Universidad de las Ciencias Informáticas

"Facultad 3"

Título: Sistema de diagnóstico a la producción: Módulos Diagnóstico y Administración.

Trabajo de Diploma para optar por el título de

Ingeniero de Ciencias Informáticas

Autor:

Rubén Alejandro Cartaya Ferro.

Tutor:

Ing. Mariela Milagros Bony Fernández

Ciudad de La Habana, Junio del 2011.

¨Año 53 de la Revolución¨

Es dudoso que el género humano logre crear un enigma que el mismo ingenio humano no resuelva.

Edgar Allan Poe

El gran estilo nace cuando lo bello obtiene la victoria sobre lo enorme.

Friedrich Wilhelm Nietzsch

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 _____

Rubén Alejandro Cartaya Ferro

_____

Firma del Autor

Ing. Mariela Milagros Bony Fernández

_____

Firma del Tutor

Resumen

En la Universidad de las Ciencias Informáticas (UCI) se lleva acabo anualmente un proceso de Diagnósticos Organizacionales (DO) a la producción. Dichos diagnósticos son de vital importancia para la alta gerencia de la universidad a la hora de conocer y tomar decisiones en dicha área. El Grupo de Diagnóstico de CALISOFT (Centro de calidad para soluciones tecnológicas) el cual es el encargado de organizarlos y realizarlos decidió la creación de un software con el fin automatizar dicho proceso. Facilitando así la aplicación de futuros diagnósticos.

Como resultado surgió el Sistema de Diagnóstico a la Producción cuya columna vertebral es el modulo Diagnóstico y Administración objetivo del presente trabajo. Este módulo posibilita la creación de un diagnóstico, la gestión de las evaluaciones, la creación de los usuarios así como la seguridad entre otras funcionalidades. Se utilizó para su creación el lenguaje de programación PHP5 y utiliza el gestor de base de datos PostgreSQL para el almacenamiento y gestión de los datos.

Índice

Índice de Figuras

Índice de Tablas

Introducción

En la actualidad la velocidad con que acontecen los adelantos científicos, la necesidad de los países pobres de insertarse en el mercado internacional, la acumulación de capital en los grandes monopolios, la inestabilidad en las condiciones climáticas, además de una creciente desigualdad social, han obligado a cualquier empresa con esperanza de un verdadero crecimiento a la necesidad de la realización una correcta planificación y estudio de factibilidades. La toma de decisiones y la gestión de la información en estas empresas son elementos que deben ser controlados por parte de las entidades administrativas ya que dependen de la eficacia y eficiencia productiva.

Cuba, producto del bloqueo, ha necesitado aún más una correcta planificación en todos los sentidos. Esto conllevó primeramente una rectificación de errores en el campo empresarial y posteriormente al perfeccionamiento empresarial. Dentro de todos estos procesos ha destacado hasta el momento la aplicación del concepto de DO a la producción.

La UCI juega un papel fundamental en el desarrollo de la industria de software del país. En ella se llevan a cabo diferentes procesos dirigidos a la actividad productiva, lo cual hace necesario que se tomen decisiones acertadas de acuerdo al estado en que se encuentren las áreas productivas de la universidad. Existen diferentes técnicas y herramientas para el apoyo a la toma de decisiones de la Alta Gerencia de las empresas, una de las más usadas es el DO. Gracias al DO se puede determinar aquello que no está funcionando bien en la empresa y a partir de la información recopilada hacer una toma de decisiones más acertada en correspondencia a los fines de la empresa.

El proceso de DO se realiza de forma anual en la UCI. Hasta el momento se han realizado 4 actividades de este tipo organizadas y dirigidas por el Grupo de Diagnóstico de CALISOFT, donde se han evidenciado avances en la preparación, uso de herramientas, las expectativas del cliente y la presentación de los resultados. Pero en la forma de almacenarlos y la posibilidad de acceder a la información de los diagnósticos anteriores de forma rápida ha sido desfavorable.

En el año 2006 se realizó el primer DO a todos los proyectos de la universidad mediante una encuesta. Los resultados no fueron correctamente documentados y no existía un procedimiento que guiara las actividades a realizar, lo que trajo como consecuencia que el DO desarrollado en diciembre del 2007 no tuviera su base en un análisis detallado de los mismos. En este último se usó el Moodle para la aplicación de encuestas y los resultados se guardaron en un archivo exportado del Moodle. Además se

realizó un levantamiento de información y los datos recopilados fueron almacenados en Excel. En el trascurso de esta tarea se comenzó a documentar el procedimiento para sustentar el proceso.

En el 2008 hubo un avance en la organización, por esta razón actividades que presentaron dificultades en los DO anteriores tuvieron una mayor calidad. Para esta fecha ya existía una primera versión del procedimiento; los datos fueron recopilados en Excel y se empleó el gestor de base de datos SQL Server 2000 para guardar la información del levantamiento, pero en el mismo las tablas no estaban relacionadas. Además se usó nuevamente el Moodle para la realización de encuestas.

En el año 2009, con el objetivo de identificar fortalezas y oportunidades de mejora en los proyectos, se elaboró un sistema de indicadores para una mejor interpretación de los resultados, logrando un análisis profundo de las estadísticas. Para el procesamiento de la información se utilizó la Plataforma Moodle, el paquete Microsoft Office, el gestor de base de datos SQL Server 2000 y el IPP-1000: 2009 Diagnóstico a las organizaciones productivas. Se comenzaron a utilizar los términos indicadores y criterios para mostrar los resultados solicitados por la alta gerencia.

A través de los años el grupo de CALISOFT ha ido ganando en claridad y experiencia en la realización de los DO, esto ha ayudado a identificar las causas más comunes que afectan el proceso de DO a la producción en la UCI, entre ellas se encuentran.

- Déficit de personal.
- Sobrecarga de trabajo.
- Ineficiente gestión de la información.
- Ausencia de mecanismos de re-planificación.
- Tiempo estimado irreal.
- Incumplimiento en la recolección de datos.
- Uso inadecuado de las tecnologías.

Todo esto conllevó a un consumo mayor de tiempo, mayor esfuerzo de los involucrados, además de un incremento en el uso de los recursos; lo cual afectó directamente la toma de decisiones de la alta gerencia de la universidad.

Luego de analizar la situación problémica expuesta anteriormente se determina como **problema de la investigación que:** La forma en que se lleva a cabo el proceso de DO a la producción en la UCI conlleva a un incremento en el empleo de recursos, aumento del esfuerzo y mayor consumo de tiempo

por parte del Grupo de Diagnóstico de CALISOFT, lo cual afecta directamente la toma de decisiones de la alta gerencia de la universidad.

Se establece como **objeto de estudio:** la gestión de la información en los DO, y con la finalidad de dar solución al problema anteriormente expuesto se define como **objetivo general:** desarrollar los módulos Diagnóstico y Administración del Sistema de Diagnóstico a la Producción en la UCI para reducir el uso de recursos, disminuir el esfuerzo y minimizar el tiempo empleado por parte del Grupo de Diagnóstico de CALISOFT. A partir del objeto de estudio planteado se define como **campo de acción:** el proceso de desarrollo de software para la gestión de información del DO a la Producción de la UCI.

Además de los elementos expuestos anteriormente se especifica la siguiente **idea a defender:** con la creación de un sistema de DO a la producción se reducirá el uso recursos, se disminuirá el esfuerzo y se minimizará el tiempo empleado por parte del Grupo de Diagnóstico de CALISOFT.

Del objetivo general se derivan los siguientes **objetivos específicos**:

- Levantar los requisitos del sistema del “Sistema de Diagnóstico a la Producción: módulos Diagnóstico y Administración”.
- Realizar el análisis y diseño del “Sistema de Diagnóstico a la Producción: módulos Diagnóstico y Administración”.
- Implementar el “Sistema de Diagnóstico a la Producción: módulos Diagnóstico y Administración”.
- Validar la solución del “Sistema de Diagnóstico a la Producción: módulos Diagnóstico y Administración”.

Para dar solución a los objetivos específicos se siguen las siguientes **tareas de la investigación:**

1. Análisis del proceso de DO a la actividad productiva de la UCI.
2. Análisis del proceso de desarrollo de software para DO.
3. Análisis de las tecnologías y herramientas que se pueden utilizar para la elaboración de la solución de software alineado con las políticas de la Universidad.
4. Levantamiento de los requisitos del “Sistema de Diagnóstico a la Producción: módulos Diagnóstico y Administración” para determinar qué actividades se pueden automatizar dentro del proceso.
5. Realización del análisis y diseño del “Sistema de Diagnóstico a la Producción: módulos Diagnóstico y Administración”.

6. Implementación del “Sistema de Diagnóstico a la Producción: módulos Diagnóstico y Administración”.
7. Validación de la solución del “Sistema de Diagnóstico a la Producción: módulos Diagnóstico y Administración”.

Métodos de investigación científica:

Durante el desarrollo de la investigación se emplearán los siguientes métodos teóricos:

- Histórico lógico: se hace uso de este método principalmente en el estudio del estado de arte ya que se debe investigar la evolución del software orientado al apoyo a la toma de decisiones mediante los diagnósticos.
- Hipotético deductivo: se parte de la idea a defender que no es más que un caso especial de la hipótesis para deducir qué es lo que se debe a hacer y de esta forma validar la misma.
- Modelación: se utiliza para modelar los procesos del negocio así como las funcionalidades del sistema y relaciones entre los componentes del mismo.

Estructura de los capítulos:

Capítulo 1: Fundamentación Teórica.

En este capítulo se definen los conceptos y términos fundamentales referentes a los DO así como un estudio del estado del arte de los sistemas de gestión de los mismos. Además se realiza una descripción de las tecnologías, herramientas y metodologías a utilizar para el desarrollo del sistema.

Capítulo 2: Características del sistema.

Incluye la descripción de los procesos del negocio que serán automatizados, así como sus actores y trabajadores. Se muestran las reglas que debe cumplir el negocio, así como los modelos de objetos y diagramas de actividades de cada caso de uso. También se identifican los actores que intervienen y las funcionalidades que brinda el sistema, teniendo como resultado los requisitos funcionales y no funcionales del mismo, que darán solución a los problemas existentes.

Capítulo 3: Diseño del Sistema.

Contiene todo lo relacionado al diseño del sistema, que incluye los diagramas de clases del diseño.

Capitulo 4: Implementación y Prueba.

Contiene la implementación del software que incluye el diagrama de despliegue, los diagramas de componentes y el patrón de arquitectura empleado, estilos de la programación, así como las pruebas basadas en caso de pruebas

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En el presente capítulo se definen los conceptos y términos fundamentales referentes a los DO así como un estudio del estado del arte de los sistemas de gestión de los mismos. Además se realiza una descripción de las tecnologías, herramientas y metodologías a utilizar para el desarrollo del sistema.

1.2 Gestión de Diagnósticos Organizacionales

1.2.1 ¿Qué son los Diagnósticos Organizacionales?

Se puede definir al DO como un proceso analítico que permite conocer la situación real de la organización en un momento dado para descubrir problemas y áreas de oportunidad, con el fin de corregir los primeros y aprovechar las segundas. El DO no es un fin en sí mismo, sino que es el primer paso esencial para perfeccionar el funcionamiento comunicacional de la organización. (1)

El concepto DO se inscribe dentro de un proceso de gestión preventivo y estratégico. Se constituye como un medio de análisis que permite el cambio de una empresa, de un estado de incertidumbre a otro de conocimiento, para su adecuada dirección, por otro lado es un proceso de evaluación permanente de la empresa a través de indicadores que permiten medir los signos vitales. (2)
El DO es una herramienta de la dirección y se corresponde con un proceso de colaboración entre los miembros de la organización y el consultor para recabar información pertinente, analizarla e identificar un conjunto de variables que permitan establecer conclusiones. (3)

Las bases del DO es que al igual que las personas, las empresas o instituciones deben someterse a exámenes periódicos para identificar posibles problemas antes de que éstos se tornen graves. Estos exámenes periódicos constituyen un sistema de control que permite optimizar el funcionamiento de las empresas o instituciones. Al ser identificados los problemas en el funcionamiento de la empresa, surgen acciones dirigidas a su eliminación o disminución que en conjunto constituyen una parte importante de la planeación operativa.

El objetivo principal del DO radica en cuantificar el estado de madurez actual de la organización con los estándares nacionales o internacionales que debería manejar la empresa, identificando de una manera rápida, precisa y concisa las áreas potenciales de desarrollo en ella.

"El DO determina lo que hay que medir y por qué, en correspondencia con la estrategia vigente en la organización y establece las alternativas de solución que mejor se adapten a la situación. Es sinónimo de saber dónde y cómo mejorar el desempeño. Debe ser aprovechado para actuar por adelantado sobre los procesos, antes de que ocurran las desviaciones, indicando los posibles obstáculos que se opondrán al rumbo estratégico fijado, creando las condiciones para que los resultados coincidan con los objetivos". (4)

1.2.2 Gestión de Diagnósticos Organizacionales

A nivel mundial son múltiples las organizaciones que realizan DO. El origen de estos pueden estar dados por el proceso natural de crecimiento de la organización, el proceso natural de deterioro de la organización , la empresa ha decidido encarar el problema de la productividad y la calidad, la organización ha sido sometida a cambios de importancia, por ejemplo, la innovación, la adecuación a nuevas situaciones y desafíos tecnológicos o el aumento de la complejidad del entorno de la organización, demanda un cambio correspondiente en la complejidad propia de la organización.

Entre las empresas que han realizado DO podemos citar a las empresas constructoras chilenas . La iniciativa surgió de un proyecto del Centro de Excelencia en Gestión de Producción de la Pontificia Universidad Católica de Chile (GEPUC) e incluye, además de la participación de empresas del sector, la participación activa de la Cámara Chilena de la Construcción, organismos estatales y centros de investigación y desarrollo nacionales y extranjeros. El DO se llevó a cabo mediante "focus groups" con administradores de obra, entrevistas a profesionales y gerentes, y formularios de evaluación. Arrojando como resultado la falta de conocimiento respecto al comportamiento humano en organizaciones productivas, la carencia de aplicación de técnicas existentes relacionadas con su gestión y la inexistencia de un área funcional que se preocupe de éste tema dentro de las empresas. (5)

Otras empresas que han realizado DO son las pequeñas y medianas empresas en Bahía de Huatulco. Con el objetivo de identificar la situación de la comunicación interna dentro de las empresas. Permitiéndole analizarlas y comprenderlas con el fin de identificar estrategias para mejorar sus procesos y así poder alcanzar sus metas y objetivos. (6)

En Cuba se han llevado a cabo DO enfocados al clima organizacional en entidades de la producción y los servicios de la provincia de Matanzas. Utilizando para ello la Metodología del PNUD (Programa de las Naciones Unidas para el Desarrollo) para estudios de Clima Organizacional en América Latina. En la provincia de Santiago de Cuba se realizó a la dirección territorial de ETECSA un Diagnóstico a la Cultura organizacional con el fin de mejorar la gestión de los recursos humanos y mejorar la eficiencia Organizacional en términos de desarrollo de sus empleados y de una cultura capaz de soportar el cumplimiento de los objetivos de la empresa, desde una posición de ventaja competitiva (7).

En Ciego de Avila se realizó un DO a la Dirección Provincial de Vivienda para lograr una mejora en la gestión del capital Humano (8). La Unidad Básica Empresarial Combinado Cubanacán aplicó un Diagnostico de las funciones administrativas con el objetivo de mejorar métodos y estilos de dirección. Evidenciando como principales deficiencias la previsión , ya que hasta ahora solo se trabaja sobre la alternativa que ofrece la empresa. Además se hace preciso la descentralización del Combinado Cubanacán de la Empresa, para que se logren mejores resultados en las funciones del ciclo administrativo.

No se tiene conocimiento de la existencia de una herramienta para la gestión de los DO a nivel internacional. La UCI desde el año 2006 viene llevando a cabo DO. Esto ha ocurrido producto de la necesidad de encontrar fortalezas y oportunidades de mejora que revelen el estado actual en la producción de software de la institución. Para su realización se han empleado diferentes herramientas informáticas, entre ellas se encuentran:hojas de cálculo para recopilar el levantamiento de la información, mediante estas, los datos fueron validados y enviados por correo. Además se utilizó SQL para almacenar y organizar la información levantada y generar los reportes mediante elaboración de vistas. También se utilizó el Moodle para llevar a cabo la aplicación de las encuestas sobre los factores potenciales de éxitos y de fracaso. Todas estas actividades se llevaron a cabo siguiendo los siguientes pasos lógicos.

- Elaboración de plantillas en hojas de cálculo para recoger los datos del levantamiento de información.
- Elaboración de la plantilla del libro de diagnóstico para guardar los resultados del diagnóstico.
- Elaboración y validación de encuestas sobre los factores de éxito y fracaso.
- Configuración y montaje de las encuestas en el Moodle.
- Diseño y montaje de la base de datos para el levantamiento de información.
- Creación de consultas y vistas para la obtención de la información.

- Realización de una prueba piloto que incluyó la medición del tiempo de realización y tiempo de obtención de resultados.
- Capacitación de los revisores.
- Preparación, ejecución y resultado de las revisiones a los proyectos.
- Realización de las encuestas sobre los factores de éxito y fracaso a los líderes de proyecto.
- Obtención de estadísticas del levantamiento de información, encuestas y revisiones.
- Publicación del libro de diagnóstico.

1.3 Metodología de desarrollo

Las metodologías ágiles y robustas se pueden aplicar a diferentes proyectos, teniendo en cuenta el flujo de información con que se trabaja en el mismo. Dentro de las robustas encontramos: Rational Unified Process (RUP), Microsoft Solutions Framework (MSF), etc. y dentro de las ágiles tenemos Extreme Programming (XP), SCRUM, Cristal Methodologies, entre otras.

1.3.1 RUP

RUP acrónimo de Rational Unified Process (Proceso Unificado de Rational) es un proceso de ingeniería de software, una forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo) en una organización o equipo de desarrollo de software. Es también una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable.

Se caracteriza por ser iterativo e incremental, centrado en la arquitectura y guiado por los casos de uso. RUP posee un ciclo de vida de desarrollo en espiral. Este divide el proceso de desarrollo del software en ciclos, donde estos son la clave que tiene el modelo para crear un proyecto de buena calidad. El ciclo de vida organiza las tareas en fases e iteraciones.

RUP está compuesto por cuatro fases:

- Inicio (puesta en marcha).
- Elaboración (definición, análisis y diseño)
- Construcción (implementación)
- Transición (fin del proyecto y puesta en producción)

cada una de ellas con sus objetivos, roles y artefactos bien definidos. (9)

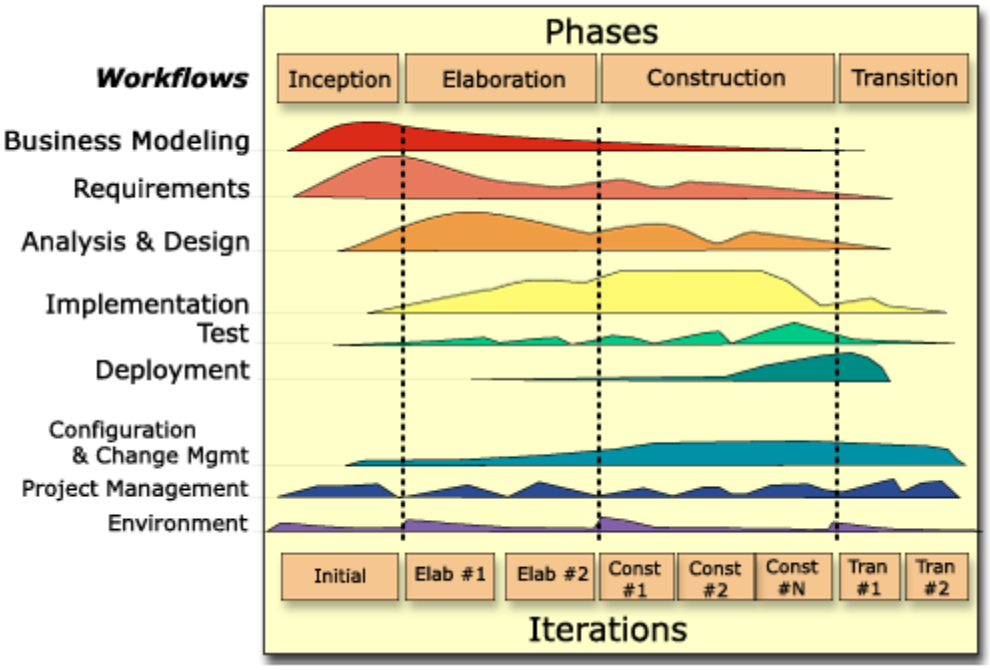


Figura1: Fases y Flujos de RUP

1.4 Lenguaje de modelado

1.4.1 UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por OMG (Object Management Group).

Algunas de las propiedades que posee UML son:

- Concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actuales y futuras.
- Ampliamente utilizado por la industria desde su adopción por OMG.
- Reemplaza a decenas de notaciones empleadas con otros lenguajes.
- Modela estructuras complejas.

- Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos tales como objetos, clases, componentes y nodos.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario.
- Comportamiento del sistema: Casos de uso, diagramas de secuencia y de colaboraciones, que sirven para evaluar el estado de las máquinas.(Rumbaugh, et al.)

UML también es un lenguaje más expresivo, claro y uniforme que los anteriores definidos para el diseño Orientado a Objetos, que no garantiza el éxito de los proyectos pero sí mejora el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios. Sus principales características son:

- Tecnología orientada a objetos.
- El cliente participa en todas las etapas del proyecto.
- Corrección de errores viables en todas las etapas.
- Aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y cliente/servidor.
- Lenguaje unificado para la modelación de sistemas. (10)

1.5 Herramienta CASE

Una herramienta CASE (Computer Aided Software Engineering) es un producto computacional enfocado a apoyar una o más técnicas dentro de un método de desarrollo de software.

De las Herramientas automatizadas para la ingeniería de software se puede decir que:

- Permiten un mayor control de proyectos complejos.
- Permiten reducir costos y retrasos en la liberación de un proyecto.
- Permiten una mayor comunicación en equipos de trabajo.
- Ayudan a determinar la complejidad del proyecto y esfuerzos necesarios.

1.5.1 Visual Paradigm

Visual Paradigm for UML (VP-UML) es una herramienta de diseño UML y herramienta CASE UML diseñada para la ayuda al desarrollo de software. VP-UML soporta los principales estándares de la industria tales como Lenguaje de Modelado Unificado (UML), SysML, BPMN, XMI, etc. Ofrece un conjunto

completo de las herramientas de desarrollo de software, necesarios para la captura de requisitos, la planificación de software, software de planificación, la planificación de controles, el modelado de clases, modelado de datos, etc. Con VP-UML, el equipo de desarrollo de software puede realizar el análisis y diseño de sistemas con eficacia.

Visual Paradigm permite la ingeniería inversa así como la generación de código, puede integrarse con distintos Entornos de Desarrollo Integrados (IDEs) y permite el trabajo en múltiples plataformas. Visual Paradigm posee además una excelente integración con PHP lenguaje que se utilizará para el desarrollo del módulo. (11)

1.6 Lenguaje de programación

Un lenguaje de programación es aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes.

1.6.1 PHP 5 (PHP: Hypertext Pre-processor)

Es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en lado del servidor. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Este lenguaje posee amplias ventajas como:

- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados extensiones). Posee una amplia documentación en su página oficial (www.php.net), entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Contiene una biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones (desde PHP5). (12)

1.7 Entorno de desarrollo Integrado (IDE)

Un IDE (Intregrated Development Environment) es un ambiente de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica de usuario (GUI). Los IDE proveen un marco de desarrollo amigable para la mayoría de los lenguajes de programación.

1.7.1 Net Beans

Net Beans 6.8 es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el Entorno de Desarrollo Integrado (IDE) Net Beans. El IDE Net Beans es un producto libre y gratuito sin restricciones de uso.

Principales características

- Nos provee de una estructura para los proyectos que podemos crear junto a este IDE, nos propone un esqueleto para organizar nuestro código fuente, el editor conjuntamente integra los lenguajes como HTML, Java Script y CSS. Además posee un sistema para examinar todos los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos, para acelerar la programación.
- Integración con framework de PHP como Symfony y Zen Framework. Además brinda un entorno amigable para ejecutar comandos de Symfony.
- El editor de PHP, es mucho más ágil y a la vez robusto, contiene más ayuda en línea, reconocimiento de sintaxis y todo lo que provee la última versión de PHP.
- Es posible crear test con PHP Unit, para diferentes funciones, luego realizar la comprobación y ver todos los resultados. En las propiedades PHP Unit puede definir una configuración personalizada de archivos XML, un archivo de arranque para las opciones de línea de comandos, una serie de pruebas a medida, o puede que el IDE genere el código esqueleto para usted.
- Net Beans integra muy bien la utilización X debug, gracias a esto, podemos inspeccionar y examinar cada variable local, establecer puntos de interrupción y evaluar el código en nuestra lógica.

- El IDE de Net Beans para PHP también ofrece la línea de comandos de depuración: La salida del programa PHP aparece en una pantalla de línea de comandos en el IDE de sí mismo y se puede inspeccionar el código HTML generado sin tener que cambiar a un navegador.
- Integración de sistemas de control de versiones, tales como SVN, CVS, Mercurial y Git.
- Desde el editor es posible realizar la administración de estos sistemas versionados, sus commit, branch, importar, exportar, revert, clonar, etc.

1.8 Framework para el desarrollo

En el desarrollo de software, un Framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un Framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Como función adicional abstraen implementaciones complejas y producen un desarrollo ágil, proveyendo soluciones a problemas cotidianos; se puede ver también como un conjunto de librerías que usan las aplicaciones.

1.8.1 Symfony

Symfony es un Framework PHP que facilita el desarrollo de las aplicaciones web. Symfony se encarga de todos los aspectos comunes y aburridos de las aplicaciones web, dejando que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto.

Symfony aumenta exponencialmente la productividad y te ayuda a mejorar la calidad de las aplicaciones web aplicando todas las *buenas prácticas* y patrones de diseño que se han definido para la web.

Symfony es además el Framework más documentado del mundo, ya que cuenta con miles de páginas de documentación distribuidas en varios libros gratuitos y decenas de tutoriales.

Dentro de las características de Symfony las más destacadas son:

- Fácil de instalar y configurar en sistemas Windows, Mac y Linux.
- Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server).
- Compatible solamente con PHP 5 desde hace años, para asegurar el mayor rendimiento y acceso a las características más avanzadas de PHP.

- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Preparado para aplicaciones empresariales, ya que se puede adaptar con facilidad a las políticas y arquitecturas propias de cada empresa u organización.
- Flexible hasta cualquier límite y extensible mediante un completo mecanismo de plugins.
- Es seguro ya que permite controlar hasta el último acceso a la información e incluye por defecto protección contra ataques XSS y CSRF.
- Probado con éxito durante años en varias aplicaciones gigantescas (Yahoo! Answers, Dailymotion, delicious) y en otros miles de sitios pequeños y medianos.
- Symfony sigue una política de tipo LTS (long term support), por la que las versiones estables se mantienen durante 3 años sin cambios pero con una continua corrección de errores.
- Su código fuente incluye más de 9.000 pruebas unitarias y funcionales.
- Publicado bajo licencia MIT de software libre y apoyado por una empresa comprometida con su desarrollo. (13)

1.9 Servidor Web

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada). Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

1. Espera peticiones en el puerto TCP indicado (el estándar por defecto para HTTP es el 80).
2. Recibe una petición.
3. Busca el recurso.
4. Envía el recurso utilizando la misma conexión por la que recibió petición.
5. Vuelve al segundo punto.

1.9.1 Servidor HTTP Apache

El Apache HTTP Server Project es un esfuerzo de desarrollo de software de colaboración destinadas a crear una implementación de código fuente robusta, de grado comercial, con muchas características, a la libre disposición de un servidor HTTP (web) del servidor.

Las principales características de Apache son:

- Está disponible para una gran multitud de plataformas como GNU/Linux, Mac OS, Mac OS X Server, Netware, Open Step/Match, UNIX, Solaris, Sun OS, Unix Ware, Windows entre otras.
- Permite la autenticación de usuarios en varias formas con el objetivo de restringir el acceso a determinadas páginas de un sitio Web de una forma sencilla y de fácil mantenimiento.
- Posee código abierto.
- Es gratuito
- Es extensible.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor dando la posibilidad de ejecutar un determinado script cuando ocurra un error en concreto.
- Permite la creación de sitios Web dinámicos mediante el uso de CGI's, de Server Side Includes (SSI), de lenguajes de Scripting como PHP, Javascript, Python, Java y páginas jsp. (14)

1.10 Sistema gestor de base de datos

Un Sistema de Gestión de Base de Datos (SGBD) es un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

1.10.1 PostGre SQL 8.4

PostGre SQL es un sistema de gestión de bases de datos objeto-relacional más usados en el mundo. Es multiplataforma, además posee todas las características de los SGDB modernos. Es un gestor de bases de datos de código abierto y gratuito, brinda un control de concurrencia multi-versión (MVCC por sus siglas en inglés) que permite trabajar con grandes volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación.

Es totalmente compatible con ACID (acrónimo de Atomicity, Consistency, Isolation and Durability; en español: Atomicidad, Consistencia, Aislamiento y Durabilidad). Posee una integridad referencial e interfaces nativas para lenguajes como ODBC, JDBC, C, C++, PHP, PERL, TCL, ECPG; PYTHON y RUBY. Funciona en todos los sistemas operativos Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows.

Algunas de las principales áreas que mejora PostGre SQL 8.4 son:

- Funciones de ventanas.
- Expresiones de tablas comunes y consultas recursivas.
- Restauración en paralelo
- Fácil edición de funciones PSQL
- Certificados de soporte SSL para la autenticación de usuarios
- Ajuste automático del espacio libre.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array. (15)

1.11 Conclusiones

En este capítulo se hizo un estudio del estado del arte de los sistemas para la gestión de los DO así como la descripción de los principales conceptos relacionados con el tema para una mayor comprensión de los mismos. Además se realiza una descripción de las herramientas, metodologías y tecnologías a utilizar con el fin de lograr un producto de calidad y que cumpla con los requerimientos impuestos por CALISOFT.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción.

En este capítulo se presentan las reglas generales que se deben cumplir en el negocio, para lo cual se identifican y describen cada uno de los actores y trabajadores, así como los casos de uso del negocio, realizando para cada uno su descripción textual, los Diagramas de Actividad y el Modelo de Objetos. También se analizan los requisitos funcionales y no funcionales de la aplicación a desarrollar modelando la misma en términos de casos de uso del sistema.

El proceso de diagnóstico propuesto para las organizaciones productivas de la Universidad de las Ciencias Informáticas ha sido diseñado por el Grupo de Diagnóstico de CALISOFT (Centro de calidad para soluciones tecnológicas).

Este proceso está guiado por un conjunto de acciones que se llevan a cabo para cumplir las metas del diagnóstico trazadas para un año en específico, las cuales están divididas por etapas. Cada una de estas etapas incluye la ejecución de un conjunto de tareas que requieren la utilización de recursos humanos y materiales. Además se genera un volumen de artefactos que se diferencian según la utilidad dentro del proceso, el tipo de información almacenada, el formato, las reglas de confidencialidad, etc.

Con la creación de una aplicación para la gestión de los DO en la UCI varias de las tares que se realizaban manualmente quedarán automatizadas. A la hora de realizar un nuevo DO se contará con la opción de reutilizar los métodos de los DO anteriores. Se tendrá disponible los datos de todos los proyectos y áreas posibles a realizar un DO. Se podrá definir desde el principio todas la actividades a realizar en el diagnóstico y modificarlas en cualquier momento si es necesario. Así como evaluar a las personas que participaron en el diagnóstico al finalizar este. Además de gestionar tanto los recursos humanos como materiales. Toda esta información será almacenada en un servidor de base de datos. Los usuarios para poder interactuar con la aplicación deberán autenticarse.

2.2 Reglas del Negocio

2.2.1 Reglas Textuales.

- Para almacenar el Expediente de Diagnóstico debe haberse creado la estructura del repositorio y puede ser realizada por cualquier miembro del grupo de Diagnóstico.
- Cada Expediente de Diagnóstico debe estar codificado haciendo referencia al año y a la actividad que se realizó.
- Cada documento generado del diagnóstico debe estar codificado, haciendo referencia al año y al área que se diagnosticó.
- Todos los documentos que se generan en el diagnóstico forman parte del Expediente del Diagnóstico y este debe ser guardado en el repositorio de información definido y puede ser realizada por cualquier miembro del grupo de Diagnóstico.

2.2.2 Reglas del Modelo de Datos

- Las fechas de la planificación de las actividades del proceso de diagnóstico deben ser una fecha válida.
- Los nombres de las Áreas pueden tener números o guiones.
- Los nombres y apellidos de una persona, roles en el proyecto y cargos deben se válidos.

2.2.3 Reglas de Relación

- La fecha de inicio del diagnóstico es enviada por el Jefe de diagnóstico y debe ser confirmada al área a diagnosticar con 7 días de antelación.
- La agenda del diagnóstico es enviada por el Jefe de diagnóstico con 3 días de antelación al área a diagnosticar.
- La elaboración de los indicadores deben tener definidos los criterios y puede ser realizada por cualquier miembro del grupo de Diagnóstico.
- Para la elaboración de las gráficas deben estar definidos los indicadores y puede ser realizada por cualquier miembro del grupo de Diagnóstico.
- La solicitud del personal que conforma el Grupo de Apoyo es enviada por el Jefe de diagnóstico y será al menos con 10 días previos a la fecha de inicio del diagnóstico.

- La evaluación de desempeño del grupo de apoyo es enviada por el Jefe de diagnóstico y se le envía a los directivos de las áreas correspondientes.
- El Jefe de área tiene 3 días para reprobar la solicitud de algún miembro del Grupo de Apoyo

2.3 Modelo del negocio

2.3.1 Actores del Negocio

Actores del Negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externa con los que el negocio interactúa. Lo que se modela como actor es el papel que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados.

Actor	Descripción
Cliente	Representa a aquella entidad o persona que solicita información a obtener en el proceso de diagnóstico, orienta la elaboración del mismo y toma decisiones a nivel gerencial.
Directivo	Es aquella entidad o persona que está al frente de la entidad a la que se le realiza el diagnóstico. A ellos les pertenecen los recursos materiales que se les solicitan como un grupo de apoyo a la realización del diagnóstico.

Tabla 1: Actores del negocio

2.3.2 Modelo de casos de uso del Negocio

El modelo de Casos de Uso del Negocio, es un modelo que describe los procesos de un negocio (casos de uso del negocio) y su interacción con elementos externos (actores). Describe las funciones que el

negocio pretende realizar y su objetivo básico es describir cómo el negocio es utilizado por sus clientes y socios.

Diagrama de Casos de Uso del Negocio

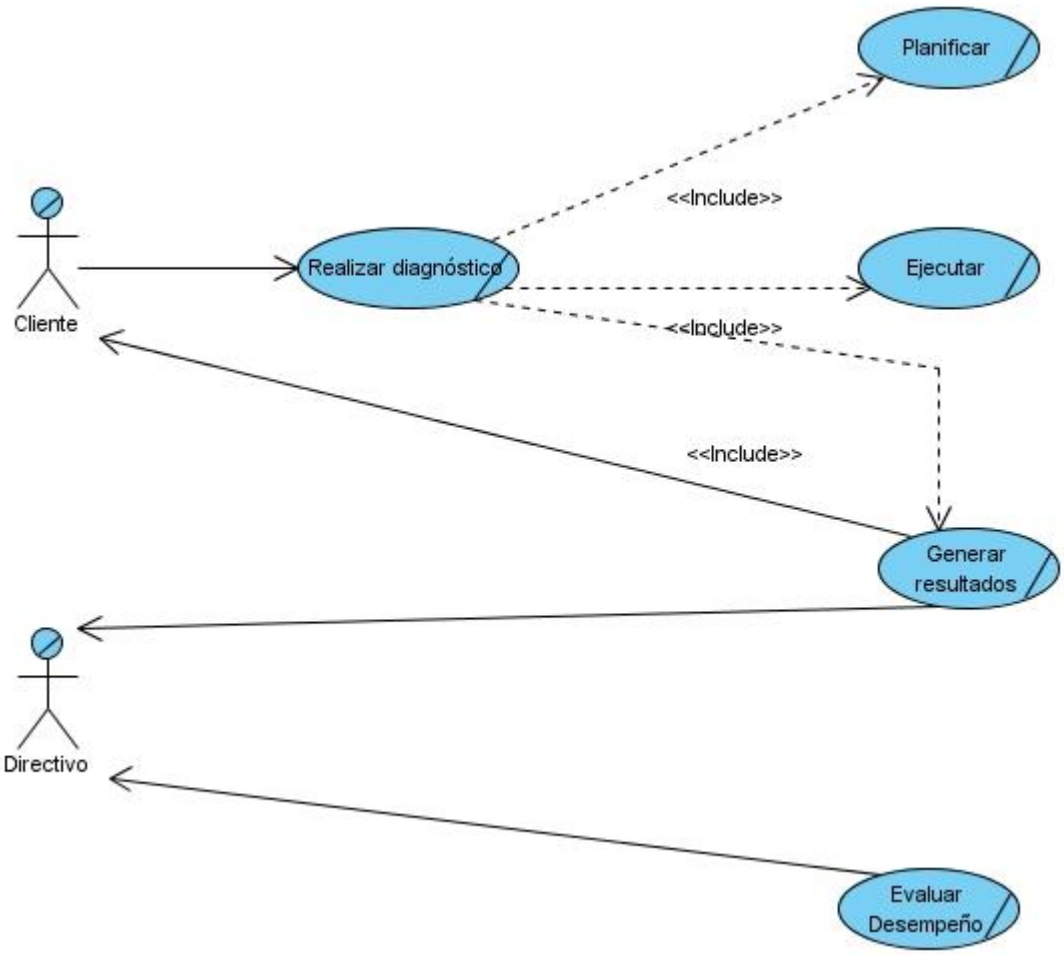


Figura 2: Diagrama Casos de Uso del Negocio

2.5 Modelo de objeto del negocio.

Diagrama de Clases del Modelo de Objeto del Negocio

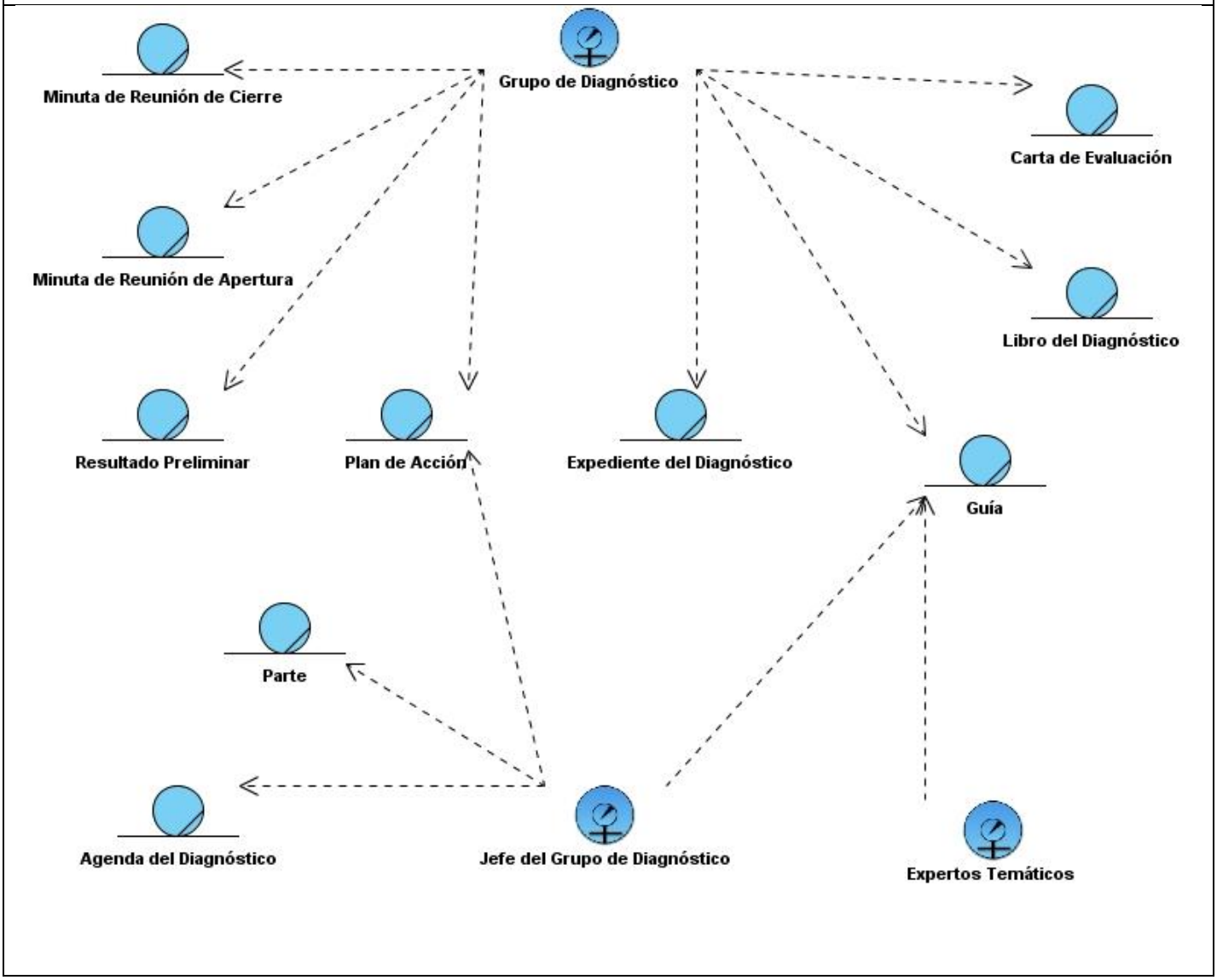


Figura 3: Diagrama de Clases del Modelo de Objeto del Negocio

2.6 Diagramas de actividades

Los diagramas de actividades explican las actividades que ocurren dentro del proceso del negocio, separando las actividades que realizan los actores del negocio de las actividades que realizan los trabajadores del negocio. (Ver Anexo 2).

2.7 Especificación de los requisitos de software

Los requerimientos son las condiciones o capacidades que tiene que ser alcanzada por un sistema para satisfacer al cliente o usuario final.

Los requerimientos se clasifican en dos tipos: requerimientos funcionales y requerimientos no funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir mientras que los requerimientos no funcionales son características o propiedades que el sistema debe tener para ser un producto con mucha aceptación.

2.7.1Requisitos Funcionales

RF-1	Gestionar Diagnóstico
RF-2	Gestionar Recursos Materiales.
RF-3	Gestionar Actividad
RF-4	Gestionar Método.
RF-5	Gestionar Proyecto.
RF-6	Gestionar Área.
RF-7	Gestionar Evaluación de desempeño.
RF-8	Gestionar Usuario
RF -9	Gestionar Rol
RF -10	Autenticar Usuario

Tabla 2:Requisitos Funcionales

El listado extendido de los requerimientos funcionales se muestra en los anexos. (Ver Anexo 3)

2.7.2 Definición de los requerimientos no funcionales

Usabilidad

RnF 1. El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.

RnF 2. El software tendrá siempre visible la opción de Ayuda, lo que posibilitará un mejor aprovechamiento por parte de los usuarios de sus funcionalidades.

Fiabilidad

RnF 3. La herramienta de implementación a utilizar debe tener soporte para recuperación ante fallos y errores. La información manejada por el sistema estará protegida de acceso no autorizado y divulgación.

RnF 4. El tiempo medio de reparación, en caso de un fallo es de 7 días.

Eficiencia

RnF 5. El tiempo de respuesta estará dado por la cantidad de información a procesar, entre mayor cantidad de información mayor será el tiempo de procesamiento.

RnF 6. Al igual que el tiempo de respuesta, la velocidad de procesamiento de la información, la actualización y la recuperación dependerán de la cantidad de información que tenga que procesar la aplicación.

Soporte

RnF 7. La aplicación recibirá mantenimiento en el período de tiempo determinado por el equipo de desarrollo y los clientes.

RnF 8. Para garantizar el soporte de esta herramienta, se documentará todo el código de la aplicación, así como la posibilidad de emitir sus quejas y sugerencias a los desarrolladores de la herramienta, para poder realizar mantenimiento al sistema y darle solución a cualquier problema que surja con la aplicación.

Restricciones de diseño

RnF 9.El diseño deberá ser coherente con la aplicación GeSoft que pertenece al grupo de Auditorías y Revisiones para futuras integraciones.

RnF 10. Diseño sencillo, con pocas entradas, donde no sea necesario mucho entrenamiento para utilizar el sistema.

RnF 11. El producto de software final debe diseñarse sobre una arquitectura cliente-servidor.

RnF 12. Se deben emplear los estándares establecidos (diseño de interfaces, base de datos y codificación).

RnF 13. Se debe lograr un producto altamente configurable y extensible, teniendo en cuenta que se desarrollará sobre el framework Symfony y framework de presentación Ext JS.

Interfaz

Interfaces de usuario

RnF 15. El sistema debe tener una apariencia profesional y un diseño gráfico sencillo

Interfaces Hardware

RnF 16. Por la parte cliente se requiere de una PC de 128 MB de RAM. A su vez el servidor debe contar con 1GBde RAM y 80 GB de disco duro que se encargue del Servidor Web y el de BD. Todas las computadoras implicadas tanto del lado de los servidores como de los usuarios, deben estar conectadas a una red y tener al menos 100 MB de velocidad.

Interfaces Software

RnF 17. En la computadora de los usuarios sólo se requiere de navegador para Internet o Intranet, bajo cualquier sistema operativo; en el Servidor Web, sistema operativo Windows Server 2003 en adelante, y en la Base de Datos, Windows Server 2003 en adelante. Para su implementación se usará como herramienta de desarrollo Symfony y Gestor de Base de Datos PostgreSQL 8.3.5

Requisitos Legales, de Derecho de Autor y otros.

RnF 18. El sistema debe ajustarse y regirse por la ley, decretos leyes, decretos, resoluciones y manuales (órdenes) establecidos, que norman los procesos que serán automatizados.

Estándares Aplicables

El sistema se desarrollará guiado por los procesos definidos en el programa de mejora que se lleva a cabo en la universidad para alcanzar el nivel 2 de CMMI.

2.8 Modelo de Casos de Uso del Sistema.

El diagrama de casos de uso del sistema muestra la relación existente entre los procesos o casos de uso del sistema y los actores del sistema. Estos casos de uso son resultado de la captura de requisitos funcionales que son los que indican las funcionalidades que debe cumplir el sistema.

Diagrama de Casos de Uso del Sistema

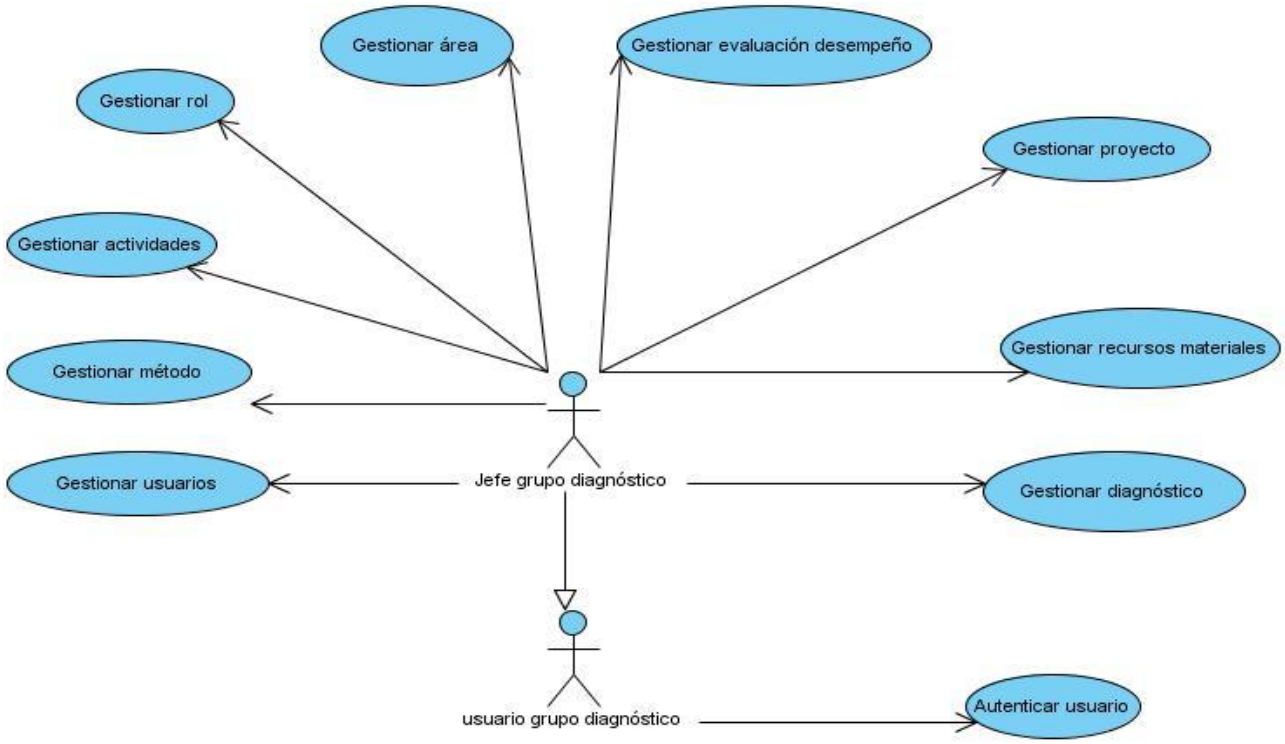


Figura 4:Diagrama de Caso de Uso del Sistema

2.8.1 Definición de los actores del Sistema

Actores	Objetivo
Jefe grupo diagnóstico	Es la persona encargada de crear y gestionar el diagnóstico. Gestionar los usuarios que participarán en el diagnóstico. Así como gestionar los recursos materiales y actividades.

Tabla 3: Definición de los actores del Sistema

2.9Descripción textual de los Casos de Uso del Sistema.

La descripción de los casos de uso del sistema, describen de forma detallada la interacción de los actores del sistema con el sistema, (Ver Anexo 4)

2.10 Conclusiones

En este capítulo se reflejaron las características que definen el negocio y se describieron los procesos del mismo, también se definieron las actividades que serían objeto de automatización las cuales conforman los requisitos funcionales que debería tener el sistema para darle solución al problema planteado. También se delimitaron los requisitos no funcionales que el sistema debería cumplir para lograr satisfacer al cliente. Además, se definió qué debería hacer la aplicación para cumplir lo propuesto en los requisitos funcionales a través de la descripción de los casos de uso.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se convierten los requisitos a una especificación que describe cómo implementar el sistema, elaborando así, un diseño que cumpla con todos los nuevos requisitos establecidos. Se muestran los diagramas de clases del diseño de los casos de uso y diagramas de interacción. Así como el diseño de la base de datos, después de haberle incluido las nuevas tablas y campos correspondientes, dentro de éste, el modelo lógico y el modelo físico.

No se realizará el análisis del sistema debido a que RUP propone que, para realizar el proceso de implementación de un software no necesariamente se deben llevar a cabo los procesos del análisis, pues con solo hacer el diseño y tenerlo bien definido, se están analizando los diferentes procesos que se van a informatizar.

3.2 Diseño

3.2.2 Diagrama de clases del diseño

Los diagramas de clase del diseño representan a las clases de diseño y las relaciones entre ellas. Las clases del diseño contienen atributos y métodos. En los distintos diagramas de clases del diseño se representa el patrón de arquitectura Modelo Vista Controlador (MVC).

Diagrama de clases del diseño, Gestionar diagnóstico

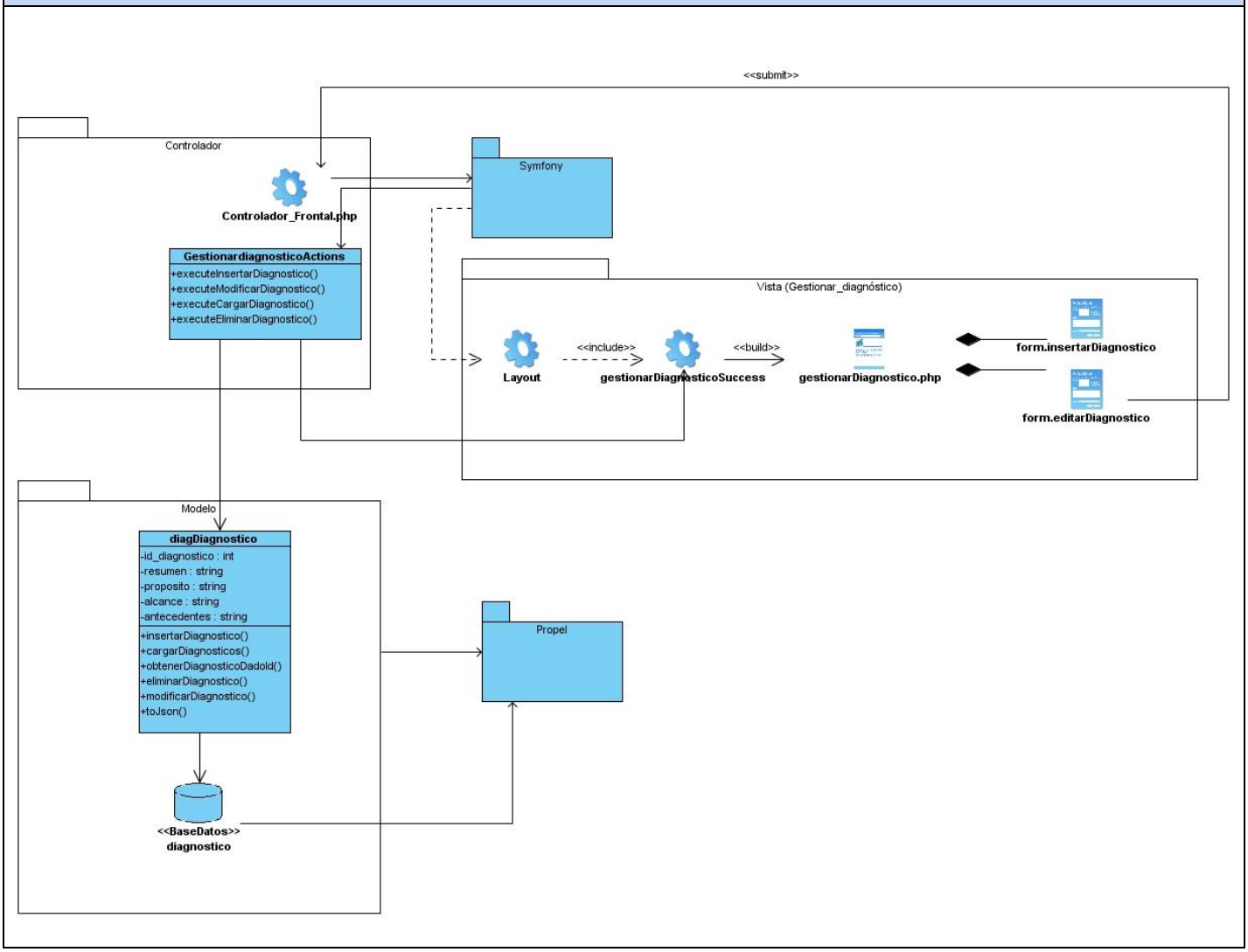


Figura 5: Diagrama de clases del diseño, Gestionar diagnóstico

Diagrama de clases del diseño, Gestionar área

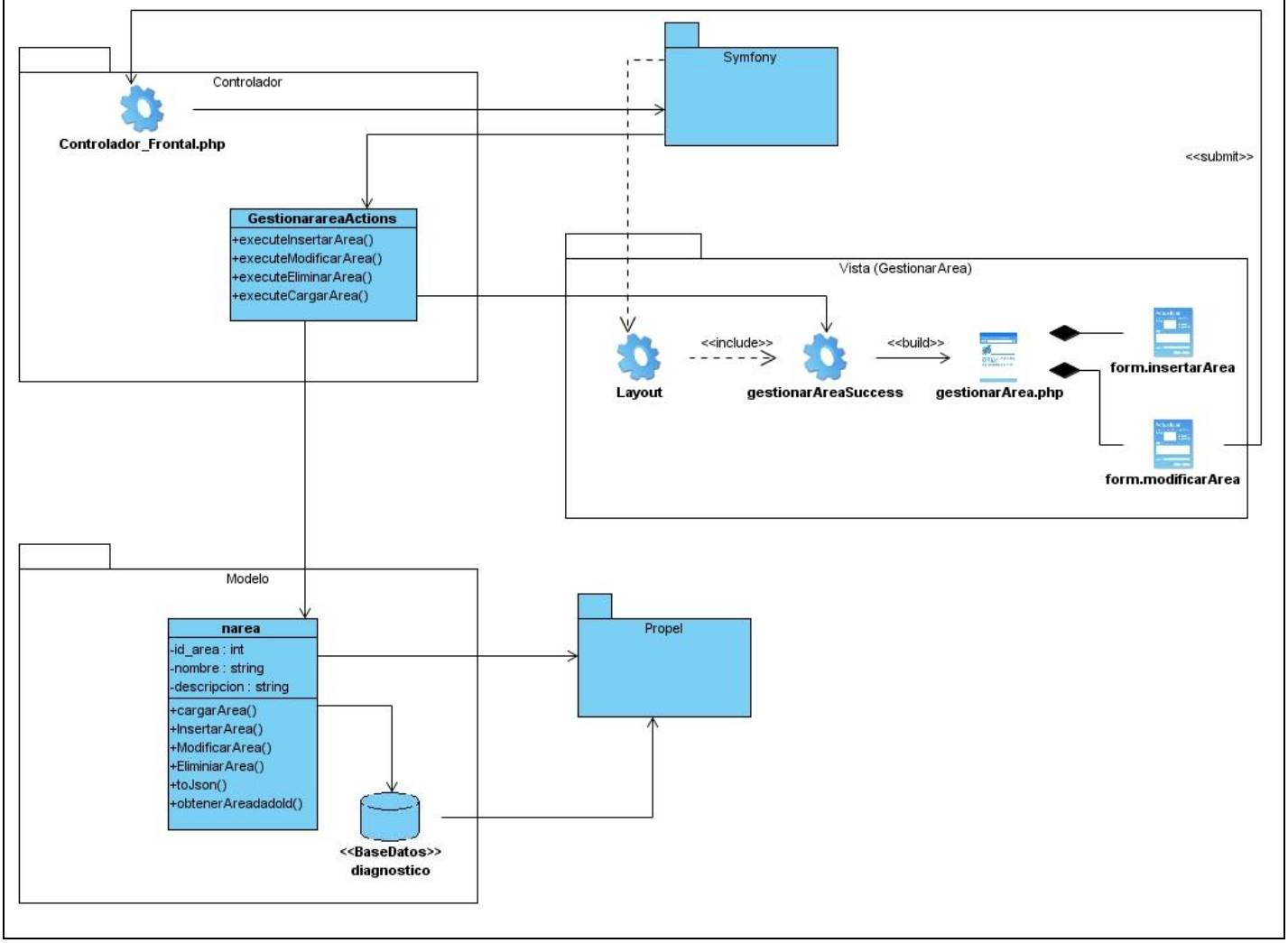


Figura 6: Diagrama de clases del diseño, Gestionar área

(Ver anexo 5)

3.3 Modelo de Datos

1.3.1 Modelo Lógico de Datos

Diagrama de clases persistentes

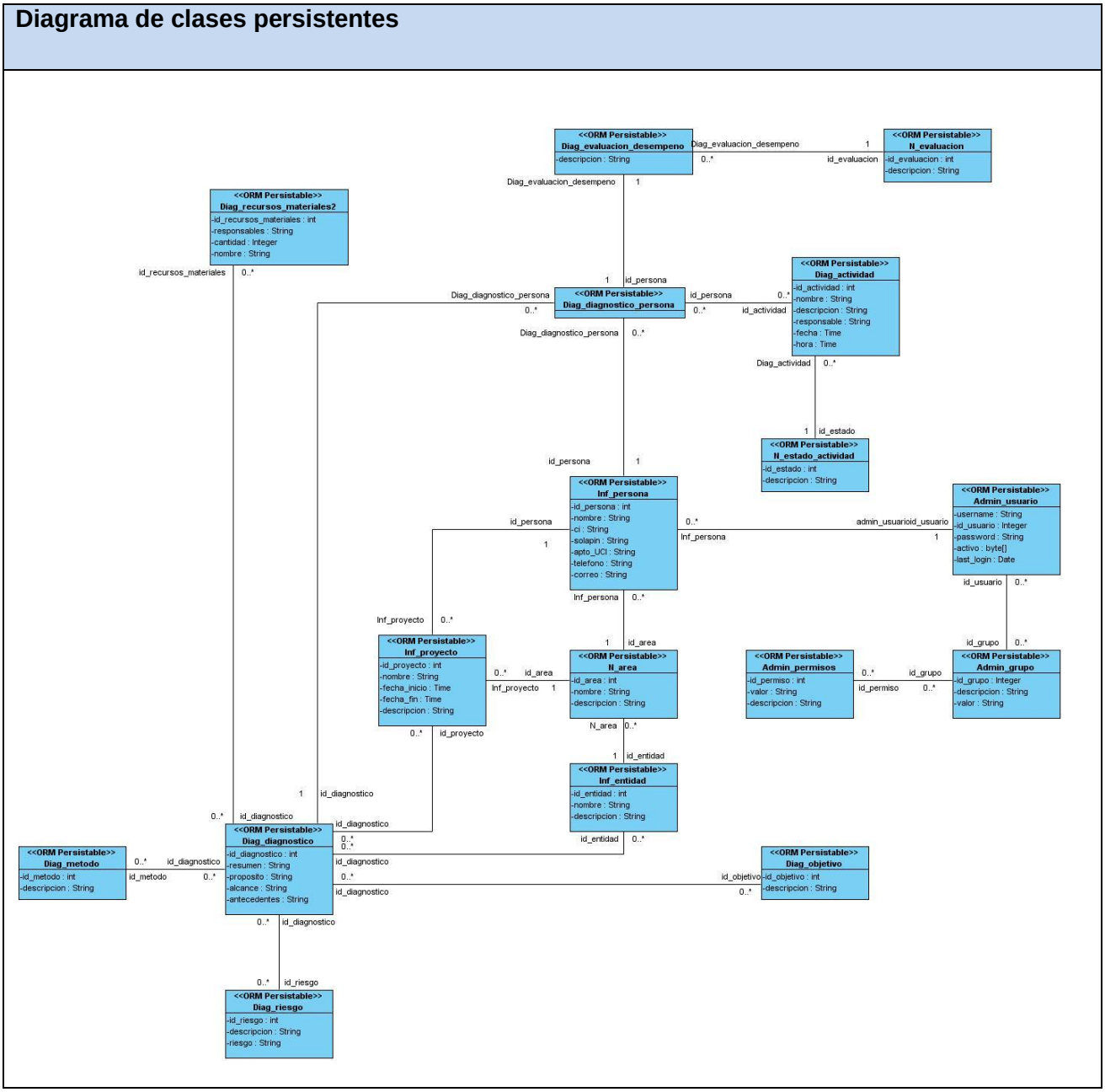


Figura 7: Modelo Lógico de Datos

1.3.1 Modelo Físico de Datos

Diagrama Entidad – Relación

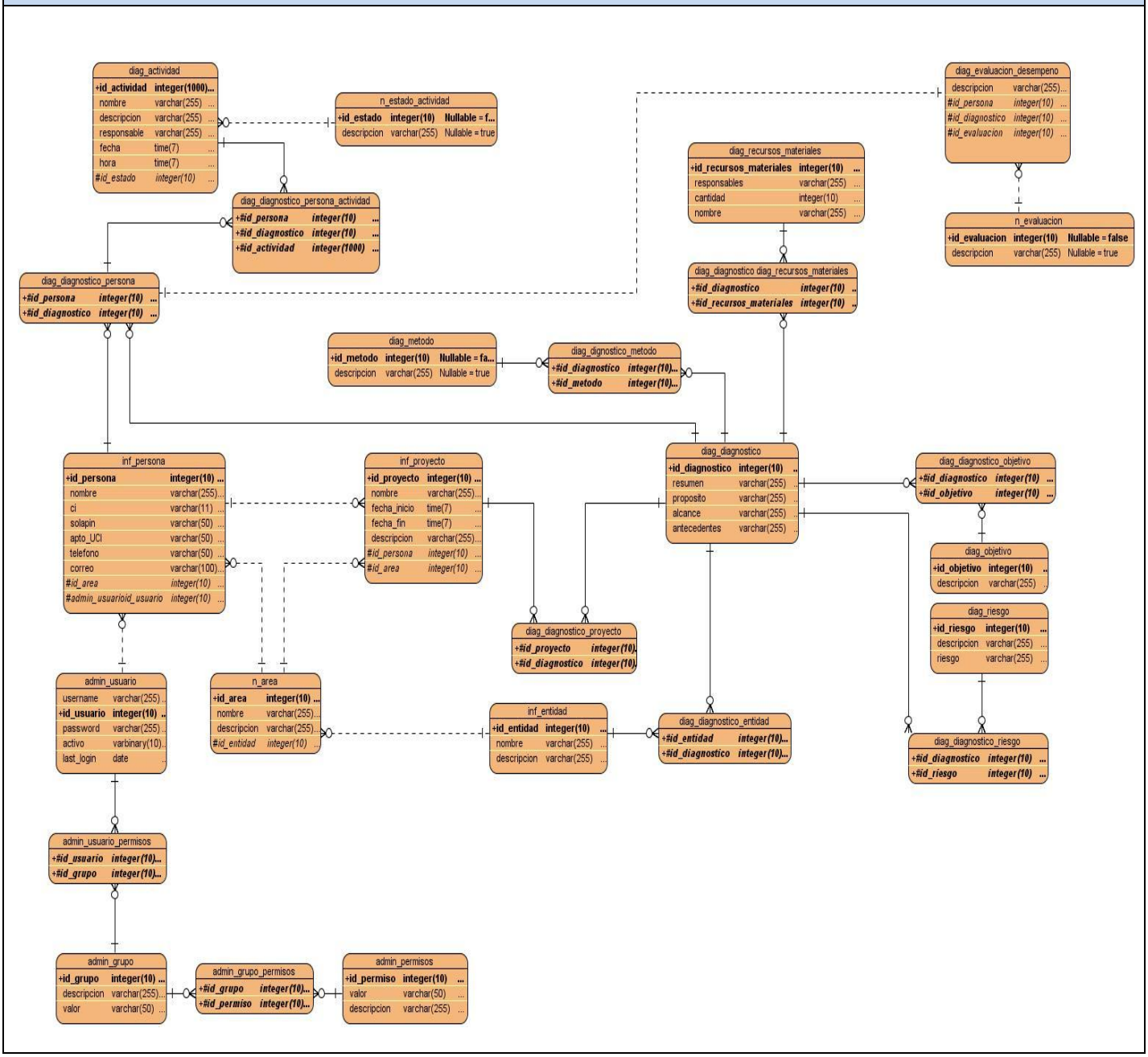


Figura 8: Modelo Físico de Datos

3.4 Arquitectura del Sistema

Una arquitectura no es más que un esqueleto o base de una aplicación. Es un diseño que muestra los bloques de construcción de una aplicación (o algún otro sistema de software). Indica la estructura, funcionamiento e interacción entre las partes del software.

Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático.

La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, programadores, y otros miembros del equipo trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

3.4.1 Estilo arquitectónico Modelo Vista Controlador

La arquitectura del patrón Modelo-Vista-Controlador (MVC) es un paradigma de programación bien conocido para el desarrollo de aplicaciones con interfaz gráfica (GUI). El propósito del MVC es aislar los cambios. Es una arquitectura preparada para los cambios, que desacopla datos y lógica de negocio de la lógica de presentación, permitiendo la actualización y desarrollo independiente de cada uno de los componentes. La figura 1.2 muestra su funcionamiento.

Las aplicaciones MVC se dividen en tres grandes áreas funcionales:

- **Vista:** es la presentación de los datos. Transforma el modelo en páginas web que permiten al usuario interactuar con él. Las páginas están compuestas por código HTML y código que provee datos dinámicos a ella como PHP.
- **Controlador:** atenderá las peticiones y componentes para la toma de decisiones de la aplicación capturando los eventos de entrada desde la vista.
- **Modelo:** es la lógica del negocio o servicio y los datos asociados con la aplicación.

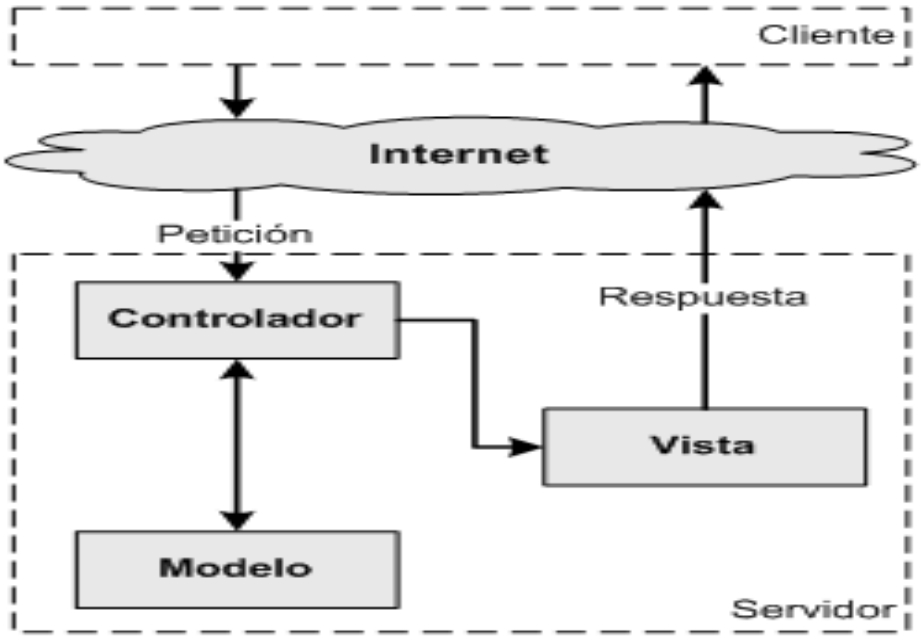


Figura 9: Patrón arquitectónico MVC.

3.4.2 La implementación del MVC que realiza Symfony

El ***modelo*** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. Típicamente las clases del modelo contendrán funciones para recuperar, insertar y actualizar información en la BD. Symfony con el ORM *(*object-relationalmapping*) o* "mapeo de objetos a bases de datos"*)*brinda abstracción a la hora de acceder a los datos, ya que se manejan como objetos. En el caso de la ***vista***, esta transforma el modelo en una página web que permite al usuario interactuar con ella. Por su lado el ***controlador*** es encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

El controlador frontal y el layout son comunes para todas las acciones de la aplicación, se pueden tener varios controladores y varios layouts, con la posibilidad de cambiarlos mediante la configuración del framework,esto es más utilizado con los layout, ya que muchas veces se hace necesario tener varios diseños de interfaces, pero solamente es obligatorio tener uno. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática.

Las clases de la capa del modelo también se generan automáticamente, en función de la estructura de datos de la aplicación. La librería Propel se encarga de esta generación automática, ya que crea el esqueleto o estructura básica de las clases y genera automáticamente el código necesario, sólo es necesario crear las consultas manejando objetos y el framework se encarga del acceso a la BD,este acceso es completamente invisible al programador, porque lo realiza un componente específico llamado Creole. Así, si se cambia el SGBD en cualquier momento, no se debe reescribir ni una línea de código, ya que tan sólo es necesario modificar un parámetro en un archivo de configuración.
La lógica de la vista se puede transformar en un archivo de configuración sencillo, sin necesidad de programarla.

3.5 Patrones de Diseño

Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles.

Los diseñadores expertos en orientación a objetos (y también otros diseñadores de software) van formando un amplio repertorio de principios generales y de expresiones que los guían al crear software. A unos y a otras podemos asignarles el nombre de patrones, si se codifican en un formato estructurado que describe el problema y su solución, y si se les asigna un nombre.

En la terminología de objetos, el patrón es una descripción de un problema y su soluci6n que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Muchos patrones ofrecen orientación sobre c6mo asignar las responsabilidades a 10s objetos ante determinada categoría de problemas.

Un patrón debe estar compuesto por:

- Nombre del patrón.
- Solución.
- Problema que resuelve.

3.5.1 Patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades)

Los patrones GRASP representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para General ResponsibilityAssignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades). A continuación se describen los patrones básicos de asignación de responsabilidades utilizados en este trabajo:

- **Experto:** Este es uno de los más utilizados, puesto que Propel es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.
- **Creador:** Este patrón es el responsable de asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A.
- **Alta Cohesión:** En cada clase Actions se definen las acciones para las plantillas, además, estas colaboran con otras para realizar diferentes operaciones, se instancian objetos, se acceden a las *properties*, es decir, en una Actions se utilizan diferentes funcionalidades estrechamente relacionadas entre sí, lo que proporciona un software flexible ante los cambios.
- **Bajo Acoplamiento:** Este patrón es el encargado de asignar una responsabilidad para conservar bajo acoplamiento.
- **Controlador:** Cada método perteneciente a una vista en las clases Actions tiene la responsabilidad de crear instancias de las clases mapeadas del modelo, a las cuales necesita acceder por que tiene datos que requiere su constructor o necesita acceder a sus datos. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

3.5.2 Patrones Gof

Los patrones GoF (Gang of Four, en español Pandilla de los Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento:

- **Creacionales:** Los patrones creacionales abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados.

- **Estructurales:** Los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.
- **Comportamiento:** Los patrones de comportamiento están relacionadas con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos.

Para la implementación del sistema se utiliza el framework Symfony, el mismo utiliza algunos patrones GoF:

En la categoría de patrones creacionales utiliza:

- Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En las acciones se usan los métodos getRequest(), getUser(), esto se debe a que, en la acción, el método getContext(), guarda una referencia a todos los objetos del núcleo de Symfony, estos métodos pueden ser accedidos desde la vista y desde el controlador, sólo varía la forma de llamarlos.

En la categoría de patrones estructurales utiliza:

- Decorator (Envoltorio): Añade funcionalidad a una clase, dinámicamente. En cada archivo layout.php se define el código HTML común de cada vista, evitando que este sea repetido en cada página.

3.6 Conclusiones

En este capítulo se mostraron los diferentes diagramas del diseño del sistema, quedando conformada la arquitectura del mismo usando el patrón MVC, la cual se manifiesta en los distintos diagramas de clases del diseño, además se muestran algunos diagramas de secuencia, los modelos de datos, tanto físico como lógico, en los que se reflejan las relaciones y las distintas tablas de la base de datos que conforma el sistema.

CAPÍTULO 4: IMPLEMENTACIÓN

4.1 Introducción.

En el presente capítulo se muestran los estilos utilizados en la programación, el diagrama de despliegue elaborado para el sistema, se muestran de forma estructurada las conexiones que se establecen entre los servidores web y de base de datos y sus respectivos protocolos a través de los cuales están conectados. También se muestran los diagramas de componentes, todos realizados siguiendo una arquitectura modelo-vista-controlador.

4.2 Estándares de codificación.

El estilo de programación se refiere a la forma en que se da formato al código fuente, esto involucra la forma en que se ubican las llaves, se indenta el código y se utilizan los paréntesis. Cada programador tiene su propio estilo para escribir. Un buen estilo para programar deberá tener una estructura de código fácil de entender, no solo para otras personas sino también para sí mismo. Existen diversos criterios para un buen estilo, dentro de los que se pueden mencionar:

- Identificadores.
- Indentación.
- Llaves
- Comentarios.

4.2.1 Identificadores*:*

En el caso de los identificadores existen estilos definidos mundialmente como el lower Camel Case y el Upper Camel Case. Cada palabra interna en identificadores compuestos comienza con mayúsculas para ambos estilos, además ocurre, que no se colocan caracteres de separación entre las palabras que conforman un identificador compuesto en ninguno de los dos casos. Para el primero, el identificador comienza con minúscula y para el segundo, el identificador comienza con mayúscula.

- Espacio de nombre: para los espacios de nombres se escogió el Upper Camel Case.
- Clase: para las clases se escogió el lower Camel Case.

Class adminAction sextends sfActions

- Función: para las funciones se escogió el lower Camel Case.

public static function obtenerPersonaDadoId()

- Variable

protected $nombre_usuario;

4.2.2 Indentación y espacio apropiado en el código

La Indentación es usada para tener una mejor visibilidad en el diseño de un programa. La Indentación muestra las líneas que están subordinadas a otras líneas. Por ejemplo, todas las líneas que forman el cuerpo de un ciclo deberán estar indentadas con la instrucción principal del ciclo.

Se definió que la indentación se hará agregando dos <tab> al inicio de la línea que se desee escribir. Se escribirá sólo una sentencia por línea de código y en el caso de cortar las líneas, se hará luego de una coma o antes de un operador. La sección de la derecha de la línea que se corte se ubicará en la línea siguiente indentada al nivel de la expresión correspondiente en la línea superior.

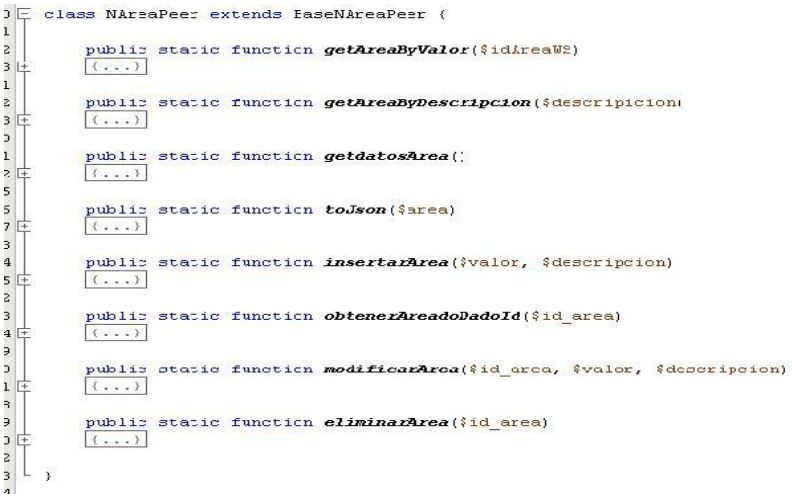


Figura 10: Ejemplo de identación.

4.2.3 Llaves

Existe diversidad de criterios en cuanto a la ubicación de las llaves que delimitan el cuerpo de los bloques de código en los lenguajes que contienen este tipo de estructuras. Algunos programadores prefieren hacerlo ubicando la llave de apertura inmediatamente detrás de la línea cabecera del bloque, mientras otros apuestan por ubicarlas de forma solitaria en la línea siguiente a la línea cabecera. Para este último estilo existen además diferencias en cuanto al nivel de indentación de las mismas. Algunos lo hacen al nivel de la línea cabecera y otros al nivel de las líneas del cuerpo del bloque. Para este trabajo: Las llaves de apertura de las funciones se colocarán inmediatamente detrás de la línea cabecera de la función al igual que las llaves de aperturas de las estructuras *for, if, while, else*. Las llaves de cierre se colocarán solitarias en la línea que sigue a la última línea dentro del bloque e indentadas al nivel de la línea cabecera del bloque.

```
public static function toJson($proyecto) {
    $json = array();
    $datosproyecto = array();
    $json["count"] = count($proyecto);

    if (is_array($proyecto)) {
        foreach ($proyecto as $index => $obj) {
            $datosproyecto[] = array("id_proyecto" => $obj->getIdProyecto(),
                "id_proyecto" => $obj->getIdProyecto(),
                "id_persona" => $obj->getIdPersona(),
                "id_area" => $obj->getIdArea(),
                "id_estado_proyecto" => $obj->getIdEstadoProyecto(),
                "nombre" => $obj->getNombre(),
                "fecha_inicio" => $obj->getFechaInicio(),
                "fecha_fin" => $obj->getFechaFin(),
                "descripcion" => $obj->getDescripcion());
        }
    }

    $json["proyecto"] = $datosproyecto;
    return json_encode($json);
}
```

Figura 11: Ejemplo de uso de llaves.

2.2.4 Comentarios

El uso de comentarios durante la codificación ha demostrado que es beneficiosa por varias razones:

- Ayuda al programador a entender cada elemento o sección de código.

- Hace más fácil el proceso de adaptación del código durante su reutilización.
- Sirve de guía en los casos en que varios programadores trabajen sobre las mismas secciones del código.
- Disminuye el esfuerzo de análisis ya que el lenguaje natural es más legible que cualquier lenguaje de programación.

Todo esto se aprecia claramente cuando se escribe gran cantidad de código en largos intervalos de tiempo donde generalmente el o los programadores olvidan lo que se pensó en un momento. Los comentarios se pueden utilizar para varios fines:

- Para explicar el propósito de las funciones.
- Para explicar las características fundamentales de las clases.
- Para sintetizar las acciones de los algoritmos complejos.
- Para aclarar los datos que representan las variables.
- Para dividir secciones de código en dependencia de los diferentes contextos y funciones.
- A veces se usan comentarios temporales para recordar cosas que faltan, cosas que se deben modificar o analizar en otro momento.

Es necesario tener en cuenta algunos detalles al escribir comentarios:

- La capacidad de síntesis.
- El uso de lenguaje técnico.
- No repetir exactamente paso por paso lo que hace el algoritmo sino expresar un resumen de su propósito.
- Usar un estilo uniforme de comentario definido en estándares para todo el equipo.

En este trabajo se decidió colocar los comentarios encima de la línea a la que se le quiera aplicar y encima de la línea cabecera de los bloques. La indentación se hará al nivel de la línea en cuestión. Se utilizarán en funciones y clases. Se puede utilizar en algoritmos no triviales y secciones de diferentes contextos dentro de los métodos.

4.3 Diagrama de despliegue

El modelo de despliegue, es usado para modelar lo topología del hardware sobre el que se ejecuta el sistema y su distribución física, los distintos dispositivos y procesadores se representan a través de nodos. En este caso el sistema debe disponer de computadoras clientes, un servidor para aplicaciones Web y un servidor para Base de Datos.

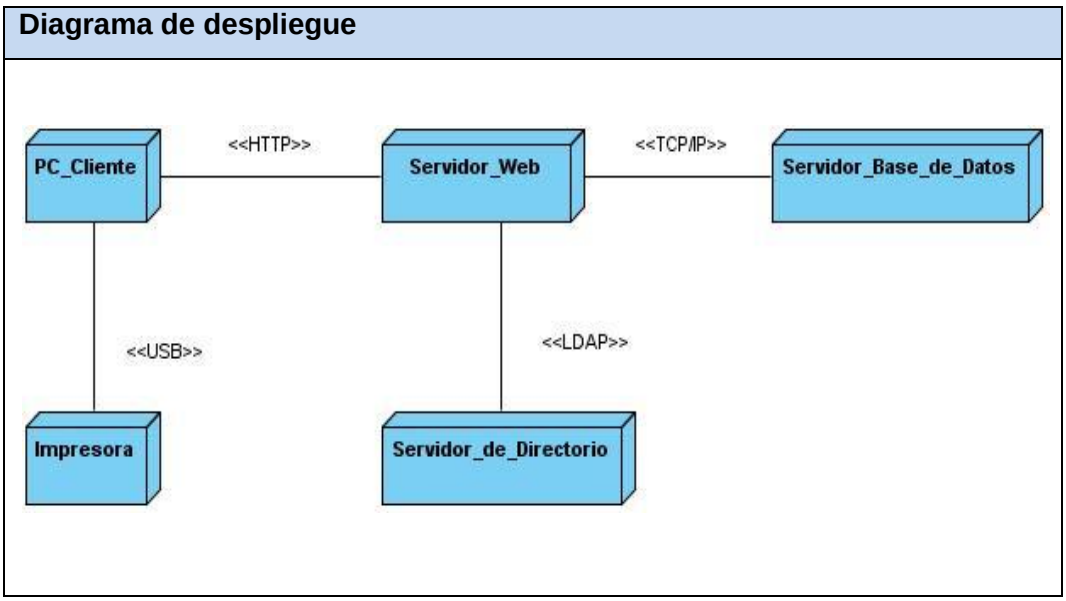


Figura 12: Diagrama de despliegue

4.4 Diagrama de componentes

El diagrama de componentes se usa para representar a los elementos físicos o componentes y sus relaciones de dependencias. Ellos incluyen código fuente y ejecutables. Los componentes representan todo tipo de elementos de software como por ejemplo: las librerías, archivos, bibliotecas cargadas dinámicamente.

Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.

A continuación se muestran el diagrama de componente de acceso a datos, código fuente y componentes web.

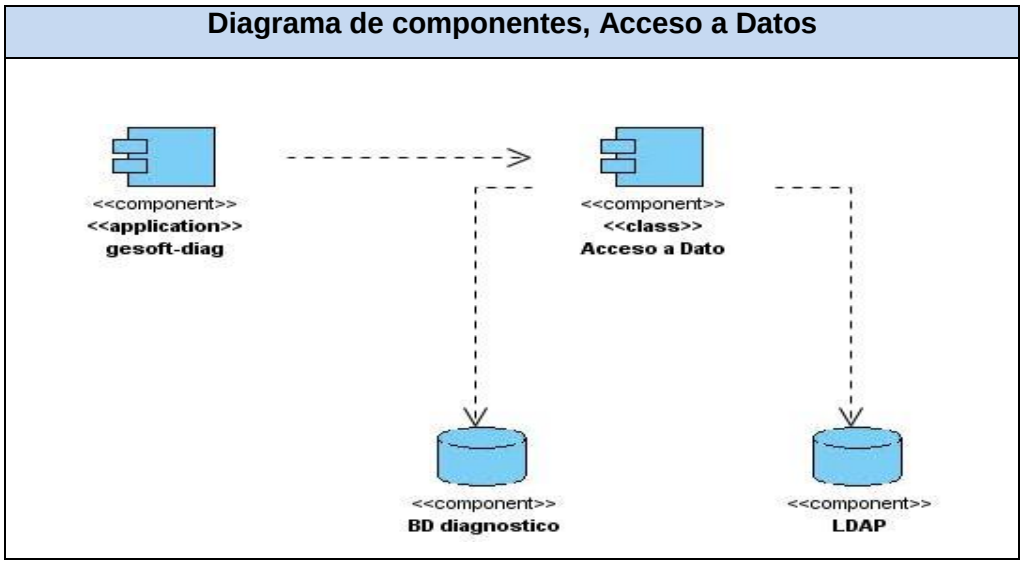


Figura 13: Diagrama de componentes, Acceso a Datos

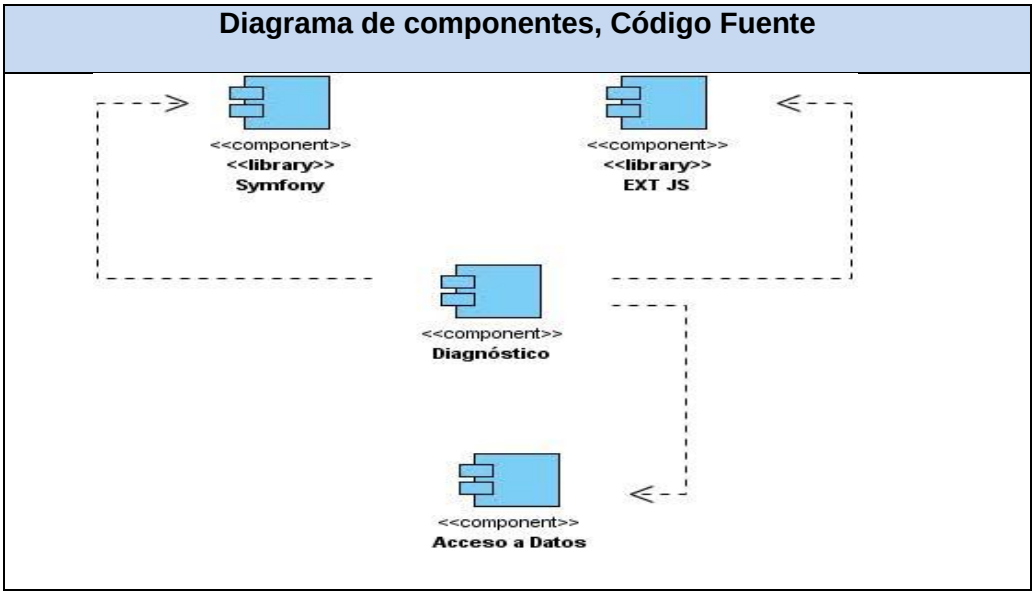


Figura 14: Diagrama de componentes, Código fuente

4.4 Pruebas

El desarrollo de un software es algo complejo y son innumerables las posibilidades de errores, por lo que este ha de ir acompañado de alguna actividad que garantice la calidad; las pruebas son un elemento crítico para la garantía de calidad del software. Es por eso que se utilizan técnicas como las pruebas unitarias que no son más que una forma de verificar el correcto funcionamiento del código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

Las pruebas y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo. Es fundamental medir la cobertura de las mismas, es decir, la determinación de cuando se han realizado las suficientes pruebas. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar.

4.5 Pruebas de Software

Objetivos

El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Esto implica:

- Verificar la interacción de componentes.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

4.6 Pruebas de unidad

Las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las pruebas de integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas:

Fomentan el cambio: facilitan que el programador cambie el código para mejorar su estructura, puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.

Simplifica la integración: permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente.

Documenta el código: Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.

Separación de la interfaz y la implementación: Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro.

Los errores están más acotados y son más fáciles de localizar: dado que tenemos pruebas unitarias que pueden desenmascararlos.

4.7 Pruebas de caja blanca

Las pruebas de caja blanca se realizan sobre las funciones internas de un módulo en concreto, están dirigidas a las funciones internas. Entre las técnicas usadas se encuentran; la cobertura de caminos, pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos, comprobación de bucles. En la siguiente tabla se representa las pruebas de Caja Blanca.

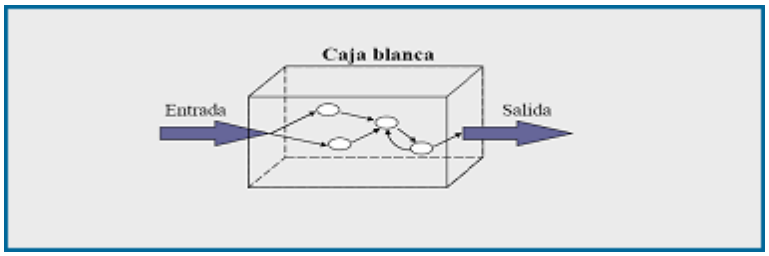


Figura 15: Representación de pruebas de Caja Blanca.

Tipos de prueba de caja blanca:

Prueba de Condición:

Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

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.

Prueba de Bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

Prueba del Camino Básico: Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

La técnica del camino básico permite obtener una medida de la complejidad lógica del código de cada método, programa o módulo dado. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes que existen en la codificación por los cuales puede circular el flujo de control. Es además una de las más eficientes en cuanto a cobertura de código, pues logra que se ejecuten todos los bucles en sus límites operacionales. (16)

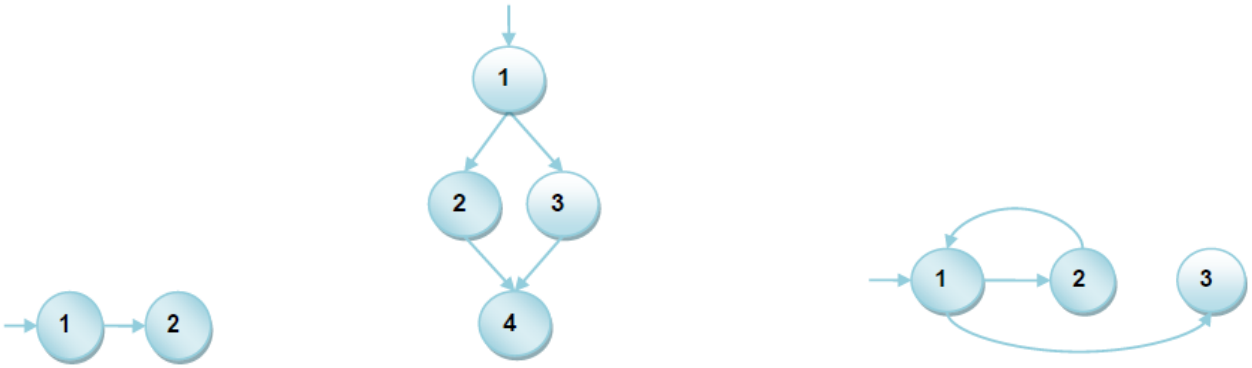
Los pasos que se siguen para aplicar esta técnica son:

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

Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un grafo de flujo en el cual:

- Cada nodo del grafo corresponde a una o más sentencias de código fuente.
- Todo segmento de código de cualquier programa se puede traducir a un grafo de flujo.
- Se calcula la complejidad ciclomática del grafo.

Para construir el grafo se debe tener en cuenta la notación para las instrucciones.



Secuencia **Instrucción IF** **Instrucción While**

Figura 16: Notación de grafos de flujo para las instrucciones: Secuenciales, If y While.

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, su comprensión y brindan información para confirmar que el trabajo se está haciendo adecuadamente.

Componentes del grafo de flujo:

Nodo: son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.

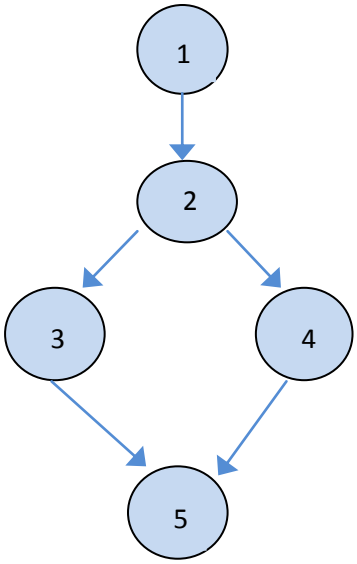
Regiones: son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

Para realizar la prueba de caja blanca específicamente la prueba del camino básico 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 asignarMetodo ($diag_resumen, $id_metdo) el cual se encarga de adicionar métodos a emplear en un diagnóstico.

```
public static function asignarMetodo($diag_resumen, $id_metdo) {
    $retorno = "";//(1)
    $c = new Criteria();//(1)

    $c->add(DiagDiagnosticoPeer::RESUMEN, $diag_resumen);//(1)
    $diag1 = DiagDiagnosticoPeer::doSelectOne($c);//(1)
    $id_diag = $diag1->getIdDiagnostico();//(1)

    $c2 = new Criteria();//(1)
    $c2->add(DiagDignosticoMetodoPeer::ID_DIAGNOSTICO, $id_diag);//(1)
    $c2->add(DiagDignosticoMetodoPeer::ID_METODO, $id_metdo);//(1)
    $resul = DiagDignosticoMetodoPeer::doSelect($c2);//(1)

    if ($resul == NULL) {//(2)
        $metodo_diag = new DiagDignosticoMetodo();//(3)
        $metodo_diag->setIdDiagnostico($id_diag);//(3)
        $metodo_diag->setIdMetodo($id_metdo);//(3)
        $metodo_diag->save();//(3)
        $retorno = 'no esta';//(3)
    } else {
        $retorno = 'esta';//(4)
    }
    return $retorno;//(5)
}
```

Figura 17: Representación del algoritmo asignarMetodo ($diag_resumen, $id_metdo).

En la siguiente figura se muestra el grafo de flujo asociado al código anterior.



4.8 Cálculo de la complejidad ciclomática a partir de un segmento de código.

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Para efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son la cantidad total de aristas del grafo, cantidad total de nodos para la siguiente fórmula:

V (G) = (A – N) + 2

V (G) = (5 – 5) + 2

V (G) = 2

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

Se puede usar también:

V (G) = P + 1

V (G) = 1 + 1

V (G) = 2 Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = R

V (G) = 2

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede plantear que la complejidad ciclomática del código es de 2, lo que significa que existen dos 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. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo. En estas representaciones se subrayan los elementos de cada camino que los hacen independientes a los demás.

En la siguiente tabla se muestran los caminos básicos.

Número	Camino básico
1	1-2-3-5
2	1-2-4-5

Tabla 4 Caminos básicos del flujo.

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico. Para realizarlos es necesario cumplir con las siguientes exigencias:

- Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- Entrada: Se muestran los parámetros que entran al procedimiento.
- Resultados Esperados: Se expone el resultado que se espera devuelva el procedimiento.

Caso de prueba para el camino básico 1:

Camino 1: [1 – 2 – 3 – 5]

Descripción:

Los datos de entrada cumplirán con los siguientes requisitos:

El parámetro id_metdo estará presente en la tabla DiagDiagnosticoMetodoPeer asignado a diag_resumen

Condición de ejecución:

El diag_resumen será “Diagnóstico realizado en el año 2008”.

El id_metdo 3.

Entrada:

$ diag_resumen = “Diagnóstico realizado en el año 2008”.

$ id_metdo = 3.

Resultados esperados:

Se espera que se lance un mensaje con el siguiente texto: “Error, ese método ya fue asignado a ese diagnóstico”.

Caso de prueba para el camino básico 2:

Camino 1: [1 – 2 – 4 – 5]

Descripción:

Los datos de entrada cumplirán con los siguientes requisitos:

El parámetro id_metdo no estará realacionado con el parámetro diag_resumen en la tabla DiagDiagnosticoMetodoPeer.

Condición de ejecución:

El diag_resumen será “Diagnóstico realizado en el año 2009 ”.

El id_metdo 3.

Entrada:

$ diag_resumen = “Diagnóstico realizado en el año 2008”.

$ id_metdo = 3.

Resultados esperados:
Se espera que se lance un mensaje con el siguiente texto: “El método ya fue asignado correctamente”.

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el flujo de trabajo de la función está correcto pues cumple con las condiciones necesarias que se habían planteado.

4.9 Pruebas de caja negra:
Son las pruebas que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código).
Caso de prueba: Conjunto de entradas con datos de prueba, unas condiciones de ejecución, y unos resultados esperados.
Propósito: Identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de la prueba son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto (casos de uso).

Diseño de casos de prueba
Caso de uso: Gestionar Diagnóstico

- **Descripción general**
  El caso de uso se inicia cuando el Jefe del grupo de diagnóstico desea gestionar un diagnóstico, se le da la posibilidad de: adicionar, modificar o eliminar dicho diagnóstico.
  - **Condiciones de ejecución**
    -El Jefe del grupo de diagnóstico se haya autenticado correctamente.
    -El sistema le otorgue los privilegios adecuados.
- **SC Adicionar diagnóstico**

Escenario	Descripción	Resumen	Propósito	Alcance	Flujo central

EC 1.1 Flujo norma de evento “Adicionar”.	Se adiciona un diagnóstico.	V Diagnóstico realizado en el 2009	V Conocer la cantidad de proyectos que usan software libre.	V Todos los proyecto de la universidad.	- El usuario accede a Gestionar Diagnóstico/Adicionar. -Se muestra una interfaz con los campos a llenar (resumen, propósito, alcance, antecedentes). -El usuario llena los campos y escoge la opción ¨Guardar¨. -El sistema valida los datos y los guarda. -El sistema muestra un mensaje ¨ Se han guardado los datos con éxito¨.
		V Diagnóstico realizado en el 2010.	V Conocer cuántos proyectos web se están realizando en la universidad.	V Todos los proyectos.	
EC 1.2 Datos incorrectos.	Se quedan campos sin llenar o con datos incorrectos.	I	V Conocer la cantidad de proyecto que usan software libre..	V Todos los proyecto de la universidad.	- El usuario accede a Gestionar Diagnóstico/Adicionar. -Se muestra una interfaz con los campos a llenar (resumen, propósito, alcance, antecedentes). -El usuario llena los campos y escoge la opción ¨Guardar¨. -El sistema valida los datos -El sistema muestra un mensaje ¨Existen campos vacios¨.
		V Diagnóstico realizado en el 2010,	I	V Todos los proyectos.	
EC 1.3 ¨Cancelar¨.	Se desea cancelar una operación.	V Diagnóstico realizado en el 2009	V Conocer la cantidad de proyecto que usan software libre.	V Todos los proyecto de la universidad.	- El usuario accede a Gestionar Diagnóstico/Adicionar. -Se muestra una interfaz con los campos a llenar (resumen, propósito, alcance, antecedentes). -El usuario llena los
		V Diagnóstico	V Conocer	V Todos los	

			realizado en el 2010.	cuántos proyectos web se están realizando en la universidad.	proyectos.	campos y elige la opción ¨Cerrar¨. -El sistema cancela la operación y regresa a la vista principal.	

Escenario	Descripción	Resumen	Propósito	Alcance	Antecedente	Respuesta del sistema	Flujo central
EC 1.1 Flujo normal de evento “Modificar Diagnóstico”.	Se modifica el diagnóstico.	V Diagnóstico realizado en el 2009	V Conocer la cantidad de proyectos web de la universidad	V Toda la universidad	V Diagnóstico del 2008	-El sistema muestra un mensaje indicando que se modificó correctamente. ..	- El usuario accede a Gestionar Diagnósticos, selecciona el diagnóstico y elige la opción: ¨ Editar ¨. -Se muestra una interfaz con los campos a modificar (resumen, propósito, alcance, antecedente). -El usuario modifica alguno de los campos y elige la opción ¨Editar¨. -El sistema valida los datos y los guarda. -El sistema muestra un mensaje ¨
		V Diagnóstico realizado en el 2009 a todas las facultades	V Conocer el número de estudiantes en proyecto	V Las Facultades de la universidad	V Diagnóstico del 2008		

							La operación se ha realizado satisfactoriamente¨
EC 1.2 Datos incorrectos.	Se dejan campos con datos sin llenar.	V Diagnóstico realizado en el 2009	V Conocer la cantidad de proyectos web de la universidad	V Toda la Universidad	I	El sistema muestra un mensaje especificando que se deben llenar los campos correctamente.	- El usuario accede a Gestionar Diagnóstico, selecciona el diagnóstico a modificar y va a la opción: ¨Editar¨ - Se muestra una interfaz con los campos a modificar (resumen, propósito, alcance, antecedente). -El usuario modifica alguno de los campos y elige la opción ¨Aceptar¨. -El sistema valida los datos entrados. -El sistema muestra un mensaje ¨Debe entrar correctamente los
		V Informática 2011.	V Feria del producto cubano.	V Presentar todo tipo de producto.	V 20/05/2011		

							campos¨.
EC 1.3 ¨Cancelar¨	Se desea cancelar la operación modificar	V Diagnóstico 2008.	V Conocer el estado de la producción.	V Toda la universidad.	V Diagnóstico del 2007	-El sistema muestra un mensaje indicando si desea cancelar la operación.	- El usuario accede a Gestionar Diagnóstico, selecciona el diagnóstico a modificar y va a la opción: Editar. - Se muestra una interfaz con los campos a modificar (resumen, propósito, alcance, antecedente). -El usuario llena los campos y elige la opción ¨Cancelar¨ -El sistema vuelve a la vista principal.
		V Diagnóstico 2008.	V Conocer la cantidad de proyectos web.	V La universidad.	V Sin antecedente.		

- **SC Eliminar Diagnósticos**

Escenario	Descripción	Título	Resumen	Descripción	Fecha creación	Fecha expiración	Respuesta del sistema	Flujo central

EC 1.1 Flujo norma de evento “Eliminar Diagnóstico”.	Se desea eliminar un diagnóstico						-El sistema elimina el diagnóstico	- El usuario accede a Gestionar Diagnóstico, selecciona el Diagnóstico a eliminar y va a la opción: ¨Eliminar ¨. -Se muestra un mensaje de confirmación. -El usuario selecciona la opción ¨Aceptar¨ -El sistema elimina el evento.
EC 1.2 ¨Cancelar¨.	Se cancela la opción eliminar.						-Se cancela la operación y se vuelve a la página principal.	- El usuario accede a Gestionar Diagnóstico, selecciona el Diagnóstico a eliminar y va a la opción: Eliminar. -Se muestra un mensaje de confirmación. -El usuario da en la opción ¨Cancelar¨ -El sistema va a la vista principal.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
V1	Resumen	Componente de diseño: texto.	No.	Se debe introducir un resumen del diagnóstico, puede tener números también ejemplo: Diagnóstico 2009.
V2	Propósito	Componente de diseño: texto.	No	Se debe introducir el propósito del diagnóstico puede ser texto o la unión de números y texto, ejemplo Realizar 25 encuestas sobre la calidad por proyecto.
V3	Alcance	Componente de diseño: texto.	No	Se debe introducir una descripción del evento puede ser texto y números.
V4	Antecedente	Componente de diseño: texto.	No	Se debe introducir el antecedente del diagnostico puede ser texto y números.

Caso de uso: Gestionar Proyecto

- **Descripción general**

El caso de uso se inicia cuando el Jefe del grupo de diagnóstico desea gestionar un proyecto, se le da la posibilidad de: adicionar, modificar o eliminar dicho proyecto.

- **Condiciones de ejecución**

-El Jefe del grupo de diagnóstico se haya autenticado correctamente.

-El sistema le otorgue los privilegios adecuados.

SC Adicionar Proyecto

Escenario	Descripción	Nombre	Fecha Inicio	Fecha Fin	Descripción	Área	Responsable	Respuesta del sistema	Flujo central
EC 1.1 Flujo norma de	Se adiciona un proyecto.	V Calidad	V 10/1/2010	V 20/5/2011	V Proyecto encargado	V Infraestructura	V Carlos.	-El sistema muestra un mensaje	- El Jefe del grupo de diagnóstico accede a Gestionar

evento “Adicionar Proyecto ”.					de la calidad en la universidad .	Productiva.		indicando que se insertó correctamente.	Proyecto. -Se muestra una interfaz con los campos a llenar (Descripción, nombre, fecha inicio, fecha fin, área, responsable). -El Jefe del grupo de diagnóstico llena los campos y escoge la opción ¨Aceptar¨ -El sistema valida los datos y los guarda. El sistema muestra un mensaje ¨ Se guardaron los datos con éxito¨
		V 12/05/2011	V 0021A	V JCE	V Este evento se está realizando …	V Elmer	V Es un archivo.		
EC 1.2 Datos incorrectos	Se quedan campos sin llenar o con datos incorrectos.	I	V 10/1/2010	V 20/5/2011	V Proyecto encargado de la calidad en la universidad	V Infraestructura Productiva.	V Carlos.	-El sistema muestra un mensaje donde especifica que debe llenar todos los campos correctamente	- El Jefe del grupo de diagnóstico accede a Gestionar Proyecto. -Se muestra una interfaz con los campos a llenar (nombre, fecha inicio, fecha fin, descripción, área, responsable). -El Jefe del grupo de diagnóstico llena los campos y escoge la opción ¨Aceptar¨ El sistema valida los datos. -El sistema muestra un mensaje ¨Existen campos vacios¨.
		V Calidad	I	V 10/5/2010	V Proyecto encargado de la calidad en la universidad .	V Infraestructura Productiva	V Carlos.		

EC 1.3 ¨Cancelar¨	Se desea cancelar una operación.	V Calidad	V 10/5/2010	V 10/8/2011	V Proyecto encargado de la calidad en la universidad	V Infraestructura Productiva.	V Carlos.	-El sistema cancela la operación y regresa a la página principal.	- El Jefe del grupo de diagnostico accede a Gestionar Proyecto. -Se muestra una interfaz con los campos a llenar (nombre, fecha inicio, fecha fin, descripción, área, responsable). -El Jefe del grupo de diagnóstico llena los campos y escoge la opción ¨Cancelar¨. -El sistema cancela la operación y regresa a la vista principal.
		V Calidad	V 25/9/2010	V 15/2/2011	V Proyecto encargado de la calidad en la universidad …	V Infraestructura Productiva.	V Raúl		

- **SC Eliminar Proyecto**

Escenario	Descripción	Nombre	Fecha Inicio	Fecha Fin	Descripción	Área	Responsable	Respuesta del sistema	Flujo central
EC 1.1 Flujo norma de evento “Eliminar Proyecto”.	Se desea eliminar un proyecto.							-El sistema elimina el proyecto.	- El Jefe del grupo de Diagnóstico accede a Gestionar Proyecto -El Jefe del grupo de Diagnostico selecciona el proyecto a eliminar. -El Jefe del grupo de diagnóstico Selecciona la opción ¨Eliminar¨ -El sistema muestra un mensaje de confirmación. -El sistema elimina

									la publicación.
EC 1.3 ¨Cancelar¨.	Se desea cancelar una operación.	V Calidad	V 20/1/2010	V 5/10/2011.	V Proyecto de gestión..	V Área 5	V Alejandro.	-El sistema cancela la operación y regresa a la página principal.	- El Jefe del grupo de diagnóstico accede a Gestionar Proyecto. -El Jefe del grupo de diagnóstico selecciona el proyecto a eliminar. -El sistema cancela la operación y regresa a la vista principal.
		V Calidad Total	V 20/12/2010	V 28/9/2011	V Proyecto especial …	V Área 2	V José		

- **Descripción de las variables.**

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
V1	Nombre	Componente de diseño: texto.	No.	El nombre puede ser la combinación de texto y números.
V2	Fecha Inicio	Componente de diseño: Date.	No	Se debe introducir una fecha seleccionándola en el formato dd/mm/aaaa.
V3	Fecha Fin	Componente de diseño: Date.	No	Se debe introducir una fecha seleccionándola en el formato dd/mm/aaaa.
V4	Descripción	Componente de diseño: texto	No	La descripción puede ser la combinación de texto y números.
V5	Área	Componente de diseño: texto.	No	El área puede ser la combinación de texto y números.

V6	Responsable	Componente de diseño: texto	No	Se debe introducir un responsable seleccionándolo.

4.10 Validación del modelo de diseño utilizando métricas

Un aspecto importante a tener en cuenta en la evaluación de la calidad del diseño ha sido la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto referenciadas por Pressman, teniendo en cuenta que este estudio brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software.

Atributos de calidad que se abarcan:

Responsabilidad. Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

Complejidad del diseño. Consiste en la complejidad que posee una estructura de diseño de clases.

Complejidad de implementación. Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización. Consiste en el grado de reutilización de 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 costes y la planificación del proyecto.

Cantidad de pruebas. Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, modulo, clase, conjunto de clases, etc.) diseñado.

Nivel de Cohesión. Consiste en el grado de especialización de las clases concebidas para modelar un dominio o concepto específico.

Abstracción del diseño. Consiste en la capacidad de modelar lo más cercano posible a la realidad un concepto o dominio determinado.

4.10.1 Tamaño Operacional de Clase (TOC)

Está dado por el número de métodos asignados a una clase, y evalúa los siguientes atributos de calidad:

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

Tabla 5: Tamaño Operacional de la Clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

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

Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Tabla 6: Rango de valores para los criterios de evaluación de la métrica TOC.

Al aplicar la métrica TOC se determinó que el módulo consta de 8 clases y 78 procedimientos, reflejados en la siguiente tabla junto a los atributos de calidad de Responsabilidad, Complejidad de Implementación y Reutilización. Se obtuvo así mismo, un promedio aproximado de 9 procedimientos por clase, y los siguientes datos sirvieron para categorizar los atributos por cada clase:

Promedio de procedimientos por clase= 9.

Responsabilidad**:** Baja <=9, 9 < Media < 18, Alta > 18.

Complejidad**:** Baja <=9, 9 < Media < 18, Alta > 18.

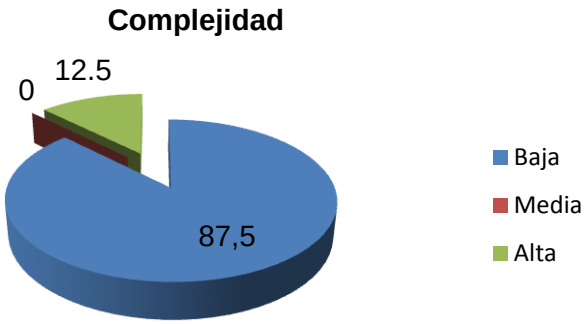
Reutilización**:** Baja > 18, 9 < Media < 18, Alta <= 9.

Clases	Funciones	Responsabilidad	Complejidad	Reutilización
adminActions	32	Alta	Alta	Baja
AdminUsuarioGrupoPeer	7	Baja	Baja	Alta
AdminUsuarioPeer	8	Baja	Baja	Alta
DiagEvaluacionDesempenoPeer	6	Baja	Baja	Alta
DiagDiagnosticoPersonaPeer	4	Baja	Baja	Alta

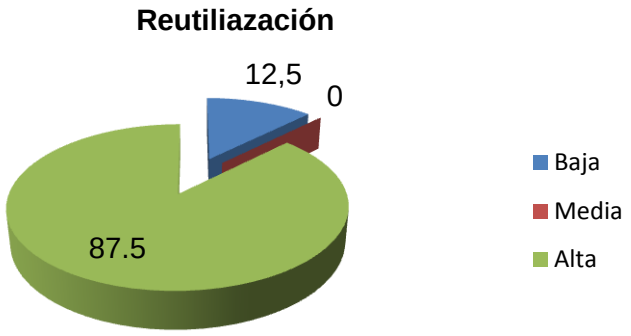
DiagDiagnosticoMetodoPeer	6	Baja	Baja	Alta
InfPersonaProyectoPeer	3	Baja	Baja	Alta
NAreaPeer	8	Baja	Baja	Alta

Tabla 7: Resultados de la evaluación de la métrica TOC.

Grafica 1: Porciento de clases por categorías del atributo Responsabilidad.



Grafica 2: Porciento de clases por categorías del atributo Complejidad.



Grafica 3: Porciento de clases por categorías del atributo Reutilización.

Tras un análisis de los resultados arrojados por la evaluación del sistema bajo los instrumentos de medición de la métrica TOC, se demuestra que los valores idóneos para cada uno de los atributos de calidad evaluados se alcanzaron, puesto que, como se puede observar, el 83.3% de las clases del software contienen un número menor que el promedio de procedimientos para una clase, lo cual influye positivamente en el hecho de que predomine una responsabilidad baja de las clases en un 83.3%, y hace que carezcan de mucha complejidad, por lo que se tornan mucho más reutilizables. Solamente un 16.7 % de las clases tienen alta posibilidades de reutilización. Estos resultados demuestran la solidez del diseño de la implementación de este sistema, y garantizan que este trabaje de forma rápida y eficiente.

4.11 Prueba de estrés

Esta prueba se utiliza normalmente para evaluar la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que presente anomalías. Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

Con la realización de las pruebas de estrés a través de la herramienta JMeter se detectó que el sistema en situaciones normales soporta la conexión de 200 usuarios conectados concurrentemente. Para un número mayor de usuarios la aplicación comienza a presentar fallas en dar respuestas a las peticiones

que se realizan, el tiempo mínimo de respuesta es de 78 milisegundo en situaciones normales aumenta a 547 milisegundo y el riesgo de presentar problemas es de 11.67 %.

4.5 Conclusiones

En este capítulo se mostró el diagrama de despliegue del sistema así como los diferentes diagramas de componentes, los cuales detallan las relaciones que se establecen entre las clases que contienen las implementaciones y validaciones, siguiendo la arquitectura del modelo-vista-controlador utilizada para desarrollar este sistema logrando un mejor entendimiento del funcionamiento basado ya en su código.

Conclusiones Generales

Durante el desarrollo del presente trabajo se llevaron a cabo una serie de fases que permitieron dar cumplimiento al objetivo planteado y que abarcaron todas las tareas investigativas propuestas.

Se hizo un estudio de los DO en el mundo y en Cuba permitiendo comprender e identificar las características que hacen único dicho proceso en la UCI. Se desarrollaron artefactos que describen los procesos de ingeniería de software mediante la aplicación de la metodología RUP. Se abordo varias bibliografías concernientes las tecnologías actuales para el desarrollo de sistemas web, enfatizando en las tecnologías y herramientas libres, de lo que se concluyó utilizar para la implementación del sistema, como lenguaje de programación PHP 5 en su versión 5.2.5, con el framework Symfony versión 1.2.8, sobre el entorno de desarrollo Netbeans 6.8 para PHP y utilizando como SGBD PostgreSQL en su versión 8.3.4.

Además se validaron las funcionalidades en el producto para comprobar que cumpliera con los requisitos funcionales, con una interfaz visual amigable y sencilla. La ejecución de las tareas en el sistema obtenido ocurre de forma aceptable.

Mediante el análisis de los resultados se llega a la conclusión de que se cumplió con el objetivo trazado en la investigación del presente trabajo de diploma. Realizar la implementación de los módulos Diagnóstico y Administración del Sistema de Diagnóstico a la Producción en la UCI para reducir el uso de recursos y disminuir el esfuerzo empleado por parte del Grupo de Diagnóstico de CALISOFT.

Recomendaciones

Después haber culminado el presente trabajo y ver su resultado se propone la siguiente recomendación:

1. Agregarle nuevas funcionalidades para mejorar la gestión de la información en el mismo, así como aumentar su valor agregado.