

Universidad de las Ciencias Informáticas

“Facultad 3”



**Título: Sistema para la gestión del banco de
problemas de la facultad 3.**

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

Autor: Yordano Yunior Osorio Cintra.
Tutor/es: Ing. Tamara Rodríguez Sánchez.
Ing. Danelys Brito González.

Ciudad Habana, Cuba
Junio, 2012
“Año 54 de la Revolución”



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 _____.

Yordano Yuniór Osorio Cintra.

Autor

Ing. Danelys Brito González

Tutor

Ing. Tamara Rodríguez Sánchez

Tutor



FRASE



"La imaginación es más importante que el conocimiento.

El conocimiento es limitado, mientras que la

imaginación no"

Albert Einstein



AGRADECIMIENTOS Y DEDICATORIA

Agradezco:

A mis padres que tanto quiero, por el cariño y el amor que me han brindado durante toda la vida, por ser el motivo de inspiración de llevar todo este sueño adelante y poder siempre contar con ellos, por guiarme y confiar en mí.

A mi hermano Pablo, una de las personas que más quiero en el mundo, y que siempre es un ejemplo a seguir para mí.

A mi novia Tatiana, una de las personas más importantes en mi vida, por estar a mi lado, por brindarme su apoyo en todo momento, y por hacerme feliz cada día....

A mis tutores por su ayuda y su apoyo incondicional en todo momento.

A mi grupo, por todos los momentos maravillosos que pasamos juntos durante la carrera.

A todos mis amigos por haber compartido conmigo todos estos años y pasar juntos buenos y malos momentos.

A todos los que de una forma u otra me ayudaron con la realización de este trabajo.

Dedicatoria:

A mis padres, mi hermano y mi novia.

...muchas gracias,

Yordano

DATOS DE CONTACTO

DATOS DE CONTACTO

Tutora:

- ✓ Nombre y apellidos: Ing. Tamara Rodríguez Sánchez.
- ✓ Correo electrónico: trodriguez@uci.cu.
- ✓ Situación laboral: Profesor.
- ✓ Institución: Universidad de las Ciencias Informáticas (UCI).
- ✓ Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.
- ✓ Rol: Analista principal del proyecto ERP-Cuba del Centro de Informatización de Gestión de Entidades (CEIGE).

Tutora:

- ✓ Nombre y apellidos: Ing. Danelys Brito González.
- ✓ Correo electrónico: dbrito@uci.cu.
- ✓ Situación laboral: Profesor.
- ✓ Institución: Universidad de las Ciencias Informáticas (UCI).
- ✓ Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.
- ✓ Rol: Jefa del departamento Soluciones de productos, perteneciente al Centro de Informatización de Gestión de Entidades (CEIGE).



RESUMEN

RESUMEN

En la Universidad de las Ciencias Informáticas (UCI), específicamente en la facultad 3, no se cuenta con una adecuada gestión de las distintas problemáticas referentes a temas científicos que afectan a la misma, lo que ha sido un impedimento para que estudiantes y profesores desarrollen temas de investigación con un objetivo específico, además de que se han creado soluciones a diferentes problemáticas sin la calidad requerida, puesto que no han sido revisadas por un personal especializado en el tema, lo cual no garantiza la confiabilidad y seguridad de la información, así como que puede darse el caso de que se estén desarrollando temas de investigación o soluciones a problemáticas que ya han sido resueltas con anterioridad. Debido a la situación antes planteada se hizo necesaria la creación de un sistema para la gestión del banco de problemas de la facultad 3 que permita la confiabilidad y seguridad de la información.

Este sistema permite realizar una adecuada gestión de las distintas problemáticas de carácter científico que sean originadas en la facultad 3 de la Universidad de las Ciencias Informáticas. Para el desarrollo del mismo fueron utilizadas varias tecnologías actuales como el lenguaje de programación PHP, el marco de trabajo Sauxe, el cual reutiliza las siguientes tecnologías libres: ExtJS, Zend Framework y Doctrine, además se utilizó como servidor de base de datos PostgreSQL y como servidor de aplicaciones web Apache.

Con el desarrollo de este sistema se logró realizar de forma confiable y segura el proceso de gestión de las distintas problemáticas que afectan a la facultad 3, lo cual posibilitó proporcionarle un seguimiento y control a las mismas por parte de varios especialistas, desde el momento de su creación, hasta una vez solucionadas.

PALABRAS CLAVES: confiabilidad, seguridad, sistema de gestión.

ÍNDICE

ÍNDICE

INTRODUCCIÓN	1
CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA	5
1.1. Introducción	5
1.2. Sistemas de gestión.....	5
1.2.2. Tendencias de los sistemas de gestión de la información a nivel internacional	6
1.2.4. Experiencias en el uso de sistemas de gestión de la información en la UCI	8
1.2.5. Análisis de los sistemas de gestión presentados anteriormente	9
1.3. Modelo de desarrollo.....	10
1.4. Marco de trabajo	12
1.5. Patrones	14
1.6. Técnicas para la captura y validación de los requisitos	19
1.7. Lenguajes de modelado.....	20
1.8. Lenguajes de programación.....	21
1.9. Tecnologías y herramientas de desarrollo.....	23
1.10. Conclusiones parciales del capítulo	27
CAPÍTULO II. ANÁLISIS Y DISEÑO.....	28
2.1. Introducción	28
2.2. Objeto de informatización	28
2.3. Propuesta de solución.....	28
2.4. Modelado del negocio	29
2.5. Modelado del sistema	30
2.5.1. Requisitos funcionales y no funcionales	30
2.5.2. Aplicación de técnicas de validación de requisitos.....	34
2.5.3. Patrones utilizados en el diseño de la aplicación	34
2.5.4. Modelo de datos del sistema	36
2.5.5. Diagrama de clases del diseño.....	43
2.6. Conclusiones parciales del capítulo	45
CAPÍTULO III. IMLEMENTACIÓN Y PRUEBA	46
3.1. Introducción	46

ÍNDICE

3.2. Estructura del marco de trabajo Sauxe	46
3.3. Estándares de código	49
3.4. Descripción de las clases y funcionalidades del sistema.....	50
3.5. Diagrama de despliegue	51
3.6. Validación del diseño propuesto	52
3.7. Pruebas de software aplicadas al sistema	57
3.8. Validación de las variables presentes en el problema	62
3.9. Conclusiones parciales	64
CONCLUSIONES GENERALES	66
RECOMENDACIONES	67
BIBLIOGRAFÍA	68
GLOSARIO DE TÉRMINOS	72

INTRODUCCIÓN

INTRODUCCIÓN

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) es incuestionable y forma parte de la cultura tecnológica que existe en la actualidad y con la que se debe convivir, estas amplían las capacidades físicas y mentales de las personas, así como las posibilidades de desarrollo social [1].

Las 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 ámbitos como la industria, los servicios, la salud, la administración, el comercio y la educación. La evolución de las tecnologías de la información tiende a incorporar, dentro de sus estructuras informáticas, la gestión de la documentación que tradicionalmente se ha conservado en los archivos, es decir en papel [1].

La información se ha convertido en la actualidad en la base del conocimiento y la vía fundamental que tienen las personas, las organizaciones y los países para comunicarse. El cómo gestionar la información es algo que en la actualidad ha tomado prioridad, pues cada vez es mayor la cantidad de información que se necesita manejar. Durante los primeros años del siglo en curso esta necesidad ha tenido un impacto no solo en los métodos y técnicas de gestión de la misma, sino también en la propia tecnología para la gestión de la información.

El éxito de los sistemas de información en una organización depende de la gestión de la información que se realice por las personas que en ella intervienen. Dicha gestión constituye un pilar importante en el desarrollo de una organización.

Un sistema de gestión ayuda a lograr los objetivos de una organización mediante una serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado. En la actualidad existen en el mundo los más diversos sistemas de gestión: desde un simple registro manual de la correspondencia, hasta los más sofisticados sistemas informáticos que manejan no solo la documentación administrativa propiamente dicha, venga ella en papel o en formato electrónico, sino que además controlan los flujos de trabajo del proceso de tramitación de expedientes, capturan información desde bases de datos, contabilidad, enlazan con el contenido de archivos, bibliotecas, centros de documentación y permiten realizar búsquedas sofisticadas y recuperar información [2].

En la Universidad de la Ciencias Informáticas, como parte de las tareas para la informatización de los servicios, el uso de los sistemas de gestión se ha vuelto una realidad. A pesar de que en esta se han creado varios sistemas de gestión como solución a diferentes problemáticas que han surgido en varias de las facultades que la integran, en estos momentos la facultad 3 no cuenta con una adecuada gestión de las distintas problemáticas referentes a temas científicos que

INTRODUCCIÓN

afectan a la misma, lo que ha sido un impedimento para que estudiantes y profesores desarrollen temas de investigación con un objetivo específico. También se han creado soluciones sin la calidad requerida, puesto que no han sido revisadas por un personal especializado en el tema, lo cual no garantiza la confiabilidad y seguridad de la información. Además de que varias personas trabajan por separado dando solución a un mismo problema sin tener conocimiento de ello, así como que puede darse el caso de que se estén desarrollando temas de investigación o soluciones a problemáticas que ya han sido resueltas con anterioridad. De manera general la gestión de las problemáticas originadas en la facultad 3 no se realiza de forma segura, debido a que no se controla, ni se da seguimiento al proceso de desarrollo de las mismas.

De acuerdo con la problemática planteada se define como **problema a resolver**: la actual gestión del banco de problemas de la facultad 3 no tributa a la confiabilidad y seguridad de la información.

El mencionado problema a resolver determina que el **objeto de estudio** de la investigación sea: gestión del banco de problemas de investigación de la UCI.

Por lo tanto para el desarrollo de la investigación se precisó como **objetivo general**: desarrollar un Sistema para la gestión del banco de problemas de la facultad 3 que permita la confiabilidad y seguridad de la información.

Como **campo de acción** se determinó: proceso de gestión del banco de problemas de investigación de la facultad 3.

Como **idea a defender** se tiene: elaborando un Sistema para la gestión del banco de problemas de la facultad 3 se logrará la confiabilidad y seguridad de la información.

Para darle cumplimiento al objetivo general se desglosan los siguientes **objetivos específicos**:

1. Realizar un estudio del estado del arte de sistemas de gestión científica.
2. Desarrollar un sistema que permita la confiabilidad y seguridad de la información.
3. Validar el sistema desarrollado.

Para dar cumplimiento a los objetivos específicos de la investigación se han propuesto las siguientes **tareas de la investigación**:

1. Elaboración del marco teórico de la investigación a partir del estudio del estado del arte.
2. Realización de la modelación del negocio.
3. Validación del modelado de negocio.
4. Realización del levantamiento de requisitos.

INTRODUCCIÓN

5. Especificación de requisitos.
6. Validación de requisitos.
7. Realización del análisis y el diseño de la solución.
8. Validación del diseño propuesto.
9. Implementación de la solución.
10. Validación de la solución mediante pruebas de caja blanca y caja negra.
11. Formalización de los resultados obtenidos en el documento de tesis.

Como **posible resultado** se tiene la obtención de un Sistema para la gestión del banco de problemas de la facultad 3 que permita la confiabilidad y seguridad de la información.

Con el propósito de desarrollar las tareas planteadas para el desarrollo de la investigación se utilizaron los **métodos de investigación** siguientes:

De los métodos teóricos se utilizaron:

Analítico-Sintético: para realizar el análisis de documentos, teorías, permitiendo la extracción de los elementos más importantes que se relacionan con los sistemas de gestión. Además para analizar a través de una profunda búsqueda las tecnologías, herramientas y metodologías a utilizar en el desarrollo de la aplicación.

Histórico-Lógico: para comprobar teóricamente cómo han ido evolucionando los sistemas de gestión en un cierto período de tiempo, a lo largo de su trayectoria o en un fragmento temporal de la lógica de su desarrollo.

Modelación: para definir el negocio durante la fase de análisis, y poder realizar la modelación del mismo durante la fase de diseño.

Dentro de los métodos empíricos se utilizó:

Entrevista: se realiza este método con el objetivo de obtener información sobre la gestión del banco de problemas de investigación de la facultad 3, intercambiando directamente con las personas involucradas en el mismo. Además permite reunir y determinar la información básica necesaria para la recogida de requisitos del sistema.

ESTRUCTURA DEL DOCUMENTO

ESTRUCTURA DEL DOCUMENTO

El presente trabajo de diploma se encuentra estructurado de la siguiente forma: introducción, tres capítulos, conclusiones generales, recomendaciones, bibliografía, anexos y glosario de términos.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

En este capítulo se muestran aspectos fundamentales relacionados con los sistemas de gestión, las diferentes técnicas de captura y validación de requisitos, así como los distintos patrones de diseño y arquitectura que existen. Además se realiza un estudio de los sistemas de gestión de la información, haciendo énfasis en las funcionalidades que debe abarcar el nuevo sistema, en aras de lograr un mejor entendimiento sobre las mismas. También se realiza la selección de las herramientas, lenguajes de programación, servidor de base de datos, marco de trabajo y metodología utilizados para la realización del Sistema para la gestión del banco de problemas de la facultad 3, a través del cual se logrará la confiabilidad y seguridad de la información gestionada en el mismo.

CAPÍTULO 2 ANÁLISIS Y DISEÑO

En el presente capítulo se describe la propuesta de solución del sistema a desarrollar y los procesos que serán informatizados con el desarrollo de esta aplicación. También se presentan los requisitos funcionales y no funcionales con los que debe cumplir el sistema propuesto. Se hace referencia a las técnicas empleadas tanto en la captura como en la validación de los requisitos, así como los patrones utilizados en el diseño de la aplicación. Además se presenta el modelo de datos del sistema y por último se presentan los diagramas de clases del diseño de los procesos mencionados anteriormente.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBA

En este capítulo se realiza una breve descripción del marco de trabajo (framework¹) utilizado, así como los estándares a utilizar durante la implementación del sistema. También se describen las clases y funcionalidades de dicho sistema, además de realizar el diagrama de despliegue para obtener una visión más clara sobre el mismo. Por último se validará el diseño propuesto a través de las métricas TOC y RC, se describirán las pruebas de caja negra y caja blanca realizadas al software y los resultados obtenidos de estas. Se realizará también la validación de las variables confiabilidad y seguridad propuestas en el problema a resolver.

¹ Estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

El término sistema de gestión 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 empresas, escuelas, institutos de investigación y datos de logística. Los sistemas de gestión, ya sean de uno u otro tipo, contribuyen a gestionar, por ejemplo, los riesgos sociales, medioambientales y financieros, mejorar la efectividad operativa, reducir costos y lograr mejoras continuas.

En el desarrollo de este capítulo se muestran aspectos fundamentales relacionados con los sistemas de gestión, las diferentes técnicas de captura y validación de requisitos, así como los distintos patrones de diseño y arquitectura que existen. También se presenta un análisis detallado de la metodología, herramientas y lenguajes de programación que serán empleados para el desarrollo de la aplicación, los cuales fueron seleccionados teniendo en cuenta la plataforma tecnológica existente en la facultad 3 para la inclusión de alguna aplicación en la misma, debido a que una vez realizada esta aplicación será integrada con otras que ya han sido creadas utilizando estas tecnologías. Teniendo en cuenta que esta plataforma tecnológica ofrece un conjunto de herramientas y tecnologías con el fin de lograr un mismo objetivo, en algunos casos se seleccionarán dentro de estas las que más se ajustan para darle solución al problema.

1.2. Sistemas de gestión

A partir del estudio realizado se puede definir como un sistema de gestión a una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de la organización. Este ayuda a lograr los objetivos de la organización mediante una serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado [3]. El uso de un sistema de gestión probado permite renovar constantemente el objetivo, las estrategias, operaciones y niveles de servicio.

1.2.1. Gestión de la información

La información se ha convertido en la actualidad en la base del conocimiento y la vía fundamental que tienen las personas, las organizaciones y los países para comunicarse. La gestión de la información es un proceso que incluye operaciones como extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre la misma. Esta propicia la generación, apropiación, intercambio y uso de

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

conocimientos necesarios para el incremento de la eficacia de las organizaciones, además utiliza la información para el apoyo a la toma de decisiones, organizándola de manera tal que se cumplan los objetivos de la organización [4].

1.2.2. Tendencias de los sistemas de gestión de la información a nivel internacional

Universidad de Barcelona, España. (Sistema GREC)

GREC es una aplicación de gestión de investigación desarrollada por la Universidad de Barcelona, actualmente utilizada por diversas instituciones y organismos de investigación en España. GREC incluye un conjunto de bases de datos como por ejemplo: Currículo Vítae (CV), grupos de investigación, proyectos y publicaciones [5].

Mediante GREC, un organismo, institución o empresa involucrada en actividades de corte científico puede obtener entre otros beneficios:

- ✓ Gestión de convocatorias.
- ✓ Gestión de proyectos, contratos, infraestructuras.
- ✓ Gestión de unidades y agrupaciones de investigación.
- ✓ Definición de actividades de investigación.

Sistema de Información Científica de Andalucía (SICA). España

El Sistema de Información Científica de Andalucía, SICA, puede ser definido como un conjunto de personas, procedimientos y equipos diseñados, construidos, operados y mantenidos para recoger, registrar, procesar, almacenar, recuperar y visualizar información relacionada con las actividades y resultados producidos por los investigadores en sus centros de desarrollo o en colaboración con otras instituciones nacionales o internacionales. Es un sistema de ámbito regional que agrupa la producción científica y que, a medida que esta se genera, es validada en poco tiempo, facilitando un análisis fiable y evolutivo de las políticas científicas [6].

Los módulos o perfiles funcionales construidos en el mismo permiten registrar, validar, almacenar, evaluar, explotar, gestionar y consultar la información relativa a la producción y actividad científico técnica.

En la Universidad de Talca en Chile se realizó un Sistema de Gestión de la Investigación (SGI), en aras de apoyar la investigación que realizan sus académicos por la vía de un sitio web que sirva de punto de encuentro entre la oferta investigativa de la universidad y la demanda de investigación de la sociedad y las empresas, de manera tal que estas planteen temas-problemas susceptibles que puedan ser resueltos a través de programas y/o proyectos de investigación [7].

El sistema es capaz de:

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

- ✓ Mantener actualizados programas y proyectos de investigación, proyectos de tesis y sus consiguientes resultados.
- ✓ Actualizar los estados en que se encuentran los proyectos y la disponibilidad de fondos concursales internos.
- ✓ Desplegar indicadores de gestión asociados a las capacidades y resultados de la investigación que se desarrolla en la Universidad de Talca.

Para que el sistema cumpliera con las expectativas planteadas se consideró esencial implementarlo sobre una plataforma web, razón por la cual se planteó la necesidad de desarrollar un sitio web que soporte el SGI.

Sistema de Gestión de Investigación en Línea (PGIL)

Este sistema pertenece a la División de Gestión de Proyectos de Investigación (DGP-CIUP), Bogotá, Colombia. Desde el año 2004 la Comunidad de Investigadores de la Universidad Pedagógica Nacional (CIUP) cuenta con un completo sistema de gestión administrativa de las convocatorias internas para proyectos de investigación [8].

El mismo, conocido por la comunidad académica como Proceso de Gestión de la Investigación en Línea (PGIL), permite participar en las convocatorias gestionadas por la división. Brinda a todos sus investigadores y personal administrativo un nombre de usuario y contraseña para participar en las distintas convocatorias que se realizan como investigadores, monitores, decanos. De esta manera, se manejan los diferentes perfiles para cada participante. Este va desde la publicación de los términos de referencia y cronograma de la convocatoria, pasando por la inscripción de líneas y grupos de investigación, la conformación de equipos de trabajo, la actualización de las hojas de vida de cada investigador, el diligenciamiento de los proyectos y de las solicitudes de presupuesto y de cargas académicas. También permite el ingreso de diversos tipos de evaluación y la publicación de informes finales del proceso.

1.2.3. Experiencias en el uso de sistemas de gestión de la información científica en Cuba

Sistema de Información para la Gestión de Programas y Proyectos de Ciencia e Innovación Tecnológica (SIPROCIT)

SIPROCIT constituye un sistema de apoyo a la planificación, control, evaluación y proyección de los programas y proyectos de ciencia e innovación, acorde a las prioridades establecidas para un período determinado en Cuba y que contribuye a incrementar el nivel de integración del Sistema de Proyectos y Programas (SPP) [9].

De manera general, con esta aplicación se puede acceder a los listados actualizados de programas y proyectos nacionales, territoriales y ramales, de proyectos no asociados a

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

programas, así como a los datos consolidados, series cronológicas y gráficos estadísticos de programas y proyectos en ejecución y terminados con información pública sobre la planificación, gerencia y participación de los territorios, organismos y entidades [9].

El Sistema de Gestión de la Nueva Universidad (SIGENU)

SIGENU es un sistema que permite la gestión de toda la información de los estudiantes de la Educación Superior en Cuba, desde el momento de su matrícula hasta que se gradúan o causan baja definitiva, incluyendo bajas temporales, licencias, repitencias, reportes evaluativos, cambio del lugar de residencia. Soporta el almacenamiento de información agregada y nominalizada de cualquier estudiante que pertenezca a una institución de Educación Superior de Cuba, aunque no utilice como sistema de gestión académica el proyecto SIGENU [3].

Este sistema ha sido concebido de manera tal que sea capaz de brindar gran seguridad e integridad de la información, y a la vez, ser tan flexible que permita ser adaptado a todos los centros de Educación Superior del país con sus diversas particularidades y distintas maneras de realizar determinados procedimientos.

1.2.4. Experiencias en el uso de sistemas de gestión de la información en la UCI

Sistema de indicadores cientiométricos (SIndiCIT)

Es un sistema de indicadores que permite evaluar la producción científica de los profesores, investigadores y estudiantes de la universidad, el cual fue desarrollado por la Dirección de Investigaciones de la Universidad de las Ciencias Informáticas [10].

Este sistema adaptó el Sistema de Indicadores de Ciencia, Tecnología e Innovación (CTI) vigente en las instituciones y universidades del Ministerio de Educación Superior (MES) y el Ministerio de Ciencia Tecnología y Medio Ambiente de Cuba (CITMA) a las condiciones existentes en la UCI, que consisten fundamentalmente en que la formación curricular y el núcleo fundamental de las investigaciones científicas que se desarrollan se centran en las ramas mencionadas [10].

Sistema de Gestión de Investigación de la facultad 2 de la Universidad de las Ciencias Informáticas, el cual tiene como objetivo principal automatizar los procesos de gestión de la información asociada a la investigación y el postgrado en esta facultad, logrando una mayor rapidez en el manejo de la información referente a ambos procesos.

Dentro de las funcionalidades que este presenta están: gestionar proyectos de investigación, gestionar cursos de postgrado, gestionar líneas de investigación, generar reportes, gestionar grupos de investigación, gestionar líneas temáticas, realizar solicitud de cursos de postgrado y realizar solicitud de programas de postgrado.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.2.5. Análisis de los sistemas de gestión presentados anteriormente

A continuación se realiza una tabla donde se muestran las principales funcionalidades de la aplicación a desarrollar, y de estas cuáles están contenidas de una forma u otra en los sistemas estudiados.

F	SG	GP	RP	GS	CP	GC	GN	GR
	GREC	X	X			X		X
	SICA	X	X					X
	SIGI	X		X		X		
	PGIL	X				X		X
	SIPROCIT	X	X			X		X
	SIGENU							X
	SIndiCIT	X		X				X
	SIGI facultad 2	X	X		X			X

Tabla 1. Funcionalidades convergentes entre los sistemas.

Leyenda

- ✓ Funcionalidades: F, sistemas de gestión: SG, gestionar problemática: GP, revisar problemática: RP, gestionar salidas: GS, contratar problemática: CP, gestionar cronograma: GC, generar notificaciones: GN, generar reportes: GR.

A pesar de los diferentes sistemas informáticos que se han identificado con el propósito de gestionar la actividad científica de una organización, no ha sido posible la adopción de ninguno de estos como solución al problema planteado, debido a que no incluyen las funcionalidades contratar problemáticas y generar notificaciones, las cuales deben estar presentes en el sistema a desarrollar, además de que varias de las funcionalidades que estos contemplan no se corresponden con el proceso de seguimiento que se le desea dar a cada una de las problemáticas originadas en la facultad 3.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Otro conjunto de estos sistemas han sido desarrollados para darle solución a problemáticas presentes en determinadas instituciones, los cuales no se encuentran disponibles, es decir no están de forma libre para su adopción.

En el ámbito nacional, la mayoría de estos sistemas constituyen portales web informativos, los cuales no están concebidos para la gestión de la información relacionada con las problemáticas de carácter científico de una institución, por lo que no cumplen con las necesidades identificadas.

Por lo que se propone el desarrollo de un Sistema para la gestión del banco de problemas de la facultad 3, el cual tiene como particularidad que se ajusta sobre todo a los intereses y estrategias de trabajo de dicha facultad.

1.3. Modelo de desarrollo

Para el desarrollo del presente trabajo de diploma se sigue el modelo de desarrollo definido por el equipo de producción del proyecto ERP-Cuba. Se hace uso de este modelo debido a que el mismo está basado en RUP² y XP³, dos de las metodologías más usadas en la construcción de software tomando de estas las características más notables [16], el cual se implantó hace poco más de dos años obteniendo resultados satisfactorios en el Centro de Informatización de la Gestión de Entidades (CEIGE), perteneciente a la facultad 3, en donde va a ser incluida la aplicación a realizar.

Este modelo está basado en componentes, centrado en la arquitectura, ágil y adaptable al cambio y es iterativo e incremental. A continuación se explicará brevemente el significado de cada una de las características mencionadas anteriormente [17]:

✓ **Centrado en la arquitectura**

La arquitectura determina la línea base y los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades en la producción y resuelve las necesidades tecnológicas y de soporte para el desarrollo.

✓ **Orientado a componentes**

Las iteraciones son orientadas según la significación arquitectónica de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.

✓ **Iterativo e incremental**

² Proceso Unificado de Rational (Rational Unified Process).

³ Programación extrema (eXtreme Programming).

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.

✓ **Ágil y adaptable al cambio**

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

1.3.1. Descripción del ciclo de vida

A continuación se explica brevemente los objetivos de cada una de las fases que integran el ciclo de vida del modelo de desarrollo basado en componentes:

Concepción:

Los objetivos de esta fase son:

- ✓ La evaluación de factibilidad del proyecto y con ello la elaboración y aprobación del proyecto técnico, definiendo los macro requisitos y macro componentes.
- ✓ Definir los procesos de negocios e identificar los requisitos. Definir la línea base de los requisitos del proyecto.

El hito de la fase Concepción es concebir el alcance del proyecto.

Elaboración:

Los objetivos de esta fase son:

- ✓ La firma del Acuerdo de Colaboración o Contrato.
- ✓ Se describen los requisitos y se realiza el diseño arquitectónico de la solución.

El hito de esta fase es establecer la línea base de la arquitectura.

Construcción:

- ✓ El objetivo de esta fase es implementar las funcionalidades del sistema informático.

El hito de esta fase son las pruebas internas y de liberación del sistema informático.

Cierre:

Los objetivos de esta fase son:

- ✓ Realizarle las pruebas piloto y de regresión al software.
- ✓ Garantizar la transferencia del sistema al cliente.
- ✓ Mantenimiento al software.

El hito de esta fase es la transferencia de una versión estable junto a su documentación al cliente del sistema informático.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.4. Marco de trabajo

El desarrollo de aplicaciones informáticas requiere de una serie de pasos para organizar dicho desarrollo y hacerlo estructuradamente. Los desarrollos en entornos web, actualmente crecientes, y debido a los procesos que deben manejar, se hace necesario contar con un software capaz de administrar, organizar y manejar ciertos procesos en la etapa de desarrollo. Este software es lo que se conoce como marco de trabajo, que es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado [15].

La plataforma tecnológica de la facultad 3 brinda la posibilidad de utilizar los siguientes marcos de trabajo para el desarrollo de cualquier aplicación: CodeIgniter, Symfony y Sauxe. Se realizó un estudio sobre los mismos a través del cual se determinó que tienen varias características comunes entre las que se encuentran que todos son marcos de trabajo para el desarrollo web, se basan en el patrón Modelo – Vista – Controlador (MVC), son multiplataforma y pueden utilizar como sistema gestor de base de datos PostgreSQL. Sin embargo Symfony usa PHP 5, tiene un alto consumo de memoria y presenta una fuerte curva de aprendizaje, lo que podría disminuir la productividad del equipo de desarrollo durante los primeros meses, por lo que no sería posible su uso teniendo en cuenta el tiempo disponible para realizar la aplicación. CodeIgniter tiene un bajo consumo de memoria, presenta una curva de aprendizaje media debido a que los usuarios se ven obligados a aprender nuevas funciones, estructuras y métodos de programación, además de su empeño por seguir soportando PHP 4 impidiéndole tomar ventaja de las mejoras que trajo este lenguaje en su última versión, limitante por la cual teniendo en cuenta que la aplicación debe realizarse con PHP 5.4 no es posible la utilización del mismo. En el caso del marco de trabajo Sauxe presenta un bajo consumo de memoria, usa PHP 5 y su curva de aprendizaje es media.

Por todo lo antes planteado para el presente trabajo de diploma se hace uso del marco de trabajo Sauxe debido a que se ajusta a todas las características necesarias para llevar a cabo la aplicación a desarrollar. Este fue creado por el departamento de Tecnología del centro CEIGE de la Universidad de las Ciencias Informáticas. Contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo [9]. Con la utilización del mismo desde el inicio del desarrollo de un proyecto se tiene garantizado aspectos como la seguridad, la multi-entidad, el multi-tema, el multi-idioma, la auditoría, la integración, la interoperabilidad y la concurrencia. Por las ventajas que proporciona mencionadas anteriormente es utilizado en más de 20 proyectos de la UCI y 10 entidades desarrolladoras de software [16].

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, reutiliza las siguientes tecnologías libres:

1.4.1. ExtJS

En el mercado actualmente existen múltiples librerías de Javascript que permiten realizar todo tipo de maravillas en el navegador web. ExtJS permite que con pocas líneas de código sea posible realizar interfaces amigables para los usuarios. Es la librería más avanzada para el desarrollo rápido de aplicaciones con una apariencia totalmente novedosa y una arquitectura flexible.

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos, así como un manejador de layouts similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox e Internet Explorer) [16].

Usar ExtJS permite tener además otros beneficios entre los que se encuentran:

- ✓ Existe un balance entre cliente – servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- ✓ Comunicación asíncrona. Este tipo de aplicación puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.
- ✓ Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija qué información desea transmitir al servidor y viceversa.

1.4.2. Zend Framework

Se trata de un framework para el desarrollo de aplicaciones web y servicios web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. Este está formado por una serie de métodos estáticos y componentes que usarán estos métodos. Entre los componentes que se encuentran tienen vital importancia: Zend_Config para temas de configuración de aplicaciones web, Zend_Db para tratar con bases de datos, Zend_Search o Zend_Feed entre otros [16]. El mismo presenta dentro de sus características principales:

- ✓ Trabaja con el patrón MVC.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

- ✓ Cuenta con módulos para manejar archivos PDF, canales RSS⁴, Web Services (Amazon, Yahoo) y presenta un soporte avanzado.
- ✓ Robustas clases para autenticación y filtrado de entrada.
- ✓ Clientes para servicios web.

1.4.3. Doctrine

Doctrine es un Object Relational Mapping (ORM) para la persistencia de datos que trabaja con lenguaje de programación PHP 5.2.3 o superior, situándose sobre su poderosa capa de abstracción de la base de datos (PHP DBAL). Es uno de los más conocidos que interactúa con este lenguaje y su utilización brinda grandes facilidades a los desarrolladores a la hora de su trabajo con los datos, automatizando la generación de ficheros que responden a las tablas relacionales de la base de datos [17].

Principales características de Doctrine

Doctrine se divide en dos capas fundamentales que interactúan en conjunto, la DBAL del inglés Database Abstraction Layer o Capa de Abstracción de la Base de Datos y la compuesta por el ORM reflejadas en la siguiente imagen.

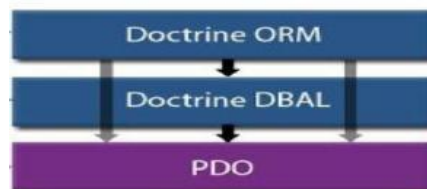


Figura 1. Estructura lógica del framework de persistencia Doctrine. [18]

Este framework tiene la facultad de construir consultas a la base de datos relacional utilizando un lenguaje OO (orientado a objetos) denominado DQL (Doctrine Query Language o Lenguaje de Consultas Doctrine). Además, implementa un patrón CRUD (Crear, Recuperar, Actualizar y Eliminar) para operaciones comunes, ya sea desde crear un nuevo registro o actualizar los registros existentes [18] y dentro de los ORM que dispone PHP es uno de los más populares.

1.5. Patrones

“Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos sobre cómo aplicarlo en nuevas situaciones, o sea, un patrón es una

⁴ Really Simple Syndication: Formato XML para syndicar o compartir contenido en la web.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

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” [19].

Existen diversos tipos de patrones, entre ellos se encuentran los de arquitectura y de diseño. A continuación se explica brevemente en qué consiste cada uno de estos y se muestran varios ejemplos de los mismos.

1.5.1. Patrones de Diseño

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables [20].

Patrones GRASP

Los Patrones de Software para la Asignación General de Responsabilidad (GRASP), representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades) [21].

GRASP destaca 5 patrones principales: Bajo acoplamiento, Experto, Alta cohesión, Creador y Controlador. A continuación se explica brevemente en qué consiste cada uno de estos: [19]

Bajo Acoplamiento: Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas.

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Hay que tener en cuenta que esto es aplicable mientras se estén considerando los mismos aspectos del sistema:

- ✓ Lógica de negocio
- ✓ Persistencia a la base de datos
- ✓ Interfaz de usuario

Alta Cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Ejemplos de una baja cohesión son clases que hacen demasiadas cosas. En todas las metodologías se considera la refactorización. Uno de los elementos a refactorizar son las clases saturadas de métodos. Ejemplos de buen diseño se producen cuando se crean los denominados “paquetes de servicio” o clases agrupadas por funcionalidades que son fácilmente reutilizables (bien por uso directo o por herencia).

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Creador: Se asigna la responsabilidad de que una clase B cree un objeto de la clase A solamente cuando:

- ✓ B contiene a A
- ✓ B es una agregación (o composición) de A
- ✓ B almacena a A
- ✓ B tiene los datos de inicialización de A (datos que requiere su constructor)
- ✓ B usa a A

La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulamiento y reutilización.

Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

Patrones GOF

Los patrones de diseño GOF⁵ se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

Patrones de creación: Los patrones de creación proporcionan ayuda a la hora de crear objetos, principalmente cuando esta creación requiere tomar decisiones. Un patrón de creación asociado a clases usa la herencia para variar la clase que se instancia, mientras que un patrón de creación asociado a objetos delegará la instanciación a otro objeto [22]. A continuación se muestran varios patrones de creación: [20]

- ✓ Abstract Factory: proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.
- ✓ Builder: separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.
- ✓ Prototype: especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.
- ✓ Singleton: garantiza que una clase solo tenga una instancia, y proporciona un punto de acceso global a ella.

⁵ Gang of Four(Banda de los cuatro): Nombre con el que se conoce comúnmente a los autores del libro Design Patterns.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Patrones estructurales: Los patrones estructurales están relacionados con cómo las clases y los objetos se combinan para dar lugar a estructuras más complejas. A continuación se muestran varios patrones estructurales: [20]

- ✓ Adapter: convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.
- ✓ Bridge: desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.
- ✓ Decorator: añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.
- ✓ Facade: proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.

Patrones de comportamiento: Estos patrones de diseño están relacionados con algoritmos y asignación de responsabilidades a los objetos. Los patrones de comportamiento describen no solamente patrones de objetos o clases, sino también patrones de comunicación entre ellos. Dentro de los patrones de comportamiento se encuentran: [20]

- ✓ Chain of Responsibility (Cadena de Responsabilidad): evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.
- ✓ Mediator (Mediador): define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.
- ✓ Observer: define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.
- ✓ Visitor: representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

1.5.2. Patrones arquitectónicos

Un patrón arquitectónico representa, en términos de componentes y las relaciones entre ellos, posibles soluciones para incorporar en el diseño, aspectos que mejoran la usabilidad del sistema final [23]. Dentro de los patrones de arquitectura se encuentra el patrón Modelo – Vista - Controlador (MVC).

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

En el desarrollo de la aplicación informática propuesta se utilizará el marco de trabajo Sauxe, el cual está basado en el patrón clásico MVC. Este 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 los conceptos para que el desarrollo esté estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente [24].

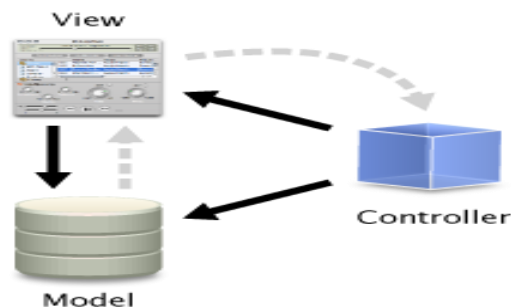


Figura 2. Estructura del patrón MVC.

Modelo: encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada. El modelo es el responsable de:

- ✓ Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- ✓ Definir las reglas de negocio (la funcionalidad del sistema).
- ✓ Llevar un registro de las vistas y controladores del sistema.
- ✓ Si se encuentra ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

Vista: intercambia la información con el usuario. Pueden existir múltiples vistas del modelo.

Cada vista tiene asociado un componente controlador. Las vistas son responsables de:

- ✓ Recibir datos del modelo y mostrarlos al usuario.
- ✓ Tienen un registro de su controlador asociado.
- ✓ Pueden dar el servicio de actualización, para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Controlador: Recibe las entradas, traducidas a solicitudes de servicio para el modelo. El controlador es responsable de:

- ✓ Recibir los eventos de entrada.
- ✓ Contiene reglas de gestión de eventos, del tipo "Si Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.6. Técnicas para la captura y validación de los requisitos

La identificación de los requisitos que debe cumplir un software es una actividad que se lleva a cabo desde el inicio del desarrollo del sistema. En este proceso los analistas extraen de diferentes fuentes de información los datos que son necesarios para conocer las funcionalidades que implementará el sistema, por lo que se hace necesario por parte de los analistas el empleo de técnicas que permitan establecer una buena comunicación con los interesados del producto y así lograr la satisfacción del cliente. También existen técnicas para la validación de los requisitos las cuales tienen como objetivo general demostrar que su definición es la que el usuario final necesita.

Entre las técnicas para llevar a cabo la **captura de requisitos** se encuentran:

- ✓ Entrevistas

Es una de las técnicas más usadas en la captura de requisitos. Consiste en establecer una conversación entre personas de ambas partes. Estas son aplicadas a los especialistas funcionales.

- ✓ Tormenta de ideas (brainstorming)

Reunión de varios interesados en la que todos expresan sus ideas sobre el problema y su solución. La forma de llevarla a cabo es que cada participante diga su idea sin ser interrumpido por otro. Al finalizar la sesión de lluvia de ideas se puede hacer una recolección de ideas sin duplicidad.

- ✓ Análisis de protocolos

La técnica consiste en pedirles a los usuarios potenciales que describan en voz alta las actividades que realizan dentro del sistema.

En cuanto a las técnicas que se utilizan para la **validación de los requisitos** se pueden mencionar:

- ✓ La Revisión Técnica Formal (RTF)

Son reuniones del personal técnico (usuario final del sistema) con el objetivo de validar la especificación de requisitos. Su aplicación a los documentos de práctica permitirá detectar deficiencias, ambigüedades, omisiones y errores, tanto de formato como de contenido. Suelen realizarla entre 3 y 5 personas y los especialistas funcionales deben ser independientes del equipo que ha realizado la especificación de requisitos.

- ✓ Auditorías

Revisar la documentación con esta técnica, consiste en un chequeo de los resultados contra una (Listas de Chequeo) predefinida o definida a comienzos del proceso, es decir solo una muestra es revisada.

- ✓ Reviews o Walk-throughs

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Esta técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar consistencia de la documentación o información faltante.

1.7. Lenguajes de modelado

La construcción de cualquier proyecto de ingeniería requiere de etapas de modelación que permitan experimentar y visualizar el sistema que se construirá. Uniendo varios conceptos y teorías, se puede conceptualizar un lenguaje de modelado como una estandarización de notaciones y reglas, que permitan graficar un sistema o parte de él.

1.7.1. Business Process Modeling Notation (BPMN)

El modelado de procesos de negocio requiere la representación de los elementos que intervienen en los procesos. Por lo tanto, es necesario utilizar un lenguaje o notación que permita modelar dichos procesos. BPMN, del español Notación para Modelado de Procesos de Negocio, es una notación gráfica que describe la lógica de los pasos de un proceso de negocio. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades [25].

El mismo es un estándar internacional de modelado de procesos, el cual es independiente de cualquier metodología de modelado de proceso. Este crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos, y permite modelar los procesos de una manera unificada y estandarizada, permitiendo un entendimiento a todas las personas de una organización.

1.7.2. Lenguaje Unificado de Modelado (UML)

UML es un lenguaje estándar para el modelado de sistemas de software, el cual es el lenguaje de modelado más conocido y utilizado en la actualidad y se utiliza para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos [26]. Este sirve como puente entre la especificación de requisitos y la implementación, provee un significado para especificar y documentar diseños de un sistema de software, además de que el mismo está particularmente preparado para desarrollo de programas orientados a objetos.

En UML se identifican:

- ✓ Elementos (Abstracciones que constituyen los bloques básicos de construcción)
- ✓ Relaciones (Ligan los elementos)
- ✓ Diagramas (Representación gráfica de un conjunto de elementos)

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Es importante destacar que UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso, sino un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

1.8. Lenguajes de programación

Un lenguaje de programación es el medio que permite a las computadoras, interpretar las órdenes que se les dan, así como controlar su comportamiento. Este consiste en un grupo de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos respectivamente. A través de estas reglas el programador crea los programas o subprogramas. Los lenguajes de programación para el desarrollo de aplicaciones web, están divididos en dos grandes grupos, los lenguajes del lado del servidor y los lenguajes del lado del cliente.

1.8.1. Lenguaje del lado del servidor

Se clasifica así al lenguaje de programación en la tecnología cliente servidor que se ejecuta del lado del servidor y del cual los usuarios solo obtienen el beneficio del procesamiento de la información.

✓ PHP (Hipertexto Pre-processor)

PHP es una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos, puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, soporta la mayoría de los servidores web de hoy en día y ofrece soporte para unos 20 gestores de bases de datos [27].

Su característica de ser software libre trae como consecuencia que implique menos costes y servidores más baratos. Es además muy rápido, contiene una biblioteca nativa de funciones sumamente amplia e incluida, no requiere definición de tipos de variables ni manejo detallado del bajo nivel, presenta mejoras de rendimiento, es un lenguaje multiplataforma que funciona en todas las plataformas que soporten Apache. No es un lenguaje de marcas y su principal meta es permitir rápidamente a los desarrolladores la generación dinámica de páginas. Soporta el uso de otros servicios que usen protocolos como SNMP⁶, HTTP⁷ y derivados y permite generar documentos en PDF (documentos de Acrobat Reader) [27].

⁶ Protocolo simple de Administración de Red.

⁷ Protocolo de transferencia de hipertexto.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.8.2. Lenguajes del lado del cliente

Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad [37].

✓ HTML

HTML es el lenguaje estándar para la web, y fue definido por una organización internacional de estandarización (el Consorcio W3). Este es un formato universal flexible, rico y compacto [28]. HTML es un conjunto de símbolos o palabras que definen varios componentes de un documento web. Es un lenguaje simple de marcado utilizado para crear documentos de hipertexto para WWW⁸.

El mismo no solo permite establecer hiperenlaces entre diferentes documentos, sino que es un lenguaje de descripción independiente de la plataforma en que se utilice. Este contiene toda la información necesaria sobre su estructura y con el usuario, es luego el navegador que se utilice, el responsable de asegurar que el documento tenga un aspecto coherente, independiente del tipo de máquina desde donde se acceda al documento. Entre las ventajas del mismo se encuentra el hecho de ser muy fácil de aprender, lo que permite que cualquier persona, pueda enfrentarse a la tarea de crear una web. Este permite utilizar estilos en formato CSS en las páginas para una mayor facilidad en su modificación y los contenidos son fáciles de actualizar.

✓ Java Script

Java Script es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. Puede ser utilizado por profesionales y para quienes se inician en el desarrollo y diseño de sitios web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos. Es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con Java Script se puede crear diferentes efectos e interactuar con nuestros usuarios [29].

Java Script tiene como principal característica ser un lenguaje independiente de la plataforma. Entre otra de sus características se tiene que, aunque el lenguaje soporta cuatro tipos de datos, no es necesario declarar el tipo de las variables, argumentos de funciones ni valores de retorno de las funciones. Además, es un lenguaje que utiliza Windows y sistemas X-Windows, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas y cargas de páginas [29].

⁸ World Wide Web o Red de Amplitud Mundial.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

✓ XML

Es el metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos. No es un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. XML no ha nacido solo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Es una tecnología en realidad muy sencilla que tiene a su alrededor otras tecnologías que la complementan y la hacen mucho más grande y con unas posibilidades enormes y básicas para la sociedad de la información [30].

XML, con todas las tecnologías relacionadas, representa una manera distinta de hacer las cosas, más avanzada, cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Así pues, el XML juega un papel importantísimo en este mundo actual, que tiende a la globalización y la compatibilidad entre los sistemas, ya que es la tecnología que permitirá compartir la información de una manera segura, fiable y fácil [30].

1.9. Tecnologías y herramientas de desarrollo

La selección correcta de las herramientas y tecnologías que se utilizan en el desarrollo de un software se traduce en ahorro de tiempo y trabajo dentro de cualquier proyecto. A continuación se realiza una breve descripción de las tecnologías y herramientas que serán utilizadas en el desarrollo de la aplicación.

1.9.1. Tecnología AJAX

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. Java Script es el lenguaje interpretado (scripting lenguaje) en el que normalmente se efectúan las funciones de llamada de AJAX mientras que el acceso a los datos se realiza mediante XMLHttpRequest⁹, objeto disponible en los navegadores actuales [31].

⁹ (Extensible Markup Language / Hypertext Transfer Protocol).

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.9.2. Servidor de aplicaciones web Apache

Un servidor web es el programa que, utilizando el protocolo de comunicaciones HTTP, es capaz de recibir peticiones de información de un programa cliente (navegador), recuperar la información solicitada y enviarla al programa cliente para su visualización por el usuario [32].

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa [33].

Entre sus principales características se encuentran:

- ✓ Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- ✓ Es una tecnología gratuita de código fuente abierto.
- ✓ Es un servidor altamente configurable de diseño modular. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que se instalen cuando se necesite.
- ✓ Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.
- ✓ Tiene una alta configurabilidad en la creación y gestión de logs.

1.9.3. Servidor de Base de Datos PostgreSQL

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS). Este está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo [34]. El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto. Este proporciona un gran número de características que normalmente solo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. La siguiente es una breve lista de algunas de esas características: [34]

- ✓ DBMS Objeto-Relacional: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas.
- ✓ Altamente_Extensible: PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- ✓ Lenguajes Procedurales: PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL.

1.9.4. Herramienta Case

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

en el desarrollo de software, reduciendo el coste de las mismas en términos de tiempo y de dinero.

Rational Rose

Es una herramienta CASE (Computer – Aided Software Engineering), desarrollada por Rational Corporation basada en el Lenguaje Unificado de Modelado, que permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo del software. Dentro de sus características se encuentran que admite notaciones UML, OMT y Booch, permite desarrollo multiusuario, genera documentación del sistema y está disponible en múltiples plataformas. Esta herramienta también presenta desventajas, por ejemplo que necesita de mucha memoria para poder de alguna forma ser manejado de manera rápida y eficiente, además de que el costo de sus licencias es elevado.

Visual Paradigm for UML

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue [35]. Ayuda a una rápida construcción de aplicaciones de gran calidad, mejoras y a un menor costo de producción. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además, incluye una herramienta llamada Visual Architect que permite la generación de código para el manejo de la base de datos y puede generar códigos para lenguajes como PHP, Java y gestores de base de datos PostgreSQL y MySQL.

Dentro de sus características se encuentran:

- ✓ Soporte de UML versión 2.1.
- ✓ Generación de bases de datos: transformación de diagramas de Entidad-Relación en tablas de base de datos.
- ✓ Ingeniería inversa de bases de datos: desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.

Analizando lo referente a estas herramientas se escoge como herramienta de modelado Visual Paradigm for UML, debido a que sustenta el lenguaje de modelado UML con una amplia documentación y todos sus diagramas de diseño, además de que es muy rápida en su tiempo de ejecución y ayuda a una ágil construcción de aplicaciones de gran calidad, mejoras y a un menor costo de producción.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.9.5. Entorno de desarrollo integrado (IDE)

Un entorno o ambiente integrado de desarrollo (Integrated Development Environment) es un programa compuesto por un conjunto de herramientas para un programador, que puede dedicarse a un solo lenguaje de programación o utilizarse para varios. Estos suelen incluir en una misma suite, un buen editor de código, administrador de proyectos y archivos, enlace transparente a compiladores y debuggers e integración con sistemas controladores de versiones o repositorios.

Eclipse

Eclipse es un entorno integrado de desarrollo de código abierto y multiplataforma, creado originalmente por la IBM Canadá y actualmente desarrollado por la Fundación Eclipse. Este no es tan solo un IDE, se trata de un marco de trabajo modular ampliable mediante complementos (plugins). El mismo permite la realización tanto de aplicaciones web como de aplicaciones de escritorios y existen complementos que permiten usar Eclipse para programar en PHP, Perl, java, Python y C/C++. Provee soporte para Java y SVN¹⁰.

NetBeans

NetBeans es un entorno integrado de desarrollo (IDE), para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación como Java, C/C++, PHP y Java Script. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Estos a su vez pueden ser desarrollados independientemente, razón por la que las aplicaciones basadas en NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

Dentro de sus características se pueden encontrar: [36]

- ✓ Mejoras en el editor de código
- ✓ Características visuales para el desarrollo web
- ✓ Soporte para PHP
- ✓ Permite desarrollar aplicaciones de escritorio, web, Mobile, Enterprises.

NetBeans IDE es un producto libre y gratuito sin restricciones de uso que permite a los desarrolladores crear rápida y fácilmente aplicaciones de gran calidad.

Luego del estudio realizado referente a las características fundamentales de las dos herramientas mencionadas anteriormente se escoge como entorno integrado de desarrollo el

¹⁰ Subversion: Software de sistema de control de versiones.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

NetBeans, debido a que el mismo brinda soporte para PHP, lenguaje de programación del lado del servidor que será utilizado en el desarrollo de la aplicación, además de que presenta una gran mejora en su editor de PHP, el cual es mucho más ágil y a la vez robusto y contiene más ayuda en línea, reconocimiento de sintaxis y todo lo que provee la última versión de este lenguaje. También a la hora de debuggear es muy sencillo, permite ver todas las variables del programa en tiempo de ejecución y los errores son bastante descriptivos, lo que ayuda a un mayor entendimiento de los mismos.

1.9.6. Navegador Mozilla Firefox

Mozilla Firefox es un navegador de código abierto, multiplataforma, basado en el código de base de Mozilla que proporciona una navegación más rápida, segura y eficiente que otros navegadores [37]. Entre sus principales características se encuentran:

- ✓ Velocidad: las páginas se abren en menor tiempo y se navega con comodidad en ellas.
- ✓ Seguridad: es un software de código abierto, esto permite que las fallas de seguridad se corrijan al instante, tienen un grupo de colaboradores encargados de esta cuestión.
- ✓ Se pueden usar varios perfiles de usuario con configuraciones diferentes para cada uno.

Este navegador consta además de un innumerable conjunto de extensiones y temas dentro de su paquete de complementos, que le permiten modificar la versión original instalada y personalizarla más al usuario, así como brindarle mayores comodidades. En el desarrollo de aplicaciones web es muy utilizado el complemento firebug, a través del cual los desarrolladores pueden llevar un control del tráfico de información desde el cliente hasta el servidor, brindando grandes facilidades a la hora de conectar la programación de la capa de presentación con la del negocio.

1.10. Conclusiones parciales del capítulo

Se realizó un estudio de diferentes sistemas de gestión de la información, así como de varias herramientas y tecnologías utilizadas en el desarrollo de software, lo cual evidenció:

- ✓ La necesidad de realizar una aplicación que gestione todo el banco de problemas de carácter científico de la facultad 3, debido a que no existe ningún sistema que cubra todas las necesidades actuales en la misma.
- ✓ Dicha aplicación debe ser desarrollada basándose en la metodología, herramientas y tecnologías antes seleccionadas, puesto que son las más idóneas dentro de la plataforma tecnológica que ofrece la facultad teniendo en cuenta las características del sistema.

CAPÍTULO II. ANÁLISIS Y DISEÑO

CAPÍTULO II. ANÁLISIS Y DISEÑO

2.1. Introducción

En el presente capítulo se describe la propuesta de solución del sistema a desarrollar y los procesos que serán informatizados con el desarrollo de esta aplicación. También se presentan los requisitos funcionales y no funcionales con los que debe cumplir el sistema propuesto. Se hace referencia a las técnicas empleadas tanto en la captura como en la validación de los requisitos, así como los patrones utilizados en el diseño de la aplicación. Además se presenta el modelo de datos del sistema, describiendo cada una de las tablas que integra el mismo. Por último se presentan los diagramas de clases del diseño de los procesos mencionados anteriormente, describiendo la función de cada una de las clases presentes en los mismos, con el objetivo de lograr una mayor comprensión del sistema a desarrollar.

2.2. Objeto de informatización

Con el presente trabajo de diploma se pretende desarrollar una aplicación capaz de informatizar todos los procesos relacionados con la gestión de los problemas de carácter científico de la facultad 3. Los procesos a informatizar son los siguientes:

- ✓ Licitación de problemáticas
- ✓ Contratación de problemáticas

2.3. Propuesta de solución

Debido a las dificultades existentes en la gestión de los problemas de carácter científico de la facultad 3, se decidió desarrollar un sistema informático que dé solución a estas dificultades. El mismo brindará la posibilidad de gestionar las problemáticas que sean originadas en la facultad por cualquier usuario del sistema, las cuales podrán ser revisadas y aprobadas o no por un grupo de expertos. Estos además podrán identificar las posibles salidas de cada una de las problemáticas aprobadas y asignarle el área de conocimiento dentro de la cual se enmarcan. También permitirá a los usuarios contratar cualquiera de las problemáticas que se encuentren disponibles, y en caso de ser aceptada dicha contratación, el sistema le creará una notificación al usuario y le dará la opción al mismo de crear un cronograma para darle solución a la problemática contratada. Una vez finalizado el tiempo de realización de cada problemática contratada, las mismas podrán ser certificadas por un grupo de desarrollo creado con este fin, el cual dirá si la solución fue certificada o no.

CAPÍTULO II. ANÁLISIS Y DISEÑO

2.4. Modelado del negocio

El primer paso hacia la construcción de cualquier aplicación es desarrollar el modelo de negocio, el cual describe los procesos de negocio, identificando quiénes participan y las actividades que requieren informatización.

2.4.1. Procesos del negocio

El negocio cuenta con dos procesos fundamentales los cuales serán descritos a continuación para un mejor entendimiento de los mismos.

Proceso Licitar problemáticas: en este proceso se crean cada una de las problemáticas de carácter científico que se originan en la facultad 3 por parte de cualquier usuario del sistema, las cuales son revisadas y aceptadas o no por un grupo de expertos. En caso de ser aceptada la problemática se le asignará una salida (Tesis, Proyecto de curso, Tesinas) en dependencia de los indicadores relacionados con la misma, además se le asocia una o varias áreas de conocimiento (Minería de datos, Ingeniería de Software, Base de datos) y finalmente se le notifica al usuario que la creó. En caso contrario se crea una notificación informándole al usuario que creó dicha problemática que la misma no fue aceptada, dándole la posibilidad de realizarle cambios a esta problemática y plantearla nuevamente.

Proceso Contratar problemáticas: en este proceso se contratan cada una de las problemáticas que han sido aceptadas. Las mismas son contratadas por cualquier usuario del sistema, los cuales le envían una solicitud de contrato al equipo de desarrollo. Estos son los encargados de revisar dicha solicitud y seguidamente el sistema creará una notificación al usuario, para que estos puedan crear el contrato y su propio cronograma para la realización de la problemática contratada. Este cronograma es controlado por un grupo de desarrollo, el cual en caso de terminada la solución en tiempo revisa la solución y la certifica o rechaza y puede o no realizarle recomendaciones sobre el trabajo realizado. En caso de que la problemática no sea realizada en el tiempo previsto, el usuario solicita una prórroga creando un nuevo cronograma, la cual es aceptada o no por el equipo de desarrollo.

Los diagramas del modelado de los procesos de negocio Licitación de problemáticas y Contratación de problemáticas, así como la descripción de los mismos se encuentran en el expediente de proyecto del Centro de Informatización de la Gestión de Entidades (CEIGE), además de los restantes artefactos entregados del sistema.

CAPÍTULO II. ANÁLISIS Y DISEÑO

2.5. Modelado del sistema

A partir de este punto se comienza a modelar la solución propuesta. Para ello se identifican los requisitos, tanto funcionales como no funcionales, se realiza el modelo de datos y se modelan los procesos a través de diagramas de clases del diseño del sistema a desarrollar.

2.5.1. Requisitos funcionales y no funcionales

Para realizar la captura de los requisitos se hizo necesaria la utilización de algunas técnicas existentes con este fin. Entre las técnicas para la captura de requisitos utilizadas se encuentra la **entrevista**, aplicada a los clientes para los cuales va a ser desarrollada la aplicación. También se llevaron a cabo **tormentas de ideas y análisis de protocolos**, ambas técnicas encargadas de posibilitar debates y talleres donde cada cliente explicó de forma explícita sus necesidades.

Listado de los requisitos funcionales

- RF-1 Gestionar problemáticas
- RF-2 Gestionar indicadores
- RF-3 Gestionar salidas
- RF-4 Gestionar áreas de conocimiento
- RF-5 Revisar problemática
- RF-6 Realizar solicitud de contrato
- RF-7 Listar solicitudes de contrato
- RF-8 Revisar solicitud de contrato
- RF-9 Consultar solicitud de contrato
- RF-10 Realizar contrato
- RF-11 Listar contratos
- RF-12 Realizar cronograma de ejecución
- RF-13 Modificar cronograma de ejecución
- RF-14 Gestionar prórrogas
- RF-15 Revisar prórrogas
- RF-16 Gestionar solución
- RF-17 Revisar solución
- RF-18 Gestionar notificaciones

Las descripciones de los requisitos funcionales se encuentran en el expediente de proyecto del Centro de Informatización de la Gestión de Entidades (CEIGE).

CAPÍTULO II. ANÁLISIS Y DISEÑO

A continuación se muestra un diagrama de paquetes para obtener una visión más clara de las funcionalidades que presentará el sistema, agrupándolas en paquetes lógicos y mostrando las relaciones de dependencia entre estas.

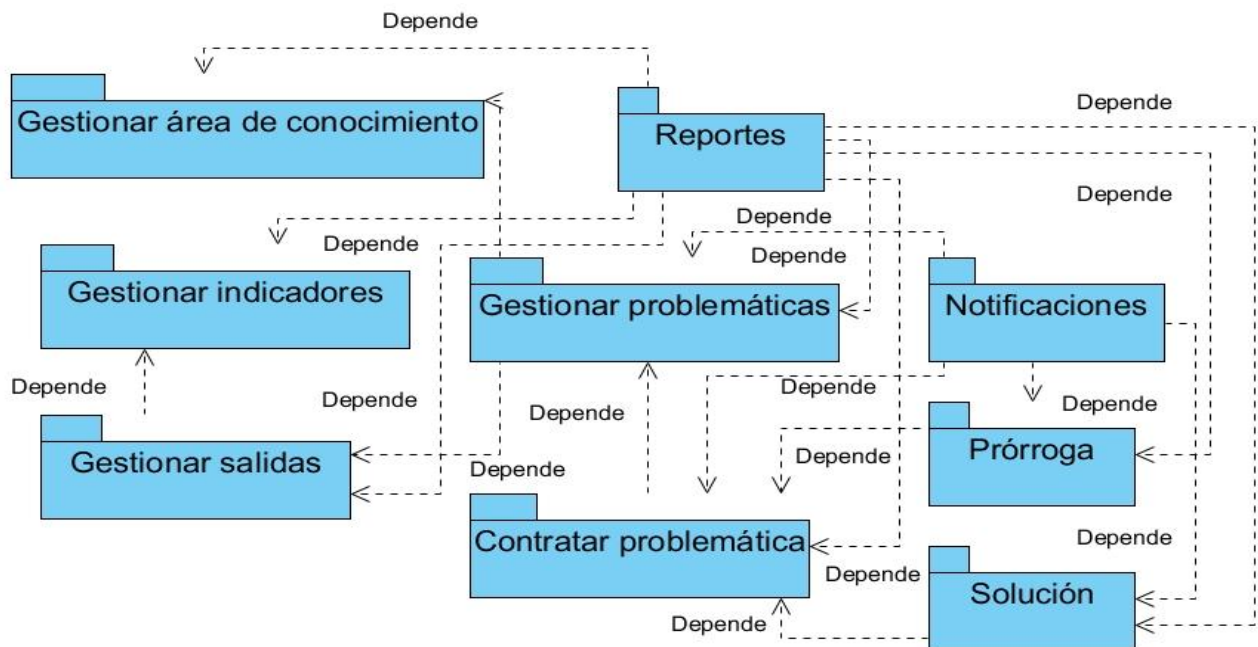


Figura 3. Diagrama de paquetes de las funcionalidades del sistema.

Del análisis de la figura 6 se puede constatar que el sistema cuenta con un total de 9 paquetes de requisitos, de los cuales el paquete Gestionar problemáticas constituye el arquitectónicamente más significativo para el sistema, debido a que este es el punto de partida para realizar varias funcionalidades del mismo, y el paquete Contratar problemática es el arquitectónicamente más significativo para el negocio, debido a que es la funcionalidad principal del sistema a desarrollar. Se puede observar que estos paquetes de requisitos se encuentran relacionados a través de conectores que muestran el flujo del comportamiento de los requisitos en el sistema y las dependencias que existen entre estos.

A continuación se describen cada uno de los paquetes de requisitos, con el objetivo de obtener un mejor entendimiento acerca de los requisitos que engloban cada uno de estos paquetes y las dependencias que existen entre estos.

Paquete Gestionar problemáticas: incluye los requisitos Adicionar, Modificar, Eliminar, Buscar, Listar y Revisar problemáticas, permitiendo gestionar cada una de las problemáticas de carácter científico del banco problemas de la facultad 3. También se revisan cada una de las problemáticas presentes en la aplicación, para aceptar o rechazar cada una de estas. En caso de ser aceptada una problemática brinda la posibilidad de asignarle un área de conocimiento, una salida y cambiarle su estado que inicialmente es Pendiente.

CAPÍTULO II. ANÁLISIS Y DISEÑO

Paquete Gestionar salidas: engloba los requisitos Adicionar, Modificar, Eliminar, Buscar y Listar las posibles salidas que se le pueden asignar a cada una de las problemáticas una vez revisadas y aceptadas, las cuales son las posibles fuentes a las que tributará la solución de cada una de las problemáticas.

Paquete Gestionar indicadores: contiene los requisitos Adicionar, Modificar, Eliminar, Buscar y Listar los indicadores a tener en cuenta para asignarle una salida a cada una de las problemáticas que hayan sido aceptadas.

Paquete Gestionar área de conocimiento: incluye los requisitos Adicionar, Modificar, Eliminar, Buscar y Listar las áreas de conocimiento dentro de la cual se enmarcan cada una de las problemáticas una vez revisadas y aceptadas.

Paquete Contratar problemática: contiene los requisitos Realizar solicitud de contrato, Listar solicitudes de contrato, Revisar solicitud de contrato, Realizar contrato y Realizar cronograma de ejecución. Permite contratar cada una de las problemáticas que hayan sido revisadas y aceptadas con anterioridad, para darle solución a cada una de las mismas en un intervalo de tiempo determinado.

Paquete Prórroga: engloba los requisitos Adicionar, Modificar, Eliminar, Consultar y Revisar prórrogas, además de Modificar cronograma de ejecución. Una vez concluido el tiempo límite para realizar la solución, en caso de no haberse terminado la misma, se le brinda al usuario la posibilidad de solicitar una prórroga con un nuevo cronograma de ejecución, la cual es revisada por el equipo de desarrollo para aprobar o no esta.

Paquete Solución: contiene los requisitos Adicionar, Eliminar, Consultar, Buscar y Revisar solución. Terminada la solución el usuario deberá subirla al sistema para que sea revisada por un grupo de desarrollo, el cual calificará la misma y emitirá sus observaciones.

Paquete Notificaciones: permite realizar la gestión de las notificaciones emitidas a los usuarios, al producirse cambios de estado durante el seguimiento que se les brinda a las problemáticas.

Paquete Reportes: permite mostrar diferentes reportes en la aplicación relacionados con las distintas problemáticas independientemente del estado en que se encuentren cada una de estas.

Listado de los requisitos no funcionales

✓ Usabilidad:

RNF1: todos los mensajes de error del sistema deberán incluir una descripción textual del error.

RNF2: el acceso al sistema debe ser fácil y rápido.

RNF3: las etiquetas de cada funcionalidad y los campos de cada interfaz tendrán títulos asociados a su función de negocio.

✓ Confiabilidad y Seguridad:

CAPÍTULO II. ANÁLISIS Y DISEÑO

RNF4: proteger la información manejada por el sistema de accesos no autorizados.

RNF5: garantizar que las funcionalidades del sistema se muestren de acuerdo al rol del usuario que esté activo.

RNF6: la base de datos debe estar fraccionada sobre varios esquemas, dividiendo así de una forma lógica las funcionalidades, evitando la pérdida total de la información en caso de algún accidente o ataque.

RNF7: permitir autenticarse a través de un Servidor de Dominio.

✓ **Rendimiento:**

RNF8: el sistema debe ser capaz de funcionar al ser instalados en una misma PC todos sus componentes tales como Gestor de bases de Datos, Aplicación Web, etc.

✓ **Portabilidad:**

RNF9: la solución debe poder ser ejecutada desde las plataformas Windows y Linux, dentro de estas en los Sistemas Operativos Microsoft Windows XP y Ubuntu 10.04.

✓ **Disponibilidad:**

RNF10: el sistema debe estar disponible las 24 horas del día sin ninguna interrupción.

✓ **Mantenibilidad:**

RNF11: el sistema debe estar en capacidad de permitir en el futuro su fácil mantenimiento con respecto a los posibles errores que se puedan presentar durante la operación del sistema.

✓ **Validación de información:**

RNF12: el sistema debe validar automáticamente la información contenida en los formularios de ingreso. En el proceso de validación de la información, se deben tener en cuenta aspectos tales como obligatoriedad de campos, longitud de caracteres permitida por campo, manejo de tipos de datos.

✓ **Escalabilidad:**

RNF13: el sistema debe estar en capacidad de permitir en el futuro el desarrollo de nuevas funcionalidades, modificar o eliminar funcionalidades después de su construcción y puesta en marcha inicial.

✓ **Hardware.**

Cliente:

RNF14: requerimientos mínimos: procesador Pentium IV a 800 MHz, 128mb de memoria RAM y un navegador web.

RNF15: las computadoras clientes deben estar conectadas en red.

Servidor:

RNF16: requerimientos mínimos: procesador Dual Core a 3.00 GHz, 1gb de memoria RAM y una capacidad de 40gb de disco duro.

CAPÍTULO II. ANÁLISIS Y DISEÑO

RNF17: el servidor debe tener al menos 1 tarjeta de red para establecer la conexión.

✓ **Software:**

Cliente:

RNF18: sistema operativo con interfaz gráfica y soporte para red.

RNF19: las interfaces deben ser compatibles con Mozilla Firefox 3.0 o superior.

Servidor:

RNF20: servidor web Apache 2.2.6 o superior.

RNF21: gestor de base de datos PostgreSQL 8.3 o superior.

2.5.2. Aplicación de técnicas de validación de requisitos

Los requisitos una vez definidos necesitan ser validados. Las técnicas que se utilizaron para la validación de los requisitos identificados fueron las siguientes:

✓ **Reviews o Walk-throughs**

Una vez terminada la especificación de los requisitos identificados, los mismos fueron entregados al analista principal del proyecto ERP-Cuba para su posterior revisión. Este análisis se realiza con el objetivo de verificar que cada requisito responda a las necesidades del usuario. Cada plantilla de especificación se revisó para encontrar incoherencias, falta de claridad o ausencia de conceptos, para poder corregirlos y hacer la descripción de forma más detallada.

✓ **La Revisión Técnica Formal (RTF)**

Una vez terminada las especificaciones de los requisitos se realizaron reuniones con los clientes, donde revisaron cada una de estas especificaciones. Ante los errores detectados se volvieron a analizar para una nueva revisión en caso que fuese necesario. Con este proceso se pudo validar que la interpretación de cada una de las especificaciones no fuera ambigua, y que cada uno de los requisitos cumplía con lo que necesitaba el usuario final.

2.5.3. Patrones utilizados en el diseño de la aplicación

Dentro de los patrones GRASP se utilizaron:

- ✓ **Experto:** se emplea al trabajar con el marco de trabajo Sauxe y un ejemplo de ello es la inclusión de Doctrine Generator para mapear la base de datos, el cual es el encargado de generar las clases para la gestión de las tablas en dicha base de datos con las responsabilidades debidamente asignadas. Cada clase cuenta con un grupo de funcionalidades que las convierte en experta de la información de la tabla a la que representa. Esto se puede observar en la clase (class Problematica extends BaseProblematica) la cual es la encargada de realizar las acciones directamente a la BD.

CAPÍTULO II. ANÁLISIS Y DISEÑO

- ✓ **Creador:** este patrón se evidencia en la clase (class InterfazprincipalController extends ZendExt_Controller_Secure), la cual contiene las acciones y es la encargada de ejecutarlas. Dentro de esta existen varias funcionalidades en las cuales se crean varios objetos o instancias de las clases que representan cada una de las tablas existentes en la base de datos.
- ✓ **Alta Cohesión:** Sauxe presenta entre sus principales características la organización del trabajo en cuanto a estructura y responsabilidades bien definidas, esto permite que se trabaje con las clases con una alta cohesión. Un ejemplo de esto es la clase (class InterfazprincipalController extends ZendExt_Controller_Secure), la cual delega funciones específicas a otras clases, encargándose solo de definir las acciones a realizar.
- ✓ **Bajo Acoplamiento:** este patrón se evidencia en la aplicación ya que el sistema presenta poca dependencia entre las clases. Es en el modelo donde únicamente se encuentran algunas relaciones de asociación entre las clases, pero no representa una gran jerarquía.

Dentro de los patrones GOF se utilizaron:

- ✓ **Mediador:** la comunicación en la base de datos se puede tornar compleja debido a las dependencias marcadas entre las tablas que la componen, esto resulta engorroso a la hora de acceder a un determinado valor. La solución a este inconveniente viene dada por la utilización de este patrón, específicamente, creando una nueva tabla entre todas las tablas unidas mediante una relación de muchos a muchos. La tabla mediadora posee una relación de uno a muchos con las vinculadas a ella. Este patrón se encuentra reflejado en la creación de una clase para la relación entre las clases (AreaConocimiento) y (Problemática), en este caso AreaConocimientoProblematica.
- ✓ **Cadena de Responsabilidad:** está concebido que ante la ocurrencia de un error al realizarse una determinada consulta a la base de datos el mismo sea manejado por el Modelo, creando una nueva excepción de tipo ZendExt_Exception. Dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista ya traducida, esta última por su parte mostrará un mensaje al usuario en un lenguaje entendible notificando el error y sin especificar detalles del mismo. De esta manera se distribuyen las responsabilidades entre los diferentes componentes, evidenciándose por lo tanto el empleo de este patrón.

CAPÍTULO II. ANÁLISIS Y DISEÑO

También fue utilizado el patrón de diseño mostrado a continuación:

- ✓ **Inversión de Control (IOC):** este es un patrón de diseño pensado para permitir un menor acoplamiento entre los componentes de una aplicación y fomentar así el uso de los mismos [55]. La utilización de este patrón en el sistema está reflejada en la creación y empleo de la clase IOC¹¹, ya que por la necesidad de utilizar varios servicios desde diferentes componentes, era necesaria la implementación de una clase donde fueran publicados los mismos, facilitando su interacción.

2.5.4. Modelo de datos del sistema

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general permite describir las estructuras de datos de la base de datos (el tipo de los datos que incluye la base de datos y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base de datos).

A continuación se presenta el modelo de datos del sistema el cual cuenta con 16 tablas.

¹¹ Inversión de Control del inglés Inversion of Control.

CAPÍTULO II. ANÁLISIS Y DISEÑO

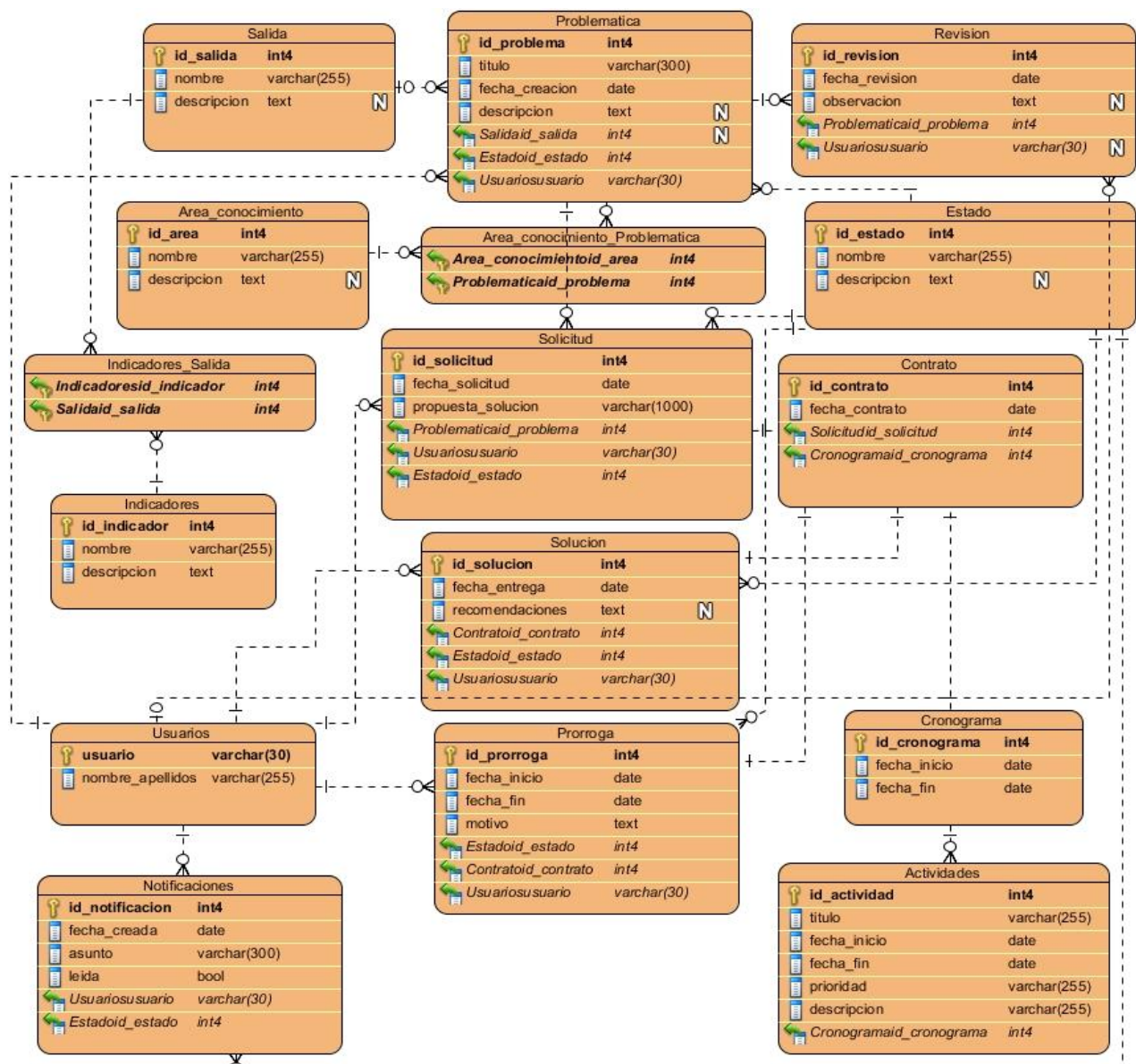


Figura 4. Modelo de datos del sistema.

2.5.4.1. Descripción de las tablas

A continuación se muestran las descripciones de las tablas del modelo de datos del sistema.

Nombre: “Problematica”		
Tipo: Entidad		
Descripción: almacena los datos correspondientes a cada una de las problemáticas.		
Atributo	Tipo	Descripción
id_problema	int4	Identificador de la tabla.
titulo	varchar	Título de la problemática.
fecha_creacion	date	Fecha de creación de la problemática.

CAPÍTULO II. ANÁLISIS Y DISEÑO

descripcion	text	Descripción de la problemática.
Salidaid_salida	int4	Identificador de la tabla "Salida" (Llave foránea).
Estadoid_estado	int4	Identificador de la tabla "Estado" (Llave foránea).
Usuariosusuario	varchar	Identificador de la tabla "Usuario" (Llave foránea).

Tabla 2. Descripción de la tabla "Problematica".

Nombre: "Revision"		
Tipo: Entidad		
Descripción: almacena los datos de las revisiones realizadas a las problemáticas en estado "Pendiente".		
Atributo	Tipo	Descripción
id_revision	int4	Identificador de la tabla.
fecha_revision	date	Fecha de revisión de la problemática.
observacion	text	Observación de la revisión.
Problematicaid_problema	int4	Identificador de la tabla "Problematica" (Llave foránea).
Usuariosusuario	varchar(30)	Identificador de la tabla "Usuario" (Llave foránea).

Tabla 3. Descripción de la tabla "Revision".

Nombre: "Salida"		
Tipo: Entidad		
Descripción: almacena los datos de las posibles salidas a las que tributará cada problemática.		
Atributo	Tipo	Descripción
id_salida	int4	Identificador de la tabla.
nombre	varchar	Nombre de la salida.
descripcion	text	Descripción de la salida.

Tabla 4. Descripción de la tabla "Salida".

Nombre: "Area_conocimiento"		
Tipo: Entidad		
Descripción: almacena los datos de las posibles áreas de conocimiento dentro de las cuales se enmarcan cada una de las problemáticas.		
Atributo	Tipo	Descripción

CAPÍTULO II. ANÁLISIS Y DISEÑO

id_area	int4	Identificador de la tabla.
nombre	varchar	Nombre del área de conocimiento.
descripcion	text	Descripción del área de conocimiento.

Tabla 5. Descripción de la tabla "Area_conocimiento".

Nombre: "Indicadores"		
Tipo: Entidad		
Descripción: almacena los datos de los indicadores a tener en cuenta para asignarle una salida a cada una de las problemáticas que hayan sido aceptadas.		
Atributo	Tipo	Descripción
id_indicador	int4	Identificador de la tabla.
nombre	varchar	Nombre del indicador.
descripcion	text	Descripción del indicador.

Tabla 6. Descripción de la tabla "Indicadores".

Nombre: "Estado"		
Tipo: Entidad		
Descripción: almacena los datos de los estados que pueden tomar cada una de las problemáticas.		
Atributo	Tipo	Descripción
id_indicador	int4	Identificador de la tabla.
nombre	varchar	Nombre del estado.
descripcion	text	Descripción del estado.
Notificacionesidnotificacion	int4	Identificador de la tabla "Notificaciones" (Llave foránea).

Tabla 7. Descripción de la tabla "Estado".

Nombre: "Solicitud"		
Tipo: Entidad		
Descripción: almacena los datos de las solicitudes que se han realizado para darle solución a las problemáticas.		
Atributo	Tipo	Descripción
id_solicitud	int4	Identificador de la tabla.
fecha_solicitud	date	Fecha de realizada la solicitud.
Problematicaid_problema	int4	Identificador de la tabla "Problematica" (Llave foránea).

CAPÍTULO II. ANÁLISIS Y DISEÑO

Usuariosusuario	varchar	Identificador de la tabla "Usuario" (Llave foránea).
-----------------	---------	--

Tabla 8. Descripción de la tabla "Solicitud".

Nombre: "Contrato"		
Tipo: Entidad		
Descripción: almacena los datos de cada contrato realizado para darle solución a una problemática en específico.		
Atributo	Tipo	Descripción
id_contrato	int4	Identificador de la tabla.
fecha_contrato	date	Fecha de realizado el contrato.
Solicitudid_solicitud	int4	Identificador de la tabla "Solicitud" (Llave foránea).
Cronogramaid_cronograma	int4	Identificador de la tabla "Cronograma" (Llave foránea).

Tabla 9. Descripción de la tabla "Contrato".

Nombre: "Solucion"		
Tipo: Entidad		
Descripción: almacena los datos de cada solución realizada para una problemática en específico.		
Atributo	Tipo	Descripción
id_solucion	int4	Identificador de la tabla.
recomendaciones	text	Recomendaciones a la solución.
Contratoid_contrato	int4	Identificador de la tabla "Contrato" (Llave foránea).
Usuariosusuario	varchar	Identificador de la tabla "Usuario" (Llave foránea).
Estadoid_estado	int4	Identificador de la tabla "Estado" (Llave foránea).

Tabla 10. Descripción de la tabla "Solucion".

Nombre: "Prorroga"		
Tipo: Entidad		
Descripción: almacena los datos de cada prórroga solicitada para poder darle solución a una problemática dada.		
Atributo	Tipo	Descripción
id_prorroga	int4	Identificador de la tabla.
fecha_inicio	date	Fecha de inicio de la prórroga.
fecha_fin	date	Fecha de fin de la prórroga.

CAPÍTULO II. ANÁLISIS Y DISEÑO

motivo	text	Motivo de la prórroga.
Estadoid_estado	int4	Identificador de la tabla "Estado" (Llave foránea).
Contratoid_contrato	int4	Identificador de la tabla "Contrato" (Llave foránea).
Usuariosusuario	varchar	Identificador de la tabla "Usuario" (Llave foránea).

Tabla 11. Descripción de la tabla "Prorroga".

Nombre: "Cronograma"		
Tipo: Entidad		
Descripción: almacena los datos de cada cronograma dentro del cual se le dará solución a una problemática.		
Atributo	Tipo	Descripción
id_cronograma	int4	Identificador de la tabla.
fecha_inicio	date	Fecha de inicio del cronograma.
fecha_fin	date	Fecha de fin del cronograma.
descripcion	text	Descripción del cronograma.

Tabla 12. Descripción de la tabla "Cronograma".

Nombre: "Usuarios"		
Tipo: Entidad		
Descripción: almacena los datos de cada usuario del sistema.		
Atributo	Tipo	Descripción
usuario	varchar	Identificador de la tabla.
nombre	varchar	Nombre del usuario.
apellidos	varchar	Apellidos del usuario.

Tabla 13. Descripción de la tabla "Usuarios".

Nombre: "Área_conocimiento_Problematica"		
Tipo: Entidad		
Descripción: relación entre las tablas "Area_conocimiento" y "Problematica".		
Atributo	Tipo	Descripción
Area_conocimientoid_area	int4	Identificador de la tabla "Area_conocimiento" (Llave foránea).
Problematicaid_problema	int4	Identificador de la tabla "Problematica" (Llave foránea).

Tabla 14. Descripción de la tabla "Area_conocimiento_Problematica".

CAPÍTULO II. ANÁLISIS Y DISEÑO

Nombre: "Indicadores_Salida"		
Tipo: Entidad		
Descripción: relación entre las tablas "Indicadores" y "Salida".		
Atributo	Tipo	Descripción
Indicadoresid_indicador	int4	Identificador de la tabla "Indicadores" (Llave foránea).
Salidaid_salida	int4	Identificador de la tabla "Salida" (Llave foránea).

Tabla 15. Descripción de la tabla "Indicadores_Salida".

Nombre: "Notificaciones"		
Tipo: Entidad		
Descripción: almacena los datos correspondientes a cada una de las notificaciones.		
Atributo	Tipo	Descripción
idnotificacion	int4	Identificador de la tabla.
fecha_creada	date	Fecha de creación de las notificaciones
asunto	varchar	Asunto de las notificaciones.
leida	bool	Indica si la notificación esta leída o no.
Usuariosusuario	varchar	Identificador de la tabla "Usuarios" (Llave foránea).
Estadoid_estado	int4	Identificador de la tabla "Estado" (Llave foránea).

Tabla 16. Descripción de la tabla "Notificaciones".

Nombre: "Actividades"		
Tipo: Entidad		
Descripción: almacena los datos correspondientes a cada una de las actividades.		
Atributo	Tipo	Descripción
Id_actividad	int4	Identificador de la tabla.
titulo	varchar	Título de las actividades
fecha_inicio	date	Fecha de inicio de las actividades.
fecha_fin	date	Fecha de fin de las actividades.
prioridad	varchar	Prioridad de las actividades.
descripcion	varchar	Descripción de las actividades.
Cronogramaid_cronograma	int4	Identificador de la tabla "Cronograma" (Llave foránea).

Tabla 17. Descripción de la tabla "Actividades".

CAPÍTULO II. ANÁLISIS Y DISEÑO

2.5.5. Diagrama de clases del diseño

Los diagramas de clases son diagramas de estructura estática donde la representación de los requisitos se lleva a cabo a través de las clases del sistema y sus interrelaciones. Específicamente los diagramas de clases de diseño son muy útiles porque muestran a través de atributos y métodos, la estructura de las clases que después serán escritas en algún lenguaje de programación (PHP en este caso). Se utilizan para modelar la vista de diseño estática de un sistema. A continuación se muestra el diagrama de clases del diseño del proceso Licitar problemáticas.

Diagrama de clases del diseño del proceso Licitar problemáticas

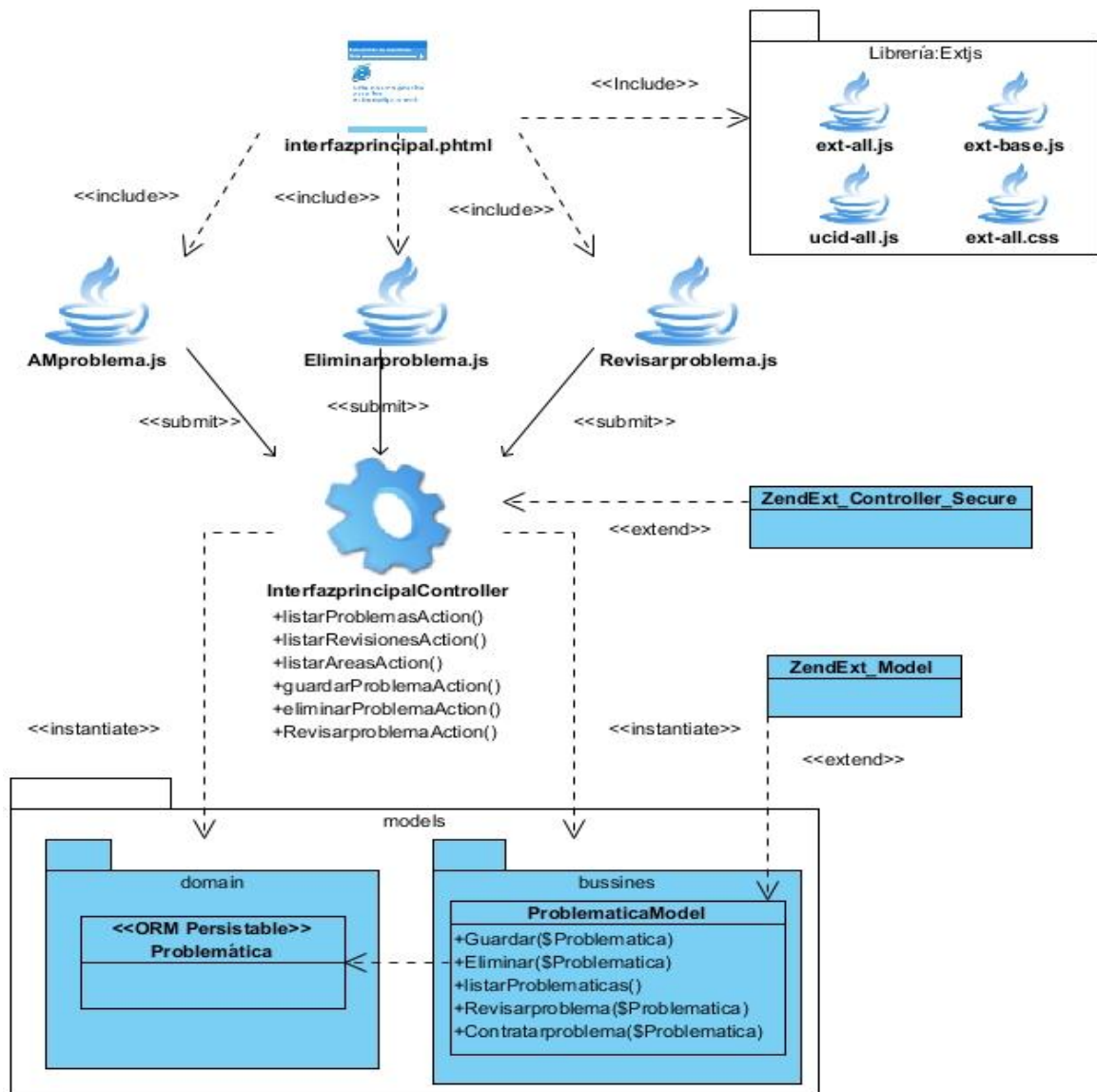


Figura 5. Proceso Licitar problemáticas.

CAPÍTULO II. ANÁLISIS Y DISEÑO

✓ Descripción del diseño de clases del proceso Licitación problemáticas:

Clases	Descripción
Librería Extjs	Contiene los componentes generados a través de la librería JavaScript Extjs.
Interfazprincipal.phtml	Responsable de visualizar a través de los js la información necesaria del proceso Licitación problemáticas.
AMproblema.js	Debe generar de forma dinámica a través del DOM y utilizando la librería Extjs componentes a través de los cuales se adicionen y se modifiquen las problemáticas. Responsable de enviar y recibir los datos de la controladora utilizando tecnología AJAX.
Eliminarproblema.js	Le corresponde generar de forma dinámica a través del DOM y utilizando la librería Extjs los componentes necesarios a través de los cuales se eliminen las problemáticas. Responsable de enviar y recibir los datos de la controladora utilizando tecnología AJAX.
Revisarproblema.js	Debe generar de forma dinámica a través del DOM y utilizando la librería Extjs los componentes necesarios para revisar las problemáticas. Le corresponde enviar y recibir los datos de la controladora utilizando tecnología AJAX.
InterfazprincipalController	Clase controladora que sirve de puente entre la vista y el modelo. Contiene los métodos necesarios para adicionar, modificar, eliminar o revisar las problemáticas.
ZendExt_Controller_Secure	Encargada de gestionar acciones personalizadas y está integrada a la seguridad.
Paquete models	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.
ZendExt_Model	Modelo gestor de negocio que permite entre otras funcionalidades iniciar la conexión a la base de datos.

Tabla 18. Descripción de las clases del diseño del proceso Licitación problemáticas.

CAPÍTULO II. ANÁLISIS Y DISEÑO

El diagrama de clases del diseño del proceso Contratar problemáticas y la descripción de las clases presentes en el mismo se encuentran en el expediente de proyecto del Centro de Informatización de la Gestión de Entidades (CEIGE).

2.6. Conclusiones parciales del capítulo

Los artefactos generados en este capítulo son la base para el desarrollo del sistema a realizar, debido a lo cual:

- ✓ Se definieron todos los requisitos del sistema, los cuales cumplen con todas las funcionalidades requeridas por el proceso de gestión del banco de problemas de la facultad 3.
- ✓ Se describieron los patrones de diseño utilizados durante el desarrollo del sistema, los cuales proporcionaron una baja dependencia y una alta reutilización entre las clases del mismo.
- ✓ Se realizó el modelo de datos del sistema con el objetivo de que este sea capaz de almacenar toda la información referente al proceso de seguimiento de cada una de las problemáticas, mostrándose además las relaciones existentes entre las diferentes tablas de la base de datos.
- ✓ Se realizaron y describieron los diagramas de clases de diseño de los procesos Licitación problemáticas y Contratar problemáticas, para alcanzar un entendimiento mejor y más profundo de cómo se realizará la implementación del sistema.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

CAPÍTULO III. IMLEMENTACIÓN Y PRUEBA

3.1. Introducción

En este capítulo se realiza una breve descripción del marco de trabajo a utilizar, así como los estándares a utilizar durante la implementación del sistema. También se describen las clases y funcionalidades de dicho sistema, además de realizar el diagrama de despliegue para obtener una visión más clara sobre el mismo. Por último se validará el diseño propuesto a través de las métricas TOC y RC, se describirán las pruebas de caja negra y caja blanca realizadas al software y los resultados obtenidos de estas. Se realizará también la validación de las variables confiabilidad y seguridad propuestas en el problema.

3.2. Estructura del marco de trabajo Sauxe

A continuación se presenta la estructura del marco de trabajo Sauxe, mostrando cómo será organizada la implementación del sistema a desarrollar, de manera que facilite la organización y claridad durante el desarrollo, además se explica cómo va a ser garantizada la seguridad del Sistema para la gestión del banco de problemas de la facultad 3.

Contenido dentro de la carpeta app:

En la carpeta denominada app se almacenan los controladores y el modelo de cada una de las funcionalidades a desarrollar.



Figura 6. Contenido de la carpeta de aplicación app.

- ✓ **comun:** contiene la carpeta recursos y dentro de esta una denominada xml. Esta última incluye a los ficheros: ioc, validator, exception que serán explicados a continuación.
- ✓ **ioc:** es donde se publican los servicios que brinda cada uno de los componentes en cuanto a nombre de clases, funciones y tipo de resultado.
- ✓ **validator:** chequea las precondiciones antes de ejecutar una determinada función en el servidor según el tipo de parámetros, la acción y el usuario que la realice.
- ✓ **exception:** se define el tipo, idioma y la descripción del mensaje que va a ser mostrado cuando se lance una excepción en el servidor.

El componente **Gestionar problemas** va a contener un conjunto de carpetas que serán especificadas a continuación:

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA



Figura 7. Contenido del componente Gestionar problemas dentro de la carpeta app.

En la carpeta **controllers** se encontrarán las clases controladoras encargadas de gestionar las funcionalidades del sistema.

La carpeta **models** se estructura de la siguiente forma:



Figura 8. Contenido de la carpeta models.

Esta carpeta contiene dos carpetas las cuales a su vez agrupan clases y otras carpetas, las mismas serán explicadas a continuación:

- ✓ **bussines:** contiene las clases necesarias para acceder a los datos que persisten en la base de datos.
- ✓ **domain:** contiene las clases generadas por el ORM Doctrine Generator a partir de cada una de las tablas existentes en la base de datos.

Cada una de estas clases mencionadas anteriormente heredará de una clase generada igualmente por el Doctrine Generator las cuales se ubican en otra carpeta dentro de esta llamado generated.

La carpeta **services** incluye todas las clases y funcionalidades de los servicios que va a ofrecer el sistema. Para solicitar un servicio que está en otro dominio, se accede a través de la carpeta services, quien analizará la solicitud e irá a la clase que tiene dicha funcionalidad y la devolverá a services, que a su vez se la entregará al dominio solicitante.

La carpeta **views** contiene las carpetas idioma y scripts, que se encargan de contener el idioma en que se va a mostrar la aplicación y las páginas clientes respectivamente.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA



Figura 9. Contenido de la carpeta views.

Contenido dentro de la carpeta web:

Al mismo nivel de la carpeta app mencionada anteriormente debe existir además una carpeta que contiene las vistas de los subsistemas y componentes. Dicha carpeta se denomina web.



Figura 10. Carpeta de diseño correspondiente al componente Gestionar problemas.

- ✓ **index:** este fichero incluye la dirección del archivo de configuración y a través de este inicializa la aplicación para que se carguen en la misma un conjunto de componentes necesarios para su funcionamiento. Su código permanece igual para todos los componentes.



Figura 11. Contenido del componente Gestionar problemas dentro de la carpeta web.

La carpeta **views** contendrá los css y los js que se explican a continuación.



Figura 12. Contenido dentro de la carpeta views.

- ✓ **css:** incluye las clases necesarias para estructurar gráficamente el componente, separando de esta forma el estilo del contenido.
- ✓ **js:** comprenderá las clases Java Script necesarias para que el usuario interactúe con el sistema y obtenga los resultados necesarios. Está compuesta por carpetas con los nombres de las funcionalidades del componente.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

3.2.1. Seguridad del sistema

La seguridad del sistema va a ser garantizada a través del Sistema de Gestión Integral de Seguridad (ACAXIA), el cual tiene como objetivo garantizar la administración de la seguridad en sistemas de gestión. Este brinda sus servicios a todos los sistemas que se suscriban a él. Para ello se gestionan las conexiones a la base de datos, las funcionalidades asociadas y las acciones que realizan las mismas. Una vez registrada esta información se procede a la creación de roles a los cuales se les dan los permisos dentro de cada sistema. Luego se crean los usuarios con el perfil seleccionado y se le asignan uno o muchos roles en una o muchas entidades respectivamente.

3.3. Estándares de código

Un estándar de código se basa en la estructura y apariencia física de un programa, con el fin de facilitar la lectura, comprensión, mantenimiento del código y reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa. Este no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y la legibilidad del código, aspecto crucial a la hora de darle mantenimiento y mejorar las funcionalidades de un software. Partiendo de lo dicho anteriormente, se definen 3 partes principales dentro de un estándar de programación:

- ✓ Convención de nomenclatura: es la forma de nombrar las variables, funciones, métodos.
- ✓ Convenciones de legibilidad de código: es la forma de organizar el código.
- ✓ Convenciones de documentación: es la manera de establecer los comentarios, la ayuda.

3.3.1. Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación **PascalCasing**, la cual define que los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula y con solo leerlo se reconoce el propósito de la misma. Ejemplo: ProblematicaModel. En este caso el nombre de clase está compuesto por 2 palabras iniciadas cada una con letra mayúscula.

Nomenclatura según el tipo de clases

- ✓ **Clases controladoras**: las clases controladoras después del nombre llevan la palabra: "Controller". Ejemplo: EstructuraController

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

- ✓ **Clases de los modelos: Business (Negocio):** las clases que se encuentran dentro de Business después del nombre llevan la palabra: "Model". Ejemplo: CampoModel.
- ✓ **Domain (Dominio):** las clases que se encuentran dentro de Domain, el nombre que reciben es el de la tabla en la base de datos. Ejemplo: Problematica.
- ✓ **Generated (Dominio base):** las clases que se encuentran dentro de Generated, el nombre que reciben comienza con la palabra: "Base" y seguido el nombre de la tabla en la base de datos. Ejemplo: BaseProblematica.

3.3.2. Nomenclatura de las funcionalidades y atributos

El nombre a emplear para las funcionalidades y los atributos se escribe con la inicial del identificador en minúscula, en caso de que sea un nombre compuesto se empleará la notación **CamelCasing** que es similar a la antes mencionada **PascalCasing**, con la excepción de la primera letra.

Ejemplo de un método: calcularAjuste. El nombre del método está compuesto por 2 palabras, la primera en minúsculas y la segunda iniciando con letra mayúscula.

Las principales funcionalidades de las clases controladoras se les escribe el nombre y seguida la palabra: "Action" Ejemplo: listarProblemasAction ().

Ejemplo de un atributo: cantidadProblemas. El nombre del atributo está compuesto por dos palabras, la primera en minúsculas y la segunda iniciando con letra mayúscula.

3.3.3. Nomenclatura de los comentarios

Los comentarios deben ser claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando. En la realización de un software se debe comentar todo lo que se haga dentro del desarrollo, para lograr un código más legible y reutilizable y así se pueda aumentar su mantenibilidad a lo largo del tiempo.

3.4. Descripción de las clases y funcionalidades del sistema

Clase del Dominio:

Nombre: Actividades	
Tipo de clase: Dominio	
Para cada responsabilidad:	
Nombre	GetTodosConCantidad(\$idfinal)

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

Descripción	Obtiene los datos de todas las actividades asociadas a un cronograma determinado, especificándole el identificador de este.
Nombre	buscarCantidadActividadesPorCronograma()
Descripción	Obtiene la cantidad de actividades asociadas a un cronograma.
Nombre	getTodasDeUnCronograma(\$idcontrato)
Descripción	Obtiene los datos de todas las actividades de un cronograma asociado a un contrato determinado, especificándole el identificador de este.

Tabla 19. Descripción de la clase Actividades.

El resto de las descripciones de las clases se encuentran en el expediente de proyecto del Centro de Informatización de la Gestión de Entidades (CEIGE).

3.5. Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Está compuesto por nodos, dispositivos y conectores. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. Permite el mapeo de procesos dentro de los nodos, asegurando la distribución del comportamiento a través de aquellos nodos que son representados [52]. Se definirá una arquitectura cliente-servidor detallada de la siguiente manera, ver **Figura 13**.

A continuación se muestra el diagrama de despliegue del sistema:

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

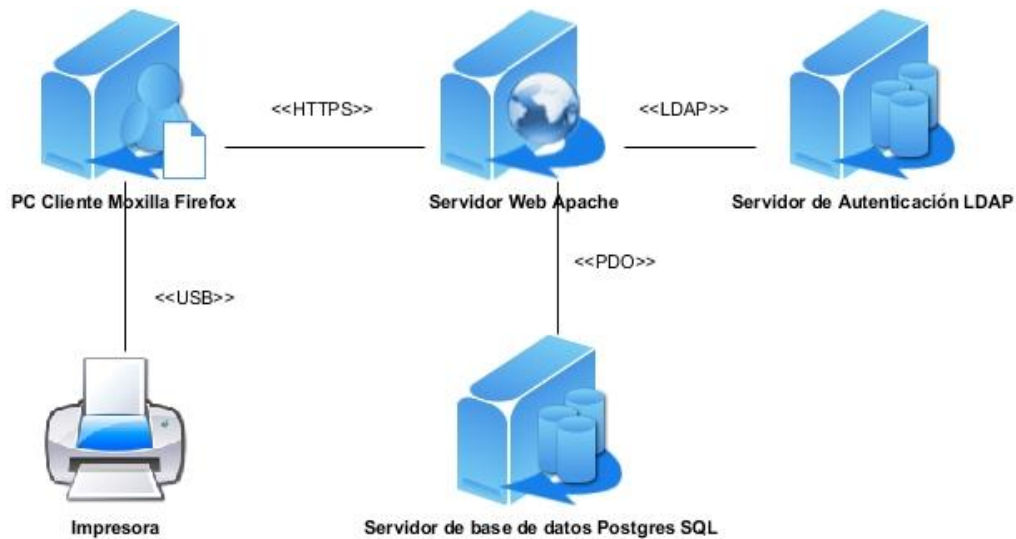


Figura 13. Diagrama de despliegue del sistema.

PC Cliente: computadora en la cual la aplicación se ejecutará a través de un navegador, en este caso se debe usar el Mozilla Firefox.

Servidor Web: radica la lógica de negocio de la aplicación. Servidor Web Apache 2.2.4, utilizando el lenguaje de programación del lado del servidor PHP 5.

Servidor de Autenticación: servidor a través del cual se realiza la autenticación de los usuarios del sistema pertenecientes al dominio uci.cu, en aras de lograr que no accedan al sistema usuarios no autorizados.

Servidor de Base de datos: servidor de Datos PostgreSQL 8.3, donde se encuentra la base de datos que utiliza el sistema, este puede estar instalado en la misma computadora donde se encuentra el servidor Web.

Impresora: utilizada para imprimir los reportes del sistema en caso de ser necesario.

3.6. Validación del diseño propuesto

Una métrica es un instrumento que cuantifica un criterio y persigue comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado al nivel del proyecto [53].

Para la evaluación de la calidad del diseño propuesto para el sistema se hizo un estudio de las métricas básicas inspiradas en la calidad del diseño orientado a objeto, en el mismo se abarcan atributos de calidad que permiten medir la calidad del diseño propuesto. Dentro de estos se encuentran [53]:

- ✓ **Responsabilidad:** consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

- ✓ **Complejidad de implementación:** consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ✓ **Reutilización:** consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ **Acoplamiento:** consiste en el grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica de Reutilización.
- ✓ **Complejidad del mantenimiento:** consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costos y la planificación del proyecto.
- ✓ **Cantidad de pruebas:** consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño y su relación con los atributos de calidad definidos son las siguientes:

1. Tamaño Operacional de Clase (TOC): se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

Resultados del instrumento de evaluación de la métrica Tamaño Operacional de Clase (TOC)

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos, ver **Figura 14**:

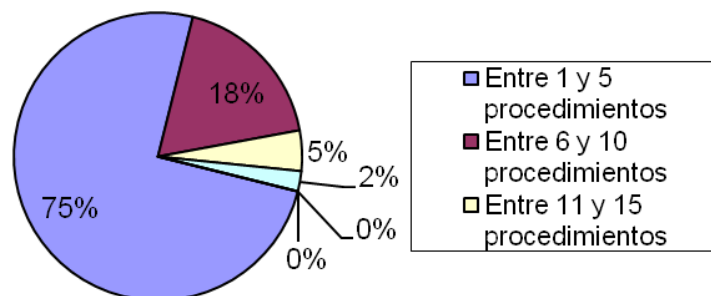


Figura 14. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad, ver **Figura 15**:

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

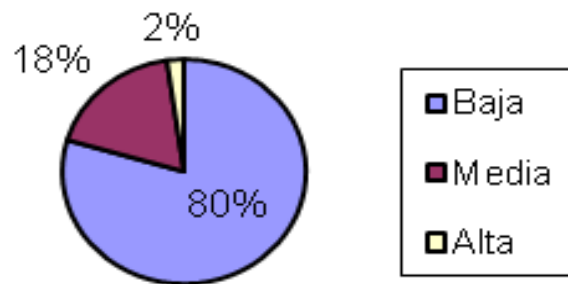


Figura 15. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación, ver **Figura 16**:

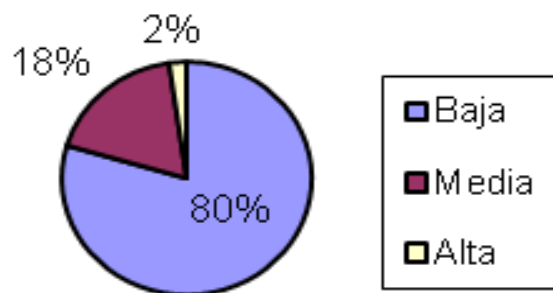


Figura 16. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización, ver **Figura 17**:

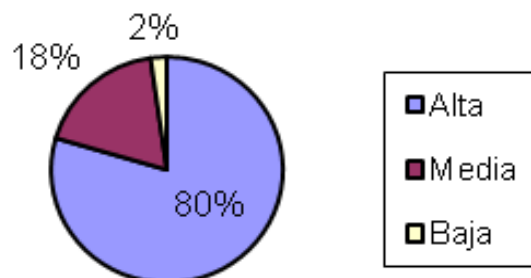


Figura 17. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para el sistema es simple y tiene una calidad

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

aceptable, teniendo en cuenta que la mayoría de las clases (75%) posee menos cantidad de operaciones que la media registrada en las mediciones. Los atributos de calidad se encuentran en un nivel satisfactorio en el 80% de las clases, de manera que se puede observar cómo se fomenta la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

2. Relaciones entre Clases (RC): dado por el número de relaciones de uso de una clase. Está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC)

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos, ver **Figura 18**:

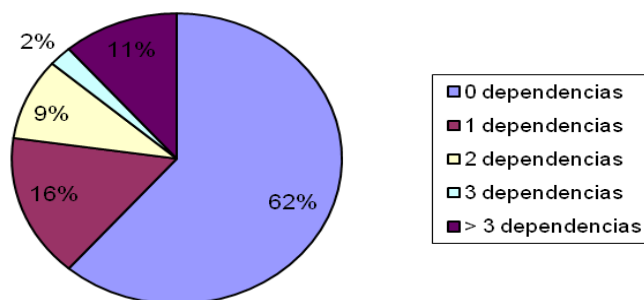


Figura 18. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento, ver **Figura 19**:

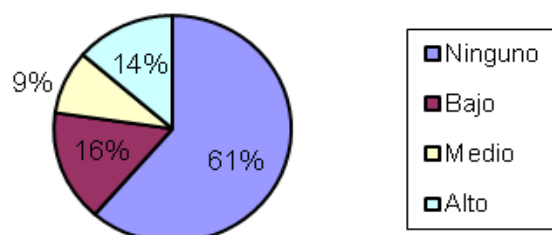


Figura 19. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de mantenimiento, ver **Figura 20**:

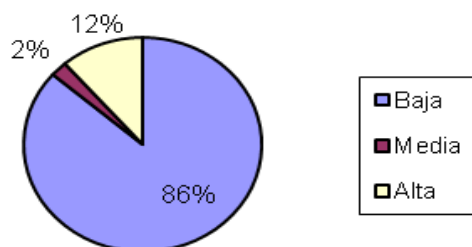


Figura 20. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de mantenimiento.

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de pruebas, ver **Figura 21**:

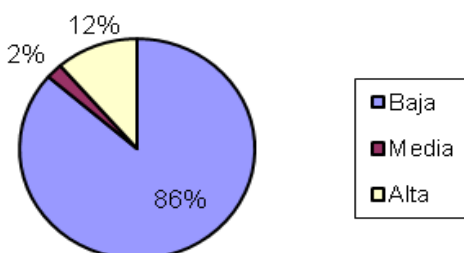


Figura 21. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de pruebas.

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización, ver **Figura 22**:

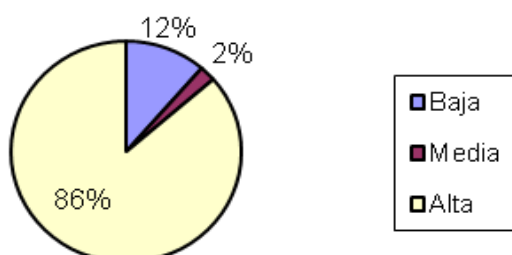


Figura 22. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el sistema es simple y tiene una calidad

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

aceptable, teniendo en cuenta que la mayoría de las clases (87%) poseen 2 o menos dependencias respecto a otras. Los atributos de calidad se encuentran en un nivel satisfactorio, en el 77% de las clases el grado de acoplamiento es mínimo, la Complejidad de mantenimiento, la Cantidad de pruebas y la Reutilización se comportan favorablemente para un 86% de las clases.

3.7. Pruebas de software aplicadas al sistema

Las pruebas de software son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador, probando el comportamiento del mismo [54].

3.7.1. Pruebas de caja blanca

La Prueba de Caja Blanca también se conoce como Prueba de Caja Transparente o de Cristal. Esta prueba consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento [53]. Dentro de la prueba de caja blanca se incluyen las Técnicas de Pruebas que serán descritas a continuación:

- ✓ **Prueba del Camino Básico:** permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos.
- ✓ **Prueba de Condición:** ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.
- ✓ **Prueba de Flujo de Datos:** se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.
- ✓ **Prueba de Bucles:** se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución de todos los bucles en sus límites operacionales.

La prueba de caja blanca empleada en la solución desarrollada fue la Prueba del Camino Básico, la cual permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa [53].

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

Para realizar esta técnica es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método `eliminarProrrogaAction()`, ver **Figura 23**, y el grafo de flujo asociado al mismo, ver **Figura 24**.

```
function eliminarProrrogaAction(){
    $var=false; //1
    $idcontrato=$this->_request->getPost('idcontrato'); //1
    $model = new ProrrogaModel(); //1
    $estado=$model->getEstadoProrroga($idcontrato); //1
    if($estado==1 ||$estado==4){ //2
        $nuevo=new ActividadesModel(); //3
        $actividades=$nuevo->getTodasDeUnCronograma($idcontrato); //3
        $cantidad=count($actividades); //3

        for($i=0;$i<$cantidad;$i++){ //4
            $actividad=Doctrine::getTable('Actividades')->find($actividades[$i][id]); //5
            $actmodel = new ActividadesModel(); //5
            $actmodel->Eliminar($actividad); //5
        } //6
        $cont=new ContratoModel(); //7
        $idprorroga=$model->getIdProrroga($idcontrato); //7
        $prorroga=Doctrine::getTable('Prorroga')->find($idprorroga); //7
        $prorrogamod = new ProrrogaModel(); //7
        $prorrogamod->Eliminar($prorroga); //7
        $idsolicitud=$cont->buscarIdSolicitud($idcontrato); //7
        $idcronograma=$cont->buscarIdCronog($idcontrato); //7
        $contrat=new ContratoModel(); //7
        $contrato=Doctrine::getTable('Contrato')->find($idcontrato); //7
        $contrat->Eliminar($contrato); //7
        $solicitudmod = new SolicitudModel(); //7
        $solicitud=Doctrine::getTable('Solicitud')->find($idsolicitud); //7
        $solicitudmod->Eliminar($solicitud); //7
        $cronogramamod = new ProrrogaModel(); //7
        $cronograma=Doctrine::getTable('Cronograma')->find($idcronograma); //7
        $cronogramamod->Eliminar($cronograma); //7
        $var=true; //7
    } //8
    echo(json_encode($var)); //9
} //10
```

Figura 23. Código del método `eliminarProrrogaAction()`.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

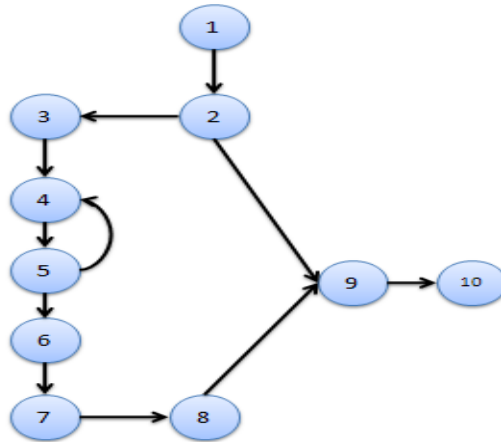


Figura 24. Grafo de flujo asociado al algoritmo eliminarProrrogaAction ().

Fórmulas para calcular la complejidad ciclomática:

1. $V(G) = (A - N) + 2$

Donde “**A**” es la cantidad de aristas y “**N**” la cantidad de nodos.

$$V(G) = (11 - 10) + 2$$

$$V(G) = 3$$

2. $V(G) = P + 1$

Siendo “**P**” la cantidad de nodos predicados (son los nodos de los cuales parten dos o más aristas).

$$V(G) = 2 + 1$$

$$V(G) = 3$$

3. $V(G) = R$

Donde “**R**” representa la cantidad de regiones en el grafo.

$$V(G) = 3$$

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 3, lo que significa que existen tres posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Número	Caminos básicos
1	1-2-9-10
2	1-2-3-4-5-6-7-8-9-10
3	1-2-3-4-5-4-5-6-7-8-9-10

Tabla 20. Caminos básicos del flujo.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

Posteriormente de haber determinado los caminos básicos se procede a ejecutar los casos de pruebas para cada uno de estos. Para definir los casos de prueba es necesario tener en cuenta:

- ✓ **Descripción:** se describe el caso de prueba y de forma general se tratan los aspectos fundamentales de los datos de entrada.
- ✓ **Condición de ejecución:** se especifica cada parámetro para que cumpla una condición deseada y así ver el funcionamiento del procedimiento.
- ✓ **Entrada:** se muestran los parámetros que serán la entrada al procedimiento.
- ✓ **Resultados Esperados:** se expone el resultado esperado que debe devolver el procedimiento después de efectuado el caso de prueba.

Caso de prueba para el camino básico # 1.

Descripción: el estado en el que se encuentra la prórroga asociada al contrato seleccionado no permite que la misma sea eliminada.

Condición de ejecución: el estado de la prórroga es “Aceptada”.

Entrada:

- ✓ \$idcontrato=9, representa el identificador del contrato seleccionado del cual se desea eliminar la prórroga.

Resultados Esperados: se espera que muestre un mensaje informando que la prórroga no puede ser eliminada, puesto que ya ha sido aceptada.

Resultados obtenidos: satisfactorio.

Caso de prueba para el camino básico # 2.

Descripción: el estado en el que se encuentra la prórroga asociada al contrato seleccionado permite que la misma sea eliminada, además presenta un cronograma asociado y una actividad contenida en este, los cuales también deberán ser eliminados.

Condición de ejecución: el estado de la prórroga es “Pendiente” o “Rechazada”.

Entrada:

- ✓ \$idcontrato=7, representa el identificador del contrato seleccionado del cual se desea eliminar la prórroga.

Resultados Esperados: se espera que muestre un mensaje informando que la prórroga fue eliminada satisfactoriamente.

Resultados obtenidos: satisfactorio.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

Caso de prueba para el camino básico # 3.

Descripción: el estado en el que se encuentra la prórroga asociada al contrato seleccionado permite que la misma sea eliminada, además presenta un cronograma asociado y varias actividades contenidas en este, los cuales también deberán ser eliminados.

Condición de ejecución: el estado de la prórroga es “Pendiente” o “Rechazada”.

Entrada:

- ✓ \$idcontrato=4, representa el identificador del contrato seleccionado del cual se desea eliminar la prórroga.

Resultados Esperados: se espera que muestre un mensaje informando que la prórroga fue eliminada satisfactoriamente.

Resultados obtenidos: satisfactorio.

3.7.2. Pruebas de caja negra

A este tipo de prueba también se le conoce como Prueba de Caja Opaca o Inducida por los Datos. Se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación, esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa [54].

En esencia permite encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las bases de datos externas.
- ✓ Errores de rendimiento.

Para cada uno de los requisitos funcionales fueron diseñados los casos de prueba, los cuales se pueden consultar en el expediente de proyecto del CEIGE, en el cual se encuentran todos los Diseños de Casos de Prueba realizados para cada requisito funcional del sistema.

El sistema fue sometido a dos iteraciones de pruebas de caja negra. En la primera iteración se detectaron 35 no conformidades, de las cuales 29 fueron de aplicación y 6 de documentación. Dentro de las no conformidades referentes a las de aplicación se encuentran 16 de validación, 12 de interfaz y 1 de funcionalidad. En cuanto a las de documentación se encontraron 2 no conformidades de redacción, 1 de ortografía y 3 de correspondencia. Todas estas no conformidades fueron resueltas seguidamente de haberlas detectado, lo que permitió que el sistema pasara a una segunda iteración en la que no se detectaron no conformidades, obteniendo resultados satisfactorios.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

La aplicación desarrollada fue presentada ante el cliente, el cual luego de haberla probado estuvo satisfecho con el producto entregado. Prueba de esto lo constituye el acta de aceptación emitida por este, en la cual consta que el sistema realizado cumple con las condiciones de calidad necesarias y satisface las necesidades plasmadas por el cliente, además contribuye a mejorar la gestión de las problemáticas de carácter científico de la facultad 3.

3.8. Validación de las variables presentes en el problema

El proceso de evaluación de las variables confiabilidad y seguridad presentes en el problema a resolver se realizará de forma cualitativa a partir de la aplicación del método QUASAR, ver **Tabla 22**.

El método QUASAR tiene como objetivo determinar el cumplimiento de requisitos no funcionales de atributos de calidad. El mismo se realiza mediante casos de calidad teniendo en cuenta tres elementos fundamentales, los cuales son [56]:

- Demandas: son definidas como afirmaciones que hace el equipo de desarrollo acerca de si el software cumple adecuadamente con las metas de calidad o si tiene en cuenta un conjunto de requisitos asociados y relacionados con la calidad.
- Argumentos: son las razones por las cuales el software debe cumplir con las demandas que se hacen acerca de determinados requisitos relacionados con la calidad.
- Evidencias: son las evidencias que se presentan para respaldar los argumentos que se realizan. Las evidencias apoyan las demandas de que el software logra las metas de calidad y cumple con los requisitos de calidad. Estas evidencias pueden ser de dos tipos: documentos oficiales o demostraciones observables. Estas últimas pueden estar basadas en escenarios, pruebas o simulaciones.

Para lograr la confiabilidad y seguridad de la información (ver **Tabla 21**) el sistema que se presenta debe cumplir con los requisitos no funcionales (RNF5, RNF6, RNF7 y RNF8), los cuales fueron descritos en el **epígrafe 2.5.1**:

Factores	Descripción
Confiabilidad	La capacidad del producto de software para mantener un nivel de ejecución especificado cuando se usa bajo las condiciones especificadas [57].

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

Seguridad	Capacidad del producto de software para proteger información y los datos, para que personas o sistemas desautorizados no puedan leer o pueden modificar los mismos, y las personas o sistemas autorizados tenga el acceso a ellos [57].
------------------	---

Tabla 21. Definición de los factores de calidad Confiabilidad y Seguridad.

Casos de calidad

Confiabilidad y Seguridad

1. Demandas:

- ✓ RNF4: proteger la información manejada por el sistema de accesos no autorizados.
- ✓ RNF5: garantizar que las funcionalidades del sistema se muestren de acuerdo al rol del usuario que esté activo.
- ✓ RNF6: la base de datos debe estar fraccionada sobre varios esquemas, dividiendo así de una forma lógica las funcionalidades, evitando la pérdida total de la información en caso de algún accidente o ataque.
- ✓ RNF7: permitir autenticarse a través de un Servidor de Dominio.

2. Argumentos:

Para garantizar la confiabilidad y seguridad de la información del sistema se validó que solo puedan acceder al sistema los usuarios registrados con anterioridad en la base de datos, que pueden o no pertenecer al dominio uci.cu. Además se validó que solo el usuario que creó la problemática puede eliminarla y modificarla. Todo lo descrito anteriormente se realizó con el objetivo de evitar la destrucción, modificación o acceso no autorizado a la información por personas indebidas, lo cual atentaría contra una adecuada gestión de las problemáticas presentes en el sistema.

También se fraccionó la base de datos del sistema en varios esquemas, evitando la pérdida total de la información en caso de algún accidente o ataque.

3. Evidencia:

Las evidencias fueron demostradas a través de escenarios de prueba realizados para cada uno de los requisitos no funcionales que corresponden con los factores de calidad antes mencionados, a través de los cuales se observó el cumplimiento de estos factores por parte del sistema, ver **Tabla 22**.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

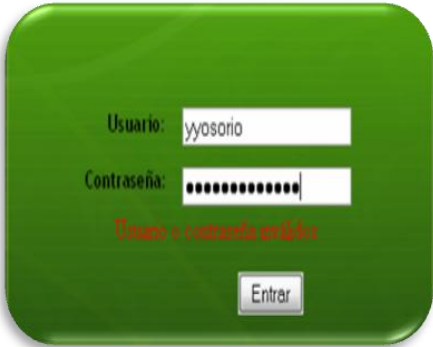


Variables	RNF	Escenario de prueba	Procedimiento realizado	Meta (Resultado esperado)
Confiabilidad y seguridad	RNF7	¿Cualquier usuario puede acceder al sistema?	Se validó a través del método <code>authenticateUserApirey</code> (<code>\$user</code> , <code>\$password</code> , <code>\$sistema</code>), que solo puedan acceder al sistema los usuarios registrados con anterioridad en la base de datos, que pueden o no pertenecer al dominio UCI.	
	RNF4 RNF5	¿Un usuario que no es autor de una problemática puede eliminar o modificar la misma?	Se validó que solo el usuario que creó la problemática puede eliminarla y modificarla. Esto se realizó teniendo en cuenta el usuario que se encuentra autenticado en el sistema y el autor de dicha problemática. Un ejemplo lo constituye el siguiente método: <code>function eliminar(idproblema).</code>	
	RNF6	¿En caso de un ataque a la base de datos se perdería toda la información?	La base de datos se encuentra fraccionada en varios esquemas, evitando la pérdida total de la información en caso de algún accidente o ataque.	

Tabla 22. Validación de las variables confiabilidad y seguridad.

3.9. Conclusiones parciales

En este capítulo se realizó la implementación y validación del sistema, lo que permitió obtener importantes resultados como:

- ✓ Se definieron los estándares de implementación a utilizar, para así hacer más fácil el entendimiento del código por los programadores y el futuro mantenimiento a la aplicación.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

- ✓ Fueron expuestas las métricas para validar el diseño, las cuales arrojaron resultados satisfactorios sobre el diseño realizado, demostrando que este era simple y los atributos de calidad alcanzaban niveles favorables.
- ✓ Se realizaron al sistema pruebas de caja blanca y caja negra, así como también se validaron las variables incluidas en el problema a resolver. Todas estas pruebas arrojaron resultados satisfactorios, demostrándose con ello la calidad del sistema realizado.

CONCLUSIONES GENERALES

CONCLUSIONES GENERALES

Una vez terminado el presente trabajo de diploma se puede llegar a la conclusión de que se le dio cumplimiento a todos los objetivos propuestos, para esto:

- ✓ Se analizaron sistemas informáticos tanto nacionales como internacionales vinculados a la gestión de la información, evidenciándose de esta manera la no existencia de una solución informática capaz de cubrir todas las funcionalidades requeridas para realizar una adecuada gestión de las problemáticas de carácter científico que afectan a la facultad 3 de la UCI, lo cual hizo necesario el desarrollo de un sistema informático que se ajuste sobre todo a los intereses y estrategias de trabajo de dicha facultad.
- ✓ Se realizó el análisis y el diseño del sistema de gestión para el banco de problemas de la facultad 3, permitiendo recoger todas las necesidades de los clientes, además de lograr un mejor entendimiento de los procesos que fueron informatizados, garantizando una comprensión exacta sobre las funcionalidades a informatizar.
- ✓ El sistema realizado fue expuesto a diferentes pruebas, las cuales arrojaron resultados satisfactorios demostrando la calidad del sistema desarrollado, puesto que permite lograr la confiabilidad y seguridad del proceso de seguimiento de las diferentes problemáticas de carácter científico de la facultad 3, lográndose así una adecuada gestión de las mismas.

RECOMENDACIONES

RECOMENDACIONES

Teniendo como base los resultados de este trabajo y la experiencia adquirida durante el desarrollo del mismo, se recomienda:

- ✓ Realizar mejoras en el sistema a la hora de generar las notificaciones, de forma tal que estas sean enviadas directamente al correo de cada usuario involucrado con dichas notificaciones.
- ✓ Debido a que actualmente no están definidos cuáles son los indicadores que tributarán a cada una de las salidas que pueden tomar las diferentes problemáticas originadas, se recomienda establecer dichos indicadores y su relación con cada salida.
- ✓ Integrar el sistema a la Plataforma Informativa de la facultad 3 (SO3), la cual tiene como objetivo centralizar el uso de las diferentes aplicaciones realizadas para dicha facultad.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

1. **Graells, Dr. Pere Marquès.** LAS TIC Y SUS APORTACIONES A LA SOCIEDAD. LAS TIC Y SUS APORTACIONES A LA SOCIEDAD. [En línea] Departamento de Pedagogía Aplicada, Facultad de Educación, UAB, 23 de Marzo de 2008. [Citado el: 12 de Enero de 2012.] <http://peremarques.pangea.org/tic.htm>.
2. **Vaca, María Isabel Pachano.** EL DESARROLLO DE LA GESTIÓN DOCUMENTAL QUE PERMITA SU FÁCIL ACCESO EN LA DIRECCIÓN DE PLANIFICACIÓN DEL H. GOBIERNO PROVINCIAL DE TUNGURAHUA EN EL PRIMER SEMESTRE DEL AÑO 2010. Ambato - Ecuador : UNIVERSIDAD TÉCNICA DE AMBATO, 2010.
3. **López, Elvert Maldonado.** DESARROLLO DE REPORTES PARA EL SISTEMA DE GESTIÓN DE CIENCIA, TECNOLOGÍA E INNOVACIÓN SIGCTI. Granma : Facultad Regional Granma de la Universidad de las Ciencias Informáticas., 2010.
4. **Rosabal, Antonio San Juan.** CUADERNOS DE EDUCACIÓN Y DESARROLLO. [Online] mayo 2011. [Cited: junio 24, 2012.] <http://www.eumed.net/rev/ced/27/jrrg.htm>.
5. **Barcelona, Universidad de GREC.** [Online] [Cited: Abril 06, 2012.] <https://webgrec.ub.edu/>.
6. **Cabrera, Francisco Manuel Solís.** Comunidad de Andalucía. [Online] [Cited: Abril 07, 2012.] http://www.madrimasd.org/informacionidi/revistas/monograficos/monografias/monografia22/las_CA_frente_IDi-sistema_informacion_cientifico_andalucia.pdf.
7. **Iván F. Palomo, Carlos G. Veloso y Rodolfo F. Schmal.** Sistema de Gestión de la Investigación en la Universidad de Talca, Chile. Talca : Universidad de Talca, Chile., 2007.
8. **Linea., Gestión de proyectos de Investigación en.** Universidad Pedagógica Nacional . [Online] 2005. [Cited: Febrero 26, 2012.] <http://www.pedagogica.edu.co/pgil/pgil.php>.
9. **O.E.Sánchez, B.Garea,S.Lantigua y otros.** Sistema de Información para la Gestión de Programas de Ciencia e Innovación en Cuba. 2008.
10. **Batista, Ing. Yunaysy Ortiz.** Sistema integrado de gestión de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas . La habana : Universidad de las Ciencias Informáticas, 2011.
11. **SIGENU.** Sigenu DSS. [Online] Julio 2011. [Cited: Abril 09, 2012.] <http://www.isch.edu.cu/sites/default/files/Manual%20de%20Usuario%20SigenuDSS%20v1.0.0.pdf>.
12. **Yunaysy Ortiz Batista, Yeslidier López Reinoso, Yordanis Medina León y otros.** Propuesta de solución para la gestión de la información de la actividad de ciencia, tecnología e innovación en la Universidad de las Ciencias Informáticas. La Habana : UCI, 2010.

BIBLIOGRAFÍA

13. **José H. Canós, Patricio Letelier yM^a Carmen Penadés.** Metodologías Ágiles en el Desarrollo de Software. Valencia : Universidad Politécnica de Valencia., 2006.
14. **Yanet Vega, L. S. 2009.** Definición del ciclo de vida del proyecto. La Habana,2009.
15. **EcuRed.** [En línea] [Citado el: 22 de Febrero de 2012.] <http://www.ecured.cu/index.php/Codelgniter>.
16. **Ing.Oiner Gómez Baryolo, Ing.Yoandry Morejón Borbón,Ing.Darien García Tejo.** ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE. Ciudad de la Habana : s.n.
17. **Doctrine.** Welcome to the Doctrine Project. [En línea] [Citado el: 18 de Enero de 2012.] <http://www.doctrine-project.org/>.
18. **Maikel Yulier Sañudo Clemente, Leandro Luis Rojas Cuesta.** Desarrollo de la versión 2.0 de la herramienta Doctrine Generator para la generación de ficheros de mapeo basado en Doctrine empleando la tecnología PHP. Ciudad de la Habana : s.n., 2010.
19. **Larman, Craig.** El mundo Informático. El mundo Informático. [En línea] 17 de Agosto de 2006. [Citado el: 15 de Enero de 2012.] <http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/>.
20. **Gracia, Joaquin.** IngenieroSoftware. IngenieroSoftware. [En línea] 27 de Mayo de 2005. [Citado el: 15 de Enero de 2012.] <http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php>.
21. **EcuRed.** EcuRed. [En línea] [Citado el: 18 de Enero de 2012.] http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_y_arquitectura.
22. **Albacete.** Albacete. [En línea] 2002. [Citado el: 23 de Enero de 2012.] <http://www.info-ab.uclm.es/asignaturas/42579/cap4/Comportamiento.htm>.
23. **Sánchez-Segura, Ana M. Moreno y Maribel.** Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento de Arquitectónico. Madrid : Universidad Politécnica de Madrid, España, 2010.
24. **Hernández, Elier Cruz.** Implementación de los Componentes Depreciación y Submayor del Subsistema Activo Fijo Tangible del Sistema integral de Gestión Cedrux. Ciudad de la Habana : s.n., 2009.
25. **Modeler, BizAgi Process.** BizAgi Process Modeler. [En línea] [Citado el: 3 de Febrero de 2012.] <http://www.bizagi.com/docs/BPMNbyExampleSPA.pdf>.
26. **Riesco, Prof. Daniel.** UML. [En línea] 2004. [Citado el: 5 de Febrero de 2012.] <http://sel.unsl.edu.ar/licenciatura/ingsoft2/UMLIntroduccion.pdf>.
27. **Garcia, Yunesky del Rio.** Implementación del Módulo Índice de Peligrosidad Pre-Delictiva del Proyecto Sistema de Gestión Fiscal. Ciudad de la Habana : s.n., 2010.

BIBLIOGRAFÍA

28. **openformats.org**. openformats.org. [En línea] 27 de Agosto de 2010. [Citado el: 10 de Febrero de 2012.] <http://www.openformats.org/es9>.
29. **Valdés, Damián Pérez**. Maestros del Web. [En línea] 2007. [Citado el: 14 de Febrero de 2012.] <http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/>.
30. **Alvarez, Miguel Angel**. desarrolloweb.com. [En línea] 13 de Junio de 2001. [Citado el: 15 de Febrero de 2012.] <http://www.desarrolloweb.com/articulos/449.php>.
31. **Albert**. Albert Módulo de Software. [En línea] Julio de 2008. [Citado el: 21 de Febrero de 2012.] <http://albertcm.wordpress.com/ajax/>.
32. **Aranzadi.es**. Aranzadi.es. [En línea] [Citado el: 24 de Enero de 2012.] <http://www.aranzadi.es/index.php/informacion-juridica/informacion-interes/glosario-de-terminos-sobre-internet-y-spam>.
33. **Ciberaula**. Ciberaula. [En línea] 2010. [Citado el: 25 de Enero de 2012.] http://linux.ciberaula.com/articulo/linux_apache_intro/.
34. **González, Carlos D**. Curso PostgreSQL, SQL avanzado y PHP. [En línea] Febrero de 2012. <http://www.usabilidadweb.com.ar/postgre.php>.
35. **Manager, Free Download**. Free Download Manager. [En línea] 5 de Marzo de 2007. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.
36. **Cerda, Felipe**. NetBeans 6.5 . [En línea] [Citado el: 9 de Febrero de 2012.] http://api.ning.com/files/PcRFEwSyAz6w2k4r-KsPqXzHa3EJ6JtOUDOCxjxcpwJJFWH8XaIgnfUFGtj2TURFJc3LgS7W9tCq-NsM68o3ZBHIMDWV12-W/netbeans65es_cl.pdf.
37. **Palacios, Rodolfo Vázquez**. Scribd. [En línea] 2 de Septiembre de 2010. [Citado el: 17 de Febrero de 2012.] <http://es.scribd.com/doc/37957736/Mozilla-Firefox-Original>.
38. **José H. Canós, Patricio Letelier y M^a Carmen Penadés**. Scribd. Scribd. [En línea] [Citado el: 13 de Enero de 2012.] <http://es.scribd.com/Jert%20Zee/d/8530565-Metodologias-agiles>.
39. **Foundation, The Apache Software**. The Apache Software Foundation. [En línea] [Citado el: 27 de Enero de 2012.] <http://www.apache.org/>.
40. **PosgreSQL**. PostgreSQL. [En línea] [Citado el: 28 de Enero de 2012.] <http://www.postgresql.org/>.
41. **Paradigm, Visual**. Visual Paradigm. [En línea] [Citado el: Febrero de 1 de 2012.] <http://www.visual-paradigm.com/product/vpuml/>.
42. **Juan Diego Pérez Jiménez, Amador Durán Toro ,Antonio Ruiz Cortes**. ¿Por qué OMG ha elegido BPMN para modelar de Procesos de. Sevilla : Universidad de Sevilla,España, 2005.

BIBLIOGRAFÍA

43. **IBM.** IBM. [En línea] [Citado el: 6 de Febrero de 2012.] <http://www-01.ibm.com/software/rational/uml/>.
44. **NetBeans.** NetBeans. NetBeans. [En línea] [Citado el: 6 de Febrero de 2012.] <http://netbeans.org/>.
45. **González, Luis.** Proyecto EATS. [En línea] 17 de Enero de 2007. [Citado el: 12 de Febrero de 2012.] <http://eats.wordpress.com/2007/01/17/lenguajes-del-lado-servidor-y-del-lado-cliente/>.
46. **Lapuente, María Jesús Lamarca.** HTML. [En línea] 19 de Noviembre de 2011. [Citado el: 13 de Febrero de 2012.] <http://www.hipertexto.info/documentos/html.htm>.
47. **Quin, Liam.** W3C. [En línea] 24 de Enero de 2012. [Citado el: 15 de Febrero de 2012.] <http://www.w3.org/XML/>.
48. **mozilla.** mozilla Firefox. [En línea] [Citado el: 16 de Febrero de 2012.] <http://www.mozilla.org/es-ES/firefox/central/>.
49. **Fiore, Santiago.** Análisis de la tecnología AJAX y Web 2.0. Buenos Aires : Universidad Nacional de Luján., 2006.
50. **López, Héctor A. Bareiro y Ricardo G.** Frameworks, conceptos.Comparativa entre Ruby on Rails, Kumbia y Symfony. [En línea] 2007 de Octubre de 12. [Citado el: 2012 de Febrero de 23.] <http://www.sicama.uma.es/sicama/independientes/argentina08/Bareiro-Lopez/index.htm>.
51. **Sparx Systems** - Tutorial UML 2 - Diagrama de Componentes. [En línea] 2000-2007. [Citado el: 5 de febrero de 2012.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html.
52. **EcuRed: Enciclopedia cubana.** [En línea] [Citado el: 12 de Febrero de 2012.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.
53. Pressman, Roger S. Ingeniería de Software, Un enfoque práctico. 2005.
54. **EcuRed: Enciclopedia cubana.** [En línea] [Citado el: 20 de Mayo de 2012.] http://www.ecured.cu/index.php/Pruebas_de_software.
55. **Guillerón, Gastón.** GLN Ingeniería & Consultoría. [Online] julio 12, 2009. [Cited: junio 24, 2012.] <http://glnconsultora.com/blog/?p=3>.
56. **Donald Firesmith, Software Engineering Institute (SEI).** QUASAR: A Method for the Quality Assessment of Software-Intensive System Architectures. 2006.
57. **Normalización, Oficina Nacional de.** NORMA CUBANA. Ciudad de La Habana : s.n., 2005

GLOSARIO DE TÉRMINOS

GLOSARIO DE TÉRMINOS

- ✓ **CSS:** Cascading Style Sheets u Hojas de Estilo en Cascada, es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación.
- ✓ **SOA:** la arquitectura orientada a servicios (SOA), es un marco de trabajo conceptual que permite a las organizaciones unir los objetivos del negocio con la infraestructura de las Tecnologías de Información (TI), integrando los datos y la lógica de negocio de sus sistemas separados.
- ✓ **Caso de prueba:** es un conjunto de entradas de prueba, condiciones de ejecución y resultados esperados.