Facultad 2

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

Título: Sistema informático para la conformación automática de tribunales de tesis en la Facultad 2.

Autora:

Suany Leyva Hernández

Tutores:

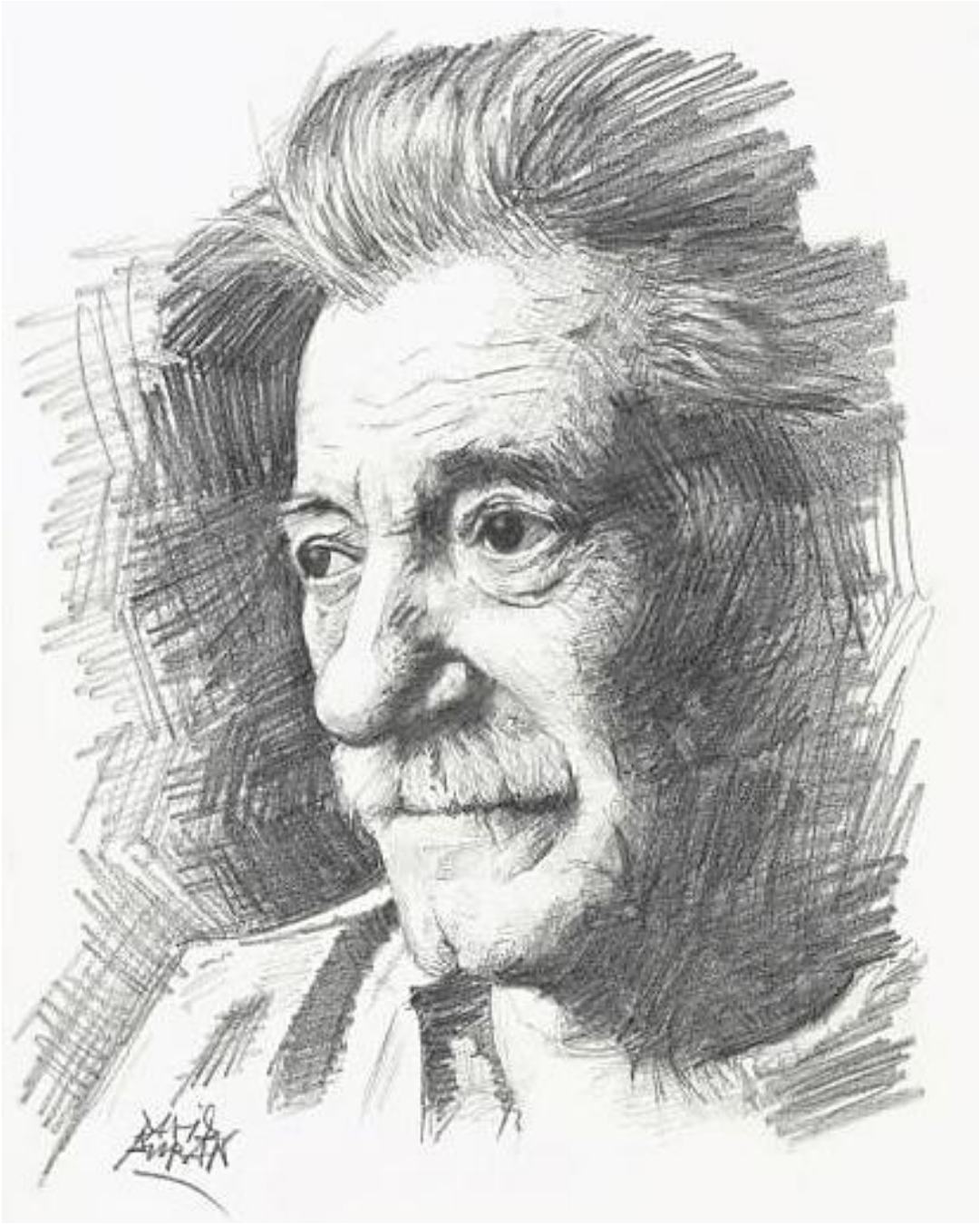
Ing. Bárbara Triana Morales

MSc. Darián Horacio Grass Boada

Co-tutor:

Dr. C. Jorge Sergio Menéndez Pérez

Junio 2013



Cinco minutos bastan para soñar toda una vida,

así de relativo es el tiempo.

Mario Benedetti.

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Suany Leyva Hernández

_____

Ing. Bárbara Triana Morales

_____

MSc. Darián Horacio Grass Boada

_____

Dr. C. Jorge Sergio Menéndez Pérez

_____

Datos de Contacto

Bárbara Triana Morales:
Correo Electrónico: btriana@uci.cu
Profesor graduado de Ingeniero en Ciencias Informáticas en 2008.
Profesor de ISW hasta el curso 2010-2011.
Profesor de Ética Informática, Debate Histórico Contemporáneo y Control de Versiones en el curso 2011-2012.
Gerente General de los Proyectos con Teltronic desde el 2009.
Ha participado en varios eventos nacionales e internacionales.

Darian Horacio Grass Boada
Correo Electrónico: dgrass@uci.cu
Experiencia como docente: 8 años
Categoría docente: Profesor Auxiliar
Título Académico: Máster en Ciencias de la Computación.
Profesor de Estructura de Datos y Algoritmos (Pregrado UCI)
Profesor de Inteligencia Artificial (Pregrado UCI)

Jorge Sergio Menéndez Pérez
Correo Electrónico: smenendez@uci.cu
Experiencia como docente: 20 años
Categoría docente: Profesor Auxiliar
Título Académico: Máster en Ciencias Pedagógicas Militares.
Grado Científico: Doctor en Ciencias Pedagógicas.
Profesor de Seguridad Nacional y Defensa Nacional (Pregrado UCI)
Profesor de Debates Históricos Contemporáneos (Pregrado UCI)
Profesor en la III edición del curso Talento Científico (Postgrado UCI)
Profesor de Formación Pedagógica (Postgrado UCI)
Jefe de la asignatura Seguridad Nacional de la disciplina Preparación para la defensa desde el 1 de marzo de 2011.

AGRADECIMIENTOS

Muchas son las personas que a lo largo de estos cinco años han colaborado con lo que ayer era una meta y hoy se ha convertido en realidad. La vida alejó a algunos, aproximó a otros, e hizo permanente la presencia de los mejores.

Mi mamá ha sido mi mejor amiga, consejera, abogada defensora y jueza imparcial; es el mejor ejemplo de sacrificio, de tenacidad, de fuerza espiritual y de amor incondicional. Mi agradecimiento mayor es para ella, por haberme dado tanto amor, una educación que rinde sus frutos con cada paso que doy, por haberme apoyado siempre e impulsado a lograr metas altas, a luchar por lo que vale la pena, y a ser lo mejor posible en mi actuar para con los demás.

Mi papá Mario siempre ha sido mucho más que un padre, es mi amigo, mi confidente, mi reparador de sueños, para él también muchas gracias por estar ahí, en nuestras vidas, dándonos todo su amor.

A mi papá Dubi, es un privilegio ser tu hija mayor; desde que tengo memoria estás en mi vida y, por ende, en mi corazón. Gracias por tu amor, por tus anécdotas tan ocurrentes, y por reafirmarme cuán importante es tener las ideas claras, el corazón caliente, y el cuerpo listo para aventuras, porque la vida es solo una, y merece ser vivida intensamente.

A mis abuelos, que son los mejores, y los únicos, por quererme tanto, y mimarme, y contribuir a mi formación desde las más disímiles maneras.

A mi novio, que me ha enseñado a ver la vida desde distintas perspectivas, que me ha hecho mejor persona, que es una fuente segura de amor, de comprensión y de conocimiento. Gracias por tu amor y apoyo, sobre todo en este período tan complejo para ambos, por tenerme tanta paciencia, por ayudarme para lograr este sueño, por creer en mí cuando yo no podía, por todo, simplemente gracias.

A mis amigos, por preocuparse, por estar siempre presentes, aunque la distancia mediara físicamente, en todo momento sentí su apoyo y cariño.

A mis tres tutores, porque nuestra relación ha ido más allá de la formalidad. A Baby, que me escogió para desarrollar este tema, y ha estado apoyándome y defendiéndome desde entonces, que ha sido mi amiga, mi confidente, gracias por la dedicación, por el esmero, por el cariño con siempre me ha tratado. A Darián, que además de guiarme en todos los aspectos técnicos del trabajo de diploma, siempre tuvo para mí consejos oportunos, me transmitió seguridad, me contagió con su alegría pero también con sus ansias de conocimiento. A Menéndez, fuente inagotable de sabiduría, pedagogo excepcional, dedicado y exigente, gracias por formar parte de mi equipo. Logramos formar un engranaje del cual ustedes son piezas claves, gracias de todo corazón.

A todas las personas que han contribuido con la realización de esta tesis, a Javier, por su incondicionalidad, y por ser tan noble y buen amigo. A Alejandro García, por sacar tiempo para dedicármelo, a pesar de estar igual de complicado. Al profesor Amaury, que accedió, sin conocerme, a brindarme su ayuda.

Gracias a todos.

A mi mamá, por ser mi inspiración.

A mis dos papás, porque con ellos mi vida se ilumina.

A mi novio, por ser el complemento de mi vida.

A mi familia y amigos, por tanta confianza y apoyo.

A mis tutores, porque su presencia se palpa en cada página de este trabajo.

RESUMEN

La defensa del trabajo de diploma es la forma de evaluación de la culminación de estudios en la UCI. Parte del éxito de esta actividad, depende de la conformación de los tribunales que se encargan de validar la calidad de este ejercicio.

Por lo general, el proceso de identificación de los miembros y conformación de tribunales de tesis se realiza manualmente, con toda la carga de implicaciones negativas que esta forma de proceder puede introducir (complejidad, excesivo consumo de tiempo al efectuar la tarea, generación involuntaria de errores).

El objetivo que se persigue con el presente trabajo es: Desarrollar un sistema informático, que permita conformar automáticamente los tribunales de tesis en la Facultad 2, cumpliendo con un conjunto de condiciones mínimas exigidas. Para lograrlo, se hace uso de técnicas de inteligencia artificial, específicamente, de la metaheurística GRASP.

Se implementó una aplicación de escritorio que, automáticamente, genera una propuesta de tribunales de tesis, identificando las figuras de Presidente, Secretario, Vocal y los Oponentes de las tesis asignadas a cada uno, atendiendo al nivel de experticia en el tema, Categoría Docente y Categoría Científica de los profesores.

Para la implementación de la solución se utilizaron los lenguajes de programación Java y Prolog; los entornos de desarrollo Swi Prolog y Netbeans respectivamente, y como metodología de desarrollo RUP, con UML como lenguaje de modelado y la notación de modelado de procesos del negocio BPMN.

PALABRAS CLAVE: *Conformación automática, metaheurística, GRASP, inteligencia artificial, Prolog.*

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La formación de los profesionales de nivel superior es el *proceso que, de modo consciente y sobre bases científicas, se desarrolla en las instituciones de educación superior para garantizar la preparación integral de los estudiantes universitarios, que se c*oncreta en una sólida formación científico técnica, humanística y de altos valores ideológicos, políticos, éticos y estéticos, con el fin de lograr profesionales revolucionarios, cultos, competentes, independientes y creadores, para que puedan desempeñarse exitosamente en los diversos sectores de la economía y de la sociedad en general. (1)(sic)

En la Universidad de las Ciencias Informáticas (UCI), se materializa este proceso con características específicas. De acuerdo a lo establecido en los Reglamentos vigentes, la evaluación de la culminación de los estudios comprueba los objetivos generales del plan de estudio. Los tipos de evaluación que pueden utilizarse son: El examen estatal y la defensa del trabajo de diploma. (1)

En el Reglamento de Trabajo docente y metodológico se plantea que: *la defensa del trabajo de diploma es 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.* (1)

Como se puede apreciar, el ejercicio de culminación de estudios es la actividad final del Proceso Docente Educativo (PDE) en el sistema de preparación de pregrado de la UCI, en el que la defensa del trabajo de diploma, juega un papel muy importante, no solo en el orden instructivo, sino también educativo. Parte del éxito de esta actividad, depende de la conformación de los tribunales que se encargan de validar la calidad de este ejercicio.

En las Facultades que conforman la estructura universitaria, se asignan los temas de tesis que han sido previamente definidos por miembros del claustro. Atendiendo a que en el PDE de la UCI, se establece que los estudiantes deben ser concebidos como sujetos activos de su propio proceso de formación, estos últimos pueden proponer un tema de tesis, que debe ser evaluado en cuanto a su alcance y relación con las líneas de investigación establecidas por la Facultad. Con posterioridad se redacta el perfil de tesis, que debe ser aprobado para poder comenzar el proceso de elaboración del trabajo de diploma. De ser aprobado, se le asigna un tutor al estudiante, en caso de que no se haya propuesto en el momento de redacción del perfil de tesis.

En el caso concreto de la Facultad 2, la Vicedecana de Formación conforma los tribunales de tesis que aprobarán los referidos perfiles confeccionados. Dichos tribunales, son los que evaluarán a los estudiantes en los actos de pre-defensa y defensa de tesis.

La designación de los miembros de los tribunales de tesis no se realiza teniendo en cuenta la clasificación de temas de tesis que pertenecerán a un tribunal, ni a la experiencia de los profesores que lo conforman con respecto a los temas asignados.

En no pocas ocasiones existen tribunales en los que no hay presencia de ningún profesor que ostente alguna categoría docente principal superior a la de Instructor, y en otras, se aprecia la ausencia de docentes con títulos académicos y grados científicos. Al mismo tiempo, es justo destacar que esta situación es totalmente diferente en la conformación de otros tribunales, en los que sí se aprecia determinado nivel en lo que a categorías docentes, títulos académicos y grados científicos se refiere. Dicho en otras palabras, existen problemas de balance respecto a esta estructuración de "status" académico y de categoría docente y científica en dichos tribunales.

En el momento de asignar los oponentes de tesis, no se tiene en cuenta la carga en el desempeño profesional de los profesores miembros de los tribunales, ni la experiencia, ni el marco referencial de estos a la hora de enfrentar determinados temas de tesis. En ocasiones, ocurre que en un tribunal, uno de sus miembros puede coincidir con el rol de oponente o tutor de una de las tesis asignadas a ese mismo tribunal, generándose una situación compleja en el desempeño de las funciones de estos profesionales. También se da el caso de presidentes de tribunales de tesis que pudieran ostentar algún grado científico, o título académico, o una categoría docente principal adecuada a la situación evaluativa que enfrenta, sin embargo, puede que no sean especialistas en el tema que defiende el estudiante.

Las condiciones mínimas definidas que debe tener un tribunal de tesis son: El presidente debe tener el mayor conocimiento del tema de la tesis. Mientras tanto, el oponente debe ser otro conocedor del tema, garantizándose con ello la calidad y exigencia que debe acompañar a este "acto evaluativo" de culminación de estudios. En un tribunal no deben coincidir los oponentes y tutores con alguno de sus miembros.

Otro aspecto al que se le debe prestar la máxima atención por parte de los Directivos del Proceso Docente Educativo, es al hecho de que el proceso de identificación y conformación de los miembros de los tribunales de tesis se realiza manualmente, con toda la carga de implicaciones negativas que esta forma de proceder puede introducir (complejidad, excesivo consumo de tiempo al efectuar la tarea, generación involuntaria de errores e imprecisiones).

Atendiendo a todo lo anteriormente referenciado se ha identificado el siguiente **problema a resolver**: ¿Cómo garantizar que el proceso de conformación de los tribunales de tesis en la Facultad 2 cumpla con un conjunto de condiciones mínimas exigidas?

El **objetivo general** que se propone alcanzar en el presente trabajo es: Desarrollar un sistema informático que permita conformar automáticamente los tribunales de tesis en la Facultad 2, cumpliendo con un conjunto de condiciones mínimas exigidas.

Se define como **objeto de estudio:** Los sistemas informáticos para la asignación de recursos.

Atendiendo al objeto declarado se ha definido en calidad de **campo de acción:** La conformación automática de tribunales de defensa de trabajos de diploma.

En calidad de **objetivos específicos** se han definido los siguientes:

- Diseñar el algoritmo a utilizar, aplicando técnicas de inteligencia artificial, para lograr la conformación automática de tribunales.
- Diseñar un sistema informático que facilite el proceso de dirección en cuanto a la planificación, organización, gestión de los tribunales de tesis en la Facultad 2.
- Implementar el sistema informático para la conformación automática de tribunales de tesis.

En calidad de **métodos empleados**, se pueden citar los siguientes:

Como métodos teóricos: Analítico-Sintético; Inductivo-Deductivo y el Análisis Documental, sustentados en el estudio de disímiles fuentes de información y en el procesamiento de los fundamentos científicos y de las apreciaciones de numerosos autores y colectivos de autores.

Como método empírico: Observación Científica, que permitió constatar la situación problemática existente, obtener resultados concretos en virtud de la aplicación de otros métodos así como validar el correcto funcionamiento del sistema propuesto.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas de investigación**:

- Formalización del estado del arte de los sistemas de distribución automática de recursos.
- Selección de las herramientas a utilizar para el desarrollo del sistema.
- Modelación del problema de optimización combinatoria.
- Selección de técnicas de inteligencia artificial a utilizar en el desarrollo del sistema.
- Captura de los requisitos funcionales y no funcionales para identificar las características que tendrá el sistema.

- Identificación de los elementos del diseño y la implementación para diseñar e implementar la solución.
- Conformación de los manuales de usuarios necesarios.
- Realización y documentación del proceso de pruebas para lograr que el sistema quede libre de errores.

El trabajo se encuentra estructurado en cuatro capítulos, los cuales se especifican a continuación:

Capítulo 1 Fundamentación Teórica: Se realiza un estudio relacionado con los principales conceptos y características del proceso de conformación de tribunales de tesis. Además se realiza un análisis de las principales herramientas utilizadas con dicho fin. Se explican las herramientas y metodologías seleccionadas para realizar la implementación del sistema.

Capítulo 2 Análisis de la solución: Se describen las características de la solución propuesta, definidas a partir de la modelación de los procesos de negocio y la especificación de los requisitos de software.

Capítulo 3 Diseño del Sistema: Se especifican los elementos del diseño de la arquitectura del sistema y los patrones de diseño utilizados. Además se describen los elementos del modelo de optimización y del algoritmo utilizado para la solución.

Capítulo 4 Implementación y Prueba: Se detalla el diagrama de despliegue y las pruebas realizadas al sistema.

1. CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DEL SISTEMA INFORMÁTICO PARA LA CONFORMACIÓN AUTOMÁTICA DE TRIBUNALES DE TESIS EN LA FACULTAD 2.

En el presente capítulo se expone el sistema categorial empleado; además se realiza un análisis de las soluciones existentes a nivel internacional y nacional que puedan ser semejantes a la solución informática a desarrollar, relacionada con la asignación automática de recursos y, finalmente, se describen las características fundamentales de las herramientas y tecnologías de software utilizadas para dar cumplimiento a la solución del problema.

1.1. Sistema categorial empleado

Proceso Docente Educativo

Carlos A. de Zayas, en su libro, “Hacia una escuela de excelencia”, se refiere al término **Proceso Docente Educativo** (PDE) y declara que, “el proceso mediante el cual se forma sistemáticamente a las generaciones de un país le llamaremos proceso docente -educativo o proceso de enseñanza - aprendizaje.” (2)

Álvarez de Zayas considera al Proceso Docente Educativo como el objeto de estudio de la Didáctica, el cual describe ampliamente cuando dice que es “aquel proceso que como resultado de las relaciones sociales que se dan entre los sujetos que participan, está dirigido, de un modo sistémico y eficiente, a la formación de las nuevas generaciones, tanto en el plano educativo como instructivo (objetivo), con vista a la solución del problema social: encargo social, mediante la apropiación de la cultura que ha acopiado la humanidad en su desarrollo (contenido); a través de la participación activa y consciente de los estudiantes (método); planificada en el tiempo y observando ciertas estructuras organizativas estudiantiles (forma); y con ayuda de ciertos objetos (medio); y cuyo movimiento está determinado por las relaciones causales entre esos componentes y de ellos con la sociedad (leyes), que constituye su esencia.” (3)

En otras palabras, el PDE puede ser comprendido como el proceso multilateral e institucionalizado de influencias intencionadas, sistemáticas y dirigidas de los educadores en los educandos, que se manifiesta como un fenómeno de relaciones interpersonales y cuyo modo de existencia es la actividad y la comunicación. Cumple tres funciones principales: cognoscitiva, educativa e integradora.

Trabajo de Diploma

Es el tipo de trabajo investigativo de los estudiantes que les permite adquirir un mayor dominio y actualización de los métodos científicos y técnicas característicos de la profesión.

La defensa del trabajo de diploma es 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. (1)

Tribunal

Una de las acepciones del concepto de Tribunal es entendida como el "Conjunto de jueces ante el cual se efectúan ciertas pruebas que han de controlar y evaluar". (4)

El trabajo de diploma es evaluado por un tribunal que para emitir la calificación tendrá en cuenta los siguientes elementos:

- Calidad del trabajo (uso de la metodología de la investigación científica, actualización científico técnica, uso de las estrategias curriculares de acuerdo con el contenido del trabajo, entre otros).
- Capacidad creadora, originalidad e independencia en el desarrollo del trabajo.
- Calidad de la exposición durante la defensa, respuestas a las preguntas y dominio del tema.
- Opiniones del profesor designado para la orientación científica del estudiante, del oponente y de la entidad laboral para la cual se realizó el trabajo. (1)

Para lograr evaluar los elementos mencionados anteriormente, el tribunal debe estar conformado de forma integral; esto se logra con la presencia de habilidades técnicas, y metodológicas distribuidas entre sus miembros.

1.1.1 Problema de Optimización

Los problemas de optimización permiten seleccionar el mejor elemento (con respecto a algún criterio) de un conjunto de elementos disponibles. Tienen como objetivo maximizar o minimizar una función real eligiendo valores de entrada y computando el valor de la función descubriendo los mejores valores de una función objetivo. (5)

Un problema de optimización se puede describir matemáticamente como:

Dado un dominio X y una función objetivo

$f(x): x \in X \rightarrow \mathbb{R}$

El objetivo es encontrar x' que verifique (en caso de maximizar)

$x' \in X: f(x') \geq f(x), \forall\, x \in X$

Atendiendo al dominio de las variables, los problemas de optimización pueden dividirse en dos categorías: de Optimización Continua y Optimización Discreta. En el caso específico del problema a resolver en la investigación, este debe tratarse como de Optimización Discreta o Combinatoria, pues la naturaleza de sus variables así lo exige.

Un problema de optimización combinatoria tiene como objetivo encontrar el máximo (o el mínimo) de una determinada función sobre un conjunto finito de soluciones (CS). Cabe destacar que dada la finitud de CS, las variables han de ser discretas, restringiendo su dominio a una serie finita de valores. Habitualmente, el número de elementos de CS es muy elevado, haciendo impracticable la evaluación de todas sus soluciones para determinar el óptimo. (6)

La mayoría de los problemas de Optimización Combinatoria pueden ser modelados como problemas de programación lineal entera. La programación entera trata los problemas de maximizar o minimizar una función de diversas variables sujeta a condiciones de igualdad y/o desigualdad, restringiéndose todas o alguna de esas variables a tomar valores enteros. En general, se usa en todas aquellas áreas donde se trata de resolver el problema de asignar recursos de cualquier tipo, solo disponibles en cantidades discretas. (7)

Nótese como el argumento esgrimido anteriormente se relaciona con la temática de esta investigación, consistente en un problema de asignación de recursos, en este caso, de recursos humanos, en interés de un Proceso Docente Educativo de excelencia.

Los problemas de optimización combinatoria son una rama de la optimización en matemáticas aplicadas y en ciencias de la computación, relacionada con la inteligencia artificial. La Inteligencia Artificial implica el uso de métodos basados en el comportamiento inteligente de los seres humanos y de otros animales, para resolver problemas complejos. (8)

Algunos ejemplos de problemas de optimización combinatoria son:

- El viajante de comercio: Un viajante de comercio ha de visitar n ciudades, comenzando y finalizando en su propia ciudad. Conociendo el coste de ir de cada ciudad a otra, determinar el recorrido de coste mínimo.
- La Mochila: Dados n elementos e_1, e_2,..., en con pesos p_1, p_2,..., p_n y beneficios b_1, b_2,..., b_n, y dada una mochila capaz de albergar hasta un máximo de peso M (capacidad de la mochila), queremos encontrar las proporciones de los n elementos x_1, x_2,..., x_n ($0 \leq x_i \leq 1$) que tenemos que introducir en la mochila de forma que la suma de los beneficios de los elementos escogidos sea máxima.

- Coloreado de Grafos: Dado un grafo conexo y un número m > 0, llamamos colorear el grafo a asignar un número i, (1 ≤ i ≤ m) a cada vértice, de forma que dos vértices adyacentes nunca tengan asignados números iguales.
- Asignación de Recursos (Tareas): Supongamos que disponemos de n trabajadores y n tareas. Sea $b_{ij} > 0$ el coste de asignarle el trabajo *j* al trabajador *i*. Una asignación de tareas puede ser expresada como una asignación de los valores 0 o 1 a las variables x_{ij}, donde $x_{ij} = 0$ significa que al trabajador *i* no le han asignado la tarea *j*, y $x_{ij} = 1$ indica que sí. Una asignación válida es aquella en la que a cada trabajador solo le corresponde una tarea y cada tarea está asignada a un trabajador. Dada una asignación válida, definimos el coste de dicha asignación como:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} b_{ij}$$

  Diremos que una asignación es óptima si es de mínimo coste.

Para la conformación de un tribunal de tesis, se deben asignar un profesor para cada rol, de ahí que el problema a resolver sea similar al de asignación de recursos enunciado anteriormente, en este caso serían recursos humanos.

Para resolver los problemas de optimización combinatoria es posible el empleo de métodos exactos y de métodos aproximados. Si el problema de optimización tiene un espacio de soluciones muy grande, el coste de ejecución de los métodos exactos puede aumentar de forma exponencial, convirtiendo la resolución en prácticamente inviable. Como consecuencia, se puede hallar una solución aproximada mediante el uso de técnicas de inteligencia artificial, como son las heurísticas y las metaheurísticas.

Las técnicas heurísticas son procedimientos para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución. (9)

Las técnicas Metaheurísticas “son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos”. (10)

Conociendo entonces que las técnicas heurísticas son más específicas, o sea, dependen del problema que se desea resolver, mientras que las metaheurísticas son más generales y flexibles, se decide optar

por el uso de estas últimas para darle solución al problema de optimización combinatoria, cuyo modelado representa el problema a resolver de la investigación.

1.1.2. Técnicas Metaheurísticas

La Figura 1 muestra una clasificación de las metaheurísticas

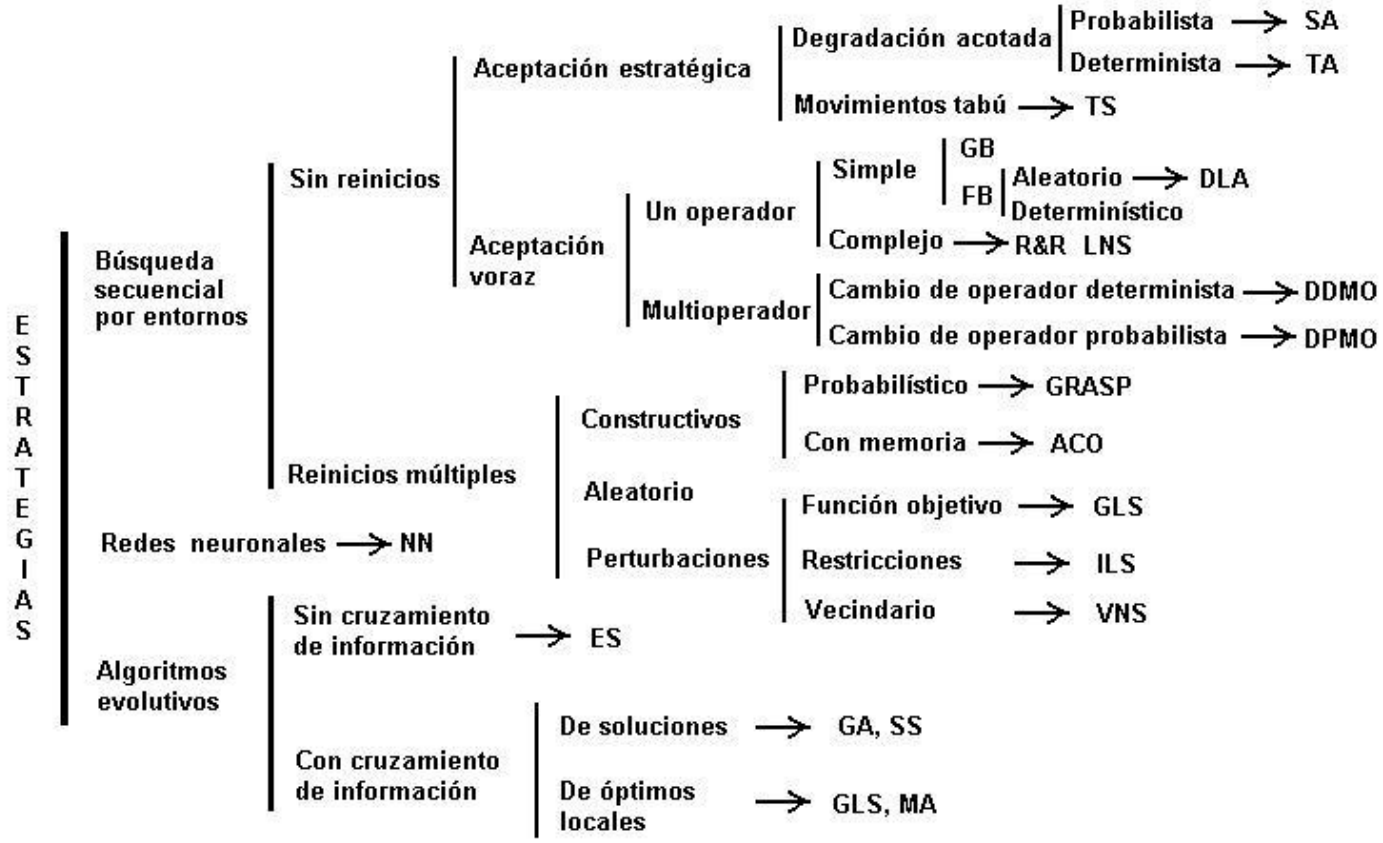


Figura 1. Taxonomía de las Metaheurísticas.

Las metaheurísticas de búsqueda guían los procedimientos que usan transformaciones o movimientos para recorrer el espacio de soluciones alternativas y explotar las estructuras de entornos asociadas. Dentro de esta clasificación se encuentran las metaheurísticas constructivas, las cuales se orientan a los procedimientos que tratan la obtención de una solución a partir del análisis y selección paulatina de las componentes que la forman.

Las metaheurísticas evolutivas están enfocadas a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones.

La conformación de tribunales se proyecta como un procedimiento por etapas, cumpliendo en cada una con un conjunto de restricciones, por lo que la metaheurística a utilizar debe tener un enfoque constructivo. Además, la conformación de tribunales está sujeta a un conocimiento local, en el cual un tribunal específico debe cumplir con ciertas condiciones, y a la vez está asociada con un conocimiento global que implica que debe existir cierto equilibrio entre la calidad de los tribunales, por lo que resulta necesario un proceso de búsqueda de posibles mejoras.

De las técnicas metaheurísticas existentes, se decidió seleccionar la denominada GRASP (Greedy Randomized Adaptative Search Procedures), atendiendo a que es una metaheurística multiarranque para optimización combinatoria, que consta de dos fases: construcción y mejora. En la fase de construcción, se aplica un procedimiento heurístico constructivo para obtener la solución inicial. Esta solución se mejora en la segunda fase mediante un algoritmo de búsqueda local. La mejor de todas las soluciones examinadas se guarda como resultado final.

1.2. Análisis de las soluciones existentes

Evidentemente, la esencia de la problemática planteada en el diseño de esta investigación gira en torno a un problema de asignación de recursos, con una cantidad nada despreciable de variables dependientes imposibles de ser tenidas en cuenta en su totalidad por los directivos de un proceso tan complejo como lo es el Proceso Docente Educativo.

Al realizar una búsqueda en distintas fuentes bibliográficas, como son el sitio de la Biblioteca UCI, Internet, y en la Red Universitaria del Ministerio de Educación Superior, de aplicaciones que asignaran recursos automáticamente, se encontraron una serie de soluciones que se enfocan en problemas parecidos, pero que no llegan a tratar concretamente una variante de solución al problema que se ha planteado en este diseño de investigación.

1.2.1 Sistema para la conformación automática de tribunales para las pruebas de acceso a la Universidad:

Herramienta informática que determina de forma eficiente la composición de los tribunales que juzgan las Pruebas de Acceso a la Universidad. Está basado en una completa interfaz de entrada de datos, dado el numeroso conjunto de restricciones legales que afectan al problema, una modelización automática, basada en técnicas de programación lineal entera, mediante el lenguaje algebraico AMPL (Algebraic Modeling Programming Language) y una resolución mediante el optimizador IBM ILOG CPLEX, de la compañía IBM. (11)

No se pudo acceder a la herramienta, debido a que la información que se brinda de esta se encuentra en un resumen del Congreso Nacional de Estadística e Investigación Operativa del año 2001 en Úbeda, España.

1.2.2 Gdarim:

Sistema automático para asignación de aulas y distribución de espacios, desarrollado en la Universidad de Palermo en Argentina. Se basa en Algoritmos Genéticos (AG), e implementa un algoritmo Epsilon Multi Objective Evolutionary Algorithm (MOEA, por sus siglas en inglés) para optimizar más de un objetivo, específicamente dos, permitiendo que ambos compitan para establecer un equilibrio general y llegar a un conjunto de soluciones factibles de forma ágil y eficiente. Está desarrollado en Java, dada su portabilidad, eficiencia y por ser Open Source.

Se divide en tres módulos:

1. Módulo API (Application Programming Interface) de información, que contiene los valores del dominio del problema, además de administrar la configuración necesaria del algoritmo MOEA en cuanto a las restricciones, evaluaciones y ponderaciones específicas de cada objetivo del problema.
2. Módulo motor AG, que implementa el núcleo del algoritmo genético y la inteligencia de la solución. Incluye la administración de las poblaciones, el proceso de los operadores genéticos y la evaluación conjunta de los objetivos a optimizar.
3. Módulo API de asignación, que registra las posibles asignaciones y el espacio de solución que genera el algoritmo. (12)

Gedarim difiere de la solución que se desea desarrollar en esta investigación, atendiendo a que el algoritmo que utiliza, al ser multiobjetivo, privilegia la asignación de aulas y la distribución de espacios, mientras que el que se tiene en cuenta en la propuesta de solución que se realiza, será modelado para optimizar un objetivo único, muy específico, toda vez que lo que interesa es, única y exclusivamente, la *conformación de tribunales*.

1.2.3 AG Mantenimiento:

Desarrollado en la Universidad de La Sabana en Colombia, le da solución a un problema de asignación de horarios de mantenimiento a máquinas, equipos e instalaciones de la empresa Powel Continental Ltda. Se obtiene un cronograma de mantenimiento preventivo empleando un algoritmo genético, que optimiza el tiempo y el costo del personal encargado de estas labores, cumpliendo con los requerimientos de mantenimiento de cada máquina de la empresa. La solución está implementada en Visual Basic, y el

algoritmo genético propuesto arroja soluciones de buena calidad, requiriendo menos operaciones para lograrlo. (13)

Este sistema no resuelve el problema planteado debido a que las restricciones a las que está sujeto el problema de asignación de horarios de mantenimiento a máquinas, son distintas a las definidas en el diseño de esta investigación. Por tanto, el resultado es igualmente diferente, con AG Mantenimiento se asigna y distribuye el tiempo de mantenimiento de equipos, a diferencia del sistema que se desea implementar, que distribuirá recursos humanos para conformar tribunales.

Como se puede apreciar, las soluciones estudiadas no resuelven el problema planteado por lo que se propone realizar un sistema informático que permita realizar la conformación de los tribunales a partir de los profesores con que se cuenten.

1.3 Tecnologías, herramientas y metodología de desarrollo de software.

Para desarrollar la solución informática se utilizarán las siguientes tecnologías, herramientas y metodología de desarrollo de software.

1.3.1 Lenguaje de modelado **UML**

El Lenguaje de Modelado Unificado (Unified Modeling Language, UML por sus siglas en inglés) es un lenguaje visual para especificar, construir y documentar los artefactos de sistemas de software intensivo. UML fue aprobado por el Object Management Group (OMG, por sus siglas en inglés) como un estándar en 1997. Se puede utilizar con todos los procesos en todo el ciclo de vida de desarrollo y a través de diferentes tecnologías de implementación. Incluye aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación y esquemas de bases de datos. (14)

Se emplea UML como lenguaje para modelar la solución propuesta debido a que permite modelar sistemas utilizando técnicas orientadas a objetos (OO), además de poder especificar todas las decisiones de análisis, diseño, e implementación, construyendo así, modelos precisos, no ambiguos y completos.

UML puede conectarse con lenguajes de programación, dando la posibilidad de aplicar ingeniería directa e inversa.

1.3.2 Lenguajes de programación

1.3.2.1 Prolog:

Lenguaje de programación para ordenadores que se basa en el lenguaje de la Lógica de Primer Orden. Se utiliza para resolver problemas en los que entran en juego objetos y relaciones entre ellos. Una de las

ventajas de la programación lógica es que se especifica qué se tiene que hacer (programación declarativa), y no cómo se debe hacer (programación imperativa). Prolog tiene construcciones como son: variables lógicas, gestión de memoria como una pila, mecanismos de retardo utilizados para codificar la propagación de restricciones y el no determinismo. (15)

Se decide emplear este lenguaje de programación por ser declarativo, por contar con mecanismos muy útiles como el backtracking o vuelta atrás, el manejo de listas y estructuras de datos flexibles y la recursividad.

1.3.2.2 Java:

Lenguaje creado en 1991 por la compañía Sun Microsystems. Desde su salida al mercado en 1995, Java ha sido probado, mejorado y ampliado por una comunidad especializada. Cuenta con una comunidad de más de nueve millones de desarrolladores, siendo la más grande y activa del planeta. Permite:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma.
- Crear programas que pueden ejecutarse dentro de un navegador web y acceder a los servicios web disponibles.
- Combinar aplicaciones o servicios para crear aplicaciones altamente personalizadas. (16)

1.3.3 Entornos de Desarrollo Integrado:

En la propuesta que se defiende, se emplearán los siguientes IDE (Integrated Development Enviroment): SWI Prolog y Netbeans sobre los que a continuación se hace referencia.

1.3.3.1 SWI Prolog:

SWI-Prolog ofrece un entorno de Software Libre, distribuido bajo la Licencia GNU Pública Menor (LGPL, por sus siglas en inglés). Su desarrollo comenzó en 1987 y ha sido impulsado por las necesidades de aplicaciones del mundo real. En estos días SWI-Prolog es ampliamente utilizado en la investigación y la educación, así como para aplicaciones comerciales. Dentro de sus principales características se encuentran:

- Es un entorno de desarrollo completo, incluye gráficos, bibliotecas y requiere aproximadamente 40 MB.
- Incluye la biblioteca de interfaz Java Prolog Library (JPL por sus siglas en inglés).
- Es portable para muchas plataformas, incluyendo casi todas las Unix / Linux, Windows (NT/2000/XP), MacOS X. (17)

El análisis de las fuentes consultadas para la selección del IDE de Prolog a emplear, arrojó que existe solo una librería para la conexión Java-Prolog, y esta se encuentra únicamente en SWI Prolog. De ahí que se haya decidido utilizar este IDE entre tantos que tiene el lenguaje Prolog, debido a que es el único que incluye la librería JPL.

1.3.3.2 Netbeans:

Se utilizará como entorno de desarrollo Netbeans en su versión 7.2, pues es un reconocido IDE disponible para Windows, Mac, Linux y Solaris. Es open source y provee una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones de escritorio, web, empresariales y móviles utilizando múltiples plataformas entre la que se encuentra Java, PHP y Ajax. Es estable y hace uso de plugins para ampliar sus funcionalidades, siendo fácil de usar. (18)

1.3.4 Metodología de desarrollo **RUP**:

El Proceso Unificado de Rational, (RUP, por sus siglas en inglés) es el conjunto de procesos de ingeniería de software que dan guía para conducir las actividades de desarrollo del equipo. Como una plataforma de procesos que abarca todas las prácticas de la industria, RUP permite seleccionar fácilmente el conjunto de componentes de proceso que se ajustan a las necesidades específicas del proyecto. Está pensado para adaptarse a cualquier proceso y no solo al de software. (19)

En el caso del Sistema informático para la conformación automática de tribunales de tesis, se decidió utilizar RUP, debido a su adaptabilidad al proyecto que se está desarrollando, permitiendo generar la documentación necesaria para futuras versiones de la aplicación por parte de desarrolladores distintos a los actuales.

1.3.5 Sistema Gestor de Base de Datos **PostgreSQL 9.2.4.1**

Se decide emplear PostgreSQL debido a que es un potente sistema de bases de datos objeto-relacional de código abierto. Está distribuido bajo licencia BSD (Berkeley Software Distribution) y con su código fuente está disponible libremente. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada, que se ha ganado una sólida reputación de fiabilidad e integridad de datos. Funciona en los principales sistemas operativos, incluyendo Linux, UNIX, MacOS y Windows. Tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados en varios idiomas. Soporta almacenamiento de objetos binarios grandes, como imágenes, sonidos o vídeo. Cuenta con interfaces nativas de programación para Java, C / C + +, .NET, Perl, Python, entre otros, y una amplia documentación. (20)

1.3.6 Herramienta de Modelado **Visual Paradigm 8.0:**

Visual Paradigm for UML (VP-UML) es una herramienta de diseño UML y herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) diseñada para ayudar al desarrollo de software. Soporta los principales estándares de la industria tales como el Lenguaje de Modelado Unificado (UML), BPMN. Ofrece un completo conjunto de herramientas de equipos de desarrollo de software necesario para la captura de requisitos, planificación de software, la planificación de controles, el modelado de clases y modelado de datos. Además, permite dibujar todos los tipos de diagramas de clases, código inverso y generar código desde los diagramas. (21)

1.3.7 Notación de modelado de Procesos de negocio **BPMN:**

La notación de modelado de procesos de negocio empleada ha sido Business Process Modeling Notation (BPMN, por sus siglas en inglés). Proporciona la comprensión de los procesos de negocio en una notación gráfica, permitiendo la comunicación de estos procedimientos de manera estándar. Esta notación ha sido diseñada específicamente para coordinar la secuencia de procesos y los mensajes que fluyen entre los diferentes procesos participantes. (22)

Conclusiones del Capítulo

En el estudio realizado y cuyos resultados aparecen refrendados en el desarrollo temático del capítulo se realizó un análisis de los conceptos fundamentales y la formalización del estado del arte de los sistemas de distribución automática de recursos, evidenciándose la inexistencia de una variante concreta que le brinde solución al problema a resolver que ha sido planteado en el diseño de la investigación. Se decide elaborar un sistema informático, basado en un problema de optimización combinatoria, empleando la técnica metaheurística GRASP. Para el desarrollo de la solución, se utilizarán los lenguajes de programación Java y Prolog, los entornos de desarrollo SWI Prolog y Netbeans para cada lenguaje respectivamente. La metodología de desarrollo a emplear será RUP, con UML como lenguaje de modelado y la notación de modelado de procesos BPMN.

2. CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA INFORMÁTICO PARA LA CONFORMACIÓN AUTOMÁTICA DE TRIBUNALES DE TESIS EN LA FACULTAD 2.

En el presente capítulo se abordan los aspectos relacionados con el proceso del negocio y las características del sistema. Además, se enuncian los requerimientos funcionales y no funcionales de la aplicación; se describe la propuesta de solución, mostrando los diagramas de casos de uso del sistema, su priorización y la descripción de los mismos.

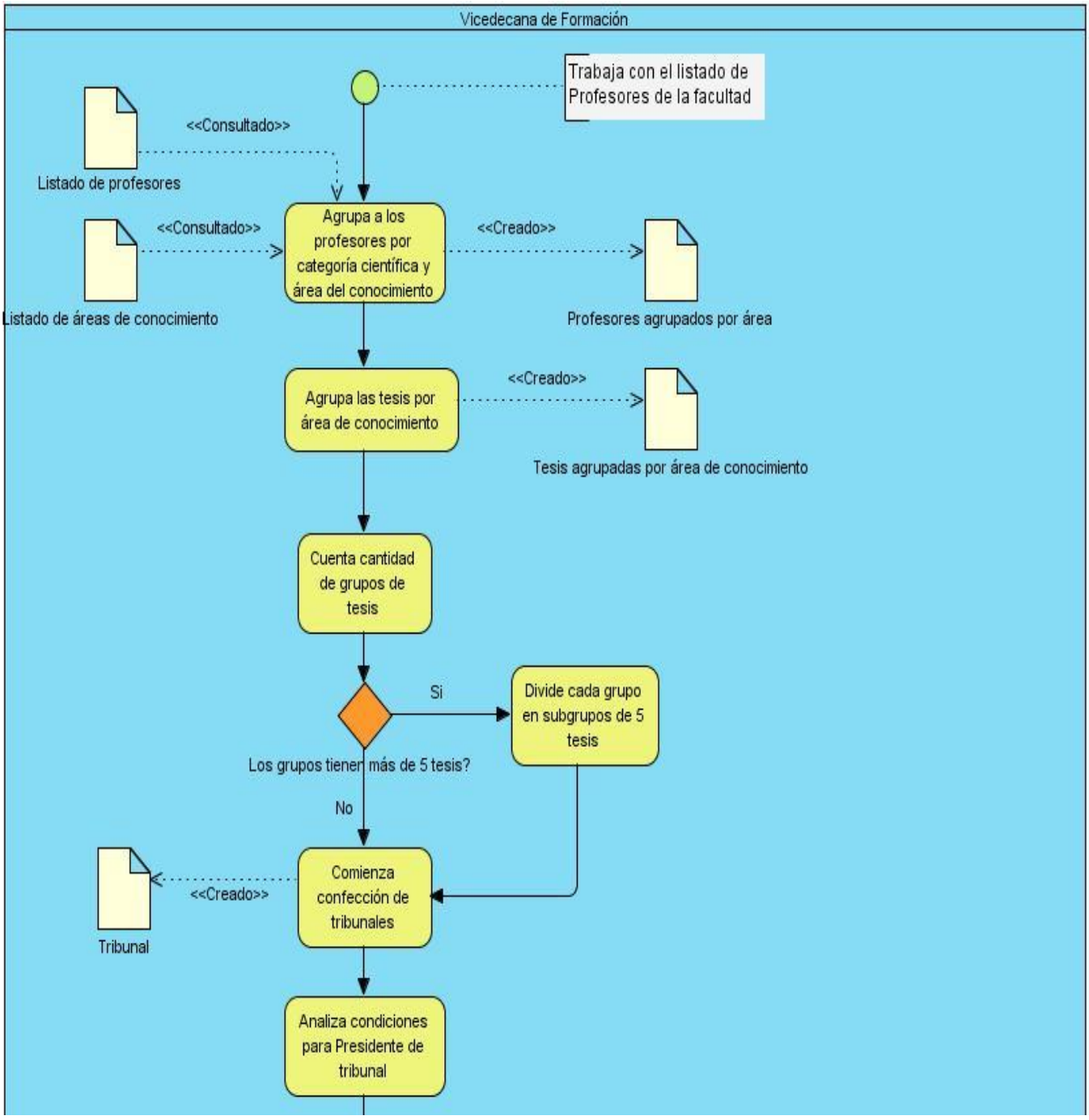
2.1. Modelo de Negocio

Con el modelado de negocio se logra comprender el entorno en que va a funcionar el sistema, identificar sus procesos, la información con la que se va a trabajar, los actores que participan en dichos procesos y los papeles que representan cada uno de ellos con respecto a la información. Como resultado de la identificación, captura y documentación de la información, se lleva a cabo la base para especificar los requisitos de un sistema.

Para modelar los procesos de negocio identificados se utilizará la Notación para el Modelado de Procesos de Negocios (BPMN, por sus siglas en inglés).

En la Figura 1 se aprecia el Diagrama de procesos del negocio que describe la conformación de tribunales de tesis en la Facultad 2.

Diagrama BPMN



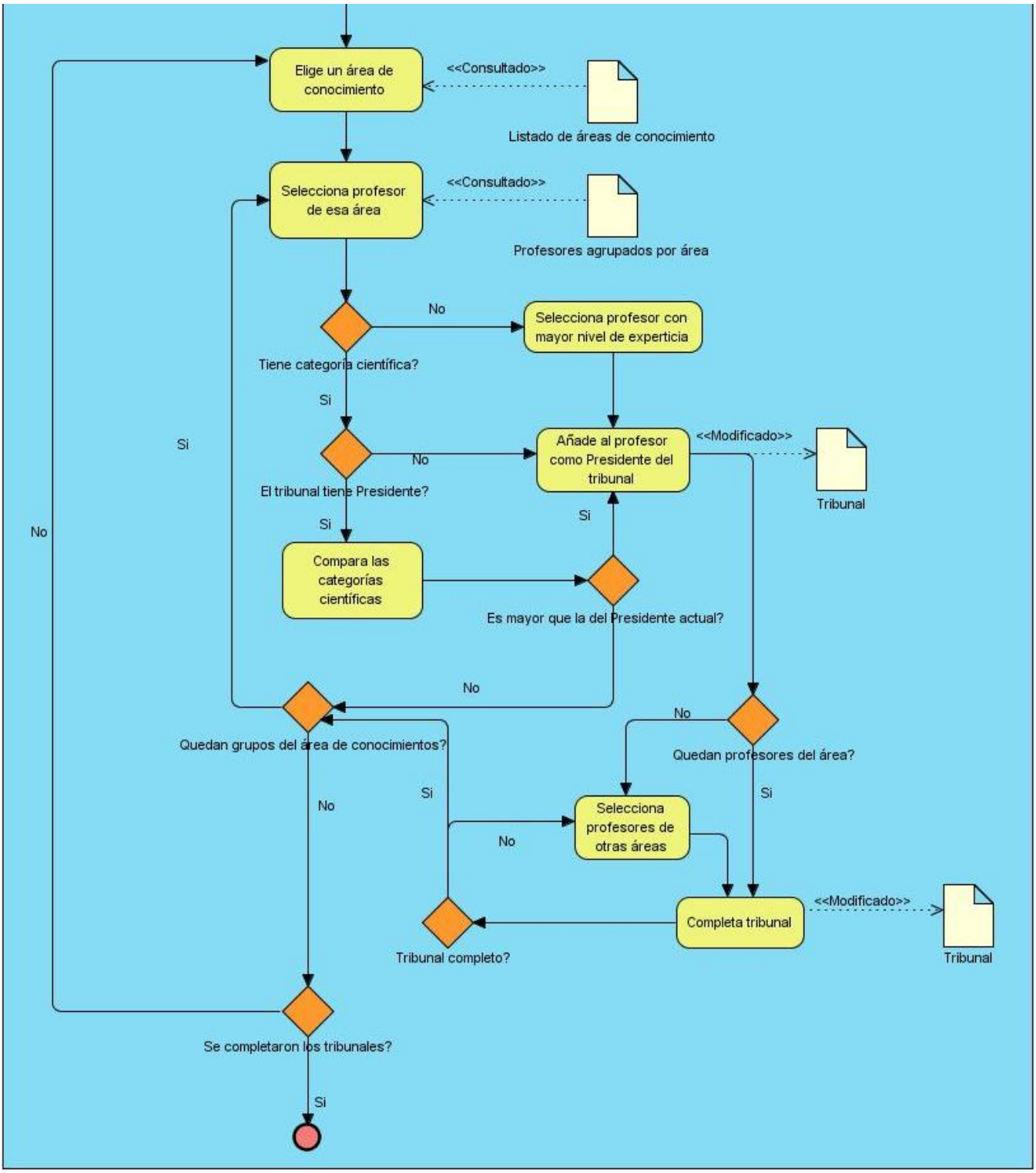


Figura 2. Modelo del negocio

Descripción del negocio

La asignación de profesores para los tribunales de tesis comienza cuando la Vicedecana de Formación agrupa a los profesores por Áreas de conocimiento ordenados por categoría científica. Con posterioridad, agrupa las tesis por área de conocimiento y las divide en grupos de cinco, para comenzar con la conformación de tribunales. Elige un área de conocimiento y selecciona un profesor para comprobar sus condiciones para Presidente de tribunal; si tiene categoría científica y no se ha asignado presidente a ese tribunal se asigna el profesor. Si ya tiene Presidente, se compara la categoría científica, si la del profesor seleccionado es mayor que la del Presidente, se intercambian los roles, si no, se verifica que existan más grupos de esa área y se comienza la asignación de Presidente nuevamente. Si quedan profesores del área escogida, se completan los tribunales. Si no se pueden completar, se añaden profesores de otras áreas. Si el área de conocimiento está completa, se comienza el proceso desde la selección de área. Finaliza cuando están constituidos todos los tribunales.

Personas que intervienen en el proceso de negocio

Tabla 1. Personas que intervienen en el proceso de negocio

Nombre	Justificación
Vicedecana de Formación	Es la encargada de realizar todo el proceso de gestión de tesis, profesores, y de conformar los tribunales de tesis.

2.2. Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Para la aplicación que se implementará se capturaron los siguientes requerimientos funcionales:

1. Autenticar usuario
2. Adicionar profesor
3. Eliminar profesor
4. Modificar profesor
5. Buscar profesor
6. Listar profesor
7. Adicionar tesis
8. Listar tesis

9. Eliminar tesis
10. Buscar tesis
11. Modificar tesis
12. Crear tribunales
13. Listar tribunales
14. Ver detalles de tribunal
15. Imprimir
16. Exportar a Excel
17. Exportar a XML
18. Gestionar Categoría científica
    - Adicionar Categoría científica
    - Eliminar Categoría científica
    - Modificar Categoría científica
19. Gestionar Categoría docente
    - Adicionar Categoría docente
    - Eliminar Categoría docente
    - Modificar Categoría docente
20. Gestionar Rol
    - Adicionar Rol
    - Eliminar Rol
    - Modificar Rol
21. Gestionar Centro productivo
    - Adicionar Centro productivo
    - Eliminar Centro productivo
    - Modificar Centro productivo
22. Gestionar Área de conocimiento
    - Adicionar Área de conocimiento
    - Eliminar Área de conocimiento
    - Modificar Área de conocimiento
23. Gestionar Nivel de experticia
    - Adicionar Nivel de experticia

- Eliminar Nivel de experticia
- Modificar Nivel de experticia

2.3. Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, que imponen restricciones en el diseño o la implementación del sistema. Para la aplicación que se desea desarrollar se identificaron los siguientes requerimientos no funcionales:

Hardware.

- Microprocesador Pentium 4 o superior.
- Memoria RAM ≥ 512 MB

Software

- Debe estar instalada en la computadora del cliente la Máquina Virtual de Java en su versión 1.6, y el IDE SWI Prolog versión 6.2.6.
- El servidor de base de datos PostgreSQL versión 9.2.4.1 debe estar instalado.

Soporte

- El sistema debe tener una documentación abundante.
- El sistema debe poseer un manual de usuario.

Usabilidad

- El sistema podrá ser utilizado por usuarios con conocimientos básicos de informática.

2.4. Propuesta de Solución

Como muestra la Figura 3, se implementará una aplicación de escritorio que de forma automática generará una propuesta de tribunales de tesis, identificando de cada uno el Presidente, Secretario, Vocal y los Oponentes de las tesis asignadas al mismo, atendiendo al nivel de experticia en el tema, Categoría Docente y Categoría Científica de los profesores. La aplicación además permitirá exportar a Excel e imprimir los tribunales conformados.

Figura 3. Propuesta de Solución

2.5. Descripción del sistema.

Para describir el sistema se han agrupado las funcionalidades en paquetes, conteniendo estos las funcionalidades que se encuentran en la misma área de interés.

Del paquete Acciones dependen Gestionar Profesores, Gestionar Tesis y Gestionar Nomencladores, porque en Acciones, está contenida la funcionalidad Autenticar usuario, de la que depende el acceso al resto de las funcionalidades del sistema. Por otra parte, el paquete Gestionar Tribunales depende de Gestionar Profesores y Gestionar Tesis, porque para generar un tribunal, es necesaria la información referente a las tesis y los profesores existentes.

El paquete **Acciones** contiene las funcionalidades: Autenticar usuario, Exportar a Hoja de cálculo, Exportar a Excel e Imprimir.

El paquete **Gestionar Profesores** contiene las funcionalidades: Adicionar profesor, Listar profesor, Buscar profesor, Eliminar profesor y Modificar profesor.

El paquete **Gestionar Tesis** contiene las funcionalidades: Adicionar tesis, Listar tesis, Buscar tesis, Eliminar tesis y Modificar tesis.

El paquete **Gestionar Tribunales** contiene las funcionalidades: Generar tribunales, Listar tribunales y Buscar tribunales.

El paquete **Gestionar Nomencladores** contiene las funcionalidades: Gestionar Categoría Científica, Gestionar Categoría Docente, Gestionar Área de Trabajo, Gestionar Rol, Gestionar Área de Conocimiento, Gestionar Nivel de Experticia.

Diagrama de Paquetes de Caso de Uso del Sistema.

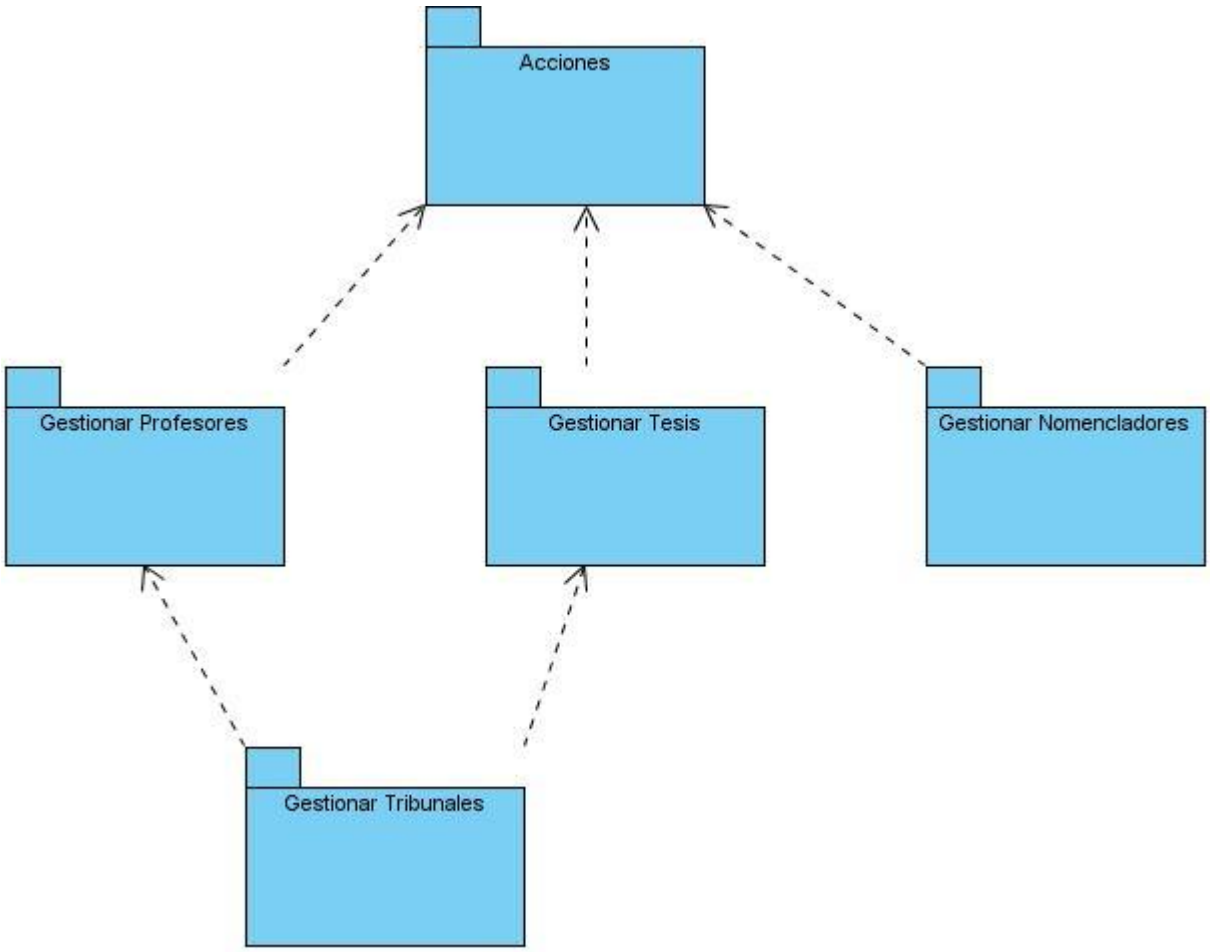


Figura 4. Diagrama de Paquetes de Caso de Uso del Sistema

A continuación se muestran los diagramas de casos de uso de cada uno de los paquetes.

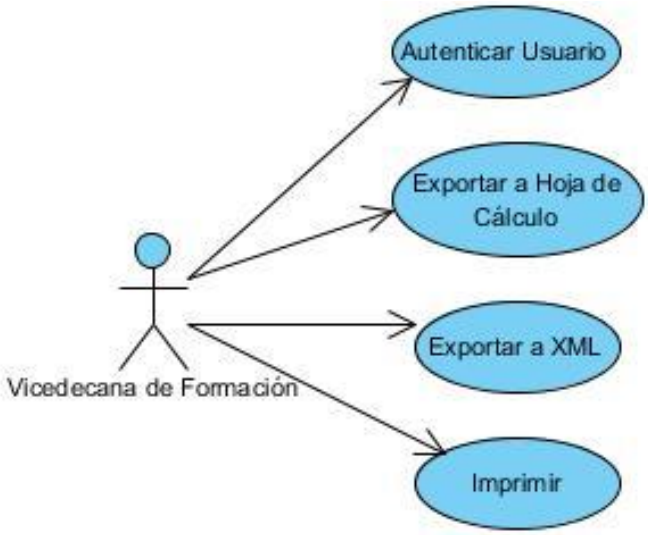


Figura 5. DCUS del paquete Acciones

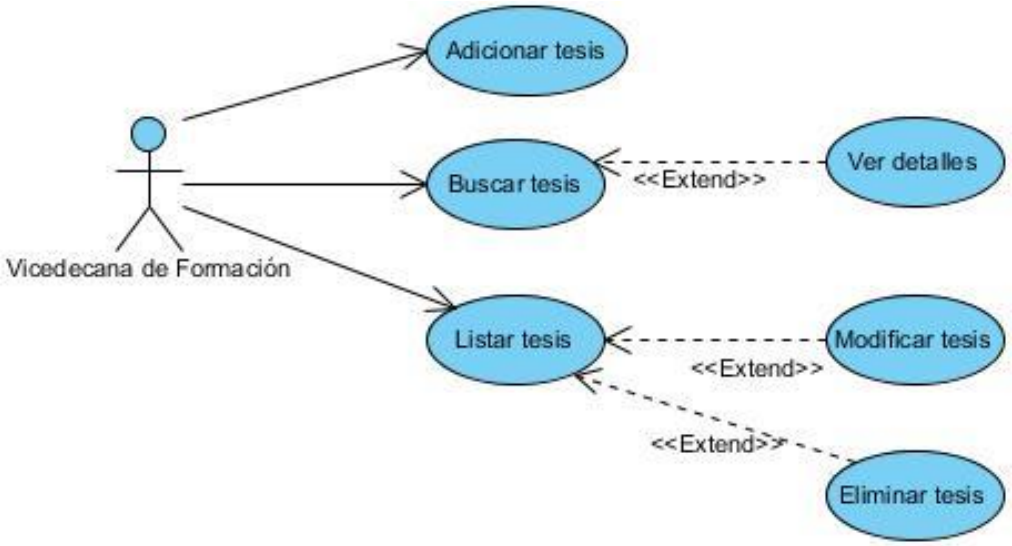


Figura 6. DCUS del paquete Gestionar Tesis

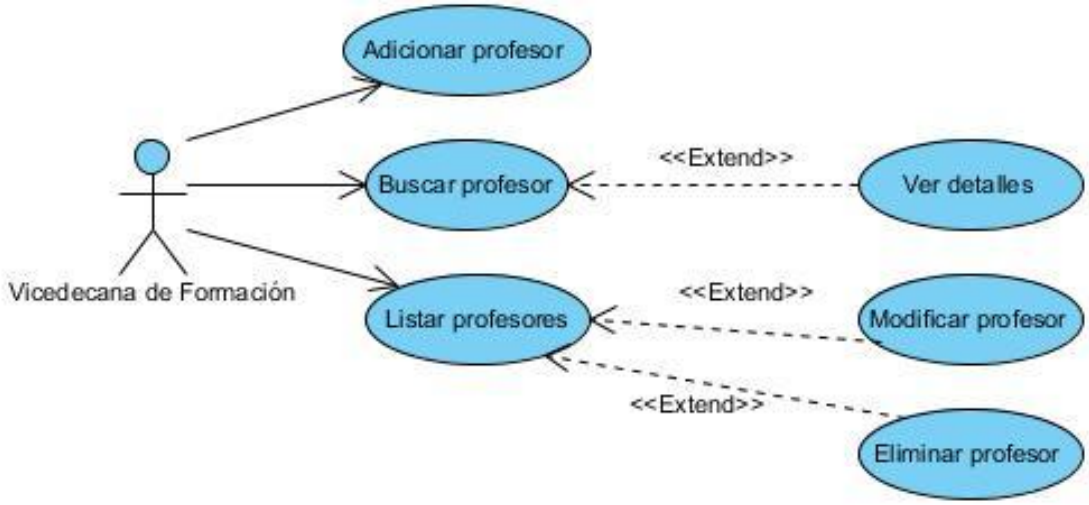


Figura 7. DCUS del paquete Gestionar Profesores

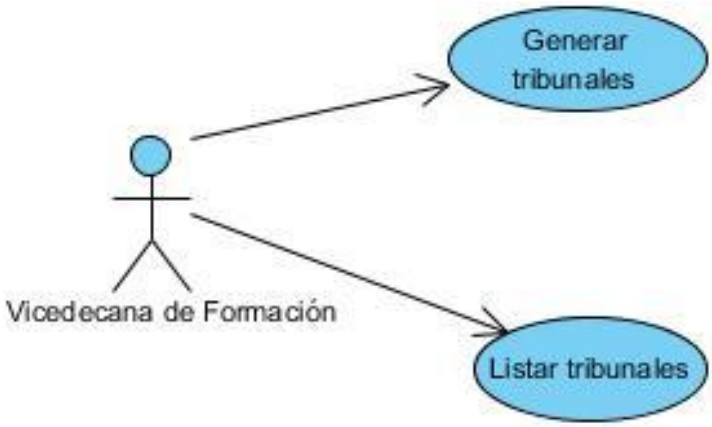


Figura 8. DCUS del paquete Gestionar Tribunales

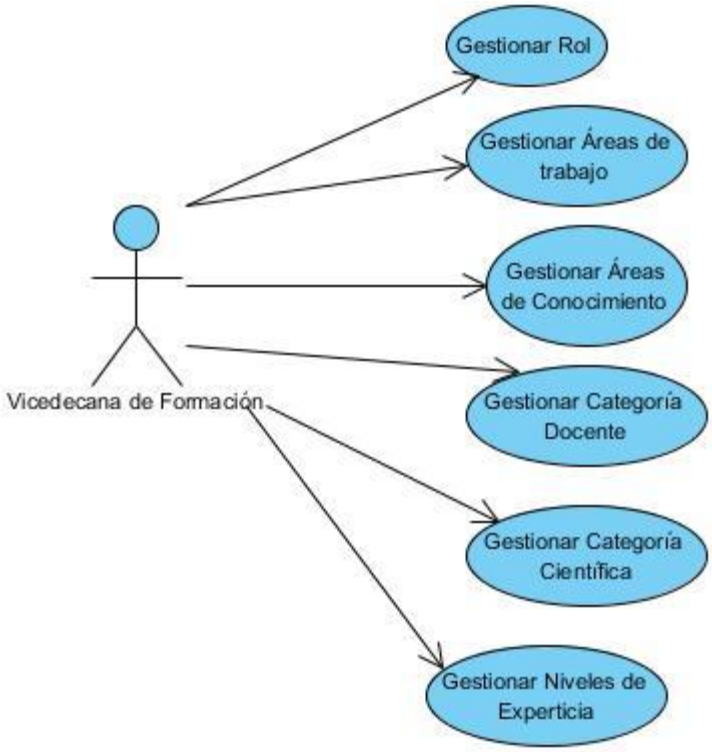


Figura 9. DCUS del paquete Gestionar Nomencladores

2.6 Priorización de los casos de uso:

La Priorización de los casos de uso es muy importante para dejar claras las funcionalidades indispensables, sin las que la aplicación no tiene razón de ser, que deben ser implementadas en el primer ciclo de desarrollo del software, y el resto de las mismas. En la Tabla 2 se puede apreciar la prioridad que se le dio a las funcionalidades del sistema, y los ciclos de desarrollo a las cuales están asociadas.

En el caso de la aplicación que se desea implementar se priorizaron los casos de uso del sistema teniendo en cuenta la importancia para el cliente.

Tabla 2. Priorización de los casos de uso

Caso de Uso	Ciclo	Justificación
Generar tribunales Ver Detalles de Tribunal	1	Es la funcionalidad más importante del sistema, pues cumple con el objetivo general del trabajo de diploma.
Adicionar profesor Eliminar profesor Modificar profesor		La gestión de los datos de los profesores es indispensable para la conformación de los tribunales, por ser la cantera para seleccionar los miembros de dichos

Buscar profesor Listar profesor		tribunales.
Adicionar tesis Listar tesis Eliminar tesis Buscar tesis Modificar tesis		La gestión de las tesis es primordial pues de ellas se parte para conformar los tribunales automáticamente. Sus datos son usados por la funcionalidad implícita en el caso de uso Generar Tribunales.
Autenticar usuario		Garantiza la seguridad del sistema al ser solo el usuario autenticado quien acceda a la información y la modifique.
Gestionar Áreas de Conocimiento Gestionar Roles Gestionar Categorías Científicas Gestionar Categorías Docentes Gestionar Niveles de Experticia Gestionar Centros Productivos	2	La gestión de nomencladores es necesaria a largo plazo, pues si fuese necesario modificar o eliminar alguno de estos elementos, o se necesitara adicionar nuevos, con acceder a cada una de estas funcionalidades sería suficiente, sin actuar directamente sobre el código de la aplicación.
Exportar a Excel		Es el formato que utiliza la Vicedecana de Formación para distribuir los tribunales conformados
Imprimir		La impresión de los tribunales conformados es importante para la distribución de los mismos a los profesores.

2.7 Descripción de casos de uso Críticos

A continuación se describen los Casos de Uso **Generar tribunales y Ver Detalles del Tribunal,** por ser los de mayor importancia para el cliente y por contener el algoritmo que va a dar solución a la situación problemática planteada.

Tabla 3. Descripción del caso de uso Generar Tribunales.

Caso de uso	
CU_1	Generar tribunales

Propósito	Generar de forma automática los tribunales de tesis de pregrado.
Actores	Vicedecana de Formación.
Prioridad	Crítico.
Resumen	El caso de uso se inicia cuando la Vicedecana de Formación necesita generar de forma automática los tribunales de tesis de pregrado. Para esto, selecciona la opción del menú lateral "Generar tribunales", y realiza una serie de pasos para completar su objetivo.
Precondiciones	El usuario debe estar autenticado en el sistema.
Referencias	RF12
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción "Generar tribunales".	2. Obtiene de la base de datos la información referente a los profesores y a las tesis.
	4. Convierte la información obtenida al fichero baseconociemientos.pl.
	5. Realiza la consulta para generar los tribunales.
	6. Obtiene el resultado de la consulta.
	7. Muestra los siguientes datos de cada tribunal: • Área de conocimiento • Presidente • Secretario • Vocal
8. Elige ver Detalles del tribunal	9. Invoca al CU "Ver Detalles del Tribunal"
Prototipo	

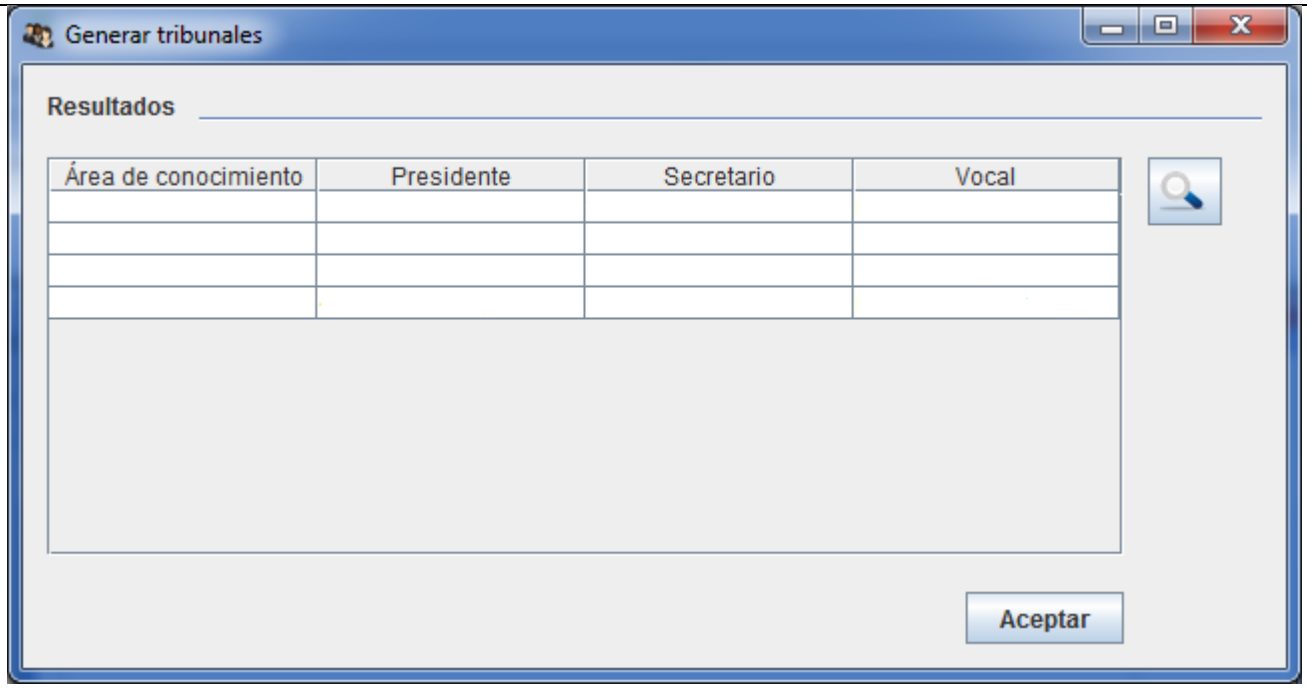


Figura 10. Generar Tribunal

Tabla 4. Ver Detalles de tribunal

Caso de uso	
CU_2	Ver Detalles de tribunales
Propósito	Generar de forma automática los tribunales de tesis de pregrado.
Actores	Vicedecana de Formación.
Prioridad	Crítico.
Resumen	El caso de uso se inicia cuando la Vicedecana de Formación desea ver más detalles de un tribunal Para esto, escoge la opción Ver Detalles del Tribunal.
Precondiciones	El usuario debe estar autenticado en el sistema.
Referencias	RF14
Acción del actor	**Respuesta del sistema**

1. Selecciona la opción "Ver Detalles del Tribunal".	2. Muestra la ventana "Detalles del Tribunal" con los siguientes datos: • Número de tribunal • Área de conocimiento • Presidente • Secretario • Vocal • Total de tesis asignadas • Listado de tesis

Prototipo

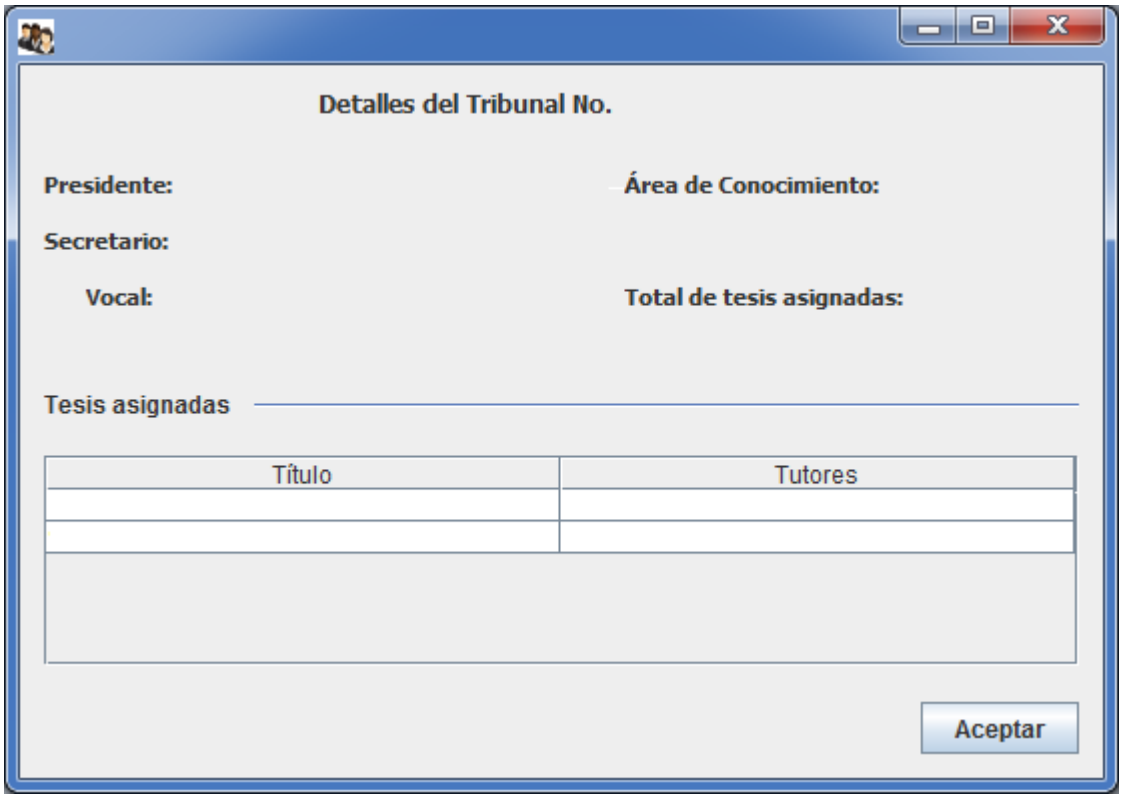


Figura 11. Ver Detalles de Tribunal

Conclusiones del Capítulo

Se implementará una aplicación de escritorio que, de forma automática, generará una propuesta de tribunales de tesis, identificando de cada uno los roles de Presidente, Secretario, Vocal y los Oponentes

de las tesis asignadas al mismo, atendiendo al nivel de experticia en el tema, Categoría Docente y Categoría Científica de sus miembros.

Se definieron los requisitos funcionales y no funcionales necesarios para el desarrollo del sistema.

Se describieron el modelo de negocio y el diagrama de paquetes del sistema con sus respectivos diagramas de casos de uso.

Se justificó la priorización de los casos de uso y se describieron los esenciales: Generar Tribunales y Ver Detalles del Tribunal.

3. CAPÍTULO 3: DISEÑO DEL SISTEMA INFORMÁTICO PARA LA CONFORMACIÓN AUTOMÁTICA DE TRIBUNALES DE TESIS EN LA FACULTAD 2.

En el desarrollo temático de este capítulo se define la notación de programación a utilizar; se describe el patrón de arquitectura a emplear y se listan los patrones de diseño empleados. Además, se especifica el modelo de optimización y se describe el algoritmo utilizado.

3.1. Pautas de Programación

Notación Camel

La notación Camel consiste en escribir los identificadores con la primera letra de cada palabra en mayúsculas y el resto en minúscula. Se le llama a esta notación "Camel" porque los identificadores recuerdan las jorobas de un camello. Existen dos variantes:

- UpperCamelCase, CamelCase o PascalCase: en esta variante la primera letra también es mayúscula.
- lowerCamelCase, camelCase o dromedaryCase: la primera letra es minúscula.

En el lenguaje Java, que es el empleado en el sistema a desarrollar para la creación de las interfaces de usuario y la interacción con la Base de datos, se usa la notación CamelCase en identificadores para clases, y dromedaryCase para métodos y variables.

Un ejemplo del uso de esta notación en la aplicación se aprecia en la siguiente figura, el identificador de la clase ControladoraTesis está escrito usando la variante UpperCamelCase, y para los atributos y métodos se empleó la variante lowerCamelCase.

```
public class ControladoraTesis {

    MProfesor modeloProfesor;
    MTesis modeloTesis;
    MAreaConocimientoAsignada areasConocProfesor;

    public ControladoraTesis() {
        modeloProfesor = new MProfesor();
        modeloTesis = new MTesis();
        areasConocProfesor = new MAreaConocimientoAsignada();
    }

    public int adicionarProfesor(Profesor p) {
        int result = modeloProfesor.adicionar(p); //en result se guarda el id del profesor para
        if (result > 0) {
            int result1 = areasConocProfesor.adicionar(p.getAreasConocAsignadas(), result);
            if (result1 > 0) {
                return result;
            }
        }
        return -1;
    }
```

Figura 12. Ejemplo de uso de Notación Camel

3.2 Patrón de Arquitectura

La arquitectura de software es una descripción de los subsistemas y componentes de un sistema software, y de las relaciones entre ellos. Los subsistemas y componentes son especificados en diferentes vistas para mostrar las propiedades funcionales y no funcionales relevantes del sistema. La arquitectura de software es un artefacto; es el resultado de la actividad de diseño de software. (23)

Un patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de la solución al problema, de forma que puede reutilizarse continuamente. (24)

Los patrones de arquitectura expresan los esquemas de organización estructural fundamental para sistemas software. Además, proveen de un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para la organización de las relaciones entre ellos. (23)

Patrón Arquitectónico seleccionado

Para el desarrollo del sistema se decide emplear el patrón arquitectónico **N-capas,** específicamente el de Tres capas o niveles. Este patrón permite organizar la estructura lógica de gran escala de un sistema, en capas separadas, con distintas responsabilidades, relacionadas en un solo sentido.

Una capa es un elemento de gran escala, a menudo compuesto de varios paquetes o subsistemas.

Como se puede apreciar en la Figura 13, la colaboración y el acoplamiento van desde las capas más altas hacia las más bajas, o sea, relación entre las capas es unidireccional, solo las capas superiores pueden usar los servicios que brindan las capas inferiores.

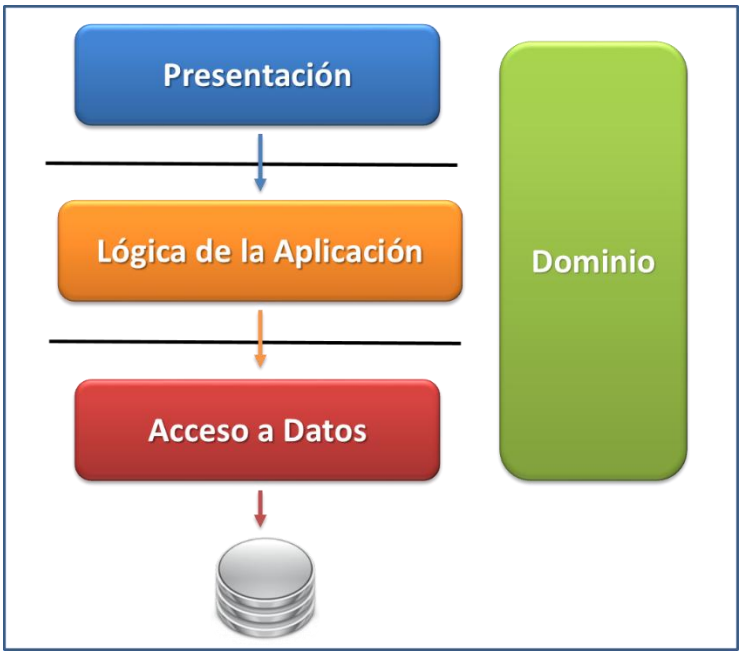


Figura 13. Patrón Tres Capas

La capa de Presentación es la encargada de la interacción con el usuario, y contiene las interfaces de usuario. En la capa intermedia o de Lógica de la aplicación es donde se localiza la lógica de negocio. Esta capa recibe la petición del usuario a través de la capa de Presentación, y se encarga de darle respuesta, implementando las reglas del negocio, las validaciones y los cálculos. Las peticiones a la base de datos se realizan a través de la capa de Acceso a Datos, encargada de proporcionar dicho servicio. La capa de Dominio contiene las clases entidades, que representan las tablas de la base de datos.

La Figura 14 ilustra el diagrama de paquetes del diseño, que muestra cómo se aplica el patrón 3 capas en la aplicación.

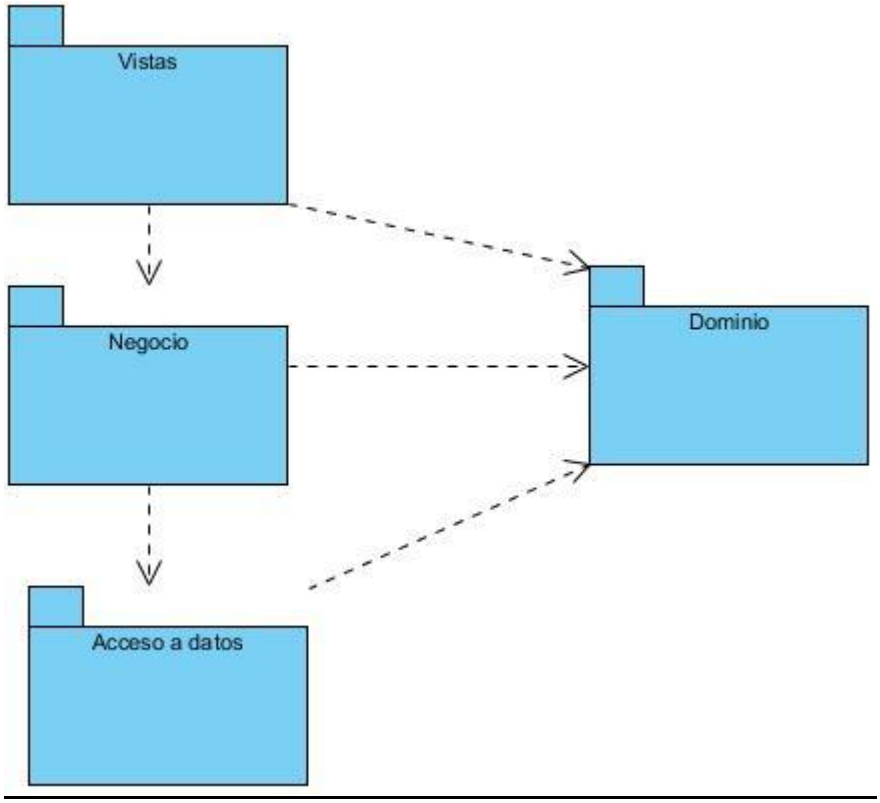


Figura 14. Diagrama de paquetes del Diseño

A continuación se muestra los diagramas de clases de los Paquetes Vistas y Dominio.

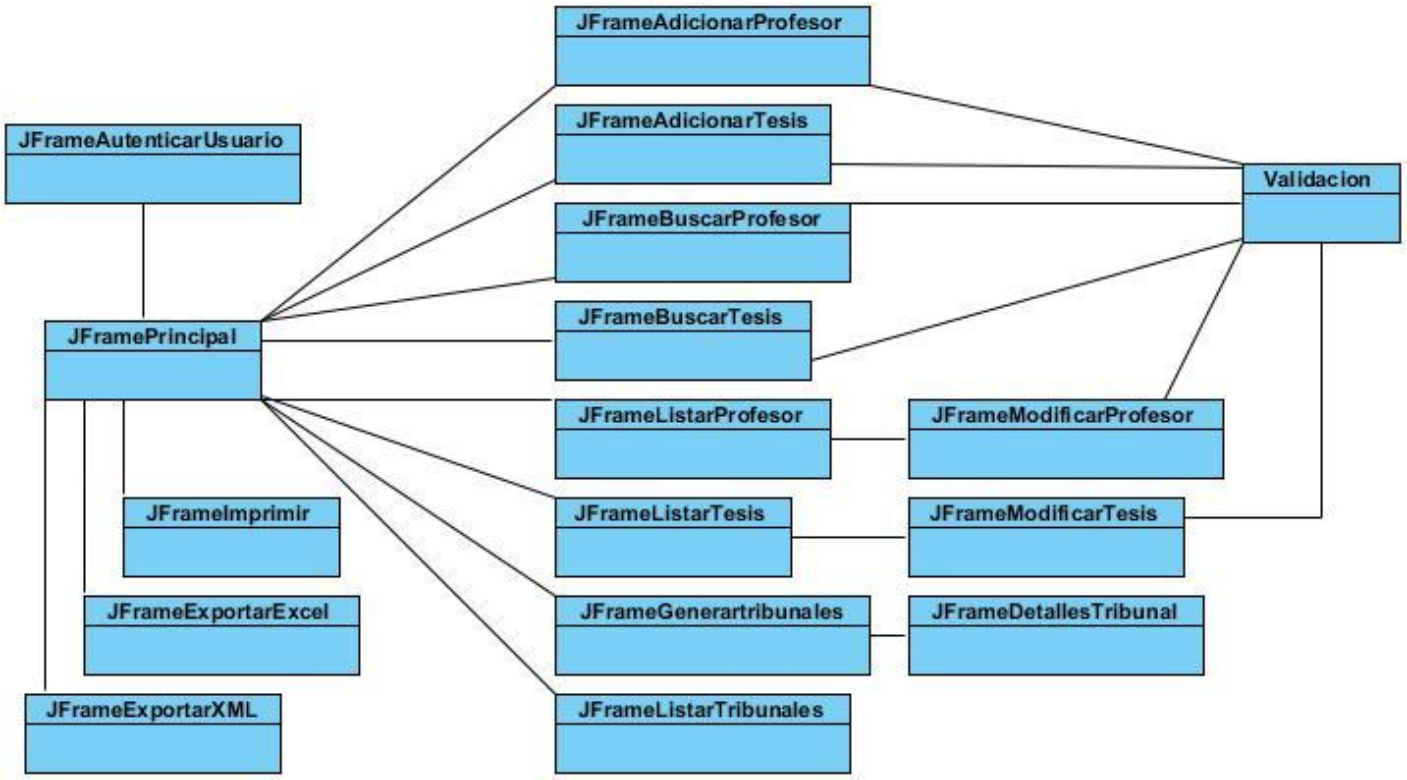


Figura 15. Diagrama de clases del Paquete Vistas

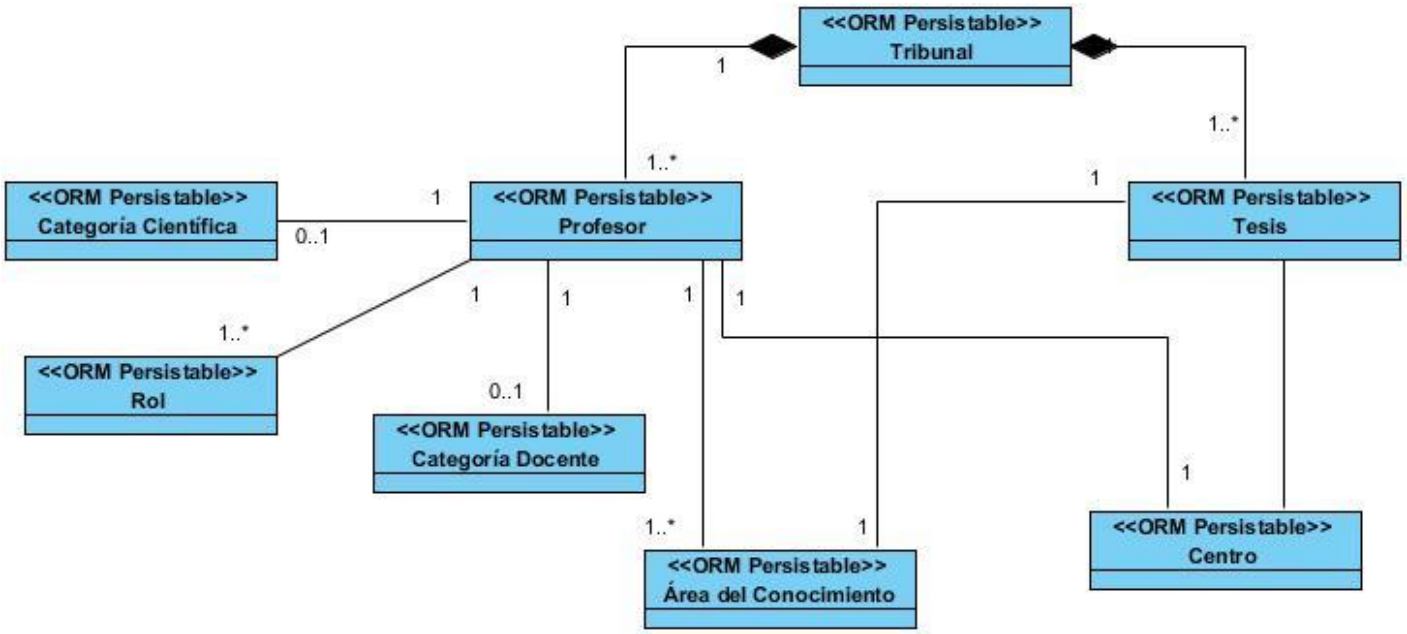


Figura 16. Diagrama de clases del paquete Dominio

3.3 Patrones de Diseño

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva y reusable. El uso de los patrones de diseño beneficia en gran medida el desarrollo y la calidad del software, pues evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y resueltos anteriormente, además formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño.
Para el desarrollo del sistema se han usado los patrones de diseño GRASP (*General Responsibility Assignment Software Patterns*), para asignar responsabilidades a las clases del sistema. En el diseño Orientado a Objetos se emplean: Experto, Creador, Alta Cohesión, Bajo Acoplamiento, Controlador.

Patrón **Experto**: Asigna una responsabilidad al experto en información, o sea, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Aplica la idea de que los objetos realizan acciones relacionadas con la información que poseen.

Patrón **Creador**: Guía la asignación de responsabilidades relacionadas con la creación de objetos. Tiene como propósito fundamental encontrar un creador para conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

Patrón **Bajo Acoplamiento**: Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. Es uno de los principios del diseño que influyen en la decisión de asignar responsabilidades.

Patrón **Alta Cohesión**: Mejora la claridad y la facilidad con que se entiende el diseño, se simplifican el mantenimiento y las mejoras en funcionalidad, a menudo se genera un bajo acoplamiento. Además, soporta una mayor capacidad de reutilización, una clase muy cohesiva puede destinarse a un propósito muy específico.

Patrón **Controlador**: Un controlador es un objeto de interfaz no destinada al usuario. Se asocia a operaciones del sistema generadas por un actor externo o en respuesta a eventos del sistema. El controlador es el encargado de asignar la responsabilidad del manejo de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- La empresa u organización global.
- Un manejador artificial de todos los eventos del sistema de un caso de uso
- Utilice la misma clase de controlador con todos los eventos del sistema en el mismo caso de uso.

A continuación se muestran cómo se evidencian en la aplicación:

El patrón Experto y el Controlador se pueden ver en la clase ControladoraTesis, que contiene toda la información necesaria para cumplir con estas responsabilidades.

```
21  public class ControladoraTesis {
22
23      MProfesor modeloProfesor;
24      MTesis modeloTesis;
25      MAreaConocimientoAsignada areasConocProfesor;
26
27      public ControladoraTesis() {
28          modeloProfesor = new MProfesor();
29          modeloTesis = new MTesis();
30          areasConocProfesor = new MAreaConocimientoAsignada();
31      }
32
33      public int adicionarProfesor(Profesor p) {...}
43
44      public int adicionarTesis(Tesis t) {...}
49
50      public Tesis buscarTesisCompleto(String nA, String apA, String nT, String apT, String aConoc) {...}
53
54      public Tesis buscarTesisPorNombreAutor(String nA, String apA) {...}
57
58      public Tesis buscarTesisPorNombreTutor(String nT, String apT) {...}
61
62      public List<Profesor> buscarProfesorPorNombre(String nombre) {...}
71
72      public List<Profesor> buscarProfesorPorApellidos(String apellidos) {...}
81
82      public List<Profesor> buscarProfesorPorNombreYApellidos(String nombre, String apellidos) {...}
92
93      public List<Profesor> buscarProfesorPorCatDoc(String catDoc) {...}
103
104     public List<Profesor> buscarProfesorPorAreaTrabajo(String areaTrab) {...}
113
114     public int modificarProfesor(Profesor profesorViejo, Profesor profesorNuevo) {...}
131
132     public int modificarTesis(Tesis tVieja, Tesis tNueva) {
```

Figura 17. Evidencia de Patrones GRASP (Controlador)

El patrón Creador se evidencia en la clase JFrameAdcicionarProfesor, que se muestra en la Figura 18.

```
public class JFrameAdicionarProfesor extends javax.swing.JFrame {
    /**...*/
    ControladoraTesis controlTesis;
    ControladoraNomencladores controlNomencladores;
    Validacion validar;
    List<AreasConocimientoAsignadas> areasAsignadas;
    String[] titulos = {"Área de conocimiento", "Nivel de experticia", "Años"};
    DefaultTableModel modelo = new DefaultTableModel(titulos, 5);

    public JFrameAdicionarProfesor(ControladoraTesis contTesis) {...}
    /**...*/
    @SuppressWarnings("unchecked")
    Generated Code

    private void jTextFieldAnnosExpeienciaProfesorActionPerformed(java.awt.event.ActionEvent evt) {...}
    private void jButtonCancelarMouseClicked(java.awt.event.MouseEvent evt) {...}
    private void jButtonAddProfesorMouseClicked(java.awt.event.MouseEvent evt) {...}

    private void jButtonAddProfesorActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        if (validarCampos()) {
            String nombre = jTextFieldNombre.getText();
            String apellidos = jTextFieldApellidos.getText();
            String catDoc = jComboBoxCatDocenteProfesor.getSelectedItem().toString();
            String catCient = jComboBoxCatCientProfesor.getSelectedItem().toString();
            String areaTrabajo = jComboBoxCentroProfesor.getSelectedItem().toString();
            boolean especialista = jCheckBox1.isSelected();
            int annosExp = Integer.parseInt(jTextFieldAnnosExpDocente.getText());

            Profesor profesor = new Profesor(0,nombre, apellidos, catDoc, catCient, areasAsignadas, areaTrabajo, especia
            if (controlTesis.adicionarProfesor(profesor) > 0) {
                JOptionPane.showMessageDialog(this, "El profesor ha sido adicionado", "Profesor adicionado", 1);
                limpiarCampos();
                areasAsignadas = new ArrayList<AreasConocimientoAsignadas>();
            }
```

Figura 18. Evidencia de Patrones GRASP (Creador)

3.4 Modelo de Datos

El modelo de datos se construye a partir del Diagrama de clases persistentes y en él, se representan las tablas en la Base de Datos. En la Figura 16, se muestra el modelo de datos del Sistema Informático para la conformación automática de tribunales de tesis.

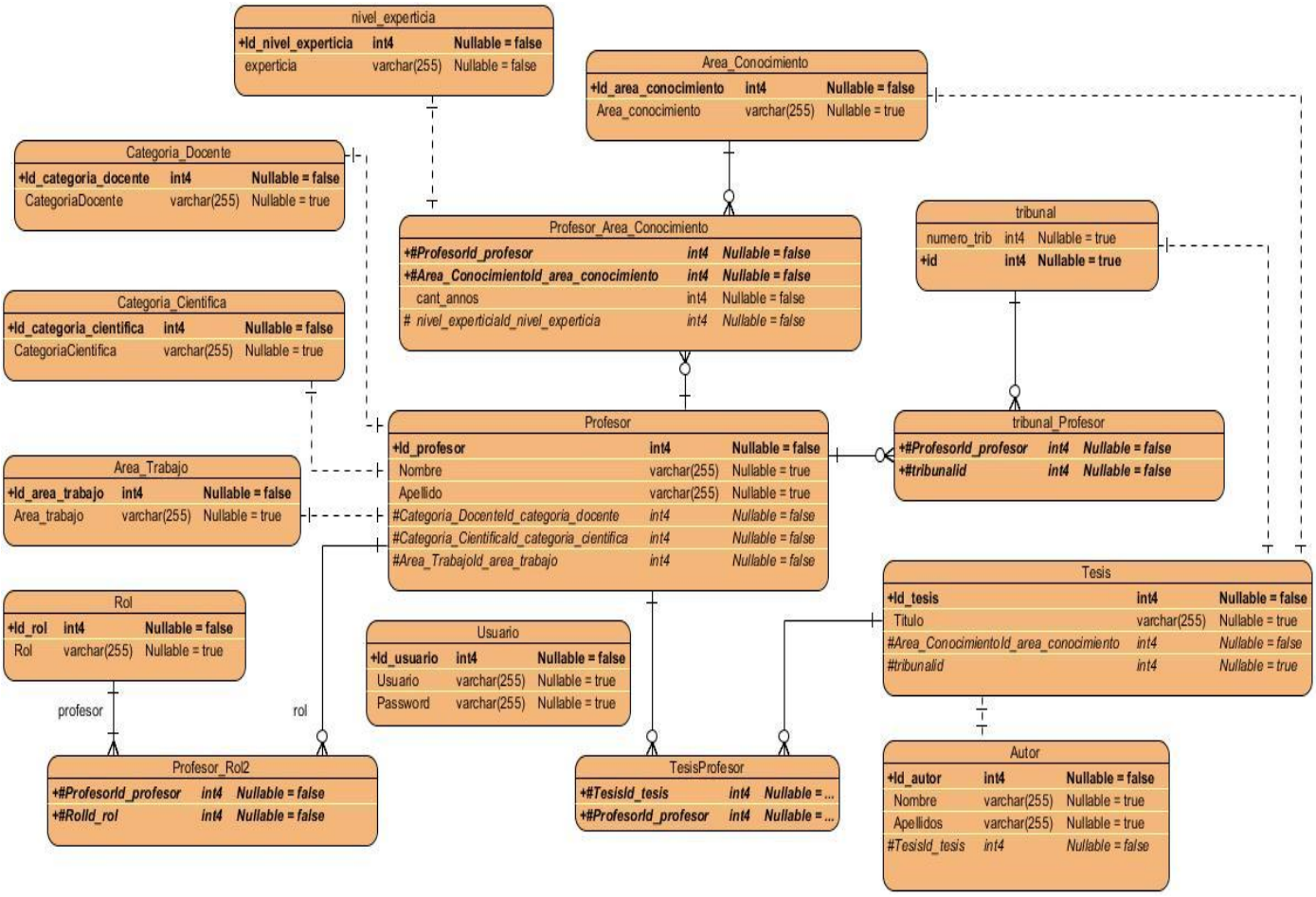


Figura 19. Modelo de Datos

El modelo contiene una tabla profesor (contiene los datos de los profesores de la facultad 2). La misma se relaciona con la tabla CategoriaDocente (nomenclador con Instructor, Asistente, Auxiliar, Titular), CategoriaCientifica (nomenclador con Ingeniero, Licenciado, Master y Doctor), AreaTrabajo (nomenclador con Facultad 2, Centro TLM y Centro ISEC), con la tabla TribunalProfesor (contiene los id de Tribunal y de Profesor), dicha tabla se forma debido a la relación de muchos a muchos existente entre la clase Profesor y Tribunal, con la tabla ProfesorAreaConocimiento (contiene los id de la tabla Profesor y AreaConocimiento, la cantidad de años que lleva desarrollando esa área el profesor, y el identificador del

nivel de experticia asociado). Dicha tabla se forma debido a la relación de muchos a muchos existente entre la tabla Profesor y AreaConocimiento (nomenclador que contiene las áreas del conocimiento que se les asocia a las tesis y son las que han trabajado los profesores), con la tabla TesisProfesor (contiene los id de la tabla Profesor y Tesis), dicha tabla se forma debido a la relación de muchos a muchos existente entre la tabla Profesor y Tesis (contiene los datos de las tesis de la facultad 2), con la tabla ProfesorRol (contiene los id de la tabla Profesor y Rol)Rol (nomenclador con Profesor, Presidente, Secretario, Vocal, Oponente).

La tabla Tesis se relaciona con la tabla Tribunal, pues una tesis está en un Tribunal y un Tribunal puede tener una o varias tesis. Además se relaciona con la tabla AreaConocimiento, pues el tema de la tesis está asociado a un área de conocimiento, y con la tabla Autor, pues una tesis tiene asociados uno o varios autores.

3.5 Modelo de Optimización a utilizar

La situación problemática que justifica la investigación se puede modelar como un problema de optimización combinatoria, particularmente un problema de distribución de recursos. En este caso, se deben distribuir recursos humanos, que son los profesores, para lograr la conformación de los tribunales, cumpliendo con el principio de integralidad (el tribunal como un todo, debe evaluar las habilidades metodológicas y las técnicas).

Un problema de optimización combinatoria tiene como objetivo encontrar el máximo o el mínimo de una función específica, en un conjunto muy grande de soluciones discretas, sujeta a restricciones. El conjunto muy grande sería el compuesto por las combinaciones de profesores al conformar tribunales, y en este caso, habría que maximizar la capacidad de evaluación del tribunal definida por la función objetivo.

3.5.1 Modelo de Optimización

Sea $Te = \{T_1, T_2, \ldots, T_m\}$ el conjunto de las tesis a asignar,

Donde

$Te_i = (At_i)$

Siendo At_i el área de conocimiento asociada a la tesis Te_i.

Sea $T = \{Tr_1, Tr_2, \ldots, Tr_n\}$ el conjunto de tribunales

Sea $Tr_i = (R_1, R_2, R_3)$ el tribunal i-ésimo

Donde

$R_i = (CC_i, CD_i, AED_i, \{Ac_{i1}, \ldots, Ac_{ic}\})$

R_i representa el profesor i-ésimo, miembro del tribunal Tr_i.

Las variables CC_i, CD_i, $NExp_i$, son categóricas ordinales; para su tratamiento se decidió asociarle un valor numérico que expresara el orden de las diferentes categorías. A partir de los requerimientos funcionales definidos, se determinó que no existía diferencia uniforme entre las categorías de estas variables, por lo cual estos valores numéricos asociados fueron definidos teniendo en cuenta lo anterior.

Siendo

$$CC_i = \begin{cases} s \; si \; Licenciado \\ \quad o \; Ingeniero \\ 2s \; si \; Máster \\ 5s \; si \; Doctor \end{cases}$$

CC_i es la categoría científica asociada al profesor i.

Como se puede apreciar, para la determinación de CC_i, se ha decidido ponderar sobre la base del supuesto de que los saltos cuantitativos s, 2s y 5s, expresan las diferencias existentes entre las Categorías Científicas. De esta forma, el valor numérico asociado a la categoría Máster será el doble de la categoría Licenciado o Ingeniero, debido a los conocimientos y la experticia que posee este último respecto al primero. Con este mismo principio, se expresó la diferencia de orden que existirá entre un Doctor y un Máster, siendo esta el doble más uno, resaltando así el conocimiento y la experticia de la categoría Doctor.

$$CD_i = \begin{cases} t \; si \; Instructor \\ t+2 \; si \; Asistente \\ 2(t+2) \; si \; Auxiliar \\ 3(t+2) \; si \; Titular \end{cases}$$

CD_i es la categoría docente asociada al profesor i.

Para la determinación de CD_i, se ha decidido ponderar sobre la base del supuesto de que los saltos cuantitativos *t, t+2, 2(t+2) y 3(t+2),* expresan las diferencias existentes entre las Categorías Docentes. De esta forma, el valor de la categoría Instructor dista poco del valor de la categoría Asistente, debido a que la cantidad de años de experiencia que se requiere para el cambio para alguna de estas categorías es similar. El valor numérico asociado a la categoría Auxiliar será el doble de la categoría Asistente, debido a los conocimientos, la categoría científica y los años de experiencia que posee este último con respecto al primero. Con este mismo principio, se expresó la diferencia de orden que existiría entre Titular y Auxiliar, siendo el valor numérico de Titular mayor, por el conocimiento los años de experiencia y la categoría científica asociados.

AED_i los años de experiencia como docente del profesor i.

$Ac_{ij} = (AExp_j, NExp_j)$

Ac_{ij} es el área de conocimiento j del profesor i.

Siendo

$AExp_j$ los años de experiencia del profesor i en el área de conocimiento j.

$$NExp_i = \begin{cases} u \; si \; Básico \\ 2u \; si \; Intermedio \\ 2u + 1 \; si \; Avanzado \end{cases}$$

$NExp_j$ representa el nivel de experticia del profesor i en el área de conocimiento j. La diferencia de orden del nivel Intermedio es el doble del valor numérico del nivel Básico, debido a que los conocimientos en un área determinada son superiores en el primero con respecto al segundo. Se siguió el mismo principio para definir la diferencia de orden del nivel Avanzado con respecto a los anteriores.

Se desea minimizar la **función objetivo** definida por:

$$g = \max_{t \in T}\{f(t)\} - \min_{t \in T}\{f(t)\}$$

Siendo

$$f(t) = \sum_{i=1}^{3}(T_i + M_i)$$

$f(t)$ es la función de evaluación de habilidades del tribunal t.

Donde

$T_i = EAC_i + CC_i$

T_i representa la evaluación de las habilidades técnicas del profesor i, y está dada por la suma de la evaluación del área de conocimiento j del profesor i y la categoría científica del profesor i.

Suponiendo que en caso de ostentar alguna Categoría científica, el profesor i tendrá como Área del conocimiento más ponderada la que está directamente relacionada con dicha categoría.

$EAC_{ij} = x \times AE_{ij} + y \times NExp_{ij}$

AC_{ij} significa la evaluación del área de conocimiento j del profesor i.

Nótese que esta evaluación es el resultado de la suma de la cantidad de años que el profesor i lleva desarrollándose en el área j y el nivel de experticia que ha alcanzado en dicha área.

$M_i = a \times CD_i + b \times AED_i + c \times CC_i$

M_i es la evaluación de las habilidades metodológicas del profesor i, está dada por la suma de la Categoría docente, la Categoría científica y los años de experiencia como docente del profesor i.

$a, b, c \in \mathbb{N}$

Sujeta a las restricciones:

$i, j, m, n > 0$

$a > c > b$

Esta restricción obedece al supuesto de que mientras mayor categoría docente se tenga mayores serán los conocimientos metodológicos, debido a los años de experiencia y a la categoría científica asociada. De esta forma, pondera en primer lugar la categoría docente, después la categoría científica y por último los años de experiencia como docente.

$y > x$

Esta restricción responde al hecho de que en la evaluación del área de conocimiento *j* del profesor *i* pondera en primer lugar el nivel de experticia, y en segundo, los años de experiencia en esta área.

3.6 Algoritmo a desarrollar

Para la solución del problema de optimización combinatoria modelado anteriormente, se empleará la metaheurística GRASP. A continuación se describe el algoritmo diseñado aplicando esta metaheurística.

En la primera fase (Construcción), se construye iterativamente una solución posible, considerando un elemento en cada paso. Los elementos son representados por los tribunales, y la posible solución sería la lista de tribunales. Para construir un tribunal, se deben agrupar las tesis registradas a partir de las áreas de conocimiento que tienen asociadas. De igual forma se procede con los profesores, que son agrupados por las áreas de conocimiento asociadas a las tesis. Se debe construir como mínimo un tribunal por área de conocimiento, y como máximo la tercera parte de la cantidad de tesis de dicha área, pues será tres el número de tesis por tribunal.

Dentro de la fase Construcción se emplea una función ávida para determinar la elección del próximo elemento que será añadido a la solución parcial. En el caso del algoritmo propuesto, se emplean dos funciones ávidas: una para escoger los presidentes, que serán evaluados con respecto a sus habilidades técnicas, y otra para los Secretarios y Vocales, cuya evaluación estará dada por las habilidades metodológicas.

Del listado de profesores general, se obtiene una lista de candidatos a Presidentes agrupados por área de conocimiento, ordenados por la función ávida correspondiente a este rol. Posteriormente, se obtendrá una sublista, denominada Lista de Candidatos Restringida, de la cual se escogerá aleatoriamente un profesor, que ocupará el rol de Presidente del tribunal. A continuación, se seleccionará del listado de tesis, aquellas en las que el Presidente en cuestión no coincida como tutor. En caso de que una tesis no pueda ser

atendida por un profesor, será asignada posteriormente, al primer tribunal que pueda atenderla. La lista de tesis junto al profesor seleccionado constituirán los primeros elementos del tribunal a formar. Se actualizarán las listas iniciales de tesis y profesores, para evitar que aparezcan resultados duplicados. El proceso desde la obtención de la lista de candidatos se repite para los otros miembros del tribunal (Secretario y Vocal), y teniendo en cuenta que ya han sido asignadas las tesis, se asignan estos miembros cuando no coincidan con los tutores de las mismas. La primera solución posible está construida cuando todos los tribunales hayan sido conformados.

Una vez conformada esta solución, se calculará el valor los tribunales en la función de evaluación de habilidades, ordenándose luego por estos resultados, para preparar el escenario de la siguiente fase.

La segunda fase de GRASP, denominada Mejora, implica una búsqueda local en la vecindad de los elementos de la solución, generando una nueva, que se compara con la existente y, de ser mejor, se actualiza. Una vez ordenados los tribunales por su valor en la función evaluación de habilidades, se escogen los extremos, cuya diferencia en la evaluación de habilidades será el valor a minimizar. Se realizarán las permutaciones posibles entre los Secretarios y Vocales de los tribunales de los extremos, para obtener una nueva solución, que deberá ser evaluada nuevamente, y comparada con la anterior, escogiendo la mejor de ambas.

El proceso de mejora y actualización de la solución concluye una vez que después de 3 iteraciones no se pueda mejorar más la solución existente.

Conclusiones del Capítulo.

Para el desarrollo del sistema se emplea la notación CamelCase en identificadores para clases, y dromedaryCase para métodos y variables, por ser la notación asociada al lenguaje Java.

Se seleccionó como patrón de arquitectura el Patrón 3 capas.

Se empleará la totalidad de los patrones de diseño descritos.

Se modeló el problema a resolver como un problema de optimización combinatoria y, finalmente, se diseñó el algoritmo para generar automáticamente los tribunales de tesis empleando la metaheurística GRASP.

4. CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA INFORMÁTICO PARA LA CONFORMACIÓN AUTOMÁTICA DE TRIBUNALES DE TESIS EN LA FACULTAD 2.

En este capítulo se muestra la distribución física del sistema mediante el Diagrama de Despliegue, además de los casos de prueba para uno de los casos de uso críticos.

4.1 Diagrama de Despliegue del Sistema Informático para la conformación automática de los tribunales de tesis de la Facultad 2.

El diagrama de despliegue representa la distribución física del sistema, partiendo de cómo se distribuyen las funcionalidades entre los nodos, los cuales se representan como máquinas físicas y procesadores. Los nodos, constituyen recursos de cómputo entre los cuales existen relaciones que representan el medio de comunicación.

En la Figura 17, se explica la utilización y conexión con el sistema. De manera general, el usuario (que en este caso es representado por la Vicedecana de Formación), debe utilizar una Computadora Personal (PC, por sus siglas en inglés), en la que estará corriendo la aplicación. En esta computadora estará instalado además, el gestor de base de datos PostgreSQL, facilitando el acceso a la información con independencia de la red de datos. Se debe contar con una impresora conectada a la computadora para permitir la impresión de los tribunales generados por el sistema.

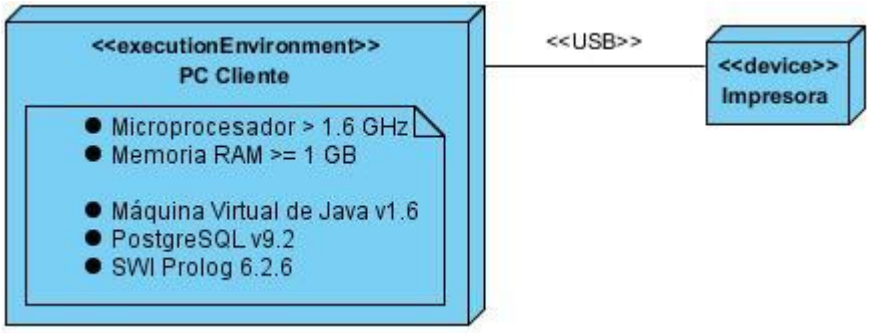


Figura 20. Diagrama de Despliegue del Sistema

4.2 Pruebas al sistema

Las pruebas son la actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas. Los resultados son observados y registrados, y se realiza la evaluación de algún aspecto del sistema o componente. La prueba de software, es un elemento crítico para la garantía de la calidad del mismo y representa una revisión final de las especificaciones del diseño y de la codificación. La prueba

está enfocada principalmente en la evaluación y determinación de la calidad del producto. Estas están definidas en estrategias, niveles, tipos, métodos y técnicas de prueba.

4.2.1 El método de prueba de caja negra

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Una prueba de este tipo, examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software.

A continuación se muestran los casos de pruebas de uno de los Casos de Uso críticos del sistema.

Caso de Uso: Autenticar usuario

Descripción general

Permitir al usuario el acceso al sistema.

Condiciones de ejecución

No aplica.

Matriz de Datos

SC Autenticar Usuario

Tabla 5. Caso de Prueba Autenticar Usuario

Escenario	Descripción	Variable 1 Usuario	Variable 2 Contraseña	Respuesta del sistema	Flujo central
EC 1.1 Usuario autenticado.	Permite que el usuario se autentique en el sistema.	V Maidelis	V password	Muestra la interfaz principal del sistema.	1. Inicializar el sistema. 2. Introducir usuario y contraseña. 3. Presionar el botón que permite autenticar el usuario en el
EC 1.2 Datos incorrectos.	Permite mostrar al usuario los datos incorrectos en el momento de autenticarse.	I Maidelis1	V password	Muestra un mensaje de error notificando que los datos son incorrectos.	

EC 1.3 Cancelar	Permite al usuario cancelar la opción de autenticarse en el sistema.	V Maidelis	V password	Cierra el sistema.	sistema.

Tabla 6. Descripción de las Variables del CU Autenticar Usuario

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Usuario.	Cuadro de Texto.	No.	Solo puede contener letras.
2	Contraseña.	Cuadro de texto.	No.	Puede contener cualquier combinación de letras, números y caracteres extraños.

4.2.2 Resultados de las pruebas

Se realizaron tres iteraciones de pruebas. Los resultados de las mismas se especifican en la Figura 20.

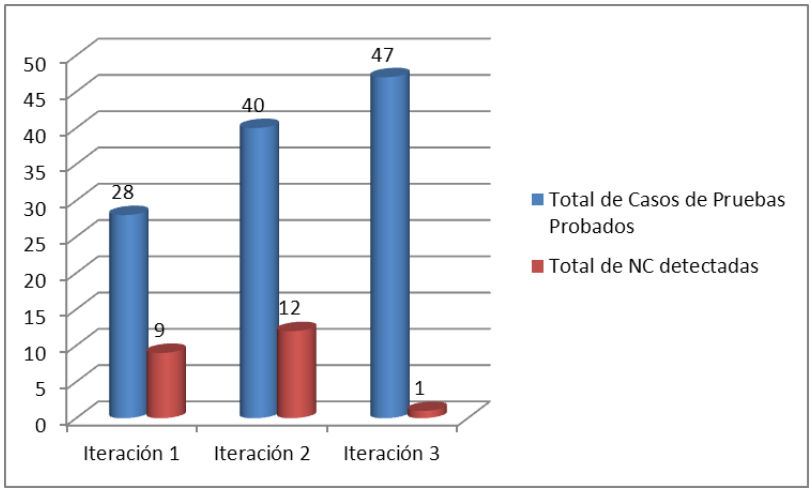


Figura 21. Resultados de las Pruebas

En la primera iteración se probaron 28 casos de pruebas y se detectaron 9 no conformidades.

En la segunda iteración se probaron 40 casos de pruebas y se detectaron 12 no conformidades.

En la tercera iteración se probaron 47 casos de pruebas y se detectó 1 no conformidad.

Todas las no conformidades detectadas fueron resueltas satisfactoriamente, quedando el sistema libre de errores.

Conclusiones del Capítulo

Para desplegar el sistema, el usuario debe utilizar una Computadora Personal en la que estará corriendo la aplicación. En esta computadora estará instalado además, el gestor de base de datos PostgreSQL, facilitando el acceso a la información con independencia de la red de datos. Se debe contar con una impresora conectada a la computadora para permitir la impresión de los tribunales generados por el sistema.

Se aplicaron pruebas de caja negra al sistema, quedando libre de fallos.

5. CONCLUSIONES GENERALES

Como fruto del decurso investigativo, se han arribado a las siguientes conclusiones:

1. En el estudio se realizó un análisis de los conceptos fundamentales y la formalización del estado del arte de los sistemas de distribución automática de recursos, evidenciándose la inexistencia de una variante concreta que le brinde solución al problema a resolver que ha sido planteado en el diseño de la investigación.
2. Para la implementación de la solución, se utilizaron los lenguajes de programación Java y Prolog, los entornos de desarrollo SWI Prolog y Netbeans para cada lenguaje respectivamente. La metodología de desarrollo empleada fue RUP, con UML como lenguaje de modelado y la notación de modelado de procesos BPMN.
3. Se modeló el problema de optimización combinatoria, y se diseñó el algoritmo para la conformación automática de tribunales, empleando la técnica metaheurística GRASP.
4. Se implementó una aplicación de escritorio que, de forma automática, genera una propuesta de tribunales de tesis, identificando de cada uno, los roles de Presidente, Secretario y Vocal, y los Oponentes de las tesis asignadas al mismo, atendiendo al nivel de experticia en el tema, Categoría Docente y Categoría Científica de sus miembros.
5. Se aplicaron pruebas de caja negra al sistema, quedando libre de fallos.

RECOMENDACIONES

- Aplicar el sistema en la Facultad 2 en el proceso de culminación de estudio del curso 2013-2014, y realizar un estudio de los resultados obtenidos. Sobre la base de estos resultados, generalizar la aplicación de la propuesta a nivel de universidad.
- Implementar el sistema como una aplicación web, para una mejor distribución y consulta de sus resultados.
- Realizar un estudio comparativo entre los resultados de la propuesta de solución y del procedimiento actual.
- Profundizar en el análisis de la selección de los coeficientes de las variables en el modelo de optimización utilizado.
- Realizar un análisis para definir la cantidad de tesis a asignar por cada tribunal.
- Utilizar otras metaheurísticas en la solución del problema y comparar las soluciones obtenidas con la propuesta realizada.
- Modelar el problema de optimización como multiobjetivo, buscando equilibrio al minimizar la diferencia entre las evaluaciones de integralidad de los tribunales, y calidad, al maximizar la evaluación integral de los mismos.

REFERENCIAS BIBLIOGRÁFICAS

1. Superior, Ministerio de Educación. RESOLUCIÓN No. 210 del 2007. *Reglamento para el Trabajo Docente y Metodológico.* La Habana : s.n., 2006.

2. Alvarez de Zayas, C. *Hacia una escuela de excelencia.* La Habana : Editorial Academia, 1996. pág. 3.

3. Conferencia PDE IPLAC. Impresión ligera. . Ciudad de La Habana : s.n., 1995.

4. WordReference.Online Language Dictionaries. *Spanish Definition from RAE.* [En línea] 2013. http://www.wordreference.com/es/en/frames.asp?es=tribunal.

5. Mathematical Programming Glossary, INFORMS Computing Society. *The Nature of Mathematical Programming .*

6. Martí, Rafael. Procedimientos Metaheurísticos en Optimización Combinatoria. Valencia : s.n.

7. Enrique Castillo, Antonio J. Conejo, Pablo Pedregal, Ricardo García y Natalia Alguacil. *Formulación y Resolución de Modelos de Programación Matemática en Ingeniería y Ciencia.* 2002.

8. Coppin, Ben. *Artificial Intelligence Illuminated.* 1. Sudbury : Jones and Bartlett Publishers, 2004. pág. 768. 0-7637-3230-3.

9. Díaz, Adenso, et al. *Optimización Heurística y Redes Neuronales.* España : Editorial Paraninfo, 1996.

10. Osman, I. H. and Kelly J. P. *Meta-Heuristics: theory and applications.* Boston : Kluwer Academic, 1996. págs. 1-21.

11. Fernando Borrás Rocher, José Vicente Segura Heras, Joaquín Sánchez Soriano, Manuel A. Pulido Cayuela, Jesús Tadeo Pastor Ciurana. Confección automática de tribunales para las pruebas de acceso a la universidad. Ubeda : XXVI Congreso Nacional de Estadística e Investigación Operativa: Úbeda, 6-9 de noviembre de 2001, 2001, ISBN 84-8439-080-2, 2001.

12. Facundo E. Cancelo, Pablo N. Cababie, Gabriel Barrera, Daniela López De Luise. *Un nuevo enfoque para asignación óptima de múltiples recursos.* Buenos Aires, Argentina : Universidad de Palermo.

13. Jorge Alberto Ospina Falla, Rafael Ricaurte Bernal. *Solución de un problema de asignación de horarios de mantenimiento en máquinas, equipos e instalaciones a la empresa Powel Continental Ltda. mediante la programación de un algoritmo genético.* Chia, Colombia : s.n., 2004.

14. IBM. *IBM.* [En línea] 2013. [Citado el: 16 de 1 de 2013.] http://www.ibm.com/software/rational/uml/.

15. Largo, Faraón LLorens. Prácicas Prolog. Alicante : s.n.

16. Java. *Learn About Java Technology.* [En línea] [Citado el: 10 de enero de 2013.] http://www.java.com/en/about/.

17. SWI-Prolog's Home. [En línea] [Citado el: 15 de diciembre de 2012.] http://www.swi-prolog.org/.

18. Netbeans. [En línea] http://netbeans.org/community/releases/72/index_es.html.

19. IBM Rational Unified Process (RUP). [En línea] http://www-01.ibm.com/software/awdtools/rup/.

20. PostgreSQL-es. *Portal en español sobre PostgreSQL.* [En línea] [Citado el: 5 de diciembre de 2012.] http://www.postgresql.org.es/sobre_postgresql.

21. Visual Paradigm for UML. *UML tool for software application development.* [En línea] http://www.visual-paradigm.com/product/vpuml/.

22. *Curso Modelado de Negocio con BPMN 2.0 y UML.* 2011.

23. al., Frank Buschmann et. *Pattern-Oriented Software Architecture. A System of Patterns.* s.l. : John Wiley & Sons. Vol. 1. 0471958697.

24. Superior, Ministerio de Educación. RESOLUCION Nº 128 / 2006. *REGLAMENTO PARA LA APLICACIÓN DE LAS CATEGORÍAS DOCENTES DE LA EDUCACION SUPERIOR.* La Habana : s.n., 2006.

25. Cuba, Consejo de Estado de la República de. DECRETO LEY NO. 133 DE 8 DE MAYO DE 1992 SOBRE GRADOS CIENTIFICOS. La Habana : s.n., 1992.

26. ¿Qué es UML? *¿Qué es UML?* [En línea] 2010. [Citado el: 15 de enero de 2013.] http://www.docirs.cl/uml.htm.

27. *Programación Basada en Restricciones: La Programación Simbólica al Servicio del Control de Producción Inteligente.* Antonio Morón, Francisco Sastrón. 010, Valencia : s.n., 4.

28. PostgreSQL. [En línea] [Citado el: 3 de 12 de 2012.] http://www.postgresql.org/about.

29. Writing JUnit Tests in NetBeans IDE. [En línea] [Citado el: 26 de abril de 2013.] https://netbeans.org/kb/docs/java/junit-intro.html.

30. Feo, Thomas A. y Resende, Mauricio GC. *Greedy randomized adaptive search procedures.* Netherland : Journal of global optimization, 1995.

31. Díaz, Francisco Javier Rodríguez. *Hybrid and Constructive Metaheuristics: Methods and Application.* Granada : Editorial de la Universidad de Granada, 2012. 978-84-9028-453-7.

BIBLIOGRAFÍA

1. **Superior, Ministerio de Educación.** RESOLUCIÓN No. 210 del 2007. *Reglamento para el Trabajo Docente y Metodológico.* La Habana : s.n., 2006.

2. **Alvarez de Zayas, C.** *Hacia una escuela de excelencia.* La Habana : Editorial Academia, 1996. pág. 3.

3. Conferencia PDE IPLAC. Impresión ligera. . Ciudad de La Habana : s.n., 1995.

4. WordReference.Online Language Dictionaries. *Spanish Definition from RAE.* [En línea] 2013. http://www.wordreference.com/es/en/frames.asp?es=tribunal.

5. **Mathematical Programming Glossary, INFORMS Computing Society.** *The Nature of Mathematical Programming .*

6. **Martí, Rafael.** Procedimientos Metaheurísticos en Optimización Combinatoria. Valencia : s.n.

7. **Enrique Castillo, Antonio J. Conejo, Pablo Pedregal, Ricardo García y Natalia Alguacil.** *Formulación y Resolución de Modelos de Programación Matemática en Ingeniería y Ciencia.* 2002.

8. **Coppin, Ben.** *Artificial Intelligence Illuminated.* 1. Sudbury : Jones and Bartlett Publishers, 2004. pág. 768. 0-7637-3230-3.

9. **Díaz, Adenso, et al.** *Optimización Heurística y Redes Neuronales.* España : Editorial Paraninfo, 1996.

10. **Osman, I. H. and Kelly J. P.** *Meta-Heuristics: theory and applications.* Boston : Kluwer Academic, 1996. págs. 1-21.

11. **Fernando Borrás Rocher, José Vicente Segura Heras, Joaquín Sánchez Soriano, Manuel A. Pulido Cayuela, Jesús Tadeo Pastor Ciurana.** Confección automática de tribunales para las pruebas de acceso a la universidad. Ubeda : XXVI Congreso Nacional de Estadística e Investigación Operativa: Úbeda, 6-9 de noviembre de 2001, 2001, ISBN 84-8439-080-2, 2001.

12. **Facundo E. Cancelo, Pablo N. Cababie, Gabriel Barrera, Daniela López De Luise.** *Un nuevo enfoque para asignación óptima de múltiples recursos.* Buenos Aires, Argentina : Universidad de Palermo.

13. **Jorge Alberto Ospina Falla, Rafael Ricaurte Bernal.** *Solución de un problema de asignación de horarios de mantenimiento en máquinas, equipos e instalaciones a la empresa Powel Continental Ltda. mediante la programación de un algoritmo genético.* Chia, Colombia : s.n., 2004.

14. IBM. *IBM.* [En línea] 2013. [Citado el: 16 de 1 de 2013.] http://www.ibm.com/software/rational/uml/.

15. **Largo, Faraón LLorens.** Prácicas Prolog. Alicante : s.n.

16. Java. *Learn About Java Technology.* [En línea] [Citado el: 10 de enero de 2013.] http://www.java.com/en/about/.

17. SWI-Prolog's Home. [En línea] [Citado el: 15 de diciembre de 2012.] http://www.swi-prolog.org/.

18. Netbeans. [En línea] http://netbeans.org/community/releases/72/index_es.html.

19. IBM Rational Unified Process (RUP). [En línea] http://www-01.ibm.com/software/awdtools/rup/.

20. PostgreSQL-es. *Portal en español sobre PostgreSQL.* [En línea] [Citado el: 5 de diciembre de 2012.] http://www.postgresql.org.es/sobre_postgresql.

21. Visual Paradigm for UML. *UML tool for software application development.* [En línea] http://www.visual-paradigm.com/product/vpuml/.

22. *Curso Modelado de Negocio con BPMN 2.0 y UML.* 2011.

23. **al., Frank Buschmann et.** *Pattern-Oriented Software Architecture. A System of Patterns.* s.l. : John Wiley & Sons. Vol. 1. 0471958697.

24. **Alexander, Christopher, ISHIKAWA, Sara y SILVERSTEIN, Murray.** *A pattern language: towns, buildings, construction. .* s.l. : Oxford University Press, 1977.

25. **Superior, Ministerio de Educación.** RESOLUCION Nº 128 / 2006. *REGLAMENTO PARA LA APLICACIÓN DE LAS CATEGORÍAS DOCENTES DE LA EDUCACION SUPERIOR.* La Habana : s.n., 2006.

26. **Cuba, Consejo de Estado de la República de.** DECRETO LEY NO. 133 DE 8 DE MAYO DE 1992 SOBRE GRADOS CIENTIFICOS. La Habana : s.n., 1992.

27. ¿Qué es UML? *¿Qué es UML?* [En línea] 2010. [Citado el: 15 de enero de 2013.] http://www.docirs.cl/uml.htm.

28. *Programación Basada en Restricciones: La Programación Simbólica al Servicio del Control de Producción Inteligente.* **Antonio Morón, Francisco Sastrón.** 010, Valencia : s.n., 4.

29. PostgreSQL. [En línea] [Citado el: 3 de 12 de 2012.] http://www.postgresql.org/about.

30. Writing JUnit Tests in NetBeans IDE. [En línea] [Citado el: 26 de abril de 2013.] https://netbeans.org/kb/docs/java/junit-intro.html.

31. **Feo, Thomas A. y Resende, Mauricio GC.** *Greedy randomized adaptive search procedures.* Netherland : Journal of global optimization, 1995.

32. **Díaz, Francisco Javier Rodríguez.** *Hybrid and Constructive Metaheuristics: Methods and Application.* Granada : Editorial de la Universidad de Granada, 2012. 978-84-9028-453-7.

ANEXOS

Tabla 7. Descripción del CU Autenticar Usuario

Caso de uso	
CU_2	Autenticar usuario
Propósito	Permitir al usuario una vez autenticado el acceso al sistema.
Actores	Vicedecana de Formación
Prioridad	Crítico.
Resumen	El caso de uso se inicia cuando la Vicedecana de Formación inicia el sistema, que le muestra la ventana de autenticación para poder interactuar con el mismo.
Precondiciones	-
Referencias	RF1
Acción del actor	**Respuesta del sistema**
1. Inicia el sistema.	2. Muestra una ventana de autenticación con los campos a completar: • Usuario • Contraseña
3. Introduce el usuario y la contraseña y presiona “Aceptar”. En caso de presionar “Cancelar” ver **Flujo alterno 3a. Opción Cancelar**	4. El sistema compara los datos introducidos con los existentes en la base de datos. En caso de ser incorrectos los datos ver **Flujo alterno 4a. Datos Incorrectos**
	5. Le da acceso al usuario al resto de las funcionalidades.
Flujo alterno 3a. Opción Cancelar	
	3a 1. Se cierra el sistema.
Flujo alterno 4a. Datos Incorrectos	

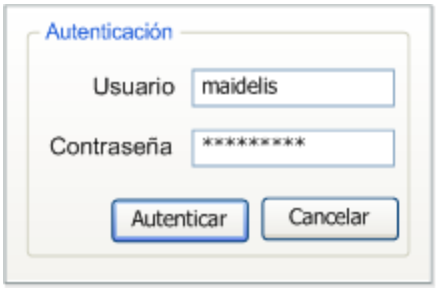
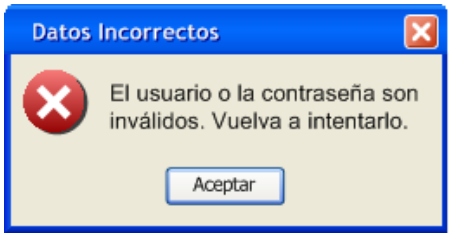
	4a. 1. Muestra un mensaje de error notificando que los datos son incorrectos. 4a. 2. Vuelve al paso 3.
Prototipos  Figura 22. Autenticar usuario  Figura 23. Datos incorrectos	

Tabla 8. Descripción del CU Adicionar Profesor

Caso de uso	
CU_3	Adicionar profesor
Propósito	Adicionar nuevos profesores al sistema
Actores	Vicedecana de Formación
Prioridad	Crítico.
Resumen	El caso de uso se inicia cuando la Vicedecana de Formación selecciona la opción “Adicionar profesor” del menú lateral de la aplicación.
Precondiciones	El usuario debe estar autenticado en el

	sistema.
Referencias	RF1, RF2
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción “Adicionar profesor”.	2. Muestra los campos a completar: • Nombre • Categoría Docente • Categoría Científica • Rol • Centro • Área del conocimiento • Nivel de experticia • Años
3. Completa los campos y presiona “Asignar”,	4. El sistema comprueba que no haya campos vacíos. Si los datos están incompletos ir al **Flujo alterno 4a. Datos Incompletos.**
	5. El sistema comprueba que el área de conocimiento insertada no esté repetida. En caso de aparecer en el listado de Áreas asignadas ir a **Flujo alterno 5a. Áreas repetidas.**
6. Presiona “Adicionar”.	7. El sistema valida los datos insertados. • El nombre solo debe tener letras. • Para cada área del conocimiento asignada la cantidad de años debe ser numérica y estar entre 1 y 60. En caso de haber algún campo incorrecto ver **Flujo alterno 7a. Datos incorrectos.**
	8. Comprueba que el profesor no haya sido insertado previamente.

	En caso de estar en la base de datos ver **Flujo alterno 8a. Profesor insertado**
	9. el sistema adiciona los datos del profesor a la base de datos.
Flujo alterno 4a. Datos Incompletos	
	4a 1. Muestra un mensaje de error notificando que los datos están incompletos. Volver al paso 3.
Flujo alterno 5a. Áreas repetidas.	
	5a. 1. Muestra un mensaje de error indicando que el área ha sido asignada anteriormente al profesor. Volver al paso 3.
Flujo alterno 7a. Datos incorrectos.	
	7a. 1. Muestra un mensaje indicando qué datos son incorrectos. Volver al paso 3.
Flujo alterno 8a. Profesor insertado	
	8a. 1. Muestra un mensaje indicando que el profesor existe en la base de datos. Volver al paso 3.
Prototipo	

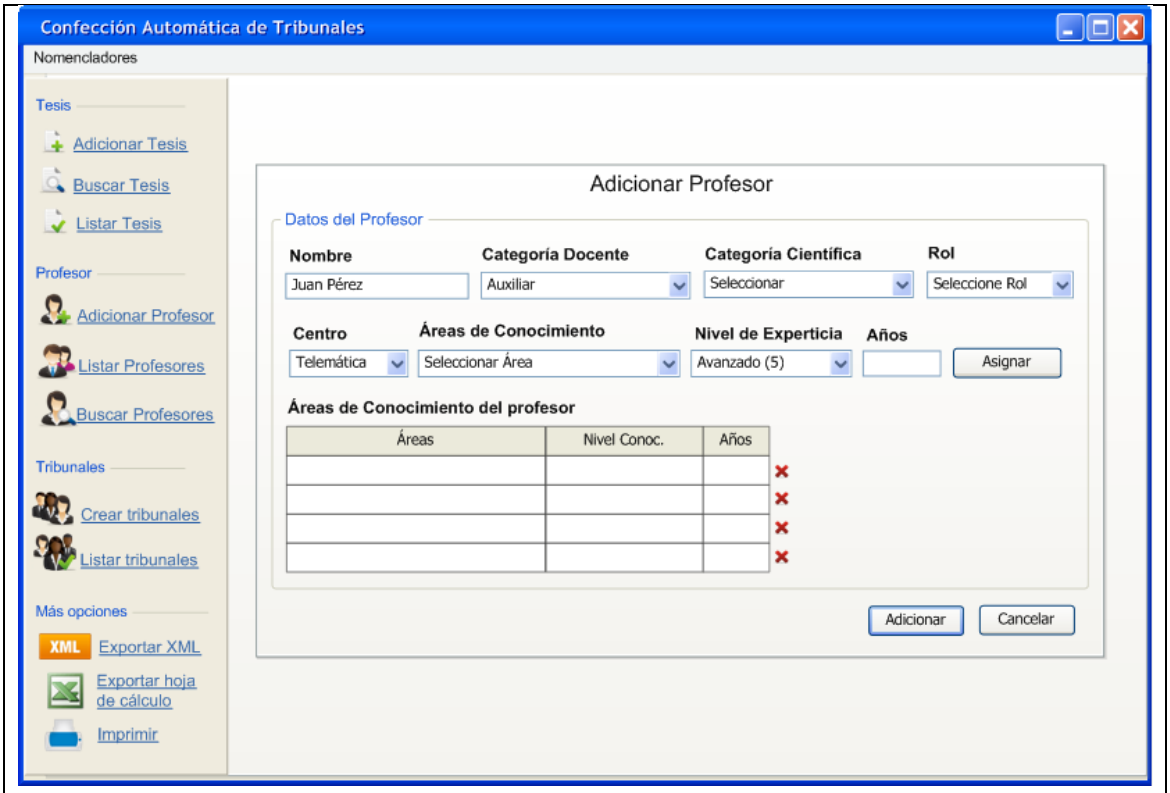


Figura 24. Adicionar **profesor**