Universidad de las Ciencias Informáticas

Facultad 3

Título: Sistema de Gestión de la información Ciencia Tecnología e Innovación del CEIGE.

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

Autores:

Luis Ángel Roque Lavandero

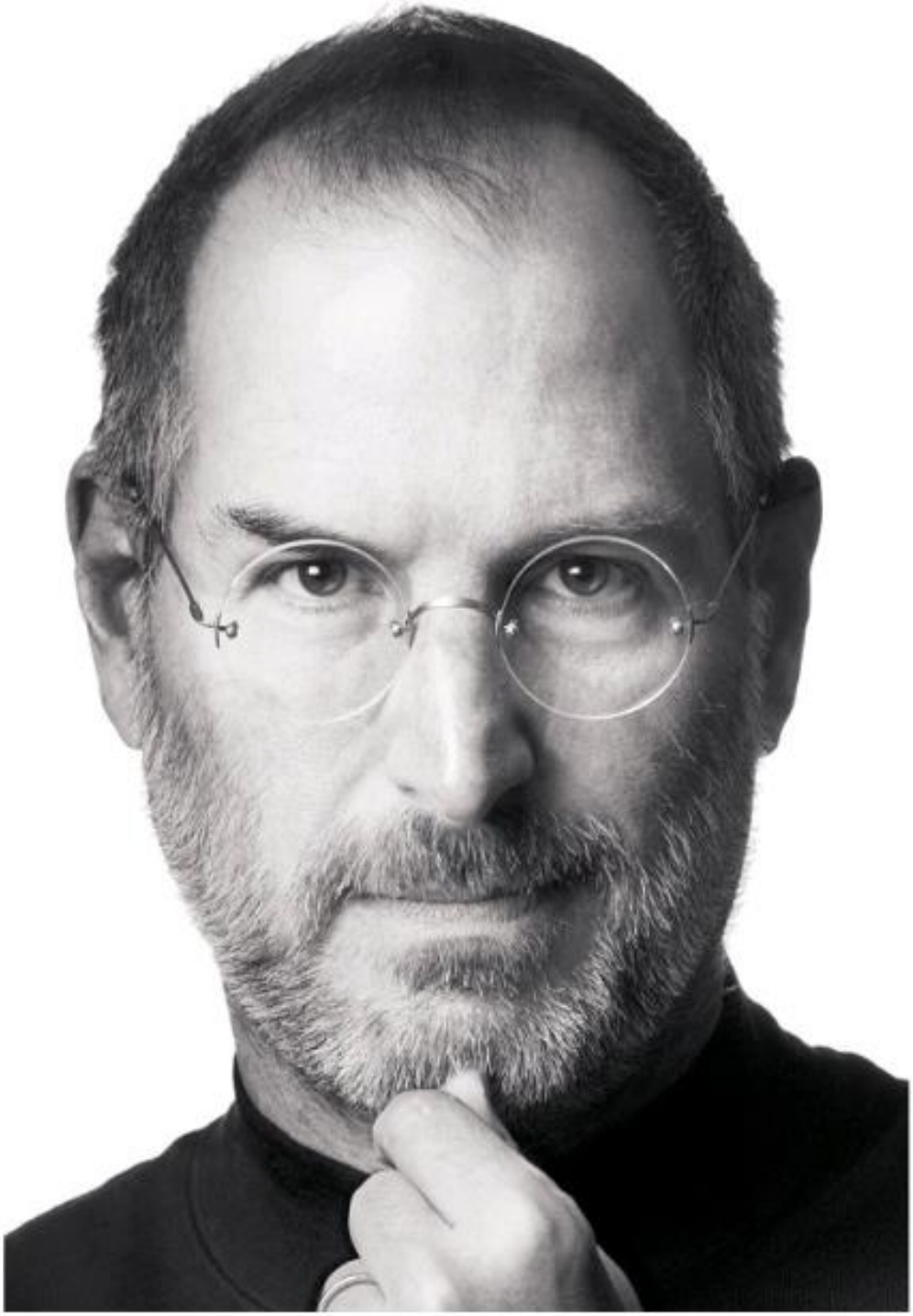
Jorge César Cabrales Camino

Tutores:

Ing. Alexis Ramírez Álvarez

Ing. Boris Luis Correa Frías

La Habana, Cuba.
Junio 2013



No tenemos la oportunidad de hacer muchas cosas, por lo que cada cosa que hagamos debe ser excelente. Porque esta es nuestra vida.

Steve Jobs.

DECLARACIÓN DE AUDITORÍA

Declaramos que somos los únicos autores de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Jorge César Cabrales Camino

_____

Firma del Autor

Luis Ángel Roque Lavandero

_____

Firma del Autor

Ing. Boris Luis Correa Frías

_____

Firma del Tutor

Ing. Alexis Ramírez Álvarez

_____

Firma del Tutor

DATOS DE CONTACTOS

Ing. Boris Luis Correa Frías

Graduado de Ingeniería en Ciencias Informáticas en la UCI en el año 2012. A partir de ese momento se incorpora como Desarrollador a la línea Finanzas del proyecto ERP Cuba del Centro para la Informatización de la Gestión de Entidades, rol que ocupa actualmente. En su trabajo de diploma desarrollo una Bloque para la Plataforma de Tele formación Moodle en su versión 2.1.x que permitiera exportar los contenidos de sus cursos a formato SCORM de manera que estos pudieran ser reutilizados en otros entornos educativos. Se encuentra cumpliendo su primer año de adiestramiento.
Categoría docente: Ninguna
Correo electrónico: *cfrias@uci.cu.*

Ing. Alexis Ramírez Álvarez

Graduado de Ingeniero en Informáticas en el 2011. Recién Graduado en Adiestramiento. Se desempeña como desarrollador y arquitecto de datos y sistemas de los subsistemas Caja, Banco, Cobros y Pagos y Gestión de Crédito del sistema Cedrux del subsistema Finanzas.
Categoría docente: Ninguna
Correo electrónico: *alexramirez@uci.cu.*

.

AGRADECIMIENTOS

Luis Ángel: *A mi mamá y a mi abuela Pilar por siempre estar cerca de mí en los momentos malos y bueno, apoyándome en cada paso de mi vida, llevándome siempre por el camino del bien, por las ideas que me inculcaron desde niño y he sabido defender, por cada beso y cada abrazo que he recibido de ellas, esforzándose mucho para que yo pudiera ser un profesional.*

A mi papá por ser un ejemplo de profesional a seguir, por inculcarme siempre la idea de superarme, el apoyo incondicional que siempre recibí de él, por confiar en mí en todo momento y toda situación, por saber escucharme y comprenderme.

A mi abuela Lili, a mi tía Idey, a mi abuelo y a mi tío Omar por ser como otros padres para mí, nunca tuve de ellos un no, siempre preocupados por mí y pendiente de cualquier situación en la que estuviera implicado, por ayudarme en todo lo que podían, por hacerme tan feliz cada vez que los iba a visitar, me ayudaron a formarme como hombre y entender el mundo tal y como es, por demostrarme que nada cae del cielo y que todo en la vida lleva un sacrificio.

A mi tío Migue que me tuvo desde pequeñito a su lado y siempre lo vi con esas ideas de superarse de afrontar todos los problemas con una mente positiva.

A mi esposa Yunay por darme un hijo tan bello que me ha hecho exigirme más como hombre, como estudiante, como persona, por entender que no estaba a su lado no porque yo no quisiera sino para darle un futuro mejor a nuestro hijo. Por cada llamada que me hacía en esos momentos malos que tenemos en ocasiones, todos por estar lejos de la familia.

A mis primos Omarito, Emmanuel, Elizabeth, Adonis, Miguel Enrique, Melisa por siempre tenerme como un ejemplo a seguir y hacerme sentir alagado e intentar darle el mejor ejemplo. A mi hermana por siempre seguir mis pasos y conseguirlos.

A mis tíos Tita, Fefa, Papo, Luis, Pepe, mis primas Iveth y Dianelis, a mi primo José Luis. A Rael y a bettys por ser tan buenos conmigo y quererme como un hijo más.

A mis amigos Edriel y Alejandro que los quiero como si fueran hermanos míos por todos estos años donde hemos sabido compartir los momentos malos y bueno, por atenderme cada vez que los necesitaba. A Carlos, Yandy, Hernan, David, Juan Carlos, Aroldo, Yasmany, Cire por no dejar que me sintiera solo cuando me incorpore a su grupo. A mis compañeritas Danay, la rubia, Yessika, Liliam por ayudarme en estos últimos años. Al butín, Daniel, Amilkar, Raul, Maylenis, el Chiki, Ricardo, a la justicia, al July y a otros que no se encuentran ya en la universidad como el Negrón, Leyandry, Jorge, el pompa, Leandro. A mi compañero de tesis que sin él no lo hubiéramos podido conseguir al igual que sin mi hermano Alejandro que siempre estuvo guiándonos y ayudándonos incondicionalmente en todo momento. A mis tutores. Y a todas aquellas personas que de una forma u otra me han ayudado a realizar este sueño en mi vida.

AGRADECIMIENTOS

Jorge César: *A mi madre por su entrega, por todo su sacrificio, por toda su dedicación, por su amor incondicional, por enseñarme a elegir el camino correcto y apoyarme en cada paso de mi vida, por esforzarse al máximo y dar todo su empeño para que me pudiera hacer un profesional, por ser la mujer que más amo y quiero en el mundo, a ella le agradezco todo lo que soy.*

A mi papá por siempre confiar en que yo podía realizar este sueño, por ser un ejemplo a seguir, por darme su amor y cariño siempre que lo necesite, por escucharme y aconsejarme cuando me pasaba algo, por darme su apoyo incondicional y por ser el mejor padre del planeta.

A mi hermano por complacerme y consentirme en todo lo que pido, por ser mi modelo a seguir como hombre y como profesional, por estar siempre que lo necesito y por ser el hombre que más quiero y admiro en el mundo.

A mi sobrino Cristian "El pichi" por ser el niño más lindo del mundo.

A mis abuelas Alida y Acela por su amor y cariño y por ayudarme a realizar este sueño. Las quiero mucho.

A mis abuelos Nene y Arquimedes que sé que siempre me observan desde el cielo, nunca se me olvidarán sus consejos, y aunque no puedan compartir este momento conmigo, siempre los recordaré.

A mis tías y tíos Adalis, Betty, Margot, Cukita, Pupo, Melo. A mis primos y primas Kenmy, Marlon, Karel, Danielito, Abelito, Acner, Isabean, Isandra, Yanela, Hético, Gretel, Anni, Tikin, Manuel, Alejandro, Andresito, gracias por confiar en mí.

A Aldo y Yuli que son como mis segundos padres. Gracias por acogerme como su familia. Los quiero mucho.

A Alejandro mi hermano muchas gracias por ayudarme te lo agradezco infinitamente tú fuiste mi tutor.

A todos mis amigos Carlitín que aunque por cosas de la vida no puedo estar a mi lado en este momento, para mí nunca dejo de estar presente. A Alexito, Laura, Ale, Carlitín Amado, Jorgito, Alionis, Alfreditin, Piro, Miclin, Alexander Trabajo, Osvaldito, Javier, Alberto, Landy, gracias por ayudarme siempre que lo necesité.

A Lisbeth que me ayudó mucho en mi carrera, te lo agradezco.

A mis compañeros de aula por haber compartido estos 5 años con ustedes, nunca los olvidaré.

A mi piquete de la UCI Raúl, Mailenys, Amílcar, Ricardo, Fito, Ramón, Luis Carlos, El Butin, Duvergel, el yuli, el justice, gracias por compartir conmigo momentos inolvidables.

A mi compañero de tesis por vivir juntos momentos de alegrías y desilusiones durante esta etapa y porque sin su ayuda no lo hubiera podido lograr. A mis tutores.

A todos aquellos que contribuyeron de una forma u otra en la elaboración de este trabajo de diploma muchas gracias.

DEDICATORIA

Luis Ángel

A mi familia, especialmente a mi hijo que amo con todo mi corazón, con el interés que comprenda desde pequeño que todo hombre necesita superarse para ser cada día mejor.

Jorge César

A mi madre por ser la razón de ser de mi vida, gracias por intuir, casi de manera exacta, el andar de mis pasos y dejarme tropezar con los escollos de la vida, así, he comprendido, que tu abrazo abarca más de una legua, abarca más que un te quiero, abarca, la vida toda.

A mi papa y mi hermano por ser mis amigos, mis hermanos, mis compañeros, mis consejeros y mi todo en la vida. Los amo.

A mis amigos, que son verdaderos y siempre están ahí para ayudarme.

Resumen

La gestión de la información es un proceso crítico para el funcionamiento adecuado de un sistema de Ciencia, Tecnología e Innovación (CTI). Acceder a investigaciones realizadas, estar al tanto del estado del arte, de la evolución de las tecnologías y el quehacer nacional e internacional es un tema de preocupación de todo centro productivo que combine elementos científicos con el desarrollo. Gestionar el conocimiento de sus profesionales se convierte en un objetivo clave para el ámbito productivo - académico. Estas necesidades se encuentran presentes en el Centro de Informatización de la Gestión de Entidades (CEIGE), perteneciente a la Universidad de las Ciencias Informáticas (UCI), centro de formación y producción por excelencia.

La presente investigación se basa en el diseño e implementación de un sistema para la gestión de la información que permite evaluar la producción científica de los profesores y especialistas de la UCI y en específico de CEIGE, potencia los resultados científicos y de innovación, premia el trabajo en equipo y se adapta a las características del centro. Para la realización del sistema se utilizó el Modelo de Desarrollo de Software de CEIGE, así como tecnologías Web basadas en el marco de trabajo Sauxe.

Palabras claves: Gestión de la información, producción científica, resultados científicos.

Índice

INTRODUCCIÓN 14

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 19

1.1 INTRODUCCIÓN 19
1.2 CONCEPTOS BÁSICOS RELACIONADOS CON EL DOMINIO DEL PROBLEMA 19
1.3 SISTEMAS DE GESTIÓN 20
1.3.1 Gestión de la información 20
1.3.2 Tendencias de los sistemas de gestión de la información a nivel internacional 20
1.3.3 Experiencias en el uso de sistemas de gestión de la información científica en Cuba 22
1.3.4 Experiencias en el uso de sistemas de gestión de la información en la UCI 23
1.3.5 Análisis de los sistemas de gestión presentados anteriormente 24
1.4 MODELO DE DESARROLLO 25
1.4.1 Descripción de las fases del ciclo de vida 25
1.5 MARCO DE TRABAJO UTILIZADO 26
1.5.1 Marcos de trabajo que utiliza Sauxe 27
1.5.2 Lenguajes de programación utilizados 28
1.5.3 Arquitectura de Sauxe: Cliente-Servidor 30
1.5.4 Patrón Arquitectónico 31
1.6 LENGUAJE DE MODELADO 32
1.7 HERRAMIENTAS DE DESARROLLO 32
1.7.1 Herramienta Case: Visual Paradigm 8.0 32
1.7.2 Entorno de desarrollo integrado (IDE) 33
1.7.3 Servidor de aplicaciones Web Apache 2.2 34
1.7.4 Servidor de Base de Datos PostgreSQL 8.3.8 34
1.7.5 Navegador Web Mozilla Firefox 3.6 35
1.8 CONCLUSIONES PARCIALES 35

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA 36

2.1 INTRODUCCIÓN 36
2.2 PROPUESTA DEL SISTEMA 36
2.3 MODELO CONCEPTUAL 36
2.4 ESPECIFICACIÓN DE LOS REQUISITOS DE SOFTWARE 37
2.4.1 Requisitos funcionales 38

2.4.2 Requisitos no Funcionales 39
2.4.3 Descripción detallada de los requisitos 41
2.5 VALIDACIÓN DE REQUISITOS 43
2.6 ANÁLISIS Y DISEÑO 43
2.6.1 Diagrama de clases del diseño 44
2.6.2 Diagrama de secuencia 45
2.6.3 Patrones de Diseño 46
2.6.4 Prototipo de interfaz de usuario 49
2.6.5 Modelo de Datos 51
2.6.6 Diagrama de clases persistentes 55
2.6.7 Diagrama de Componentes 55
2.6.8 Métricas de validación del diseño 56
2.6.9 Estándares de codificación 63
2.7 CONCLUSIONES PARCIALES 65

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA 66

3.1 INTRODUCCIÓN 66
3.2 ASPECTOS FUNDAMENTALES DE LA IMPLEMENTACIÓN 66
3.2.1 Descripción de las clases y funcionalidades del sistema. 68
3.3 DIAGRAMA DE DESPLIEGUE 71
3.4 PRUEBAS DE SOFTWARE 72
3.4.1 Prueba de caja blanca 72
3.4.2 Prueba de caja negra 75
3.5 CONCLUSIONES PARCIALES 77

CONCLUSIONES GENERALES 78

RECOMENDACIONES 79

REFERENCIAS 80

Índice de Figuras

Figura 1. Arquitectura de Sauxe. 26
Figura 2. Estructura lógica del marco de trabajo Doctrine. 28
Figura 3. Arquitectura Cliente-Servidor. 30
Figura 4. Modelo Vista Controlador. 32
Figura 5. Modelo Conceptual..... 37
Figura 6. DCD Gestionar Evento..... 44
Figura 7. Diagrama de secuencia Adicionar Evento. 45
Figura 8. Clase ControlctiController. 47
Figura 9. Clase DatEvento que depende de la clase BaseDatEventoModel. 48
Figura 10. Ejemplo de implementación del patrón singleton en la clase ZendExt_Event. 48
Figura 11. Estructura del patrón Fachada. 49
Figura 12. Prototipo de interfaz de usuario del requisito Adicionar evento. 50
Figura 13. Prototipo de interfaz de usuario del requisito Adicionar trabajo. 50
Figura 14. Prototipo de interfaz de usuario del requisito Adicionar publicación. 51
Figura 15. Modelo de datos del sistema. 52
Figura 16. Diagrama de clases persistentes. 55
Figura 17. Diagrama de Componentes. 56
Figura 18. Representación en porciento agrupado en intervalos definidos de los resultados obtenidos tras aplicar el instrumento TOC. 59
Figura 19. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad. 59
Figura 20. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad. 60
Figura 21. Representación de la incidencia de los resultados de la evaluación del instrumento TOC en el atributo Reutilización. 60
Figura 22. Representación en porciento de la aplicación del instrumento RC en intervalos definidos. 61
Figura 23. Representación de la incidencia de los resultados de la evaluación del instrumento RC en el atributo Acoplamiento. 61
Figura 24. Representación de la incidencia de los resultados de la evaluación del instrumento RC en el atributo Complejidad de mantenimiento. 62

FIGURA 25. REPRESENTACIÓN DE LA INCIDENCIA DE LOS RESULTADOS DE LA EVALUACIÓN DEL INSTRUMENTO RC EN EL ATRIBUTO CANTIDAD DE PRUEBAS. 62
FIGURA 26. REPRESENTACIÓN DE LA INCIDENCIA DE LOS RESULTADOS DE LA EVALUACIÓN DEL INSTRUMENTO RC EN EL ATRIBUTO REUTILIZACIÓN 63
FIGURA 27. MÉTODO EXPORTPDFACTION (). 67
FIGURA 28. FRAGMENTO DE CÓDIGO DE LA FUNCIÓN *ADDTRABAJOACTION ()*. 68
FIGURA 29. DIAGRAMA DE DESPLIEGUE. 71
FIGURA 30. MÉTODO ADDEVENTOACTION (). 73

Índice de Tablas

Tabla 1. Descripción del requisito Adicionar evento. 43
Tabla 2. Descripción de la tabla dat_evento. 53
Tabla 3. Descripción de la tabla dat_publicación. 54
Tabla 4. Descripción de la tabla dat_trabajo. 55
Tabla 5. Atributos de calidad y modo en que afectan al instrumento TOC..... 57
Tabla 6. Criterios de evaluación para la métrica TOC 57
Tabla 7. Atributos de calidad y modo en que afectan al instrumento RC 58
Tabla 8. Criterios de evaluación para la métrica TOC. 58
Tabla 9. Descripción de la clase ControlctiController. 69
Tabla 10. Descripción de la clase DatTrabajoModel. 70
Tabla 11. Descripción de la clase DatTrabajo. 70
Tabla 12. Caminos básicos del flujo. 74
Tabla 13. Caso de prueba para el requisito Adicionar evento. 77

Introducción

En las sociedades modernas, los objetivos prioritarios de la política científica de los países son el desarrollo de la ciencia, la tecnología y la innovación, a través del fomento de la investigación, el desarrollo tecnológico y el fortalecimiento de la competitividad industrial. Para poder planificar, ejecutar y evaluar la actividad científica o técnica, se requiere necesariamente un trabajo estadístico previo de toma de datos básicos y posterior análisis de los mismos, para llegar a construir los necesarios indicadores de dicha actividad. [1]

La Universidad de las Ciencias Informáticas (UCI) como centro generador y productor de conocimiento, se ha dado a la tarea de medir de forma eficiente sus avances científicos, además promueve una estrecha interrelación docencia-investigación-producción, con el objetivo de elevar el nivel y la calidad de la propia docencia universitaria y de contribuir directamente a mejorar las condiciones económicas del país.

El Centro de Informatización de la Gestión de Entidades (CEIGE) cuenta con un claustro de profesionales dedicados al desarrollo de sistemas de software en cada una de sus áreas o departamentos, en aras de que el esfuerzo empeñado se materialice en productos integrales se exhorta constantemente por la superación profesional, a través de la investigación científica, y que los resultados arrojados sean presentados en eventos o conformen publicaciones de carácter científico.

Debido al cúmulo de información a archivar, teniendo en cuenta que el número de trabajos es significativamente grande y de los mismos es imprescindible conservar un mínimo de datos específicos como son título, autores y evidencia que lo certifica, se hace necesaria la existencia de un sistema que brinde la posibilidad de llevar un control de los indicadores relacionados con la ciencia, tecnología e innovación que se realiza en la Universidad de las Ciencias Informáticas y específicamente en el CEIGE.

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

archivos, bibliotecas, centros de documentación y permiten realizar búsquedas sofisticadas y recuperar información.

Actualmente el proceso de control de los parámetros de CTI del centro no aprovecha al máximo estas ventajas, el mismo se realiza de la siguiente manera: en cada departamento existe una persona responsable de registrar a través de alguna herramienta ofimática el accionar investigativo de cada profesional según su plan de resultados de CTI anual, cada una de estas recopilaciones es entregada a la sub-dirección de postgrado e investigación donde se resguarda a través de un proceso informatizado. Como consecuencia de esto en muchas ocasiones los jefes de departamento no cuentan con información actualizada de la participación de los profesionales en los diferentes eventos y/o redacción de publicaciones en el momento que lo requieran, generándose inconsistencia en la información entregada a la dirección de formación y postgrado del centro, que a su vez dificulta la correcta verificación de los objetivos CTI del año, teniendo que repetir este proceso en varias ocasiones, a su vez la verificación de los trabajos debe ser persistida como formato digital en espacios (discos duros) que no fueron pensados para almacenar toda la información derivada de este proceso de control.

A lo anterior se puede añadir que actualmente el Sistema de Indicadores de Ciencia, Tecnología e Innovación vigente en las instituciones y universidades del Ministerio de Educación Superior (MES) y el Ministerio de Ciencia Tecnología y Medio Ambiente de Cuba (CITMA) , no se adaptan a las condiciones existentes en la UCI, la cual presenta características que la diferencian de las demás universidades del país, la más importante de ellas consiste en que la formación curricular y el núcleo fundamental de las investigaciones científicas que se desarrollan se centran en las ramas de las Ciencias de la Informática y de la Computación [1], lo que dificulta la puesta en práctica de cualquier solución existente en el resto de las universidades relacionada con la gestión y control de los parámetros de CTI.

En la UCI y específicamente en el CEIGE parte de este proceso se realiza mediante una aplicación de escritorio. Dicha aplicación, no resuelve completamente el problema, ya que presenta una serie de desventajas que hacen que su uso no sea confiable. Entre las principales dificultades se destacan: Su base de datos está implementada en Access limitando la conexión desde un servidor público donde todos los usuarios interesados puedan interactuar con este, el sistema no es multiplataforma lo que dificulta su acceso a usuarios que utilicen el sistema operativo Linux, no posee ningún método que

garantice la seguridad de la información, ya que no requiere autenticación alguna y el sistema tampoco soporta la mayoría de los flujos necesario para realizar un buen control de CTI en el centro.

Ante la situación expuesta anteriormente surge como **problema a resolver**: ¿Cómo facilitar la gestión de la información relacionada con la actividad de Ciencia, Tecnología e Innovación que se desarrolla en el CEIGE?

Tomándose como **objeto de estudio:** Los Procesos de gestión de información de Ciencia, Tecnología e Innovación, para centrar la investigación se tiene como **campo de acción:** Los Procesos de gestión de información de Ciencia, Tecnología e Innovación en el CEIGE.

Para dar solución a la problemática descrita se establece como **objetivo general**: Desarrollar un sistema que facilite la gestión de la información relacionada con la actividad de CTI que se desarrolla en el CEIGE.

Se trazan como **objetivos específicos**:

- Elaborar el marco teórico-conceptual para fundamentar la investigación.
- Diagnosticar la situación actual en cuanto a la gestión de información relacionada con los parámetros CTI en CEIGE.
- Realizar un estudio y selección de las herramientas necesarias para la construcción del componente.
- Diseñar la solución de software para facilitar la implementación del componente de gestión de información CTI en CEIGE.
- Implementar el diseño de la solución de software para la obtención de un producto acorde a las necesidades.
- Validar el correcto funcionamiento del sistema mediante la aplicación de pruebas.

La investigación está basada en la siguiente **idea a defender:** Con la implementación de un sistema para la gestión de información de CTI se facilitará su manejo, seguimiento y control en el CEIGE.

Para cumplir con el objetivo general y dar solución a la situación problemática, se proponen las siguientes **tareas de la investigación**:

- Analizar los conceptos y tecnologías más utilizadas en el campo de los sistemas de gestión de información.
- Diseñar la interfaz de usuario.

- Descripción de los requisitos.
- Identificación de la metodología y herramientas a utilizar en el desarrollo de la aplicación.
- Elaborar el modelo conceptual de la aplicación.
- Elaborar diagramas de clases del diseño.
- Elaborar diagramas de secuencia.
- Proponer el estilo arquitectónico de la solución.
- Elaborar el diseño lógico de la base de datos.
- Elaborar el diseño físico de la base de datos.
- Desplegar la herramienta.

Métodos Teóricos:

- **Analítico-Sintético:** Este método fue empleado en el análisis de los documentos generados en el levantamiento y captura de requisitos, extrayendo y examinando los principales elementos relacionados con el objeto de estudio. Fue utilizado además en el análisis y selección de la metodología, las herramientas y tecnologías a utilizar. También fue empleado en el análisis de soluciones existentes para determinar características comunes y problemas que estos poseen con el objetivo de determinar los elementos que puedan ser empleados en la solución en cuestión.
- **Histórico-Lógico:** Este método se aplica para conocer los antecedentes y el estado actual de los sistemas informáticos para el análisis del código fuente, así como las causas de la evolución de los mismos.
- **Modelación:** Es usado en la realización de diagramas para expresar la información de manera organizada.

Métodos Empíricos:

- **Entrevista:** En este caso se utiliza este método para el proceso de entendimiento del negocio y los requerimientos que el software deberá satisfacer, además del proceso de familiarización con el personal capacitado sobre el tema a tratar.

El trabajo de diploma está definido estructuralmente en tres capítulos, descritos a continuación:

- **Capítulo 1 Fundamentación Teórica**: En este capítulo se describe el estado del arte de la investigación, se realiza el análisis de los conceptos fundamentales

relacionados con el tema en cuestión y se fundamenta sobre las técnicas de programación y herramientas utilizadas así como el modelo de desarrollo empleado.

- **Capítulo 2: Análisis y diseño del Sistema**: 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 definición del modelo conceptual y el modelo de clases, al igual que los métodos a desarrollar más significativos. Se lleva a cabo una explicación de la organización del sistema expuesta en el diagrama de clase para el correcto funcionamiento de este.
- **Capítulo 3: Implementación y prueba**: En este capítulo se describe la construcción de la solución, explicando los aspectos principales de la implementación. También se realiza una descripción de las clases y funcionalidades del sistema para evaluar la mantenibilidad, así como una muestra de los resultados obtenidos en la validación del sistema, además se hace el diseño y ejecución de pruebas sobre dicha aplicación en busca de errores relacionados con su funcionalidad.

Se espera como **posible resultado** de la investigación:
Contar con un sistema para la gestión de información de CTI de manera que facilite el acceso, seguimiento y control sobre los trabajos presentados por los profesionales en los diferentes eventos y publicaciones.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo, se abordan los principales conceptos relacionados con el proceso de gestión de información de los parámetros Ciencia, Tecnología e Innovación en el CEIGE que ayudan a comprender la solución propuesta. Se hace un análisis de los sistemas informáticos que se utilizan para evaluar dicho proceso y se explica en detalle el proceso de investigación llevado a cabo para la implementación del componente, además se fundamenta el uso de las herramientas y lenguajes de programación seleccionados; así como el modelo de desarrollo utilizado.

1.2 Conceptos básicos relacionados con el dominio del problema

Publicación científica

Publicación periódica que divulga artículos científicos y/o información de actualidad sobre investigación y desarrollo acerca de un campo científico determinado. Estas revistas contienen artículos originales inéditos que han pasado por revisión de pares, para asegurar que se cumple con las normas de calidad y validez científica. Su objetivo es comunicar el resultado de las investigaciones realizadas por personas o grupos que se dedican a crear ciencia.

¿Qué es una publicación científica primaria?

Una publicación científica primaria aceptable debe ser la primera divulgación y contener información suficiente para que los colegas del autor puedan: evaluar las observaciones, repetir los experimentos, evaluar los procesos intelectuales. Debe estar a la disposición de la comunidad científica sin restricciones y estar disponible también para su examen periódico. [2]

Evento científico

Es parte de un proceso que se inicia en la organización de las investigaciones, pasa por una producción científica y continua con un proceso de divulgación como una vía para introducir los resultados en la práctica social. El evento genera los conocimientos implícitos en los resultados de las ponencias, requiere de un modelo estratégico de gestión de conocimientos acorde con las necesidades de compartir el mismo, con el objetivo de acortar los plazos de introducción y generalización en la práctica social.

Un evento tiene objetivos, se desarrolla en un plazo de tiempo definido con un alcance previsto, consume recursos, dispone de un presupuesto, cuenta con el respaldo de los

patrocinadores, clientes y partes interesadas, se desarrolla sobre la base de los contratos, se rige por procedimientos y normas que garantizan la calidad del proceso, se rige por un sistema contable de gastos y satisface los requerimientos del financiamiento con los beneficios previstos.

Sistema de Información

Un sistema de información es un conjunto formal de procesos que, operando sobre una colección de datos estructurada según las necesidades de la empresa, recopilan, elaboran y distribuyen la información (o parte de ella) necesaria para las operaciones de dicha empresa y para las actividades de dirección y control correspondientes (decisiones) para desempeñar su actividad de acuerdo a su estrategia de negocio.

1.3 Sistemas de gestión

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

1.3.1 Gestión de la información

La información se ha convertido en la actualidad en la base del conocimiento y la vía fundamental que tienen las personas, las organizaciones y los países para comunicarse. La gestión de la información es un proceso que incluye operaciones como extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre la misma. Esta propicia la generación, apropiación, intercambio y uso de conocimientos necesarios para el incremento de la calidad de las organizaciones, además utiliza la información para el apoyo a la toma de decisiones, organizándola de manera tal que se cumplan los objetivos de la organización [4].

1.3.2 Tendencias de los sistemas de gestión de la información a nivel internacional

Universidad de Barcelona, España. (Sistema GREC)

GREC es una aplicación de gestión de investigación desarrollada por la Universidad de Barcelona, actualmente utilizada por diversas instituciones y organismos de investigación en España. GREC incluye un conjunto de base de datos como por ejemplo: Currículo Vítae (CV), grupos de investigación, proyectos y publicaciones [5].
Mediante GREC, un organismo, institución o empresa involucrada en actividades de corte científico puede obtener entre otros beneficios:

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

Sistema de Información Científica de Andalucía (SICA). España

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

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

Sistema de Gestión de la Investigación (SGI). Universidad de Talca en Chile

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

- Mantener actualizados programas y proyectos de investigación, proyectos de tesis y sus consiguientes resultados.
- Actualizar los estados en que se encuentran los proyectos y la disponibilidad de fondos concursales internos.

- Desplegar indicadores de gestión asociados a las capacidades y resultados de la investigación que se desarrolla en la Universidad de Talca.

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

Sistema de Gestión de Investigación en Línea (PGIL)

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

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

1.3.3 Experiencias en el uso de sistemas de gestión de la información científica en Cuba

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

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

De manera general, con esta aplicación se puede acceder a los listados actualizados de programas y proyectos nacionales, territoriales y ramales, de proyectos no asociados a programas, así como a los datos consolidados, series cronológicas y gráficos estadísticos de programas y proyectos en ejecución y terminados con información pública sobre la planificación, gerencia y participación de los territorios, organismos y entidades [9].

Sistema de Gestión de la Nueva Universidad (SIGENU)

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

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

1.3.4 Experiencias en el uso de sistemas de gestión de la información en la UCI

Sistema de Indicadores Cienciométricos (SIndiCIT)

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

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

Dentro de las funcionalidades que este presenta están: gestionar premios, gestionar capacitación, gestionar proyectos I+D, gestionar patentes y registros, gestionar empleo de estudiantes, mostrar resultados, generar reportes y generar gráficas.

Sistema de Gestión de Investigación de la facultad 2 de la Universidad de la Ciencias Informáticas

Es un sistema que tiene como objetivo principal automatizar los procesos de gestión de la información asociada a la investigación y el postgrado en esta facultad, logrando una mayor rapidez en el manejo de la información referente a ambos procesos.

Dentro de las funcionalidades que este presenta están: gestionar proyectos de investigación, gestionar cursos de postgrado, gestionar líneas de investigación, generar

reportes, gestionar grupos de investigación, gestionar líneas temáticas, realizar solicitud de cursos de postgrado y realizar solicitud de programas de postgrado.

Base de datos de acciones de CTI del centro CEIGE

Este sistema es el que se utiliza actualmente en la sub-dirección de investigación y postgrado del CEIGE para llevar a cabo el proceso de informatización del plan CTI. El sistema no garantiza un correcto funcionamiento para tener un buen seguimiento y control de los indicadores CTI en el centro debido a que presenta una serie de dificultades como son:

- No presenta una arquitectura cliente – servidor que permita la interacción de varios usuarios.
- El sistema no es multiplataforma, lo que dificulta la ejecución del mismo desde plataformas Linux.
- No provee ningún mecanismo de autenticación, gestión de roles y seguridad de los datos almacenados.
- Su base de datos no está soportada sobre tecnologías de software libre, en su lugar fue utilizado Microsoft Access.

1.3.5 Análisis de los sistemas de gestión presentados anteriormente

A pesar de los diferentes sistemas informáticos que se han identificado con el propósito de gestionar la actividad científica de una organización, no ha sido posible la adopción de ninguno de estos como solución al problema planteado, debido a que no incluyen las funcionalidades aceptar o rechazar un trabajo por el administrador, gestionar autores y gestionar jefes de departamentos, las cuales deben estar presentes en el sistema a desarrollar, además de que varias de las funcionalidades que estos contemplan no se corresponden con el proceso de seguimiento y control que se le desea dar a cada una de las problemáticas originadas en el CEIGE.

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

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

Por estas razones se propone el desarrollo de un sistema para la gestión de la información de los parámetros CTI en CEIGE, el cual tiene como particularidad que se ajusta sobre todo a los intereses y estrategias de trabajo de dicho centro.

1.4 Modelo de Desarrollo

Para el desarrollo de cualquier producto de software se realizan una serie de tareas entre la idea inicial y el producto final. Este proceso de desarrollo se hace muy difícil de controlar sin un modelo que establezca el orden en el que se realizarán las actividades planificadas para la elaboración del proyecto, por lo cual se hace imprescindible apoyarse en uno de ellos para llevar a cabo la construcción de un producto.

Para el desarrollo del presente trabajo de diploma se sigue el modelo de desarrollo propuesto por la Subdirección de Producción del CEIGE, denominado Modelo de Desarrollo del CEIGE, el cual define 2 fases por las que transitarán los proyectos de desarrollo de software.

1.4.1 Descripción de las fases del ciclo de vida

Inicio o Estudio preliminar: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel, la evaluación de la factibilidad del proyecto y el registro de este. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto.

Los objetivos de la fase son:

- Asegurar la factibilidad del proyecto.
- Establecer un plan para la ejecución del proyecto.

Hitos:

- Plan de desarrollo de software.
- Acta de inicio del proyecto firmada. [11]

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.

Hito:

- Producto liberado por 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. [11]

1.5 Marco de trabajo utilizado

Un marco de trabajo (framework) es una estructura de soporte definida mediante la cual otro proyecto de software puede ser desarrollado y organizado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Son diseñados con el intento de facilitar el desarrollo de software, permitiendo al programador abstraerse de un conjunto de detalles de bajo nivel para proveer un sistema funcional. [12]

Sauxe 2.0

El marco de trabajo Sauxe constituye un híbrido, de diferentes marcos de trabajo. Reúne en él las principales y mejores características de los framework: Doctrine, Zend Framework y ExtJs, proporcionando un entorno de trabajo agradable para el que desee utilizarlo. Utiliza ExtJs para implementar la capa de presentación, Zend framework para el desarrollo de la lógica del negocio y para la gestión de los datos utiliza Doctrine. Presenta una arquitectura cliente-servidor y tiene como patrón arquitectónico Modelo Vista Controlador (MVC).

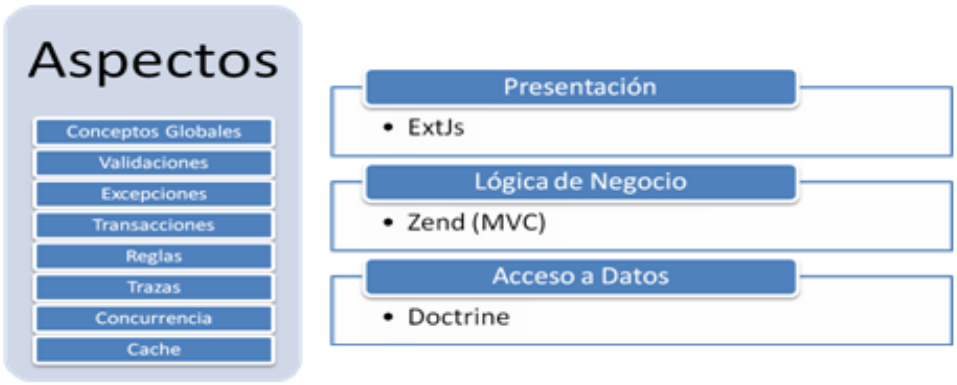


Figura 1. Arquitectura de Sauxe. [13]

1.5.1 Marcos de trabajo que utiliza Sauxe

ExtJs 2.2

ExtJs es una librería JavaScript para la programación del lado del cliente. Es una potente herramienta para crear las interfaces de usuarios de una aplicación Web. Incluye un conjunto de clases, y métodos que facilitan su vinculación con el lenguaje utilizado del lado del servidor. Ofrece gran cantidad de controles (widgets) para crear interfaces de usuario complejas. Sauxe incluye PHP-Ext, librería de código abierto que permite potenciar la capa de interfaz de usuario de JavaScript en las aplicaciones. Para ello ofrece una serie de librerías para integrar ExtJs en el sistema a desarrollar. Entre las posibilidades que ofrece se encuentran la creación de formularios, combos, paneles de tablas o menús, componentes utilizados en la implementación de la solución. Además ayuda a la comunicación entre el cliente y el servidor mediante JSON (notación de objetos de JavaScript) y XML (lenguaje de marcas extensible).Tiene un sistema dual de licencia: Comercial y Código Abierto (Open Source por sus siglas en inglés). En tiempo de ejecución carga y crea todos los objetos HTML a través del uso de DOM (Modelo de Objetos del Documento).

Ventajas de utilizar ExtJs:

- Código reutilizable.
- Independiente o adaptable a diferentes marcos de trabajo.
- Orientada a la programación de interfaces de tipo escritorio en la Web.
- Implementación basada en patrones de diseño.
- Amplia librería de componentes gráficos fácilmente extensibles. [14]

Zend Framework 1.9.8

Se trata de un marco de trabajo para el desarrollo de aplicaciones Web y servicios Web con PHP, brinda soluciones para construir sitios Web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. Este está formado por una serie de métodos estáticos y componentes que usarán estos métodos. Entre los componentes que se encuentran tienen vital importancia: Zend_Config para temas de configuración de aplicaciones Web, Zend_Db para tratar con base de datos, Zend_Search o Zend_Feed. [13]

Principales características:

- Implementa el patrón Modelo Vista Controlador.
- Completa documentación.

- Cuenta con módulos para manejar archivos PDF, canales RSS[1] y Servicios Web.
- Robustas clases para autenticación y filtrado de entrada. [13]

Doctrine 1.2.1

Doctrine es una librería para PHP que permite trabajar con un esquema de base de datos como si fuese un conjunto de objetos, y no de tablas y registros. Es un ORM (Object Relation Mapper), que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de base de datos pasan a ser clases y los registros objetos que se pueden manejar con facilidad.

Principales características de Doctrine

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

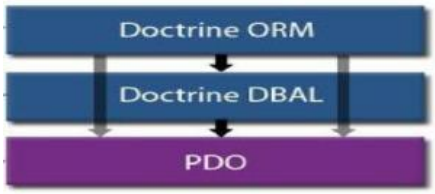


Figura 2. Estructura lógica del marco de trabajo Doctrine. [15]

Este marco de trabajo tiene la facultad de construir consultas a la base de datos relacional utilizando un lenguaje OO (orientado a objetos) denominado DQL (Doctrine Query Language o Lenguaje de Consultas Doctrine). Además, implementa un patrón CRUD (Crear, Recuperar, Actualizar y Eliminar) para operaciones comunes, ya sea desde crear un nuevo registro o actualizar los registros existentes [15] y dentro de los ORM que dispone PHP es uno de los más populares.

1.5.2 Lenguajes de programación utilizados

Un lenguaje de programación es el medio que permite a las computadoras, interpretar las órdenes que se les dan, así como controlar su comportamiento. Este consiste en un grupo de reglas sintácticas y semánticas que definen su estructura y el significado de

[1] Really Simple Syndication: Formato XML para sindicar o compartir contenido en la Web.

sus elementos respectivamente. A través de estas reglas el programador crea los programas o subprogramas.

Los lenguajes de programación para el desarrollo de aplicaciones Web, están divididos en dos grandes grupos, los lenguajes del lado del servidor y los lenguajes del lado del cliente.

Lenguaje del lado del servidor

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

PHP 5.2.6 (Hipertexto Pre-processor)

Es un lenguaje de programación interpretado de alto nivel, diseñado originalmente para la creación de páginas Web dinámicas y donde el código es ejecutado en el servidor, que es lo que distingue a PHP de la tecnología Java Script [16]. No es un lenguaje de marcas y su principal meta es permitir rápidamente a los desarrolladores la generación dinámica de páginas. Soporta el uso de otros servicios que usen protocolos como SNMP[2], HTTP[3] y derivados y permite generar documentos en PDF (documentos de Acrobat Reader).

Dentro de sus características podemos mencionar:

- Libre y abierto.
- Multiplataforma.
- Soporte para varios servidores Web.
- Fácil acceso a base de datos.
- Abundante documentación y fácil aprendizaje. [16]

Lenguaje del lado del cliente

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

Java Script

Java Script es el lenguaje de programación Web del lado del cliente más extendido. Es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho,

[2] Protocolo simple de Administración de Red.
[3] Protocolo de transferencia de hipertexto.

ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera y Mozilla Firefox. [17]

1.5.3 Arquitectura de Sauxe: Cliente-Servidor

La arquitectura Cliente/Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales [18]. Es un modelo basado en la idea del servicio, en el que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios. Además esta relación está establecida en función del intercambio de mensajes que es el único elemento de acoplamiento entre ambos. Los tres elementos fundamentales sobre los cuales se desarrollan e implantan los sistemas Cliente/Servidor son: el proceso cliente que es quien inicia el diálogo, el proceso servidor que pasivamente espera a que lleguen peticiones de servicio y el middleware[4] que corresponde a la interfaz que provee la conectividad entre el cliente y el servidor para poder intercambiar mensajes. Se caracteriza por existir un nodo (o más) servidor donde reside el servicio que se expone y varios clientes que consumen dicho servicio, en el sistema este estilo responde al hecho de que se trate de una aplicación Web y se concreta con un servidor de base de datos (PostgreSQL) un servidor Web (con función de servidor de aplicaciones, Apache 2) y varios clientes que acceden al sistema a través de un navegador. [19]

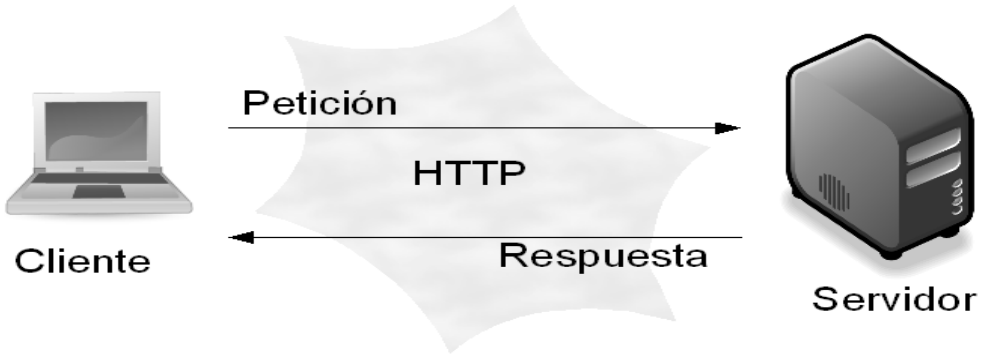


Figura 3. Arquitectura Cliente-Servidor.

[4]Software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones.

1.5.4 Patrón Arquitectónico

Un patrón arquitectónico representa, en términos de componentes y las relaciones entre ellos, posibles soluciones para incorporar en el diseño, aspectos que mejoran la usabilidad del sistema final. [20]

Modelo-Vista-Controlador (MVC)

En el desarrollo de la aplicación informática propuesta se utiliza el marco de trabajo Sauxe, el cual está basado en el patrón MVC. Este es un patrón que separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes el modelo, la vista y el controlador.

- Modelo: El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- Vista: Maneja la visualización de la información.
- Controlador: Recibe las entradas y las traduce a solicitudes de servicio para el modelo. Informa al modelo y/o a la vista para que cambien según resulte apropiado.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación de los sistemas jerárquicos en capas permite construir y probar el modelo independientemente de la representación visual.

En el sistema se concreta con la identificación de 3 elementos diferentes: la vista implementada en Java Script o HTML reside del lado del cliente en tiempo de ejecución, el Controlador, y el Modelo que junto al controlador reside del lado del servidor, la interacción entre la vista y el controlador se realiza a través de una solicitud AJAX o peticiones HTML y la respuesta dada por el controlador puede encontrarse en JSON, XML o HTML según corresponda la solicitud [19]. La finalidad del MVC es mejorar la reusabilidad por medio del desacople entre la vista y el modelo.

Ventajas de utilizar MVC

- Es posible tener diferentes vistas para un mismo modelo (ej. representación de un conjunto de datos como una tabla).
- Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.

- Proporciona un mecanismo de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción). [21]

Figura 4. Modelo Vista Controlador.

1.6 Lenguaje de modelado.

Lenguaje unificado de Modelado (UML 2.1)

El Lenguaje Unificado de Modelado (UML), es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML es un lenguaje de propósito general para el modelado orientado a objetos, es también un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes. [22]

Objetivos del UML:

- Es un lenguaje de modelado que puede ser usado por todos los modeladores.
- Incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno iterativo, basado en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso.
- Es un lenguaje universal. [22]

1.7 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.

1.7.1 Herramienta Case: Visual Paradigm 8.0

Las herramientas de Ingeniería de Software Asistida por Ordenador (Computer Aided Software Engineering, CASE por sus siglas en inglés) facilitan a los ingenieros de

software y analistas el desarrollo del software de principio a fin o en alguna de sus fases. Las mismas fueron creadas para automatizar las distintas etapas del producto y facilitar la organización de las tareas a cumplir durante su ciclo de vida. Aceleran el desarrollo de los sistemas y aumentan la calidad del software.

Visual Paradigm 8.0 for UML

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue [23]. Ayuda a una rápida construcción de aplicaciones de gran calidad, mejoras y a un menor costo de producción. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además, incluye una herramienta llamada Visual Architect que permite la generación de código para el manejo de la base de datos y puede generar códigos para lenguajes como PHP, Java y gestores de base de datos PostgreSql y MySQL.

Dentro de sus características se encuentran:

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

1.7.2 Entorno de desarrollo integrado (IDE)

Un entorno de desarrollo integrado, llamado también IDE (siglas en inglés Integrated Development de Environment) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, PHP, C#, Delphi, Visual Basic y Object Pascal. [24]

NetBeans 7.2.1

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento. Sun Microsystems fundó el proyecto de código

abierto NetBeans en junio 2000 y continúa siendo su patrocinador principal. Hoy en día hay disponibles dos productos: el entorno de desarrollo integrado NetBeans y la Plataforma NetBeans. Ambos productos son de código abierto y gratuito para uso tanto comercial como no comercial.

NetBeans es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas.

La Plataforma NetBeans es una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Funciona sobre disimiles sistemas operativos como Open Solaris, Linux, Windows y Mac OS. [25]

1.7.3 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. [26]

Apache es el servidor Web de suma confiabilidad por su robustez y estabilidad. Está diseñado para ser un servidor Web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Su diseño modular se adapta a una gran variedad de entornos, permitiendo a los administradores de sitios Web elegir qué características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. Es un software de libre distribución que publica su código fuente, lo que permite que cualquiera pueda modificarlo y colaborar así a su desarrollo. Tiene capacidad para servir páginas tanto de contenido estático, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante base de datos, ficheros u otras fuentes de información. [26]

Entre sus principales características se encuentran:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierta.
- Permite la configuración de ficheros de log, dándole al administrador un mayor control sobre lo que sucede en el servidor. [26]

1.7.4 Servidor de Base de Datos PostgreSQL 8.3.8

PostgreSQL es un Sistema de Gestión de Base de Datos Objeto-Relacionales (ORDBMS). Este está ampliamente considerado como el sistema de base de datos de

código abierto más avanzado del mundo [27]. El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto.

Entre sus principales características se encuentran:

- DBMS Objeto-Relacional: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas.
- Altamente Extensible: PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- Lenguajes Procedurales: PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. [27]

1.7.5 Navegador Web Mozilla Firefox 3.6

Es un navegador libre y de código abierto. Es usado para visualizar páginas Web. Incluye corrector ortográfico, búsqueda progresiva y marcadores dinámicos. Además se pueden añadir funciones a través de complementos desarrollados por terceros. Es multiplataforma, permite la integración con el antivirus, realiza la navegación por pestañas, presenta compatibilidad para múltiples extensiones y utiliza el sistema SSL para proteger la comunicación con los servidores Web, utilizando fuerte criptografía cuando se utiliza el protocolo HTTP. [28]

1.8 Conclusiones parciales

En este capítulo se definieron algunos conceptos necesarios para el entendimiento de la investigación científica en cuestión, se realizó un estudio de diferentes sistemas de gestión de la información, así como de varias herramientas, lenguajes y tecnologías a utilizar en el desarrollo del software, lo cual evidenció:

- Los sistemas expuestos no cumplen con todos los parámetros para cubrir las necesidades actuales del CEIGE.

- La aplicación debe ser desarrollada basándose en el modelo, las herramientas y las tecnologías antes seleccionadas, puesto que son las idóneas dentro de la plataforma tecnológica que ofrece el centro teniendo en cuenta las características del sistema.

Por tanto se determinó la necesidad de crear un sistema propio que responda a las características particulares del CEIGE y que posibilite la gestión y el control de la información referente a los parámetros CTI en el mismo.

Capítulo 2: Análisis y diseño del sistema

2.1 Introducción

En este capítulo se presentan los primeros flujos de trabajo propuestos por el Modelo de Desarrollo de Software del CEIGE para la confección de la solución propuesta: Requisitos y Análisis y Diseño. En la etapa de Requisitos se identifican y describen los requisitos funcionales y no funcionales con los que debe cumplir el sistema, además se hace referencia a las técnicas empleadas tanto en la captura como en la validación de los mismos. En el Análisis y el Diseño, se definen todos los artefactos propuestos por el modelo para cada caso. Se exponen además los patrones de diseño utilizados en la solución propuesta.

2.2 Propuesta del sistema

El sistema que se propone es una aplicación Web que garantiza el manejo, seguimiento y control de los parámetros de CTI en el CEIGE. Cuenta con funcionalidades que se encargan de gestionar, evaluar y controlar los eventos y publicaciones científicas realizadas por los profesionales del centro. Brinda la posibilidad de registrar y almacenar trabajos presentados en algún espacio con todos sus datos como son: la fecha en que se expusieron, los autores que lo componen y el área a la que pertenece. Este sistema también tiene la posibilidad de visualizar los reportes que tengan cada publicación o evento registrado en la aplicación.

2.3 Modelo conceptual

Los modelos de dominio o modelos conceptuales se construyen a partir de las definiciones fundamentales que se van a tratar a lo largo de la investigación. En ellos se representa cómo se relacionan dichos conceptos entre sí para hacer más fácil su entendimiento y contribuir con esto a la solución del problema. [29]

Definición de las entidades y conceptos principales

Espacio: Lo común entre evento y publicación. Puede ser un evento o una publicación.

Evento: Espacio donde se exponen o presentan trabajos.

Tipo de Evento: Característica que tiene un evento.

Publicación: Espacio donde se presentan trabajos.

Tipo de Publicación: Característica que tiene una publicación.

Trabajo: Documento donde se describe algún tema.

Tipo de Trabajo: Característica que tiene un trabajo.

Autor: Usuario que representa un trabajo.

Jefe de Departamento: Usuario que controla todo lo referente a un departamento.

Departamento: Área a la que pertenecen varios autores.

País: Lugar donde se hace un evento o se realiza una publicación.

Archivo: Formato duro de un trabajo.

En la Figura 5 se muestra el modelo conceptual que se generó para el sistema de gestión de la información de CTI en el CEIGE.

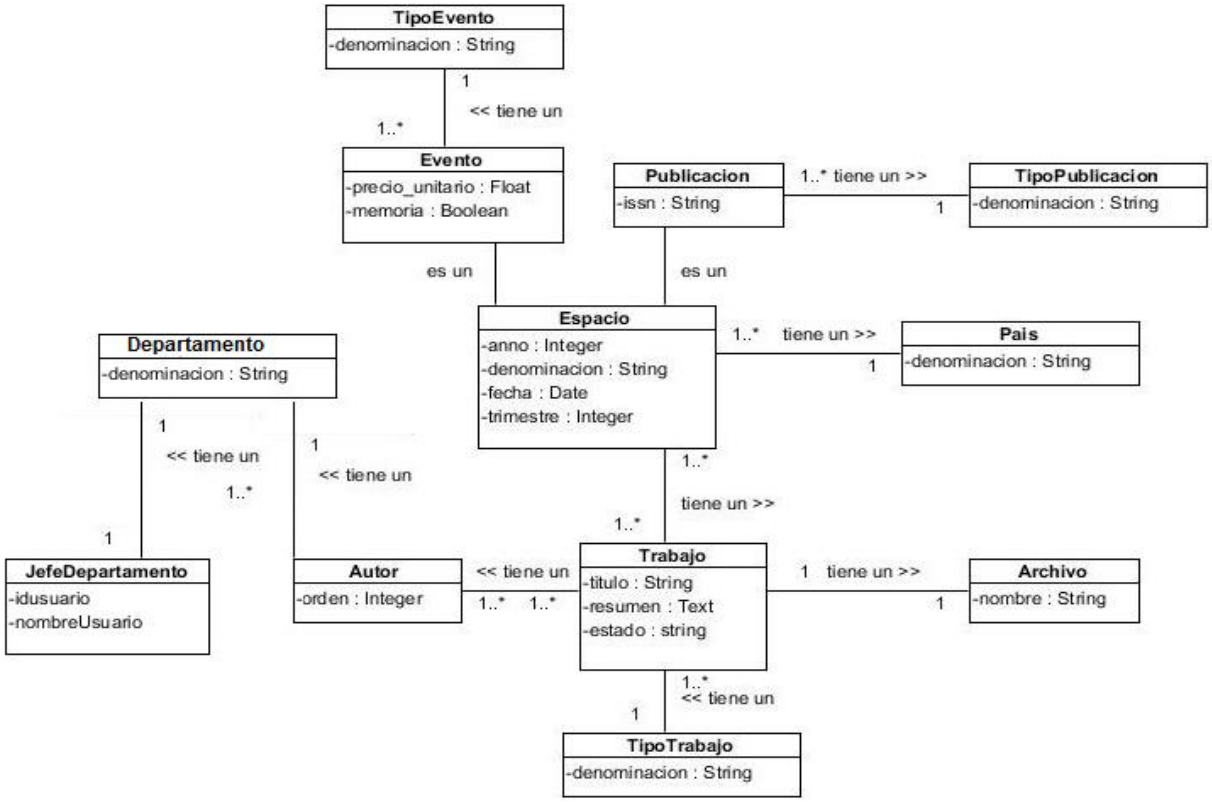


Figura 5. Modelo Conceptual.

2.4 Especificación de los requisitos de software

Con la especificación de los requisitos de software se obtiene una descripción detallada de las necesidades de un producto informático. [30]

Para realizar la captura de los requisitos se hizo necesaria la utilización de algunas técnicas existentes con este fin. Entre las técnicas para la captura de requisitos utilizadas se encuentra la **entrevista**, aplicada al cliente para el cuál va a ser desarrollada la aplicación y **tormentas de ideas**, técnica encargada de posibilitar debates y talleres donde el cliente explicó de forma explícita sus necesidades.

Los requisitos se pueden clasificar en:

- Funcionales
- No funcionales

2.4.1 Requisitos funcionales

Los requisitos funcionales no son más que capacidades o condiciones que el sistema debe cumplir, estos especifican comportamientos particulares de un sistema.

A continuación se listan los requisitos funcionales identificados que debe cumplir el sistema.

RF1. Gestionar evento:

- RF1.1 Adicionar evento.
- RF1.2 Modificar evento.
- RF1.3 Eliminar evento.
- RF1.4 Listar eventos.

RF2. Gestionar tipo de evento:

- RF2.1 Adicionar tipo de evento.
- RF2.2 Modificar tipo de evento.
- RF2.3 Eliminar tipo de evento.
- RF2.4 Listar tipos de eventos.

RF3. Gestionar publicación:

- RF3.1 Adicionar publicación.
- RF3.2 Modificar publicación.
- RF3.3 Eliminar publicación.
- RF3.4 Listar publicaciones.

RF4. Gestionar tipo de publicación:

- RF4.1 Adicionar tipo de publicación.
- RF4.2 Modificar tipo de publicación.
- RF4.3 Eliminar tipo de publicación.
- RF4.4 Listar tipo de publicaciones.

RF5. Gestionar trabajo:

- RF5.1 Adicionar trabajo.
- RF5.2 Modificar trabajo.
- RF5.3 Eliminar trabajo.
- RF5.4 Listar trabajos.

RF6. Gestionar tipo de trabajo:

- RF6.1 Adicionar tipo de trabajo.
- RF6.2 Modificar tipo de trabajo.
- RF6.3 Eliminar tipo de trabajo.
- RF6.4 Listar tipos de trabajos.

RF7. Gestionar autor:

- RF7.1 Adicionar autor.
- RF7.2 Modificar autor.
- RF7.3 Eliminar autor.
- RF7.4 Listar autores.

RF8. Gestionar departamento:

- RF8.1 Adicionar departamento.
- RF8.2 Modificar departamento.
- RF8.3 Eliminar departamento.
- RF8.4 Listar departamentos.

RF9. Gestionar jefe de departamento:

- RF9.1 Adicionar jefe de departamento.
- RF9.2 Modificar jefe de departamento.
- RF9.3 Eliminar jefe de departamento.
- RF9.4 Listar jefes de departamentos.

RF10. Gestionar país:

- RF10.1 Adicionar país.
- RF10.2 Modificar país.
- RF10.3 Eliminar país.
- RF10.4 Listar países.

RF11. Generar Reporte

2.4.2 Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que debe tener el producto. Estas propiedades son las características que hacen al producto atractivo, usable, rápido o confiable. Entre los requisitos no funcionales del sistema propuesto se encuentran:

Usabilidad

- El acceso al sistema debe ser fácil y rápido.
- Todos los mensajes de error del sistema deberán incluir una descripción textual del error.
- Las etiquetas de cada funcionalidad y los campos de cada interfaz tendrán títulos asociados a su función de negocio.

Soporte

- La aplicación recibirá mantenimiento en el período de tiempo determinado por el equipo de desarrollo y el cliente.

Rendimiento

- Teniendo en cuenta que el producto se debe diseñar sobre una arquitectura cliente - servidor, los tiempos de respuestas del sistema deben ser rápidos, al igual que la velocidad de procesamiento de la información para lograr respuestas rápidas del mismo.

Interfaz

- El sistema debe contar con una interfaz fácil de usar, sencilla, amigable, permitiendo que los usuarios sean capaces de interactuar con la aplicación.
- Será diseñada para adaptarse a la resolución del usuario, utilizando colores refrescantes, agradables y se emplearán imágenes identificadas con el negocio donde se utilizará el sistema.

Portabilidad

- El sistema debe ser compatible con los sistemas operativos Windows y Linux.

Seguridad

- El sistema debe verificar que el usuario esté autenticado antes de que pueda realizar alguna acción sobre el sistema.
- La información manejada por el sistema estará protegida de acceso no autorizado y divulgación.
- El sistema debe mostrar las funcionalidades de acuerdo a los permisos del usuario que esté activo.

Integridad:

- La información manejada por el sistema será objeto de cuidadosa protección contra corrupción y estados inconsistentes.

Disponibilidad:

- La información se encontrará disponible en todo momento para aquellos usuarios autorizados a acceder al sistema.

Software:

Para el cliente:

- Navegador Mozilla Firefox 3.0 o superior.
- Sistema operativo: Linux o Windows.

Para el servidor:

- Un servidor Web Apache 2.2 o superior.
- Gestor de base de datos PostgreSQL 8.3 o superior.
- PHP 5.0 0 superior.

Hardware:

Para el cliente:

- Requerimientos mínimos: Procesador Pentium IV a 1.40 GHZ con 256 Mb de memoria RAM (Recomendado 512).
- Tarjeta de red.

Para el servidor:

- Requerimientos mínimos: Procesador Dual Core a 3.00 GHz de velocidad de procesamiento, 1Gb de memoria RAM y una capacidad de 40gb de disco duro.
- Tarjeta de red.

Aplicación de Estándares:

- Utilizar los estándares establecidos (codificación, diseño de interfaz de usuario, diseño de datos, entre otros).
- Emplear como servidores Web y de base de datos Apache y PostgreSQL respectivamente.
- Utilizar como lenguaje del lado del servidor al PHP 5.0 o superior y del lado del cliente el JavaScript.

2.4.3 Descripción detallada de los requisitos

A continuación se muestra la descripción detallada del requisito adicionar evento. Las descripciones de los requisitos restantes se encuentran en el Expediente de proyecto**.**

Especificación de requisito: Adicionar Evento

Precondiciones		El usuario se ha autenticado en el sistema y debe tener permiso para realizar esta acción.
Flujo de eventos		
Flujo básico		
1	Se introducen los datos del evento: Denominación Fecha Año Tipo de evento Precio unitario País Memoria Trimestre	
2	El sistema valida (ver validación 1) los datos introducidos.	
3	Si los datos son correctos el sistema los registra.	
4	El sistema confirma el registro de los datos.	
5	Concluye el requisito.	
Pos-condiciones		
1	Se registró en el sistema un nuevo evento.	
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		
1	Se validan los datos según lo establecido en el modelo conceptual.	
Relaciones	**Requisitos Incluidos**	N/A.
	Extensiones	N/A.
Conceptos	**Publicación**	Visibles en la interfaz: Denominación Fecha

	Año Tipo de evento Precio unitario País Memoria Trimestre Utilizados internamente: N/A.
Requisitos especiales	N/A.
Asuntos pendientes	N/A.

Tabla 1. Descripción del requisito Adicionar evento.

2.5 Validación de requisitos

Esta actividad se realizó con el objetivo de garantizar que los requisitos fueran correctos y cumplieran con las necesidades del cliente. Las técnicas que se utilizaron para la validación de los requisitos identificados fueron las siguientes:

- Validación de requisitos mediante prototipos de interfaz y escenarios.

Se presentaron los prototipos de interfaz elaborados durante la especificación a especialistas funcionales, para corroborar su correspondencia con las necesidades y aspiraciones del cliente. Para ello, se desarrollaron varios escenarios posibles con el auxilio de juegos de datos, de forma tal que se visualizaran las diferentes funcionalidades que tendría el sistema. Las no conformidades fueron documentadas y corregidas.

- Revisión técnica de artefactos.

Se realizaron revisiones de los artefactos por parte del equipo de calidad del proyecto y fueron corregidas las no conformidades identificadas para la liberación de la documentación.

2.6 Análisis y Diseño

Para que el diseño, la implementación, la gestión de la información y el conocimiento de un proyecto sean adecuados y eficaces, el análisis es clave pues permite estudiar, interpretar y comprender con mayor facilidad y exactitud los requisitos descritos con anterioridad en su captura.

Durante esta fase es modelado el sistema para que soporte todos los requisitos. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la próxima fase. Los artefactos generados en esta etapa son más formales y específicos de una implementación. En caso de llevarse a cabo la reutilización de componentes

software ya desarrollados, durante esta fase se ajusta el modelado existente a los requisitos actuales. [11]

El resultado obtenido de la etapa de diseño facilita enormemente la implementación posterior del sistema, proporcionando su estructura básica, así como los diferentes componentes que actúan y se relacionan entre sí. [31]

2.6.1 Diagrama de clases del diseño

El diseño de los diagramas de clases permite describir la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases se utilizan durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se maneja en el sistema, y los componentes que se encargan del funcionamiento y la relación entre uno y otro.

A continuación se muestra el diagrama de clase del diseño del requisito funcional Gestionar evento (Figuras 6). Los diagramas de clases del diseño de los requisitos restantes se encuentran en el Expediente de proyecto.

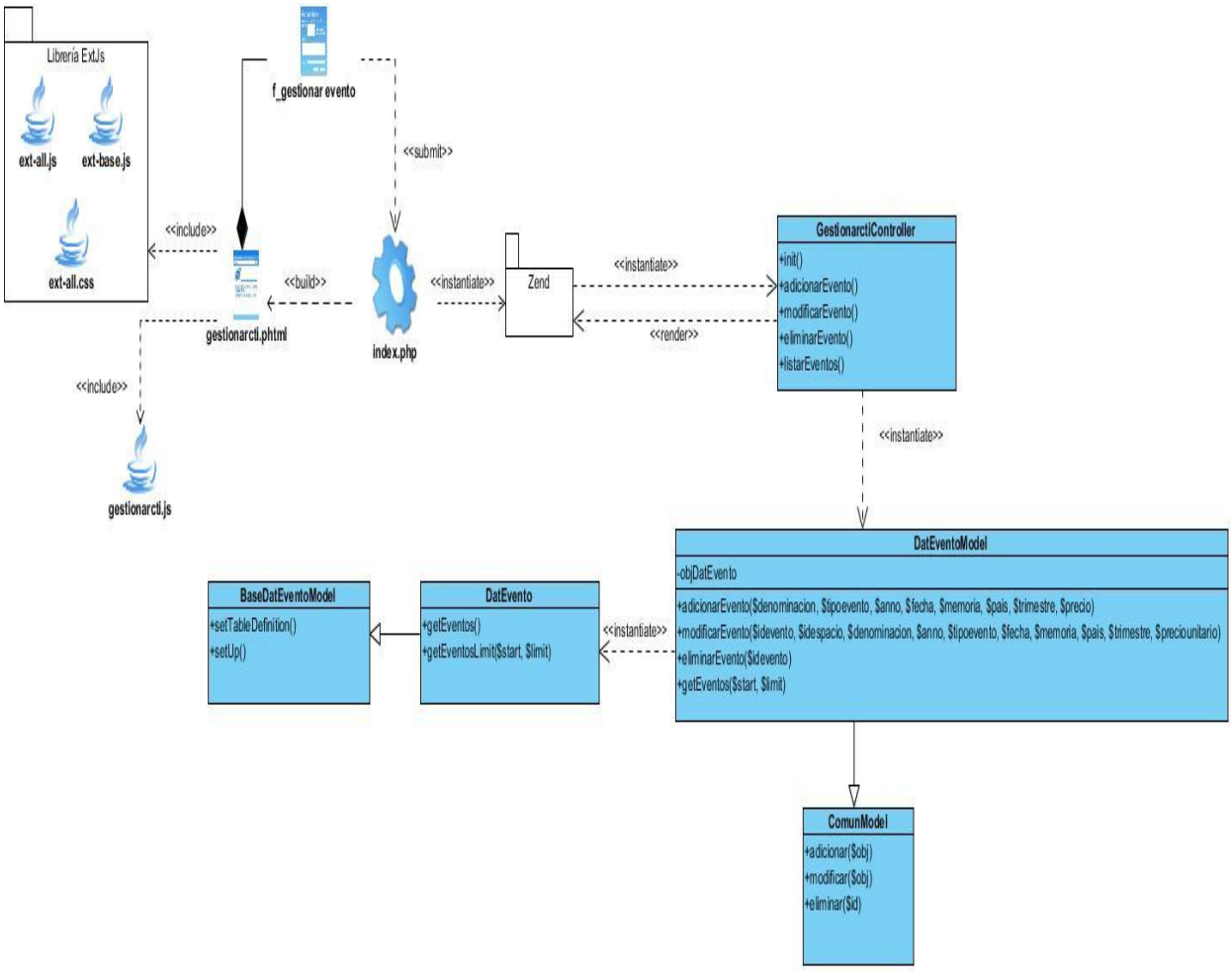


Figura 6. DCD Gestionar Evento.

En la figura 6 se muestra el diagrama de clases del diseño del requisito Gestionar Evento, en el cual se representan las principales clases, operaciones y relaciones que se necesarias para darle cumplimiento a dicho requisito, donde la capa arquitectónica de presentación está formada por las clases **gestionarCTI.js** y **gestionarCTI.phtml**. La comunicación entre la vista y el modelo es realizada por la clase **GestionarCTIController**, la clase **DatEventoModel** es la encargada de la lógica del negocio, implementando funcionalidades que garantizan el cumplimiento del requisito identificado, encargadas del acceso a los datos se encuentra **DatEvento** así como la clase **BaseDatEventoModel** de la cual extiende.

2.6.2 Diagrama de secuencia

El diagrama de secuencia, representa la forma en que un cliente (actor) u objetos (clases) se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente.

A continuación se muestra el diagrama de secuencia del requisito funcional Adicionar evento. Los diagramas de secuencia de los requisitos restantes se pueden encontrar en el Expediente de proyecto.

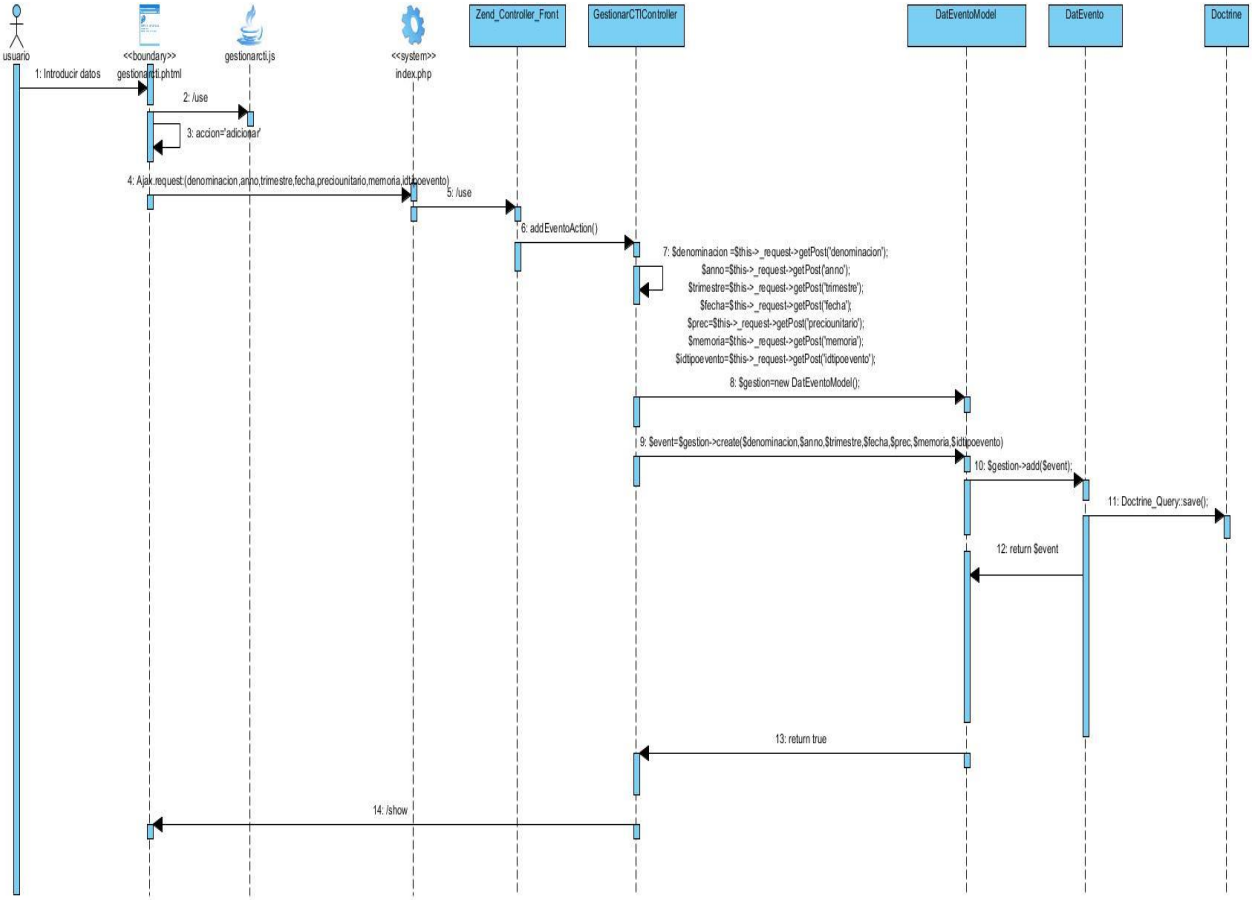


Figura 7. Diagrama de secuencia Adicionar Evento.

2.6.3 Patrones de Diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema particular dentro de un contexto específico. Los patrones de diseño no son más que soluciones a problemas recurrentes en el entorno de desarrollo, que pueden facilitar el trabajo en muchas situaciones a la hora de implementar una aplicación. [32]

Los patrones de diseño se dividen en dos grupos principales, los patrones GRASP (patrones para la asignación de responsabilidades) y los patrones GOF, estos últimos se clasifican de la siguiente forma:

- **Patrones creacionales:** Los patrones creacionales proporcionan ayuda a la hora de crear objetos, principalmente cuando esta creación requiere tomar decisiones. Un patrón de creación asociado a clases usa la herencia para variar la clase que se instancia, mientras que un patrón de creación asociado a objetos delegará la instanciación a otro objeto. [32]
- **Patrones estructurales:** Los patrones estructurales están relacionados con cómo las clases y los objetos se combinan para dar lugar a estructuras más complejas. [32]
- **Patrones de comportamiento:** Estos patrones de diseño están relacionados con algoritmos y asignación de responsabilidades a los objetos. Los patrones de comportamiento describen no solamente patrones de objetos o clases, sino también patrones de comunicación entre ellos. [32]

Patrones GRASP (General Responsibility Assignment Software Patterns)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. A continuación se describen los patrones GRASP utilizados en el diseño del componente.

Patrón Experto: Asignación de responsabilidades a las clases para que cada objeto realice la funcionalidad de acuerdo a la información que domina, la cuestión a la hora de diseñar es asignar responsabilidades a la clase que mayor información posee para cumplir con dicha tarea. Cada clase cuenta con un grupo de funcionalidades que las convierte en experta de la información de la tabla a la que representa. Este patrón se puede observar en el diseño del sistema en las clases Dat, ej. DatEventoModel o DatTrabajoModel la cuales son las expertas en realizar las funcionalidades correspondientes a cada requisito.

Patrón Creador: Se utiliza para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto

sólo pueda ser creada por el objeto que contiene la información necesaria para ello. Este patrón se evidencia en la clase (class DatTrabajoModel extends DatTrabajo). Dentro de esta existen varias funcionalidades en las cuales se crea un objeto o instancia de la clase DatTrabajo.

Patrón Controlador: Se aplica para asignar la responsabilidad del manejo de los eventos del sistema y definir sus operaciones, es el intermediario entre una determinada interfaz y el algoritmo que la implementa. Zend Framework contribuye a la utilización de dicho patrón ya que define un controlador frontal que implica que todas las solicitudes son dirigidas a un único script PHP que se encarga de instanciarlo y redirigir las llamadas. Este patrón se evidencia en la clase controladora ControlctiController que es la encargada de definir las principales operaciones que se hacen en el sistema.

```
 9  class ControlctiController extends ZendExt_Controller_Secure {
10      function init() {...}
13      function controlctiAction() {...}
16      function getEventosAction() {...}
24      function getPublicacionesAction() {...}
36      function getTrabajosEventoAction() {...}
39      function getTrabajosPublicacionesAction() {...}
42      function getTipoEventoAction() {...}
46      function getTipoPubAction() {...}
50      function getPaisAction() {...}
54      function addEventoAction() {...}
74      function modEventoAction() {...}
96      function eliminarEventoAction() {...}
106     function addPublicacionAction() {...}
126     function modPublicacionAction() {...}
129
130 }
```

Figura 8. Clase ControlctiController.

Alta Cohesión: Sauxe presenta entre sus principales características la organización del trabajo en cuanto a estructura y responsabilidades bien definidas, esto permite que se trabaje con las clases con una alta cohesión. Un ejemplo de esto es la clase (class ControlctiController extends ZendExt_Controller_Secure), la cual delega funciones específicas a otras clases, encargándose solo de definir las operaciones a realizar.

Patrón Bajo Acoplamiento: El acoplamiento mide la fuerza con que una clase está conectada a otra, de esta forma una clase con bajo acoplamiento debe tener un número mínimo de dependencia con otras clases. Este patrón se tiene en cuenta para asignar una responsabilidad permitiendo que el acoplamiento permanezca bajo. Este patrón se evidencia en el marco de trabajo Sauxe ya que dentro de la capa modelo, las clases de

abstracción de datos son las más reutilizables y no tienen asociaciones con las clases de la capa de la vista ni con el controlador, dependiendo solo de sus clases base las que contienen los atributos correspondientes a su tabla en la base de datos.

```
3   class DatEvento extends BaseDatEvento
4   {
5       public function setUp()
6       {
7           parent :: setUp ();
8           $this->hasOne ('NomTipoevento', array ('local' => 'idtipoevento', 'foreign' => 'idtipoevento'));
9           $this->hasOne ('DatEspacio', array ('local' => 'idespacio', 'foreign' => 'idespacio'));
10      }
```

Figura 9. Clase DatEvento que depende de la clase BaseDatEventoModel.

Patrones GOF (Gang or Four)

Singleton (Instancia única): Permite la existencia de una única instancia para una clase permitiendo un mayor rendimiento de la misma y consistencia del código fuente. Zend Framework posee una instancia única del controlador frontal disponible mediante este patrón para lograr una vía de entrada única a las solicitudes. Entre sus ventajas destacan que es fácilmente modificable para permitir más de una instancia y, en general, para controlar el número de las mismas (incluso si es variable), además se reduce el espacio de nombres (frente al uso de variables globales).

```
1   <?php
2       class ZendExt_Event {
3           static private $_instance;
4           static private $_xml;
5
6           private function __construct() {
7               self :: $_xml = ZendExt_FastResponse::getXML('events');
8           }
9
10          static function getInstance () {
11              if (! self::$_instance)
12                  self::$_instance = new self ();
13
14              return self::$_instance;
15          }
```

Figura 10. Ejemplo de implementación del patrón singleton en la clase ZendExt_Event.

Facade (Fachada): Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar. Este patrón simplifica los accesos a las clases de la capa de acceso a datos proporcionando un objeto que todas las clases de capas superiores utilizarán para acceder a las clases contenidas en la capa del modelo. Se utiliza una clase

intermediaria entre la clase controladora de Zend Framework y las clases de acceso a datos de Doctrine, la que brindará, de las operaciones de acceso a datos, solo las que necesiten los controladores para su funcionamiento, lo que reduce la dependencia entre clases.

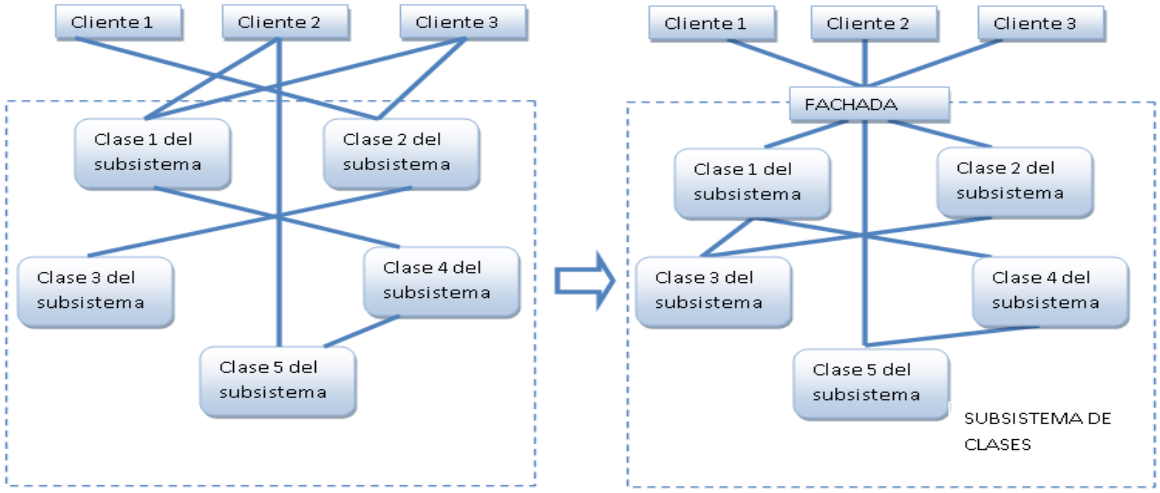


Figura 11. Estructura del patrón Fachada.

Patrón Inversión de Control (IoC)

Este es un patrón de diseño pensado para permitir un menor acoplamiento entre los componentes de una aplicación y fomentar así el uso de los mismos [33]. Este patrón es utilizado por el marco de trabajo Sauxe para crear y brindar servicios. Cada módulo del proyecto cuenta con un esquema propio en la base de datos y ninguno puede acceder directamente al otro, en el caso que un módulo necesite acceder al esquema de otro, lo que se hace es crear y brindar un servicio para que el que lo necesite lo consuma. La utilización de este patrón en el sistema está reflejada en la creación y empleo de la clase IOC, ya que por la necesidad de utilizar los servicios getUsuarios () y cargarAcciones () del componente seguridad, era necesaria la implementación de una clase donde fueran publicados los mismos, facilitando su interacción.

2.6.4 Prototipo de interfaz de usuario

Un prototipo es una visión preliminar del sistema futuro, es un modelo operable, fácilmente ampliable y modificable, que tiene todas las características que hasta el momento debe tener el sistema.

Las ventajas principales de los prototipos son:

- Posibilidad de cambiar el modelo.
- Oportunidad para suspender el desarrollo del modelo si no es funcional.
- Posibilidad de crear un nuevo modelo que se ajuste mejor a las necesidades y expectativas del usuario.

Para el desarrollo de la aplicación se ha decidido crear una interfaz de usuario amigable y sencilla que permita a los usuarios interactuar con el sistema sin necesitar un conocimiento profundo.

A continuación se muestran los prototipos de 3 interfaces del software, la interfaz del requisito adicionar evento, la interfaz del requisito adicionar trabajo y la del requisito adicionar publicación. Los prototipos de interfaces de los restantes requisitos se encuentran en el Expediente de proyecto.

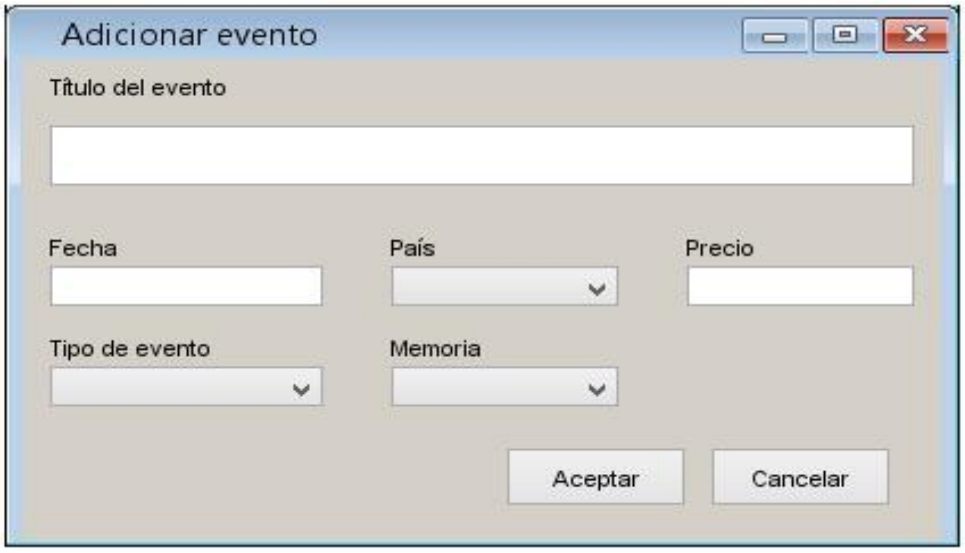


Figura 12. Prototipo de interfaz de usuario del requisito Adicionar evento.

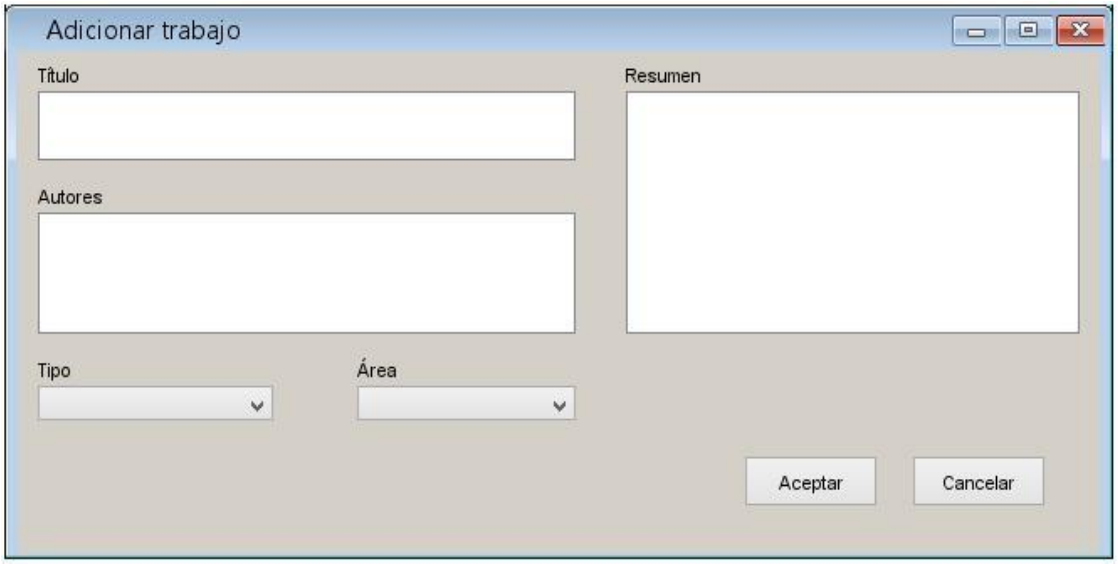


Figura 13. Prototipo de interfaz de usuario del requisito Adicionar trabajo.

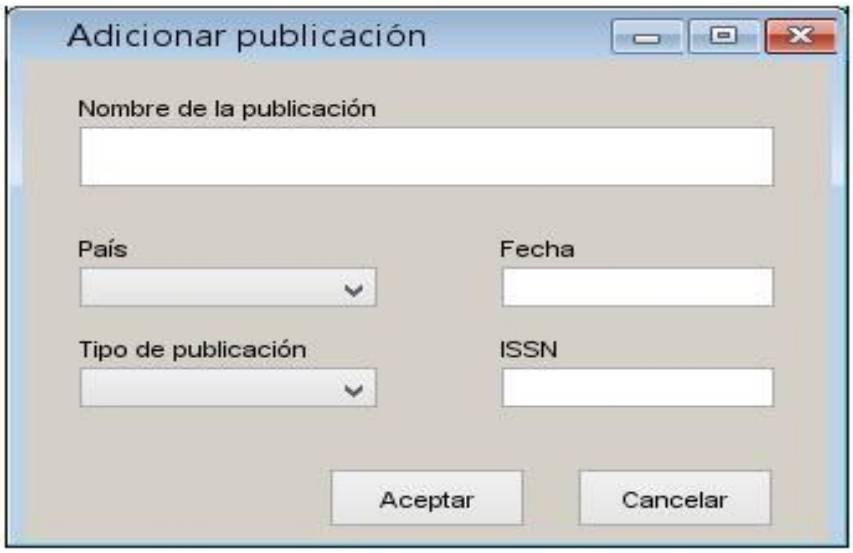


Figura 14. Prototipo de interfaz de usuario del requisito Adicionar publicación.

2.6.5 Modelo de Datos

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

A continuación se presenta el modelo de datos del sistema, el cual cuenta con 13 tablas.

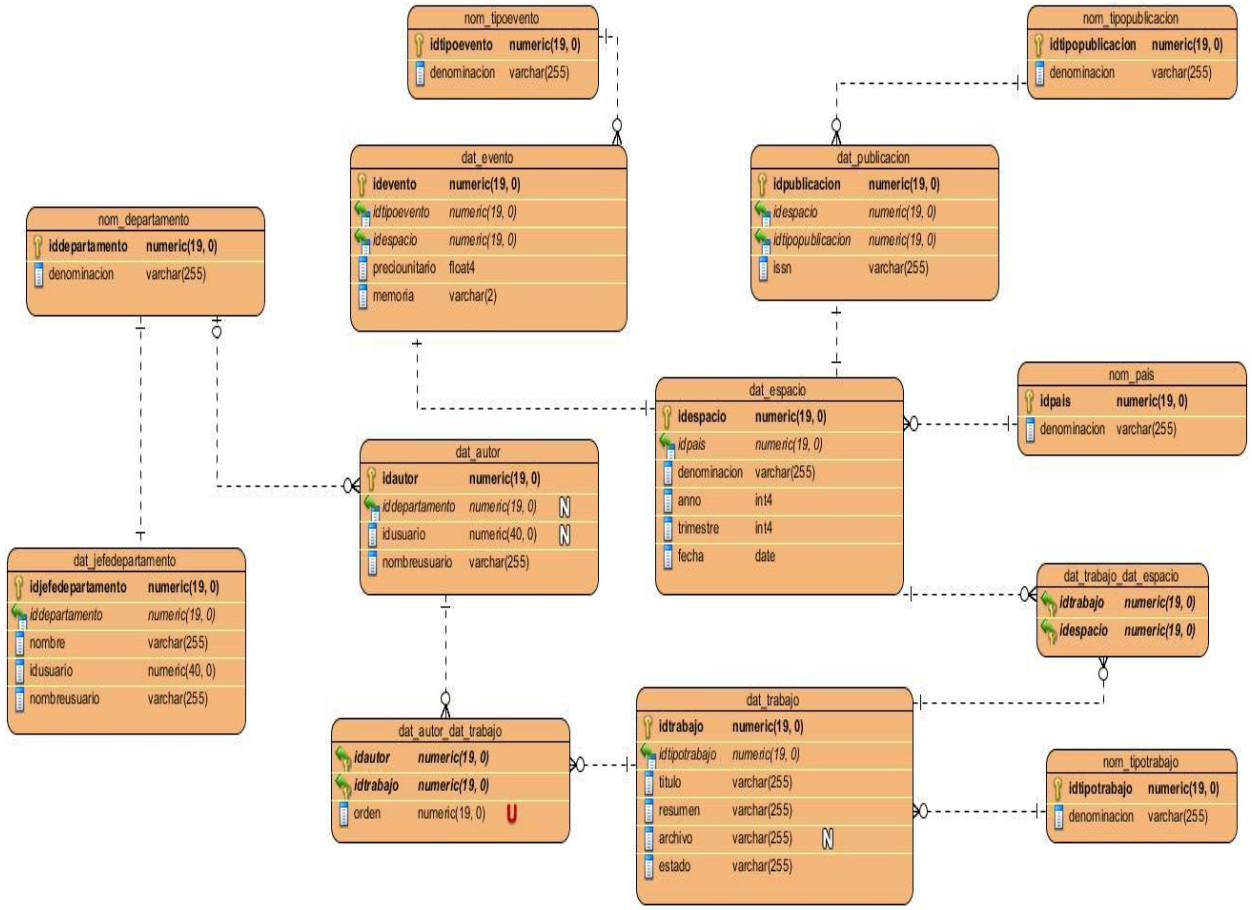


Figura 15. . Modelo de datos del sistema.

Descripción de las tablas

A continuación se muestran las descripciones de las principales tablas del modelo de datos del sistema. Las descripciones de las tablas restantes se encuentran en el Expediente de proyecto**.**

Nombre: dat_evento		
Atributo	**Tipo**	**Descripción**
idevento	numeric	Identificador de la tabla.
idtipoevento	numeric	Identificador de la tabla nom_tipoevento
idespacio	numeric	Identificador de la tabla dat_espacio.

preciounitario	float	Valor que identifica el precio unitario del evento
memoria	varchar	Cadena de caracteres que muestra si el evento tiene memoria(Si o No)
Tablas Relacionadas	**Descripción de la relación**	
nom_tipoevento	Uno a muchos.	
dat_espacio	Uno a uno.	
Descripción de la tabla Almacena los datos correspondientes a los eventos.		

Tabla 2. Descripción de la tabla dat_evento.

Nombre: dat_publicación		
Atributo	**Tipo**	**Descripción**
idpublicación	numeric	Identificador de la tabla.
Idtipopublicación	numeric	Identificador de la tabla nom_tipopublicación.
idespacio	numeric	Identificador de la tabla dat_espacio.
issn	varchar	Valor que identifica el número correspondiente al issn.
Tablas Relacionadas	**Descripción de la relación**	
nom_tipopublicación	Uno a muchos.	
dat_espacio	Uno a uno.	

Descripción de la tabla
Almacena los datos correspondientes a las publicaciones.

Tabla 3. Descripción de la tabla dat_publicación.

Nombre: dat_trabajo		
Atributo	**Tipo**	**Descripción**
idtrabajo	numeric	Identificador de la tabla.
idtipotrabajo	numeric	Identificador de la tabla nom_trabajo.
título	varchar	Cadena de caracteres que identifica el título correspondiente al trabajo.
resumen	varchar	Cadena de caracteres donde se resume el trabajo.
archivo	varchar	Cadena de caracteres donde se guarda la URL del trabajo.
estado	varchar	Cadena de caracteres que muestra el estado del trabajo(Nuevo, Rechazado, Aceptado o Certificado)
Tablas Relacionadas	**Descripción de la relación**	
nom_tipotrabajo	Uno a muchos.	
dat_trabajo_dat_espacio	Uno a muchos.	
dat_autor_dat_trabajo	Uno a muchos.	
Descripción de la tabla Almacena los datos correspondientes los trabajos.		

Tabla 4. Descripción de la tabla dat_trabajo.

2.6.6 Diagrama de clases persistentes

Las clases persistentes son aquellas clases que tienen un mayor tiempo de vida, más allá del tiempo de ejecución de la aplicación. Generalmente las clases persistentes coinciden con los conceptos de información que se usan en el sistema. En la figura 15 se observa el diagrama de clases persistentes del componente Control de indicadores CTI en el CEIGE.

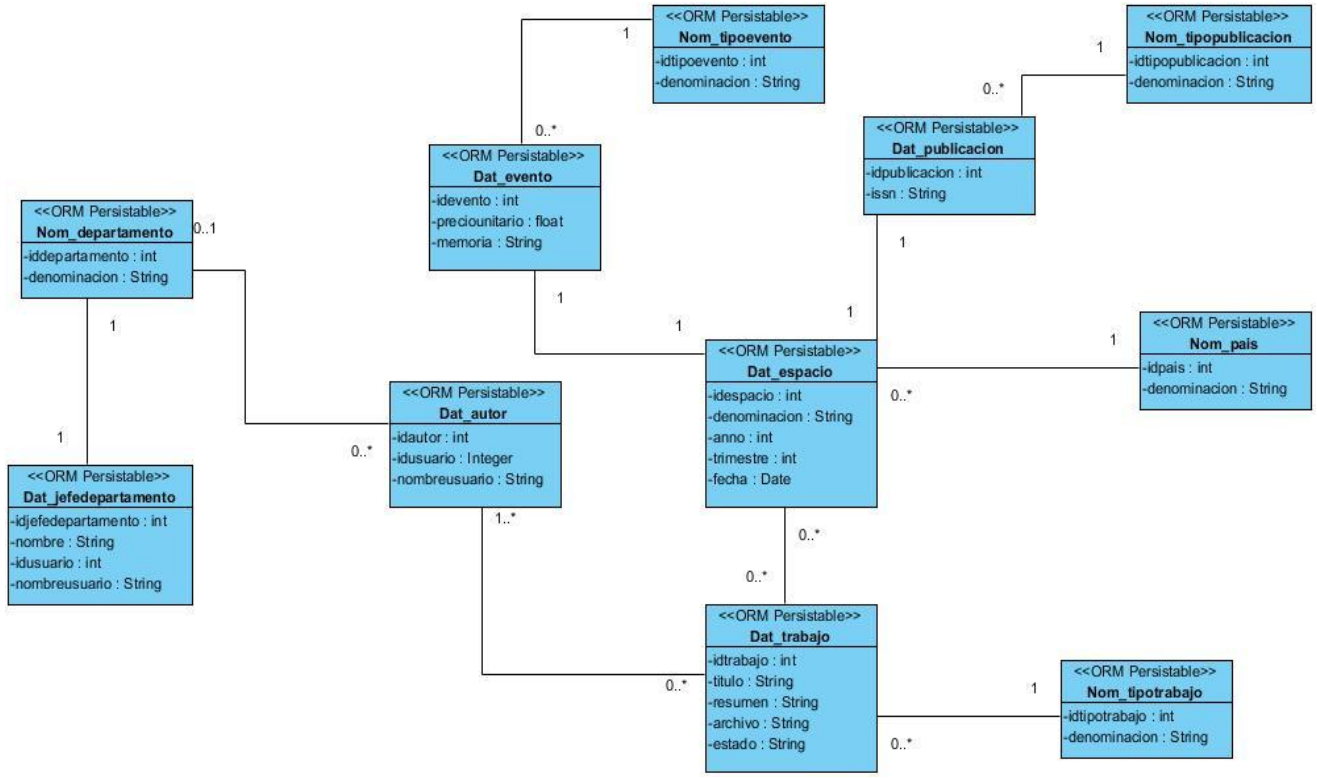


Figura 16. Diagrama de clases persistentes.

2.6.7 Diagrama de Componentes

Se considera componente a la parte física y reemplazable de un sistema que conforma un conjunto de interfaces y suministra la implementación de dicho conjunto. Los diagramas de componentes se utilizan para modelar la vista estática de un sistema y muestran la organización y las dependencias entre un conjunto de componentes [34]. La solución propuesta consta de un solo componente llamado Control de indicadores CTI, el cual interactúa con el componente Seguridad del marco de trabajo Sauxe, al que le pide los servicios getUsuarios () y cargarAcciones () mediante la clase SeguridadServiceProxy. La siguiente imagen muestra el diagrama de componentes desarrollado para el sistema.

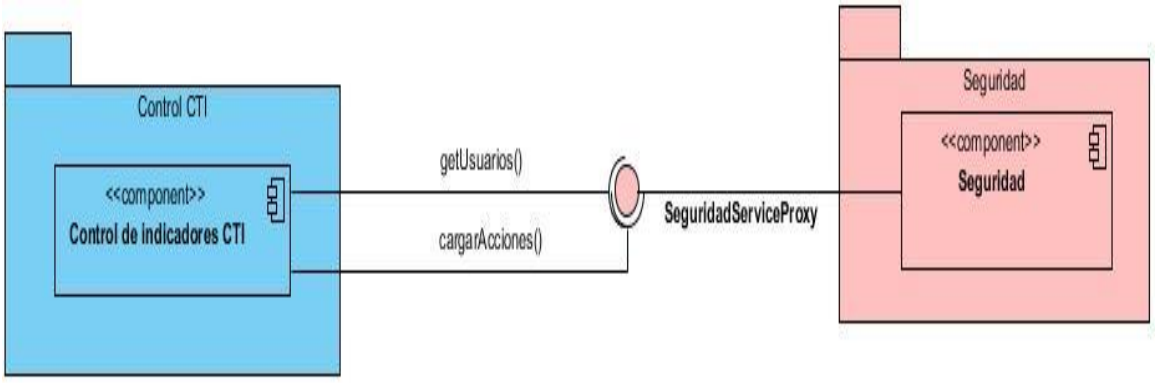


Figura 17. Diagrama de Componentes.

2.6.8 Métricas de validación del diseño

Las métricas de software permiten medir de forma cuantitativa la calidad de los atributos internos del software lo que permite evaluar la calidad durante el desarrollo del sistema, están inspiradas en el estudio de la calidad del diseño orientado a objeto. Inclinan sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo. [31] Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- **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:** Grado de dificultad que tiene implementado un diseño de clases.
- **Reutilización:** Grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento:** 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:** Grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software.
- **Cantidad de pruebas:** Número o grado de esfuerzo para realizar las pruebas de calidad del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

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

Tamaño Operacional de Clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad. [35]

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

Tabla 5. Atributos de calidad y modo en que afectan al instrumento TOC.

Los criterios y categorías definidos para la evaluación de los atributos de calidad anteriores se presentan en la siguiente tabla:

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

Tabla 6. Criterios de evaluación para la métrica TOC

Relaciones entre Clases (RC): Está dado por el número de relaciones de uso de una clase con otra. [35]

Atributos	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 7. Atributos de calidad y modo en que afectan al instrumento RC

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	< = Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Reutilización	Baja	> 2* Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio
Cantidad de Pruebas	Baja	< = Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio

Tabla 8. Criterios de evaluación para la métrica TOC.

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

El instrumento de evaluación TOC fue aplicado a una muestra de 14 clases escogidas aleatoriamente del componente Control de indicadores CTI, la Figura 18 muestra el

porciento de la cantidad de procedimientos presentes en las clases agrupados en intervalos definidos, resultando 12 clases entre 1-5 procedimientos, 1 clase entre 6-10 procedimientos, 1 clase entre 11-15 procediminetos.

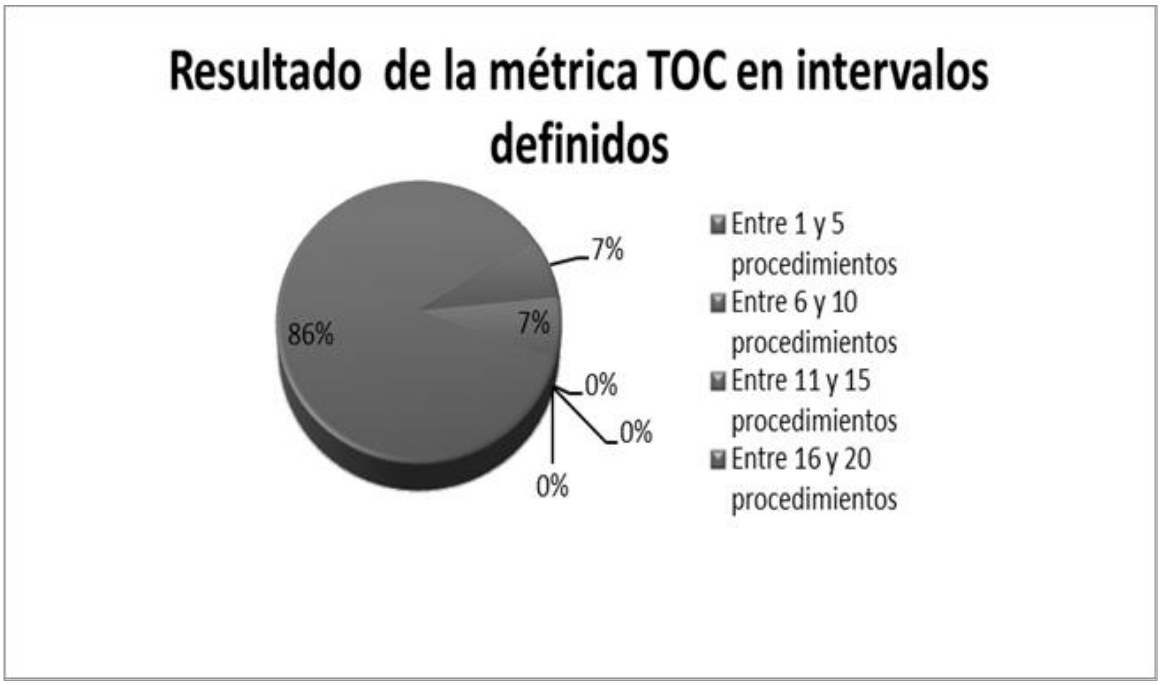


Figura 18. Representación en porciento agrupado en intervalos definidos de los resultados obtenidos tras aplicar el instrumento TOC.

Figura 19. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

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

Figura 21. Representación de la incidencia de los resultados de la evaluación del instrumento TOC en el atributo Reutilización.

Al hacer un análisis de los resultados obtenidos se concluye que la mayoría de las clases incluidas poseen de 1 a 5 procedimientos en su composición representando el 57% de la muestra. Debido a esto, el resultado influye positivamente en los demás atributos de calidad puesto que ayuda a que el 57% de las clases posea una responsabilidad baja, es decir que no estén demasiado sobrecargadas, además puede observarse que el 57% del total de clases analizadas tiene una complejidad también baja y que se pueda realizar una explotación alta del atributo reutilización.

Resultado de la aplicación del instrumento de evaluación de la métrica Relaciones entre Clases (RC).

El instrumento de evaluación RC fue aplicado a una muestra de 10 clases del subsistema Control de indicadores CTI seleccionadas aleatoriamente, la Figura 22

ilustra la cantidad de dependencias y el porciento que representan de la muestra seleccionada. Las figuras 23, 24, 25 y 26 muestran los resultados que arrojó la aplicación de la métrica en los atributos de calidad acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas.

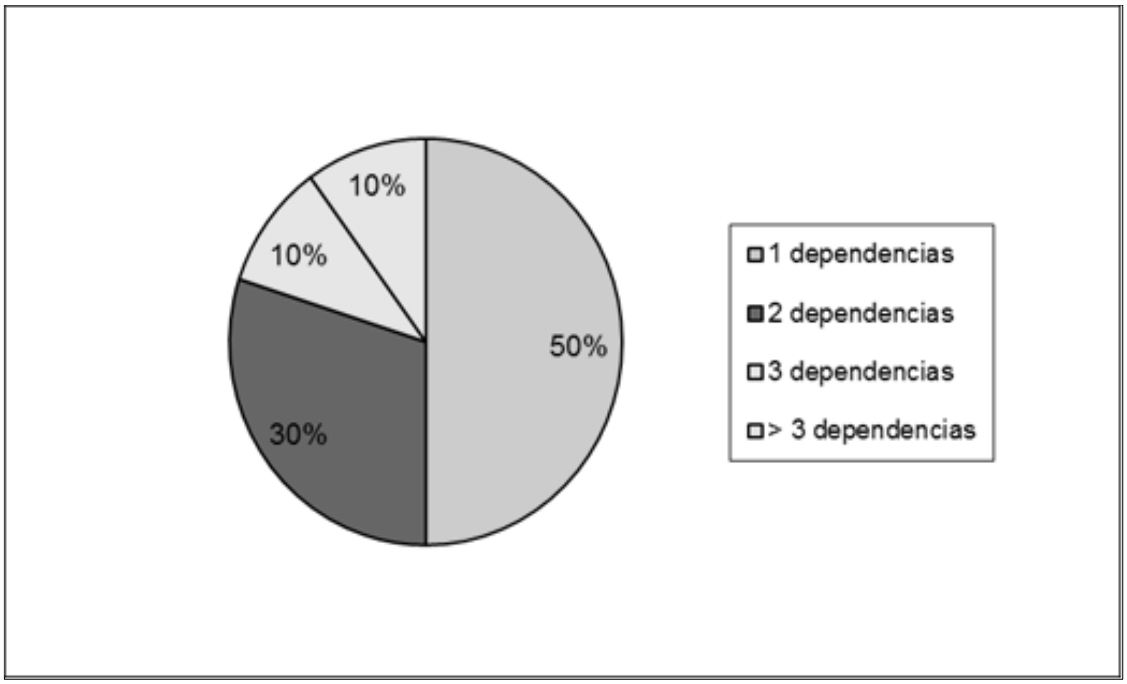


Figura 22. Representación en porciento de la aplicación del instrumento RC en intervalos definidos.

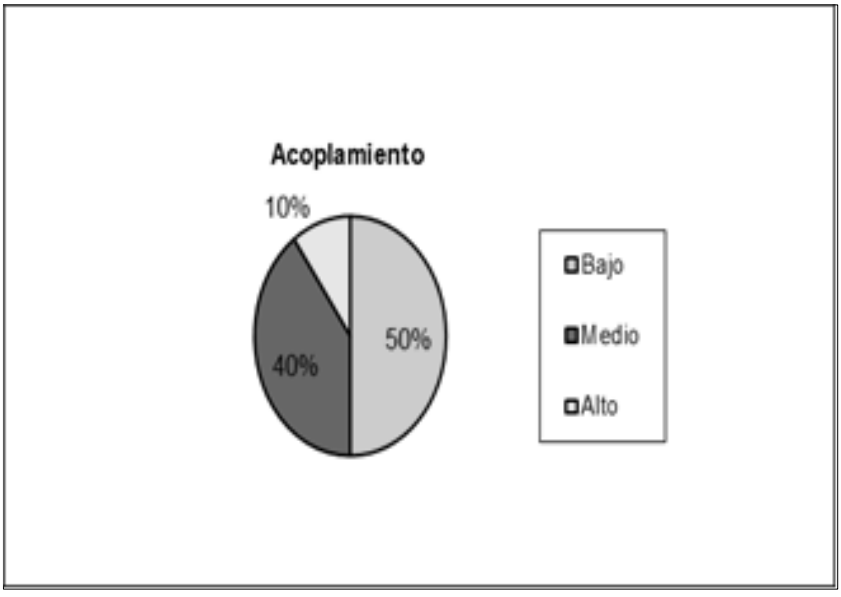


Figura 23. Representación de la incidencia de los resultados de la evaluación del instrumento RC en el atributo Acoplamiento.

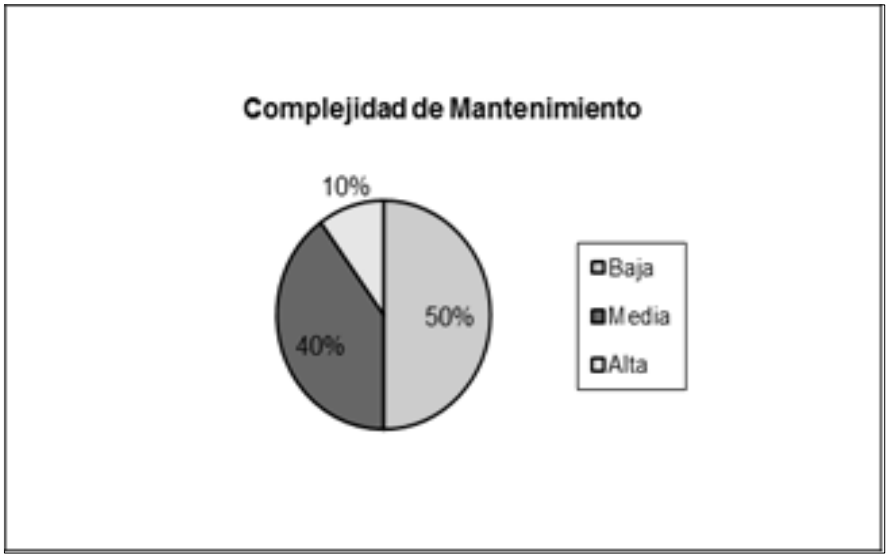


Figura 24. Representación de la incidencia de los resultados de la evaluación del instrumento RC en el atributo Complejidad de mantenimiento.

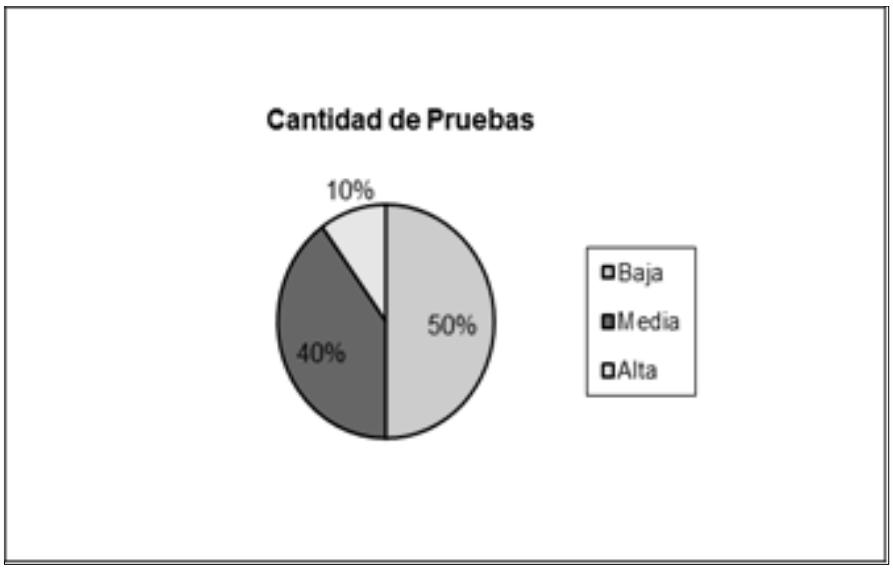


Figura 25. Representación de la incidencia de los resultados de la evaluación del instrumento RC en el atributo Cantidad de Pruebas.

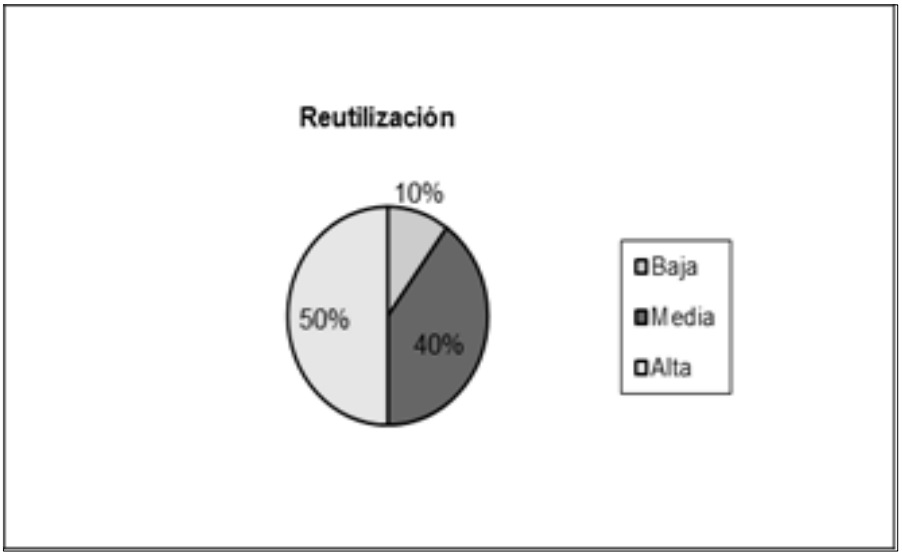


Figura 26. Representación de la incidencia de los resultados de la evaluación del instrumento RC en el atributo Reutilización

Al hacer un análisis de los resultados obtenidos a través de la métrica de software RC aplicada se resuelve que el 50% de las clases incluidas no poseen acoplamiento alguno lo cual ayuda significativamente en el sentido de que al ser afectada una clase este cambio no repercutirá en las demás. El porciento de reutilización de clases (50%) es alto lo cual fomenta y potencia el uso de las mismas en otras clases, en caso de ser necesario, además la complejidad de mantenimiento es en un 50% baja, facilitando que no resulte difícil el mantenimiento de las mismas (por ejemplo la optimización de métodos) y por último el atributo relacionado con la cantidad de pruebas es 50% bajo lo cual señala que a las clases se les puede realizar un bajo número de pruebas de poca complejidad y de bajos costes.

2.6.9 Estándares de codificación

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

- Convención de nomenclatura: es la forma de nombrar las variables, funciones, métodos.
- Convenciones de legibilidad de código: es la forma de organizar el código.

- Convenciones de documentación: es la manera de establecer los comentarios, la ayuda.

Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación **PascalCasing**, la cual define que los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula y con solo leerlo se reconoce el propósito de la misma. Ejemplo: DatEventoModel. En este caso el nombre de la clase está compuesto por 3 palabras iniciadas cada una con letra mayúscula.

Nomenclatura según el tipo de clases

- **Clase controladora:** La clase controladora después del nombre lleva la palabra:"Controller". Ejemplo: ControlCTIController.
- **Clases de los modelos: Business (Negocio):** Las clases que se encuentran dentro del negocio después del nombre llevan la palabra: "Model". Ejemplo: GestionarEventoModel.
- **Domain (Dominio):** Las clases que se encuentran dentro del dominio antes del nombre llevan la palabra: "Dat". Ejemplo: DatEvento.
- **Generated (Dominio base):** Las clases que se encuentran dentro de Generated, el nombre que reciben comienza con la palabra: "Base". Ejemplo: BaseDatEventoModel.

Nomenclatura de las funcionalidades y atributos

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

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

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

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

Nomenclatura de los comentarios

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

2.7 Conclusiones parciales

Como resultado de este capítulo se generaron los artefactos del análisis y el diseño del sistema propuesto. En el análisis se analizaron los requisitos funcionales y no funcionales del sistema, permitiendo de esta forma, adquirir una adecuada comprensión de las necesidades actuales para la construcción del Componente de control de indicadores CTI. El diseño se llevó a cabo mediante la correcta utilización de patrones, que posibilitó la organización del trabajo. Este fue validado a través de métricas que arrojaron resultados satisfactorios. El trabajo realizado en el transcurso del capítulo sirve de base fundamental para la confección del componente, pues se logra una mayor organización, proporciona rapidez y precisión en el momento de la implementación.

Capítulo 3: Implementación y Prueba

3.1 Introducción

En este capítulo se describen las principales clases y funcionalidades implementadas además de la realización de un estudio de los diferentes métodos y tipos de pruebas a llevar a cabo, con el objetivo de garantizar la calidad del sistema y el total cumplimiento de los requisitos establecidos con el cliente.

3.2 Aspectos fundamentales de la implementación

En este epígrafe se explica de forma concisa los métodos o algoritmos más complejos implementados en la confección del componente para el control de indicadores CTI.

La función *exportPDFAction()* responde al requisito funcional Generar reporte. El método haciendo un llamado al algoritmo *getAll()* de la clase DatAutorModel, obtiene todos los autores gestionados por el componente, luego para cada autor busca los trabajos realizados, los eventos en los que ha participado así como las publicaciones, haciendo uso de los métodos getTrabajosPorAutor(), getEventos($idtrabajo), getPublicaciones($idtrabajo) respectivamente. Esto es posible mediante consultas realizadas a la base de datos y la utilización de la librería fpdf.php que permite la generación de un reporte en formato pdf. A continuación se muestra en la figura 27 fragmentos del código que admite la realización de dichas operaciones.

```
public function exportPDFAction() {
    $pdf = new generadorpdf();
    $pdf->Open();
    $title = 'Reportes del Sistema para el Control de Indicadores CTI ';
    $pdf->SetAuthor('CTI');
    $pdf->SetTitle($title);
    $obj = new DatAutorModel();
    $autores = $obj->getAll();
    $objTrab = new DatTrabajoModel();
    $datos = array();
    foreach ($autores as $autor) {
        $idautor = $autor['idautor'];
        $trabajos = $objTrab->getTrabajosPorAutor($idautor);
        foreach ($trabajos as $trabajo) {
            $idtrabajo = $trabajo['idtrabajo'];
            $titulo = $trabajo['titulo'];
            $eventos = $objTrab->getEventos($idtrabajo);
            foreach ($eventos as $evento) {
                $denEvent = $evento['denominacion'];
                $tipoEvent = $evento['tipoevento'];

                $datos[] = array(
                    $autor['nombre'],
                    $titulo,
                    $denEvent,
                    $tipoEvent,
                );
            }
        }
    }
    $datosPublic = array();
```

```
foreach ($autores as $autor) {
    $idautor = $autor['idautor'];
    $trabajos = $objTrab->getTrabajosPorAutor($idautor);
    foreach ($trabajos as $trabajo) {
        $idtrabajo = $trabajo['idtrabajo'];
        $titulo = $trabajo['titulo'];
        $publicaciones = $objTrab->getPublicaciones($idtrabajo);
        foreach ($publicaciones as $publicacion) {
            $publicDenom = $publicacion['denominacion'];
            $tipopublicacion = $publicacion['tipopublicacion'];

            $datosPublic[] = array(
                $autor['nombre'],
                $titulo,
                $publicDenom,
                $tipopublicacion,
            );
        }
    }
}
$cabeceras = array(
    'AUTOR' => array('colorletra' => '', 'colorfondo' => '', 'estilo' => '', 'alineacion' => '')
    , 'TRABAJO' => array('colorletra' => '', 'colorfondo' => '', 'estilo' => '', 'alineacion' => '')
    , 'EVENTO' => array('colorletra' => '', 'colorfondo' => '', 'estilo' => '', 'alineacion' => '')
    , 'TIPO EVENTO' => array('colorletra' => '', 'colorfondo' => '', 'estilo' => '', 'alineacion' => ''));
$fuente = array('tamano' => 10, 'tipo' => 'Arial', 'estilo' => '');
$titleTable = 'CEIGE - EVENTOS';
$titleTablePiblic = 'CEIGE - PUBLICACIONES';
$pdf->crearTabla($titleTable, $cabeceras, $datos, true, $fuente, 10);
$pdf->crearTabla($titleTablePiblic, $cabeceras, $datosPublic, true, $fuente, 10);
$pdf->Output();
```

Figura 27. Método exportPDFAction ().

Otra funcionalidad que se necesitó para la implementación del componente fue *addTrabajoAction()* la cual posibilitó persistir los trabajos presentados por un autor en un evento o publicación. La función *addTrabajoAction()* responde al requisito funcional Gestionar trabajos. El algoritmo obtiene los parámetros necesarios para crear un trabajo los cuales son enviados desde la vista haciendo uso del método POST, luego hace un llamado a la función *saveTrabajo($idtipotrabajo, $titulo, $resumen, $archivo,$estado, $idespacio, $idusuarioaut, $orden, $idautor)*, de la clase DatTrabajoModel la cual se encarga de persistir el trabajo, cuando este proceso termina informa a la vista el estado de la operación. A continuación se muestra en la figura 28 fragmentos del código que posibilita hacer dichas operaciones.

```php
function addTrabajoAction() {
    $titulo = $this->_request->titulo;
    $resumen = $this->_request->resumen;
    $archivo = $this->_request->archivo;
    $estado = "Nuevo";
    $idtipotrabajo = $this->_request->idtipotrabajo;
    $idespacio = $this->_request->idespacio;
    $idusuario = $this->global->Perfil->idusuario;
    $nombreautor = $this->_request->nombreautor;
    $orden = $this->_request->orden;
    $dpto = $this->_request->dpto;
    $idusuarioaut = $this->_request->idusuario;
    $nombreusuario = $this->_request->nombreusuario;
    $idautor = $this->_request->idautor;
    $objTrab = new DatTrabajoModel();
    $control = $objTrab->saveTrabajo($idtipotrabajo, $titulo, $resumen, $archivo,
               $estado, $idespacio, $idusuarioaut, $orden, $idautor);
    if ($control)
        echo json_encode(array('messaje' => "Si"));
    else
        echo json_encode(array('messaje' => "No"));
}
```

Figura 28. Fragmento de código de la función *addTrabajoAction ()*.

3.2.1 Descripción de las clases y funcionalidades del sistema.

Para un mayor entendimiento de la implementación, a continuación se detallan las clases de mayor peso en el sistema y sus principales funcionalidades. Se describe la clase controladora ControlctiController encargada de manejar todas las peticiones realizadas desde la vista, DatTrabajoModel la cual realiza todas las operaciones asociadas a los trabajos y DatTrabajo que brinda los métodos necesarios para gestionar el acceso a los datos.

Clase Controladora:

Nombre: ControlctiController	
Tipo de clase: Controladora	
Para cada funcionalidad:	
Nombre	getEventosAction ()
Descripción	Devuelve todos los eventos.
Nombre	getPublicacionesAction ()
Descripción	Devuelve todas las publicaciones.
Nombre	getTrabajosAction ()

Descripción	Devuelve todos los trabajos.
Nombre	addEventoAction ()
Descripción	Permite guardar un evento.
Nombre	modEventoAction ()
Descripción	Permite modificar un evento previamente seleccionado.
Nombre	eliminarEventoAction ()
Descripción	Permite eliminar un evento seleccionado en el sistema.
Nombre	addPublicacionAction ()
Descripción	Permite guardar una publicación.
Nombre	modPublicacionAction ()
Descripción	Permite modificar cada una de las publicaciones creadas en el sistema.
Nombre	eliminarPubAction ()
Descripción	Permite eliminar una publicación seleccionada en el sistema.
Nombre	addTrabajoAction ()
Descripción	Permite guardar un trabajo.
Nombre	modTrabajoAction ()
Descripción	Permite modificar cada uno de los trabajos creados en el sistema
Nombre	delTrabajoAction ()
Descripción	Permite eliminar un trabajo seleccionado en el sistema.
Nombre	exportPDFAction ()
Descripción	Permite exportar al formato pdf los trabajos, publicaciones y eventos de un autor seleccionado en el sistema.

Tabla 9. Descripción de la clase ControlctiController.

Clase del modelo:

Nombre: DatTrabajoModel
Tipo de clase: Modelo

Para cada funcionalidad:	
Nombre	getTrabajos($start,$limit,$idespacio,$idusuario)
Descripción	Permite listar los trabajos presentes en el sistema según los parámetros pasados.
Nombre	saveTrabajo($idtipotrabajo,$titulo,$resumen,$archivo,$estado, $idespacio,$idusuario,$orden,$idautor)
Descripción	Permite guardar un trabajo según los parámetros pasados.
Nombre	updateTrabajo($idtipotrabajo, $titulo, $resumen, $archivo, $idtrabajo,$orden,$idautor,$idautorviejo)
Descripción	Permite actualizar un trabajo según los parámetros pasados.
Nombre	deleteTrabajo($idtrabajo)
Descripción	Permite eliminar un trabajo según los parámetros pasados.

Tabla 10. Descripción de la clase DatTrabajoModel.

Clase del dominio:

Nombre: DatTrabajo	
Tipo de clase: Dominio	
Para cada responsabilidad:	
Nombre	getTrabajos($start,$limit,$idespacio,$idautor)
Descripción	Obtiene los datos de todos los trabajos asociadas a un autor en un espacio (Evento, Publicación) determinado.
Nombre	getAutores($idusuario)
Descripción	Obtiene los datos de todos los autores, especificándole el identificador de este.
Nombre	getTrabajosByAutor($start, $limit, $idautor)
Descripción	Obtiene los datos de todos los trabajos asociados a un autor determinado, especificándole el identificador de este.

Tabla 11. Descripción de la clase DatTrabajo.

3.3 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 [36]. Se definirá una arquitectura cliente-servidor detallada de la siguiente manera, ver Figura 29.

A continuación se muestra el diagrama de despliegue del sistema:

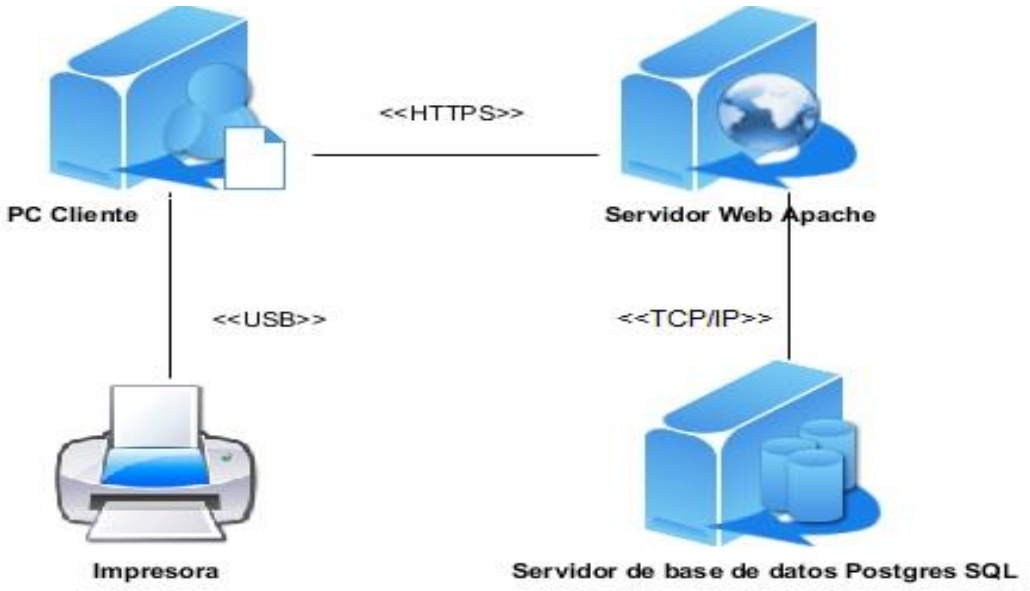


Figura 29. Diagrama de despliegue.

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.

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

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

3.4 Pruebas de software

Las pruebas de software son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador, probando el comportamiento del mismo. [37]

La práctica de pruebas en la construcción de software reduce el tiempo de desarrollo y así la cantidad de tiempo necesario para integrar y estabilizar versiones. Mejora la productividad encontrando y arreglando errores rápidamente, además de incrementar la calidad global del software garantizando que todo el código nuevo ha sido probado, y a todo el código existente se le aplican pruebas de regresión antes de ser integrados a la base central de código.

3.4.1 Prueba de caja blanca

La Prueba de Caja Blanca también se conoce como Prueba de Caja Transparente o de Cristal. Esta prueba consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento [31]. Dentro de la prueba de caja blanca se incluyen las Técnicas de Pruebas que serán descritas a continuación:

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

  Prueba de Bucles: Se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución de todos los bucles en sus límites operacionales

La técnica de caja blanca empleada en la solución desarrollada fue la Prueba del Camino Básico, la cual permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa. [31]

Para realizar esta técnica es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método addEventoAction (), ver Figura 30, y el grafo de flujo asociado al mismo, ver Figura 31.

```
function addEventoAction() {
    $denominacion = $this->_request->denominacion;
    $tipoevento = $this->_request->tipoevento;
    $anno = $this->_request->anno;
    $memoria = $this->_request->memoria;
    $fecha = $this->_request->fecha;
    $pais = $this->_request->pais;
    $trimestre = $this->_request->trimestre;
    $precio = $this->_request->precio;

    $objEvento = new DatEventoModel();

    $objAddEvento = $objEvento->addEvento($denominacion, $tipoevento, $anno,
            $memoria, $fecha, $pais, $trimestre, $precio);

    if ($objAddEvento)
        echo json_encode(array('messaje' => "Si"));
    else
        echo json_encode(array('messaje' => "No"));
}
```

Figura 30. Método addEventoAction ().

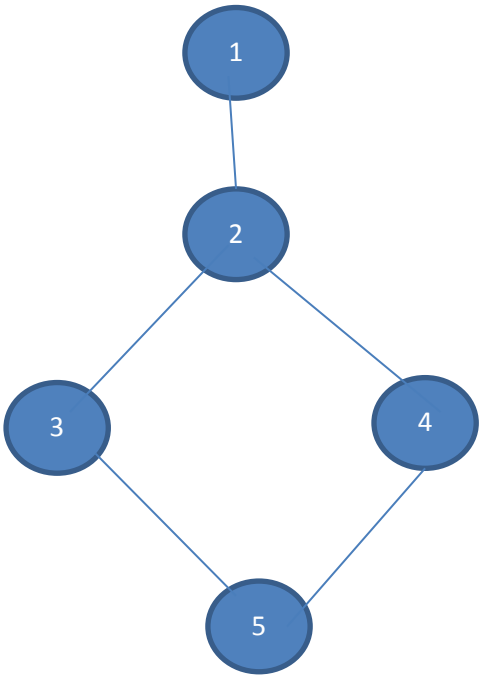


Figura 31. Grafo de flujo asociado al algoritmo addEventoAction ().

Fórmulas para calcular la complejidad ciclomática:

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

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

V (G) = (5 - 5) + 2

V (G) = 2

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

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

V (G) = 1 + 1

V (G) = 2

3. **V (G) = R**

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

V (G) = 2

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 2, lo que significa que existen 2 posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Número	Caminos básicos
1	1-2-3-5
2	1-2-4-5

Tabla 12. Caminos básicos del flujo.

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

- **Descripción:** Se describe el caso de prueba y de forma general se tratan los aspectos fundamentales de los datos de entrada.
- **Condición de ejecución:** Se especifica cada parámetro para que cumpla una condición deseada y así ver el funcionamiento del procedimiento.
- **Entrada:** Se muestran los parámetros que serán la entrada al procedimiento.

Resultados Esperados: Se expone el resultado esperado que debe devolver el procedimiento después de efectuado el caso de prueba.

Caso de prueba para el camino básico # 1.

Descripción: Se introducen los datos asociados al evento.

Condición de ejecución: El evento es “Adicionado”.

Entrada:

- $objAddEvento= true, representa que el evento se ha adicionado correctamente.

Resultados Esperados: Se espera que muestre un mensaje informando que el evento se ha adicionado satisfactoriamente.

Resultados obtenidos: Satisfactorio.

Caso de prueba para el camino básico # 2.

Descripción: Se introducen los datos asociados al evento.

Condición de ejecución: El evento no es “Adicionado”.

Entrada:

- $objAddEvento= false, representa que ha ocurrido un error al adicionar el evento.

Resultados Esperados: Se espera que muestre un mensaje informando que ha ocurrido un error al adicionar el evento.

Resultados obtenidos: Satisfactorio.

3.4.2 Prueba 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. [31]
En esencia permite encontrar:

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

Algunos de los métodos empleados en las pruebas de Caja Negra son los siguientes:

- Métodos de prueba basados en grafos.
- Partición de equivalencia.
- Análisis de valores límite.

- Prueba de la tabla ortogonal.

Las pruebas se realizaron por el método partición de equivalencia sobre todos los requisitos funcionales de la herramienta, haciendo uso de los casos de prueba correspondientes a cada uno de estos. A continuación se muestra un ejemplo:

Caso de prueba para el requisito Adicionar evento

Condiciones de ejecución

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el subsistema Control de Indicadores CTI/Control de Indicadores CTI (CEIGE).

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Adicionar evento.	El sistema permite realizar registros de eventos	EP 1.1: Adicionar evento introduciendo datos válidos presionando el botón **Aceptar.**	– Se presiona el botón **Adicionar.** – Se abre la interfaz Adicionar evento. – Se introducen los datos del evento correctamente. – Se presiona el botón **Aceptar.** – Se muestra un mensaje de información. – Se presiona el botón **Aceptar** del mensaje.
		EP 1.2: Adicionar evento dejando campos vacíos.	– Se presiona el botón **Adicionar.** – Se abre la interfaz Adicionar evento. – Se introducen los datos dejando algún campo en blanco. – Se presiona el botón **Aceptar.** – Se muestra un mensaje informando del error.

		EP 1.3: Cancelar	– Se presiona el botón **Adicionar.** – Se abre la interfaz Adicionar evento. – Se introducen o no los datos del evento. – Se presiona el botón **Cancelar**.

Tabla 13. Caso de prueba para el requisito Adicionar evento.

Las pruebas realizadas al sistema fueron satisfactorias desde el punto de vista interno y funcional, estas abarcaron requerimientos, funciones y la lógica interna del sistema. Se aplicaron los métodos de caja negra y caja blanca para validar tanto la interfaz como el correcto funcionamiento interno del software. Combinar ambos enfoques permitió lograr mayor fiabilidad en el proceso de pruebas al diseñar los casos de prueba.

El sistema luego de resueltas las 5 NC detectadas durante las pruebas realizadas fue liberado por el Grupo de Verificación y Validación del CEIGE.

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

3.5 Conclusiones parciales

Con las pruebas realizadas y la validación de la solución finaliza el presente trabajo. La confección de este último capítulo posibilitó arribar a las conclusiones siguientes:

- Se describieron las principales clases y operaciones que tienen mayor peso el sistema para hacer más fácil el entendimiento de la implementación.
- Se realizaron al sistema pruebas de caja blanca y caja negra. Estas pruebas lograron su objetivo principal: demostrar la calidad de la solución propuesta a través de un código limpio y un funcionamiento general exitoso.

Conclusiones Generales

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

- Se analizaron sistemas informáticos tanto nacionales como internacionales vinculados a la gestión de la información, evidenciándose de esta manera la no existencia de una solución informática capaz de cubrir todas las funcionalidades requeridas para realizar una adecuada gestión de la información referente a los parámetros CTI en el CEIGE, lo cual hizo necesario el desarrollo de un sistema informático que se ajuste sobre todo a los intereses y estrategias de trabajo de dicho centro.

- Se realizó el análisis y el diseño del sistema, permitiendo recoger todas las necesidades de los clientes, además de lograr un mejor entendimiento del proceso que fue informatizado, garantizando una comprensión exacta sobre las funcionalidades a informatizar.

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

Recomendaciones

Una vez cumplidos los objetivos de la investigación y teniendo en cuenta las experiencias obtenidas en la misma, se recomienda:

- Que la aplicación obtenida sea utilizada en otros centros productivos y no solo en el centro CEIGE.
- Continuar el estudio del tema con el objetivo de encontrar nuevas funcionalidades para futuras versiones de la aplicación.
- Aplicar las experiencias obtenidas en el desarrollo de aplicaciones similares.

Referencias

1- **Sancho, R.** Directrices de la Ocde para la obtención de Indicadores de Ciencia y Tecnología, Ministerio de Ciencia y Tecnología, Madrid, España. 2010.

2- **Council of Biology Editors.** Proposed definition of a primary publication: Newsletter. Council of Biology Editors. Noviembre de 1968: 1-2.

3- **López, Elvert Maldonado.** DESARROLLO DE REPORTES PARA EL SISTEMA DE GESTIÓN DE CIENCIA, TECNOLOGÍA E INNOVACIÓN SIGCTI. Granma: Facultad Regional Granma de la Universidad de las Ciencias Informáticas., 2010.

4- **Rosabal, Antonio San Juan.** CUADERNOS DE EDUCACIÓN Y DESARROLLO. [Online] mayo 2011. [Cited: junio 24, 2012.] http://www.eumed.net/rev/ced/27/jrrg.htm.

5- **Barcelona, Universidad de GREC.** [Online] [Cited: Abril 06, 2012.] https://Webgrec.ub.edu/.

6- **Cabrera, Francisco Manuel Solís.** Comunidad de Andalucía. [Online] [Cited: Abril 07,2012.]http://www.madrimasd.org/informacionidi/revistas/monograficos/monografias/monografia22/las_CA_frente_IDi-sistema_informacion_cientifico_andalucia.pdf.

7- **Iván F. Palomo, Carlos G. Veloso y Rodolfo F. Schmal.** Sistema de Gestión de la Investigación en la Universidad de Talca, Chile. Talca : Universidad de Talca, Chile., 2007.

8- **Linea., Gestión de proyectos de Investigación en.** Universidad Pedagógica Nacional . [Online] 2005. [Cited: Febrero 26, 2012.] http://www.pedagogica.edu.co/pgil/pgil.php.

9- **O.E.Sánchez, B.Garea,S.Lantigua y otros.** Sistema de Información para la Gestión de Programas de Ciencia e Innovación en Cuba. 2008.

10- **Batista, Ing. Yunaysy Ortiz.** Sistema integrado de gestión de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas . La habana : Universidad de las Ciencias Informáticas, 2011.

11**- Modelo de desarrollo de software de CEIGE**. V1.1

12- **Gutierrez, Javier J.** [Online] [Cited: marzo 15, 2013.] http://www.lsi.us.es/-javierj/investigación_ficheros/framework.pdf.

13- **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.

14- **Sanchez, Miguel Angel Alvarez.** ExtJS. [Online] [Cited: febrero 5, 2013.] http://www.desarrolloWeb.com/wiki/un-vistazo-a-ext-js.thml.

15- **Maikel Yulier Sañudo Clemente, Leandro Luis Rojas Cuesta.** Desarrollo de la versión 2.0 de la herramienta Doctrine Generator para la generación de ficheros de mapeo basado en Doctrine empleando la tecnología PHP. Cuidad de la Habana : s.n., 2010.

16- **PHP, Ciber aula.** Ciber aula PHP. [Online] [Cited: febrero 5, 2013.] http://php.ciberaula.com/articulo/PHPoASP.

17- **Val, Miguel Burillo.** [Online] 2011. [Cited: febrero 20, 2013.] http://upcommons.upc.edu/pfc/bitstream/2099.1/5837/5/part%204.pdf.

18- **TIC, TEMARIO.** [Online] [Cited: marzo 10, 2013.] http://temariotic.wikidot.com/la-arquitectura-cliente-servidor.

19- **Productiva, D. d. C. d. l. I.** (2011). MODELO DE SISTEMA V2.0. SIT. Módulo Administración y Gobierno: 354.

20- **Sánchez-Segura, Ana M. Moreno y Maribel.** Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento de Arquitectónico. Madrid : Universidad Politécnica de Madrid, España, 2010.

21- **Sebastián, J.** (2010) Modelo Vista Controlador – Definición y Características.

22- **Riesco, Prof. Daniel.** UML. [En línea] 2004. [Citado el: 5 de Febrero de 2012.] http://sel.unsl.edu.ar/licenciatura/ingsoft2/UMLIntroduccion.pdf.

23- **Manager, Free Download.** Free Download Manager. [En línea] 5 de Marzo de 2007.http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

24- **EcuRed. In: IDE Desarrollo** EcuRed http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n

25- **NetBeans, Comunidad.** [Online] [Cited: febrero 5, 2013.] http://netbeans.org/index_es.html.

26- **Ciber Aula, Linux.** [Online] [Cited: marzo 15, 2013.] http://linux.ciberaula.com/articulo/linux_apache_intro.

27- . **González, Carlos D.** Curso PostgreSQL, SQL avanzado y PHP. [En línea] Febrero de 2012. http://www.usabilidadWeb.com.ar/postgre.php.

28- **Features, Firefox.** [Online] [Cited: marzo 15, 2013.] http://www.mozilla-europe.org/es/firefox/features/$top-new-features.

29- **Modelo de dominio.** ongei.gob.pe. [En línea] [Citado el 6 de Febrero del 2013]. http://www.ongei.gob.pe.

30- **The Unified Software Development Process.** [En línea] [Citado el 22 de Enero del 2013.] http://www0.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-01-02/aswe5.pdf.

31- **Hill, Roger S Pressman & Mc Graw** Ingeniería del Software: Un enfoque práctico 4ª Edición2005

32- **Gracia, Joaquin.** IngenieroSoftware. IngenieroSoftware. [En línea] 27 de Mayo de 2005. [Citado el: 15 de Enero de 2012.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

33- **Guillerón, Gastón.** GLN Ingeniería & Consultoría. [Online] julio 12, 2009. [Cited: junio 24, 2012.] http://glnconsultora.com/blog/?p=3.

34- **Cuesta, P. David.** Diagramas UML. [Online] [Cited: abril 8, 2013.] http://www.slideshare.net/e1da4/diagrama-uml.

35- **Lorenz, Mark y Kidd, Jeff** Object-Oriented Software MetricsObject-Oriented Software MetricsPrentice-Hal994.

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

37- **EcuRed: Enciclopedia cubana.** [En línea] [Citado el: 20 de Mayo de 2012.] http://www.ecured.cu/index.php/Pruebas_de_software.