Universidad de las Ciencias Informáticas

Facultad 3

Título: “Sistema Informático para la Gestión de la Planificación de las Acciones de CTI y de Postgrado del centro CEIGE”.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autora: Yissell Soto Hidalgo

Tutora: Ing. Iyugnis Leyva Báez

Consultante: Ing**.** Damián Oscar Falcón Borras

La Habana, Cuba, 2013.

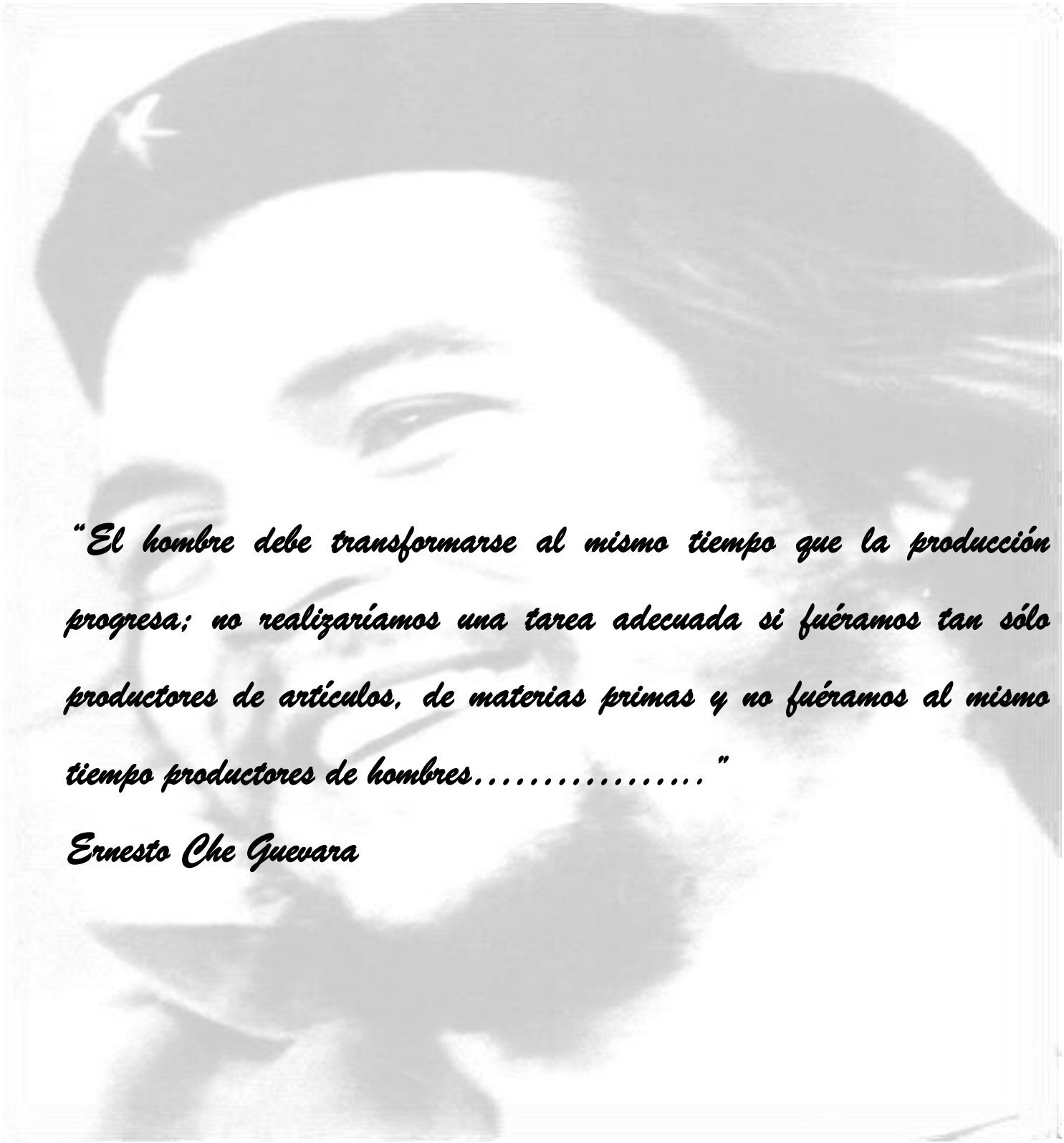
“Año del 54 Aniversario del Triunfo de la Revolución.”

Declaración de autoría

Declaro que yo Yissell Soto Hidalgo, soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año __2013____.

Yissell Soto Hidalgo	Ing. Iyugnis Leyva Báez
_____	_____
Autora	Tutora



"El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres....."

Ernesto Che Guevara

Datos de contacto

Tutora:

- Nombre y apellidos: Ing. Iyugnis Leyva Báez.
- Correo electrónico: ileyva@uci.cu.
- Situación laboral: Profesor.
- Institución: Universidad de las Ciencias Informáticas (UCI).
- Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.
- Área: CEIGE. Departamento de Desarrollo de Productos.

Agradecimientos

Muchas han sido las personas que han hecho de este sueño, una realidad. Por lo que hoy, quiero agradecerles por todo lo conseguido, y por ser parte de esos momentos que nunca saldrán de mi mente: A mis padres Addis Hidalgo y Sergio Soto; a mis hermanas por permanecer a mi lado en todo momento, dándome apoyo moral y fuerzas. A Luis Jonás Gonzáles García por formar parte de mi vida.

A mis compañeros por todos esos momentos que hemos vivido como una gran familia, y en especial a mis mejores amigos: Martha Rocío Fonseca Vega, Javier Carmona Martínez y Luis Ángel Valderrama.

A mi tutora Iyugnis Leyva Báez, al tribunal y al oponente que han contribuido en mi formación como profesional y me han guiado en el desarrollo de esta investigación, gracias por haber colaborado en mi carrera universitaria. A todos aquellos que han brindado su ayuda desinteresada para la creación de esta investigación.

A mi grupo de estos 5 años, los cuales me han ayudado mucho a lo largo de mi trayectoria como estudiante, a mi grupo de kisomba "Los sensuales de la pista", quienes adoro con el corazón.

Dedicatoria

A mi madre Addis Hidalgo Anazco, por ser mi gran ejemplo y enseñarme a ser fuerte y seguir adelante. A mi padre Sergio Soto Rodríguez por siempre estar a mi lado. A mis hermanas Niurka y Eisel por todo el apoyo que me han brindado. A mis sobrinos Evelyn, Ever, Melany, Claudia por ser lo que más aprecio en la vida. A toda mi familia en general por su apoyo incondicional.

Resumen

Una de las principales tareas de la Subdirección de Investigación y Postgrado del Centro de Informatización de la Gestión de Entidades (CEIGE), es la planificación de las acciones de ciencia, tecnología e innovación (CTI) y postgrado, donde se lleva a cabo la formación profesional de los investigadores y profesores de la Universidad de las Ciencias Informáticas (UCI). Dicho personal se encarga de realizar el control de las diferentes actividades como talleres, eventos científicos, tutorías a estudiantes en diferentes eventos, publicación de artículos, maestrías y diplomados. Actualmente no se cuenta con un sistema informático que agilice el proceso de planificación y seguimiento de las tareas a cumplir en el plan trimestral. Para la búsqueda de una solución se hace necesario realizar un estudio de aplicaciones informáticas existentes a nivel mundial y nacional relacionadas con el tema, así como una panorámica de las tendencias actuales y la tecnología a utilizar. Fue definido como lenguaje de programación PHP y la utilización del marco de trabajo Sauxe, el cual reutiliza las siguientes tecnologías libres: ExtJS, Zend Framework y Doctrine. Además se utilizó como servidor de base de datos PostgresSQL y como servidor de aplicaciones web Apache. El presente trabajo tiene como principal objetivo proponer una solución que facilite el proceso de gestión de la planificación de la Subdirección de Investigación y Postgrado del CEIGE.

Palabras claves: postgrado, ciencia, tecnología e innovación.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Cuba es un país que dada la voluntad política de su gobierno por universalizar la enseñanza, hoy cuenta con una amplia red de instituciones de Educación Superior. La palabra clave en su proyección es la mejora continua de la calidad de los procesos que desarrollan. La calidad y la pertinencia han estado siempre en el centro del que hacer universitario en Cuba en estas últimas décadas. El eje de todo el trabajo que se realiza y emana de la universidad, lo constituye su dimensión educativa, conformada en lo esencial por el desarrollo de valores de alto contenido humanista, la construcción de una ética profesional y la concientización de la responsabilidad ciudadana [1].

El conjunto de saberes y habilidades, además de las propias de la profesión, tienen que acentuar aquellas que distingan al profesional por sus valores y convicciones, que le promuevan el desarrollo de un pensamiento crítico, creativo e integrador. La universidad es para el estudiante el lugar que le brinda la oportunidad de poner en tensión toda su disposición de aprendizaje; que le eleva sustancialmente su responsabilidad individual como gestor del conocimiento; le revela las contradicciones en la sociedad y en las ciencias, y le enseña a identificarlas. La universidad es la institución que lo involucra en la búsqueda de soluciones a los problemas reales de la sociedad y lo incita a adentrarse a través de la investigación en los misterios aún no revelados de las ciencias y la naturaleza [1].

La investigación científica es una actividad social que requiere de personas con capacidades para realizarla. Actualmente la investigación científica es efectuada por equipos multidisciplinarios, los investigadores de un área o de áreas afines forman las llamadas comunidades científicas, socializando aún más el desarrollo científico, por lo que los estudios de ciencia, tecnología e innovación son factores fundamentales en el desarrollo de la sociedad [2]. De ahí que se considere indispensable la formación integral de los graduados universitarios, aunque muchos en su afán de superarse continuamente aspiran a mejorar el título universitario de grado a través de especializaciones en la rama estudiada.

La inquietud intelectual, la necesidad de actualización y la especificidad de tareas o trabajos profesionales conllevan a la necesidad de realizar estudios de postgrado. Al graduarse como ingenieros, se tiene la posibilidad de acceder a los diferentes programas de postgrados, cursos, diplomados, maestrías, doctorados. Se le llama estudio de posgrado o postgrado a los estudios de especialización posteriores al título de grado. Dicha educación constituye la superación profesional, productora de profesores e intelectuales del más alto nivel [2].

Los estudios de postgrado profundizan los conocimientos teóricos y tecnológicos, los cuales se pueden orientar hacia una maestría, pero si el objetivo es la realización de verdaderos aportes originales a la ciencia con proyección de la universalidad en un marco de nivel de excelencia académica, se hace entonces referencia al doctorado. En Cuba el postgrado se ha desarrollado aceleradamente en los últimos años, debido al difícil reto de obtener una mayor cantidad de profesionales en las especialidades, esto demuestra la importancia alcanzada por la superación en la comunidad universitaria cubana.

La UCI como las demás instituciones de la Educación Superior del país, trabajan sistemáticamente en la formación y creación de programas para la ejecución de cursos, como entrenamientos y diplomados para los graduados universitarios. Además el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha posibilitado un mejor manejo de la información y el acceso a datos a nivel mundial y forma parte del desarrollo tecnológico que existe en la actualidad. La gestión de los datos a través de Excel se hace difícil dado que la tendencia va hacia el crecimiento exponencial de estos, lo cual hace que el proceso sea abstracto y dificulta la planificación y control de las tareas a diferentes niveles en las cuales participan los profesores, motivando el desarrollo de nuevas herramientas de gestión.

En la UCI el uso de los sistemas de gestión de la planificación se ha vuelto una necesidad, como solución a diferentes problemáticas que han surgido. El personal de la Subdirección de Investigación y Postgrado en el CEIGE debe realizar el control de los cursos de postgrado y doctorados para lograr la superación de los profesionales, así como velar por el análisis y cumplimiento del plan de las acciones de CTI asignada a los profesionales según sus categorías docentes. El proceso de planificación de las acciones de CTI y de postgrado se ejecuta con una serie de deficiencias que dificultan la ejecución en tiempo y con calidad de algunos procesos relacionados. Esto afecta el seguimiento y control de los objetivos estratégicos anuales del centro que están relacionados con los planes de CTI que se establecen desde el nivel individual hasta los niveles superiores. Algunas de las problemáticas que se presentan son:

- No existe una vista única de las tareas de CTI y Postgrado que son planificadas para cada profesional en un año natural, lo que provoca que haya demoras en el análisis de la carga de trabajo que representan estas tareas en su plan de trabajo anual.
- Existen dificultades para consolidar la planificación de CTI y Postgrado de las líneas de desarrollo, los departamentos y posteriormente a nivel de centro, esto está dado porque el

proceso se hace manualmente y consume tiempo, en la misma proporción a la cantidad de personas que se les está planificando tareas de CTI.

- Cuando se realizan cambios en la planificación de CTI a cualquier nivel entonces cada nivel administrativo superior debe actualizar los planes que previamente se habían consolidado, duplicando el esfuerzo.
- Las actividades de seguimiento y control de los planes de CTI se realizan con lentitud y deficiencia debido a que los profesionales no pueden hacer cambios directamente en los planes y tienes que realizarlo a través de la comunicación verbal con su jefe inmediato el cual posteriormente tiene que informar de igual modo a la estructura administrativa superior.
- El proceso de toma de decisiones en la actividad de seguimiento y control del plan de CTI se realiza con deficiencias debido a que no es posible tener actualizado el estado de cumplimiento de los planes de CTI individuales y del área, lo que provoca que la dirección del departamento y el centro no puedan realizar acciones inmediatas para corregir las desviaciones y problemas que pudieran afectar el cumplimiento de este.

A partir de la problemática descrita, se define el siguiente **problema a resolver**: La actual gestión de las acciones de CTI y Postgrado del centro CEIGE afecta su planificación y control.

Dando lugar a que el **objeto de estudio** de la investigación sea: Sistemas informáticos para la planificación y control de las acciones de CTI y Postgrado.

Estableciendo como **campo de acción**: Sistema informático para la planificación y control de las acciones de CTI y Postgrado del CEIGE.

Se plantea como **objetivo general** para darle solución al problema establecido: Desarrollar un Sistema Informático para la Gestión de la Planificación y Control de las Acciones de CTI y de Postgrado en el centro CEIGE.

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

- Establecer las principales tendencias en el proceso de gestión de la planificación de las acciones de CTI y Postgrado para realizar el análisis de ciertos softwares que sean similares a la propuesta en desarrollo a nivel nacional e internacional.

- Realizar el análisis y diseño del Sistema Informático para la Gestión de la Planificación de las Acciones de CTI y Postgrado del CEIGE para satisfacer las necesidades de la Subdirección de Investigación y Postgrado.
- Implementar el Sistema Informático para la Gestión de la Planificación de las Acciones de CTI y Postgrado del CEIGE bajo la tecnología establecida para darle solución a la problemática planteada.
- Validar la solución propuesta para el buen funcionamiento del sistema.

Se define como **idea a defender:** El desarrollo de un Sistema Informático para la Planificación y Control de las Acciones de CTI y Postgrado en el centro CEIGE posibilitará mejorar la actual gestión de las mismas.

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

De los métodos teóricos se utilizaron:

- **Analítico-Sintético:** este método se utilizará para realizar el análisis de teorías, permitiendo la extracción de los elementos más importantes que se relacionan con los sistemas de gestión. Para analizar a través de una profunda búsqueda las tecnologías, herramientas y metodologías a utilizar en el desarrollo de la aplicación. Además permitirá arribar a conclusiones teóricas conceptuales.
- **Histórico-Lógico:** este método se utiliza en dicha investigación con el objetivo de estudiar todo lo referente a los sistemas de gestión de la planificación de actividades existentes a nivel mundial y nacional, específicamente en la Universidad de las Ciencias Informáticas, para analizar sus características. Este método ayuda a dar cumplimiento al estudio del estado del arte.
- **Modelación:** para definir el negocio durante la fase de análisis y poder realizar la modelación del mismo durante la fase de implementación.

Dentro de los métodos empíricos se utilizó:

- **Entrevista:** se aplica este método con el objetivo de obtener información sobre la gestión de postgrado en el centro CEIGE, intercambiando directamente con el personal involucrado en el mismo. Además permite reunir y determinar la información básica necesaria para la recogida de requisitos del sistema.

A continuación se detallará la estructura del trabajo del diploma.

Estructura del documento

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

Capítulo 1: se realiza un análisis acerca de la fundamentación teórica necesaria para la investigación del proceso de postgrado, donde se mencionan y especifican diferentes sistemas de gestión de planificación existentes a nivel mundial y nacional. Se describen los patrones de diseño y arquitectónicos a utilizar. Finalmente, se justifican las metodologías, lenguajes de programación y tecnologías empleadas para el modelado de la aplicación y necesarias para su posterior implementación.

Capítulo 2: en este capítulo se hace un análisis y descripción general de la propuesta de desarrollo del sistema, así como la realización del levantamiento de los requisitos funcionales y no funcionales, entrada de la implementación del software. También se tienen en cuenta la confección del modelo conceptual, el mapa de procesos del negocio, los diagramas de clase del diseño, el modelo de datos, los diagramas de secuencia y el de despliegue. Por último fueron definidos los estándares de codificación a utilizar.

Capítulo 3: en este capítulo se describe la construcción de la solución, explicando los aspectos principales de la implementación, haciendo una descripción de las clases y funcionalidades más importantes del software. Se evalúa el diseño propuesto empleando las métricas Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC). Finalmente se aplican las pruebas de caja negra para validar el correcto funcionamiento del sistema.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

La planificación es indispensable para llevar a cabo la organización del proceso de postgrado. Los sistemas de gestión permiten que la información esté disponible, mejora la comunicación, eficiencia y la toma de decisiones relacionadas con el proceso de seguimiento y control del cumplimiento de los objetivos estratégicos de CTI y postgrado del CEIGE. En este capítulo se realiza un estudio de diferentes sistemas informáticos vinculados a la gestión de la planificación de actividades y se especifican los lenguajes de programación, herramientas y tecnologías a utilizar teniendo en cuenta el marco de trabajo Sauxe y el modelo de desarrollo definido por el CEIGE. Además se hará referencia a las técnicas de captura y validación de requisitos.

1.2. Conceptos asociados al dominio del problema

Posgrado o postgrado: se le llama a los estudios universitarios posteriores al título de grado y comprende los estudios de maestría (también denominados máster o magíster) y doctorado. Se trata de un nivel educativo que forma parte del tipo superior o de tercer ciclo. Es la última fase de la educación formal, tiene como antecedente obligatorio la titulación de pregrado [2].

Tutoría: entendida como elemento individualizador a la vez que integrado de la educación, es un componente esencial de la función docente. Se utiliza con la finalidad de comprender y ayudar al estudiante y mejorar su desempeño escolar también su mentalidad frente a las tendencias tradicionales. Tiene por objetivo asegurar que la educación sea verdaderamente integral y personalizada y no quede reducida a un simple trasvase de conocimientos [3].

Planificación: la planificación, la planeación o el planeamiento, es el proceso metódico diseñado para obtener un objetivo determinado. En el sentido más universal, implica tener uno o varios objetivos a realizar junto con las acciones requeridas para concluirse exitosamente. Es un proceso de toma de decisiones para alcanzar un futuro deseado, teniendo en cuenta la situación actual y los factores internos y externos que pueden influir en el logro de los objetivos [3].

Ciencia: la ciencia (del latín scientia conocimiento) es el conjunto de conocimientos sistemáticamente estructurados y susceptibles de ser articulados unos con otros. La ciencia surge de la obtención del conocimiento mediante la observación de patrones regulares, de razonamientos y de experimentación en ámbitos específicos [3].

Tecnología: es el conjunto de conocimientos técnicos, ordenados científicamente, que permiten diseñar, crear bienes, servicios que facilitan la adaptación al medio ambiente y satisfacer tanto las necesidades esenciales como los deseos de las personas. Tecnología, puede referirse tanto a la

disciplina teórica que estudia los saberes comunes a todas las tecnologías como la educación tecnológica, la disciplina escolar abocada a la familiarización con las tecnologías más importantes [3].

Innovación: significa literalmente novedad o renovación. La palabra proviene del latín innovare. En el uso coloquial y general, el concepto se utiliza de manera inespecífica en el sentido de nuevas ideas e inventos y su implementación económica. En el sentido estricto, en cambio, se dice que de las ideas solo pueden resultar innovaciones luego de que ellas se implementan como nuevos productos, servicios o procedimientos y que realmente encuentran una aplicación exitosa imponiéndose en el mercado, a través de la difusión [4].

1.3. Fundamentación Teórica

Una correcta planificación del proceso de postgrado y de las acciones de CTI permite mejorar la toma de decisiones con la meta de concretar un fin buscado. Por otra parte, la tendencia a la internacionalización en la educación superior y el proceso de globalización reclaman niveles de calidad. Todos los universitarios que realizan actividades de postgrado exigen su derecho a conocer datos y especificaciones acerca de la calidad ofrecida por la institución a la que ingresan y en donde reciben su formación obligando a las universidades a ofrecer evidencia de la planificación que realiza en los procesos de postgrado. Esto requiere realizar un análisis acerca de las actividades a desarrollar en el mismo y cómo se lleva a cabo el proceso de planificación de las tareas asignadas a los profesionales. Con el fin de garantizar la calidad del proceso de superación en la UCI se realizó un estudio de la existencia de algunos sistemas informáticos para la gestión de la planificación de los cursos de postgrado tanto a nivel internacional como nacional.

1.3.1. Experiencias en el uso de sistemas de gestión de la planificación de postgrado a nivel internacional

Sistema Automatizado de información y evaluación de procesos SICOTLAX

Sistema de información integral diseñado e implementado en el Colegio mexicano de Tlaxcala para automatizar los servicios de apoyo académico, de divulgación y administrativos que se realizaban de forma manual, contribuyendo así al desarrollo de las actividades de investigación y enseñanza-aprendizaje [5]. A través de Sicotlax en este colegio se realizan automáticamente los procesos de admisión, carga académica, programas de asignatura y se obtiene además reportes de la trayectoria académica de profesores e investigadores.

Brinda funcionalidades como:

- Permite contar con diferentes accesos según la instancia, profesores y responsables de postgrado, de manera que la información sea proporcionada o modificada según el responsable.
- En forma conjunta con la dirección académica, al concluir los cursos la coordinación de postgrado realiza un informe y un balance de los resultados de la evaluación de profesores y con base en ello se comunica a los profesores los puntos críticos, los aspectos sobresalientes y las observaciones con la finalidad de buscar alternativas para mejorar el desempeño laboral.
- Permite contar con los elementos para desarrollar una evaluación permanente de las actividades académicas y aplicar sus resultados en la mejora de la calidad [5].

Con el estudio de este software se adquiere una idea de la organización y estructura, así como las funcionalidades necesarias para la confección de la solución. Vale resaltar que aunque este sistema posee características muy apropiadas es desarrollado para los procesos específicos del colegio Tlaxcala. Además de las limitaciones que constituye el elevado precio de un producto extranjero, el pago de sus licencias y la dependencia que se crea a un suministrador externo.

SIU-Guaraní

El SIU-Guaraní es un sistema de gestión de alumnos que registra y administra todas las actividades académicas de la universidad, desde que los alumnos ingresan como aspirantes hasta que obtienen el diploma. Fue concebido para administrar la gestión académica de forma segura, con la finalidad de obtener información consistente para los niveles operativos y directivos [6].

Brinda posibilidades como:

- Permite la planificación del plan de acción a seguir en el curso.
- Las personalizaciones necesarias para adaptar el sistema a requerimientos propios de las unidades académicas.
- Consulta de planes de estudio.
- Actualización de datos censales.

El estudio de este software posibilitó identificar algunas funcionalidades iniciales para la confección de la solución. Sin embargo es un sistema en línea, o sea, que se requiere de conexión a Internet y es recomendado que dicha conexión sea con banda ancha, pero debido a las limitaciones que Cuba presenta no es posible utilizarlo ampliamente.

1.3.2. Experiencias en el uso de sistemas de gestión de la planificación de postgrado a nivel nacional

Sistema Automatizado para la gestión de la Educación de Postgrado en un Departamento Docente de la Universidad de Matanzas "Camilo Cienfuegos"

Este sistema informático fue creado por la Universidad de Matanzas "Camilo Cienfuegos", el cual permite almacenar toda la información concerniente a los postgrados, dígase cursos, diplomados, maestrías y doctorados, de manera que pueda ser consultada por parte de aquellos profesores que estén interesados en matricular en un postgrado específico, garantizando de esta forma la gestión de la información y su actualización.

Facilidades que brinda de manera general:

- Permite una correcta manipulación y renovación de la información por parte del personal autorizado.
- Permite que la confección del plan de postgrado se realice de forma rápida y eficiente, siempre y cuando la información se encuentre actualizada, permitiendo el seguimiento del mismo durante el período de ejecución.
- Permite la obtención de reportes del plan de postgrados del departamento, así como el conocimiento del estado de la superación del claustro, su planificación y control.

Este software cumple parcialmente con las posibles funcionalidades de la aplicación en el desarrollo de este trabajo, pero pese a las bondades que brinda para la propuesta de solución utiliza en el manejo de la información el gestor de base de datos Firebird [7], establecido como política de la Universidad de Matanzas "Camilo Cienfuegos", diferente al empleado en el marco de trabajo Sauxe establecido por el cliente.

Sistema Informático para la gestión de la formación de postgrado en los profesionales del municipio Mayarí, SICOP

Es un sistema informático para la gestión de la formación de postgrado en los profesionales del municipio Mayarí. La aplicación del referido proyecto ha significado para los profesionales del territorio la articulación de todo ese potencial humano en función de la superación científica, técnica y cultural; así como la atención de sus necesidades de aprendizaje más apremiantes, para trazar los planes de capacitación y la toma de decisiones al respecto [8].

Brinda posibilidades como:

- Permite la comunicación dinámica entre los actores de esta gestión: entidades, profesionales y la Sedes Universitarias Municipales (SUM); así como el flujo de otras informaciones que esto requiere para la toma de decisiones.

- Apoya a los directivos del municipio, PCC, Gobierno, entidades y la SUM a tomar decisiones respecto a las necesidades de superación de las entidades para los próximos cinco años.

Este sistema se elaboró utilizando la herramienta FoxProx versión 8.0 y solo se ejecuta sobre la plataforma de Windows, por lo que para su uso se deben pagar altos precios de licencias, además el país está apostando por una tecnología libre.

GESPRO

GESPRO es un Paquete para la Gestión de Proyectos desarrollado por la UCI y comercializable desde las empresas comercializadoras asociadas a la universidad. Se encarga de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en los proyectos dentro del alcance, el tiempo y coste definidos. Ventajas de esta herramienta:

- Contiene un sistema operativo y herramientas que soportan los servidores web. Se recomienda Mozilla Firefox versión 3.6 o superior.
- Gestor de bases de datos y herramientas que soportan la conexión a las bases de datos (ORM).
- Herramientas que soportan la toma de decisiones (GESPRO Reportes Plugin v1.0.6).
- Control de las tareas y proyectos (DIP) gestión de recursos (Redmine v1.1.2).

Esta herramienta permite el control y seguimiento de los proyectos del centro. Además, prioriza elementos como la soberanía tecnológica, la seguridad y las propias funcionalidades para la toma de decisiones a diferentes niveles en las organizaciones, sin embargo utiliza el framework Ruby on Rails v2.3.5 de código abierto para aplicaciones web [49], lo que implica tiempo de estudio debido a que esta tecnología no es utilizada en el marco de trabajo Sauxe.

SIPAC

El proyecto SPA es el encargado de desarrollar el producto denominado Sistema de Planificación de Actividades (SIPAC), que es precisamente uno de los subsistemas que componen el sistema integral de gestión de entidades CEDRUX desarrollado por el centro de producción CEIGE de la UCI. Principales funcionalidades que brinda:

- Permite crear grupos de usuarios y a partir de ellos se puede planificar, teniendo en cuenta un dominio de acceso.
- Permite configurar los nomencladores o características generales de la información.
- Gestiona los diferentes elementos de la planificación dígase planes de actividades, áreas de resultados clave (áreas de concentración de metas), objetivos a alcanzar, actividades a cumplir, así

como funcionalidades para registrar otros factores que intervienen en la planificación como son las órdenes, indicaciones o acuerdos entre otros.

- Permite la ejecución del proceso de aprobación de los elementos de la planificación a diferentes niveles organizacionales.

Es un sistema en desarrollo y proceso de prueba que no ha sido aún registrado, por tanto integrarse a él implica riesgos de estabilidad y dependencia. Además, los requisitos funcionales que gestiona son diferentes al sistema propuesto.

Se realizó una búsqueda de sistemas informáticos existentes que contribuyeran a la solución del mismo, sin resultados satisfactorios; por lo que se hace necesario el desarrollo de uno propio para lograr que la información almacenada pueda ser actualizada, esto permitirá planificar, organizar, ejecutar y controlar de forma rápida y segura el plan de acción de CTI y postgrado del CEIGE.

1.4. Modelo de desarrollo de software

El proceso de informatización de la sociedad cubana, ha propiciado el aumento del uso de herramientas informáticas en los principales sectores del país. En tal sentido, la UCI desempeña un rol protagónico, radicando en ella el CEIGE [9]. La producción se concentra en el desarrollo de proyectos generalmente de gran magnitud, por lo que se hace necesario contar con un modelo estandarizado, que establezca las distintas fases por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas. El CEIGE para hacer posible lo mencionado ha creado un Modelo de Desarrollo de Software propio, utilizado en la propuesta de solución.

1.4.1. Fases del ciclo de vida

Inicio o Estudio preliminar: durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación, la evaluación de la factibilidad del proyecto y el registro de este. En esta fase se realiza un estudio inicial de la organización del sistema con el cliente para obtener información fundamental acerca del alcance, realizar estimaciones de tiempo, esfuerzo, costo y decidir si se ejecuta o no el proyecto. Los objetivos de la fase son: asegurar la factibilidad del proyecto y establecer un plan para la ejecución del proyecto. Incluye el plan de desarrollo de software y acta de inicio del proyecto firmada.

Desarrollo: en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

El objetivo de esta fase es: obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales. Incluye el producto liberado por la entidad certificadora de calidad.

En esta fase se ejecutan las disciplinas Modelado de negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas internas y Pruebas de liberación [9].

Ciclo de vida

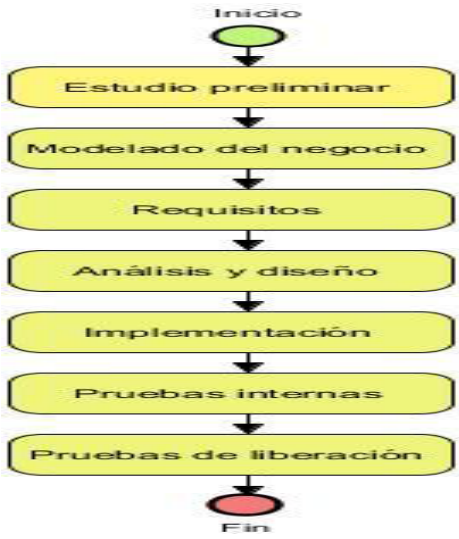


Figura 1: Ciclo de vida de proyectos del CEIGE.

1.5. Marco de trabajo

Para lograr mayores niveles de calidad, productividad y respuesta al cliente en la industria del software, es de vital importancia contar con una base tecnológica sólida o repositorio de componentes reutilizables que apoye y agilice el proceso de desarrollo, de esta forma cuando se inicia un proyecto, ya se cuenta con un por ciento significativo de requisitos resueltos. La UCI desarrolló el marco de trabajo denominado Sauxe con tecnologías libres que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo [10]. Al iniciar un desarrollo sobre Sauxe, se tiene garantizado aspectos como la seguridad, la multi-entidad, el multi-tema, el multi-idioma, la auditoría, la integración, la interoperabilidad, la concurrencia, la administración de transacciones, entre otros. Por las ventajas que proporciona se reutiliza en más de 20 proyectos de la UCI y 10 entidades desarrolladoras de software [11]. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, reutiliza las siguientes tecnologías libres:

Figura 2: Arquitectura de Sauxe.

1.5.1. ExtJS 2.2

Es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM, fue desarrollada por Sencha[1]. En el mercado actualmente existen múltiples librerías de JavaScript que permiten realizar todo tipo de maravillas en el navegador web. ExtJS permite que con pocas líneas de código sea posible realizar interfaces amigables para los usuarios. Es la librería más avanzada para el desarrollo rápido de aplicaciones con una apariencia totalmente novedosa y una arquitectura flexible. ExtJS es una librería JavaScript que permite construir aplicaciones complejas en Internet [12].

1.5.2. Zend Framework 1.9.7

Zend Framework es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. Es una implementación que usa código 100% orientado a objetos [13]. La estructura de los componentes de Zend Framework es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. Entre sus principales características se encuentra: ofrece un gran rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos [14].

1 Sencha Ext JS: plataforma de desarrollo de aplicaciones de escritorio y web para interfaz de usuario moderna.

1.5.3. Doctrine 1.2.1

Doctrine es un mapeador de objetos-relacional (ORM) escrito en PHP que proporciona una capa de persistencia para objetos PHP, uno de sus puntos fuertes es su lenguaje Doctrine Query Language (DQL) inspirado en el HQL de Hibernate. Es una capa de abstracción que se sitúa justo encima de un Sistema de Gestión de Base de Datos [15]. Utilizar un ORM tiene una serie de ventajas que facilita enormemente tareas comunes y de mantenimiento:

1. **Reutilización:** la principal ventaja que aporta un ORM es la reutilización permitiendo llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones.

2. **Encapsulación:** la capa ORM encapsula la lógica de los datos pudiendo hacer cambios que afectan a toda la aplicación únicamente modificando una función.

3. **Portabilidad:** utilizar una capa de abstracción nos permite cambiar en mitad de un proyecto de una base de datos sin ningún tipo de complicación, utiliza una sintaxis propia del ORM que es capaz de traducir a diferentes tipos de bases de datos.

4. **Seguridad:** los ORM suelen implementar mecanismos de seguridad que protegen a la aplicación de los ataques más comunes como SQL Injection.

5. **Mantenimiento del código:** gracias al correcto ordenamiento de la capa de datos, modificar y mantener el código es una tarea sencilla [16].

1.6. Patrones

Los patrones de software no son más que un conjunto de soluciones a problemas habituales en el diseño de software orientado a objetos. Una definición más formal podría ser: “Un patrón es una solución de diseño de software a un problema, aceptada como correcta, a la que se ha dado un nombre y que puede ser aplicada en otros contextos” [17]. Existen diversos tipos de patrones, entre ellos se encuentran los de arquitectura y de diseño. A continuación se explica brevemente en qué consisten.

1.6.1. Patrones de Diseño

Patrones de diseño: son patrones de un nivel de abstracción menor que los patrones de arquitectura. Están por lo tanto más próximos a lo que sería el código fuente final. Su uso no se refleja en la estructura global del sistema [17]. Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o

interfaces. Una característica que tiene es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Patrones GRASP

GRASP son patrones generales de software para asignación de responsabilidades, es el acrónimo de "GRASP (object-oriented design General Responsibility Assignment Software Patterns)". Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. GRASP destaca entre sus patrones principales: Experto, Creador, Controlador, Bajo acoplamiento y Alta cohesión. A continuación se explica brevemente en qué consisten [18]:

- **Experto:** el GRASP de experto en información es el principio básico de asignación de responsabilidades. Indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). Mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida, son más fáciles de entender y mantener.

- **Creador:** el patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. Si se asigna bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

- **Controlador:** un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema, define además el método de su operación. El patrón controlador es un patrón que se utiliza como intermediario entre una determinada interfaz y el algoritmo que la implementa. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, para aumentar la reutilización de código y a la vez tener un mayor control. Permite mayor potencial de los componentes reutilizables, garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz.

- **Bajo acoplamiento:** asigna responsabilidades de modo que se mantenga el bajo acoplamiento. Debe haber pocas dependencias entre las clases, si todas las clases dependen de todas.

- **Alta cohesión:** la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Posibilita mejorar la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y permite una mayor reutilización.

Patrones GOF

Los patrones GOF (conocidos mundialmente por Gang of Four o Pandilla de los cuatro) son patrones de diseño. Los patrones de diseño GOF se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento [19].

- **Singleton** (instancia única): es un patrón creacional que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. A primera vista, uno puede pensar que pueden utilizarse clases con miembros estáticos para el mismo fin. Sin embargo, los resultados no son los mismos, ya que en este caso la responsabilidad de tener una única instancia recae en el cliente de la clase. El patrón Singleton hace que la clase sea responsable de su única instancia, quitando así este problema a los clientes [20].

1.6.2. Patrones arquitectónicos

Patrones de arquitectura: son esquemas fundamentales de organización de un sistema software. Especifican una serie de subsistemas y sus responsabilidades respectivas e incluyen las reglas y criterios para organizar las relaciones existentes entre ellos.

En el desarrollo de la aplicación informática propuesta se utilizará el marco de trabajo Sauxe, el cual está basado en el patrón clásico MVC. Modelo Vista Controlador (MVC) es un patrón o modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista [21].

Modelo: esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones

visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

Vista: la vista presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y probablemente, a la vista.

El flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por él.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente [21].

1.7. Técnicas para la captura y validación de los requisitos

En el proceso de identificación de requisitos se extraen las diferentes fuentes de información, los datos que son necesarios para conocer las funcionalidades que se implementarán en el sistema y se establece una buena comunicación con los interesados del producto.

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

- Entrevistas: es una de las técnicas más usadas en la captura de requisitos. Consiste en establecer una conversación entre personas de ambas partes. Estas son aplicadas a los especialistas funcionales, donde se recoge la información necesaria para el desarrollo de la aplicación.
- Tormenta de ideas (brainstorming): consiste en la reunión de varios interesados en la que todos expresan sus ideas sobre el problema y su solución. La forma de llevarla a cabo es que cada participante diga su idea sin ser interrumpido por otro. Al finalizar la sesión de lluvia de ideas se puede hacer una recolección de ideas sin duplicidad.

Entre las técnicas que se utilizan para la validación de los requisitos se encuentran:

- Revisión técnica: son reuniones del personal técnico (usuario final del sistema) con el objetivo de validar la especificación de requisitos. Su aplicación a los documentos de práctica permitirá detectar deficiencias, ambigüedades, omisiones y errores, tanto de formato como de contenido.
- Prototipos: se muestra un modelo ejecutable del sistema a los usuarios finales y los clientes, para que puedan ver si dicho modelo cumple con sus necesidades reales.

1.8. Lenguajes de modelado

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

1.8.1. Businesses Process Modeling Notation (BPMN)

Es una notación estándar para el modelamiento de los procesos de negocio, la cual permite entender los procedimientos internos a través de una notación grafica (Business Process Diagram-BPD-) permitiendo la comunicación de estos procedimientos en una forma estándar. El modelado de procesos de negocio requiere la representación de los elementos que intervienen en los procesos. Por lo tanto, es necesario utilizar un lenguaje o notación que permita modelar dichos procesos. Esta notación facilita además el entendimiento de las colaboraciones de rendimiento y de transacciones permitiendo la reutilización [22]. BPMN crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos, y permite modelar los procesos de una manera unificada y estandarizada, permitiendo un entendimiento a todas las personas de una organización.

1.8.2. Lenguaje Unificado de Modelado (UML)

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo [23].

1.9. Lenguaje de programación

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático [50]. Permite a un programador especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. A través de estas reglas el programador crea los programas o subprogramas. Los lenguajes de programación para el desarrollo de aplicaciones web, están divididos en dos grandes grupos, los lenguajes del lado del servidor y los lenguajes del lado del cliente.

1.9.1. Lenguaje de programación del lado del Cliente

Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda estar en cualquier sitio.

XML

El Lenguaje de Marcas Extensible (XML), es un lenguaje desarrollado por el World Wide Web Consortium (W3C). A diferencia de otros lenguajes XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información. XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores [24]. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

HTML

El lenguaje de marcado de hipertexto (HTML), fue definido por una organización internacional de estandarización (el Consorcio W3[2]). Hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes [25]. Este lenguaje no describe la apariencia de un documento sino que ofrece a cada plataforma la información para establecer el formato según su capacidad y la de su navegador (tamaño de la pantalla, fuentes que tiene instaladas). Por eso es importante, diseñar los documentos con un contenido claro y bien estructurado que resulte fácil de leer y entender en cualquier navegador. Puede incluir un script (por

[2] El World Wide Web Consortium (W3C): comunidad internacional de desarrollo de estándares para el crecimiento de la Web a largo plazo.

ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

Java Script 1.6

JavaScript se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, en bases de datos locales al navegador. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM) [26]. Es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos.

1.9.2. Lenguaje programación del lado del Servidor

PHP (Hypertext Preprocessor) 5.2

PHP es un lenguaje de programación de uso general de script del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. PHP puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo. Permite la conexión a diferentes tipos de servidores de bases de datos tales como PostgreSQL [27].

1.10. Tecnologías y herramientas de desarrollo

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

1.10.1. Tecnología Ajax

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios

sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones [28]. Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dados que está basado en estándares abiertos como JavaScript y Document Object Model (DOM). Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor.

1.10.2. Servidor de aplicaciones web Apache 2.2

Un servidor web es el programa que, utilizando el protocolo de comunicaciones HTTP, es capaz de recibir peticiones de información de un programa cliente (navegador), recuperar la información solicitada y enviarla al programa cliente para su visualización por el usuario [29].

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, que implementa el protocolo HTTP y la noción de sitio virtual. Tiene como principales ventajas: es modular, presenta código abierto, multi-plataforma, extensible, popular (fácil conseguir ayuda/soporte). Características generales de Apache [30]:

- Es altamente configurable, posee bases de datos de autenticación y negociado de contenido.
- La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente.
- La arquitectura del servidor Apache es modular.

El servidor web **Apache 2.2** proporciona contenidos al cliente web o navegador como:

- Páginas estáticas**:** es el uso más generalizado que se hace de un servidor web. De esta forma se transfieren archivos HTML, imágenes y otros elementos. No se requiere un servidor potente en lo que al hardware se refiere.
- Páginas dinámicas**:** la información que muestran las páginas que sirve Apache cambia ya que se obtiene a partir de consultas a bases de datos u otras fuentes de datos. Son, por tanto, páginas con contenido dinámico, cambiante [31].

1.10.3. Sistemas de Gestores de Base de Datos

Los servidores de bases de datos surgen por la necesidad que tienen las empresas de manejar grandes y complejos volúmenes de datos, al tiempo que requieren compartir la información con un conjunto de clientes de una manera segura.

PostgreSQL 9.1: es un Sistema de Gestión de Base de Dato relacional orientado a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyada por organizaciones comerciales [32]. Entre sus características se tiene [33]:

- Alta concurrencia: mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Altamente Extensible: PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys).
- Permite la creación de vistas, herencia de tablas.

1.10.4. Navegador Web

Mozilla Firefox 3.6 o superior: Mozilla Firefox es un navegador web libre y de código abierto desarrollado para Microsoft Windows, Mac OS X y GNU/Linux coordinado por la Corporación Mozilla y la Fundación Mozilla. Usa el motor Gecko para renderizar páginas web, el cual implementa actuales y futuros estándares web. Es un navegador de código abierto, multiplataforma, basado en el código de base de Mozilla que proporciona una navegación más rápida, segura y eficiente que otros navegadores [34].

1.10.5. Herramienta Case

Existen varias herramientas para el modelado UML, entre ellas las de Ingeniería de Software Asistida por Ordenador (Computer Aided Software Engineering, CASE por sus siglas en inglés), sistemas que intentan proporcionar ayuda automatizada a las actividades del proceso de software. Estas herramientas tienen vital importancia en múltiples aspectos del ciclo de vida del desarrollo del software, como son: la mejora de la productividad en el mantenimiento y desarrollo del software, la mejora del tiempo, el costo de desarrollo y el mantenimiento de los sistemas informáticos, la implementación del código con el diseño dado, el aumento de la calidad del software y documentación o detección de errores [35].

Visual Paradigm 6.4

Visual Paradigm es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Su mayor éxito

consiste en la capacidad de ejecutarse sobre diferentes sistemas operativos que le confiere la característica de ser multiplataforma. Utiliza UML como lenguaje de modelado ofreciendo soluciones de software que permiten a las organizaciones desarrollar las aplicaciones de calidad de forma rápida y barata. Presenta un ambiente gráfico agradable para el usuario.

1.10.6. Entorno de desarrollo integrado (IDE)

IDE: Integrated Development Environment (Entorno de Desarrollo Integrado), constituye un editor de código para depurar y facilitar las diferentes tareas necesarias en el desarrollo de cualquier tipo de aplicación que pueda funcionar con diferentes lenguajes de programación.

NetBeans 7.1: es un entorno de desarrollo integrado libre, es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Esta hecho principalmente para el lenguaje de programación Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Es un proyecto de código abierto de gran éxito con una gran base de usuarios y una comunidad en constante crecimiento [36].

1.10.7. Herramientas de desarrollo colaborativo

Subversion TortoiseSVN 1.6.6: es un sistema de control de versiones usado para que varios desarrolladores puedan trabajar en un mismo proyecto en forma más o menos ordenada. Tiene una arquitectura cliente servidor con controles de concurrencia para cuando varios desarrolladores están trabajando en el mismo archivo y funciona más o menos así. Registra los cambios (revisiones) y los logs que se vayan generando. El cliente de SVN se baja una copia local de alguna revisión (generalmente la última), el desarrollador hace los cambios y los sube al servidor para que estén disponibles para los otros desarrolladores [37].

Principales ventajas que le ofrece **Subversión** sobre **CVS:**

- **Fuerte integración con Apache**: esto permite definir controles de acceso avanzados y navegación vía web para consultar el depósito de archivos, proceso carente en CVS.
- **Transparencia al eliminar y cambiar nombres de archivos**: si ha intentado este último proceso en CVS, seguramente sabrá que requiere intervención manual en el depósito para lograrlo, Subversión contempla esta deficiencia y la corrige con éxito.
- **Copias diferenciales de archivos binarios** : basado en el mismo principio de copias ligeras, Subversión es capaz de mantener un control diferencial sobre cualquier archivo binario del depósito así

reduciendo el consumo de espacio, esto contrastado con CVS que requiere archivar copias completas de un archivo binario cada vez que éste cambia [38].

1.11. Conclusiones del capítulo

En este capítulo se realizó la fundamentación teórica de la investigación, evidenciándose:

- La necesidad de realizar una aplicación que garantice la planificación y control de los cursos de postgrado, debido a que en los resultados del análisis de los diferentes sistemas informáticos similares a la propuesta de solución no fue encontrado ninguno que aportara elementos concretos para la presente investigación.
- Se determinó que el software debe ser elaborado basándose en el Modelo de Desarrollo del CEIGE, utilizando las herramientas y tecnologías mencionadas en el transcurso del capítulo, ya que son las definidas por el marco de trabajo Sauxe.
- El estudio de los patrones posibilitó sustentar la base para futuras búsquedas de soluciones a problemas comunes en el desarrollo de software.

CAPÍTULO II. PROPUESTA DE SOLUCIÓN

2.1. Introducción

En el presente capítulo se realizará la descripción de la propuesta para dar solución a la problemática existente. Se efectuará el análisis y diseño del sistema donde se generan los artefactos siguientes: modelo conceptual, mapa de procesos del negocio, diagramas de clase del diseño, modelo de datos, diagramas de secuencia y de despliegue. Además se identificarán los requisitos funcionales y no funcionales que deben satisfacer la solución a desarrollar, dando paso a la etapa de diseño donde se especificará la utilización de diferentes patrones. Por último se establecerán los estándares de codificación a utilizar que permitirán una mejor organización del trabajo.

2.2. Propuesta del sistema

El sistema será capaz de realizar los procesos de gestión de la planificación de las acciones de CTI y postgrado en el centro CEIGE y realizar el control de la planificación, permitiendo verificar la aprobación del plan de CTI. Para ello el usuario deberá autenticarse previamente en la aplicación y el sistema verificará sus privilegios, permitiéndole solo el acceso a las funcionalidades y acciones que le están permitidas según su rol. Además debe permitirle al usuario gestionar las actividades que le son asignadas así como realizar cambios en las mismas una vez analizadas con sus superiores, posibilitando realizar una evaluación del porciento de tareas cumplidas y un análisis del plan establecido en el trimestre. Por tanto, la aplicación debe permitir adicionar, modificar, eliminar, listar y buscar las actividades de los profesionales, cambiar el estado de las mismas, realizar el seguimiento y control de las tareas, además de la generación de reportes. La utilización del marco de trabajo Sauxe posibilitará en nuevas versiones la integración del sistema al ERP Cedrux.

2.3. Modelación de los procesos de negocio

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido**.** El modelado de procesos de negocio es la base para comprender mejor la operación de una organización, documentar y publicar los procesos buscando una estandarización en la organización, buscar eficiencias en la operación e integrar soluciones en arquitecturas orientadas a servicios. Los procesos de negocio son la base para comprender mejor la forma en que opera un negocio en sus diferentes áreas y son una herramienta fundamental para acceder a modelos de calidad [39]. Permite comprender las características del negocio a través de la descripción de los procesos.

2.3.1. Mapa de procesos del negocio

El mapa de procesos de negocio ofrece una visión general del sistema de gestión. En él se representan los procesos que componen el sistema así como sus relaciones principales. Dichas relaciones se indican mediante flechas y registros que representan los flujos de información. Los procesos describen como se realiza el trabajo en la organización caracterizándose por ser observables, medibles, mejorables y repetitivos [48].

Para comprender los procesos del negocio se realizó un previo estudio de los sistemas de planificación de actividades de postgrado analizados en el estado del arte. El mapa de proceso para el sistema informático para la gestión de la planificación de las acciones de CTI y postgrado abarca dos procesos: Realizar planificación de las acciones de CTI y postgrado y el proceso Realizar el control de la planificación.

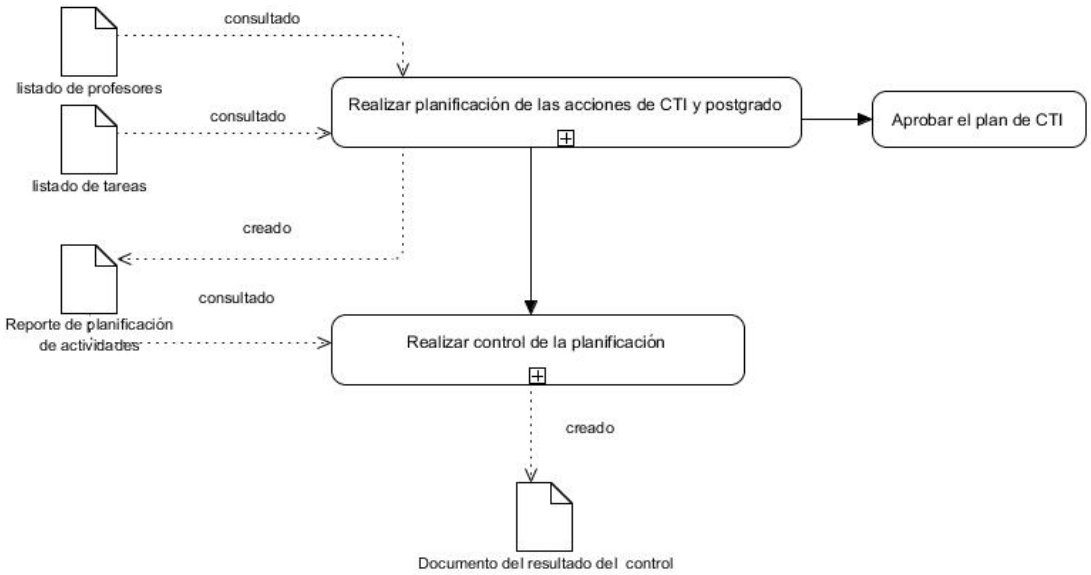


Figura 3: Mapa de procesos del negocio.

2.3.2. Diagramas de procesos del negocio

El diagrama de procesos del negocio proporciona facilidades para determinar las personas que forman parte de un proceso donde se limita la actividad y responsabilidad de cada individuo; lograr la estandarización correcta de la ejecución de los procesos garantiza que se alcancen los objetivos trazados.

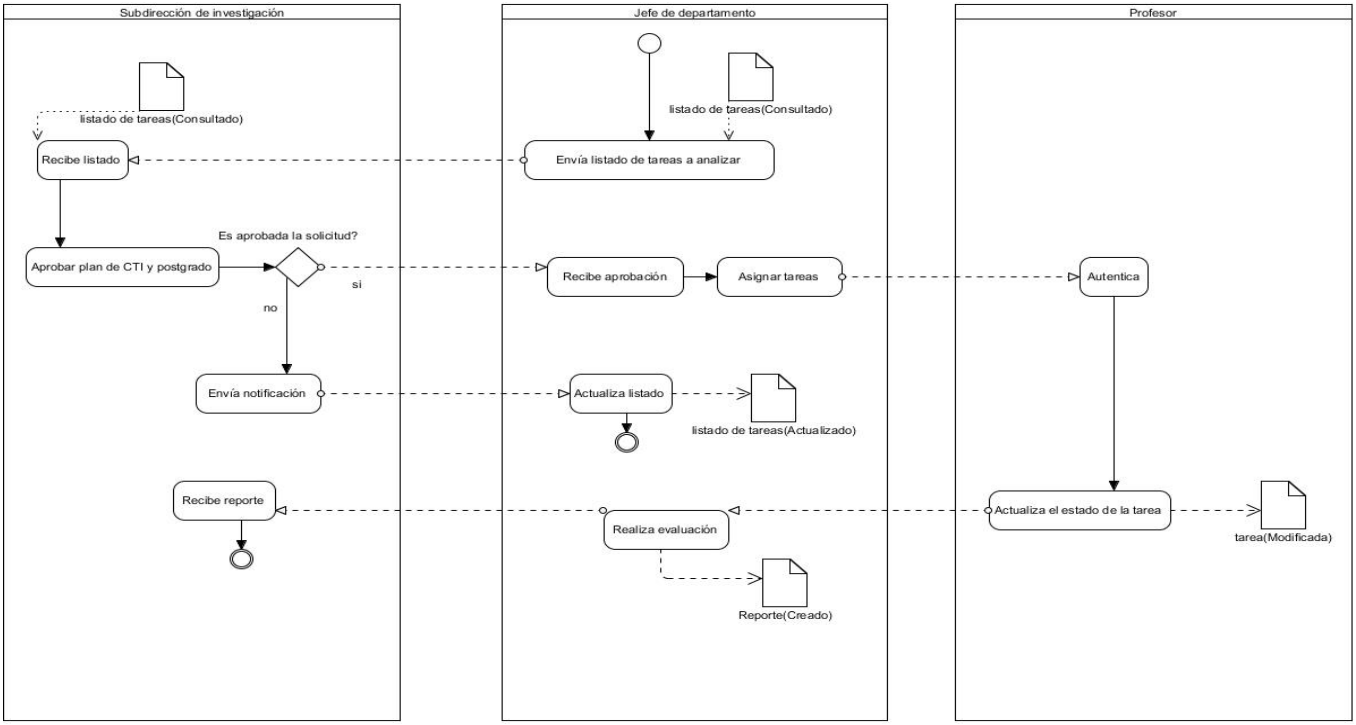


Figura 4: Diagrama de procesos del negocio Realizar planificación de las acciones de CTI y postgrado.

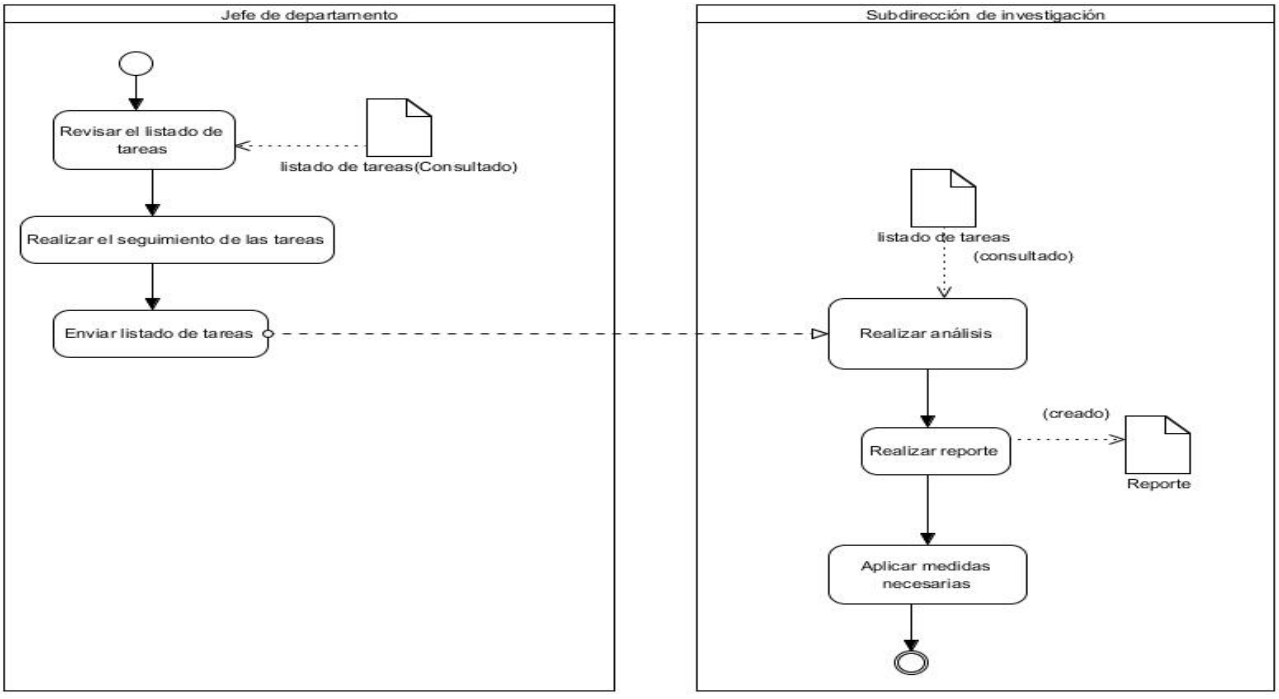


Figura 5: Diagrama de procesos del negocio Realizar control de la planificación.

2.3.3. Descripción del proceso de negocio Realizar planificación de las acciones de CTI y postgrado.

Objetivo	Almacenar correctamente la información de las actividades asignadas a los profesionales para luego planificar y darle seguimiento a las tareas orientadas en los cursos de postgrado y acciones de CTI.
Evento(s) que lo genera(n)	N/A.
Pre condiciones	Debe existir el listado de profesores. Debe existir el listado de tareas.
Marco legal	N/A.
Clientes internos	Profesores.
Entradas	Listado de profesores. Listado de tareas.

Flujo de eventos	
Flujo básico Realizar planificación de las acciones de CTI y postgrado.	
1	Enviar listado de tareas a analizar: antes de asignar una tarea se le debe notificar a la subdirección de postgrado el listado de actividades a orientar.
2	Recibe listado: el listado es recogido por el subdirector de investigación y postgrado para su análisis.
3	Aprobar plan de CTI y postgrado: una vez aprobadas las actividades de cada una de las personas se puede adicionar una nueva tarea.
4	Recibe aprobación: el departamento de investigación envía la confirmación de aprobación al jefe de departamento.
5	Asignar tareas: el jefe de departamento debe asignar a los profesionales las actividades orientadas a cumplir.
6	Autenticar usuario: los usuarios deberán autenticarse para poder acceder a los datos y al sistema.
7	Actualizar tarea: al culminar la actividad esta se debe actualizar y cambiar su estado.
8	Realizar evaluación: se debe analizar el porciento de tareas cumplidas y dar una evaluación final.
9	Enviar reporte: luego de haber confirmado los datos pertinentes, se le debe enviar a la subdirección de investigación los resultados alcanzados.

Pos-condiciones	
1	Se obtiene un reporte con los resultados alcanzados, cantidad de tareas cumplidas y rechazadas.
Salidas	
1 Reporte impreso con el listado de resultados de cada área.	
Flujos paralelos	
Salidas	
Flujos alternos Aprobar plan de CTI y postgrado	
Flujo alternativo 1ª.	

1 En caso de ser denegada alguna de las actividades orientadas a los profesionales se le informa a la persona.
2 Actualizar listado de actividades aprobadas
N/A
Salidas
N/A
Asuntos pendientes
N/A

Tabla 1: Descripción del proceso de negocio Realizar planificación de las acciones de CTI y postgrado.

2.3.4. Descripción del proceso de negocio Realizar control de la planificación.

Objetivo	Durante el proceso de planificación de las actividades, se realiza un control para verificar el cumplimiento de las tareas asignadas a los profesionales.
Evento(s) que lo genera(n)	N/A.
Pre condiciones	Debe existir el listado de tareas.
Marco legal	N/A.
Clientes internos	Profesores.
Entradas	Listado de tareas.
Flujo de eventos	
Flujo básico Control de la Planificación	
	Revisar el listado de tareas: verificar que las actividades asignadas estén actualizadas y los datos sean correctos.
	Realizar el seguimiento de las tareas: se debe velar porque el plan trazado para cada uno de los implicados se cumpla en el tiempo establecido.
	Realizar análisis: se realiza un análisis de las actividades por cada uno de los involucrados.
	Realizar el reporte: se debe generar un reporte que muestre un resumen del porciento asignado y el plan real logrado.
	Aplicar medidas necesarias: la subdirección de investigación realiza un control para revisar los profesionales que hayan incumplido y luego toma las medidas pertinentes.
Pos-condiciones	
	Se obtiene un reporte con el resultado de cada área, usuario y tipo de actividad.
Salidas	Reporte impreso con el listado de resultados.
Flujos paralelos	
Salidas	

Flujos alternos
1 En caso de ser rechazada la actividad se debe realizar un análisis de la misma.
Salidas
N/A
Asuntos pendientes
N/A.

Tabla 2: Descripción del proceso de negocio Realizar control de la planificación.

2.3.4. Modelo conceptual

El Modelo Conceptual explica los conceptos significativos en el dominio del problema. Muestra: conceptos, asociaciones entre conceptos y atributos de conceptos. Una cualidad esencial que debe ofrecer un Modelo Conceptual es que debe representar cosas del mundo real [40].

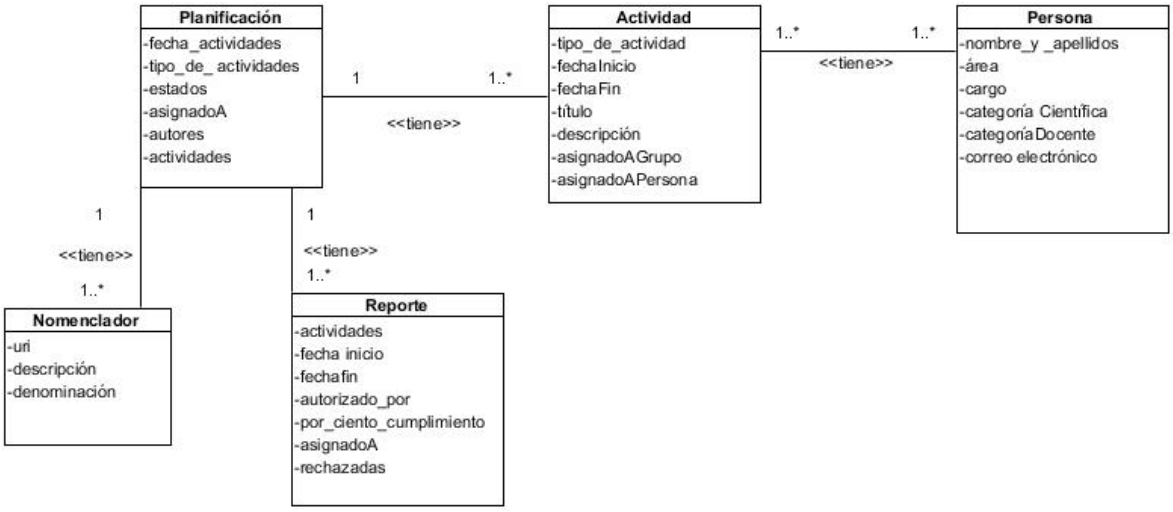


Figura 6: Modelo conceptual.

En el modelo conceptual se puede apreciar la relación que existe entre la clase planificación y las actividades que les son asignadas a los profesionales. Contiene los nomencladores (tipos de actividades, áreas, categoría docente, categoría científica, cargos) e incluye la generación de reportes. El significado de muchos de estos conceptos se puede encontrar en el glosario de términos y en los conceptos asociados al dominio del problema.

2.4. Requisitos

Un requerimiento es la condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. Por lo que se puede decir que los requerimientos deben ser especificados por escrito, claros y

precisos. Estos se clasifican en funcionales y no funcionales. Los requisitos funcionales son capacidades o condiciones que el sistema debe tener, mientras que los no funcionales son propiedades o cualidades que el producto debe tener. Los procesos realizar la planificación de las acciones de CTI y realizar el control de la planificación contienen los siguientes requisitos funcionales para su ejecución:

El sistema debe posibilitar realizar las siguientes funcionalidades sobre las actividades que le son asignadas a los profesionales: (Gestionar actividades)

- **RF 1:** Adicionar actividades.
- **RF 2:** Eliminar actividades.
- **RF 3:** Modificar actividades.
- **RF 4:** Listar actividades.
- **RF 5:** Asociar actividades a personas.
- **RF 6:** Asociar actividades a áreas.

Para realizar el seguimiento y control de las actividades debe permitir:

- **RF 7:** Cambiar estado de cumplimiento de las actividades.
- **RF 8:** Rechazar una actividad.

El sistema debe posibilitar realizar las siguientes funcionalidades sobre los nomencladores, los cuales pueden ser: tipos de actividades, áreas, cargos, categoría docente, categoría científica: (Gestionar tipo de nomencladores)

- **RF 9:** Adicionar tipo de nomenclador.
- **RF 10:** Eliminar tipo de nomenclador.
- **RF 11:** Modificar tipo de nomenclador.
- **RF 12:** Adicionar datos de los nomencladores.
- **RF 13:** Modificar datos de los nomencladores.
- **RF 14:** Eliminar datos de los nomencladores.

El sistema debe posibilitar realizar las siguientes funcionalidades para realizar la caracterización de las personas, pues esto permite un mejor acceso a los datos del sistema según los permisos establecidos: (Gestionar persona)

- **RF 15:** Definir jefe de centro.
- **RF 16:** Definir jefe de área.
- **RF 17:** Definir profesionales por área.
- **RF 18:** Adicionar persona.

- **RF 19:** Eliminar persona.
- **RF 20:** Modificar datos de la persona.
- **RF 21:** Listar persona.

El sistema debe permitir la generación de los reportes (Mostrar reportes)

- **RF 22:** Mostrar reporte por área.
- **RF 23:** Mostrar reporte por personas.
- **RF 24:** Mostrar reporte por tipo de actividad.
- **RF 25:** Exportar información de reportes a Excel.

El sistema permitirá la búsqueda tanto de actividades, personas o reportes por cada uno de sus atributos según convenga.

- **RF 26:** Filtrar por atributos.

2.4.1. Especificación de requisitos funcionales

El objetivo de la especificación de requisitos funcionales es obtener una clara comprensión del problema a resolver, extraer las necesidades del usuario y derivar de ellas las funciones que debe realizar el sistema. A continuación se muestra la descripción del requisito funcional Adicionar actividades, el resto de las descripciones de los requisitos funcionales pueden consultarse en el **anexo 1**.

Especificación del requisito Adicionar Actividades

Precondiciones	Debe haberse autenticado antes en el sistema y tener permiso para realizar esta acción.
Flujo de eventos	
Flujo básico	
1	Selecciona el botón “**Adicionar**”.
2	Se levanta una ventana con los datos a introducir de la actividad: Fecha inicio Fecha fin Tipo de actividad Descripción Denominación
3	Selecciona el botón “**Aceptar**”.
4	El sistema valida (ver validación 1) los datos introducidos.
5	Si los datos son correctos el sistema los registra.
6	El sistema confirma el registro de los datos.
7	Concluye el requisito.
Pos-condiciones	
1	Se registró en el sistema una nueva actividad.
Flujos alternativos	

Flujo alternativo 3.a Información errónea		
1	El sistema señala los datos erróneos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 2 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo 3.b Información incompleta		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 2 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se registran los datos.	
Validaciones		
Relaciones	**Requisitos Incluidos**	N/A.
	Extensiones	N/A.
Conceptos	**Actividad**	Visibles en la interfaz: Fecha inicio Fecha fin Tipo de actividad Descripción Denominación Utilizados internamente: N/A.
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

Tabla 3: Especificación del requisito Adicionar Actividades.

2.4.2. Prototipo de interfaz de usuario.

A continuación se muestra el prototipo de interfaz de usuario funcional del requisito Adicionar actividades que se obtuvo como resultado del diseño del sistema. El resto de los prototipos de interfaz de usuario se encuentran en el **anexo 5**.

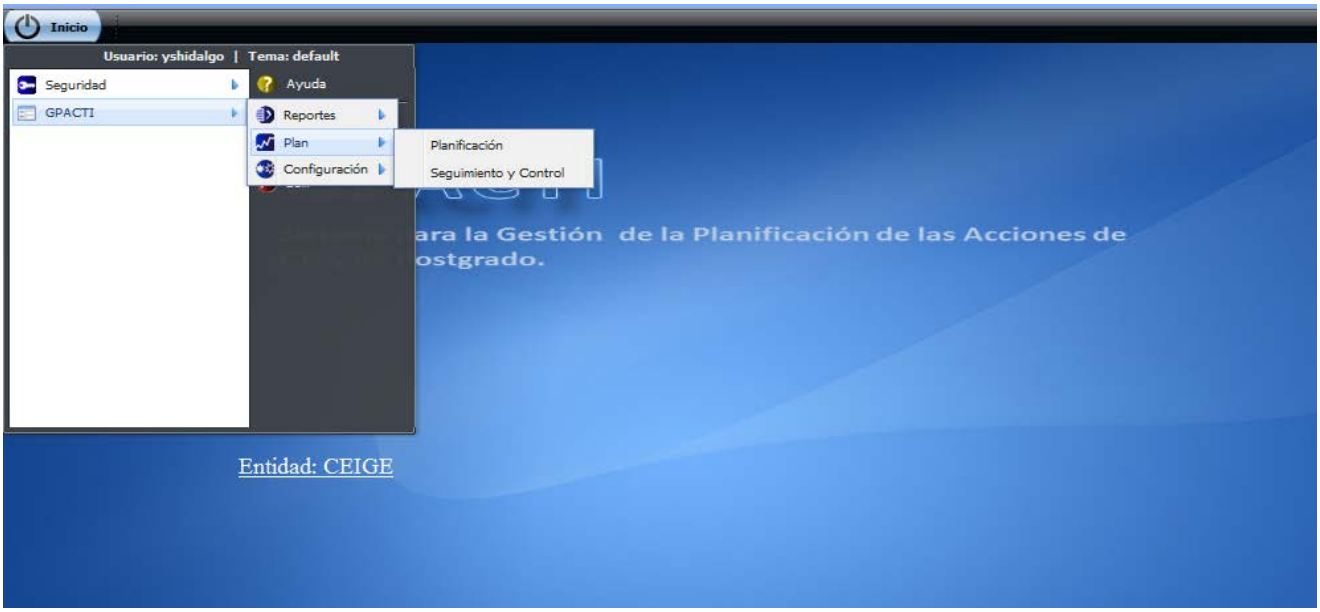


Figura 7: Prototipo de interfaz de usuario Principal.

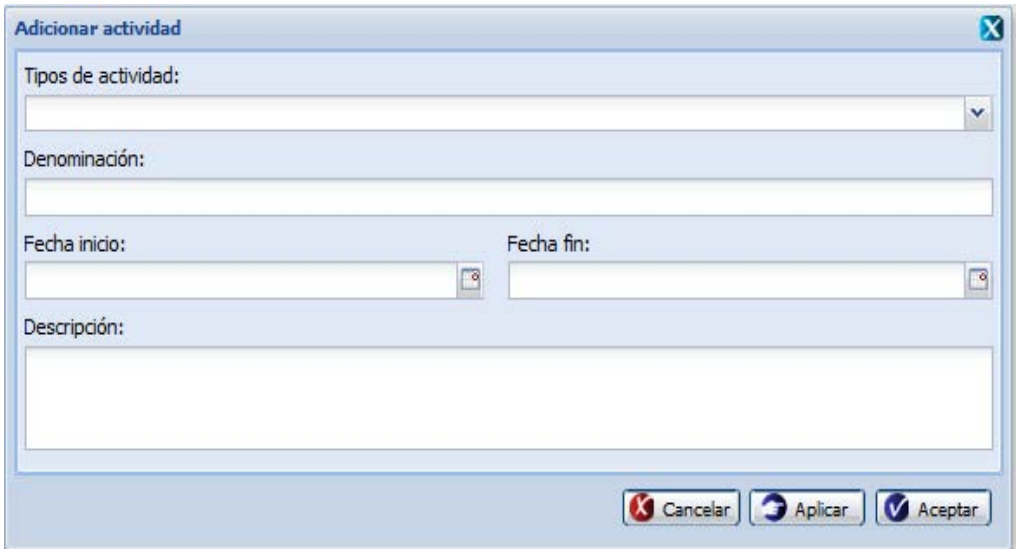


Figura 8: Prototipo de interfaz de usuario del Adicionar Actividades.

2.4.3. Requisitos no funcionales

Los requerimientos no funcionales son las propiedades o cualidades que el producto debe tener. Deben pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable [41]. Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. Estos surgen de la necesidad del usuario, debido a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con otros sistemas de software o hardware o a factores externos como los reglamentos de seguridad, las políticas de privacidad, entre otros [42].

Apariencia o interfaz externa

La aplicación deberá desarrollarse en un ambiente web. Debiendo presentar una interfaz sencilla, amigable y de fácil uso, así como poseer un diseño orientado a llamar la atención del usuario con una navegación sencilla y construcción de enlaces rápidos. Además, debe combinar correctamente los colores, tipo de letra, tamaño y los íconos deben estar en correspondencia con lo que representan.

Rendimiento

El sistema deberá tener una respuesta rápida para la interacción con el usuario y consumir la mínima cantidad de recursos. Además, el software debe ser capaz de funcionar en una pc que contenga todos los componentes que lo conforman tales como gestor de base de datos y aplicación web.

Seguridad

Confidencialidad: al sistema deberán tener acceso solo el personal autorizado, con los permisos correspondientes, por lo que será necesaria una autenticación previa.

Integridad: el software deberá estar protegido contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Disponibilidad: al sistema deberá ser posible acceder las 24 horas del día para todos los usuarios autorizados.

Portabilidad

La herramienta desarrollada deberá ser multiplataforma (Linux o Windows).

Hardware

- Para el cliente los requerimientos mínimos deben ser de un procesador Pentium IV a 800 MHz, 512 MB de memoria RAM y un navegador web. Las computadoras clientes deben estar conectadas en la red.

- Para el servidor los requerimientos mínimos deben ser un procesador Dual Core a 3.00 GHz, 1gb de memoria RAM y una capacidad de 60gb de disco duro. El servidor debe tener al menos 1 tarjeta de red para establecer la conexión.

Software

Cliente: sistema operativo con interfaz gráfica y soporte para red. Las interfaces deben ser compatibles con Mozilla Firefox 3.0 o superior.
Servidor: servidor web Apache 2.2 o superior y gestor de base de datos PostgreSQL 9.1.

Usabilidad

El sistema podrá ser utilizado de forma fácil por cualquier persona, con conocimientos básicos de computación. Las operaciones de la aplicación a informatizar serán lo más parecidas posible a los procesos que se realizan actualmente en la universidad, para así lograr menor tiempo en cuanto a la comprensión y adaptación del sistema.

2.4.4. Aplicación de técnicas de captura y validación de requisitos

Para llevar a cabo la captura de los requisitos necesarios del sistema se le realizó una entrevista al subdirector de investigación y postgrado del CEIGE, la cual se puede ver en los anexos. Varias personas aportaron ideas en cuanto al diseño del software y los requisitos no funcionales que debía cumplir llevando a cabo una tormenta de ideas, para lograr un sistema con una interfaz amigable y fácil de usar. Los requisitos una vez definidos necesitaban ser validados, entre las técnicas que se utilizaron para la validación de los requisitos identificados se encuentran:

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

Se presentaron los prototipos elaborados durante la especificación, para corroborar su correspondencia con las necesidades y aspiraciones del cliente. De forma tal que se visualizaran las diferentes funcionalidades que tendría el sistema.

2.5. Diagrama de clases del diseño

Los diagramas de clases describen gráficamente las especificaciones de las clases de software y expresan la definición de clases como componentes del software. Representan una abstracción del dominio de modo que es formalizado el análisis de conceptos y constituyen el pilar básico del modelado, mostrando en términos generales qué debe hacer el sistema.

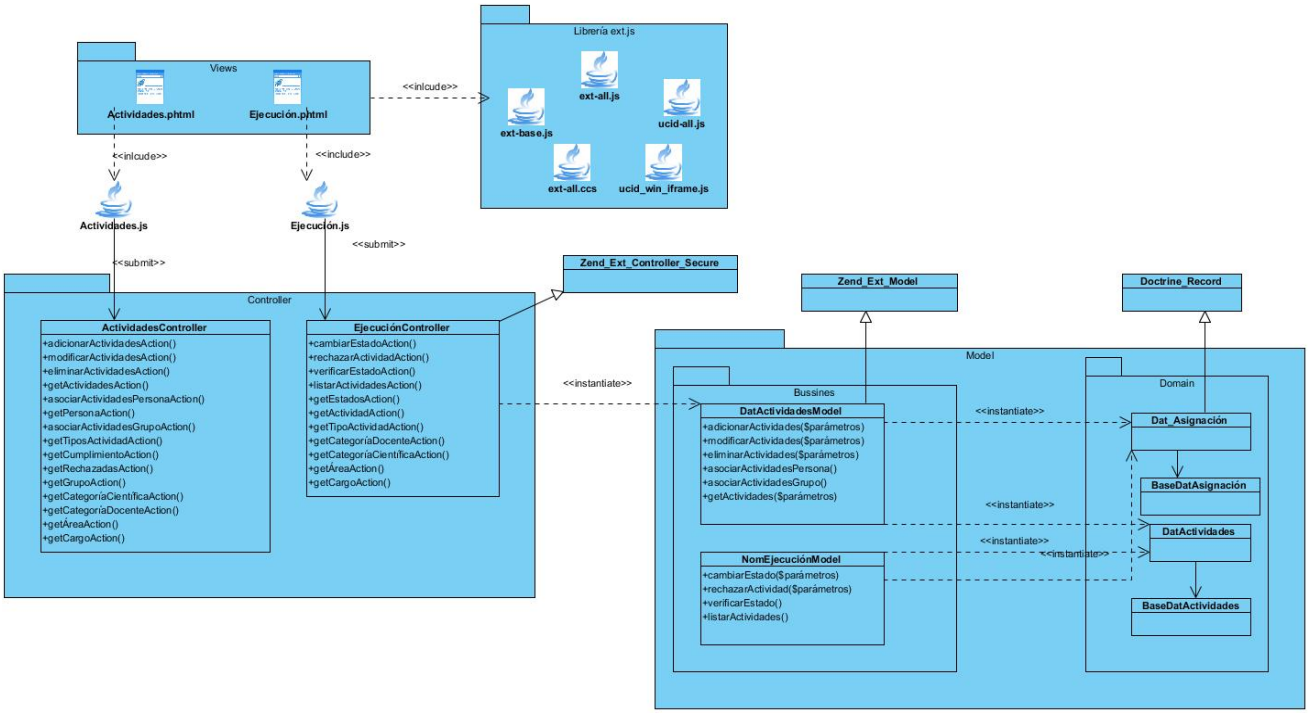


Figura 9: Diagrama de clases del diseño del componente Plan.

Se representan las clases que forman parte de la capa lógica del negocio y de la cual se nutren las clases controladoras. El paquete Model maneja los datos persistentes dentro del componente y contiene el Domain y el Bussines, la cual hereda de ZendExt_Model para gestionar la seguridad de los datos persistentes ubicados en el Model. El resto de los diagramas de clases se encuentran en el **anexo 3**.

2.5.1. Aplicación de los patrones de diseño

Patrones GRASP

Experto: se emplea al trabajar con el marco de trabajo Sauxe y un ejemplo de su uso está en la clase (class ActividadesController extends ZendExt_Controller_Secure), debido a que cada clase cuenta con un grupo de funcionalidades que las convierte en experta de la información.

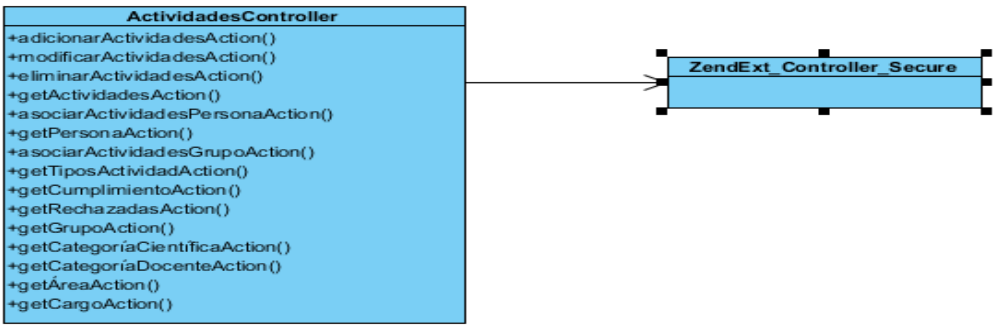
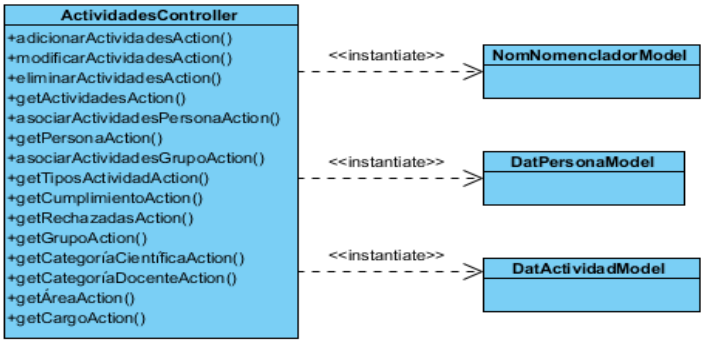


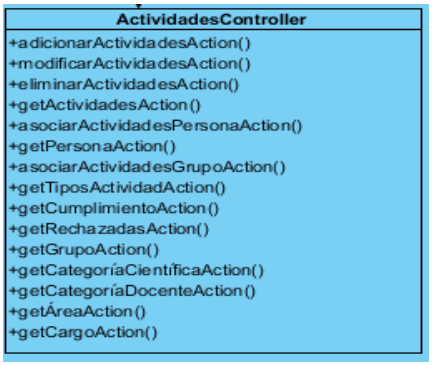
Figura 10: Ejemplo del patrón Experto.

Creador: este patrón se evidencia en la clase controladora ActividadesController, ya que es la encargada de la creación de objetos de varias clases como NomNomencladorModel, DatPersonaModel y DatActividadModel. La clase controladora es la encargada de instanciar en cada uno de los objetos del modelo que será utilizado para almacenar la información para presentarla a la vista posteriormente.



Figuras 11: Ejemplo del patrón Creador.

Controlador: la clase controladora ActividadesController es la encargada de llevar el control de todos los eventos relacionados con el componente Planificación. Implementa las funcionalidades que dan respuesta a las peticiones del usuario.



Figuras 12: Ejemplo del patrón Controlador.

Alta cohesión: el uso de este patrón indica que la información almacenada en las clases debe ser coherente y relacionada con los datos manejados en ellas. La utilización del patrón se evidencia igualmente en la clase controladora ActividadesController, ya que presenta responsabilidades y colabora con otras clases para dar solución a las tareas que surgen en el flujo del negocio.

Patrones GOF

Patrón creacional Singleton: se puede ver su aplicación en el uso del patrón de Inversión de Control (IoC[3]) utilizado por Sauxe, en el que se encuentran los servicios a consumir dentro de los componentes que integran al marco de trabajo, donde se garantiza que cada clase tenga una instancia única y permite que la misma sea accedida desde cualquier parte del sistema, en la solución no se va a implementar el Singlenton pero se consumen servicios del IoC como se puede apreciar en el siguiente ejemplo.

```
class PersonaController extends ZendExt_Controller_Secure
{
    public function getUsuariosAction()
    {
        $parametros['usuarios'] = $this->integrator->seguridad->UsuariosPorRol();
        $parametros['limit'] = $this->_request->getPost('limit');
        $parametros['start'] = $this->_request->getPost('start');
        $filtro = $this->_request->getPost('filtros');
```

Figuras 13: Ejemplo de aplicación del patrón Singleton.

[3] Inversión de control (Inversion of Control en inglés, IoC): es un patrón de diseño pensado para permitir un menor acoplamiento entre componentes de una aplicación y fomentar así el reuso de los mismos.

2.6. Diagrama de componentes

Un diagrama de componentes representa los componentes, sus interfaces y las relaciones de los componentes con las interfaces que utilizan [43].

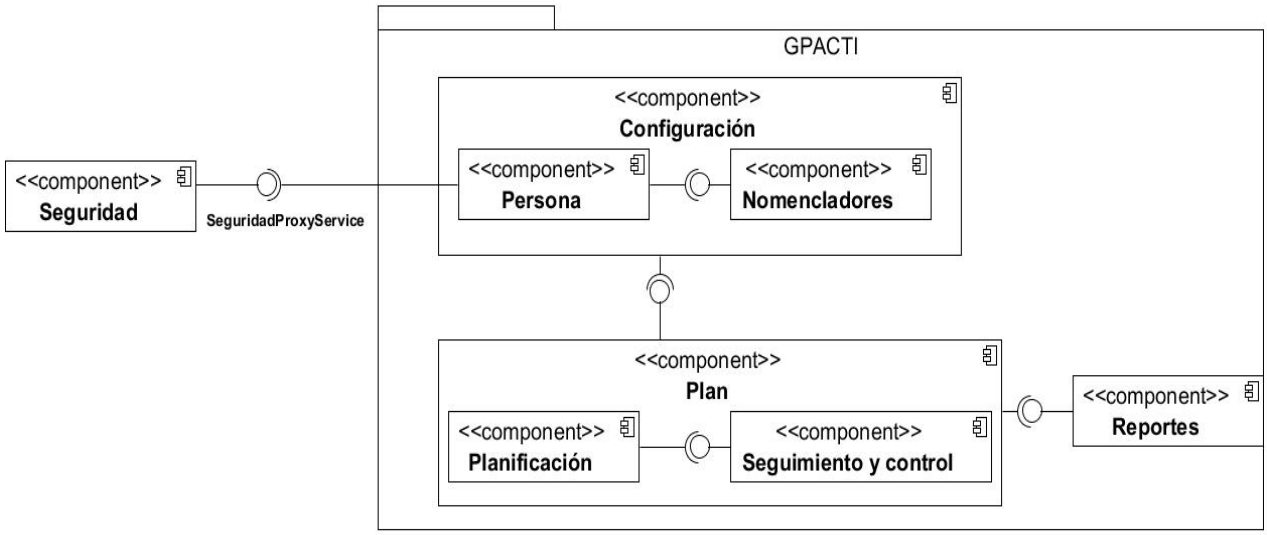


Figura 14: Diagrama de componentes.

El sistema contiene los componentes Configuración, Plan y Reportes los cuales integran los componentes encargados de realizar las funcionalidades identificadas en la captura de requisitos. El mismo hace uso de servicios internos mediante el IoC para obtener información de otros subsistemas del marco de trabajo, entre ellos los que brinda Seguridad. Se realizará una explicación de los mismos de manera general:

Configuración: es el encargado de la gestión de las personas que se encuentran registradas en el sistema, permite adicionar una persona siempre que sea usuario de la misma, definir los cargos (profesional, jefe de área, jefe de centro), modificar y eliminar información. Además este componente posibilita la creación y asignación de los datos a los distintos tipos de nomencladores que se crean.

Plan: es el encargado de la gestión y asignación de actividades al personal, garantizando que solo los jefes de áreas y jefe de centro puedan asociar actividades. Permite realizar el seguimiento y control de las mismas a través de los estados y por ciento de cumplimiento de estas.

Reportes: se encarga de mostrar los resultados obtenidos durante la planificación teniendo en cuenta el estado de cumplimento de las actividades asignadas por áreas, persona y tipo de actividad.

Seguridad: es un componente del marco de trabajo Sauxe que se reutilizará para limitar el acceso a la información, es el encargado de mantener la seguridad en todo el sistema mediante la comprobación de las peticiones y sus permisos en dependencia del usuario que la realice, de aquí la necesidad de que exista una comunicación con los demás componentes. Permitirá la gestión de usuarios definiendo por cada uno de ellos la autenticación dentro del sistema GPACTI.

2.7. Diagrama de secuencia

Otros de los artefactos generados durante el diseño del sistema fueron los diagramas de secuencia correspondientes a cada uno de los requisitos funcionales identificados, estos constituyen una forma efectiva para modelar la interacción entre los diferentes objetos del sistema, mostrando gráficamente las interacciones del actor y de las operaciones a que dan origen. A continuación se muestra el diagrama de secuencia correspondiente al requisito Adicionar actividades, el resto de los diagramas se pueden ver en el **anexo 4**.

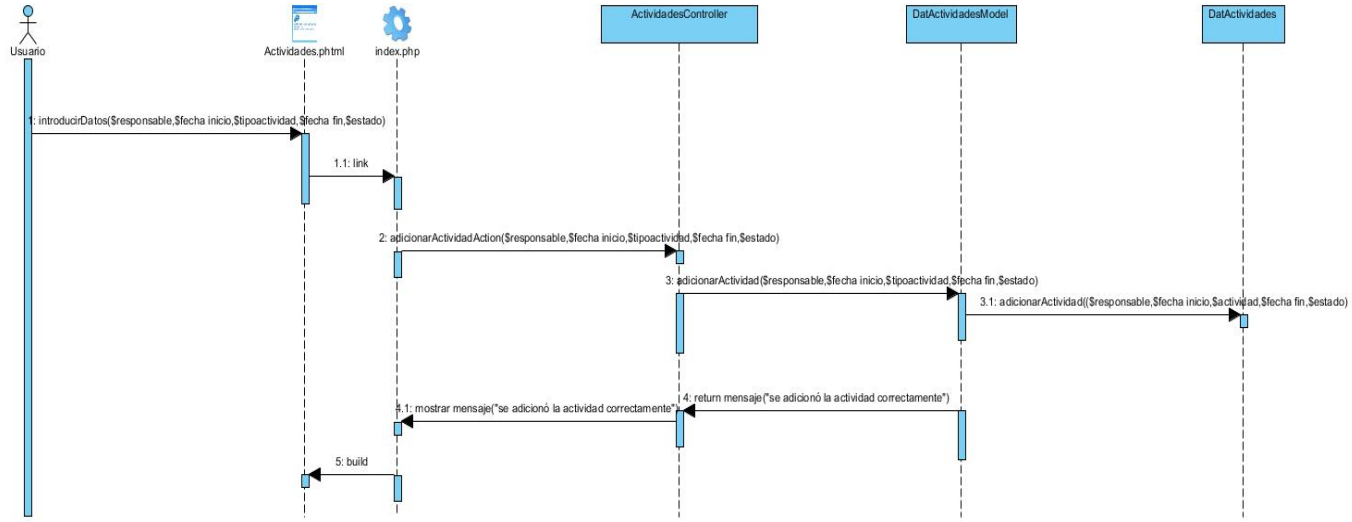


Figura 15: Diagrama de secuencia del requisito Adicionar actividades.

2.8. Modelo de datos

Un modelo de datos es la descripción de una base de datos. Típicamente un modelo de datos permite describir las estructuras de datos de la base, su tipo, descripción y la forma en que se relacionan, restricciones de integridad entre otros, es factible pensar que un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí.

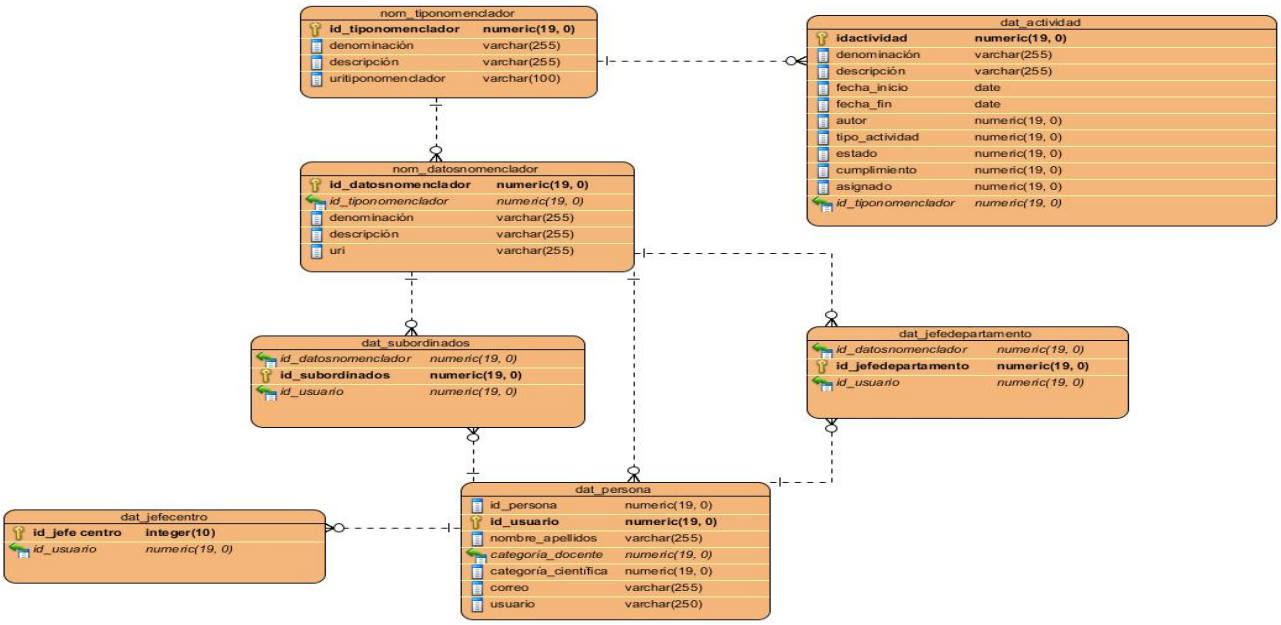


Figura 16: Modelo de datos.

2.9. Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos y las instancias de los componentes y objetos que residen en ellos. Está compuesto por nodos, dispositivos y conectores. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. Permite el mapeo de procesos dentro de los nodos, asegurando la distribución del comportamiento a través de aquellos nodos que son representados [44].

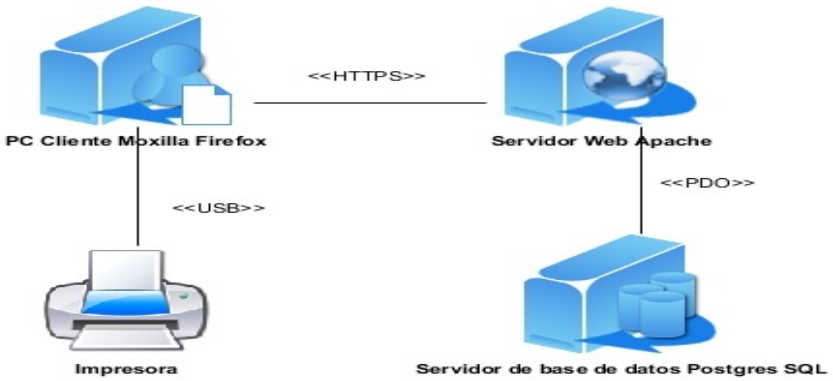


Figura 17: Diagrama de despliegue.

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

Servidor Web: radica la lógica de negocio de la aplicación. Servidor Web Apache 2.2, utilizando el lenguaje de programación del lado del servidor PHP 5.2.

Servidor de Base de datos: servidor de Datos PostgreSQL 9.1, donde se encuentra la base de datos que utiliza el sistema, este puede estar instalado en la misma computadora donde se encuentra el servidor Web.

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

2.10. Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Los estándares de codificación en el marco del ERP establecerán las pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenibilidad a lo largo del tiempo.

2.10.1. Estándares de Nomenclatura

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Con sólo leerlo se reconoce el propósito de la misma.

2.10.2. Nomenclatura según el tipo de clases

Tipo de clase	Ejemplo de Nomenclatura	Especificaciones
Controller (Controladora)	ActividadesController.	Las clases controladoras después del nombre llevan la palabra: “Controller”.
Business (Negocio)	DatActividadesModel	Las clases que se encuentran dentro de Business después del nombre llevan la palabra: “Model”.
Domain (Dominio)	DatActividad	Las clases que se encuentran dentro de Domain reciben el nombre de su tabla correspondiente en la Base de Datos.

Generated (Dominio bases)	BaseDatActividad	El nombre de las clases que se encuentran dentro de Generated comienza con la palabra: “Base” y seguido el nombre de la tabla correspondiente en la Base de Datos.

Tabla 4: Ejemplo de nomenclaturas según el tipo de clases.

2.10.3. Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing* y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: adicionarActividades.

En caso de ser una acción de la clase controladora se le pone el nombre y seguida la palabra:”Action”.

Ejemplo: adicionarActividadesAction.

2.10.4. Normas de comentariado

Es una necesidad comentar todo lo que se haga dentro del desarrollo, es decir, establecer las pautas que conlleven a lograr un código más legible y reutilizable, de manera que se pueda aumentar su mantenibilidad a lo largo del tiempo. Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

2.11. Conclusiones del capítulo

Con el desarrollo del presente capítulo se realizó el análisis y diseño de la solución propuesta generando los siguientes artefactos:

- Modelación del negocio con el objetivo de identificar sus procesos.
- Requisitos funcionales y no funcionales que satisfacen las necesidades del cliente, siendo el punto de partida para realizar el diseño de la propuesta de solución.
- Descripción de los patrones de diseño utilizados durante el desarrollo del sistema, lo que permitió la comprensión y organización del software.
- Confección de los diferentes diagramas como: el modelo conceptual, el mapa de procesos del negocio, los diagramas de clases del diseño, el modelo de datos, los diagramas de secuencia y de despliegue permitiendo mostrar las relaciones e interacciones entre clases.
- Definición de políticas, normas y estándares de codificación para un mejor mantenimiento del sistema a largo plazo.

CAPÍTULO III. IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA.

3.1. Introducción

En este capítulo se describen algunas clases y funcionalidades que conforman el sistema. Se evalúa el diseño empleando las métricas de software Tamaño operacional de clase (TOC) y Relaciones entre Clases (RC), las cuales proporcionan una medida de la complejidad y calidad del software. Se realizan pruebas de caja negra entre ellas las funcionales y las de aceptación, con el objetivo de verificar la funcionalidad y estructura de cada componente desarrollado, lo que posibilita la validación de la solución informática.

3.2. Seguridad del sistema

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

3.3. Descripción de las clases y funcionalidades del sistema

A continuación se describen algunas clases y sus operaciones más importantes, el resto de las descripciones se pueden ver en el **anexo 2.**

Clase Controladora:

Nombre: ActividadesController	
Tipo de clase: Controladora	
Para cada funcionalidad:	
Nombre	getActividadesAction ()
Descripción	Devuelve todas las actividades que existen en el sistema.
Nombre	addActividadesAction()

Descripción	Permite adicionar una actividad en el sistema.
Nombre	getTipoActividadesAction ()
Descripción	Devuelve los tipos de actividades que existen.
Nombre	modificarActividadesAction()
Descripción	Permite modificar cada una de las actividades creadas en el sistema.
Nombre	eliminarActividadesAction()
Descripción	Permite eliminar la actividad seleccionada.
Nombre	asociarActividadesPersonaAction()
Descripción	Permite asignar actividades a la persona seleccionada.
Nombre	asociarAÁreasAction()
Descripción	Permite asociar actividades a las áreas seleccionadas.
Nombre	getPersonasAction()
Descripción	Devuelve todas las personas que se han adicionado al sistema.
Nombre	getEstadosAction()
Descripción	Devuelve los estados de las actividades.
Nombre	getGrupoAction()
Descripción	Devuelve los grupos de personas que existen.
Nombre	getCategoríaCientíficaAction()
Descripción	Devuelve la categoría científica de la persona a la cual se le asignó la actividad.
Nombre	getCategoríaDocenteAction()
Descripción	Devuelve la categoría docente de la persona a la cual se le asignó la actividad.
Nombre	getCargoAction()
Descripción	Devuelve el cargo de la persona a la cual se le asignó la actividad.

Nombre	getÁreaAction()
Descripción	Devuelve el área de la persona a la cual se le asignó la actividad
Nombre	getCambiarEstadosAction()
Descripción	Devuelve los estados que han sido cambiados.
Nombre	getÁreaCargoAction()
Descripción	Devuelve las áreas de la personas según el cargo que desempeña el usuario autenticado.
Nombre	cambiarEstadoActividadAction()
Descripción	Permite cambiar el estado de las actividades.

Tabla 5: Descripción de la clase ActividadesController.

Clase del Modelo:

Nombre: DatActividadesModel.	
Tipo de clase: Modelo	
Para cada funcionalidad:	
Nombre	adicionarActividades($parámetros)
Descripción	Permite adicionar las actividades al sistema según los parámetros establecidos.
Nombre	modificarActividades($parámetros)
Descripción	Permite modificar las actividades del sistema según los parámetros establecidos.
Nombre	eliminarActividades($parámetros)
Descripción	Permite eliminar las actividades según los parámetros establecidos.
Nombre	AsociarActividadesÁrea ($parámetros)

Descripción	Permite asignar actividades a las áreas seleccionadas según los parámetros establecidos.
Nombre	asociarActividadesPersona($parámetros)
Descripción	Permite asignar actividades a la persona seleccionada según los parámetros establecidos.
Nombre	getActividad($parámetros)
Descripción	Devuelve todas las actividades que existen según los parámetros definidos.
Nombre	cambiarEstado(parámetros)
Descripción	Permite cambiar el estado de las actividades según los parámetros establecidos.

Tabla 6: Descripción de la clase DatActividadesModel.

Clase Entidad:

Nombre: DatActividades.	
Tipo de clase: Entidad	
Para cada responsabilidad:	
Nombre	cambiarEstadoActividad($parametros)
Descripción	Permite cambiar el estado de las actividades según los parámetros definidos teniendo en cuenta el cargo de la persona autenticada.
Nombre	validarParámetrosActividades($parámetros)
Descripción	Valida los parámetros definidos para las actividades.
Nombre	getActividades($parámetros)
Descripción	Devuelve las actividades que existen según los parámetros establecidos teniendo en cuenta el cargo de la persona autenticada.

Tabla 7: Descripción de la clase DatActividad.

3.4. Implementación

A continuación se presenta la estructura del marco de trabajo Sauxe, mostrando cómo será organizada la implementación del sistema a desarrollar, de manera que facilite la organización y claridad durante el desarrollo.

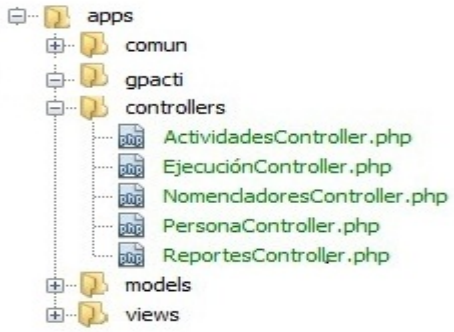


Figura 18: Contenido del sistema GPACTI dentro de la carpeta apps.

En la carpeta denominada **apps** se almacenan las clases controladores y las clases modelo de cada una de las funcionalidades a desarrollar. Las clases controladoras son responsables de la lógica de negocio, gestionando todo el flujo de datos y operaciones entre la vista y demás clases del negocio, se definió una clase por cada uno de los componentes del sistema. La carpeta apps incluye además:

comun : contiene la carpeta recursos y dentro de esta una denominada xml. Esta última tiene los ficheros: ioc, validator, exception.

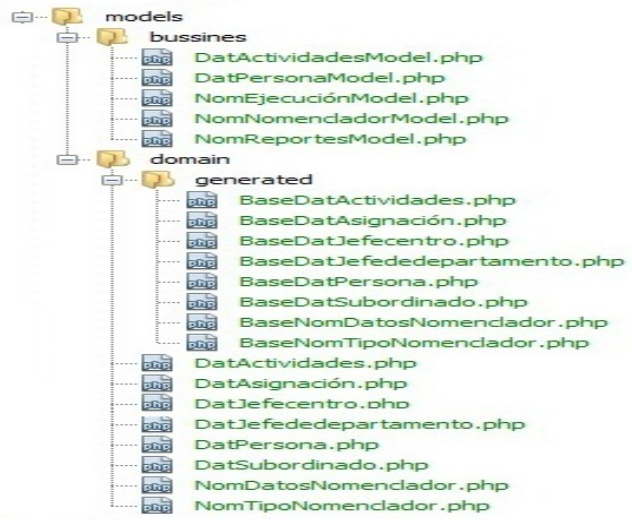


Figura 19: Contenido de la carpeta models.

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

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

Cada una de estas clases mencionadas anteriormente heredará de una clase generada igualmente por el Doctrine Generator las cuales se ubican en otra carpeta dentro de esta llamado generated. Las clases modelo actúan como intermediarias entre las clases controladoras y las clases entidades. Complementan las funcionalidades de las clases controladoras y realizan las funciones de adicionar, modificar y eliminar datos.

Figura 20: Contenido de la carpeta views dentro del apps.

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

Al mismo nivel de la carpeta apps se encuentra la carpeta **web,** esta contiene las vistas de los subsistemas y componentes.

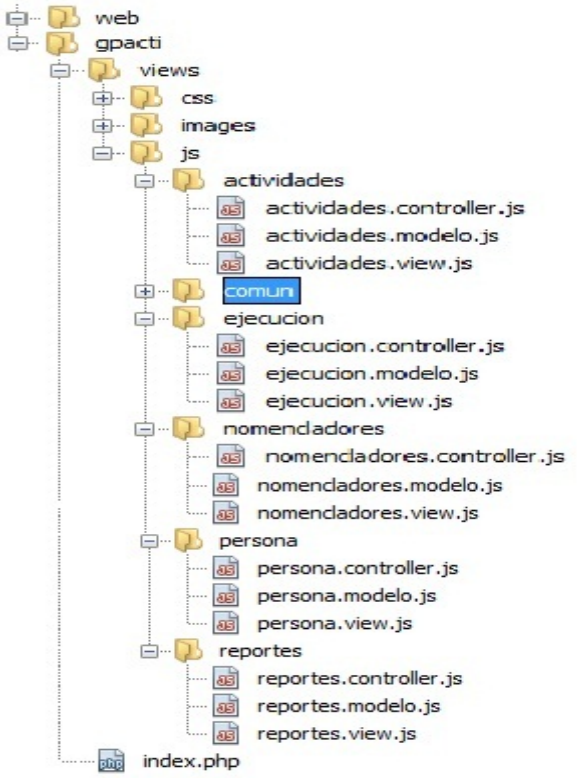


Figura 21: Contenido de la carpeta web.

La carpeta views contiene los css, que incluyen las clases necesarias para estructurar gráficamente el componente, separando de esta forma el estilo del contenido. La carpeta js comprenderá las clases Java Script necesarias para que el usuario interactúe con el sistema y obtenga los resultados necesarios. Además icluye la clase index.php, en esta se establece el código igual para todos los componentes. En ella se encuentra la dirección del archivo de configuración y a través de este inicializa la aplicación para que se carguen en la misma un conjunto de componentes necesarios para su funcionamiento.

3.5. Validación del modelo de diseño propuesto

Una métrica es un instrumento que cuantifica un criterio y persigue comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado al nivel del proyecto. Entre los atributos de calidad que permiten medir la calidad del diseño propuesto se encuentran [43]:

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

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

3.5.1. Métrica Tamaño Operacional de Clase (TOC)

Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

Atributo que afecta	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 en el grado de reutilización de la clase.

Tabla 8: Tamaño operacional de clase (TOC).

Atributo	Categoría	Criterio
Responsabilidad	Baja	< =Prom (11,06)
	Media	Entre Prom Y 2* Prom
	Alta	> 2* Prom

Complejidad de implementación	Baja	< =Prom
	Media	Entre Prom y 2* Prom
	Alta	> 2* Prom
Reutilización	Baja	> 2*Prom
	Media	Entre Prom y 2* Prom
	Alta	<= Prom

Tabla 9: Rango de valores para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

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

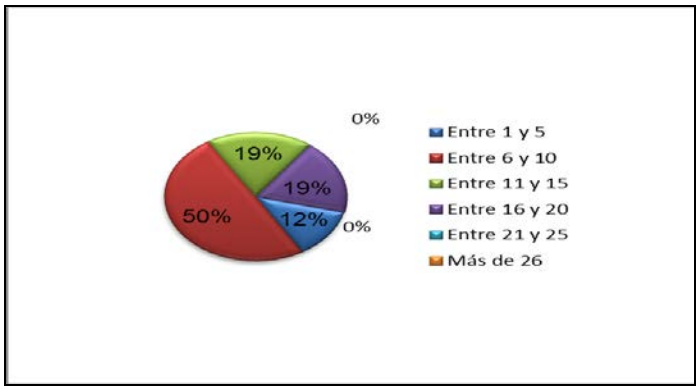


Figura 22: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

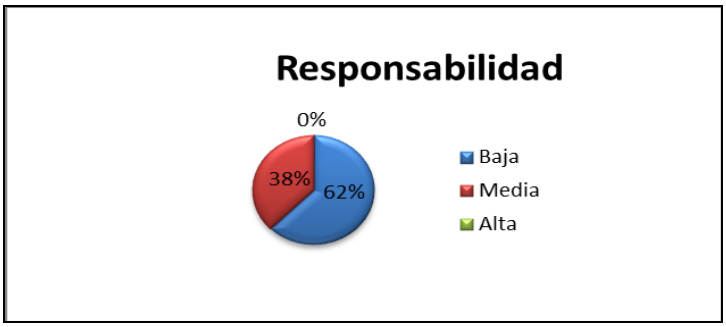


Figura 23: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

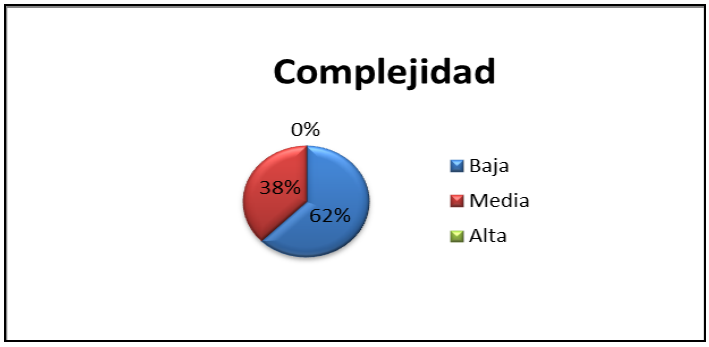


Figura 24: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación.

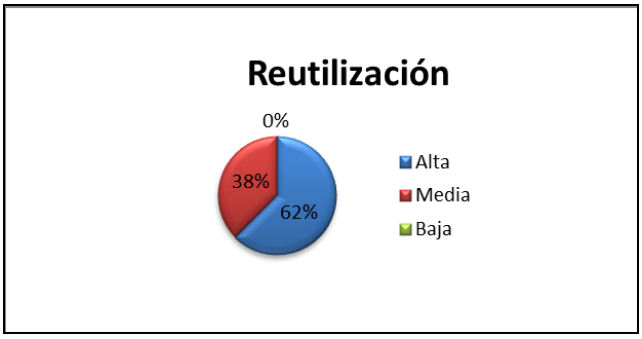


Figura 25: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para el sistema es simple y tiene una calidad aceptable, teniendo en cuenta que la mayoría de las clases (81%) se distribuyen las operaciones. Los atributos medidos se encuentran en un nivel satisfactorio en el 62% de las clases, de manera que se puede observar cómo se fomenta la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

3.5.2. Métrica Relaciones entre Clases (RC)

Esta métrica está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

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

Tabla 10: Relaciones entre clases (RC).

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	< =Prom (1,6)
	Media	Entre Prom y 2* Prom
	Alta	> 2* Prom
Reutilización	Baja	> 2*Prom
	Media	Entre Prom y 2* Prom
	Alta	<= Prom

Cantidad de Pruebas	Baja	< =Prom
	Media	Entre Prom y 2* Prom
	Alta	> 2* Prom

Tabla 11: Rango de valores para la evaluación técnica de los atributos de calidad (Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de Pruebas) relacionados con la métrica RC.

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

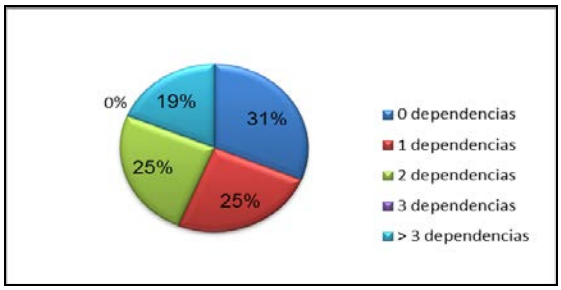


Figura 26: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

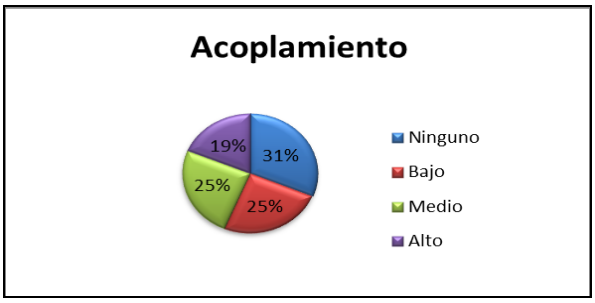


Figura 27: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

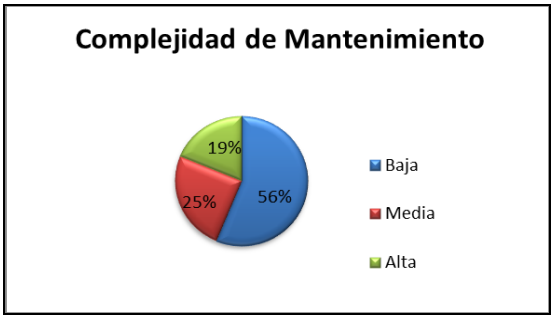


Figura 28: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

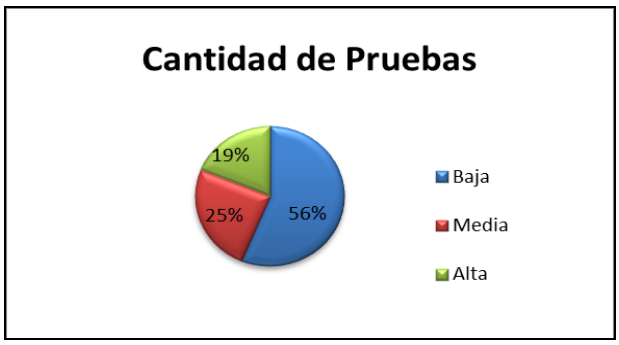


Figura 29: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

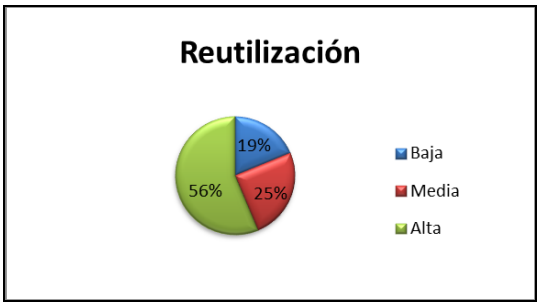


Figura 30: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el sistema es simple y tiene una calidad aceptable, teniendo en cuenta que la mayoría de las clases (81%) poseen 2 o menos dependencias respecto a otras. Los atributos medidos se encuentran en un nivel satisfactorio; en el 56% de las clases el grado de

dependencia o acoplamiento es mínimo, la Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 56% de las clases.

3.6. Pruebas de software aplicadas al sistema

Las pruebas constituyen una etapa imprescindible durante el proceso de desarrollo del software pues permiten detectar y corregir el máximo de errores posibles antes de la entrega al cliente del software desarrollado, por lo que el éxito de las mismas puede mejorar la percepción de calidad del usuario final y lograr su satisfacción. El objetivo principal de las pruebas es asegurar que el software cumpla con las especificaciones requeridas y eliminar los posibles defectos que este pudiera tener. Es importante considerar que las pruebas de software no garantizan que un sistema esté libre de errores, sino que se detecten la mayor cantidad de defectos posibles para su debida corrección [40].

3.6.1. Pruebas de caja negra

A este tipo de prueba también se le conoce como Prueba de Caja Opaca o Inducida por los Datos. Se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación, esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa. En esencia permite encontrar [45]:

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

A la hora de evaluar dinámicamente un sistema se debe comenzar por los componentes más simples y pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Las pruebas se aplican en distintos niveles de trabajo, dentro de estos se distinguen:

- **Pruebas de Unidad:** prueba individual a las unidades separadas de un sistema de software.
- **Pruebas de Integración:** los componentes individuales son combinados con otros componentes para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente.

- **Pruebas del Sistema:** son usualmente conducidas para asegurar que todos los módulos trabajan como sistema sin error. Es similar a la prueba de integración pero con un alcance mucho más amplio.

- **Pruebas de Aceptación:** son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales [46].

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas se encuentran [47]:

Análisis de Valores Límites: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Grafos de Causa-Efecto: permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Partición de Equivalencia: técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Se centra en la evaluación de clases de equivalencia que representan un conjunto de estados válidos o no válidos para condiciones de entradas existentes en el software.

3.6.2 Pruebas realizadas

Para verificar el correcto funcionamiento del sistema se realizaron pruebas funcionales aplicando el método de caja negra y específicamente la técnica de partición de equivalencia, donde se realizó los diseños de caso de pruebas de los requisitos funcionales principales, a continuación se muestra un ejemplo de estos:

Diseño de Caso de Prueba para el requisito Adicionar actividades

Condiciones de ejecución

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seguir el flujo de trabajo **GPACTI/Plan/Planificación/Adicionar.**

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar actividades.	Se adiciona los datos de la actividad, con los atributos: Tipos de actividad, denominación, fecha inicio, fecha fin, descripción .	EP 1.1: Adicionar datos de la actividad introduciendo datos correctos.	– Se presiona el botón **Adicionar**. – Se insertan todos los datos. – Se presiona el botón **Aceptar**. – Se muestra el mensaje de información: "La actividad ha sido adicionada satisfactoriamente." – Se cierra la ventana. – Se pasa el mouse por encima del mensaje que se muestra para cerrarlo.
		EP 1.2: Adicionar una actividad dejando campos requeridos en blanco.	– Se presiona el botón **Adicionar**. – Se introducen los datos dejando campos requeridos en blanco. – Se presiona el botón **Aceptar**. – Se muestra el mensaje de información: "Tiene errores en los campos del formulario". – Se cierra la ventana. – Se pasa el mouse por encima del mensaje que se muestra para cerrarlo.
		EP 1.3: Adicionar una actividad presionando el botón Aplicar	– Se presiona el botón **Adicionar**. – Se introducen los datos en el formulario. – Se presiona el botón **Aplicar**. – Se muestra el mensaje de información: "La actividad ha sido adicionada satisfactoriamente." – El sistema posibilita seguir adicionando datos de la actividad. – Se pasa el mouse por encima del mensaje que se muestra para cerrarlo.
		EP 1.4: Cancelar	– Se presiona el botón **Adicionar**. – Se introducen o no los datos en el formulario. – Se presiona el botón **Cancelar**. – Se cierra la ventana.

Tabla 12: Caso de prueba del requisito Adicionar actividades.

El sistema fue sometido a tres iteraciones de pruebas de caja negra. En la primera iteración se detectaron 9 no conformidades, de las cuales 4 fueron de aplicación y 5 de documentación. Dentro de las no conformidades referentes a las de aplicación se encuentran 3 de validación y 1 de ortografía. En cuanto a las de documentación se encontraron 5 no conformidades de correspondencia. En la segunda iteración se detectaron 3 no conformidades de documentación, las cuales fueron de no correspondencia. Todas estas no conformidades fueron resueltas seguidamente de ser detectadas, lo que permitió que el sistema pasara a una tercera iteración en la que no se identificaron no conformidades, obteniendo resultados satisfactorios.

La aplicación desarrollada fue presentada ante el cliente, el cual luego de haberla probado estuvo satisfecho con el producto entregado. Prueba de esto lo constituye el acta de aceptación emitida por este, en la cual consta que el sistema realizado cumple con las condiciones de calidad necesarias y satisface las necesidades plasmadas por este. Para más información acerca del acta de aceptación emitida, ver **Anexo 7**.

3.7. Conclusiones del capítulo

En este capítulo se realizó la implementación y validación del software, lo que permitió:

- Realizar la descripción de las clases y funcionalidades del sistema, posibilitando la obtención de una vista organizada de estos en el marco de trabajo Sauxe.
- Aplicar las métricas para validar el diseño, las cuales arrojaron resultados satisfactorios, demostrando que este era simple y los atributos de calidad alcanzaban niveles favorables.
- Realizar pruebas de caja negra para detectar y corregir el máximo de errores posibles antes de la entrega al cliente, liberándose el sistema por calidad interna.

CONCLUSIONES

Una vez terminado el presente trabajo de diploma se logró alcanzar el objetivo general propuesto, dando cumplimiento a los objetivos específicos, donde:

- El estudio de los sistemas de planificación de actividades y postgrado existentes a nivel nacional e internacional, posibilitó formalizar el marco teórico-conceptual de la investigación, evidenciándose de esta manera la no existencia de una solución informática capaz de cubrir todas las funcionalidades requeridas.

- El modelamiento del negocio, el análisis y diseño del Sistema de Gestión para la Planificación de las Acciones de CTI y Postgrado en el CEIGE, permitió la correcta implementación y automatización de los procesos y funcionalidades identificadas.

- La realización de pruebas de caja negra permitió validar la implementación de las funcionalidades desarrolladas corroborando la calidad del sistema y demostrando que el mismo está listo para su uso.

RECOMENDACIONES

Después de realizadas las conclusiones de este trabajo se recomienda:

- Generar reportes mediante gráficas, lo que permitirá determinar hacia dónde deben enfocar los profesionales su mayor esfuerzo para cumplir las metas trazadas por el centro en un menor tiempo.

BIBLIOGRAFÍA

1. **CubaDebate.** CubaDebate. *cuba-supero-el-millon-de-graduados-universitarios.* [En línea] 29 de 01 de 2011. [Citado el: 5 de 12 de 2012.] http://www.cubadebate.cu/opinion.

2. **Morales, V. .** *Postgrado y Desarrollo en América Latina.* Caracas : Ediciones del Centro de Estudios e Investigaciones sobre Educación Avanzada (CEISEA), Coordinación Central de Estudios de Postgrado, Universidad Central de Venezuela., 2010.

3. **Colectivo de autores.** Diccionario de la Academia de la Real Lengua Española. [En línea] 2001. [Citado el: 15 de 01 de 2013.] dpd.rae.es. 22 edición.

4. **Braun-Thürmann.** *Innovation - Eine Einführung.* Bielefeld : Transcript-TB, 2005. ISBN 978-3-89942-291-7.

5. **Derechos reservados S.A.** Sistema automatizado de información y evaluación de los procesos. [En línea] 10 de Enero de 2010. [Citado el: 16 de enero de 2013.] http://www.coltlax.edu.mx/index.php?pagina=163&menuIzquierdo=7.

6. **Dirección General de Servicios Informáticos**. *SIU GUARANI.* [En línea] 11 de Octubre de 2006 . [Citado el: 20 de enero de 2013.] http://siu.unf.edu.ar/index.php.

7. **Alain VA.** *Sistema de Gestión Académica para Universalización.* Matanzas : Universidad de Matanzas "Camilo Cienfuegos", 2005 . 102.

8. **Licenciado Ángel Juan Otero Méndez.** *Sistema Informático para la gestión de la formación de los profesionales del municipio Mayarí.* Moa : Instituto Superior Minero Metálurgico Dr. "Antonio Nuñez Jiménez", 2008.

9. **Derechos Reservados.** *Modelo de desarrollo de software del centro CEIGE.* Habana : Centro de Informatización de Gestión de Entidades, 10/12/2012.

10. **O.E.Sánchez, B.Garea,S.Lantigua y otros.** *Sistema de Información para la Gestión de Programas de Ciencia e Innovación .* Cuba : s.n., 2008.

11. **Ing.Oiner Gómez Baryolo, Ing.Yoandry Morejón Borbón, Ing.Darien García Tejo.** *Arquitectura Tecnológica para el Desarrollo de Software.* Cuidad de la Habana : s.n.

12. **Jimmy Wales .** Ext JS en Sencha.com (en inglés). [En línea] 15 de mayo de 2012. [Citado el: 22 de enero de 2013.] http://www.sencha.com/products/js/.

13. **Esser, S.** *Secure Programming with the Zend-Framework.* 2009.

14. **Pedro Boda, K.N., Javier Martínez, Benjamín Gonzales.** *Zend Framework Manual en Español.* 14 de noviembre del 2009.

15. **Smith L.** *Introduction to the Doctrine Object Relational Mapper.* Abril 2009.

16. **Qué es Doctrine ORM? .** Qué es Doctrine ORM? . [En línea] 15 de noviembre de 2012. http://www.tecnoretales.com/programacion/que-es-doctrine-orm/.

17. **Larman, Craig.** *El mundo Informático. El mundo Informático.* 17 de Agosto de 2006.

18. **EcuRed. EcuRed.** Patrones_de_dise%C3%B1o_y_arquitectura. [En línea] [Citado el: 18 de Enero de 2013.] http://www.ecured.cu/index.php/.

19. **Erich Gamma, R.H., Ralph Johnson, John Vlissides - Addison Wesley .** *(GoF- Gang of Four), Design Patterns. Elements of Reusable Object-Oriented Software.*2002.

20. **Larman - Prentice Hall.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.*2008.

21. **Hernández, Elier Cruz.** *Implementación de los Componentes Depreciación y Submayor del Subsistema Activo Fijo Tangible del Sistema integral de Gestión Cedrux.* Ciudad de la Habana : s.n., 2009.

22. **Modeler, BizAgi Process.** BizAgi Process Modeler. [En línea] [Citado el: 20 de febrero de 2013.] http://www.bizagi.com/docs/BPMNbyExampleSPA.pdf.

23. **Perdita Stevens, Rob Pooley, Addison Wesley.** *Utilización de UML en Ingeniería del Software con Objetos y Componentes.* 2002.

24. **Eric Brown .** *The Myth of Self-Describing XM The Myth of Self-Describing XML.* 2003.

25. **openformats.org.** openformats.org. [En línea] 27 de Agosto de 2010. [Citado el: 25 de febrero de 2013.] http://www.openformats.org/es9.

26. **David Flanagan.** *JavaScript: The Definitive Guide (Fifth Edition).* (ISBN-13: 9780596101992).2005.

27. **Garcia, Yunesky del Rio.** *Implementación del Módulo Índice de Peligrosidad Pre-Delictiva del Proyecto Sistema de Gestión Fiscal.* Cuidad de la Habana : s.n., 2010.

28. **Dave Crane, Eric Pascarello, Darren James .** *Ajax in Action.* (ISBN-13: 978-1932394610). 2009.

29. **Aranzadi.es.** Aranzadi.es. [En línea] [Citado el: 2 de marzo de 2013.] http://www.aranzadi.es/index.php/informacion-juridica/informacion-interes/glosario-de-terminos-sobre-internet-y-spam.

30. **The Apache Software Foundation.** *Apache License and Distribution FAQ.* 2007.

31. **Apache** 2.2. In: Apache 2.2. [En línea] [Citado el: 15 de marzo de 2013.] http://recursostic.educacion.es/observatorio/web/es/software/servidores/580-elvira-mifsud.

32. **postgresql.org .** PostgreSQL, Award Winning Software. [En línea] 19 de enero de 2008. [Citado el: 16 de febrero de 2013.] www.postgresql.org.

33. **González, Carlos D.** Curso PostgreSQL, SQL avanzado y PHP. [En línea] febrero de 2012. [Citado el: 24 de febrero de 2013.] http://www.usabilidadweb.com.ar/postgre.php.

34. **Palacios, Rodolfo Vázquez .** Scribd. [En línea] 2 de Septiembre de 2010. [Citado el: 5 de marzo de 2013.] http://es.scribd.com/doc/37957736/Mozilla-Firefox-Original..

35. **Herramienta Case Visual Paradigm.** Herramienta Case Visual Paradigm. [En línea] [Citado el: 17 de marzo de 2013.] http://dianbeel.blogspot.com/2012/06/segundo-trabajo-herramienta-case-visual.html.

36. **M. Domínguez-Dorado .** *La alternativa a Eclipse.Todo Programación.NetBeans IDE 4.1.* Madrid : Editorial Iberprensa, 2005. DL M-13679-2004.

37. **Que es SVN.** In: Que es SVN . [En línea] [Citado el: 22 de febrero de 2013.] http://lihuen.linti.unlp.edu.ar/index.php?title=C%C3%B3mo_usar_SVN.

38. **Principales Ventajas del Subversión**. In:Principales Ventajas del Subversión. [En línea] 4 de febrero de 2013. http://www.osmosislatina.com/subversion/basico.html.

39. **Tera.loc.** [En línea] [Citado el: 16 de marzo de 2013.] http://www.teraloc.com/portal/index.php?option=com_content&view=article&id=51&Itemid=92.

40. **Unidad de Compatibilización Integración y Desarrollo.** *Proceso de Desarrollo y Gestión de Proyectos de Software.* Ciudad de la Habana : s.n., 2009.

41. **FELIPE HERNÁNDEZ SALAZAR Y IVIS MARAY AMARÓ MORENO. .** *Componente para el control de acceso en el marco de trabajo Symfony 1.3 para el proyecto productivo Sistema de Informatización de la Gestión Fiscal II .* Ciudad Habana : UCI, 2012.

42. **Metodología de Gestión de Requisitos**. [En línea] [Citado el: 7 de marzo de 2013.] https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales.

43. **Fernández González, Mairelys y Zorrilla Rivera, Osley.** *Diseño e implementación del componente Ajuste al Costo del Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CEDRUX.* La Habana : UCI, 2010.

44. **EcuRed.** Enciclopedia cubana. [En línea] 12 de Febrero de 2012. [Citado el: 16 de marzo de 2013.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.

45. **EcuRed.** Enciclopedia cubana. [En línea] [Citado el: 15 de Mayo de 2013.] http://www.ecured.cu/index.php/Pruebas_de_software.

46. **Carrillo, Fernando.** Pruebas del Software: Niveles de Prueba del Software. [En línea] 3 de enero de 2011. [Citado el: 17 de mayo de 2013.] http://ingenieriadepruebasdelsoftware.blogspot.com/2011/01/niveles-de-prueba-del-software.html.

47. **Moreno Álvarez, José Luis.** Colección de Tesis Digitales. [En línea] Universidad de las Américas Puebla. [Citado el: 25 de mayo de 2013.] http://catarina.udlap.mx/u_dl_a/tales/ documentos/lis/ moreno_a_jl/portada.html.

48. **Samespinosa.** Mapa de procesos. Business & Mgmt. S.l. [En línea] 20 de febrero de 2009. [Citado el: 17 de mayo de 2013.] http://www.slideshare.net/samespinosa/mapa-de-procesos. 1053479.

49. **UCI.** GESPRO. [En línea] [Citado el: 26 de mayo de 2013.] http://gespro-help.prod.uci.cu/introduction/.

50. **EcuRed.** Enciclopedia cubana. [En línea] [Citado el: 26 de mayo de 2013.] http://www.ecured.cu/index.php/Lenguaje_de_programación.