descendiente de la licencias BSD. El servidor web Apache corre en una multitud de sistemas operativos, lo que lo hace prácticamente universal. Apache es una tecnología gratuita de código abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si se quiere ver qué es lo que se instala como servidor, se pueda saber, sin ningún secreto, sin ninguna puerta trasera. Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que sean instalados cuando sea necesario. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un módulo para realizar una función determinada. Apache trabaja con gran cantidad de lenguajes como Perl, PHP y otros lenguajes de script. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto. Tiene una alta configurabilidad en la creación y gestión de registros. Permite la creación de ficheros de registro a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en el servidor. (24)

1.6 Conclusiones parciales

El estudio de los referentes teóricos y metodológicos asociados a la gestión de los trabajos de diploma permitió una mejor comprensión sobre el dominio de la investigación. Los sistemas estudiados (contexto nacional e internacional) no son viables ya que no cumplen con las necesidades del cliente y varios están desarrollados con tecnologías privativas; estos elementos justifican el desarrollo de un sistema informático que cumpla con las demandas del cliente. El estudio de las herramientas y tecnologías , permitió profundizar los conocimientos necesarios para el desarrollo del Sistema para la Gestión de trabajos de diploma para la Facultad 3.

Capítulo 2: DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

En el presente capítulo se describe la propuesta de solución y los artefactos generados por el escenario número 4 en la metodología Agile Iniciad Procesos (AUP) en su variación para la UCI en las etapas de análisis, diseño e implementación.

2.1 Análisis del proceso de gestión de tesis

El proceso que se lleva a cabo en la Universidad de las Ciencias Informáticas referente a la evaluación de la culminación de estudios es realizado a partir del tipo de evaluación defensa del trabajo de diploma, definido en el Reglamento para el Trabajo Docente-Metodológico en la educación superior. (25) Dicho proceso tiene inmerso varios subprocesos (Ver Figura 1), los cuales se describen a continuación:

- **Asignación de tesis a estudiante:** La Facultad asigna, a los estudiantes y tutores, temas de trabajos de diplomas, los cuales pueden ser desarrollados por 2 estudiantes como máximo.
- **Aprobación de Perfil de tesis:** Los tesistas y tutores participan en la elaboración de los perfiles de tesis. Los tutores, conjunto al vicedecano de formación y otros profesores se reúnen y aprueban o sugieren modificaciones a cada perfil.
- **Confección de los comités o comisiones de tesis:** Se elaboran en la Facultad por el vicedecano de formación y son aprobados por el decano.
- **Evaluación en seminarios de tesis:** Se realizan 3 seminarios, el primero se mide el alcance finalmente del tema de la tesis. En cada seminario se evalúa el avance de la tesis con respecto a la fecha en la que se realiza y se emite un estado final de cada tesis (atrasada o en tiempo).
- **Confección de los tribunales de tesis:** Se elaboran los tribunales por el Comité de tesis y aprobados por el Decano(a). Entre sus miembros se encuentra el vocalista, el secretario, el presidente y el oponente.
- **Predefensa de tesis:** Es la antesala de la defensa final de la tesis, donde se define si dicha tesis tiene la calidad requerida para presentarse a la defensa final.

- **Defensa de tesis:** Ejercicio académico donde se evalúa la calidad de la investigación desarrollada mediante la revisión del documento de tesis entregado, la expresividad ética y moral correspondiente a un Ingeniero en Ciencias Informáticas.

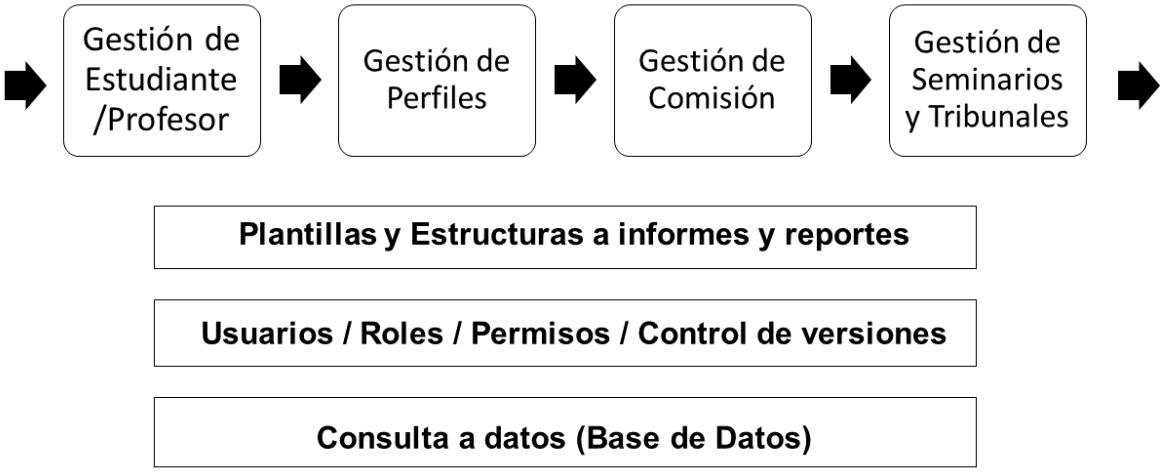

Figura 1. Descripción de la propuesta de solución.

2.2 Requisitos de software

2.2.1 Requisitos Funcionales

Los requisitos funcionales son aquellos que indican lo que debe hacer el producto, son las capacidades con las que debe cumplir el mismo. (8) En la presente investigación, a partir de entrevistas (Ver Anexo 1) y reuniones con el cliente, se obtuvieron en total 37 requisitos funcionales, los cuales se muestran en la siguiente tabla:

Nº	Nombre	Descripción	Prioridad	Complejidad
RF1	Autenticar usuario	El usuario introduce sus datos, el sistema lo comprueba y si son válidos, el usuario queda autenticado con el nivel de privilegios asignados; si los datos no son válidos el sistema muestra un mensaje de error.	Alta	Alta

RF2	Crear usuario	El sistema permite que los usuarios con rol administrador creen nuevos usuarios con el nombre, usuario, y rol.	Alta	Media
RF3	Modificar usuario	El sistema permite que los usuarios con rol administrador modifiquen los usuarios ya existentes.	Alta	Baja
RF4	Eliminar usuario	El sistema permite que los usuarios con rol administrador eliminen cuentas de usuarios existentes.	Alta	Media
RF5	Listar usuario	El sistema permite que los usuarios con rol administrador visualicen el listado de usuarios existentes.	Baja	Media
RF6	Mostrar calendarios de eventos	El sistema permite que los usuarios visualicen el flujo de eventos con respecto al procesos de trabajo de diploma.	Alta	Media
RF7	Crear perfil	El sistema permite que los usuarios con el rol de tutor crear los perfiles de trabajos de diploma con el título, nombre del tutor(es), el nombre del autor(es), la clasificación, la línea científica, descripción de la necesidad, situación problemática, problema a resolver, objeto de estudio, objetivo general, campo de acción, objetivos específicos, posible resultado, cronograma y las tareas asignas.	Alta	Alta
RF8	Modificar perfil	El sistema permite que los usuarios con el rol de tutor modifiquen los perfiles de trabajos de diploma con el título, nombre	Alta	Alta

		del tutor(es), el nombre del autor(es), la clasificación, la línea científica, descripción de la necesidad, situación problemática, problema a resolver, objeto de estudio, objetivo general, campo de acción, objetivos específicos, posible resultado, cronograma, tareas asignas y estado. Los usuarios con el rol de coordinador de la comisión y secretario modifiquen el estado del perfil.		
RF9	Listar perfil	El sistema permite que los usuarios existentes visualicen los perfiles de trabajos de diploma.	Media	Baja
RF10	Exportar	El sistema permite que los usuarios puedan exportar los perfiles de trabajos de diploma.	Baja	Media
RF11	Filtrar perfil	El sistema permite que los usuarios con el rol de tutor, estudiante, vicedecano, profesor principal de año académico, asesor del vicedecano y miembros de las comisiones y tribunales puedan buscar los perfiles de trabajos de diploma.	Baja	Baja
RF12	Crear seminario	El sistema permite que los usuarios con el rol de coordinador y secretario de la comisión puedan crear los seminarios de trabajos de diploma con la fecha, hora, local.	Alta	Alta
RF13	Modificar seminario	El sistema permite que los usuarios con el rol de coordinador y secretario de la comisión puedan modificar los seminarios	Alta	Media

		de trabajos de diploma con la fecha, hora, local, estado de la tesis y observaciones. El tutor emitirá su criterio respecto a la tesis.		
RF14	Listar seminario	El sistema permite que los usuarios existentes puedan visualizar los seminarios de trabajos de diploma.	Alta	Baja
RF15	Filtrar seminario	El sistema permite que los usuarios con el rol de tutor, vicedecano, profesor principal de año académico, asesor del vicedecano y miembros de las comisiones y tribunales puedan buscar los seminarios de trabajos de diploma.	Alta	Baja
RF16	Crear defensa	El sistema permite que los usuarios con el rol de presidente o secretario del tribunal puedan crear seminario de defensa con la hora, fecha y local.	Alta	Alta
RF17	Modificar defensa	El sistema permite que los usuarios con el rol de presidente o secretario del tribunal puedan modificar seminario de defensa con la hora, fecha y local y la nota emitida en la defensa.	Alta	Media
RF18	Listar defensa	El sistema permite que los usuarios existentes visualicen las defensas de trabajos de diploma.	Alta	Alta
RF19	Crear comisión	El sistema permite que los usuarios con el rol de vicedecano de formación o asesor pueda crear los comité o comisiones de trabajos de diploma con el nombre de la	Alta	Alta

		comisión, el coordinador, el secretario y los miembros.		
RF20	Modificar comisión	El sistema permite que los usuarios con el rol de vicedecano de formación o asesor puedan modificar los comité o comisiones de trabajos de diploma con el nombre de la comisión, el coordinador, el secretario y los miembros.	Alta	Media
RF21	Eliminar comisión	El sistema permite que los usuarios con el rol de vicedecano de formación o asesor pueda eliminar los comité o comisiones de trabajos de diploma.	Alta	Media
RF22	Listar comisión	El sistema permite que los usuarios existentes puedan visualizar los comité o comisiones de trabajos de diploma	Alta	Baja
RF23	Filtrar comisión	El sistema permite que los usuarios con el rol de tutor, vicedecano, profesor principal de año académico, asesor del vicedecano y miembros de las comisiones puedan buscar los comité o comisiones de trabajos de diploma.	Baja	Baja
RF24	Exportar comisión	El sistema permite que los usuarios puedan exportar los comité o comisiones de trabajos de diploma	Media	Alta
RF25	Crear tribunal	El sistema permite que los usuarios con el rol de vicedecano de formación y asesor creen los tribunales de trabajos de diploma con el presidente, secretario, vocal, oponente y posibles suplentes.	Alta	Alta

RF26	Modificar tribunal	El sistema permite que los usuarios con el rol de vicedecano de formación y asesor modifiquen los tribunales de trabajos de diploma con el presidente, secretario, vocal, oponente y posibles suplentes.	Alta	Media
RF27	Eliminar tribunal	El sistema permite que los usuarios con el rol de vicedecano de formación o asesor pueda eliminar los tribunales de trabajos de diploma.	Alta	Media
RF28	Listar tribunal	El sistema permite que los usuarios existentes puedan visualizar los tribunales de trabajos de diploma.	Alta	Media
RF29	Filtrar tribunal	El sistema permite que los usuarios con el rol de tutor, vicedecano, profesor principal de año académico, asesor del vicedecano y miembros de las comisiones puedan buscar los tribunales de trabajos de diploma.	Baja	Media
RF30	Exportar tribunal	El sistema permite que los usuarios puedan exportar los tribunales	Media	Alta
RF31	Elaborar aval	El sistema permite que los usuarios con el rol de presidente o secretario del tribunal elaboren el aval de trabajos de diploma.	Alta	Alta
RF32	Mostrar resumen estadístico de aprobación de perfil	El sistema permite que los usuarios existentes puedan visualizar los resúmenes estadístico de aprobación de perfil.	Baja	Baja
RF33	Mostrar resumen estadístico por	El sistema permite que los usuarios existentes puedan visualizar los	Baja	Baja

	áreas de la Facultad	resúmenes estadístico por áreas de la Facultad		
RF34	Mostrar resumen estadístico de otras áreas	El sistema permite que los usuarios existentes puedan visualizar los resúmenes estadístico de otras áreas	Baja	Baja
RF35	Exportar resumen estadístico por áreas de la Facultad	El sistema permite que los usuarios existentes puedan exportar un listado los resúmenes estadístico por áreas de la Facultad.	Baja	Baja
RF36	Exportar resumen estadístico de aprobación de perfil	El sistema permite que los usuarios existentes puedan exportar un listado los resúmenes estadístico de aprobación de perfil.	Baja	Baja
RF37	Exportar resumen estadístico de otras áreas	El sistema permite que los usuarios existentes puedan exportar un listado los resúmenes estadístico de otras áreas.	Baja	Baja

Tabla 3. Descripción de los requisitos funcionales del sistema.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) se refieren a las características o cualidades que debe tener el producto para hacerlo atractivo, usable, rápido y confiable. (8) Para el desarrollo del sistema se definieron 5 RNF, los que se clasificaron en RNF de software, de hardware, de interfaz, de seguridad, que se encuentran a continuación:

No.	Descripción	Tipo de requisito
RNF1	La interfaz debe ser sencilla, intuitiva, legible, amigable y mantener el formato en vistas similares.	Requisito de interfaz o apariencia externa

RNF2	Se requiere de un navegador web instalado en las computadoras clientes.	Requisito de Software
RNF3	El sistema requiere de una computadora que haga la función de servidor, esta debe cumplir con las siguientes características: • Memoria RAM con 1 GB o más • Capacidad de Disco Duro de 50 GB o más	Requisito de Hardware
RNF4	La información estará protegida contra accesos no autorizados mediante mecanismos de autenticación y encriptación	Requisito de Seguridad
RNF5	Para garantizar la integridad de la información se utilizarán políticas de salvas de la base de datos.	Requisito de Seguridad

Tabla 4. Descripción de los requisitos no funcionales del sistema.

2.3 Historias de Usuario

Las historias de usuarios son las técnicas utilizadas para especificar los requisitos del software, son equivalentes a los casos de uso en el proceso unificado y constituyen la base para las pruebas funcionales. (8) En total se definieron 37 historias de usuarios, a continuación se muestra la historia de usuario de mayor criticidad, “Modificar seminario”:

Número: 7	**Requisito:** Modificar seminario
Programador: Laura Romero Romero	**Iteración Asignada:1**
Prioridad: Alta	**Tiempo Estimado:14 horas**

Riesgo en Desarrollo:	**Tiempo Real:**
Descripción: El componente permitirá modificar los datos de un perfil de trabajo de diploma seleccionado. Del perfil se podrán modificar los campos recogidos en la HU 6: Crear seminarios de trabajos de diploma**.** **Tesis:** Campo de selección donde se debe escoger la tesis asignada a esta comisión. Es obligatorio. **Estado de la tesis:** Campo de selección que permite escoger el tipo de estado por el que se presenta la tesis. Es obligatorio. **Observaciones:** Campo de texto donde se registran algunas observaciones del trabajo de diploma. **Criterio del tutor:** Campo de texto donde se registra el criterio del tutor(es). Es obligatorio.	
Observaciones: Al modificar un seminario de trabajos de diploma se necesita llenar los campos obligatorios.	
Prototipo elemental de interfaz gráfica de usuario:	

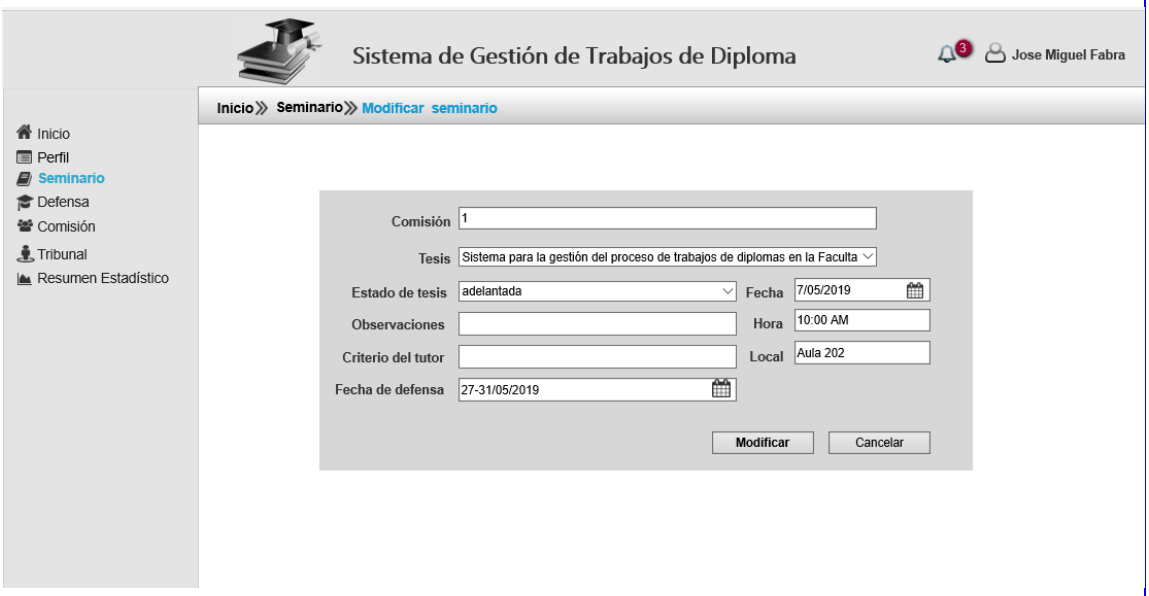

Tabla 5. Historia de usuario del requisito funcional Modificar seminario.

2.4 Descripción de la arquitectura

La arquitectura del software constituye una especificación de las principales ideas del diseño proporcionando una descripción más detallada de cómo realizar dicho sistema. (10)

Patrón de arquitectura Modelo Vista Controlador

La propuesta de solución está basada en el patrón Modelo-Vista-Controlador, el cual describe la forma de organizar el código de la aplicación separando los datos de la misma, la interfaz de usuario, y la lógica de control en tres componentes distintos:

- Modelo: Componente encargado del acceso a datos, en este componente se encuentran las tablas de la base de datos.
- Vista: Transforma el modelo en una página web que permite al usuario interactuar con ella, en este componente están las clases servidoras, las clientes, los formularios, la portada y el Layout, el Layout tiene relación de inclusión con todas las clases servidoras, las servidoras construyen las clientes y las clientes contienen los formularios. La portada es la que se relaciona directamente con el componente controlador.
- Controlador: Se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista, (10) en este componente esta la clase controladora y la index.php, la controladora se encarga de manejar toda la información tanto del modelo como de la vista y la index.php es la que se relaciona con la portada y envía información a los componentes de Symfony.

En la siguiente figura, se muestra cómo interactúan estos tres componentes en la arquitectura definida:

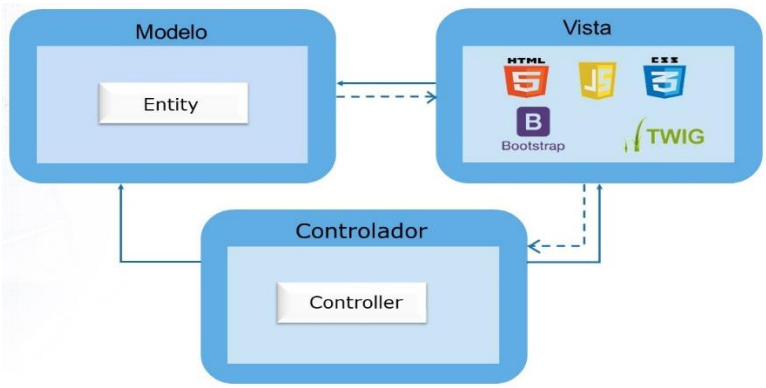

Figura 2. Arquitectura Modelo Vista Controlador.

A continuación, se muestra la aplicación de la arquitectura a la propuesta de solución:

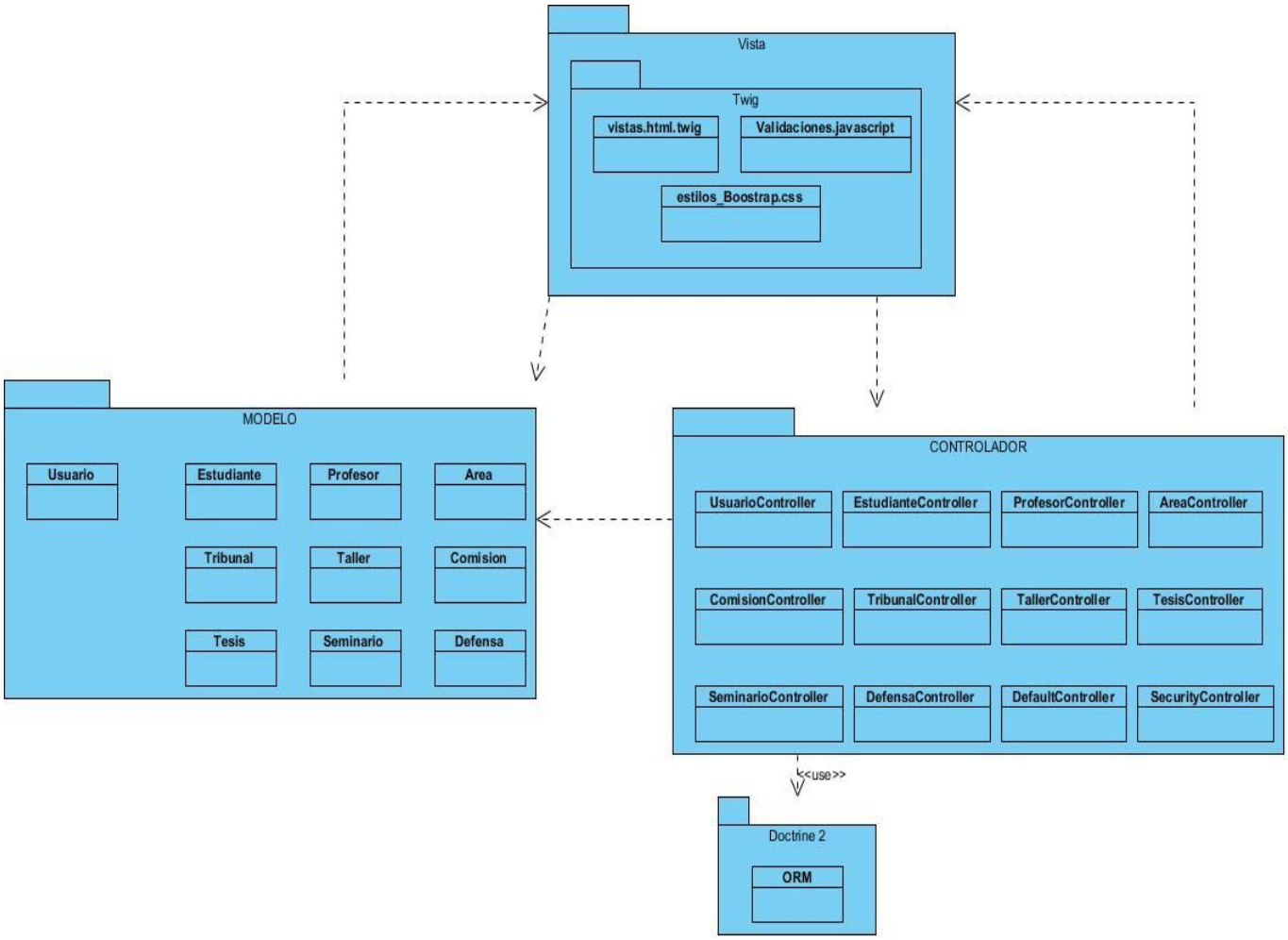

Figura 3. Aplicación de la arquitectura a la solución propuesta.

2.5 Diagrama de clases diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases. Tiene el propósito de describir las clases que conforman el modelo de un determinado sistema. Dado el carácter de refinamiento iterativo que caracteriza un desarrollo orientado a objetos, el diagrama de clase va a ser creado y refinado durante las fases de análisis y diseño, estando presente como guía en la implementación del sistema. (10)

A continuación, se muestra el diagrama de clases de la propuesta de solución:

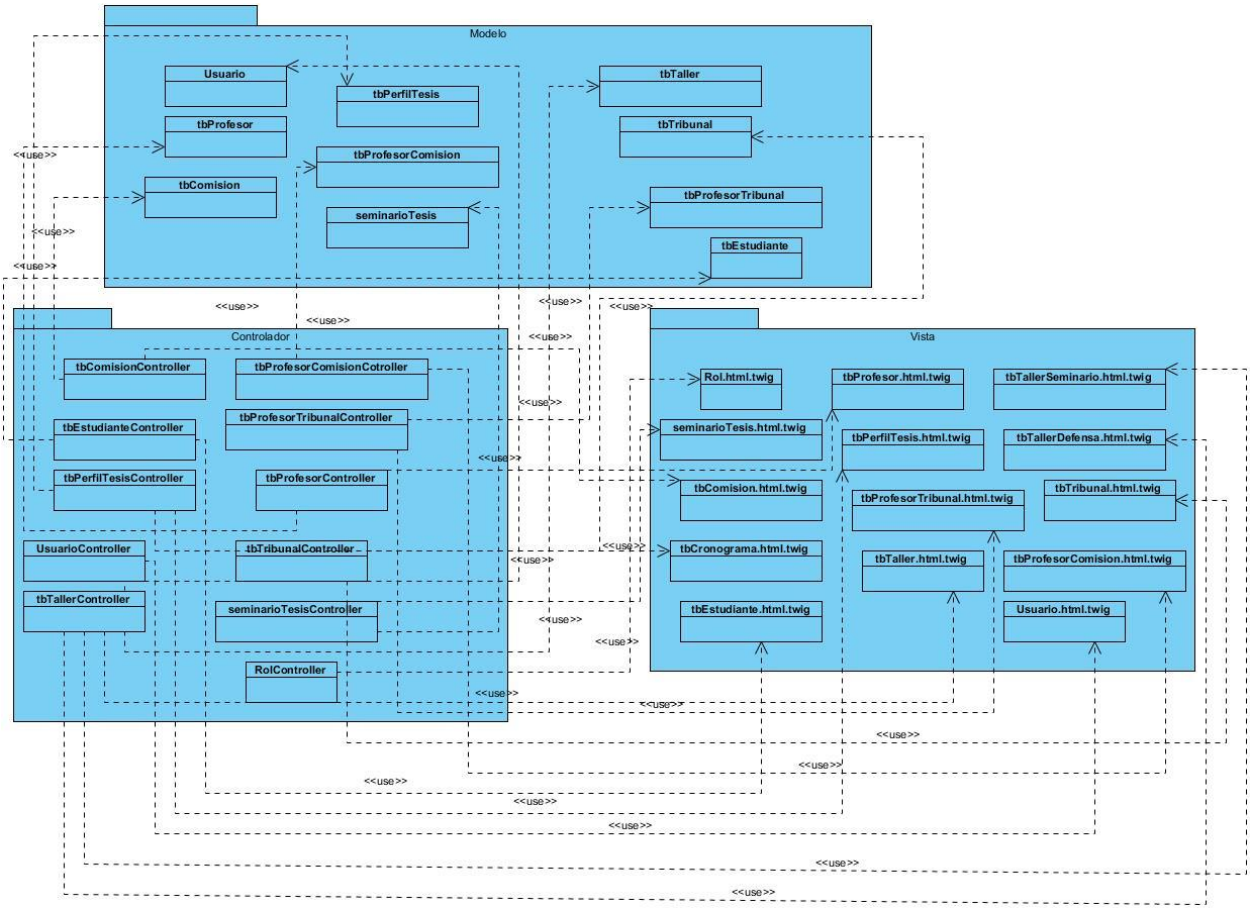

Figura 4. Diagrama de clases del diseño.

2.6 Patrones de diseño utilizados

Los patrones de diseño son aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. (10)

Patrones GRASP: es un acrónimo de General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para

sugerir la importancia de aprehender (grasping en inglés) estos principios para diseñar con éxito el software orientado a objetos. Los patrones de asignación de responsabilidades GRASP dan la medida de un refinamiento del diseño, los cuales se describen a continuación: (10)

Experto: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Se evidencia en la solución en las clases entidades, que son las que cuentan con la información necesaria para cumplir la responsabilidad sobre los elementos del negocio, crearlos y modificarlos, así como las clases repositorio.

Creador: El patrón creador permite identificar quién debe ser el responsable de la instanciación de nuevos objetos o clases. Este patrón se utilizó para identificar qué clase A debe crear elementos de una clase B, apoyándose en que la clase A debería: contener, agregar, registrar, utilizar y tener los datos de inicialización de la clase B. Se evidencia que la clase tbComisionController tiene métodos y funcionalidades correspondientes para crear las instancias de los objetos tbComision.

Controlador: Permite asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Se evidencia en las clases controladoras que se encargan de obtener datos, enviarlos a las bibliotecas y a las vistas.

Bajo acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento. El grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta Cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño.

Alta Cohesión: Asignar una responsabilidad de modo que la cohesión siga siendo alta. Las clases controladoras contienen los patrones de bajo acoplamiento y alta cohesión nivelados, pues permite el uso de los componentes de forma individual, evidenciando el bajo acoplamiento, así como la dependencia entre ellos o alta cohesión.

Patrones GoF: Los patrones que se presentan proceden de Design Patterns, un libro básico y muy popular que presenta 23 patrones que son útiles durante el diseño de objetos. Puesto que el libro fue escrito por cuatro autores, estos patrones se conocen como los patrones de la "pandilla de los cuatro" o patrones "GoF". Los patrones GoF describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación entre clases y la formación de estructuras de mayor complejidad. Nos permiten crear grupos de objetos para ayudarnos a realizar tareas complejas.

Existen tres tipos de patrones: de creación, estructurales y de comportamiento. Los patrones de creación abstraen la forma en la que se crean los objetos, permitiendo tratar las clases a crear de forma genérica dejando para más tarde la decisión de qué clases crear o cómo crearlas, los cuales se describen a continuación: (10)

Patrón Decorator: Este método pertenece a la clase base.html.twig, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. Este archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en al resto de las vistas (clases html.twig). Este procedimiento es una implementación del patrón Decorator.

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Este patrón se refleja en las clases controladoras que son instancias únicas.

Mediator (Mediador): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto. Se evidencia en las librerías, las cuales son mediadoras entre las clases controladoras y las modelos o acceso a datos.

2.7 Modelo de datos

Las bases de datos necesitan una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. La estructura tiene que ser desarrollada para la necesidad de las aplicaciones que la usarán. La puesta en práctica de la base de datos es el paso final en el desarrollo de aplicaciones. Se conforman con los requisitos del proceso del negocio, que es la primera abstracción de la vista de la base de datos. (8)

Un modelo de datos se puede definir como un conjunto de conceptos, reglas y convenciones bien definidos que permiten aplicar una serie de abstracciones a fin de describir y manipular los datos de un cierto mundo real que se desea almacenar en la base de datos. El modelo relacional se caracteriza a muy grandes rasgos por disponer que toda la información debe estar contenida en tablas, y las relaciones entre datos deben ser representadas explícitamente en esos mismos datos. (8)

A continuación, se muestra el diseño del esquema de base de datos propuesto a través del modelo de datos:

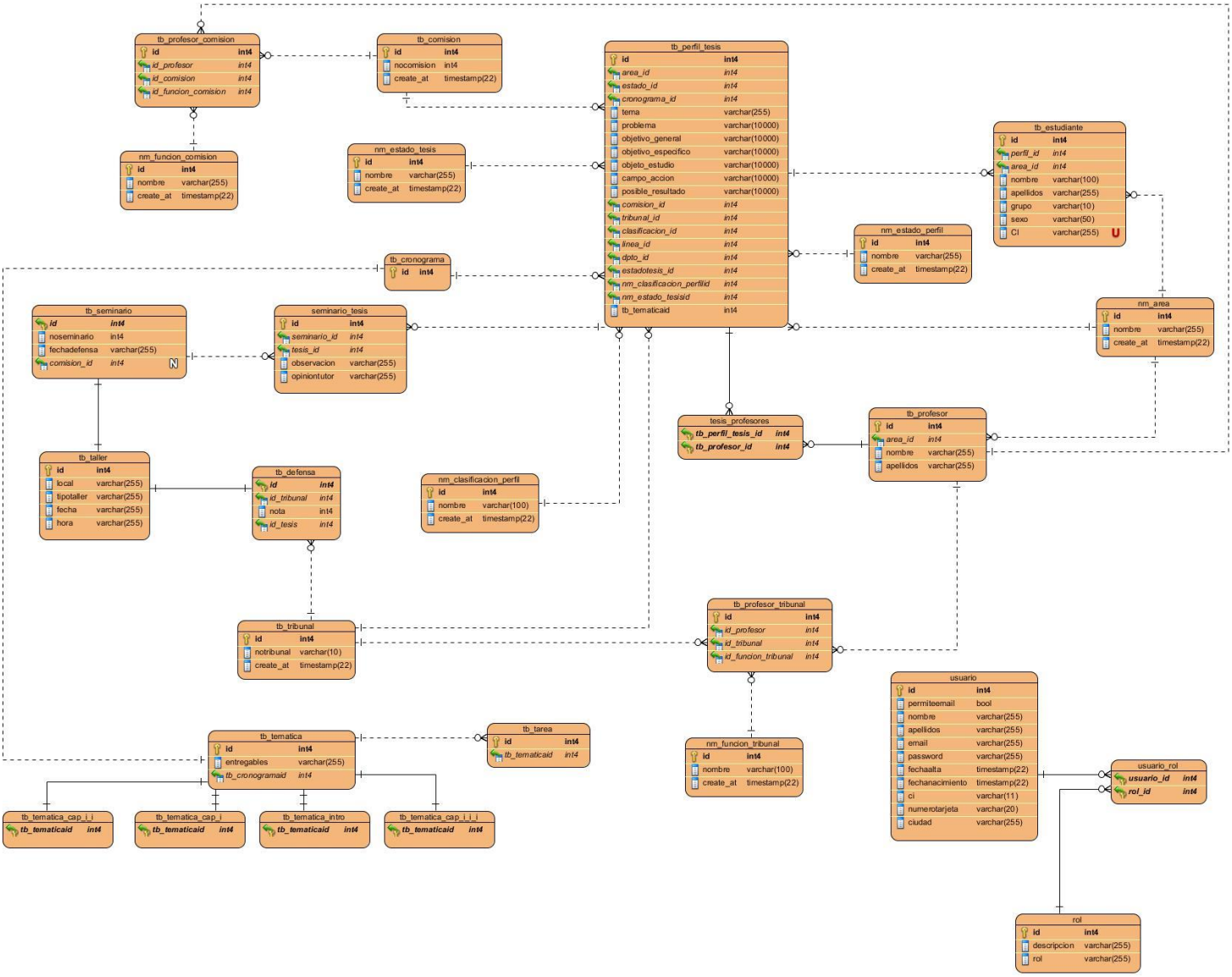

Figura 5. Modelo de datos del sistema de trabajo de diploma.

2.8 Estándar de codificación empleado

Con el objetivo de lograr una estandarización en la programación del sistema se decide aplicar el estándar de codificación CamelCase, el cual facilitará la lectura, comprensión y mantenimiento del código. A continuación se describe a grandes rasgos las convenciones de nomenclatura.

Convenciones de nomenclatura

General

- Se exceptúan el uso de las tildes y la letra ñ, la que será sustituida por nn.

- En todo momento se utilizarán nombres que sean claros, concretos y libres de ambigüedades. Ejemplo: "idSeminario" y no solamente "id".
- El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras, todas las palabras internas que lo componen comienzan con mayúscula. Ejemplo: “campoAccion”

Identación

- El contenido siempre se identará con *tb* para las tablas y *nm* para los nomencladores, nunca utilizando espacios en blanco.

Clases

- El nombre de las clases comenzará con minúsculas si este está compuesto por varias palabras, todas las palabras internas que lo componen comienzan con mayúscula. Ejemplo: ‘’tbPerfilTesis’’.
- Intentar mantener los nombres de las clases descriptivos y simples. Usar palabras completas, evitar acrónimos y abreviaturas, a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML.

Nombre de variables

- No se utilizarán nombres de variables que puedan ser ambiguos.
- Se procurará evitar dar nombres sin sentido a variables temporales. Por ejemplo: temp, i, tmp.
- Las variables booleanas deben tener nombres que sugieran respuestas o contenidos de tipo Sí/No, por ejemplo: “esPredefensa”
- Los nombres de las variables booleanas deben ser positivos, por ejemplo:” si/true”

2.9 Conclusiones parciales

El sistema diseñado garantiza y facilita la gestión del proceso de los trabajos de diplomas. La arquitectura permite la ampliación de las funcionalidades del sistema. Las aplicaciones de los patrones de diseño garantizan lograr un esquema del mismo de fácil entendimiento, extensión y mantenimiento, además de permitir la reutilización de componentes en aplicaciones futuras. Por otra parte, el empleo del estándar de codificación facilita la lectura, comprensión y mantenimiento del código para los desarrolladores.

Capítulo 3: VALORACIÓN DE LA VIABILIDAD DE LA PROPUESTA DE SOLUCIÓN

En el presente capítulo se valora la viabilidad de la propuesta de solución mediante la validación de los requisitos y el diseño propuesto, además de la realización de pruebas funcionales, unitarias y de aceptación al sistema informático SIGETESIS.

3.1 Técnicas de validación de requisitos

Con el objetivo de ratificar que los requisitos del software obtenidos definen el sistema que el cliente desea se llevó a cabo un proceso de validación de los mismos, para el cual se emplearon las siguientes técnicas: (10)

Revisiones de los requisitos: Se realizaron revisiones a cada uno de los requisitos por parte del equipo de desarrollo y por el cliente. Las revisiones internas generaron no conformidades de tipo técnicas y ortográficas, las cuales fueron corregidas satisfactoriamente en tiempo. Con el cliente se desarrollaron 2 revisiones. En la primera, el mismo quedó satisfecho con el trabajo efectuado por el equipo de desarrollo, sin embargo, se añadieron detalles a algunos de los requisitos funcionales para su perfeccionamiento. En el segundo encuentro, se añadieron detalles a requisitos funcionales y se aprobaron finalmente los requisitos.

Diseño de prototipos web: Mediante los prototipos se le mostró al cliente un modelo ejecutable del sistema que le permitió tener una visión preliminar de cómo sería este y a través de la interacción con los mismos se comprobó si satisfacía sus necesidades, los mismos fueron aprobados por el cliente.

Generación de casos de prueba: Como parte del proceso de validación de los requisitos funcionales del sistema fueron diseñados casos de pruebas para cada historia de usuario.

3.2 Métricas aplicadas a los requisitos

Con el objetivo de medir la calidad de la especificación de los requisitos se aplicó una de las métricas Calidad de la especificación (CE). Para obtener cuán entendibles y precisos son los

requisitos, primeramente, se calcula el total de requisitos de la especificación como se muestra a continuación:

***Nr*:** El total de requisitos de especificación.

***Nf*:** Cantidad de requisitos funcionales.

***Nnf*:** Cantidad de requisitos no funcionales.

Como resultado de la sustitución de los valores, para el sistema se obtiene:

Nr = Nf + Nnf

Nr = 37 + 5

Nr = 42

Para determinar, finalmente, la Especificidad de los Requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación:

ER = *Nui / Nr*

Donde es el número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas. Mientras más cerca de 1 esté el valor de ER, menor será la ambigüedad.

Para el caso de los requisitos obtenidos para el portal solo uno produjo contradicción en las interpretaciones. Sustituyendo las variables se obtiene:

ER = *40 / 42*

ER = *0.95*

Arrojando un resultado final satisfactorio, indicando que el grado de ambigüedad de los requisitos es sumamente bajo (5%) ya que el 95% es entendible. Los requisitos ambiguos fueron modificados y validados para garantizar su correcta comprensión.

3.3 Validación del diseño

Las métricas de software son una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas, de manera que se puedan desarrollar los remedios y mejorar el proceso del software. (8)

Las métricas para evaluar la calidad del diseño del sistema y su relación con los atributos de calidad son las siguientes:

- **Tamaño operacional de clases (TOC)**

El tamaño operacional de las clases está dado por el número de métodos asignados una clase. Mediante el cual se calcula el nivel de Responsabilidad de los métodos, la Complejidad de implementación de los mismos y su Reutilización a fin de inspeccionar la efectividad del diseño, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado. (11)

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

Tabla 5. Atributos de calidad evaluados por la métrica TOC.

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

Tabla 6. Criterios de evaluación de la métrica TOC.

Resultados obtenidos de la aplicación de la métrica TOC

El siguiente gráfico refleja que la mayoría de las clases tienen de 1 a 5 procedimientos. Este resultado demuestra que el funcionamiento general del sistema está distribuido equitativamente

entre la mayoría de las clases, de esta forma se garantiza que, en caso de ocurrir algún cambio en el sistema de clases del componente, estaría menos comprometido el funcionamiento correcto del mismo.

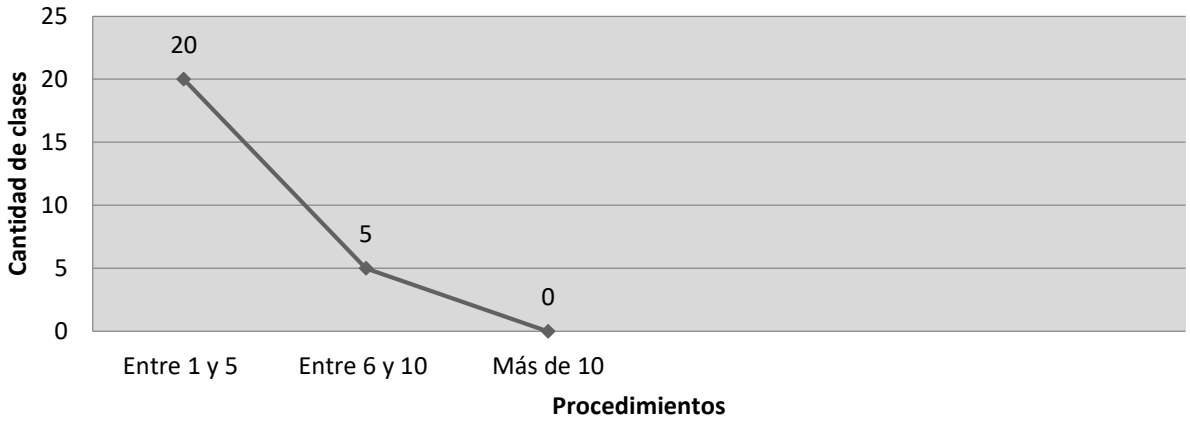

Figura 6. Resultados gráficos obtenidos de la aplicación de la métrica TOC.

La siguiente figura muestra la representación en % de los resultados obtenidos.

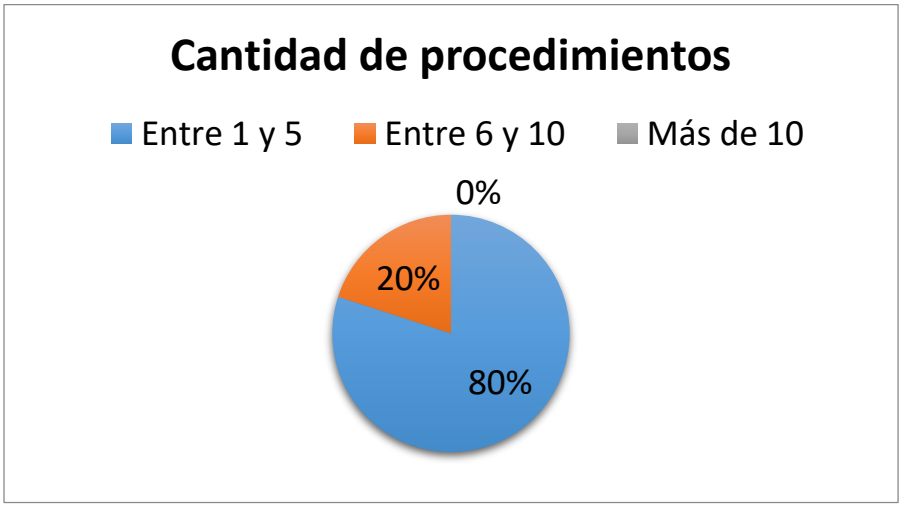

Figura 7. Resultados obtenidos de aplicar la métrica TOC en %.

A continuación, se muestra la representación de la incidencia de los resultados de la evaluación de la métrica TOC en los diferentes atributos.

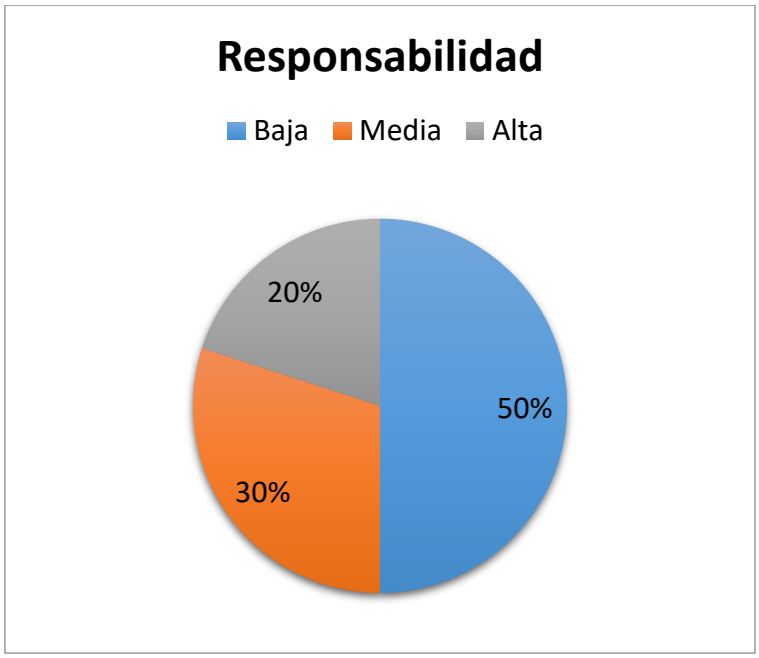

Figura 8. Atributo responsabilidad con la métrica TOC.

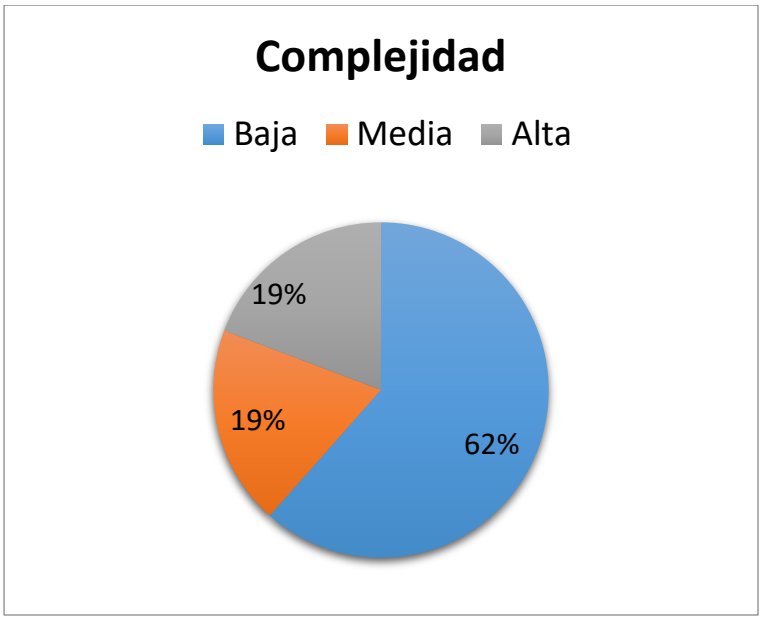

Figura 9. Atributo Complejidad con la métrica TOC.

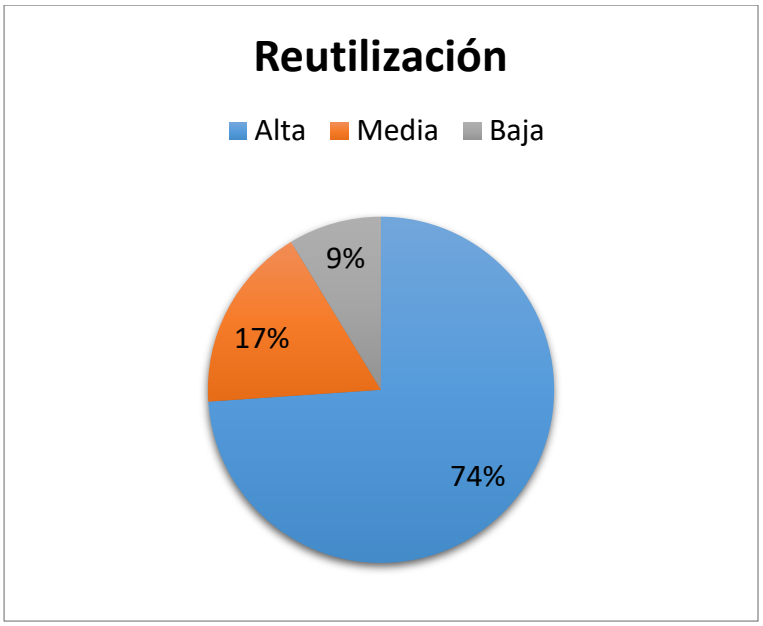

Figura 10. Atributo Reutilización con la métrica TOC.

Haciendo un análisis de los resultados obtenidos en la evaluación de la herramienta de medición de la métrica TOC, se puede concluir que el diseño del sistema tiene una calidad aceptable teniendo en cuenta que el 74 % de las clases incluidas en este sistema poseen una alta reutilización. Además, este mismo por ciento de clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad y Complejidad de Implementación) en el diseño propuesto, evidenciando que las clases no tienen tanta responsabilidad, no son tan complejas, y poseen un elevado grado de reutilización. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

- **Relaciones entre clases**

Las relaciones entre las clases están dadas por el número de relaciones de uso que tenga una clase con otras, o sea el número de dependencias que una clase tiene con otra. Mediante la cual se calcula el Acoplamiento, la Complejidad de mantenimiento, la Reutilización y la Cantidad de pruebas a fin de inspeccionar la efectividad del diseño, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado. (11)

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.

Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 7. Atributos de calidad evaluados por la métrica RC.

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

Tabla 8. Criterios de evaluación para la métrica RC.

Resultados obtenidos de la aplicación de la métrica RC

La gráfica muestra la representación de los resultados obtenidos agrupados en los intervalos definidos. Después de un análisis se decidió que no es posible eliminar estas dependencias por la responsabilidad arquitectónica que deben mantener las clases.

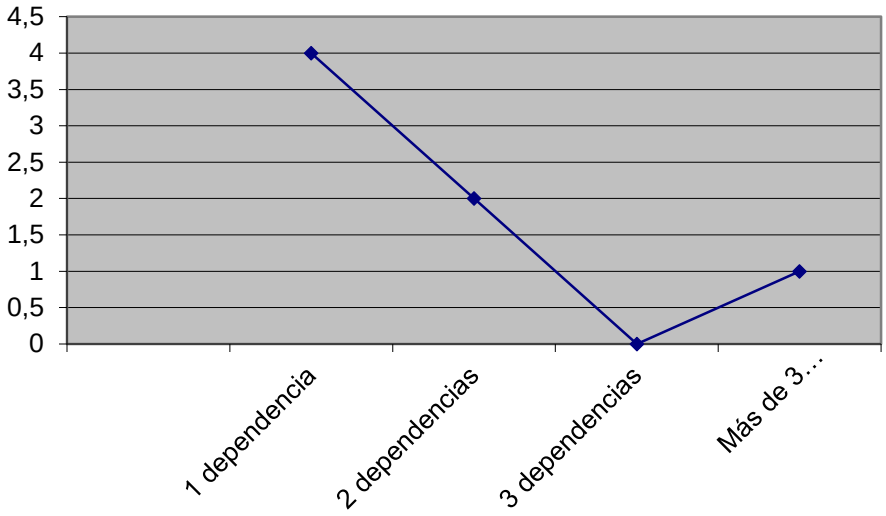

Figura 11. Representación de la evaluación de la métrica RC.

A continuación, se muestra la representación en % de los resultados obtenidos, agrupados en los intervalos definidos.

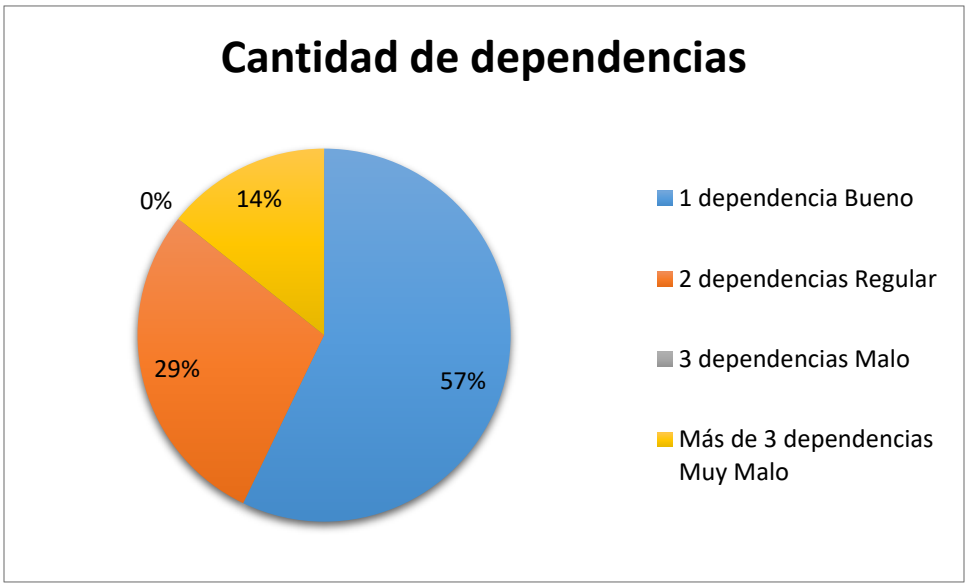

Figura 12. Representación en % de la evaluación de la métrica RC.

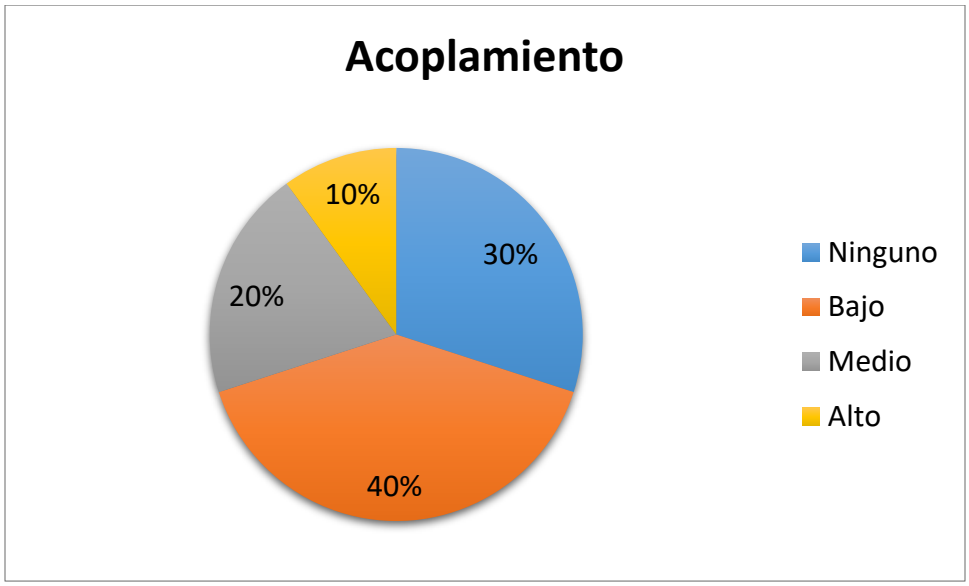

Figura 13. Atributo acoplamiento con la métrica RC.

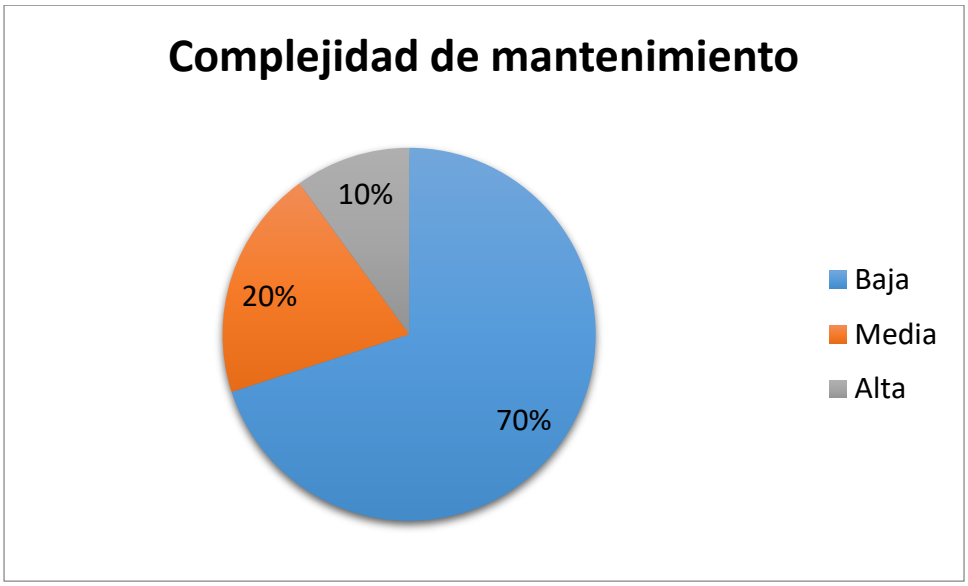

Figura 14. Atributo complejidad de mantenimiento con la métrica RC

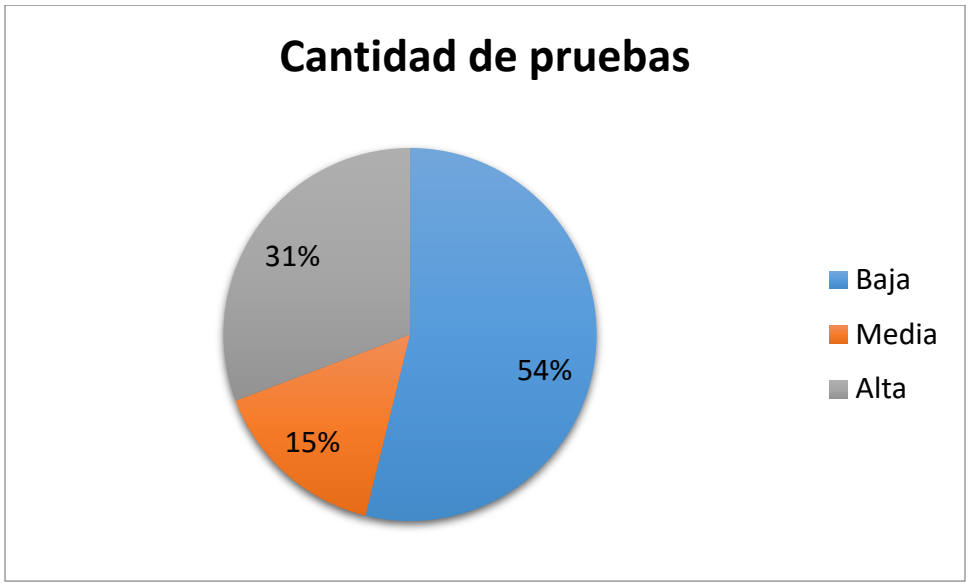

Figura 15. Representación de la evaluación de la métrica RC en el atributo Cantidad de pruebas.

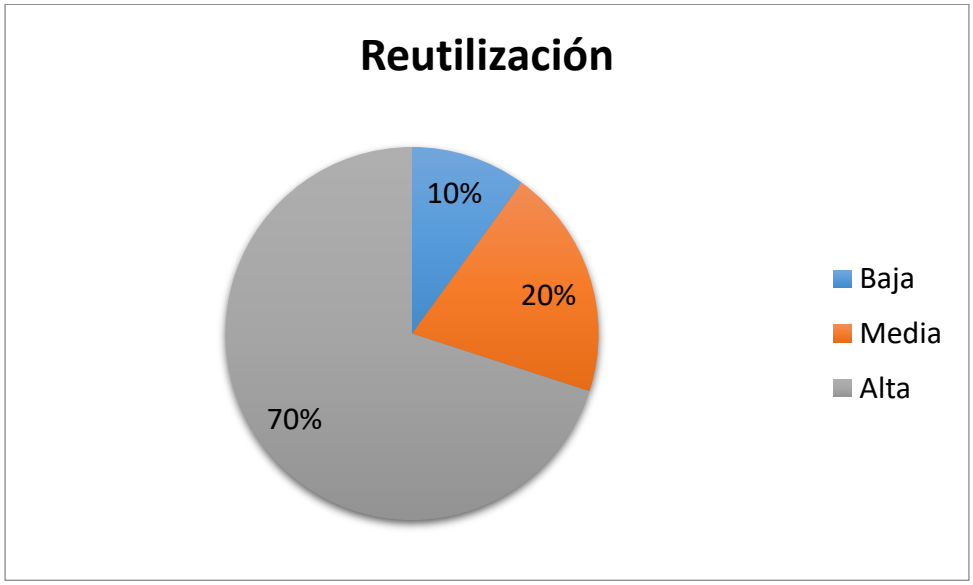

Figura 16. Representación de la evaluación de la métrica RC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación de la métrica RC, se puede concluir que el diseño del sistema está entre los niveles de calidad requeridos, pudiéndose observar que el 57% de las clases posee entre 0 y 3 dependencias. Igualmente, el 40% posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad

de mantenimiento, Reutilización y Cantidad de pruebas se comportan satisfactoriamente con un 54% y 60% de valor en las clases. Confirmando la elevada reutilización, el bajo acoplamiento, la baja complejidad y la baja cantidad de pruebas que se necesitan realizar.

3.4 Pruebas de software

Niveles de pruebas

Un nivel de prueba es un grupo de actividades que se organizan y administran juntas dentro de la ejecución de un proceso de pruebas que tiene como objetivo verificar y validar los componentes de un producto. Estos niveles permiten seleccionar diferentes tipos y técnicas de pruebas a realizar en cada nivel. (10)

- Prueba de unidad: Tienen como objetivo probar las unidades de software que componen el producto. Típicamente las pruebas de unidad se aplican a componentes de producto tales como: código fuente, archivos binarios, archivos de datos, entre otros, y tienen como objetivo verificar que los flujos de control, los flujos funcionales y los flujos de datos del elemento software son cubiertos, y que ellos funcionan como se espera.
- Prueba de integración: La prueba de integración es ejecutada para asegurar que los componentes software y hardware del producto operan adecuadamente cuando interactúan entre ellos para ejecutar un caso de uso (o transacción de negocio). Las pruebas de integración exponen la no compleción o los errores en las especificaciones de la interfaz de cada paquete software siendo integrado con los demás.
- Prueba de sistema: Tradicionalmente, las pruebas del sistema se realizan cuando el producto software está completado. El objetivo es evaluar si un producto software cumple con los requisitos que han sido especificados. Un ciclo de vida iterativo permite probar el sistema mucho más tempranamente tan pronto como los subconjuntos bien formados de requisitos funcionales se han construido.
- Prueba de aceptación: La prueba de aceptación de usuario es la prueba final antes de desplegar el software en los ambientes de operación. El objetivo de la prueba de aceptación es verificar que el software está listo, que puede ser utilizado, que satisface los criterios de aceptación, y que cubre aquellas necesidades y expectativas de los clientes para los cuales el software construyó. (10)

En diferentes libros se han propuestos varias estrategias de prueba de software, todas ellas proporcionan una plantilla, o sea brindan una guía al profesional y un conjunto de hitos al jefe de proyecto para la correcta realización de las pruebas. Estas deben incluir pruebas de bajo

nivel que verifiquen que se ha implementado correctamente todos los pequeños segmentos de código y pruebas de alto nivel que validen que las funcionalidades del sistema tengan correspondencia con lo especificado por el cliente. Para los efectos de la presente investigación se propone la siguiente estrategia de pruebas:

Tipo de Prueba	Método de prueba	Técnica de prueba
Unitarias	Caja blanca	Camino básico
Funcionalidad	Caja negra	Partición de equivalencia

Tabla 9. Estrategia de prueba aplicada a la solución.

3.4.1 Pruebas de caja blanca

La prueba de caja blanca, denominada a veces “prueba de caja de cristal” es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales; y ejerciten las estructuras internas de datos para asegurar su validez. (8)

Para comprobar que las funciones internas del sistema se ejecutan correctamente se le realizaron pruebas al código de las principales funcionalidades del sistema. Para ello se empleó la técnica de **la prueba del camino básico**, el cual consiste en garantizar que se pueda probar al menos una vez cada una de las sentencias del programa, partiendo de la obtención de la medida de la complejidad de un procedimiento o algoritmo y la obtención de un conjunto básico de caminos de ejecución de este, los cuales son utilizados para obtener los casos de prueba. (8)

Seguidamente se muestra el proceso de pruebas realizado a la funcionalidad newAction, específicamente a la controladora tbTallerSeminario por ser de mayor criticidad en el sistema. Primeramente, se comienza por analizar el código y enumerar las instrucciones:

```
public function newAction(Request $request)
{
    $tbTallerSeminario = new Tbtallerseminario();//1
    $form = $this->createForm('AppBundle\Form\tbTallerSeminarioType',
$tbTallerSeminario);//2
    $form->handleRequest($request);//3

    if ($form->isSubmitted() && $form->isValid()) {//4
        $em = $this->getDoctrine()->getManager();//5
        $em->persist($tbTallerSeminario);//6
        $em->flush();//7

        return $this->redirectToRoute('tbtallerseminario_show', array('id' =>
$tbTallerSeminario->getId()));//8
    }

    return $this->render('tbtallerseminario/new.html.twig', array(
        'tbTallerSeminario' => $tbTallerSeminario,
        'form' => $form->createView(),//9
    ));
}//10
```

Figura 17.Funcionalidad newAction (nuevo seminario de tesis).

Para obtener los casos de prueba a partir de la técnica seleccionada se debe construir seguidamente el grafo de flujo correspondiente al código de la función como se muestra en la Figura .

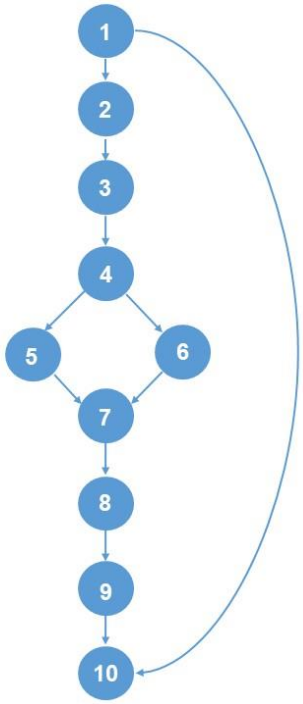

Figura 18. Grafo de flujo del código de la función newAction

Luego se determina la complejidad ciclomática V (G) del grafo resultante, la cual es un indicador del número de caminos independientes que existen en un grafo, es decir, es cualquier camino dentro del código que introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición. La complejidad ciclomática puede ser calculada de 3 formas:

1. V (G) = a - n + 2, siendo a el número de arcos o aristas del grafo y n el número de nodos.
2. V (G) = r, siendo r el número de regiones cerradas del grafo.
3. V (G) = c + 1, siendo c el número de nodos de condición.

Realizando los cálculos correspondientes se obtiene por cualquiera de las variantes el siguiente resultado:

1. V(G)= a – n + 2
   V(G)= 10 – 11 +2
   V(G)=3
2. V(G)= 3
3. V(G)= c + 1
   V(G)= 2 + 1
   V(G)= 3

Por lo que el conjunto de caminos básico sería:
Camino básico 1: 1-2-3-4-5-7-8-9-10
Camino básico 2: 1-2-3- 4-6-7-8-9-10
Camino básico 3: 1-10
Luego se definen los casos de prueba para cada uno de los caminos básicos obtenidos. A continuación se muestra el resultado de las pruebas aplicadas a los caminos básicos 1 y 2.

Descripción: se verificará que se cree un nuevo Seminario.		
Condición de ejecución:	**Entrada:**	**Resultados esperados:**
La variable request no debe ser vacía, para ello se debe haber seleccionado la opción Crear Seminario y haber llenado los campos correspondientes al formulario añadir Seminario.	$form (formulario llenado correctamente).	Que se inserte correctamente el nuevo Seminario añadido y muestra los datos específicos del mismo.

Resultado obtenido: Satisfactorio.

Tabla 10. Caso de prueba del camino básico No.1.

Descripción: se verificará que no se cree un nuevo Seminario.		
Condición de ejecución:	**Entrada:**	**Resultados esperados:**
La variable request no debe ser vacía, para ello se debe haber seleccionado la opción Crear Seminario y haber llenado los campos correspondientes al formulario añadir Seminario.	$form (formulario llenado incorrecto)	Que no se inserte en la base de datos y muestre un mensaje con el error cometido: ´ Campo obligatorio vacío o incorrecto ´ .
Resultado obtenido: Satisfactorio.		

Tabla 11. Caso de prueba del camino básico No.2.

Se logró el cálculo de la complejidad ciclomática para los algoritmos de la aplicación. La misma visualiza la forma que se emplea para obtener dicho cálculo con las estructuras especificadas. Los resultados obtenidos en el cálculo de la complejidad ciclomática definen el número de caminos independientes dentro de un fragmento de código y la cantidad de casos de pruebas diseñados.

3.4.1 Pruebas de caja negra

Las pruebas de caja negra, también denominada “prueba de comportamiento”, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los detectados por los métodos de caja blanca. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación. (10)

Para desarrollar las pruebas de caja negra existen varias técnicas, entre ellas se encuentran:

- **Técnica de la Partición de Equivalencia:** esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Técnica del Análisis de Valores Límites:** esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Técnica de Grafos de Causa-Efecto:** es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Dentro del método de Caja Negra la técnica de la **Partición de Equivalencia** es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores. (8)

Con el objetivo de comprobar que las funcionalidades del sistema se realizan de forma correcta y responden a las necesidades del cliente se realizaron un total de 3 iteraciones de pruebas de caja negra, obteniéndose los siguientes resultados:

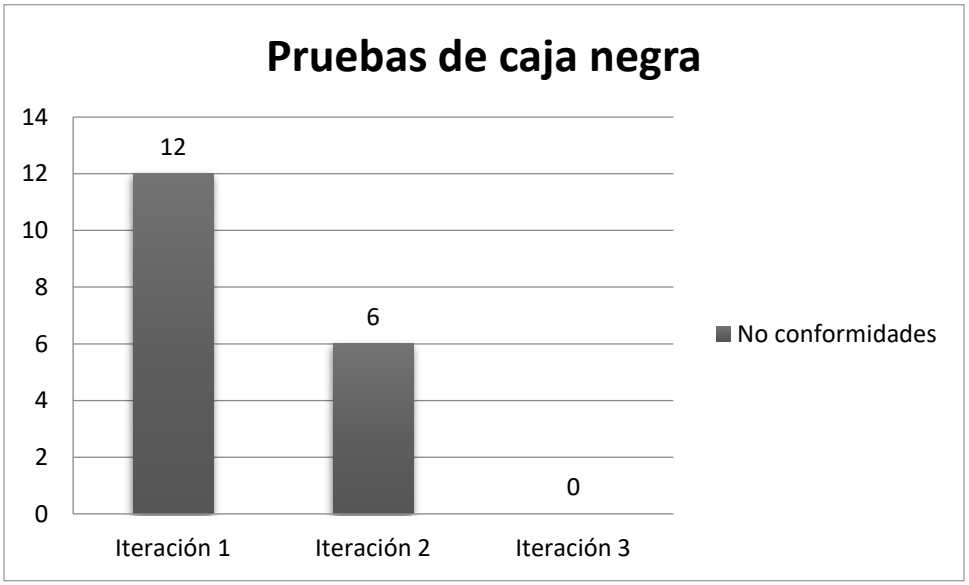

Figura 19. Cantidad de no conformidades detectadas en las pruebas de Caja Negra.

En una primera iteración se detectaron un total de 12 no conformidades (NC), 2 NC errores de validación, 3 NC funcional, 2 NC de correspondencia entre la aplicación y el documento, 1 NC de errores ortográficos y 1 NC de funcionalidad. En la segunda iteración se obtuvieron 6 no

conformidades donde aún existían errores de validación y se detectaron además problemas de escritura. Finalmente, en una tercera iteración se obtuvieron resultados satisfactorios al no detectarse no conformidades.

2.5 Pruebas de aceptación

Las pruebas de aceptación son especificadas por el cliente, se centran en las características y funcionalidades generales del sistema que son visibles. Estas pruebas derivan de las HU que se han implementado como parte de la liberación del software. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección y toma de decisiones acerca de estas pruebas. (10)

Con el objetivo de comprobar la aceptación del cliente con la solución desarrollada se realizaron un total de 2 iteraciones, obteniéndose los siguientes resultados:

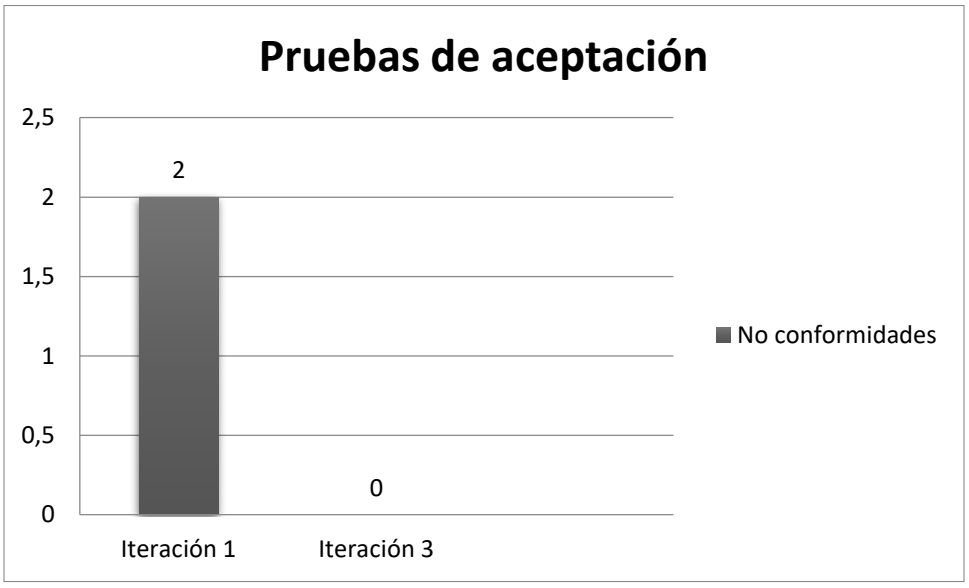

Figura 20. Cantidad de no conformidades detectadas en las pruebas de aceptación.

En una primera iteración se detectaron un total de 2 no conformidades (NC) de errores ortográficos. En la segunda iteración se obtuvieron resultados satisfactorios al no detectarse no conformidades. Finalmente, el cliente emitió un acta de aceptación (Ver Anexo 3).

2.6 Conclusiones parciales

La aplicación de técnicas de validación de requisitos permitió ratificar que los requisitos obtenidos estaban en correspondencia con las solicitudes del cliente. Además el uso de la métrica CE proporcionó una medida cuantitativa de la calidad de la especificación de estos, proveyendo las ambigüedades existentes para su corrección. Por su parte, la validación del diseño mediante las métricas TOC y RC permitió obtener, de forma general, el grado de complejidad de implementación y mantenimiento, de responsabilidad, reutilización, acoplamiento y la cantidad de pruebas necesarias para realizar a las clases favoreciendo la creación de un diseño lo más sencillo posible, de fácil mantenimiento e implementación y que promoviera la reutilización. Además, las pruebas de Caja Negra permitieron comprobar que las funciones son operativas a través de la interfaz del software, que la entrada se acepta de forma adecuada y se produce un resultado correcto, manteniendo así la integridad de la información externa y las pruebas de Caja Blanca sirvieron para comprobar internamente las funciones del portal, facilitando la detección de no conformidades para su corrección.

Conclusiones generales

Con el desarrollo del presente trabajo de diploma se obtuvo un sistema para la gestión del proceso de trabajos de diploma en la Facultad 3, para lo cual:

- El análisis de los sistemas que gestionan los procesos de tesis, permitió identificar que estos no pueden ser utilizados como propuesta de solución en el contexto de la Facultad 3, ya que no cumplen con las demandas del cliente y el dominio del problema.

- La selección de AUP variación UCI como metodología para guiar el desarrollo de la propuesta de solución, permitió la generación de los artefactos ingenieriles: historias de usuario, patrón arquitectónico MVC, diagrama de clases y modelo de datos, utilizados estos como referencias para la implementación.
- La utilización de las herramientas y tecnologías: lenguaje de modelado UML 2.0 soportado por la herramienta CASE Visual Paradigm 15.1, lenguaje de programación PHP 7.0, NetBeans 8.2 como entorno de desarrollo y PostgreSQL como gestor de base de datos; permitió establecer la infraestructura tecnológica para el desarrollo de la propuesta de solución.
- La implementación del sistema para la gestión del proceso de trabajos de diploma en la Facultad 3, permite la estructuración, consistencia y persistencia de la información que se gestiona en el proceso de gestión de tesis.
- Las pruebas realizadas y la aplicación de la técnicas de validación propiciaron determinar la aceptación por parte del cliente, siendo de gran valor para la calidad del sistema desarrollado, contribuyendo a la estructuración, consistencia y persistencia de la información en el proceso de tesis.

RECOMENDACIONES

Para dar continuidad a la presente investigación se recomienda:

- Continuar el estudio con el objetivo de añadir nuevas funcionalidades al sistema.
- Valorar la incorporación de mecanismos de inteligencia artificial para facilitar el procesamiento de la información y la toma de decisiones.
- Valorar el desarrollo de mecanismos de comunicación con el Sistema de Gestión Universitaria (Akademos).

REFERENCIAS

1. BATISTA, Yamilka González. Estrategia pedagógica para el desarrollo de la motivación profesional pedagógica en los estudiantes de los Centros de Educación Superior. En: Revista de Innovación Tecnológica. CIGET. Las Tunas. [línea]. 31 marzo 2012, Vol. 18, no. 1. [Consultado 2 mayo 2019]. Disponible en: http://innovacion.ciget.lastunas.cu/index.php/innovacion/article/view/10.

2. MES-UCI. PLAN DE ESTUDIOS “D” INGENIERÍA EN CIENCIAS INFORMÁTICAS. 2014. S.l.: s.n.

3. MES. Reglamento para el trabajo docente-metodológico de la Educación Superior. Resolución 2 del 2018. 2018. S.l.: La Habana.

4. SERVICIO DE INFORMÁTICA: ÁREA DE GESTIÓN. Manual del Sistema de Gestión Académica TESEO. mayo 2004. S.l.: s.n. Revisión 1.0

5. GIL GARCÍA, PILAR, GONZÁLEZ RODRÍGUEZ, SAGRARIO y HERNÁNDEZ RABILERO, LUISA Mª. La gestión documental de las tesis doctorales en la Universidad de Castilla-La Mancha. Texto de la comunicación presentada en las III Jornadas Andaluzas de Documentación JADOC03. En: Publicada en Organizaciones electrónicas. Situación actual y perspectivas de la documentación. 2003, pp. 36-41.

6. MACHADO, Ing. Reydel León y Pupo, Ing. Oscar Gabriel Reyes. Módulo para el control de la baja estudiantil en el SIGENU. En: Ciencias Holguín [línea]. 11 julio 2011, Vol. 17, no. 2. [Consultado 23 noviembre 2018]. Disponible en: http://www.ciencias.holguin.cu/index.php/cienciasholguin/article/view/591.

7. DÍAZ PÉREZ, Zuzel, GONZÁLEZ CÁRDENAS, Adony y SILVA BARRERA, David. Sistema Gestor de Trabajos de Diploma. En: [línea]. 2009, [Consultado 15 noviembre 2018]. Disponible en: http://repositorio.uci.cu/jspui/handle/ident/TD_2779_09.

8. ROGER S. PRESSMAN. Ingeniería de Software. Un enfoque práctico. 7ma. México: s.n., 2010. 978-607-15-0314-5.

9. TAMARA RODRÍGUEZ SÁNCHEZ. PROGRAMA DE MEJORA. Metodología de desarrollo para la Actividad productiva de la UCI. 2005. S.l.: Universidad de las Ciencias Informáticas.

10. IAN SOMERVILLE. Ingeniería de Software [línea]. 9na edición. México: Pearson, 2011. ISBN 978-607-32-0606-7. Disponible en: artemisa.unicauca.edu.co/~cardila/Libro_Somerville_9.pdf.

11. Mejora de Procesos de Software. En: [línea]. [Consultado 2 mayo 2019]. Disponible en: http://mejoras.prod.uci.cu/.

12. What is Symfony. En: [línea]. [Consultado 15 noviembre 2018]. Disponible en: https://symfony.com/what-is-symfony. Symfony is a set of PHP Components, a Web Application framework, a Philosophy, and a Community; all working together in harmony

13. Home - Twig - The flexible, fast, and secure PHP template engine. En: [línea]. [Consultado 15 noviembre 2018]. Disponible en: https://twig.symfony.com/.

14. Doctrine, The Open-Source PHP ORM and Persistence Tools Project. En: [línea]. [Consultado 15 noviembre 2018]. Disponible en: https://www.doctrine-project.org/index.html.

15. Bootstrap · The most popular HTML, CSS, and JS library in the world. En: [línea]. [Consultado 15 noviembre 2018]. Disponible en: https://getbootstrap.com/.

16. CRAIG LARMAN. UML y Patrones [línea]. 2da edición. México: Pearson, 2010. ISBN 978-84-205-3438-1. Disponible en: http://fmonje.com/UTN/ADES%20-%20208/UML%20y%20Patrones%20%202da%20Edicion.pdf .

17. BPMN Diagram and Tools. En: [línea]. [Consultado 15 noviembre 2018]. Disponible en: https://www.visual-paradigm.com/features/bpmn-diagram-and-tools/.

18. PHP: Manual de PHP - Manual. En: [línea]. [Consultado 22 mayo 2019]. Disponible en: https://www.php.net/manual/es/index.php.

19. JavaScript.com. En: [línea]. [Consultado 22 mayo 2019]. Disponible en: https://www.javascript.com/.

20. jQuery. En: [línea]. [Consultado 22 mayo 2019]. Disponible en: http://jquery.com/.

21. NetBeans IDE 8.2 Release Information. En: [línea]. [Consultado 15 noviembre 2018]. Disponible en: https://netbeans.org/community/releases/82/.

22. PostgreSQL: Documentation: 9.5: What is PostgreSQL? En: [línea]. [Consultado 15 noviembre 2018]. Disponible en: https://www.postgresql.org/docs/9.5/intro-whatis.html.

23. pgAdmin - PostgreSQL Tools. En: [línea]. [Consultado 22 mayo 2019]. Disponible en: https://www.pgadmin.org/.

24. Compilar e Instalar - Servidor HTTP Apache Versión 2.4. En: [línea]. [Consultado 12 febrero 2019]. Disponible en: http://httpd.apache.org/docs/2.4/es/install.html.

25. MSC. YOAN MARTÍNEZ MÁRQUEZ. Manual Docente-Meteorológico para el Vicedecanato de Formación. 2018. S.l.: s.n.