

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD DE CIENCIAS Y TECNOLOGÍAS
COMPUTACIONALES



SISTEMA DE GESTIÓN DEL CALENDARIO DE TESIS DE GRADO EN
LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

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

Autor: Idel González Sierra

Tutores:

MSc. Manuel Enrique Puebla Martínez

Ing. Yoel Rojo Corrada

La Habana, Junio 2017

“Año 59 de la Revolución”



"En la tierra hace falta personas que trabajen más y critiquen menos, que construyan más y destruyan menos, que prometan menos y resuelvan más que esperen recibir menos y dar más que digan mejor ahora que mañana."

Ernesto "Che" Guevara

DECLARACIÓN DE AUTORÍA

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

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

Tutor: MSc. Manuel Enrique Puebla Martínez

Autor: Idel González Sierra

Tutor: Ing. Yoel Rojo Corrada

DATOS DEL TUTOR

Nombre y apellidos: Manuel Enrique Puebla Martínez

Correo electrónico: mpuebla@uci.cu

Institución: Universidad de las Ciencias Informáticas

Breve descripción: Licenciado en ciencia de la computación de la Universidad de Oriente en el año 2004. Profesor asistente del departamento de programación de la Facultad de Ciencias y Tecnologías Computacionales de la Universidad de las Ciencias Informáticas. En el año 2008 defendió su tesis de maestría en Nuevas Tecnologías de la Educación. Desde el 2010 está vinculado a la línea de investigación Semántica Espacial. Pertenece al grupo de investigación de web semántica de la UCI (<http://gws-uci.blogspot.com/>) y desarrolla su investigación doctoral en Recuperación de Información Geográfica GIR (por sus siglas en inglés).

DATOS DEL TUTOR

Nombre y apellidos: Yoel Rojo Corrada

Correo electrónico: yrojo@uci.cu

Institución: Universidad de las Ciencias Informáticas

Breve descripción: Ingeniero en ciencias informáticas de la Universidad de las Ciencias Informáticas en el año 2016. Actualmente se desempeña como profesor de la asignatura matemática discreta en la Facultad Introdutoria de las Ciencias Informáticas.

DATOS DEL AUTOR

Nombre y apellidos: Idel González Sierra

Correo electrónico: isierra@estudiantes.uci.cu

Breve descripción:

Estudiante perteneciente al centro GEYSED en la Facultad de Ciencias y Tecnologías Computacionales de la Universidad de las Ciencias Informáticas.

Agradecimientos

A mis padres por haberme apoyado siempre.

A mi familia por ayudarme y formar mis valores.

A mis tutores y amigos Mauel y Yoel que sin su apoyo no lo hubiera logrado.

A la UCI por ser mi casa durante cinco años.

A todos los profesores de la carrera que de alguna manera han influido en mi formación.

Al profesor Enier por su ayuda y su amistad incondicional.

A mis compañeros de curso durante estos cinco años.

A Betty por todo su cariño.

A todos los que una vez creyeron en mí y me ayudaron cuando más lo necesitaba.

A todos de verdad.

Muchas gracias.

Dedicatoria

A mis padres...

RESUMEN

En la Universidad de las Ciencias Informáticas en los meses de mayo a julio se realiza el proceso de defensa de tesis de grado por parte de los estudiantes de quinto año de cada una de las facultades. Dicho proceso, debido a su marcada importancia es rigurosamente planificado con el objetivo de que se realice con la mayor calidad posible. En la planificación intervienen diferentes recursos como son locales, horarios, profesores, tribunales, estudiantes, tesis y un grupo de información relacionada con estos que en la mayoría de las ocasiones tiende a ser voluminoso. El proceso que hoy en día se realiza manualmente resulta tedioso y costoso en cuanto a tiempo laboral. Además estos recursos poseen diferentes restricciones a la hora de planificarlos, en múltiples ocasiones han coincidido actividades en las que interviene un mismo recurso por lo que se ha tenido que planificar nuevamente. En la presente investigación se describe una aplicación web que tiene como objetivo favorecer la gestión del Calendario de tesis de grado en la Universidad de las Ciencias Informáticas. Posee entre otras funcionalidades la gestión de tribunales, gestión del calendario, y la posibilidad de exportar dicho calendario en formato PDF. Obteniéndose un sistema funcional que reúne las condiciones para facilitar el desarrollo de dicho proceso y la documentación ingenieril asociada al mismo; validado mediante pruebas de caja blanca, caja negra y con una fundamentación del cumplimiento del objetivo general.

Palabras Clave: Asignación de recursos, Calendario de Tesis, Problemas de Horario.

ABSTRACT

At the University of Informatics Sciences the process of defense of degree theses is carried out by the fifth year students of each of the faculties in the months of May to July. This process, due to its remarkable importance, is rigorously planned with the aim of attaining the highest possible quality. The planning involves different resources such as venues, times, teachers, boards of examiners, students, theses, and information -that tend to be voluminous- related with them. The process, that is done manually today, is tedious and expensive in terms of working time. In addition, these resources have different restrictions when planning them and in many occasions the coincidence of different activities in which the same resource intervenes results in unnecessary re-planning. The present research describes a web application that aims to favor the management of the degree theses calendar at the University of Informatics Sciences. It has among other functions the management of boards of examiners, calendar management, and the possibility of exporting the calendar in PDF format, obtaining a functional system that meets the conditions to facilitate the development of the above mentioned process and the associated engineering documentation; validated by means of tests of white box, tests of black box and a rationale of the fulfillment of the general objective.

Key Words: Assignment of resources, Thesis Calendar, Schedule Problems.

Contenido

RESUMEN	V
ABSTRACT	VI
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS.....	X
INTRODUCCIÓN	1
CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA	5
1.1 INTRODUCCIÓN	5
1.2 MARCO TEÓRICO	5
<i>Proceso de Tesis</i>	<i>5</i>
<i>Conceptos asociados al dominio del problema</i>	<i>6</i>
<i>Sistemas de Gestión.....</i>	<i>6</i>
<i>Calendario o cronograma de tesis</i>	<i>6</i>
<i>Problemas de horario</i>	<i>6</i>
<i>Solución a los Problemas de Asignación de Horarios</i>	<i>7</i>
1.3 ESTADO DEL ARTE.....	10
<i>Análisis de soluciones existentes</i>	<i>10</i>
<i>Análisis de: Sistema de Gestión de Tesis Facultad 2 (UCI)</i>	<i>10</i>
<i>Akademios: Análisis y Diseño del módulo de Gestión de Tesis</i>	<i>11</i>
<i>Sistema de Planificación de Actividades (SIPAC)</i>	<i>12</i>
<i>Sistemas de Gestión de Calendario de Tesis.....</i>	<i>13</i>
<i>FET Generador de Horarios</i>	<i>14</i>
1.4 HERRAMIENTAS Y TECNOLOGÍAS A UTILIZAR PARA EL DESARROLLO DEL SISTEMA.....	15
1.4.1 METODOLOGÍAS DE DESARROLLO.....	15
<i>Agile Unified Process (AUP) o Proceso Unificado Ágil</i>	<i>15</i>
<i>Descripción de los flujos de trabajo ingenieriles.....</i>	<i>16</i>
1.4.2 MARCO DE TRABAJO.....	17
1.4.3 SERVIDOR WEB: APACHE.....	18
1.4.4 UML 2.0 COMO LENGUAJE UNIFICADO DE MODELADO	18
1.4.5 SISTEMAS GESTORES DE BASES DE DATOS	19
1.4.6 NETBEANS COMO IDE	19
1.4.7 VISUAL PARADIGM COMO HERRAMIENTA CASE	20
1.5 LENGUAJES DE PROGRAMACIÓN.....	20
<i>Lenguajes web de programación del lado del cliente</i>	<i>20</i>
<i>Lenguajes web de programación del lado del servidor</i>	<i>21</i>
1.6 CONCLUSIONES PARCIALES.....	21
CAPÍTULO 2. PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA.....	23
2.1 INTRODUCCIÓN.....	23
2.2 MODELO CONCEPTUAL EN EL DESARROLLO DEL SOFTWARE.....	23

2.2.1	<i>Descripción del Modelo Conceptual</i>	24
2.2.2	<i>Glosario de Términos del Modelo del Dominio</i>	24
2.3	MODELO DEL NEGOCIO	24
2.3.1	<i>Actores del negocio</i>	24
2.3.2	<i>Diagrama de casos de uso del negocio</i>	25
2.3.3	<i>Descripción de los casos de uso del negocio</i>	25
2.4	MODELO DEL SISTEMA.....	26
2.4.1	<i>Requisitos</i>	26
2.4.2	<i>Estrategia de captura de requisitos</i>	26
2.4.3	<i>Listado de requisitos funcionales</i>	27
2.4.4	<i>Requisitos No Funcionales</i>	28
2.5	DEFINICIÓN DE CASOS DE USO DEL SISTEMA	30
2.5.1	<i>Descripción de los actores que interactúan con el sistema</i>	30
2.5.2	<i>Identificación de los casos de uso</i>	31
2.5.3	<i>Diagrama de Casos de uso del sistema</i>	31
2.5.4	<i>Descripción textual de los casos de uso del sistema</i>	32
2.5.5	<i>Patrones de Casos de Uso</i>	36
2.6	ARQUITECTURA DE SOFTWARE	36
2.6.1	<i>Patrones de arquitectura</i>	37
2.7	MODELO DE DISEÑO.....	38
2.7.1	<i>Diagrama de paquetes</i>	38
2.7.2	<i>Diagrama de clases del diseño</i>	40
2.7.3	<i>Patrones de Diseño</i>	41
2.8	DISEÑO DE LA BASE DE DATOS	43
2.9	MÉTODO DE SOLUCIÓN.....	44
2.10	CONCLUSIONES PARCIALES.....	47
CAPÍTULO 3. CONSTRUCCIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA		48
3.1	INTRODUCCIÓN	48
3.2	MODELO DE IMPLEMENTACIÓN.....	48
3.2.1.	<i>Diagrama de componentes</i>	48
3.3	MODELO DE DESPLIEGUE.....	49
3.4	DESCRIPCIÓN DE LAS PRUEBAS	50
3.4.1.	<i>Pruebas de software aplicadas</i>	52
3.4.2.	<i>Diseño de las pruebas para validar la solución propuesta</i>	52
3.4.3.	<i>Resultados de las pruebas realizadas</i>	56
3.5	OTRAS PRUEBAS REALIZADAS	57
3.6	FUNDAMENTACIÓN DEL CUMPLIMIENTO DEL OBJETIVO GENERAL	59
3.7	CONCLUSIONES PARCIALES.....	60
CONCLUSIONES GENERALES		61
RECOMENDACIONES		62
REFERENCIAS BIBLIOGRÁFICAS		63

ANEXOS66

ÍNDICE DE FIGURAS

Fig. 1: Sistema de gestión de tesis Facultad2 11

Fig. 2: Sistema Akademos..... 12

Fig. 3: Sistema de Planificación de Actividades..... 13

Fig. 4: Generador de horarios..... 15

Fig. 5: Ciclo de vida del proceso unificado ágil (AUP, 2015) 16

Fig. 6: Modelo Conceptual 23

Fig. 7: Diagrama de casos de uso del negocio 25

Fig. 8 : Diagrama de Casos de Uso del Sistema 32

Fig. 9: Diagrama de Paquetes 39

Fig. 10: Diagrama de Clases del Diseño Gestionar Tribunales 40

Fig. 11: Diagrama Relacional 44

Fig. 12: Algoritmo de generación de tribunales..... 45

Fig. 13: Diagrama de Componentes Gestionar Profesor 49

Fig. 14: Diagrama de Despliegue 50

Fig. 15: Resultado de las Pruebas 56

Fig. 16: Grafo del flujo asociado a la funcionalidad createAction 58

ÍNDICE DE TABLAS

Tabla 1: Descripción de actores del Negocio.....	25
Tabla 2: Descripción del Caso de Uso Generar Tribunales	25
Tabla 3: Descripción del Caso de Uso Generar Calendario.....	26
Tabla 4: Descripción de Actores del Sistema	30
Tabla 5: Requisitos Funcionales y Casos de Uso.....	31
Tabla 6: Descripción del Caso de Uso “Gestionar Tesis”	32
Tabla 7: Tabla de ponderaciones	46
Tabla 8: Caso de Prueba: “Gestionar Estudiante”	53
Tabla 9: Caso de Prueba: “Gestionar Estudiante”	55
Tabla 10: Caminos básicos	58
Tabla 11: Camino básicos 1	59
Tabla 12: Camino básico 2	59
Tabla 13: No conformidades.....	67

INTRODUCCIÓN

Según la Real Academia de la Lengua Española, la planificación constituye un plan general, metódicamente organizado y frecuentemente de gran amplitud, para obtener un objetivo determinado (Rae, 2017), por lo que no debe extrañar que usualmente todo objetivo que amerite ser alcanzado conlleve un proceso de planificación previo. Desde que el hombre se convirtió en un ser racional, utiliza gran cantidad de recursos y tiempo en la realización de esta actividad con el objetivo de alcanzar sus metas propuestas.

Las Tecnologías de la Informática y las Comunicaciones (TIC) tienen, día a día, una mayor presencia en todos los aspectos de la vida laboral y personal, ofreciendo un nuevo espacio de innovación en áreas como la industria, los servicios, la salud, la administración, el comercio y la educación. Poco a poco han logrado facilitar la coordinación entre horarios, eventos, logística y otros aspectos que conlleva la planificación de actividades. En el área de la educación el flujo de información que se debe controlar es voluminoso, sobre todo en la educación superior, enseñanza que culmina con la formación de un profesional.

Al igual la que en mayoría de las universidades cubanas, en la UCI todos los años entre los meses de mayo y julio los estudiantes deben defender un trabajo de diploma como ejercicio de culminación de estudios con el propósito de demostrar las habilidades adquiridas en la carrera. El proceso de tesis, como todo proceso docente, es supervisado por el vicerrector de formación y a la vez por los vicedecanos de formación en cada una de las facultades. Con antelación a los actos de defensa de tesis, los vicedecanos con el apoyo del profesor principal del año y los directivos de los centros de desarrollo de software deben planificar y organizar dichos actos en cada facultad; con el objetivo de que los profesionales que forman los tribunales no tengan afectaciones laborales o personales que impidan su participación. El proceso consiste en la confección de un calendario teniendo en cuenta la disponibilidad de los locales, la cantidad de tesis a planificar, el tribunal asociado a las mismas y los horarios.

Cada tribunal está conformado por un presidente, un secretario y un vocal. Para la generación de los tribunales se recopilan datos relacionados con los profesores pues es necesario determinar quiénes son los candidatos más idóneos para cada rol. Cada tribunal es responsable de evaluar la calidad de un grupo de tesis que le son asignadas.

Cada una de las tesis está elaborada por estudiantes, tutorados por profesores, y a cada tesis se le asigna un oponente que no debe coincidir con los tutores. Los profesores pueden ser tutores de más de una tesis y estas a su vez poseer varios tutores.

Actualmente la conformación del calendario de tesis se realiza de forma manual por lo que es propensa a poseer errores que atentan contra la calidad del proceso. En ocasiones los profesionales implicados en el acto tienen más de una responsabilidad en dicho proceso y pudieran coincidir en el calendario por lo que se debe volver a planificar. Otro problema frecuente ocurre cuando es necesario sustituir un profesor por otro pues hay que velar que este no tenga responsabilidades que le puedan coincidir dentro del proceso. Esta planificación repetitiva puede resultar agotadora sobre todo cuando hay un gran volumen de datos.

La información asociada a la generación del calendario se realiza en una plantilla Excel, por lo que no posee ningún autocompletado automático ni ninguna verificación de condiciones como las antes mencionadas, por lo que los datos deben introducirse con mucho cuidado y atención.

De lo expuesto anteriormente, se identificó el siguiente **problema de la investigación**:

¿Cómo favorecer la gestión del calendario de tesis de grado en la Universidad de las Ciencias Informáticas?

Con el fin de dar solución al problema propuesto se define como **objeto de estudio** informatización problemas de asignación de recursos y como **campo de acción** informatización los problemas de horarios. Como **objetivo general** se define, desarrollar una herramienta de software que favorezca la gestión del Calendario de tesis de grado en la Universidad de las Ciencias Informáticas.

Para darle solución al objetivo de la investigación se le dará respuesta a las siguientes **preguntas científicas**:

1. ¿Qué factores afectan gestión del calendario de tesis en la Universidad de las Ciencias Informáticas?
2. ¿Cómo influyen estos factores en dicha gestión?

3. ¿Qué factores Pudieran favorecer la gestión del calendario de tesis?
4. ¿Cuáles serían las tecnologías informáticas adecuadas para favorecer la gestión del calendario de tesis?
5. ¿Cómo validar que la herramienta de software a desarrollar favorecerá la gestión del calendario de tesis?

Con el fin de darle cumplimiento al objetivo general y al problema anteriormente planteado se trazaron las siguientes **tareas de la investigación**:

- ✓ Identificación de los factores afectan gestión del calendario de tesis en la Universidad de las Ciencias Informáticas
- ✓ Evaluación del nivel de influencia de los factores afectan gestión del calendario de tesis en la Universidad de las Ciencias Informáticas
- ✓ Identificación de los factores Pudieran favorecer la gestión del calendario de tesis
- ✓ Identificación de las tecnologías informáticas adecuadas para favorecer la gestión del calendario de tesis
- ✓ Demostrar que la herramienta de software a desarrollar favorecerá la gestión del calendario de tesis

A lo largo del desarrollo de la investigación se utilizan un conjunto de **métodos científicos** como son:

Los métodos teóricos:

1. **Analítico-Sintético**: Se utilizó en el análisis de las teorías, documentos, y materiales, de diferentes autores con el fin de extraer los elementos más importantes de las bibliografías especializadas en los problemas de asignación de recursos.
2. **Modelación**: Se empleó en la elaboración de los artefactos generados como resultado del proceso de ingeniería de software.

Los métodos empíricos:

1. **Entrevista**: Se empleó la entrevista individual no estructurada en la fase inicial para el entendimiento del negocio y levantamiento de requisitos. Se realizaron una serie de preguntas a las personas con dominio de la problemática en función de obtener la mayor cantidad de información posible referente al proceso de conformación de tribunales y la creación del calendario. Ver anexo 1.

El Trabajo de investigación está estructurado de la siguiente manera: introducción, tres capítulos, conclusiones generales, recomendaciones, referencias bibliográficas y anexos:

Capítulo #1. Fundamentación teórica: Recoge en el marco teórico aspectos concernientes al objeto de estudio, se hace una revisión de la bibliografía donde se analizan los principales conceptos. Además de una análisis crítico del estado del arte de la investigación. Además se reúnen todas las tecnologías y herramientas a utilizar en la solución propuesta.

Capítulo #2. Presentación de la solución propuesta: Contiene información asociada al sistema, descripción del modelo de dominio, especificación de los requisitos de software y definición de los casos de uso, incluye el diseño de la solución propuesta, la arquitectura utilizad y la representación de la base de datos. Además se seleccionan las tecnologías y herramientas a utilizar para el desarrollo de la solución propuesta.

Capítulo #3. Construcción y validación de la solución propuesta: Contiene los diagramas de despliegue y de componentes y del flujo de trabajo de pruebas realizadas al sistema, con la descripción de los casos de prueba.

Con el cumplimiento de todas las tareas y actividades desarrolladas en el proceso de investigación se esperan los siguientes **resultados**:

- ✓ Un sistema que permita la gestión semiautomática del calendario de tesis para la Universidad de las Ciencias Informáticas.
- ✓ La documentación técnica del proceso ingenieril correspondiente al diseño e implementación del sistema.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo, se analizan los principales conceptos asociados al dominio del problema, con el propósito de brindar al lector una mayor comprensión del marco teórico de la investigación. Por otra parte, se realiza el análisis de sistemas y aplicaciones que realizan procesos similares al que se desea desarrollar, que servirán como punto de partida para el desarrollo de la solución.

1.2 Marco Teórico

Proceso de Tesis

En la resolución No. 210 del 2007 Ministro de Educación Superior se hace referencia a consideraciones legales que rigen el proceso de tesis. El estudio de esta legislación posibilita el esclarecimiento de las particularidades del proceso y una mejor comprensión del mismo. A continuación se referencia varios artículos que por su importancia es conveniente analizar:

ARTÍCULO 23: Para culminar la carrera, el estudiante deberá aprobar un ejercicio de culminación de los estudios, de acuerdo con lo establecido para cada plan de estudio. Se refiere al carácter obligatorio del trabajo de diploma u otra modalidad como requisito para obtener un título.

ARTÍCULO 28: El tutor desempeña un papel esencial en la formación integral del estudiante y tiene la responsabilidad de integrar el sistema de influencias educativas presentes en los distintos ámbitos de su desarrollo personal. Se refiere a la labor del tutor como formador y guía en proceso.

ARTÍCULO 123: El 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. Se refiere a la importancia de la realización de un trabajo de diploma.

ARTÍCULO 151: 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 profesión, con independencia y creatividad, de un problema propio de la profesión, utilizando la metodología de la investigación científica. En el caso de la UCI cada Facultad es responsable de crear tribunales para la calificación, quienes funcionarán centralmente y por Facultad.

Conceptos asociados al dominio del problema

A continuación se brindan varios conceptos que se consideran indispensables para el correcto entendimiento de la investigación realizada.

Sistemas de Gestión

Un sistema de gestión es una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de una organización. El término abarca un amplio campo, dado que la gestión y planificación son una necesidad desde los inicios de las organizaciones, empresas e instituciones de diversas índoles. Un sistema de este tipo puede ser concebido para múltiples propósitos según la esfera en que se desee implantar, o sea, puede gestionarse información referente a, por solo citar algunos, empresas, escuelas, institutos de investigación y datos de logística. (Diccionario, 2015).

Calendario o cronograma de tesis

El cronograma es una herramienta muy importante en la gestión de proyectos. Puede tratarse de un documento impreso o de una aplicación digital; en cualquier caso, el cronograma incluye una lista de actividades o tareas con las fechas previstas de su comienzo y final (Pérez Porto , y otros, 2014.). Entonces el cronograma de tesis constituye un listado con la planificación de las tesis.

Problemas de horario

El problema de la generación de horarios es uno de los clásicos de las ciencias de la computación. Desde la perspectiva de la Investigación de Operaciones, este tipo de problemas se enmarcan dentro del área conocida como *timetabling* o programación horaria. Los problemas de esta área consisten en la asignación de ciertos eventos a distintos bloques horarios respetando una serie de requerimientos y condiciones. Zhipeng Lu y Jin-Kao Hao, definen *timetabling* como: “Asignar un número de eventos, cada uno con ciertas características, a un número limitado de recursos sujeto a restricciones” (Zhipeng , y otros, 2010). Anterior a ellos Anthony Wren en 1996, determina el *timetabling*, como “la asignación, sujeta a restricciones, de un grupo de recursos a objetos ubicados en tiempo y espacio, de tal manera que se satisfagan un conjunto de objetivos deseados” (Wren, 1996).

Este particular problema se puede apreciar claramente en diferentes escenarios en el mundo, el transporte, la industria, la cultura, el deporte, entre otros; por lo que es objeto de estudio por parte de grupos de investigación, para optimizar los resultados y lograr soluciones automatizadas de alta calidad.

Dentro de estos problemas existe una rama específica, llamada *University or Course Timetabling*, que estudia problemas relacionados con la programación horaria para entidades educativas y más específicamente es recogido en la literatura *Examination Timetabling* referente a la calendarización de exámenes (Sarmiento, 2012).

Solución a los Problemas de Asignación de Horarios

Los Problemas de Asignación de Horarios (en inglés *timetabling problem*) pertenecen al área de la optimización combinatoria, donde cada horario representa una solución al problema. La optimización combinatoria se encuentra enfocada en la búsqueda de la mejor combinación, construyendo funciones sobre el espacio de las combinaciones que permitan determinar cuál es la mejor. El objetivo de estos problemas es encontrar el máximo (o el mínimo) de una determinada función sobre un conjunto finito de soluciones denotadas por S . El conjunto de soluciones S trabaja con variables discretas, restringiendo su dominio a valores finitos (ALFONSO, 2011).

Existen métodos que recorren todo el espacio de búsqueda, por tanto, se dice que encuentran todas las soluciones al problema, se les considera como algoritmos completos. Sin embargo estos métodos, dependen del número de variables que intervienen en el problema. (de Werra, 1985). En este Grupo se encuentran: programación lineal, programación entera, *backtracking*, entre otras.

Para describir los problemas de optimización de forma concisa es necesario desarrollar un modelo matemático. Si el modelo matemático de cualquier problema de optimización se ajusta al formato general del modelo de programación lineal, el problema es considerado entonces un problema de programación lineal (Boj del Val, y otros, 2012). El modelo matemático en los problemas de programación lineal se formula a partir de la cantidad de recursos disponibles, el conjunto de restricciones lineales para definir las soluciones admisibles y la función objetivo que debe ser optimizada (minimizada o maximizada) (Sarmiento, 2012).

Los problemas donde las variables que intervienen tienen valor entero reciben el nombre de problemas de Programación Entera. El modelo matemático es formulado como un modelo de programación lineal teniendo en cuenta que todas las variables que intervienen deben ser enteras (Programación lineal., 2012). Los problemas de optimización combinatoria presentan un espacio de soluciones elevado y la evaluación de todas sus soluciones para determinar el óptimo conllevan mucho tiempo computacional

(PASTOR MORENO, 2012). Por tal motivo se necesitan procedimientos que aseguren un menor tiempo computacional. Existen técnicas que son aplicadas a problemas de este tipo como es el caso de las heurísticas y las meta heurísticas (MARTÍ, 2005).

Los métodos heurísticos se utilizan para encontrar soluciones a problemas para los que no existe un algoritmo que converja a la solución ni una fórmula explícita que la encuentre; o que dichos algoritmos posean limitaciones computacionales en correspondencia con el problema a resolver. Un método heurístico es definido como: “un procedimiento para resolver un problema de optimización mediante una aproximación intuitiva, donde la estructura del problema se utiliza de forma inteligente para obtener una buena solución” (MARTÍ, 2003). Las heurísticas pueden converger en soluciones de baja calidad (óptimos locales muy lejanos al óptimo global). Para permitir una mejora adicional en la calidad de las soluciones han sido diseñadas técnicas de propósito general que guían la construcción de soluciones o la búsqueda local en las distintas heurísticas. Estas técnicas son conocidas por el término meta heurísticas (Potvin, y otros, 2005). Las técnicas meta heurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, donde las heurísticas no son efectivas. Las meta heurísticas proporcionan un marco general para crear nuevos algoritmos híbridos, combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos. Son aplicadas a algoritmos que dan solución a los problemas de optimización combinatoria (OSMAN, 1996).

Dentro de este grupo están: Recocido Simulado (*Simulated Annealing*), Algoritmos Evolutivos (*Evolutionary Algorithms*), búsqueda tabú (*Tabu Search*), algoritmos voraces, redes neuronales (*Neuronal Networks*), entre otras. Este tipo de métodos son conocidos como “meta heurísticos”.

Algoritmos ávidos (o voraces)

El método que produce algoritmos ávidos puede ser aplicado a numerosos problemas, especialmente los de optimización. Dado un problema con una cantidad de entradas el método consiste en obtener un subconjunto de estas que satisfaga una determinada restricción definida para el problema. Cada uno de los subconjuntos que cumplan las restricciones son soluciones prometedoras. Una solución prometedora que maximice o minimice una función objetivo constituye una solución óptima (Vallecillo, y otros, 2000).

Con estos elementos, se resume el funcionamiento de los algoritmos ávidos en los siguientes puntos. Para resolver el problema, un algoritmo ávido tratará de encontrar un subconjunto de candidatos tales que, cumpliendo las restricciones del problema, constituya la mejor solución. Para ello trabajará por

etapas, tomando en cada una de ellas la decisión que estime mejor, sin considerar las consecuencias futuras, y por tanto escogerá de entre todos los candidatos el que produce un óptimo local para esa etapa, suponiendo que será a su vez óptimo global para el problema. Antes de añadir un candidato a la solución que está construyendo comprobará si es prometedora al añadirlo. En caso afirmativo lo incluirá en ella y en caso contrario descartará este candidato para siempre y no volverá a considerarlo. Cada vez que se incluye un candidato comprobará si el conjunto obtenido es solución (Vallecillo, y otros, 2000). De los métodos estudiados es el que con una menor complejidad computacional obtiene un mejor resultado, por esos factores y por relativa rapidez en obtener resultados es idóneo para implementar en la solución de la problemática.

Búsqueda Tabú

La Búsqueda Tabú pertenece a la clase de técnicas de búsqueda local y fue diseñada para obtener una aproximación a la solución óptima de un problema de optimización combinatoria. El rendimiento del método de búsqueda local aumenta mediante el uso de estructuras de memoria que se incorporan en el procedimiento. La configuración inicial de la búsqueda puede ser obtenida aleatoriamente, por medio de un algoritmo constructivo o alguna técnica heurística que utilice índices de sensibilidad. Una configuración aleatoria puede tener la ventaja de evitar una convergencia prematura, pero una configuración inicial de mala calidad conduce a un esfuerzo computacional mayor. La principal desventaja de este método es el hecho de que se puede mover a una solución peor (Hocaoglu, y otros, 2007).

Recocido Simulado

El Recocido Simulado (en inglés *Simulated Annealing*) es una técnica para resolver problemas de optimización combinatoria. Los algoritmos de recocido pueden combinarse con otras técnicas heurísticas como: los sistemas expertos, los algoritmos genéticos, las redes neuronales, entre otros, consiguiendo sistemas híbridos que pueden resultar de muy eficientes en la resolución de problemas muy complejos. La principal desventaja que tiene esta técnica es que no guarda la información de movimientos previos para guiar los nuevos movimientos, pudiendo encontrar varias veces la misma solución. Para resolver problemas de gran complejidad es necesario realizar muchas ejecuciones para encontrar una solución satisfactoria. La solución final suele encontrarse en un mínimo local no explorando con suficiente amplitud el espacio de soluciones y una ejecución del algoritmo puede requerir mucho tiempo de cálculo (Nandhini, y otros, May 2009).

Método de la escalada máxima (en inglés *Hill Climbing*)

El método de la escalada máxima es un método iterativo de búsqueda local que es empleado en la solución de problemas de optimización. La técnica de Hill Climbing hace uso del cálculo de la función objetivo para la vecindad del punto actual y de esta forma determinar hacia donde crece la función, seleccionando el punto de mayor pendiente. Si el valor de la función objetivo es mejor en el punto nuevo, el punto anterior se reemplaza por el que fue hallado. Este proceso continúa hasta que no es posible encontrar ninguna mejora. Una de las desventajas de este método es su incapacidad para escapar de óptimos locales (Grosan, y otros, 2011).

Algoritmos Genéticos

Uno de los métodos más utilizados para la solución de problemas de asignación de horario son los Algoritmos Genéticos (AG). Los AG simulan el proceso de evolución natural, usando un conjunto de términos de las ciencias biológicas para un mejor entendimiento. El algoritmo se basa en cambios aleatorios de soluciones candidatas, utilizando la función objetivo para determinar si esos cambios producen una mejora al proceso. Para la solución se basan en operadores probabilísticos, en vez de los típicos operadores determinísticos de otras técnicas. La principal desventaja es que pueden converger prematuramente si se utilizan poblaciones pequeñas, donde una variación aleatoria en el ritmo de reproducción provoca que una solución sea dominante sobre las otras (Mejía Caballero, 2008).

1.3 Estado del Arte

Análisis de soluciones existentes

Diversas son las aplicaciones informáticas que tratan la problemática antes mencionada, siendo muy provechoso su análisis crítico. A continuación se detallan algunos sistemas, sus características principales y su aporte al desarrollo de la presente investigación.

Análisis de: Sistema de Gestión de Tesis Facultad 2 (UCI)

Aplicación web que permite optimizar el trabajo y la documentación referente al Proceso de Tesis de Grado en la Facultad 2, brindando la posibilidad de llevar un control durante el transcurso de cada período sobre cada una de las tesis y otros eventos que se efectúan como parte del proceso.

El sistema brinda una amplia información sobre todo el tema referente a las tesis de grado, garantiza la centralización de los documentos generados en torno a este proceso, permite publicar y asignar los temas de tesis propuestos a los estudiantes de quinto año y mantener un seguimiento de la evaluación y evolución de las mismas. El sistema pretende tener un control preciso del personal disponible para asumir tutorías, ni del estado evolutivo de los trabajos de tesis (Veranes, 2009).

El sistema presenta un avance pues pone de manifiesto una concepción teórica del proceso de tesis. Además, algunas funcionalidades de la aplicación como la centralización de la documentación referente al proceso, resultan de gran interés para la investigación, pero el sistema no se ajusta a las necesidades individuales de la solución a desarrollar pues no tiene en cuenta la generación automática de calendario de tesis.



Fig. 1: Sistema de gestión de tesis Facultad2

Akados: Análisis y Diseño del módulo de Gestión de Tesis

En el curso 2007 - 2008 se realizó en el proyecto Akados un trabajo de diploma relacionado con el análisis y diseño de un módulo para gestionar los procesos de tesis que se realizan en la universidad.

El sistema está concebido por módulos, entre los que se diferencian, los módulos de actualización de datos y el sitio web o módulo de información, a través del cual se muestran las diversas salidas de la aplicación.

Para el diseño del subsistema se utilizaron herramientas que respetan la política de migración a software libre de Cuba, tal es el caso de la herramienta de modelado Visual Paradigm, guiados por la metodología RUP y el lenguaje de modelado UML. También se tendrán en cuenta el lenguaje de programación PHP, y el Framework Symfony para elaborar un diseño superior al existente (Velázquez, 2008).

Ese sistema se implementó lo más genérico posible para que pudiera ser usado por todos los implicados en el proceso de tesis en cada una de las facultades. Gestiona un conjunto de información que puede ser usada por cualquier facultad, por tanto, hay muchas funcionalidades que sirven de base para la especificación de requisitos que se necesita en la presente investigación como son la gestión de los profesores, estudiantes y las tesis; pero aun así no resuelve el problema planteado, pues son necesarias otras funcionalidades como la generación de tribunales que no están contempladas en dicho diseño.

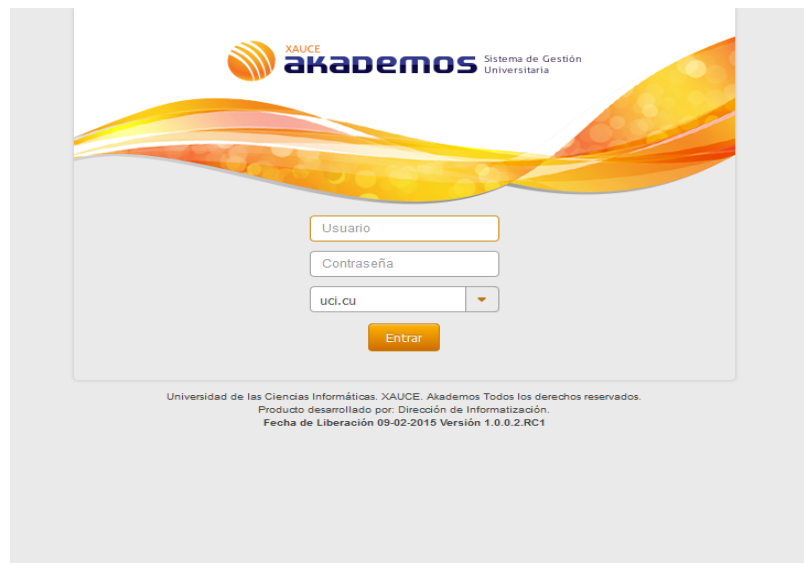


Fig. 2: Sistema Akademos

Sistema de Planificación de Actividades (SIPAC)

El SIPAC 2.1 es una aplicación web que permite interrelacionar objetivos de trabajo y actividades en tiempo real; garantizando el seguimiento y cumplimiento de los mismos en las entidades. Solución encaminada también a puntualizar las actividades que debe efectuar cada individuo, como parte de la planificación a corto plazo (denominada por los especialistas como Planificación Operativa), posibilita una mayor coincidencia entre lo que aspira la dirección y lo que debe proponerse cada miembro de la organización, garantizando que todo el personal se encuentre identificado con las actividades a desempeñar, las metas a alcanzar y las prioridades establecidas.



Fig. 3: Sistema de Planificación de Actividades

Resulta uno de los sistemas de planificación más completos desarrollado en la universidad por lo cual constituye un referente en esta materia. Es una solución genérica especializada en planificar eventos en entidades y planes de trabajo individuales. La problemática exige la implementación de otras funcionalidades con las que este sistema no cuenta como son la generación de tribunales y la generación del calendario de tesis.

Sistemas de Gestión de Calendario de Tesis

Sistema informático de planificación automática de pre-defensas y defensas de los trabajos de diploma en la Facultad 2.

Este software constituye un sistema informático para solucionar las inconsistencias que ha presentado el proceso de planificación de exposiciones de los trabajos de diploma en la Facultad 2 de la Universidad de las Ciencias Informáticas. Por tal motivo se realiza la investigación de heurísticas y meta heurísticas, seleccionando un algoritmo genético que pueda brindar una mejor solución en la confección del horario de exposiciones. Para la implementación del algoritmo es necesaria la incorporación del *framework* JGAP (en inglés *Java Genetic Algorithms Package*) el cual brinda un conjunto de mecanismos genéticos ya programados. Para validar la propuesta del algoritmo se desarrolló un sistema informático que permite la planificación del horario de exposiciones para las pre-defensas y defensas de los trabajos de diploma.

La planificación de las exposiciones de los trabajos de diploma por las características que presenta se clasifica como un problema de optimización combinatoria y puede modelarse como un problema de programación lineal. El espacio de soluciones que puede generar las combinaciones de salones, tribunales y horas disponibles para ambos es elevado. Dada la dificultad práctica para resolver de forma

exacta el problema, fue necesario comenzar un estudio de las principales técnicas meta heurísticas que proporcionan soluciones factibles (Lopez Diaz, 2013).

La herramienta al utilizar la Inteligencia Artificial y específicamente un algoritmo genético para el desarrollo de su solución le da un valor agregado y obtiene un resultado que pudiera considerarse óptimo por aplicación de técnicas meta heurísticas, a la par que lo complejiza en alguna medida. Sin embargo la solución está diseñada y desarrollada para necesidades específicas del proceso de tesis de la Facultad 2 en el 2013, tiene en cuenta que los locales están parcialmente utilizados por lo cual existen una serie de restricciones que no se ponen de manifiesto en el problema antes planteado asociado a la presente investigación, además la aplicación es de escritorio y no web; los requisitos y las condiciones en nuestro caso han variado.

FET Generador de Horarios

FET es una aplicación Software Libre y de código abierto para generar horarios académicos para cualquier tipo de institución. Utiliza un algoritmo de programación de horarios rápido y eficiente y está licenciado bajo GNU GPL. Por lo general, FET es capaz de resolver un calendario complicado en un máximo de 5 a 20 minutos para realizar los horarios más simples, pero para construir los horarios más complejos o con mayor cantidad de datos y restricciones, puede necesitar más tiempo de ejecución.

Una vez introducidos todos los datos en la aplicación basta con pedir que genere el horario, el cual se puede consultar dentro del mismo programa desde diferentes vistas: por profesor, por clase, por materia, etc. Al generarlo se informará de los “problemas” que tiene para generarlo o las condiciones que ha roto para hacerlo (ya que éstas pueden ser estrictas o tener la flexibilidad que se decida)

Además, genera un documento HTML¹, con todas las posibles vistas y compartir con el resto de usuarios del horario. Y no solo eso, el programa puede crear múltiples horarios para que se pueda elegir cuál es la mejor opción para el centro. (Baltolkien, 2017)

Esta herramienta está pensada para la generación de horarios genéricos y no valora algunas restricciones como las antes presentadas en la problemática. Resulta de utilidad su estudio sobre todo para el diseño de interfaces, aunque hay que considerar que es de escritorio.

¹ HyperText Markup Language (lenguaje de marcas de hipertexto)

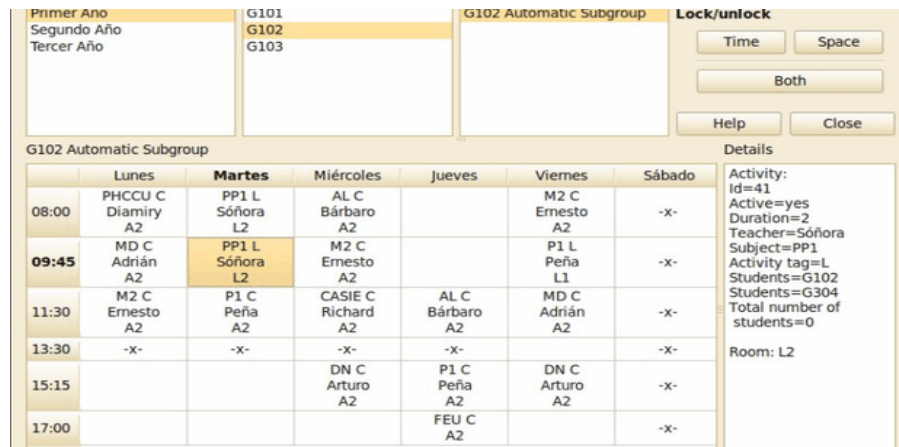


Fig. 4: Generador de horarios

1.4 Herramientas y tecnologías a utilizar para el desarrollo del sistema

En el presente epígrafe se justifica la elección de las tecnologías y herramientas necesarias para el desarrollo de esta tarea. La solución estará desarrollada con herramientas y tecnologías libres, y sustentada sobre tecnología Web.

1.4.1 Metodologías de desarrollo

En la actualidad la construcción de una aplicación informática que cumpla con los requerimientos planteados es una tarea intensa y difícil de cumplir. Las metodologías para el desarrollo del software imponen un proceso disciplinado sobre el desarrollo de los mismos con el fin de hacerlo más predecible y eficiente. Estas son usadas para estructurar, planificar y controlar el proceso de desarrollo. Estas garantizan el cumplimiento de la planificación de producción y tienen como objetivo guiar a los desarrolladores en la creación de un producto de calidad que cumpla con los requerimientos planteados por el cliente.

Agile Unified Process (AUP) o Proceso Unificado Ágil

Es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (AUP, 2015).

La metodología AUP (figura 5) es más simple que RUP porque reúne en una única disciplina las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de disciplinas

(Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP. Dispone de cuatro fases igual que RUP: Inicio, Elaboración, Construcción y Transición (AUP, 2015).

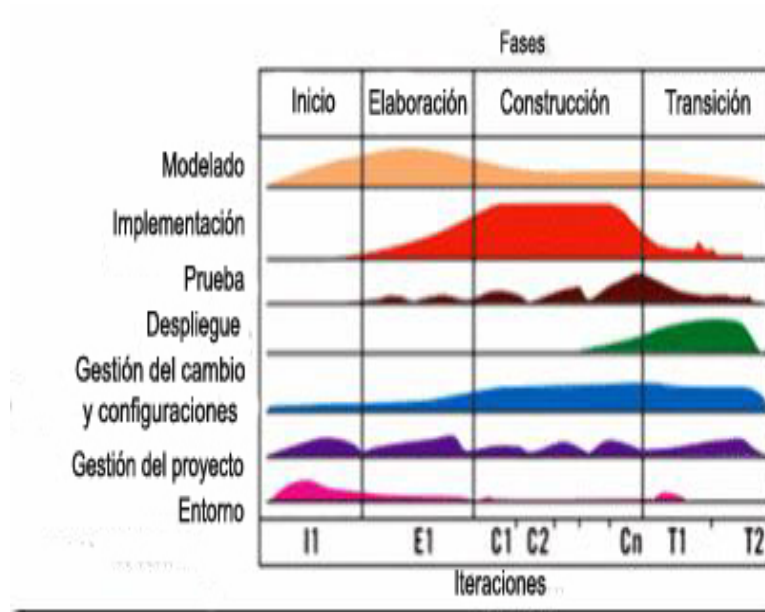


Fig. 5: Ciclo de vida del proceso unificado ágil (AUP, 2015)

Descripción de los flujos de trabajo ingenieriles

El modelado es el flujo de trabajo que tiene el objetivo de entender el negocio de la organización, el problema que se aborda en el proyecto y determinar una solución viable para resolver el problema de dominio. El flujo de trabajo Implementación tiene como objetivo transformar sus modelos en código ejecutable y realizar un nivel básico de las pruebas. El flujo de trabajo de Prueba tiene como objetivo realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, verificando que se cumplan los requerimientos. Por último dentro de los flujos de trabajo ingenieriles se tiene el Despliegue, cuyo objetivo es el plan para la prestación del sistema y la ejecución de dicho plan, para que el sistema quede a disposición de los usuarios finales (AUP, 2015).

Debido a que el proyecto se desarrolla con la participación del cliente, donde están especificados y bien definidos los requisitos, no posee una alta criticidad y el equipo de desarrollo es relativamente pequeño se decide la utilización de una metodología ágil. Además el proceso se recomienda que sea iterativo-incremental con el objetivo de garantizar la entrega temprana de partes operativas del software así como

un mayor grado de reutilización. La técnica basada en escenarios por casos de uso resulta recomendable para la descripción del negocio. Por estas características se decide utilizar la metodología AUP, como metodología de desarrollo, pues la misma es ideal para la guía del proceso de desarrollo de software sin añadir más trabajo que el desarrollo en sí, ya que opta por un paradigma de trabajo con entregables esenciales y específicos para el entendimiento de la solución final.

1.4.2 Marco de trabajo

La utilización de un marco de trabajo sin dudas agiliza en gran medida el proceso de desarrollo pues presenta una estructura o modelo que se puede ajustar a las necesidades del proyecto, facilidades en la generación de código, arquitectura, seguridad y otros elementos que sin duda favorecen el proceso de desarrollo.

Symfony V2.7 es un marco de trabajo PHP basado en la arquitectura Modelo-Vista-Controlador, que permite en teoría desarrollar aplicaciones de manera rápida, estructurada y comprensible. Symfony separa la lógica de negocio, la lógica del servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Además, automatiza las tareas más comunes, lo que le permite al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Lo positivo de todas estas ventajas es que no se debe realizar nuevamente algunos procesos cada vez que se crea una nueva aplicación Web (Potencier, 2008). Esta idea es una apuesta inteligente porque se reutilizan conceptos y desarrollos exitosos de terceros, integrados como librerías para ser utilizados para el desarrollo de la solución.

Un ejemplo concreto se evidencia en la capacidad de integrarse plenamente con uno de los ORM (*Object Relational Mapping*) más importantes dentro de los existentes para PHP llamado Doctrine. Este ORM es el encargado de la comunicación con la base de datos, brinda un control casi total de los datos sin importar si se utiliza MySQL, PostgreSQL, SQL Server, entre otros gestores porque la mayoría de las sentencias SQL (Structured Query Language) no son generadas por el programador sino por el mismo Doctrine.

Posee una gran cantidad de información y documentación, además de una comunidad de desarrolladores que brindan soporte asistencia y capacitación así como componentes para su reutilización. Por todo ello

además del dominio que posee el autor se elige Symfony2 como marco de trabajo más conveniente para el desarrollo de la solución.

1.4.3 Servidor Web: Apache

El desarrollo de una aplicación web permite un mayor flujo de información con la cual trabaja el sistema. Para ello es necesaria la utilización de un servidor web. Apache es un servidor Web de código abierto para plataformas Unix, Microsoft Windows, Macintosh, potente y estable, sencillo para las tareas más comunes de mantenimiento. Es altamente configurable, con gran robustez y estabilidad. Es un servidor seguro, eficiente y extensible que proporciona servicios HTTP² en sincronía con los estándares HTTP actuales (The Apache Software Apache, 2012).

Por las cualidades antes mencionadas que lo avalan como uno de los servidores más utilizados sobre todo por su facilidad de integrarse con otras tecnologías en servidores. Además por ser de código abierto y en correspondencia con la política de migración a software libre de la UCI es seleccionarlo como servidor web.

1.4.4 UML 2.0 como Lenguaje Unificado de Modelado

Los diseños logrados usando UML se pueden realizar en cualquier lenguaje orientado a objetos ya que este se aplica a una multitud de diferentes tipos de sistemas, dominios, y métodos o procesos. UML se seleccionó como lenguaje de modelado para el desarrollo de la aplicación puesto que es el lenguaje estándar de modelado más popular para crear planos de software, además de visualizar, especificar, construir, documentar y detallar los artefactos del sistema. Es un lenguaje que permite modelar estructuras complejas utilizando técnicas orientadas a objetos (Larman, 2003).

Se decide la utilización de este lenguaje de modelado pues posibilita estandarizar la documentación referente al proceso de desarrollo de software. Además de ser uno de los lenguajes de modelados más utilizados y difundidos.

² Hypertext Transfer Protocol o **HTTP** (protocolo de transferencia de hipertexto)

1.4.5 Sistemas Gestores de Bases de Datos

PostgreSQL 9.3

PostgreSQL es un potente gestor de base de datos, bajo licencia BSD³. Cuenta con más de 15 años de desarrollo activo y arquitectura probada que ha ganado mucha reputación por su confidencialidad e integridad en los datos (PostgreSQL, 2013). Está diseñado para ambientes de altos volúmenes de datos. Al ser multiplataforma está disponible para muchos sistemas operativos.

En términos de eficiencia y recursos, es capaz de ajustarse al número de procesadores y a la cantidad de memoria que posee el sistema de forma óptima, lo que posibilita atender un mayor número de peticiones concurrentemente. Además se selecciona para el desarrollo de esta aplicación por ser un proyecto de código abierto y en correspondencia con las políticas de la UCI para la migración de software.

1.4.6 NetBeans como IDE

NetBeans es un IDE⁴ que posee un excelente completamiento de código. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo. Es un producto libre y gratuito sin restricciones de uso (NetBeans IDE, 2015). Permite desarrollar aplicaciones para diferentes entornos como escritorio y web, utilizando como lenguaje Java, C/C++, PHP, JavaScript, entre otros. Presenta mejoras para SOA⁵ y UML⁶. Después de haber analizado varias herramientas de este tipo e identificar características similares como licencias, completamiento de código e integración con los lenguajes de programación empleados para la construcción de la solución, eleva la productividad y se garantiza la calidad del producto final (Palau, y otros, 2012). Se decide su utilización por que reúne los requisitos para el desarrollo, y por la familiarización de los desarrolladores con dicha herramienta.

³ Berkeley Software Distribution (Distribución de software Berkeley).

⁴ Integrated development environment (Entorno Integrado de Desarrollo).

⁵ Service Oriented Architecture (Arquitectura Orientada a Servicios).

⁶ Unified Modeling Language (Lenguaje Unificado de Modelado).

1.4.7 Visual Paradigm como Herramienta Case

Las herramientas CASE⁷ son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software permiten incrementar la productividad y el control de la calidad en cualquier proceso de elaboración de software, pues transforman la actividad de desarrollar software en un proceso automatizado (Valle, 2009). Visual Paradigm 8.0 utiliza UML como lenguaje de modelado. Esta herramienta está pensada para usuarios interesados en construir sistemas de software fiables con el uso del paradigma orientado a objetos, incluyendo actividades entre las que se destacan: ingeniería de software, análisis de sistemas y análisis de negocios. La utilización de la herramienta en cuestión ayudará a aumentar la productividad durante el desarrollo de la solución, automatizando parte del proceso, además, la documentación será enriquecida con diagramas para lograr un mejor entendimiento del funcionamiento del sistema por futuros desarrolladores.

1.5 Lenguajes de Programación

Lenguajes web de programación del lado del cliente

Los lenguajes del lado del cliente basan su procesamiento en el cliente web, es decir, que se ejecutan en el navegador del usuario. En este sentido existen varios lenguajes, pero la presente investigación utiliza JavaScript para la solución que se propone.

JavaScript

JavaScript es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario (Gauchat, 2012).

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, en bases de

⁷ Computer Aided Software Engineering, (Ingeniería de Software Asistida por Computadora).

datos locales al navegador aunque existen implementaciones de dicho lenguaje del lado del servidor conocidos como SSJS⁸ (Espinosa, y otros, 2012).

El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único con que cuenta este lenguaje, es el propio navegador.

Debido a su compatibilidad con la mayoría de los navegadores modernos, es actualmente el lenguaje más utilizado. Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, en ocasiones con ligereza. Una de sus ventajas radica en que las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia. Es un lenguaje de programación interpretado (Palau, y otros, 2012).

Lenguajes web de programación del lado del servidor

Los lenguajes de lado del servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. (Espinosa, y otros, 2012).

PHP 5.5

Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, y con extensa documentación. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores (Palau, y otros, 2012). La selección de dicho lenguaje se debe a la selección del marco de trabajo y por la familiarización del desarrollador con dicho lenguaje.

1.6 Conclusiones parciales

En este capítulo se delimitaron los límites de la investigación, se centró un objetivo y un grupo de tareas a desarrollar. Con el estudio realizado, se llega a la conclusión de que ninguno de los sistemas analizados responde a las necesidades de la problemática, determinándose de esta forma que ninguno puede ser

⁸ acrónimo de Server-side JavaScript

Capítulo 1. Fundamentación Teórica

parte íntegra de la solución, pero que permitió identificar un conjunto de funcionalidades básicas que debe poseer la aplicación que se desea construir. Las herramientas y tecnologías descritas en el presente capítulo presentan la base para etapas posteriores del desarrollo durante el diseño y la implementación de la solución.

CAPÍTULO 2. PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

En este capítulo se realiza un análisis de la solución propuesta, relacionan los conceptos y entidades que están presentes entorno al sistema. Se identifican los requisitos funcionales y no funcionales con los que contará el sistema, se detallan los casos de uso y se describen los actores. Además, se incluye el diseño de la solución propuesta, la arquitectura utilizada, los patrones de diseño que se usaron, los diagramas de clases del diseño, y el diseño de la base de datos.

2.2 Modelo Conceptual en el Desarrollo del Software

El modelo conceptual no representa el sistema a desarrollar, pero permite un mayor entendimiento de la situación existente en un entorno determinado. “La etapa orientada a objetos esencial del análisis o investigación es la descomposición de un dominio de interés en clases conceptuales individuales u objetos. El modelo conceptual describe conceptos del mundo real, no componentes software” (Larman, 2003). Se decide utilizar el modelo conceptual para un mayor entendimiento de los conceptos asociados al sistema y las relaciones entre dichos conceptos, logrando una mejor descripción de estos debido a que se encuentran bien definidos los procesos necesarios.

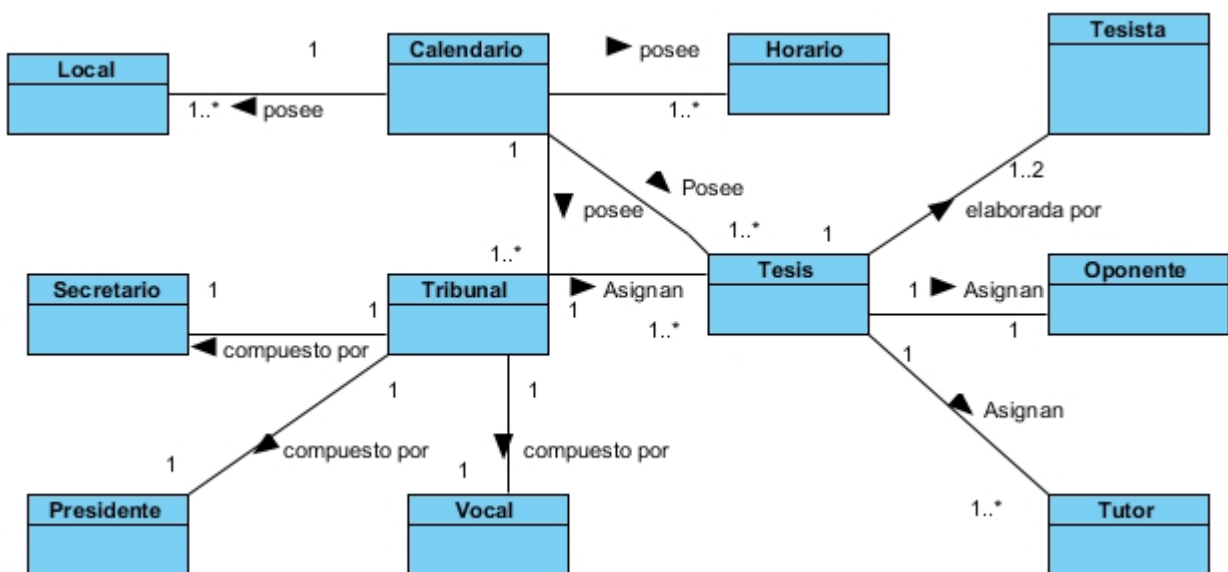


Fig. 6: Modelo Conceptual

Capítulo 3. Construcción y validación de la solución propuesta

2.2.1 Descripción del Modelo Conceptual

Un calendario de tesis está conformado por: locales, tribunales, horarios y tesis. Las tesis son elaboradas por uno o dos estudiantes, poseen uno o más profesores en calidad de tutores y se le asigna un oponente. A un tribunal se le asignan tesis y están compuestos por tres miembros: presidente, secretario y vocal.

2.2.2 Glosario de Términos del Modelo del Dominio

Local: Lugar donde se realiza la exposición de una tesis.

Calendario: Horario que recoge, tesis, tribunales y locales relacionados con el acto de tesis.

Tesis: Ejercicio final permite comprobar los conocimientos adquiridos durante la carrera.

Tesista: Autor de una tesis.

Tutor: Profesor que instruye, guía o apoya al tesista en la elaboración de la tesis.

Oponente: Profesor que realiza oposición en la exposición de la tesis.

Tribunal: Conjunto de profesores que poseen cierto dominio y experiencia y que evalúan al tesista en el desarrollo de la tesis.

Presidente: Encargado de dirigir la comisión que evalúa las tesis.

Secretario: Documenta el proceso de exposición de una tesis.

Vocal: Es el portavoz de las decisiones que tome el tribunal con respecto a la tesis.

2.3 Modelo del Negocio

Se puede definir el modelo de negocio como una herramienta conceptual que contiene un conjunto de objetos, conceptos y sus relaciones con el objetivo de expresar la lógica del negocio de una empresa (Osterwalder, y otros, 2005). Proporciona una vista simplificada de la estructura de negocios que actúa como la base para la comunicación, mejoras o innovación y define los requisitos de los sistemas de información que apoyan la empresa (Eriksson, y otros, 2000).

2.3.1 Actores del negocio

Un actor es una agrupación uniforme de personas, sistemas o máquinas que interactúan con el sistema que se está construyendo.

Capítulo 3. Construcción y validación de la solución propuesta

Tabla 1: Descripción de actores del Negocio

Actor	Descripción
Vicedecano de Formación	Este actor es el encargado de gestionar los tribunales de tesis y de gestionar el calendario de tesis de la facultad.

2.3.2 Diagrama de casos de uso del negocio

Un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus Servicios. (Ceria, 2002)

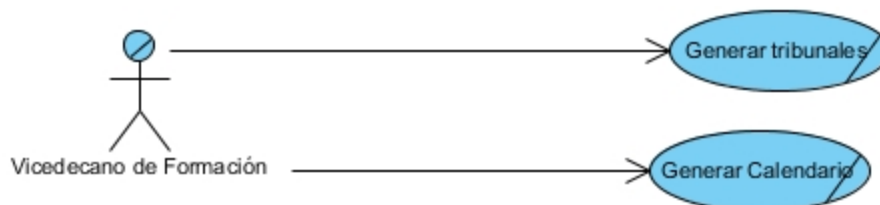


Fig. 7: Diagrama de casos de uso del negocio

2.3.3 Descripción de los casos de uso del negocio

Tabla 2: Descripción del Caso de Uso Generar Tribunales

Caso de uso del negocio	Gestionar Tribunales
Actores	Vicedecano de Formación
Resumen	El caso de uso se inicia cuando se inicia el proceso de tesis, se seleccionan los profesores disponibles y con ellos se conforman los tribunales de tesis.
Casos de uso asociados	
Mejoras propuestas	

Tabla 3: Descripción del Caso de Uso Generar Calendario

Caso de uso del negocio	Gestionar Calendario
Actores	Vicedecano de Formación
Resumen	El caso de uso se inicia previo a la ocurrencia de los eventos que tiene el proceso de tesis (pre defensas, defensas). A los tribunales se les asignan tesis, locales, fecha y hora.
Casos de uso asociados	
Mejoras propuestas	

2.4 Modelo del Sistema

Luego de haber relacionado las clases que conforman el dominio, así como la descripción de los principales conceptos que utilizan los usuarios y con los que debe trabajar la aplicación a través del modelo conceptual, se representan los principales artefactos del flujo de trabajo Modelado, que tiene como objetivo principal guiar el desarrollo hacia un sistema que cumpla con las necesidades planteadas por el cliente.

2.4.1 Requisitos

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir (PRESSMAN, 2005). Definen lo que el sistema tiene que hacer y los servicios que debe proporcionar al usuario. Para que un requisito se considere satisfactorio debe cumplir las siguientes características: implementación independiente, consistente y no ambiguo, preciso, verificable, que pueda ser leído y modificable. Existen dos grandes categorías en las que pueden clasificarse los requisitos, estas son: requisitos funcionales y requisitos no funcionales.

2.4.2 Estrategia de captura de requisitos

El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto. El proceso de captura se basa en investigar, lo que se debe construir. Para la captura de requisitos se utilizó como método de apoyo la entrevista. Esta estrategia es una de las técnicas más usadas en la captura de requisitos. A través del contacto directo con ambas partes (cliente y el equipo de

Capítulo 3. Construcción y validación de la solución propuesta

desarrollo) posibilita de manera eficiente reunir toda la información relevante para la realización del software, y de esta forma se garantiza que la solución se adecue a las necesidades del cliente. Si se cumple con todos los requisitos, el software posee lo que el cliente desea entonces tendrá buena calidad.

2.4.3 Listado de requisitos funcionales

A continuación se detallan los requisitos funcionales que la solución propuesta debe cumplir como respuesta al problema identificado.

- RF 1. Autenticar usuario. Esta funcionalidad permite al usuario autenticarse en el sistema.
- RF 2. Registrar usuario. Esta funcionalidad permite al usuario registrar sus datos en el sistema.
- RF 3. Editar perfil de usuario. Esta funcionalidad permite editar los datos del usuario ya registrado.
- RF 4. Adicionar Profesor. Esta funcionalidad permite adicionar un profesor.
- RF 5. Editar Profesor. Esta funcionalidad permite editar los datos de un profesor.
- RF 6. Listar Profesor. Esta funcionalidad permite listar los profesores existentes.
- RF 7. Eliminar Profesor. Esta funcionalidad permite eliminar un profesor.
- RF 8. Adicionar Estudiante. Esta funcionalidad permite adicionar un estudiante.
- RF 9. Editar Estudiante. Esta funcionalidad permite editar los datos de un estudiante.
- RF 10. Listar Estudiante. Esta funcionalidad permite listar los estudiantes existentes.
- RF 11. Eliminar Estudiante. Esta funcionalidad permite eliminar un estudiante.
- RF 12. Adicionar Local. Esta funcionalidad permite adicionar un local.
- RF 13. Editar Local. Esta funcionalidad permite editar los datos de un local.
- RF 14. Listar Locales. Esta funcionalidad permite listar los locales existentes.
- RF 15. Eliminar Local. Esta funcionalidad permite eliminar un local.
- RF 16. Adicionar Área. Esta funcionalidad permite adicionar un área.
- RF 17. Editar Área. Esta funcionalidad permite editar los datos de un área.
- RF 18. Listar Áreas. Esta funcionalidad permite listar las áreas existentes.
- RF 19. Eliminar Área. Esta funcionalidad permite eliminar un área.
- RF 20. Adicionar Tesis. Esta funcionalidad permite adicionar una tesis.
- RF 21. Editar Tesis. Esta funcionalidad permite editar los datos de una tesis.
- RF 22. Listar Tesis. Esta funcionalidad permite listar las tesis existentes.
- RF 23. Eliminar Tesis. Esta funcionalidad permite eliminar una tesis.
- RF 24. Adicionar Línea de Investigación. Esta funcionalidad permite adicionar una línea de investigación.

Capítulo 3. Construcción y validación de la solución propuesta

- RF 25. Editar Línea de Investigación. Esta funcionalidad permite editar los datos de una línea de investigación.
- RF 26. Listar Línea de Investigación. Esta funcionalidad permite listar las líneas de investigación existentes.
- RF 27. Eliminar Línea de Investigación. Esta funcionalidad permite eliminar una línea de investigación.
- RF 28. Adicionar Tribunales. Esta funcionalidad permite adicionar un tribunal.
- RF 29. Editar Tribunales. Esta funcionalidad permite editar un tribunal.
- RF 30. Listar Tribunales. Esta funcionalidad permite listar los tribunales existentes.
- RF 31. Eliminar Tribunales. Esta funcionalidad permite eliminar un tribunal.
- RF 32. Editar Calendario. Esta funcionalidad permite editar los datos de un calendario.
- RF 33. Eliminar Calendario. Esta funcionalidad permite eliminar un calendario.
- RF 34. Adicionar Afectación. Esta funcionalidad permite adicionar una afectación.
- RF 35. Editar Afectación. Esta funcionalidad permite editar los datos de una afectación.
- RF 36. Listar Afectación. Esta funcionalidad permite listar afectaciones existentes.
- RF 37. Eliminar Afectaciones. Esta funcionalidad permite eliminar una afectación.
- RF 38. Generar Tribunales. Esta funcionalidad permite generar los tribunales automáticamente.
- RF 39. Generar Calendario. Esta funcionalidad permite generar el calendario automáticamente.
- RF 40. Exportar a PDF. Esta funcionalidad permite exportar el calendario en un documento en formato PDF.
- RF 41. Eliminar Usuario. Esta funcionalidad permite eliminar un usuario previamente registrado en la aplicación.

2.4.4 Requisitos No Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. (PRESSMAN, 2005).

Los requisitos no funcionales pueden ser más críticos que los funcionales, puesto que si un requisito funcional no se cumple, el sistema se degrada, pierde eficacia, y puede no responder a la totalidad de los requisitos del usuario, pero en cambio si un requisito no funcional no se cumple, el sistema puede inutilizarse.

RNF 1. Usabilidad

Capítulo 3. Construcción y validación de la solución propuesta

El sistema debe permitir un alto nivel de facilidades de uso, basado en el cumplimiento de los siguientes aspectos:

- ✓ El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.
- ✓ Las funcionalidades principales del sistema estarán orientadas a íconos para un mayor reconocimiento por parte del usuario.
- ✓ El diseño deberá ser sencillo, con pocas entradas, donde no sea necesario mucho entrenamiento para utilizar el sistema.

Interfaces de usuario

- ✓ El sistema debe poseer una interfaz gráfica uniforme incluyendo pantallas, menús y opciones.
- ✓ Tanto los títulos de los componentes de la interfaz, como los mensajes para interactuar con los usuarios, deberán ser en idioma español y tener una apariencia uniforme guiada por los colores negro y azul claro.
- ✓ Los mensajes definidos deberán ser lo suficientemente informativos para dar a conocer la severidad de los mismos.

Interfaces hardware

Para ordenadores clientes:

- ✓ Se requiere tengan tarjeta de red.
- ✓ Al menos 128 MB de memoria RAM.
- ✓ Procesador 512 MHz como mínimo.

Para los servidores:

- ✓ Se requiere tarjeta de red.
- ✓ El Servidor de base de datos debe tener como mínimo 2 GB de RAM y 40 GB de disco duro.
- ✓ Procesador de 3 GHz como mínimo.

Interfaces software

La construcción de la aplicación funcionará bajo los conceptos de la arquitectura cliente/servidor. Por tanto se deben tener como requerimientos mínimos de software:

Para los ordenadores clientes:

- ✓ Un navegador como Mozilla Firefox, Zafari u otro navegador.

Para los Servidores:

Capítulo 3. Construcción y validación de la solución propuesta

- ✓ Servidor Web Apache 2.2 o superior, con módulo PHP 5 configurado con la extensión pgsql incluida.
- ✓ PostgreSQL 9.1 como Sistema Gestor de Base de Datos.

RNF 2. Requisitos de licencia

De acuerdo a los tipos de licencias de los componentes y herramientas que propone utilizar para el desarrollo del producto se puede catalogar legalmente esta arquitectura de modelo libre, permitiendo la utilización, modificación y distribución de las mismas por terceros sin necesidad de obtener la autorización de sus respectivos titulares.

RNF 3. Seguridad

La información manejada por el sistema estará protegida de acceso no autorizado y divulgación

2.5 Definición de casos de uso del sistema

Una vez recopilados los requisitos funcionales del sistema es necesario conformar el Diagrama de Casos de Uso del Sistema (DCUS). Para tener una mejor comprensión y organización, se agrupan los requisitos según su relación en CUS, además de especificar los actores que interactúan con el sistema.

2.5.1 Descripción de los actores que interactúan con el sistema

Los actores se definen como los roles que puede tener un usuario; pueden ser humanos, otros sistemas, máquinas, hardware, etc. que interactúan con un sistema, para de esta forma intercambiar datos, jugando un papel muy importante en los casos de uso. A continuación en la tabla 4 se describen los actores que interactúan e interactúan

Tabla 4: Descripción de Actores del Sistema

Planificador	Usuario que podrá realizar todas las funcionalidades correspondientes a gestionar: usuario, locales, profesor, estudiante, tribunales, líneas de investigación, áreas; administrar calendario, generar tribunales y generar calendario; además todas las funcionalidades del visualizador.
Miembro de tribunal	Usuario que podrá Gestionar Afectaciones y además todas las funcionalidades del visualizador.
Visualizador	Usuario que podrá realizar todas las funcionalidades correspondientes a visualizar calendario, exportarlo a formato PDF y autenticar usuario.

2.5.2 Identificación de los casos de uso

Los CUS son un conjunto de escenarios que identifican una línea de utilización para el sistema que va a ser construido y que facilitan una descripción de cómo el sistema se usará (PRESSMAN, 2005). A continuación en la tabla 5 se identifican los CUS que hacen referencia a los requisitos funcionales identificados para desarrollar el sistema.

Tabla 5: Requisitos Funcionales y Casos de Uso

Referencia a requisitos funcionales	Nombre del CUS
RF 4, RF 5, RF 6, RF 7	Gestionar Profesor
RF 8, RF 9, RF 10, RF 11	Gestionar Estudiante
RF 12, RF 13, RF 14, RF 15	Gestionar Locales
RF 16, RF 17, RF 18, RF 19	Gestionar Área
RF 20, RF 21, RF 22, RF 23	Gestionar Tesis
RF 24, RF 25, RF 26, RF 27	Gestionar Línea de Investigación
RF 28, RF 29, RF 30, RF 31	Gestionar Tribunales
RF 32, RF 33	Administrar Calendario
RF 34, RF 35, RF 36, RF 37	Gestionar Afectaciones
RF 38	Generar Tribunales
RF 39	Generar Calendario
RF 40	Exportar a PDF
RF2, RF3, RF41	Gestionar Usuario
RF1	Autenticar usuario

2.5.3 Diagrama de Casos de uso del sistema

El diagrama de CUS es el modelo de las funciones deseadas para el sistema y su entorno, que sirve como contrato entre el cliente y los desarrolladores. Estas funciones por lo general agrupan uno o más requisitos funcionales definidos en el flujo de trabajo de Modelado, como lo propone la metodología AUP. Una vez recopilados los requisitos funcionales del sistema, se muestra en la Fig. 8 el Diagrama de Caso de Uso del Sistema, en el cual se representan los actores y su relación con el sistema:

Capítulo 3. Construcción y validación de la solución propuesta

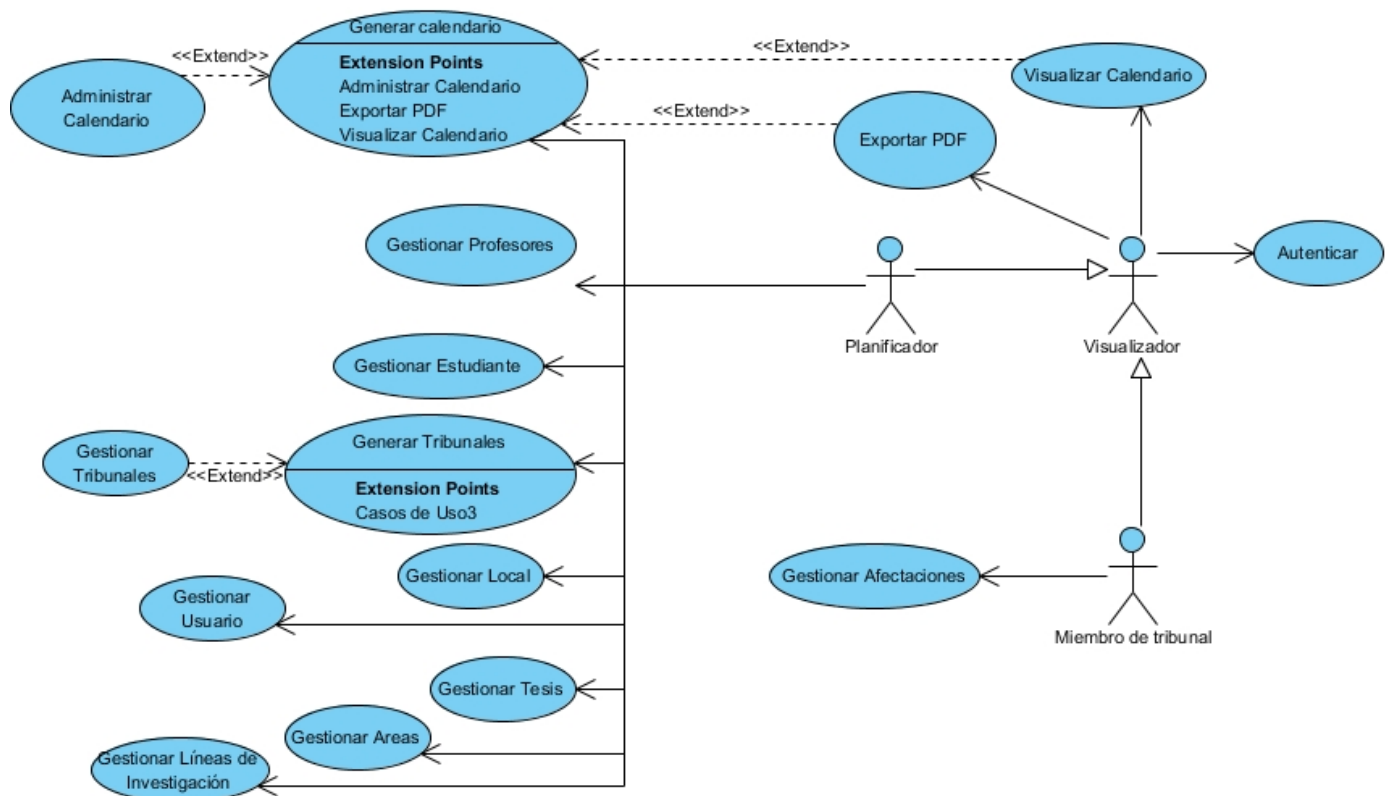


Fig. 8 : Diagrama de Casos de Uso del Sistema

2.5.4 Descripción textual de los casos de uso del sistema

Teniendo en cuenta que los casos de uso son los que dirigen todo el proceso de desarrollo y debido a que la mayoría de las actividades como el análisis, diseño y prueba se llevan a cabo partiendo de los casos de uso, se hace necesario realizar una descripción de los mismos. Debido a que es muy amplia la información, se muestra a continuación en la tabla 6 la descripción de uno de los caso de uso más crítico del sistema, los demás pueden ser consultados en el expediente de proyecto.

Tabla 6: Descripción del Caso de Uso “Gestionar Tesis”

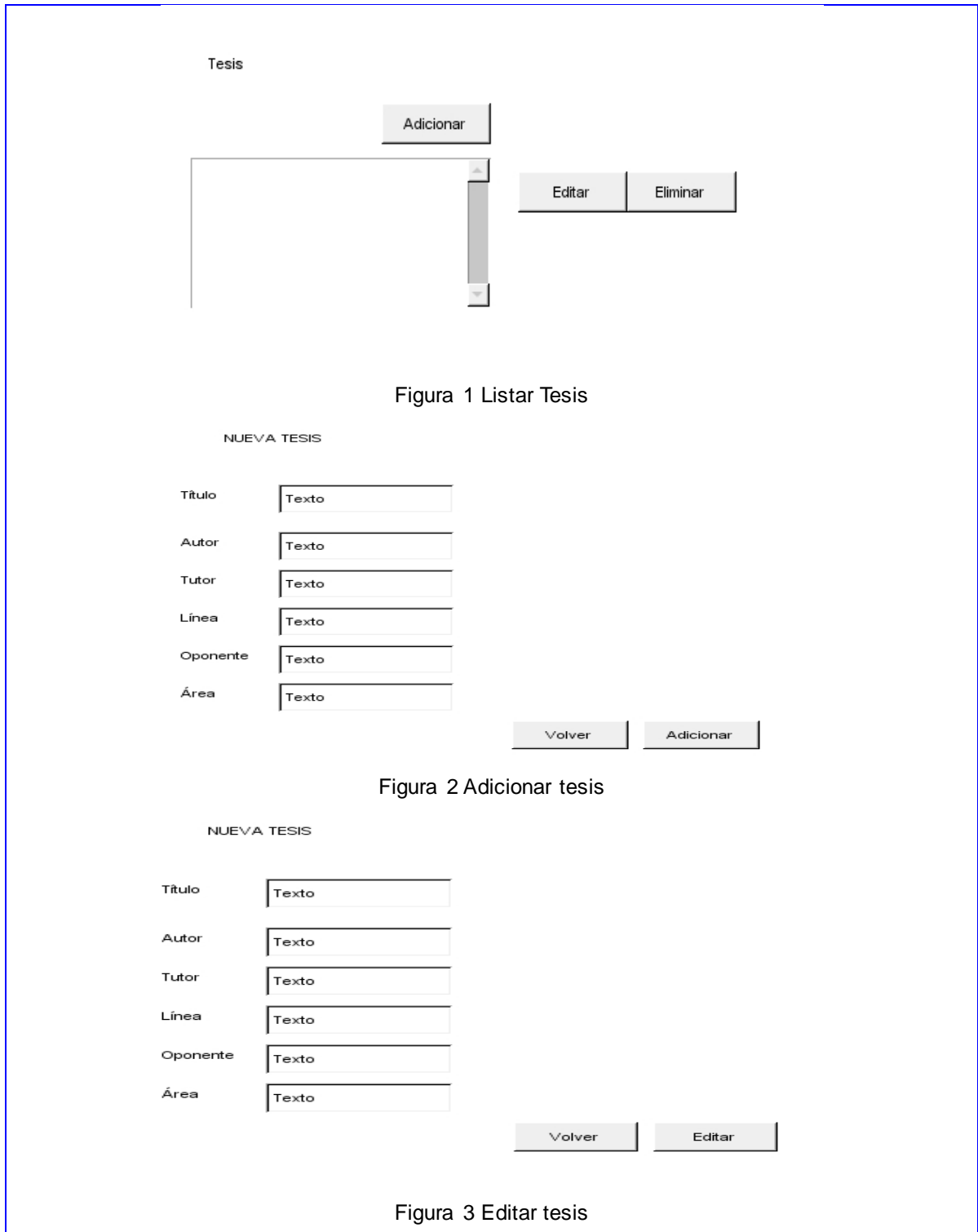
Objetivo	Gestionar Tesis.
Actores	Planificador
Resumen	Este caso de uso se lleva a cabo con el objetivo de adicionar, editar ,listar o eliminar una tesis
Complejidad	Alta
Prioridad	Alta
Precondiciones	Planificador debe estar autenticado en el sistema deben existir, profesores,

Capítulo 3. Construcción y validación de la solución propuesta

	áreas, líneas de investigación y estudiantes en la base de datos	
Pos condiciones		
Flujo de eventos		
Flujo básico Gestionar imagen		
	Actor	Sistema
1.	El planificador selecciona la opción para Adicionar, Listar, Editar, Eliminar las Tesis	
2.		El sistema muestra un menú con las acciones que el usuario puede realizar. Ver Interfaz 1. - Si selecciona "Adicionar" (A), ver flujo de eventos. - Si selecciona "Editar" (C), ver sección "Editar". - Si selecciona "Eliminar" (D), ver sección "Eliminar". Los tribunales se listan automáticamente al iniciarse el CU.
3.	Presiona el botón Adicionar (Ver Interfaz 2).	
4.		Muestra la interfaz con los siguientes datos: - Título(Campo obligatorio) - Autor (Campo obligatorio) - Tutor(Campo obligatorio) - Línea de investigación (Campo obligatorio) - Área(Campo obligatorio) - Oponente(Campo obligatorio)
5.	Introduce los datos necesarios.	
6.	Selecciona la opción Adicionar.	
7.		Valida que los datos insertados sean correctos.
8.		Inserta una nueva Tesis en la BD
10		Actualiza el listado de Tesis. Termina el CU.
.		
Flujos alternos		
6a Evento Selecciona la opción Volver.		
	Actor	Sistema
7.		Cancela la operación y muestra el listado de Tesis. Termina el CU.
Flujos alternos		
7a Evento Existen campos vacíos.		
	Actor	Sistema
8.		Muestra el mensaje de error: -El campo título no puede estar vacío. -El campo autor no puede estar vacío. -El campo tutor no puede estar vacío. -El campo línea de investigación no puede estar

Capítulo 3. Construcción y validación de la solución propuesta

		vacío. -El campo área no puede estar vacío. -El campo oponente no puede estar vacío.
Sección 1: "Editar"		
Flujo básico Editar Tesis		
	Actor	Sistema
2.	Presiona el botón Editar (Ver Interfaz 3)	
3.		Muestra la interfaz con los datos de la Tesis seleccionada.
4.	Modifica los datos que desee.	
5.	Selecciona la opción Editar.	
6.		Valida que los datos modificados sean correctos.
7.		Actualiza los datos de la imagen seleccionada.
9.		Actualiza el listado de imágenes. Termina el CU.
Flujos alternos		
5a Evento Selecciona la opción Volver.		
	Actor	Sistema
6		Cancela la operación y muestra el listado de Tesis. Termina el CU.
Sección 2: "Eliminar Tesis"		
Flujo básico Eliminar tesis		
	Actor	Sistema
2.	Selecciona la opción Eliminar (Ver Interfaz 1).	
3.		Muestra mensaje de confirmación de la operación.
4.	Selecciona la opción Eliminar.	
5.		Elimina la tesis seleccionada.
7.		Actualiza el listado de tesis. Termina el CU.
Flujos alternos		
4a Evento Selecciona la opción Volver.		
	Actor	Sistema
5		Cancela la operación y muestra el listado de tesis. Termina el CU.
Relaciones	CU incluidos	No existen.
	CU extendidos	No existen.
Requisitos no funcionales	No procede.	
Asuntos pendientes	No procede.	
Prototipo elemental de interfaz gráfica de usuario		



2.5.5 Patrones de Casos de Uso

Los Patrones de Caso de Uso se presentan a modo de herramientas que permiten resolver los problemas que se les planteen a los desarrolladores de una forma ágil y sistemática describiendo cómo deberían ser estructurados y organizados los casos de uso, capturando las mejores prácticas para modelarlos. Estos patrones se enfocan hacia el diseño y las técnicas utilizadas en modelos de alta calidad, y no en cómo modelar usos específicos (PRESSMAN, 2005).

Patrones de casos de uso empleados

Patrón CRUD Completo: Este patrón consta de un caso de uso, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples (Wei , 2003). Se pone de manifiesto en los casos de uso Gestionar: Estudiantes, Profesores, Locales, etc.

Patrón Múltiples actores Rol común: Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso (Wei , 2003). En la figura correspondiente al Diagrama de casos de uso del sistema, se observa este patrón en la relación de los actores “Miembro de tribunal” y “Visualizador”.

2.6 Arquitectura de Software

La arquitectura de software de un sistema es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente y las relaciones entre ellos (PRESSMAN, 2005). La misma está compuesta por los artefactos más significativos, permitiendo establecer un esquema de cómo va a quedar constituido el software. Sobre la arquitectura inicial se van agregando nuevos artefactos propiciando así una arquitectura más robusta, describiéndose en la misma los principales aspectos tomados en cuenta para la construcción del software de forma progresiva.

MVC (por sus siglas en inglés) es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo esté estructurado de una mejor manera, facilitando

Capítulo 3. Construcción y validación de la solución propuesta

la programación en diferentes capas de manera paralela e independiente. MVC sugiere la separación del software en tres estratos: Modelo, Vista y Controlador (Pavón Mestras, 2009).

Para el desarrollo de la aplicación se seleccionó dicha arquitectura debido a que posee la ventaja de que el modelo refresque la vista sin pasar por el controlador disminuyendo el tiempo de respuesta del sistema, además de poseer múltiples vistas lo que permite una mejor interacción con usuario. La estructuración del código beneficia la escalabilidad del software posibilitando mayor facilidades para realizar modificaciones o mantenimientos. Una gran cantidad de marcos de trabajo implementa esta arquitectura por defecto uno de ellos es Symfony.

2.6.1 Patrones de arquitectura

Un patrón es una descripción de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, consecuencias de utilizar el patrón, ejemplos de implementación y lista de patrones relacionados (Craig Larman, 1999). En el mundo del software los patrones implementan soluciones a problemas frecuentes, brindando un vocabulario de entendimiento común entre los desarrolladores (Pavón Mestras, 2004).

Un patrón se define como una solución probada con éxito que aparece una y otra vez ante determinado tipo de problema en un contexto dado. Los patrones se definen por un nombre, un problema, una solución y las consecuencias de su aplicación. Éste define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones (PRESSMAN, 2005).

Patrones arquitectónicos empleados

El sistema se desarrolla bajo el patrón arquitectónico MVC que es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

La estrategia de MVC se basa en la separación del código en tres capas diferentes, acotadas por su responsabilidad, llamadas Modelo, Vista y Controlador. Fue creada hace varias décadas incluso antes de la aparición de la Web. No obstante, en los últimos años ha ganado mucha fuerza y seguidores gracias a

la aparición de numerosos marcos de trabajo para el desarrollo Web que utilizan el patrón MVC como modelo para la arquitectura de las aplicaciones.

2.7 Modelo de Diseño

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tiene impacto en el sistema a considerar. Además, sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación. En el modelo de diseño, los casos de uso son realizados por las clases de diseño y sus objetos (Jacobson, y otros, 2000).

2.7.1 Diagrama de paquetes

El diagrama de paquetes es una de las representaciones más comunes en el diseño de una aplicación. Un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones (PRESSMAN, 2005). Estos diagramas proporcionan la composición de la jerarquía lógica de un sistema. Los paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada uno y minimizar el acoplamiento externo entre ellos. El siguiente diagrama de paquetes (Fig. 9) muestra la estructura del sistema.

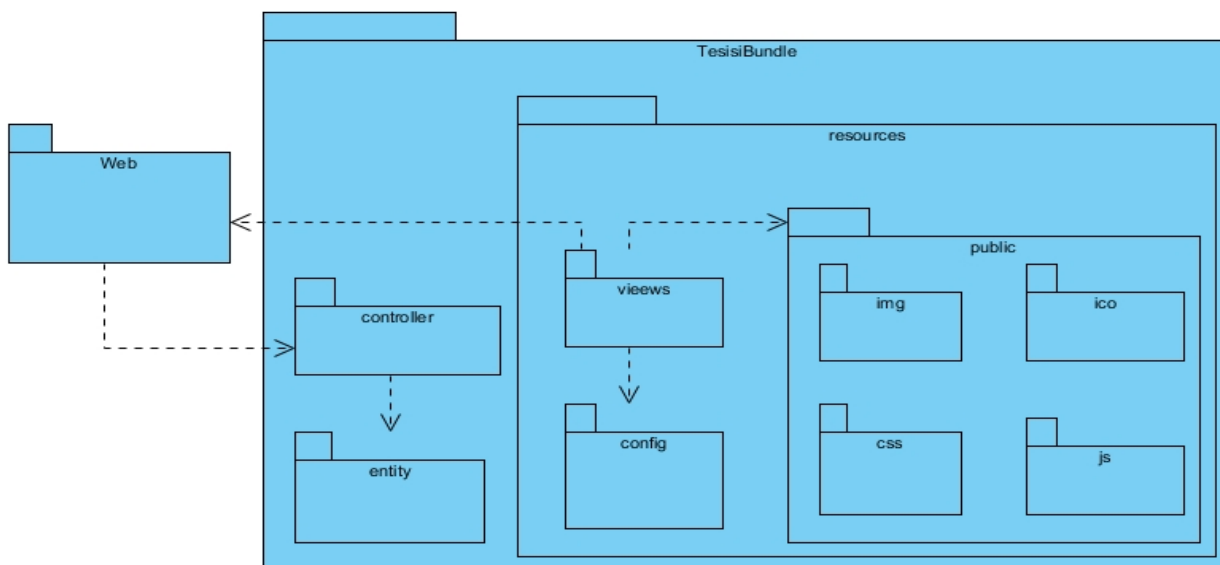


Fig. 9: Diagrama de Paquetes

A continuación una breve descripción de los paquetes que componen el sistema:

TesisBundle/controller/ contiene todos los componentes (de extensión php) que consultan los modelos para dar una respuesta.

TesisBundle/entity/ contiene todos los componentes (de extensión php) que mapean las tablas de la base de datos.

TesisBundle/resources/views/ contiene todos los componentes que construyen las vistas.

TesisBundle/resources/config/ contiene todas las direcciones que se manejan en el sistema.

TesisBundle/resources/public/img contiene todas las imágenes que se utilizan en el sistema.

TesisBundle/resources/public/ico contiene todos los iconos que se utilizan en el sistema.

TesisBundle/resources/public/css contiene todos los componentes (de extensión css) que se utilizan en el sistema.

TesisBundle/resources/public/js contiene todos los componentes (de extensión js) que se utilizan en el sistema.

La carpeta *resources* contiene el paquete de vistas (*views*) que conforman la parte visual del sistema, estas hacen uso de los elementos que conforman el paquete *public*. La navegación entre las vistas se realiza con el apoyo de los archivos que se encuentran en *config*. Además, se utilizan los paquetes *controller* y *entity* para tratar todas las funcionalidades que brinda el sistema. Todas las peticiones se gestionan a través del paquete *Web* que contiene el controlador frontal de Symfony 2.

2.7.2 Diagrama de clases del diseño

Una clase del diseño es una abstracción de una clase o construcción similar en la implementación del sistema. El lenguaje que se utiliza en dichas clases es el mismo que se emplea para la implementación del sistema; se especifican los atributos y las operaciones; se pueden realizar interfaces si tienen sentido para la programación y los métodos tienen correspondencia con las operaciones que fueron utilizadas en la programación.

A continuación en la Fig. 10 se presenta el diagrama de clases del diseño correspondiente al caso de uso Gestionar Tribunales, los demás pueden ser consultados en el expediente de proyecto.

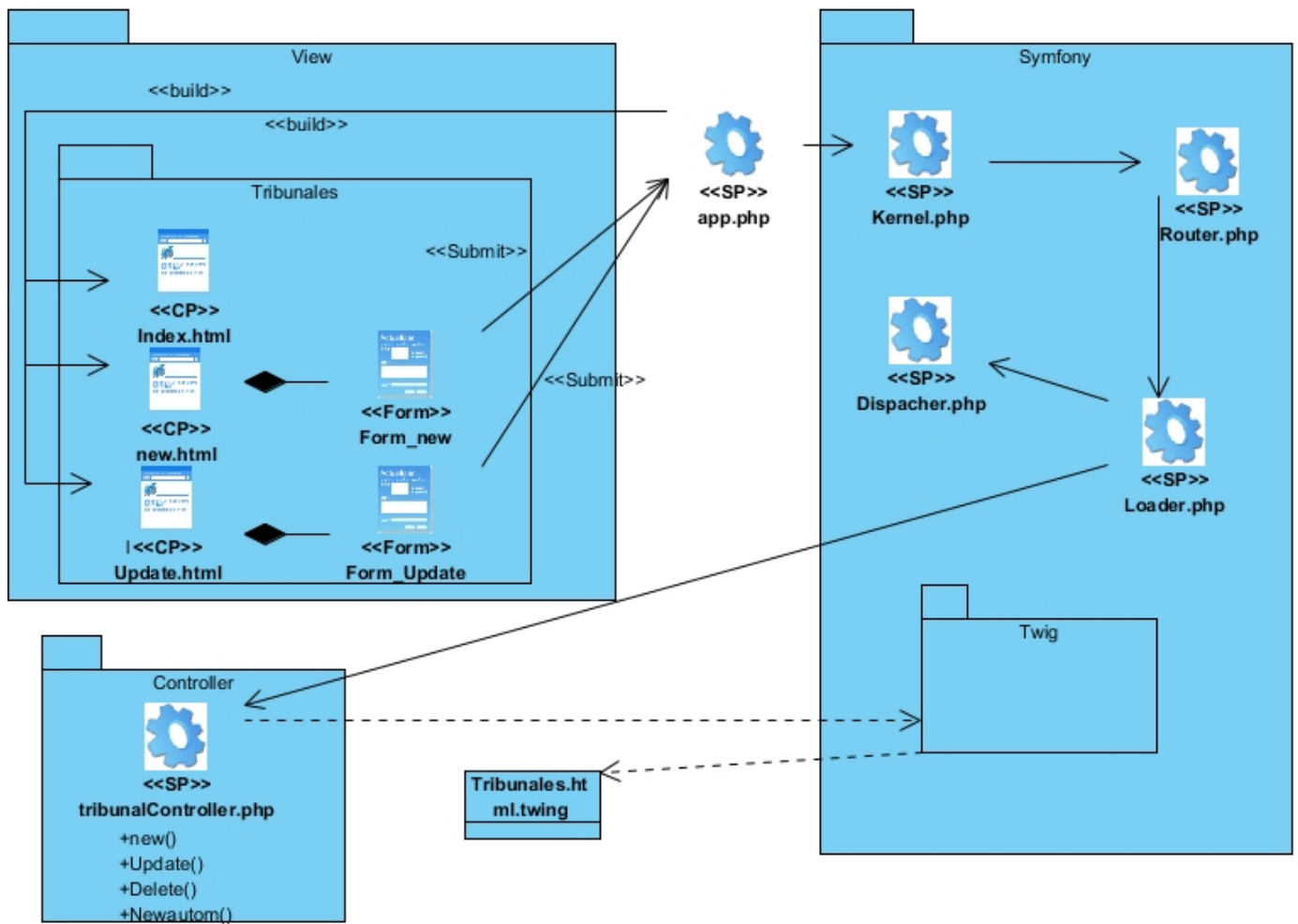


Fig. 10: Diagrama de Clases del Diseño Gestionar Tribunales

2.7.3 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular (Prieto, 2008).

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interfaces, o sea, un patrón de diseño es una solución a un problema de diseño. Expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. Éstos facilitan la reutilización de arquitecturas y diseños de software. Un patrón de diseño identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. De forma general, se clasifican en dos grupos: Patrones de Principios Generales para Asignar Responsabilidad (GRASP: *General Responsibility Assignment Software Patterns*, por sus siglas en inglés) y Patrones de Diseño Pandilla de los Cuatro (*GoF:Gound-of-Four*) (Craig Larman, 1999). Para el desarrollo del producto, se hace uso de patrones pertenecientes a ambos grupos. Seguidamente, se argumenta al respecto.

En el diseño de la propuesta de solución se aplican los siguientes patrones **GRASP**, que permiten describir los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

Experto: Este patrón tiene como objetivo principal asignar una responsabilidad determinada a la clase que tenga la mayor cantidad de información para hacer esta tarea. Es la razón anterior la que le da su apellido Experto “en Información”. La propuesta de solución cuenta con la clase Profesores donde se puede evidenciar que esta presenta todos los datos necesarios como el nombre, la línea de investigación, área; para brindar la información de las entidades relacionadas al problema.

Creador: Se aplica para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto solo pueda ser creada por el objeto que contiene la información necesaria para ello. En este caso el patrón se refleja en la clase ProfesorController, encargada de crear las instancias de la clase Profesor, que serán utilizadas posteriormente por TribunalController, TesisController y CalendarioController. Este patrón favorece el bajo acoplamiento entre clases lo que permite una mayor reutilización de esta solución.

Bajo Acoplamiento: El acoplamiento es una medida de la fuerza en que una clase está conectada a otras, que las conoce y recurre a ellas. Para esta solución, se refleja el bajo acoplamiento, en gran parte de las clases del sistema como ProfesorController y TribunalController, con el objetivo de que una no

Capítulo 3. Construcción y validación de la solución propuesta

dependa de muchas otras, de esta forma, no se afectan las clases por cambios de otros componentes, brinda una mayor facilidad para entenderlos y reutilizarlos.

Alta Cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Se puede afirmar que cada una de las clases del sistema tiene alta cohesión, de manera que estas poseen la característica de tener las responsabilidades estrechamente relacionadas. Esta particularidad evita en cada caso, tener que realizar un trabajo enorme al garantizar un mejor diseño en ocasiones para el resultado global. Este patrón permite que se pueda mejorar la claridad y facilidad en que se entiende el diseño, se simplifique el mantenimiento y existan mejoras de funcionalidad. Se pone de manifiesto en las relaciones entre las clases ProfesorController y TesisController ya que ambas tienen responsabilidades estrechamente relacionadas.

Controlador: Para este caso se hace necesario conocer que un evento del sistema es una operación que se realiza en este, generada por un usuario externo. Un controlador, es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. En esta solución, se encuentran ejemplos en las clases TribunalController y TesisController, Con ambas clases se puede obtener toda la información necesaria sobre el calendario del modelo y de gestionar todos los eventos que ocurren en el mismo.

Symfony 2, como marco de trabajo, influye en el uso de algunos patrones **GOF**, muchos de estos se implementan indirectamente al interés y vista del usuario. En esta sección se mencionan algunos de los que fueron utilizados para realizar la solución.

El patrón de diseño **Decorator** (Decorador) forma parte de la familia de patrones denominados estructurales. Este tipo de patrones describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Decorator permite modificar, retirar o agregar responsabilidades a un objeto dinámicamente. Para estos ejemplos el término dinámicamente, significa que las funcionalidades se modifican, agregan o retiran durante la ejecución del script o aplicación.

La gran ventaja es que permite extender objetos incluso en situaciones donde la extensión vía herencia no es viable o necesaria. Adicionalmente ayuda a conservar el principio de Abierto/Cerrado, en donde se dicta que cada entidad debe estar abierta a extensión, pero cerrada a modificación. Otra ventaja es que las decoraciones evitan la labor de crear clases con mucho código, que en la mayoría de los casos no será evaluado. Se pone de manifiesto en la ejecución de script para la validación de formularios del lado

Capítulo 3. Construcción y validación de la solución propuesta

del cliente, evitando así sobrecargar las clases controladoras en el servidor y un mayor tiempo de respuesta de sistema.

El patrón de diseño **Front Controller** (Controlador frontal) es uno de los patrones de diseño de Symfony más conocido. Es una sección de código que atiende todas las solicitudes en la aplicación y devuelve una respuesta al navegador. Además de que este controlador se utiliza para direccionar todas las solicitudes, es el encargado de iniciar el núcleo del sistema lo que le permite decorar el mismo con características adicionales.

Uno de los principales problemas que evita es la duplicación de código, porque define un método comando en una clase abstracta, que luego se podrá redefinir en las otras clases según las particularidades que se necesiten, con el uso de decoradores.

2.8 Diseño de la Base de Datos

La base de datos necesita de una definición de su estructura, de manera que permita almacenar datos, reconocer el contenido y recuperar la información. Para diseñar una base de datos se necesita seguir un conjunto de pasos que comienzan con definir las clases persistentes, luego refinarlas y clasificarlas junto con sus atributos, para más tarde realizar el diagrama de entidad-relación (Hernández Inza, y otros, 2010).

Para la construcción del presente sistema se utilizó PostgreSQL como Sistema Gestor de Base de Datos SGBD⁹. La base de datos fue diseñada utilizando la herramienta Visual Paradigm. A continuación en la Fig. 11 se representa el diagrama relacional de la base de datos.

⁹ Sistema Gestor de Base de Datos.

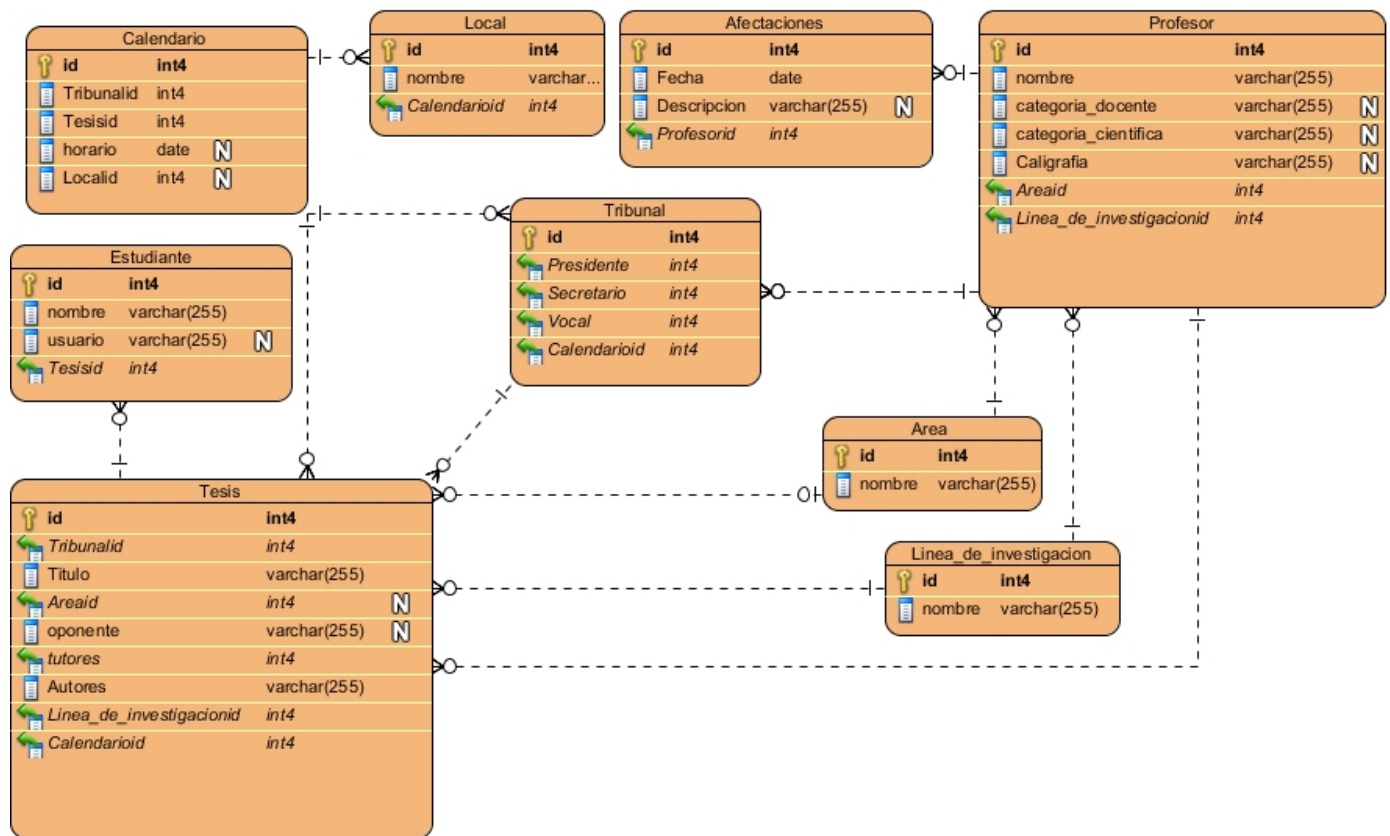


Fig. 11: Diagrama Relacional

2.9 Método de solución

El método de solución desarrollado consta de tres partes fundamentales que tributan a un mismo objetivo: la generación del calendario de tesis.

- Conformación de los tribunales de tesis
- Asignación de tesis a tribunales
- Asignación de horarios y locales a los tribunales.

La generación semiautomática de los tribunales de tesis es uno de los requisitos más importantes debido a que facilita la labor del usuario, al no tener que asignar los roles manualmente, sino que el sistema le hace una propuesta que puede modificar a su conveniencia. Para ello fue generado un algoritmo ávido que posee como variables de entrada un conjunto de profesores con sus características, así como la cantidad de tesis que intervienen en el proceso y como salida devuelve la conformación de tribunales. A continuación la Fig. 12 representa el algoritmo para la generación de tribunales.

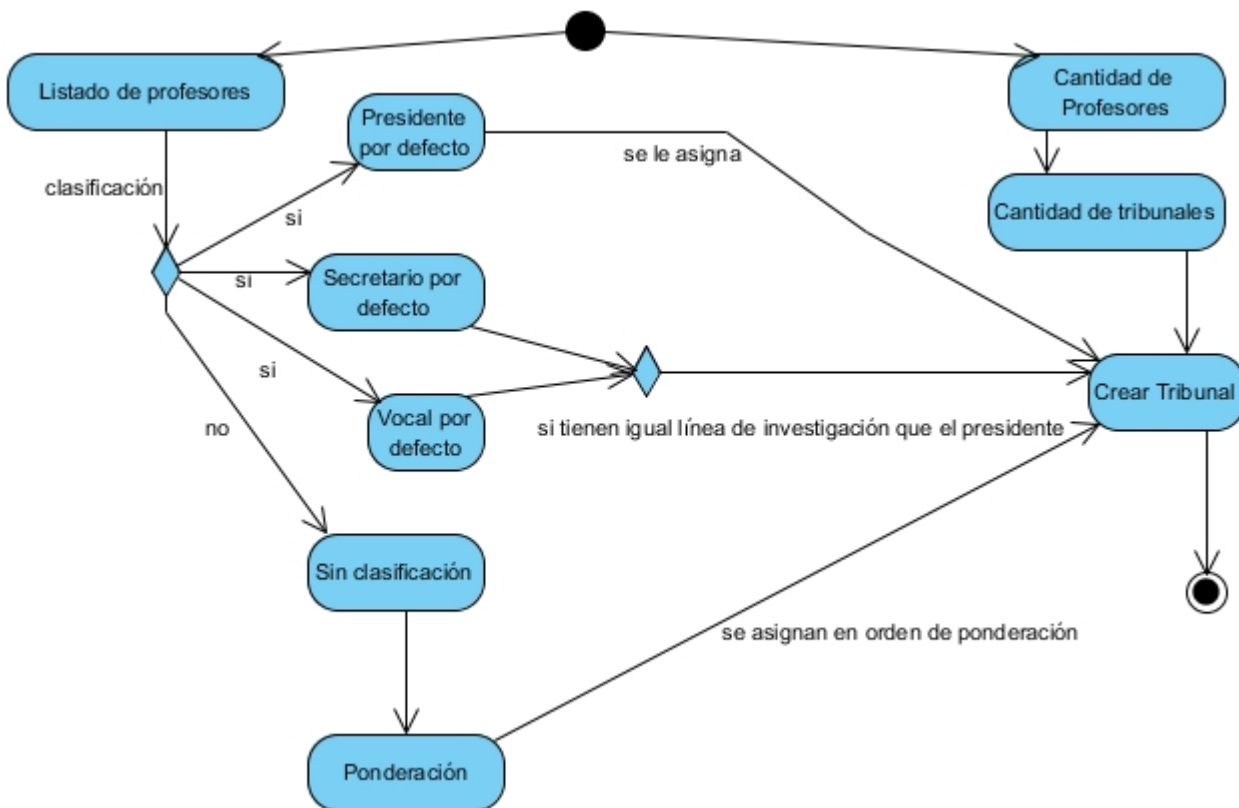


Fig. 12: Algoritmo de generación de tribunales

La entrada del algoritmo es un listado de profesores y el número de tesis. A partir de este último se obtiene la cantidad de tribunales a conformar. Del listado de profesores se seleccionan los profesores que ya son predefinidos en el sistema como presidentes, secretarios o vocales, los profesores que no son preclasificados se seleccionan para ser ponderados según sus características entre las que se encuentran nivel profesional, categoría docente, años de experiencia, las cuales ponderan en ese orden. El valor de la ponderación y el orden de importancia de las características fueron seleccionados basados en el estudio de la problemática, aunque estos criterios pudieran ser variables. Ver tabla 7. Estos valores se ha tomado con el objetivo de simular el proceso que se realiza en la realidad por ejemplo para crear un tribunal los planificadores seleccionan para presidentes los profesores que tienen un mayor nivel profesional, una mayor categoría docente y mayor cantidad de años de experiencia.

Capítulo 3. Construcción y validación de la solución propuesta

Tabla 7: Tabla de ponderaciones

Ponderación	nivel profesional	categoría docente	años de experiencia
4	doctor	titular	>8
3	máster	auxiliar	>6
2	ingeniero licenciado	o asistente	>4
1	especialista	instructor	>2

Se crean los tribunales seleccionando primeramente el presidente de cada uno y posteriormente el secretario y el vocal atendiendo a que estos últimos posean la misma línea de investigación que el presidente del tribunal. En cualquier caso en que no exista por defecto presidente, secretario o vocal se tomará de la lista ponderada el que posea una mayor ponderación. El caso especial de la caligrafía es una característica que se valora en el momento de seleccionar un secretario.

En un segundo paso del método le son asignadas tesis a los tribunales para ello primeramente se calcula el número de tesis máximo que puede tener un tribunal dividiendo la cantidad de tesis entre la cantidad de tribunales y posteriormente se asignan las tesis en correspondencia con la línea de investigación del presidente del tribunal. En caso de no existir ningún presidente de tribunal con la línea de investigación de la tesis se verifica si coincide con la de algún otro miembro del tribunal con el objetivo de que al menos algún miembro posea la misma línea de investigación de la tesis. En caso de no haber sido asignadas, es sistema las asigna en el tribunal que menos tesis asignadas posea con el objetivo de que los tribunales posean en alguna medida la misma cantidad de tesis. Antes de hacer cualquier asignación el sistema verifica que ese tribunal no excede la cantidad de tesis posible a asignar.

En un tercer paso se destinan locales a tribunales considerando solamente que el número de locales debes ser mayor que el número de tribunales debido a que se considera que los locales poseen las mismas condiciones y no hay preferencias. Para la asignación de horarios el sistema exige algunos datos necesarios de las exposiciones como son, fecha de comienzo, hora de comienzo, hora de culminación, duración y receso. Posteriormente comienza a asignar fecha y hora en correspondencia con los parámetros establecidos, sumando la duración y el receso para obtener el momento en que inicia la

Capítulo 3. Construcción y validación de la solución propuesta

siguiente tesis. El sistema valora un horario de almuerzo correspondiente a una hora y en caso de llegar al horario de fin y no haberse planificado todas las tesis se realizará el mismo proceso al día siguiente.

La confección del método simula computacionalmente el proceso que realiza manualmente el planificador del calendario obteniéndose como resultado una propuesta de solución mejor por la utilización de métodos ávidos, pero que en cualquier caso el planificador puede modificar a su preferencia o consideración, por ello es sistema es considerado semiautomático.

2.10 Conclusiones Parciales

La representación del modelo conceptual proporcionó el punto de partida para la correcta elaboración del sistema, el levantamiento de los requisitos tanto funcionales como no funcionales proporciona las medidas necesarias con las que debe cumplir el sistema para su correcta implementación. Por lo que se puede concluir que los artefactos generados según la metodología seleccionada en el capítulo anterior describen el funcionamiento de los aspectos relacionados con el sistema a desarrollar. Entonces quedan sentadas las bases para la implementación y la validación en etapas posteriores del proceso.

CAPÍTULO 3. CONSTRUCCIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En este capítulo se abarca todo lo referente al proceso de implementación y prueba del sistema. Se presenta el diagrama de componentes que muestra las dependencias entre los componentes de la solución a desarrollar. Además se realiza las pruebas de software las que tienen como objetivo garantizar la calidad del software y encontrar errores que no fueron descubiertos con anterioridad.

3.2 Modelo de implementación

El modelo de implementación es una visión general de lo que se debe implementar, este planifica las integraciones de sistemas necesarias en cada iteración, distribuye el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue, implementa las clases y subsistemas encontrados durante el diseño y describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y módulos disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros, probando así los componentes individualmente para después integrarlos (Jacobson, y otros, 2000).

3.2.1. Diagrama de componentes

Un componente es un elemento de funcionalidad del sistema reutilizable que proporciona y utiliza el comportamiento a través de las interfaces (Alrcón, 2008). Los diagramas de componentes proveen una vista arquitectónica de alto nivel del sistema; posibilitando visualizar el camino para la implementación. Los diagramas de componentes son utilizados para modelar la vista estática del sistema, mostrando la organización y las dependencias lógicas entre los componentes, además de que describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable y representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas.

A continuación en la Fig. 13 se muestra el diagrama de componentes referentes al caso de uso Gestionar Profesor donde los componentes visuales (index, new, show, edit) no interactúan directamente con Symfony sino que lo hacen a través de un componente que representa un controlador frontal (app.php).

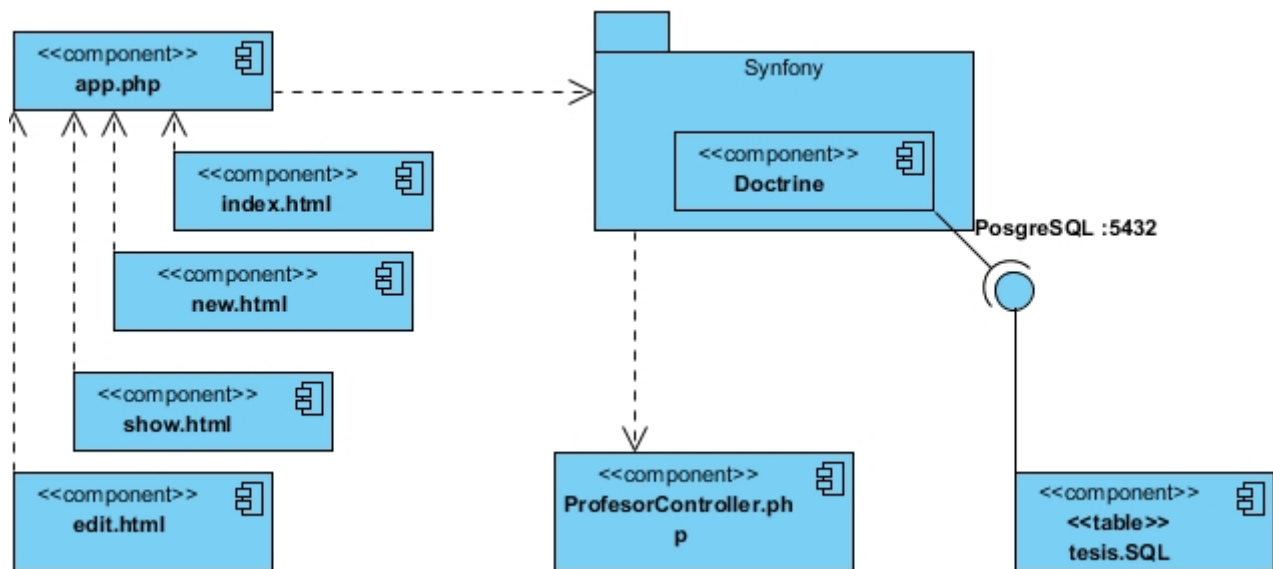


Fig. 13: Diagrama de Componentes Gestionar Profesor

3.3 Modelo de despliegue

Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que componen un sistema y la distribución de los componentes sobre dichos nodos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene algo de memoria y, a menudo, capacidad de procesamiento (Santana Cardoso, 2011).

Los nodos se utilizan para modelar la topología del hardware sobre el que se ejecuta el sistema. Representa típicamente un procesador o un dispositivo sobre el que se pueden desplegar los componentes.

En el diagrama de despliegue (Fig.14) se muestra como la aplicación va a estar alojada en un nodo principal el cual se va a comunicar mediante el protocolo TCP/ IP ¹⁰ con un servidor de bases de datos donde se guarda la información que utiliza el sistema. Además el servidor de aplicación obtiene servicios como la información de autenticación, del directorio activo mediante el protocolo. LDAP¹¹

¹⁰ Protocolo de Control de Transmisión/Protocolo de Internet

¹¹ Protocolo Ligero/Simplificado de Acceso a Directorios

Capítulo 3. Construcción y validación de la solución propuesta

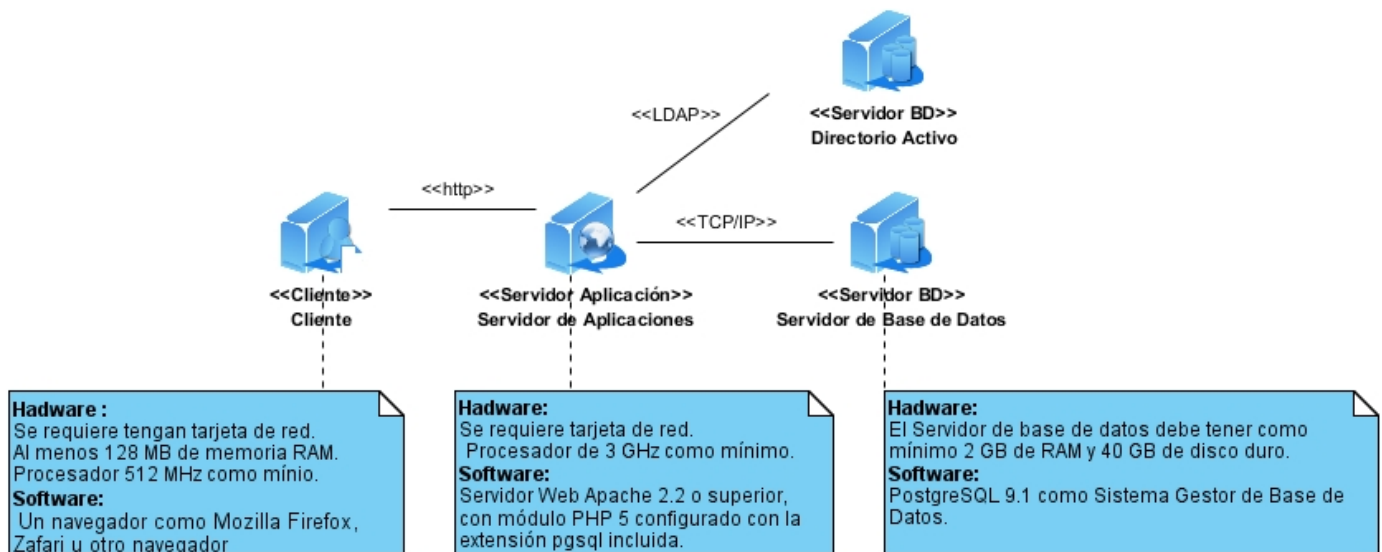


Fig. 14: Diagrama de Despliegue

3.4 Descripción de las pruebas.

La prueba es el proceso de ejecución de un programa con la intención de descubrir un error. Las pruebas del software son un elemento crítico para la garantía de la calidad del software (PRESSMAN, 2005).

Estrategia de Pruebas

Las pruebas de software constituyen un conjunto de tareas y actividades, que pueden ser planificadas y organizadas antes de ejecutarse, mediante modelos y plantillas asegurando la calidad de este proceso de pruebas. Varios autores como Pressman proponen diferentes estrategias de prueba, las cuales presentan las características siguientes (PRESSMAN, 2005):

- ✓ Comienzan a nivel de módulo.
- ✓ Son apropiadas diferentes técnicas de pruebas.
- ✓ La prueba la lleva a cabo el responsable del desarrollo del software y en los casos en que el proyecto es grande un grupo independiente de pruebas.

Una estrategia de prueba de software debe incluir pruebas de alto nivel, las cuales se encargan de validar que los requisitos implantados por el cliente se correspondan con las funcionalidades del sistema.

Niveles de Pruebas (Jacobson, y otros, 2000)

- ✓ *Pruebas Unitarias*: Se basan en detectar errores en los datos, lógica y algoritmos.

Capítulo 3. Construcción y validación de la solución propuesta

- ✓ *Pruebas de Integración:* Se llevan a cabo para detectar errores de interfaces y relaciones entre componentes.
- ✓ *Pruebas Funcionales:* Se llevan a cabo para detectar errores en la implementación de los requerimientos.
- ✓ *Pruebas de Sistema:* Se realizan para detectar fallas en el cubrimiento de los requerimientos.
- ✓ *Pruebas de Aceptación:* Se realizan para detectar fallas en la implementación del sistema.

Tipos de Pruebas

- ✓ *Pruebas de Función:* Este tipo de pruebas se realiza con el propósito de verificar el cumplimiento de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.
- ✓ *Pruebas de Regresión:* Se realizan para asegurar que cuando una no conformidad encontrada en el sistema ha sido corregida ninguna de las funcionalidades liberadas previamente falla como resultado de las correcciones o que las características nuevamente agregadas no han creado conflicto con las versiones anteriores del software.
- ✓ *Pruebas de Integración:* Las pruebas de integración se realizan durante la construcción del sistema, con el objetivo de crear la estructura del programa. Así se comprobaría la unión de todos los componentes mediante sus interfaces, además de asegurar que cumplen con las funcionalidades requeridas.
- ✓ *Pruebas de Rendimiento:* Se realizan generalmente para observar el rendimiento del sistema, para observar el tiempo de respuesta que sea el adecuado; también se pueden utilizar para el trabajo con grandes volúmenes de datos así como para el funcionamiento de la aplicación bajo una cantidad de peticiones esperadas (Pressman, 2000).

Métodos de pruebas

- ✓ *Método de caja blanca:* Se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento.
- ✓ *Método de caja negra:* Se centran fundamentalmente en los requisitos funcionales del software. Estas se realizan sobre la interfaz de la aplicación, con el objetivo de demostrar que las funciones son correctas, que la entrada se acepta de forma adecuada y que se produce una salida correcta (PRESSMAN, 2005).

Para desarrollar la prueba de Caja Negra existen varias técnicas entre ellas:

Capítulo 3. Construcción y validación de la solución propuesta

- ✓ *Técnica de la Partición de Equivalencia*: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- ✓ *Técnica del Análisis de Valores Límites*: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Estas pruebas permiten encontrar: (PRESSMAN, 2005)

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

“El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada, una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica” (Pressman, 2000).

3.4.1. Pruebas de software aplicadas

Las pruebas aplicadas al sistema, según los niveles de pruebas, fueron las *pruebas funcionales*, las mismas están dirigidas a que el sistema cumpla con las expectativas del cliente y para detectar errores en la implementación de los requisitos.

Según el tipo de prueba se seleccionó las *pruebas de función* con el propósito de verificar el cumplimiento de los requisitos funcionales y como método de prueba para probar los requerimientos, las funciones y las respuestas del sistema se decidió utilizar las *pruebas de caja negra*. Se diseñaron además los casos de pruebas por cada caso de uso.

Se selecciona la *técnica de partición equivalente* con el objetivo de dividir el campo de entrada en clases de datos de los que se pueden derivar casos de prueba. Estas divisiones son denominadas diseños de caso de prueba. Permite examinar los valores válidos e inválidos de las entradas existentes en el sistema, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

3.4.2. Diseño de las pruebas para validar la solución propuesta

El diseño de las pruebas se basa en la creación de casos de prueba cuya ejecución permitirá observar posibles síntomas de defectos. Se puede definir un caso de prueba como “el conjunto de entradas,

Capítulo 3. Construcción y validación de la solución propuesta

condiciones de ejecución y resultados esperados desarrollados para un objetivo particular como, por ejemplo, ejercitar un camino concreto de un programa o verificar el cumplimiento de un determinado requisito" (Lara, 2001).

Estos son desarrollados para verificar el cumplimiento total o parcial de un requisito. Las entradas representan las variables que se pueden especificar y las mismas contienen: V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante. A continuación en la tabla 8 se muestra el diseño caso prueba del caso de uso "Gestionar Estudiantes" utilizando la técnica de Partición de Equivalencia, los demás pueden ser consultados en el expediente de proyecto.

Caso de Uso: Gestionar Estudiantes.

Descripción general del caso de uso: El caso de uso inicia cuando el planificador necesita gestionar un estudiante, insertando, modificando o eliminando determinado estudiante y termina cuando se visualiza cambios realizados.

Condiciones de Ejecución: Solo puede realizar esta acción el planificador y debe para ello estar autenticado en la aplicación.

Tabla 8: Caso de Prueba: "Gestionar Estudiante"

Nombre de la sección	Escenario	Descripción	Variables		Respuesta de sistema	Flujo central
			Nombre	Usuario		
SC 1: "Adicionar Estudiante".	EC 1.1: "Adicionar Estudiante" con éxito.	El planificador inserta un Estudiante a partir de sus datos, dicho estudiante se añade al sistema y se muestra en una tabla que lo representa.	V Julio Cortázar	V jcortazar	Se inserta correctamente un Estudiante.	Seleccionar en la Barra de herramientas/ Botón "Adicionar"/Llenar campos /Botón "Adicionar /.

Capítulo 3. Construcción y validación de la solución propuesta

	EC 1.2: “Adicionar Estudiante” con campos vacíos.	Si el planificador no llena los campos correspondientes, el sistema muestra un mensaje de error.	I	V	Emite un mensaje de error: “Debe rellenar todos los campos”.	
	EC 1.3: “Adicionar Estudiante” con variables repetidas.	Si el planificador inserta un valor repetido en alguna variable, se muestra un mensaje de error.	V Julio Cortázar	V jcortazar	Emite un mensaje de error: “Datos Repetidos.”	
	EC 1.4: Volver	El planificador decide volver al listado			Es sistema muestra el listado	
SC 2: “Modificar Estudiante”.	EC 2.1: “Editar un Estudiante” con éxito.	El planificador edita un Estudiante a partir de sus datos, dicho estudiante se guarda al sistema y se muestra una tabla que lo representa.	V Alejandro Dumas	V adumas	Se modifica un Estudiante.	Seleccionar en las opciones del estudiantes/ Botón “Editar”/Llenar campos /Botón “Editar/.
	EC 2.2:	Si el planificador	I	V	Emite un	

Capítulo 3. Construcción y validación de la solución propuesta

	“Editar Estudiante” con campos vacíos.	no llena los campos correspondientes, el sistema muestra un mensaje de error.			mensaje de error: “Debe rellenar todos los campos”.	
	EC 2.3: “Editar Estudiante” con variables repetidas.	Si el planificador inserta un valor repetido en alguna variable, se muestra un mensaje de error.	V Julio Cortázar	V jcortazar	Emite un mensaje de error: “Datos Repetidos.”	
	EC 2.4: Volver	El planificador decide volver al listado			Es sistema muestra el listado	
SC 2: “Eliminar Estudiante”.	EC 3.1: “Eliminar Estudiante”	Si el planificador selecciona eliminar. Se elimina de la base de datos	V Julio Cortázar	V jcortazar	El sistema pide confirmación y muestra la tabla de estudiantes sin el estudiante eliminado.	
	EC 3.2: Volver	El planificador decide volver al listado			Es sistema muestra el listado	

Tabla 9: Caso de Prueba: “Gestionar Estudiante”

No	Nombre del campo	Clasificación	Requerido	Descripción
1	Nombre	Campo de texto	Si	Especifica el nombre del estudiante.
2	Usuario	Campo de texto	Si	Especifica el usuario del estudiante.

3.4.3. Resultados de las pruebas realizadas

Al sistema se le realizaron pruebas de caja negra, de las cuales se exponen sus resultados:

Las pruebas de caja negra se centran en verificar el cumplimiento de los requisitos funcionales del software (PRESSMAN, 2005), con el fin de encontrar la mayor cantidad de no conformidades existentes en el producto. Dentro de las técnicas empleadas por las pruebas de caja negra se utilizó, partición de equivalencia, que según define Pressman: “se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar” (PRESSMAN, 2005).

A continuación se exponen las no conformidades encontradas en las iteraciones de las pruebas realizadas. Las pruebas funcionales al sistema se realizaron en tres iteraciones para verificar que las funcionalidades implementadas fueron las acordadas con el cliente y que respondían correctamente a sus necesidades.

La siguiente grafica en la Fig. 15 muestra el resumen de los defectos encontrados como resultado de estas pruebas.

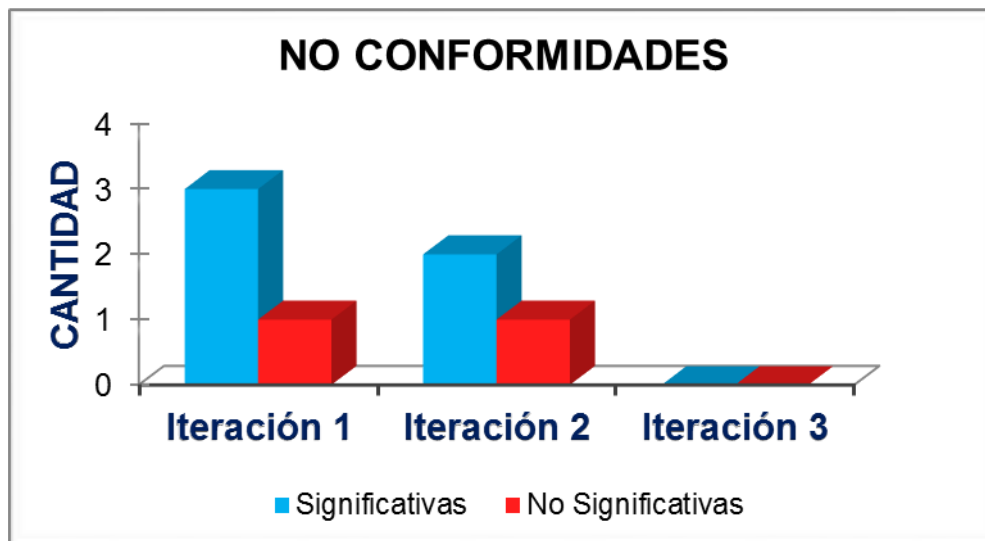


Fig. 15: Resultado de las Pruebas

En la primera iteración de pruebas al sistema se encontraron 4 No Conformidades de las cuales 3 son significativas siendo corregidas todas ellas, luego en la segunda iteración se encontraron 3 No Conformidades de las cuales 2 son Significativas siendo todas corregidas, se procedió a realizar una tercera iteración, la cual arrojó resultados satisfactorios, por lo que no se hizo necesaria la realización de

Capítulo 3. Construcción y validación de la solución propuesta

más iteraciones. Además con dichas pruebas se validaron los requisitos de usabilidad descritas en el capítulo anterior.

Para validar el requisito de seguridad se probó que cada rol tuviera solamente los permisos pertinentes. Adicionalmente se comprobó algunas funcionalidades de Symfony con respecto a diversos tipos de ataque como la inyección SQL y Cross Site Request Forgery (CSRF o XSRF). Como resultado de estas pruebas se determinó que el sistema es seguro.

3.5 Otras pruebas Realizadas

Para asegurar adecuadamente el software conviene aplicar, además de las pruebas funcionales, alguna prueba de tipo estructural (o de caja blanca), centrada en la implementación. La caja blanca son pruebas estructurales, conociendo el código y siguiendo su estructura lógica, se pueden diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas situaciones (María, 2009).

Una de las principales técnicas de diseño de pruebas de caja blanca es la prueba de caminos básicos o método de McCabe que demuestra el conjunto de pasos base del programa, lo que quiere lograr es que cada sentencia de código se ejecute mínimo una vez. El método opera así: (Thomas J. McCabe, 1996)

- Partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico

Para ese caso se muestra el análisis sobre el método CreateAction de la clase Profesor Controller en el cual numeramos cada línea con el número correspondiente a cada nodo.

```
1/public function createAction(Request $request)
{
1/     $entity = new Profesor();
1/     $form = $this->createForm($entity);
1/     $form->handleRequest($request);

2/     if ($form->isValid()) {
3/         $em = $this->getDoctrine()->getManager();
3/         $em->persist($entity);
3/         $em->flush();
```

Capítulo 3. Construcción y validación de la solución propuesta

```
3/     return $this->redirect($this->generateUrl('profesor_show',
3/array('id' => $entity->getId())));
    }

4/     return $this->render('TesisTesisBundle:Profesor:new.html.twig',
4/array(
    'entity' => $entity,
    'form'   => $form->createView(),
    ));
5/ }
```

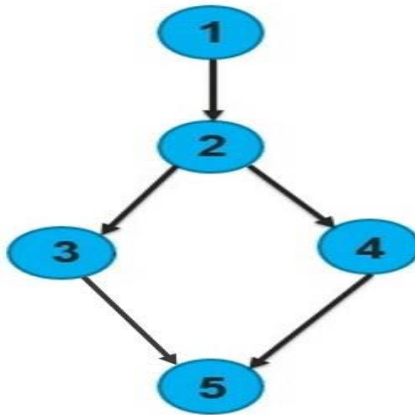


Fig. 16: Grafo del flujo asociado a la funcionalidad createAction

Resultado del cálculo de la complejidad ciclomática:

- El grafo de flujo tiene 2 regiones.
- $V(G) = A - N + 2 = 5 - 5 + 2 = 2$
- $V(G) = P + 1 = 1 + 1 = 2$

A partir del resultado obtenido, se determina que la funcionalidad presenta una complejidad ciclomática de 2, lo que deriva que existen a lo sumo 2 caminos lógicos por donde ejecutarse dicha funcionalidad. En la siguiente tabla se presentan los caminos básicos:

Tabla 10: Caminos básicos

Nro.	Camino básico
1	1, 2, 3, 5
2	1, 2, 4, 5

Capítulo 3. Construcción y validación de la solución propuesta

Tabla 11: Camino básicos 1

Caso de prueba	
Camino	1, 2, 3, 5
Descripción	Adicionar un nuevo profesor.
Condición de ejecución	Si todos los datos están correctos
Entrada	Se introducen los datos del profesor.
Resultados esperados	Muestra una vista con el profesor creado.

Tabla 12: Camino básico 2.

Caso de prueba	
Camino	1, 2, 4, 5
Descripción	Adicionar un nuevo profesor.
Condición de ejecución	Si los datos no son validos
Entrada	Se introducen los datos pertenecientes a la creación del profesor.
Resultados esperados	Se genera una vista para la creación de un nuevo profesor.

3.6 Fundamentación del cumplimiento del objetivo general

El proceso que se pretende informatizar se realiza de forma manual con los inconvenientes ya mencionados en la problemática. El sistema descrito es funcionalmente correcto como resultado de las pruebas de software realizadas y las correcciones de las no conformidades encontradas. Por tanto es posible afirmar que el sistema reduce el tiempo de generación del calendario de defensa de tesis y disminuye la probabilidad de ocurrencia de coincidencia entre los recursos que intervienen en el proceso. Entonces contribuye a favorecer la gestión del calendario de tesis de grado en la Universidad de las Ciencias Informáticas.

3.7 Conclusiones parciales

Podemos arribar a la conclusión d que el sistema implementado se encuentra validado mediante pruebas de caja blanca y caja negra y responde al objetivo general por lo cual reúne las condiciones para su correcta utilización y despliegue en la Universidad.

CONCLUSIONES GENERALES

A partir de los retos planteados en los objetivos, se ha desarrollado un trabajo de investigación que ha permitido llegar a las siguientes conclusiones:

- El estudio de la problemática y de los diferentes métodos de solución de problemas de horario arrojó que los métodos más complejos computacionalmente no siempre brindan una mejor solución, por ello se decidió la implementación de un método de solución ávido.
- Los sistemas estudiados brindaron una visión general de funcionalidades y características que favorecieron la creación de un sistema que se adecue en mejor medida al objetivo de esta investigación.
- Las tecnologías seleccionadas favorecieron el desarrollo, por la familiaridad con el autor de la presente investigación o por estar en correspondencia con la política migración al software libre de la universidad. Estas posibilitaron la implementación de una solución en correspondencia con el objetivo propuesto y en un tiempo considerablemente menor.
- Se validó el sistema que encapsula al método propuesto para la generación de calendario mediante pruebas de caja blanca, caja negra y una fundamentación del objetivo general de la presente investigación dando como resultado un sistema funcional y útil.
- El Sistema de Gestión de Calendario de Tesis constituye una herramienta de software que favorece la planificación y actualización de los tribunales de tesis de grado en la Universidad de las Ciencias Informáticas.

RECOMENDACIONES

- Se recomienda incorporar al sistema funcionalidades que abarquen otras áreas del proceso de tesis, como algún módulo que soporte la recogida de documentos y la planificación de pre-defensas y cortes.
- Capacitar sobre la utilización de la aplicación a los encargados de realizar la planificación.

REFERENCIAS BIBLIOGRÁFICAS

- Hernández Inza , Yadira y Sánche Falero, Héctor René. 2010.** *Gestión de Nomencladores para el Programa Nacional de Informatización del Conocimiento Geológico.* 2010.
- Pavón Mestras, Juan. 2004.** Patrones de diseño. [En línea] 2004. [Citado el: 11 de febrero de 2017.] <http://mit.ocw.universia.net/6.170/6.170/f01/pdf/lecture-12.pdf>.
- Potvin, Jean-Yves y Gendreau, Michael. 2005.** Metaheuristics in combinatorial optimization. *Annals of Operations Research. Operation Research.* 2005.
- Vallecillo, Antonio y Guerequeta, Rosa. 2000.** *Técnicas de Diseño de Algoritmos.* s.l. : Servicio de Publicaciones de la Universidad de Málaga, 2000.
- ALFONSO, H. 2011.** *Resolviendo problemas de optimización con técnicas inteligentes.* s.l. : XIII Workshop de Investigadores en Ciencias de la Computacion., 2011.
- Alrcón, Raúl. 2008.** Diseño Orientado a Objetos con UML. 2008. 2008.
- AUP. 2015.** AUP. [En línea] 2015. [Citado el: 25 de febrero de 2017.] http://ingenieriadesoftware.mex.tl/63758_AUP.html.
- Baltolkien. 2017.** Kde Blog. [En línea] 2017. [Citado el: 5 de mayo de 2017.] <http://www.kdeblog.com/>.
- Boj del Val, Eva y Getán Oliván, Jesús. 2012.** *Programación lineal.* 2012.
- Ceria, Santiago. 2002.** *Casos de Uso un Metodo Práctico para Explorar Requerimientos.* 2002.
- Craig Larman. 1999.** Patrones Grasp. [En línea] 1999. [Citado el: 2017 de enero de 18.] <http://jorgesaaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/>.
- de Werra, D. 1985.** *An introduction to timetabling.* s.l. : European Journal of Operational, 1985.
- Diccionario. 2015.** DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. [En línea] 2015. [Citado el: 16 de junio de 2017.] <http://www.alegsa.com.ar/Dic/aplicacion.php>.
- Eriksson, Hans Erik y Magnus, Penker. 2000.** *Business Modeling with Patterns Business at Work.* s.l. : omgpress, 2000.
- Espinosa, Antonio Membrides, Ramos, Eduanys Puerta y Antunez, Romanuel Ramon. 2012.** *Genesisig. Desarrollo de Sistemas de Información Geográficos sobre una plataforma soberana.* 2012.
- Gauchat, Juan Diego. 2012.** *El Gran Libro de Html, CCS y Javascript.* s.l. : Ediciones Técnicas, 2012.
- Grosan, C y Abrham, A. 2011.** *Intelligent Systems: A Modern Approach. Intelligent Systems.* s.l. : Springer, 2011.
- Hocaoglu, GÄulsÄum y Aladag, Cagdas Hakan. 2007.** *A TABU SEARCH ALGORITHM TO SOLVE A COURSE TIMETABLING PROBLEM.* s.l. : Journal of Mathematics and Statistics, 2007.
- Jacobson, Ivar , Booch, Grady y Rumbaugh, James. 2000.** *Proceso unificado de desarrollo de software.* Madrid : ISBN, 2000. s.n.
- Lara, P. and Fernández, L. 2001.** *Generación de casos de prueba a partir de especificaciones UML.* 2001.
- Larman, Craig. 2003.** *UML y Patrones.* 2003. 2da, Edición..

- Lopez Diaz, Lizzi. 2013.** *Sistema informático de planificación automática de pre-defensas y defensas de los trabajos de diploma en la Facultad 2.* 2013.
- María, L. J. 2009.** *Pruebas de Caja Blanca.* 2009.
- MARTÍ, R. 2005.** *Algoritmos heurísticos en optimización combinatoria.*, in *Departament d'Estadística i Investigació Operativa, Facultat de Matemàtiques.* s.l. : Universitat de València., 2005.
- MARTI, R. 2003.** *Procedimientos metaheurísticos en optimización combinatoria.* s.l. : Matemàtiques, vol. 1., 2003.
- Mejia Caballero, I.J.M. 2008.** *Asignacion de horarios de clases universitarias mediante algoritmos evolutivos.* Division de Universidad del norte Barranquilla : s.n., 2008.
- Nandhini, M y Kanmani, S. May 2009.** *A Survey of Simulated Annealing Methodology for University Course Timetabling.* s.l. : International Journal of Recent Trends in Engineering, May 2009.
- NetBeans IDE. 2015.** [En línea] 2015. [Citado el: 2 de mayo de 2015.] <https://netbeans.org/features/index.html>.
- OSMAN, I.H.K., James. 1996.** *Meta-heuristics: theory and applications.* s.l. : Springer., 1996.
- Osterwalder, Alexander, Pigneur, Yves y Tucci, Christopher. 2005.** *Clarif yin Business Models: Origins, Present, and Future of the Concept.* 2005.
- Palau, Pedro Enrique y Rodríguez, Luis José Tristá. 2012.** *Sistema de Información Geográfica para el Ministerio de Educación Superior.* 2012.
- PASTOR MORENO, R. 2012.** *Metgoritmo de optimización combinatoria mediante la exploración de grafos..* 2012.
- PostgreSQL. 2013.** PostgreSQL. [En línea] 2013. [Citado el: 2 de mayo de 2017.] <http://www.postgresql.org/about..>
- Potencier, Fabien. 2008.** *Symfony 1.1, la guía definitiva.* [En línea] 2008.
- PRESSMAN. 2005.** *Ingeniería del Software: Un enfoque práctico.* 2005. [En línea] 2005.
- Pressman, Roger. 2000.** *Ingeniería de Software un Enfoque Práctico.* 2000. Vol. 6ta Edición.
- Prieto, F. 2008.** *Patrones de diseño.* [En línea] 2008. [Citado el: 11 de febrero de 2015.] http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.
- Programación lineal. Gonzalo Piedrahita, E. 2012.** s.l. : Revista Universidad EAFIT, 2012.
- Rae. 2017.** Real Academia Española. [En línea] 2017. [Citado el: 12 de junio de 2017.] <http://www.rae.es/>.
- Santana Cardoso, Yesenia. 2011.** *Tecnologías de la Información y la Comunicación.* 2011. 2011.
- Sarmiento, Angélica. 2012.** *Programación y asignación de horarios de clases universitarias: un enfoque de programación entera.* 2012.
- Scribd. 2015.** *Arquitectura Basada en Componentes.* [En línea] 2015. [Citado el: 12 de marzo de 2015.] <http://es.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes#scribd>.
- The Apache Software Apache. 2012.** Apache. [En línea] Apache License, 2012. <http://www.apache.org/licenses/LICENSE-2.0.html>.
- Thomas J. McCabe, Arthur H. Watson. 1996.** *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric.* Computer Systems Laboratory National Institute of Gaithersburg : NIST Special Publication, 1996. 500-235.
- Valle, Dianelys del. 2009.** *Desarrollo del Portal WAP para la plataforma de gestión de contenidos Gina.* Habana : s.n., 2009. s.n.
- Velázquez, Catherine Muñoz. 2008.** *Sistema Automatizado para la Gestión Académica – “Akademos 2.0.* 2008.
- Veranes, Maité Sosa. 2009.** *Sistema de Gestión de Tesis Facultad 2.* 2009.

Wei , Pan. 2003. *Understanding types of use cases and artifacts.* s.l. : IBM Developerworks. , 2003.

Wren, A. 1996. *Scheduling, timetabling and rostering --- A special relationship.* s.l. : Practice and Theory of Automated Timetabling, 1996.

Zhipeng , Lü y Hao, Jin-Kao. 2010. *Adaptive tabu search for course timetabling.* s.l. : European Journal of Operational Research 200, 2010.

ANEXOS

Anexo 1. Preguntas de la guía de entrevista:

Entrevistados:

Msc. Yuniel Eliades Proenza Arias jefe del departamento de Práctica Profesional del Centro de Geoinformática y Señales Digitales (GEYSED).

Msc. Adisley Apellidos Reyes Crespo profesora principal de 5to año en Facultad De Ciencias y Tecnologías Computacionales

Objetivos: Conocer las peculiaridades del proceso de planificación de tesis en la Universidad de Ciencias Informáticas (UCI)

Preguntas:

¿Cómo se planifica el proceso de tesis hoy en la Universidad de Ciencias Informáticas (UCI)?

¿Quién es responsable de realizar dicha planificación?

¿En qué momento del año se realiza y que recursos intervienen en dicha planificación?

¿Considera que dicho proceso se realiza de manera eficiente?

¿Qué factores atentan contra la calidad del proceso?

¿Qué medidas pudieran tomarse para erradicar o minimizar dichos factores?

¿Considera que la informatización del proceso pudiera ayudar a mejorar la calidad del mismo?

¿Qué consideraciones habría que tener en caso de que se estime dicha informatización?

¿Qué apariencia considera que debería tener el diseño resultante de dicha informatización?

¿Qué resultados esperaría en la práctica con la utilización de un sistema que informatice dicho proceso?

Anexo 2. No conformidades detectadas al aplicar los casos de prueba.

Tabla 13: No conformidades

Iteración	No. NC	Descripción	Significativas	No Significativas	Ubicación	Estado
1.	1	Falta de ortografía en la ventana Adicionar profesor. Faltan tildes.		x	Barra de herramientas/ opción "Profesor"/nuevo.	Resuelta.
	2	No se alerta cuando hay profesores repetidos en la base de Datos.	x		Barra de herramientas/ opción "Profesor"/nuevo.	Resuelta.
	3	El sistema no guarda los datos del Excel.	x		Barra de herramientas/ opción "Profesor"/nuevo.	Resuelta.
	4	No se alerta cuando los profesores coinciden con los oponentes en Adicionar tesis.	x		Barra de herramientas/ opción "tesis"/nuevo.	Resuelta.
2.	1	El calendario no tiene en cuenta un horario de almuerzo	x		Barra de herramientas/ opción "Calendario"	Resuelta.
	2	El sistema no tiene en cuenta las afectaciones d los profesores para realizar el calendario.	x		Barra de herramientas/ opción "Calendario"	Resuelta.
	3	Algunos errores que se muestran para especificar los valores que el usuario debe entrar son escritos en inglés.		x	Barra de herramientas/ opción "Línea de investigación"/nuevo.	Resuelta.