Universidad de Ciencias Informáticas

Facultad 3

Trabajo de Diploma para optar por el título de Ingeniro en las Ciencias Infomáticas

Título: Desarrollo del módulo Técnicos de Laboratorio del sistema para la gestión de los procesos administrativos SIGPA del centro de Informatización de Entidades.

Autor: Marcos Antonio Escobar Couto

Tutores: Ing. Silvia María Llarch Leyva

Msc. Yoansy López Reyes

La Habana, junio 2017

Declaración de Autoría

Declaro ser el único autor del presente trabajo de diploma y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio. Para que así conste, firmo la presente a los ___ días del mes de ___ del año 2017.

Marcos Antonio Escobar Couto

_____

Firma del Autor

Yoansy López Reyes

_____

Firma del Tutor

Silvia María Llarch Leyva

_____

Firma del Tutor

Dedicatoria

Dedico el presente trabajo a mis padres por brindarme todo su apoyo y confianza en el trannscurso de estos cinco años de carrera. A mi familia en general. Los amo a todos.

Agradecimientos

Agradesco a toda mi familia, los que tengo cerca y los que están lejos y en especial a mis padres por todo su apoyo y sacrificio durate estos cinco años de carrera. A mi novia Sucel por haberme soportado en las buenas y en las malas te amo. A mis tutores por su amistad y por su tiempo dedicado en este trabajo. A mis compañeros de aula, los nuevos conocidos y los que deje atraz en la FRG. A mis amigo por todo el apoyo brindado en los momentos malos y buenos. En fin a todos los que hicieron posible la realizacion de este trabajo. A todos ustedes Muchas Gracias.

RESUMEN

El Centro de Informatización de Entidades, tiene la necesidad de controlar los procesos que se desarrollan en torno a los Técnicos en Ciencias Informáticas (TCI), los cuales tienen a su cargo la custodia de los laboratorios docentes y de producción. El presente trabajo tiene como objetivo la realización de un sistema web que permita informatizar los procesos sobre los cuales interactúan los TCI, tales como: proceso de entrega y recibo de laboratorios, asignación y seguimiento de sus tareas y evaluaciones de desempeño, que hoy se realizan de manera manual. La solución permitirá establecer las bases para lograr un mayor control y seguridad sobre los procesos mencionados anteriormente, logrando así reducir gastos y disminuir el tiempo en que ocurren las actividades realizadas.

El desarrollo del sistema informático estuvo guiado por la metodología AUP-UCI. Para la implementación del sistema se utilizó como marco de trabajo Symfony 2.8, Bootstrap 3.1, como lenguaje del lado del servidor PHP 5.5 y del lado del cliente HTML 5, CSS3 y JavaScript, como Entorno de Desarrollo Integrado Neatbeans 8.0, servidor de aplicaciones web Apache 2.4.9. La validación del sistema se realizó mediante la ejecución de pruebas que permitieron identificar defectos, que al ser corregidos; incrementan la calidad y aceptación del producto.

SUMMARY

The Center for Computerization of Entities, has the need to control the processes that are developed around the Technicians in Computer Science (TCI), who are in charge of the custody of teaching and production laboratories. The objective of this work is to develop a web system that allows computerization of the processes on which ICTs interact, such as the process of delivery and receipt of laboratories, assignment and monitoring of their tasks and performance evaluations, which are carried out today Manually. The solution will establish the basis for greater control and security over the processes mentioned above, thus reducing costs and reducing the time in which the activities carried out.

The development of the computer system was guided by the AUP-UCI methodology. For the implementation of the system, Symfony 2.8, Bootstrap 3.1, as a PHP 5.5 server-side language and HTML5, CSS3 and JavaScript client side were used as the Neatbeans 8.0 Integrated Development Environment, Apache web application server 2.4.9. The validation of the system was performed through the execution of tests that allowed to identify defects, that when being corrected; Increase the quality and acceptance of the product.

Índice

Índice de ilustraciones

Índice de Tablas

INTRODUCCIÓN

Los adelantos científico-técnicos a nivel mundial constituyen un poderoso pilar en el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC). El acelerado avance de estas tecnologías ha impulsado a los centros de educación superior a estar inmersos en una constante búsqueda de nuevas formas y métodos para perfeccionar la gestión de sus procesos.

La Universidad de las Ciencias Informáticas (UCI) es una de las pioneras en Cuba en la incorporación de estas tecnologías para la gestión de sus procesos. Varias son las soluciones implantadas para gestionar los procesos de formación, producción, investigación, gestión académica, gestión contable, gestión de recursos humanos y gestión tecnológica, entre otros.

Dentro de la estructura que tiene la universidad se encuentran los centros de desarrollos, estos tienen como misión fundamental el desarrollo de aplicaciones y servicios informáticos para colaborar en la solución de necesidades y problemas existentes en la sociedad, permitiendo la gestión de los diversos procesos antes mencionados (T. López, 2008).

Actualmente unos de los centros de desarrollo que permite suplir las necesidades y problemas de la sociedad con la introducción de productos informáticos es el Centro de Informatización de Entidades (CEIGE), este tiene como misión desarrollar productos y servicios informáticos para la gestión de entidades, contribuyendo a la formación de estudiantes y profesionales, permitiendo un posicionamiento en el mercado nacional e internacional (Objetivos y Metas).

Para su funcionamiento desarrolla un conjunto de procesos transversales que involucran a todos los departamentos. Algunos de procesos fundamentales son los siguientes:

- Gestión del control interno
- Gestión de los técnicos de laboratorio
- Gestión de la planificación y control de los proyectos
- Gestión de la mercadotecnia

Este centro teniendo en cuenta la gama de procesos que en él ocurren, tiene la necesidad de gestionar y controlar la información que se maneja en el proceso que se desarrolla en torno a los Técnicos en Ciencias Informáticas (TCI) que trabajan en cada uno sus laboratorios.

Los TCI tienen a su cargo la custodia de los laboratorios docentes y de producción, implicando ser responsables de una gran cantidad de medios tecnológicos dentro de estos locales. Estos técnicos de acuerdo a lo establecido por (López Reyes, 2017) tienen las siguientes funciones:

- Garantizar el cuidado y protección de los medios puestos bajo su custodia.
- Mantener un estricto control sobre los medios y su estado técnico, además de reportar cualquier rotura al asesor de tecnología.
- Velar por el cumplimiento de las políticas de seguridad informática en los laboratorios que atiende, así como tener amplio conocimiento de las regulaciones que rigen la seguridad informática en la UCI.
- Garantizar en cumplimiento de los reglamentos y normas vigentes en la UCI.
- Realizar el mantenimiento a las estaciones de trabajo.
- Mantener una constante superación individual en los aspectos técnicos del trabajo que realiza en los laboratorios.

Los TCI tienen asignados uno o varios laboratorios, en los cuales deben realizar un proceso de entrega recepción en cada cambio de turno. Este proceso se realiza mediante la observación directa del estado de los sellos e integridad y funcionamiento de los medios. El resultado de esta revisión se debe reflejar en un modelo de entrega – recepción, que actualmente se realiza en un modelo impreso llenado manualmente. El modelo de entrega – recepción contiene el registro de los medios del local que se entregan, así como las incidencias y observaciones necesarias. De esta forma se deja constancia de la cantidad de medios entregados y recibidos.

En entrevistas realizadas a directivos del centro CEIGE se identificó la necesidad y la importancia de gestionar las tareas asignadas a los TCI, en función de los siguientes aspectos:

- Exista una vía para la medición del desempeño de los TCI y esta información pueda ser utilizada en la toma de decisiones.
- El seguimiento a las tareas asignadas proporcione una fuente para el mejoramiento de las habilidades y competencias de los TCI.
- Poder disponer de indicadores y estadísticas que permitan adoptar estrategias de superación de los TCI.

El procedimiento usado actualmente para el seguimiento a las tareas de los TCI es completamente manual y la información se almacena en formato duro (papel).

Desde esta perspectiva se identifican las siguientes deficiencias:

- Pérdida de información relevante por deterioro del papel con el paso del tiempo.
- Insuficiente seguridad de la información en los documentos pudiendo existir alteración por parte de personal no autorizado.
- No se dispone con facilidad de la información debido formato en que se almacena.
- Incrementa el consumo de recursos materiales (hojas).
- Se hace difícil la consulta de la información histórica del proceso de entrega.
- Las incidencias u observaciones que se reportan pueden pasar desapercibidas debido a errores humanos.
- La información obtenida en la generación de los modelos de entrega es propensa a errores.

En las investigaciones realizadas por (Mar Cornelio, 2011) y (Saborit & Alemán, 2015), se aborda el proceso de entrega – recepción de medios tecnológicos de laboratorios. Ambas investigaciones coinciden en que la forma en que se realiza el proceso manualmente tiene deficiencias. Para resolver este problema ambos proponen el desarrollo de sistemas informáticos para la gestión del proceso de entrega – recepción.

Como limitante común de estas aplicaciones, se identifica que ninguna informatiza los procesos relacionados con el resto de las funciones que realizan los TCI. Si bien

es importante el control del proceso de entrega – recepción, también es de interés poder darle seguimiento al resto de las funciones en las que están implicados como son la asignación y seguimiento de las tareas y evoluciones.

En el estudio realizado no se identificaron aplicaciones que gestionaran el proceso de asignación, seguimiento y control de tareas desarrolladas por los TCI y que estas a su vez sean consideradas para la evaluación mensual de los mismos.

Teniendo en cuenta las deficiencias detectadas anteriormente se define como **problema a resolver:** ¿Cómo elevar el control sobre los procesos desarrollados por los Técnicos en Ciencias Informáticas en el Centro de Informatización de Entidades de la Facultad 3?

Estableciéndose como **objeto de estudio:** Informatización de los procesos de gestión administrativa.

Siendo el **campo de acción**: Informatización de los procesos desarrollados por los TCI en la UCI.

Para darle solución al problema planteado se define como **objetivo general:** desarrollar el módulo Técnicos de Laboratorio del Sistema para la Gestión de los Procesos Administrativos SIGPA del centro de Informatización de Entidades, de manera que se contribuya a elevar el control de los procesos desarrollados por los Técnicos en Ciencias Informáticas en el centro CEIGE de la Facultad 3.

Se propone **como idea a defender**:

El desarrollo del módulo Técnicos de Laboratorio del Sistema para la Gestión de los Procesos Administrativos SIGPA del centro de Informatización de Entidades elevará el control sobre los procesos desarrollados por los Técnicos en Ciencias Informáticas en el Centro de Informatización de Entidades del centro CEIGE de la Facultad 3

Para lograr el objetivo planteado se definen los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación a partir de los principales referentes teóricos acerca de los procesos de gestión administrativa en el contexto universitario.

- Realizar el análisis y diseño del módulo Técnicos de Laboratorio para sentar las bases de la implementación del sistema.
- Implementar el módulo Técnicos de Laboratorio del Sistema para la Gestión de los Procesos Administrativos.
- Validar la solución propuesta mediante la aplicación de métricas y pruebas de software.
- Validar la investigación mediante criterio de expertos (método Delphi).

Para la concepción y desarrollo del presente trabajo se toman en cuenta métodos científicos de investigación. Permitiendo entender la esencia del problema abordado y las relaciones de los procesos de gestión que se llevan a cabo en el departamento CEIGE de la Facultad 3.

Métodos teóricos:

- Analítico–Sintético: se utiliza para analizar la bibliografía encontrada referente a la gestión de los procesos administrativos del Centro CEIGE con el objetivo de realizar un estudio y valoración de los mismos, determinando los aspectos esenciales para arribar a conclusiones prácticas y teóricas.
- Histórico-Lógico: se emplea con el objetivo de conocer toda la evolución histórica sobre el contenido de los sistemas de gestión para la Facultad 3 y aquellos sucesos significativos en los procesos de administración, que puedan ser de gran aporte a la investigación.
- Modelación: se utiliza para crear todos los modelos y diagramas que permitan un mejor entendimiento de los procesos del sistema, destacando: el modelo conceptual, la especificación de requisitos, los diagramas de clases del diseño y el modelo de datos.
- Hipotético-deductivo: se utiliza para elaborar la idea a defender y trazar estrategias para cumplirla.

Métodos empíricos:

- Observación: se utiliza para conocer la realidad mediante la percepción directa de los procesos de administración de centro CEIGE.

Técnica de recopilación de información:

- Entrevista: se realiza a las personas encargadas de controlar el proceso de administración de técnicos, con el objetivo de precisar el problema a resolver e identificar el modo de funcionamiento del mismo.

Estructura del contenido

Capítulo 1: Fundamentación Teórica

Describe la fundamentación teórica que sirve como base para la investigación referente a los procesos administrativos del centro. Se tratan los principales conceptos para una correcta comprensión del tema, abarcando el estudio del estado del arte en relación a los procesos de administración de técnicos y medios de laboratorio en el ámbito nacional e internacional. Se describen las tecnologías, lenguajes y herramientas necesarias para el desarrollo del sistema, así como la metodología que guiará el proceso de desarrollo de software.

Capítulo 2: Propuesta de solución

Refleja cada uno de los pilares del desarrollo del sistema propuesto, se identifican los requisitos funcionales y no funcionales, se muestran los artefactos de la metodología de desarrollo de software seleccionada, se describen los diagramas correspondientes al diseño del sistema, el modelo de datos, así como la arquitectura y los patrones usados para la implementación de la propuesta de solución.

Capítulo 3: Implementación y validación de la propuesta de solución

En este capítulo se abordan aspectos relacionados con la implementación del sistema, se describe la nomenclatura usada en la implementación de la solución, la estrategia de pruebas definida a partir de la metodología de desarrollo, con el objetivo de garantizar la validación de la solución propuesta. Se analizan los niveles de pruebas ejecutados y los resultados alcanzados.

Capítulo 1. Fundamentos Teóricos

Introducción

En el presente capítulo se describen los principales conceptos para una mejor asimilación del tema en cuestión. Se estudian los principales sistemas de gestión de la información para procesos de administración y control de técnicos y recursos de laboratorios en Cuba y el resto del mundo. Además, se fundamenta la selección de las tecnologías, lenguajes, metodología y herramientas que guían el desarrollo de la presente investigación.

1.1 El proceso de gestión de la información administrativa relacionado con los Técnicos en Ciencias Informáticas del Centro de Informatización.

Según (Augier, 2005), la información es la parte fundamental de toda entidad para tener un alto nivel de competitividad y posibilidades de desarrollo, es considerada un recurso que se encuentra al mismo nivel que los recursos financieros, materiales y humanos, que hasta el momento habían constituido los ejes sobre los que había girado la gestión empresarial, por lo que constituye un elemento fundamental para el desarrollo, que al transcurrir de los años, ha ocupado, cada vez más, un espacio mayor en la economía de los países a escala mundial.

La gestión de la información aglutina los factores que en cualquier organización interactúan con diferentes actitudes, intereses y expectativas, para acceder y usar productos y servicios de la información. Integra en su organización el conjunto de instancias que la constituyen en función de hacer cumplir: cómo la información se adquiere, registra, almacena, distribuye y usa, cómo el personal designado maneja y hace llegar la información a los usuarios directos, cómo las personas usan la información, desarrollan habilidades informativas y se convierten en divulgadores de la misma, cómo las tecnologías de la información se incorporan y perfeccionan los diferentes procesos de la gestión, todo lo que incide en los costos y beneficios de la organización (Augier, 2005).

(Torres, 2015) manifiesta que la gestión de información es el proceso de organizar la información, evaluar, presentar, comparar los datos en un determinado contexto,

controlar la calidad, que sea oportuna, significativa, exacta y que esté disponible en el momento que se le necesite. Ella se encamina al manejo de la información, documentos, metodologías, informes, publicaciones, soportes y otras actividades en función de los objetivos estratégicos de una organización.

Para el manejo de la información en la presente investigación se decide trabajar con el concepto pronunciado por (Torres, 2015), puesto que el concepto que este aborda sobre la gestión de información, está más enfocado al manejo de la calidad, control y eficacia de la información en función de los objetivos trazados en la presente investigación.

Gestionar correctamente la información administrativa en el proceso desarrollado por los Técnicos en Ciencias Informáticas en el Centro de Informatización de Entidades resulta de gran importancia, puesto que provee a los máximos directivos la información detallada y actualizada en correspondencia con las principales funciones que realizan los mismos en el cumplimiento, control y desarrollo de las actividades que le son asignadas.

Este proceso de gestión comienza con el control de los medios físicos del laboratorio registrados en el control diario, en el cual el técnico que recibe el local, en conjunto con el que entrega, debe verificar el estado técnico y la integridad de los medios antes de hacerse responsable del recibo de la guardia, así como reportar cualquier tipo de incidencia o faltante. Una vez realizado este proceso, el responsable del turno tiene como principales funciones, además de conservar la integridad de los medios y el local recibido, mantener un control del personal que utiliza los servicios del laboratorio y cumplir con las actividades que les son asignadas por parte de los directivos, para obtener una evaluación mensual por su desempeño.

A los TCI se les asignan tareas por parte del asesor de Seguridad Informática y Tecnología, o por el director del centro, dichas tareas deben estar en correspondencia con las funciones definidas para el cargo, así como con las normas organizativas y disciplinarias establecidas para el trabajo en los laboratorios docentes y productivos.

Una vez que los técnicos concluyen una tarea deben reportar el cumplimiento de la misma para su evaluación. Esta evaluación es asignada mediante una escala de excelente (E), bien (B), regular (R) y mal (M). La misma será otorgada por el asesor de y Tecnología o en su defecto el director de centro. Estas son tenidas en cuenta en la evaluación su mensual.

Para la evaluación mensual de los TCI, además de la evaluación de sus tareas, se tienen en cuanta los siguientes componentes: calidad en el cumplimiento de las tareas asignadas, organización y limpieza de los locales de trabajo, cumplimiento de las normas y orientaciones, disciplina laboral y actividades de superación. El jefe inmediato de los técnicos debe realizar una propuesta de evaluación en los componentes antes mencionados. Una vez que se realice la evaluación de cada componente se debe generar el reporte de evaluación, de acuerdo al formato siguiente; deficiente (D), adecuado (A) y superior (S).

Una vez analizado el proceso de gestión de información administrativa de los TCI, se hace necesario realizar un estudio de sistemas similares con el objetivo de obtener características que contribuyan al objetivo de la presente investigación.

1.2 Estudio de sistemas similares

En gran parte del mundo, casi todos los procesos económicos y de control interno se encuentran automatizados para ofrecer una mayor claridad y seguridad, así como una mejor gestión sobre dichos procesos. Muchos de estos programas se encargan de gestionar el control de técnicos y recursos de laboratorios, por esta razón se hace necesario realizar un análisis sobre algunos de estos sistemas que poseen características similares a las requeridas en el presente problema, con el objetivo de comprobar si es posible adaptarlo a los requerimientos de la problemática planteada.

Sistemas similares usados en el mundo

Sistema de Control y Administración de Activos Fijos (SCAAF): su objetivo es mantener administrado de forma integrada y detallada los activos fijos de la organización, controlando la ubicación física, valores históricos y depreciación

acumuladas. De igual forma permite proporcionar información para mejorar el costo de mantenimiento de los activos ("Alcofin," 2012). El sistema permite:

- Consultar con rapidez todos los datos.
- Aplicación automática y por centros de costo, del gasto de depreciación.
- Aplicación automática de la revaluación y su depreciación (también por centros de costo).
- Avisos automáticos del sistema.
- Control automático de acciones a tomar para el mantenimiento preventivo.
- Avisos automáticos sobre término de la vida útil estimada del activo.

Sistemas similares utilizado en la UCI

Asistente planificador para la asignación de actividades: El sistema permite la planificación, control y asignación de tareas en los diferentes departamentos de la UCI. Actúa como asistente planificador con el fin de lograr de manera automatizada la planificación, el control y asignación de actividades a cada usuario en dependencia de sus afectaciones de manera tal que se pueda optimizar el tiempo y recurso empleado en este proceso (Rodríguez, 2015).

Sistema de control de entrega y recibo de activo fijo (CERAF): permite establecer las bases para lograr un mayor control y seguridad sobre el proceso de entrega y recibo de activos, logrando así reducir gastos y disminuir el tiempo en que ocurren las actividades realizadas a los directivos de esta área, tener un control elevado sobre todos los medios, dando la posibilidad, además, de conocer el estado de las incidencias y de los reportes generados para cada activo (Saborit & Alemán, 2015).

Ventajas del sistema:

- Cuenta con un historial que se encarga de gestionar todo lo relacionado con los reportes e incidencias sobre los activos.
- Permite analizar los reportes y observaciones realizados sobre los activos, así como comparar los informes de entrega de un área en diferentes períodos.
- Contribuye además a disminuir el tiempo de respuesta de las diferentes actividades.

Sistema de entrega de guardia para los laboratorios de la UCI

Es un sistema que gestiona usuarios, áreas, laboratorios, puesto de trabajo, problemas, afectaciones de los laboratorios y algunos reportes. Sin embargo, este sistema tiene limitaciones: no maneja los números de medios básicos de los activos, no permite el registro de los movimientos de medios y no incluye otras funcionalidades relacionadas con el resto de las funciones que realizan los TCI (Mar Cornelio, 2011).

Después de realizar un estudio de los sistemas de gestión mencionados anteriormente, los cuales incluyen dentro de sus funcionalidades el control de medios básicos y actividades; describiéndose sus principales características. Se concluyó que los mismos, no satisfacen las necesidades existentes en el Centro de Informatización de Entidades, pues estos sistemas tienen una concepción económica para la gestión de dichos medios, mientras que la presente investigación está enfocada a gestionar los procesos de control de entrega-recibo de los medios básicos del laboratorio, realizados por los TCI, así como su evaluación de desempeño.

Estos sistemas en su mayoría, son softwares propietarios, lo que constituye un aspecto negativo para el país y para la UCI en cuanto a su soberanía tecnológica. Para una mejor comprensión, se muestra una comparación de estos sistemas en la siguiente tabla.

Sistemas similares		Bases de Datos Compatibles	Sistema Operativos Compatibles	Tipo de Aplicación	Licencia Comercial
Internacionales	Sistema de Control y Administración de Activos Fijos (SCAAF)	Microsoft SQL Server	Microsoft Windows	Cliente-Servidor	Privativa

Nacionales(UCI)	CERAF	Postgres SQL	Microsoft Windows, sistema operativo linux	Cliente-Servidor	Software libre
	Asistente planificador para la asignación de actividades	Postgres SQL	Microsoft Windows, sistema operativo linux	Cliente-Servidor	Software libre
	Sistema de entrega de guardia para los laboratorios de la UCI	Postgres SQL	Microsoft Windows, sistema operativo linux	Cliente-Servidor	Software libre

TABLA 1 COMPARACIÓN DE COMPATIBILIDAD DE LOS SISTEMAS SIMILARES. FUENTE: ELABORACIÓN PROPIA

.

	Gestionar Técnico	Gestionar Control de medios	Gestionar Evaluación mensuales	Entrega-Recibo de guardia	Gestionar Tarea	Gestionar movimientos
Sistema de Control y Administración de Activos Fijos (SCAAF)	no	sí	no	no	No	no

Sistema de entrega de guardia para los laboratorios de la UCI	sí	sí	no	sí	No	no
CERAF	sí	sí	no	sí	No	no
Asistente planificador para la asignación de actividades	no	no	no	no	sí	no

TABLA 2 COMPARACIÓN DE FUNCIONALIDADES DE LOS SISTEMAS ESTUDIADOS. FUENTE: ELABORACIÓN PROPIA.

Por otra parte, el sistema CERAF aporta funcionalidades para la gestión y control de medios, pero no cuenta con las principales funciones para la gestión y control de los técnicos, sus tareas y evaluaciones. Por lo que se decide desarrollar un nuevo sistema, que sea capaz de gestionar los procesos desarrollados por los técnicos en ciencias informáticas en el Centro de Informatización de Entidades.

Para un buen desarrollo del sistema es necesario hacer uso de una metodología de desarrollo de software que guie el proceso de desarrollo de software.

1.3 Metodología de desarrollo del software

(Cataldi, 2000) reconoce que para desarrollar un proyecto de software es necesario establecer un enfoque disciplinado y sistemático. Las metodologías de desarrollo de software influyen directamente en el proceso de construcción y se elaboran a partir del marco definido por uno o más ciclos de vida.

Para que un software tenga éxito en su desarrollo es necesario hacer uso de una metodología que guíe el ciclo de vida del mismo. Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de crear un determinado producto.

Estas pretenden guiar a los desarrolladores para garantizar la entrega del mismo con la calidad y tiempo requerido (Carrillo, Pérez, & Rodríguez, 2008).

Estas metodologías se han entablado en un intenso debate entre dos grandes corrientes. Por un lado, las denominadas metodologías tradicionales, centradas en el control del proceso, con un riguroso seguimiento de las actividades involucradas en ellas. Por otro lado, las metodologías ágiles, centradas en el factor humano, en la colaboración y participación del cliente en el proceso de desarrollo y a un incesante incremento de software con iteraciones muy cortas (Tinoco, 2010).

Las metodologías ágiles se adaptan mejor a las características del proyecto a desarrollar, están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes, indicada para proyectos a corto plazo y con un equipo de desarrollo pequeño.

La UCI propone para el desarrollo de sus proyectos de software la metodología AUP-UCI, en el presente trabajo se hará uso de esta en su versión 1.2.

AUP o Agile Unified Process en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles como son: Desarrollo Dirigido por Pruebas (test driven development - TDD en inglés), Modelado ágil, Gestión de Cambios ágil, Refactorización de Base de Datos para mejorar la productividad (T. Rodríguez, 2015).

Esta metodología consta de 4 fases las cuales son:

- Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.

- Transición: El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

Variación de AUP para la UCI

De estas fases se decide para el ciclo de vida del proyecto mantener la fase de inicio, pero modificando el objetivo de la misma, es decir: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. 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.

Las restantes faces se unifican en una a la cual se tratará como Ejecución en la cual 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 modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto y se agrega una nueva fase de Cierre en la que se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (T. Rodríguez, 2015).

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Para una mejor compresión de las disciplinas se muestra la siguiente Tabla (T. Rodríguez, 2015).

Disciplinas Variación AUP-UCI	Objetivos Disciplinas (Variación AUP-UCI)
Modelado de negocio	El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar

	para tener garantías de que el software desarrollado va a cumplir su propósito.
Requisitos	El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
Análisis y diseño	En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
Implementación	En la Implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
Pruebas interna	En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas
Pruebas de liberación	Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
Pruebas de Aceptación	Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Despliegue (Opcional)	Constituye la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente.

TABLA 3 DESCRIPCIÓN DE LAS DISCIPLINAS DE LA METODOLOGÍA. FUENTE: ELABORACIÓN PROPIA.

En el presente trabajo se hace uso de las disciplinas; modelado del negocio, análisis y diseño, implementación y pruebas de aceptación.

Escenarios para la disciplina de requisitos:

La metodología propone cuatro escenarios para las disciplinas de requisitos. El presente trabajo estará regido por el escenario 3, su selección está basada en que dentro de sus características cumple, que se está desarrollando un módulo, el cual será integrado en un sistema más complejo, proporcionando objetividad, solidez, y su continuidad.

Al ser identificada la metodología a utilizar, así como las fases y disciplinas que rigen la misma, se hace necesario describir las herramientas y las principales tecnologías a utilizar en el proceso de desarrollo del software.

1.4 Lenguaje y herramienta de modelado.

Lenguaje unificado de modelado 2.1: (Unified Modeling Language, UML 2.1) para el modelado del proceso de negocio de la presente investigación fue aplicado el lenguaje de modelado de sistemas de software UML. Constituye un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios, funciones del sistema y aspectos concretos (Larman, 1999).

Herramienta de modelado: para el modelado de los procesos de esta investigación se hace uso del Visual Paradigm-UML 8.0. Esta herramienta fue creada para utilizarse tanto en el entorno de Windows como en el entorno de Linux. Soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Este software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los

tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (P. López, 2014).

1.5 Ambiente de desarrollo

Lenguajes de programación: el lenguaje del lado del servidor a usar en el desarrollo web es PHP 5.5 (Hypertext Pre-processor). Es un lenguaje de secuencia de comandos de servidor que fue diseñado específicamente para la Web. También se conoce que es un producto de código abierto, lo que quiere decir que se puede acceder a su código. Este lenguaje fue concebido en 1994 y es fruto del trabajo de un hombre, Rasmus Lerdorf, que ha sido adoptado por otras personas de talento y ha experimentado tres trasformaciones importantes hasta convertirse en el producto actual (Rubio, 2006).

En cuanto a los lenguajes de programación usados del lado del cliente para el desarrollo de la investigación, se encuentran: JavaScript, es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Eguiluz, 2015).

El Lenguaje de Marcado de Hipertexto (HTML de sus siglas en inglés) se utiliza para crear páginas web, una misma página se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo. Una de las características fundamentales es su universalidad, esto posibilita que prácticamente cualquier ordenador, independientemente del sistema operativo que tenga, pueda leer o interpretar una página web (De Luca & De Luca, 2010).

Las Hojas de Estilo en Cascada del inglés Cascading Style Sheets (CSS), es un lenguaje simple, creado para controlar como se va a mostrar un documento en la pantalla, la forma en que se imprimirá e incluso la manera en que va a ser

pronunciada la información presente en documentos a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. Se utiliza para dar estilo a documentos HTML, XML (acrónimo de Extensible Markup Language) y XHTML (acrónimo de Extensible HTML), para separar el contenido de la presentación (Sawyer, 2009).

Luego de definir los lenguajes a utilizar se seleccionan los marcos de trabajo los cuales definen una serie de buenas prácticas aplicables a un proceso de codificación

Marco de trabajo (Framework): contiene numerosos archivos y directorios cuyo propósito es facilitar la creación de aplicaciones incorporando diferentes funcionalidades ya desarrolladas y probadas, esto para un determinado lenguaje de programación (Orichijuan, 2014).

Symfony2: es un proyecto PHP de software libre que permite crear aplicaciones y sitios web rápidos y seguros de forma profesional, su código y el de todos los componentes y librerías que incluye se publican bajo la licencia MIT (Massachusetts Institute of Technology) de software libre. Su arquitectura interna está completamente desacoplada, posibilitando reemplazar o eliminar fácilmente aquellas partes que no encajan en el desarrollo de un proyecto determinado. Siendo este independiente del sistema gestor de bases de datos, fácil de instalar y configurar en la mayoría de plataformas (Pacheco, 2012). Para el desarrollo del sistema se utilizará la versión Symfony 2.8.

Bootstrap: permite crear interfaces web con CSS y JavaScript que adaptan la interfaz dependiendo del tamaño del dispositivo en el que se visualice de forma nativa, es decir, automáticamente se adapta al tamaño de un ordenador o de una tablet sin que el usuario tenga que hacer nada. Incluye varios estilos predefinidos fáciles de configurar: botones, menús desplegables, entre otros (Cimo, 2015). Por lo antes expuesto se propone utilizar la versión Bootstrap 3.1, la cual permite el uso de interfaces capaces de adaptarse automáticamente al tamaño de cualquier dispositivo.

Herramientas de desarrollo propuestas para la confección de la solución:

- NetBeans IDE 8.0 es una herramienta pensada para escribir, compilar, depurar y ejecutar programas. Tiene soporte para crear interfaces gráficas de forma visual, crear aplicaciones para móviles, desarrollar aplicaciones web y además estas funcionalidades son ampliables mediante la instalación de paquetes adicionales. Es un proyecto de código abierto, permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. La plataforma NetBeans es una base modular y extensible usada como una estructura de integración (Vásquez, 2012).
- PostgresSQL 9.1 es un gestor de base de datos de código abierto, distribuido bajo licencia Berkeley Software Distribution (BSD) y con su código fuente disponible libremente, maneja una serie de características y funcionalidades que permiten un trabajo flexible sobre los datos, como son la alta concurrencia, claves ajenas y disparadores (Avila, 2015).
- Servidor Apache 2.4.9 es un servidor web de código libre robusto cuya implementación se realiza de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales. El proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo que, usando Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada. El código fuente es totalmente abierto. Su arquitectura modular, construida sobre un pequeño núcleo, se adapta a las necesidades específicas de cada usuario (Mateu, 2008).
- Geany 0.16 es un editor de texto ligero con características básicas de entorno de desarrollo integrado que solo depende en menor medida de otros paquetes. Está disponible para múltiples sistemas operativos y tiene soporte para muchos lenguajes de programación distintos. Para su funcionamiento solo requiere las bibliotecas de tiempo de ejecución GTK 2 (acrónimo de Gimp Tool Kit) (Geany, 2015). Posee algunas características como auto completamiento, coloreado de sintáxis, compatible con la mayoría de lenguajes.

1.6 Validación mediante criterios de expertos: Método Delphi.

El Delphi es una metodología estructurada para recolectar sistemáticamente juicios de expertos sobre un problema, procesar la información y a través de recursos estadísticos, construir un acuerdo general de grupo (García Valdés & Suérez Marín, 2013).

Este método es considerado como uno de los métodos subjetivos de pronosticación más confiables. El procesamiento estadístico y matemático de la información es la característica más importante del método que lo diferencia del resto de los métodos de pronosticación de base subjetiva, ya que la decisión final que toma el investigador es un criterio fuertemente avalado por la experiencia y conocimiento del colectivo consultado, así como por indicadores objetivos (Win2PDF, 2015).

Para la aplicación de este método se deben tener en cuenta dos aspectos fundamentales, el primero la selección de los expertos y el segundo la elaboración y análisis del cuestionario a aplicar. Atendiendo a estos aspectos (García Valdés & Suérez Marín, 2013) plantea las siguientes fases:

I. Preparatoria.

Selección de expertos.

- Preparación del instrumento.
- Decisión de la vía de consulta.

II. Consulta
    - Rondas de consulta.
    - Procesamiento estadístico.
    - Retroalimentación.

III. Consenso
    - Construcción de consenso.
    - Reporte de resultados.

Por su importancia se realizará marcado énfasis en la fase de consulta y específicamente en el procesamiento estadístico, esta fase incluye el tratamiento

matemático y estadístico al cuestionario aplicado. Se propone por (Win2PDF, 2015) el uso del siguiente procedimiento:

1. Determinación de la media aritmética por pregunta:

   $$\bar{C_j} = \frac{\sum_{i=1}^{m_j} C_{ij}}{m_j}$$

   Donde:

   m: cantidad de expertos.

   n: cantidad de preguntas.

   mj: cantidad de expertos que evalúan la pregunta j.

   Cij: evaluación en puntos de la pregunta j realizada por el experto j.

2. Grado de concordancia de los expertos por pregunta:

   $$\sigma_j^2 = \frac{\sum_{i=1}^{m_j}(C_{ij} - \bar{C_j})^2}{m_j - 1}$$

   A partir de la fórmula anterior para determinar la varianza, se determina el coeficiente de variación (v_j), este coeficiente es una medida del grado de concordancia de los expertos por cada pregunta, donde mientras mayor sea el valor de v_j menor será el grado de concordancia de los expertos.

   $$v_j = \frac{\sqrt{\sigma_j^2}}{C_j}$$

En diversas investigaciones se utiliza el coeficiente de concordancia de Kendall, lo cual proporciona un coeficiente de concordancia global de todas las preguntas que se incluyen en el cuestionario.

Conclusiones del capítulo

- Los procesos desarrollados por los TCI presentan insuficiencias que limitan un adecuado control de los mismos.
- El estudio realizado de los sistemas existentes relacionados con el tema de la presente investigación arrojó la necesidad de desarrollar una solución capaz de gestionar íntegramente los procesos desarrollados por los TCI.

- El desarrollo de la investigación estuvo guiado por la metodología AUP-UCI en su escenario 3.
- Para la implementación del sistema de seleccionaron las herramientas: Symfony 2.8, Bootstrap 3.1 como marcos de trabajo, como lenguaje del lado del servidor PHP 5.5 y del lado del cliente HTML 5, CSS3 y JavaScript, como Entorno de Desarrollo Integrado Neatbeans 8.0, servidor de aplicaciones web Apache 2.4.9.

Capítulo 2. Propuesta de solución

Introducción

En este capítulo se describen los artefactos de trabajo propuestos por AUP-UCI para el desarrollo de la solución propuesta en las fases de: requisitos, análisis y diseño. En la etapa de requisitos se identifican y describen los requisitos funcionales, no funcionales. En el análisis y el diseño, se detallan los principales artefactos de dicha disciplina, destacando el modelo de conceptual, los diagramas de clases, así como el modelo de datos. Igualmente, se especifica la utilización de los patrones de arquitectura y de diseño para la concepción de la solución.

Descripción de los procesos de negocio

Dentro de los procesos desarrollados por los TCI se encuentran el proceso de entrega-recibo de laboratorio, asignación, seguimiento y control de tareas, control de los movimientos de medios básicos, asignación evaluación mensual.

A continuación, se muestra un ejemplo de diagramas de proceso de negocio; asignación, seguimiento y control de tareas donde se define el flujo de las principales actividades realizadas en este proceso. Ver los demás descripciones y diagramas en el anexo 1.

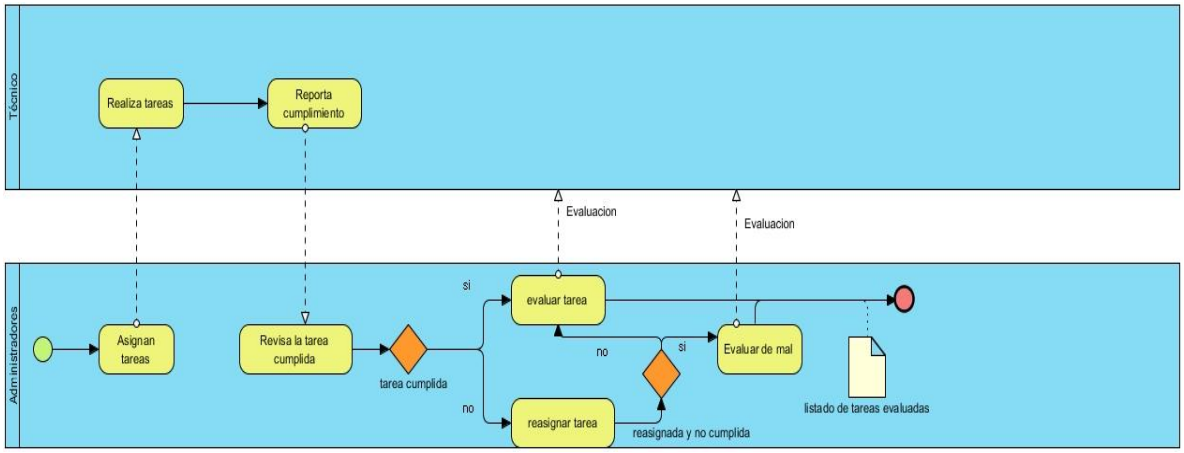

ILUSTRACIÓN 1 DIAGRAMA DE PROCESO DE NEGOCIO: ASIGNACIÓN Y SEGUIMIENTO DE TAREAS. FUENTE: ELABORACIÓN PROPIA.

Modelo conceptual

Un modelo conceptual explica los conceptos significativos en el dominio del problema para entender los principales términos asociados a los procesos internos de una entidad, identificando los atributos y las asociaciones existentes entre ellos. Se realiza el mismo, para tener una representación de entidades, ideas, conceptos u objetos identificados, en los procesos realizados por los TCI en el centro de informatización de entidades. Sirviendo además como fuente de obtención de requisitos (Pantoja, 2005). La siguiente imagen muestra los conceptos fundamentales relacionados con el desarrollo de la presente investigación.

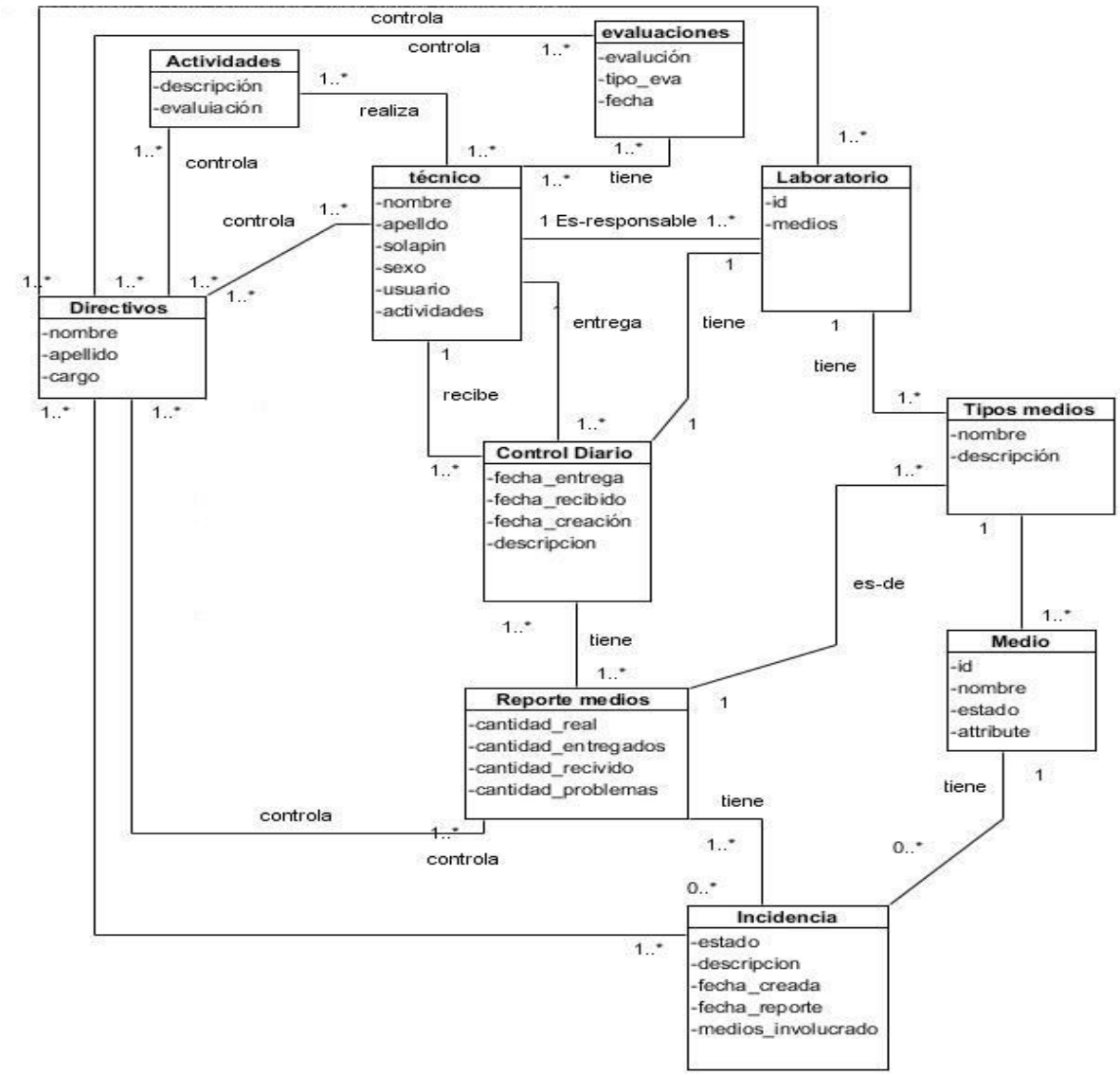

ILUSTRACIÓN 1 MODELO CONCEPTUAL DEL NEGOCIO. FUENTE: ELABORACIÓN PROPIA.

2.1 Requisitos

El propósito de esta disciplina es hacer que los requerimientos alcancen un estado óptimo antes de desarrollar la fase de diseño en el proyecto, definiendo, las características de un sistema que satisfaga las necesidades del cliente y que se integre con éxito en el entorno en el que se explote. Además de gestionar las líneas base y las peticiones de cambios que se produzcan en la especificación de requisitos, manteniendo la trazabilidad entre los requisitos y otros productos del desarrollo (Pressman, 2010).

Para la definición de los requisitos funcionales del sistema se hace necesario recopilar información detallada de las funcionalidades que se quieren implementar para que el sistema cumpla las necesidades del cliente.

Técnica para la captura de requisitos funcionales

La captura de requisitos es la actividad mediante la cual, el equipo de desarrollo de un sistema de software, extrae de cualquier fuente de información las necesidades que debe cubrir dicho sistema (Méndez, 2009), las técnicas utilizadas son:

- Entrevistas: fueron realizadas a las siguientes personas; Msc. Yoansy López Reyes quien se desempeña como director del centro CEIGE y además tutor de la presente investigación y al Especialista en Ciencias Informáticas Miguel Ángel Sánchez Palmero quien dirige el proceso realizado por los TCI, partiendo de un conjunto de preguntas para obtener un mejor entendimiento de lo que se quiere lograr. Los especialistas platearon la necesidad de que la solución contara con las funcionalidades siguientes:
    - Un mecanismo de autenticación que permita garantizar la presencia física del responsable en el área.
    - Un historial de los reportes generados en cada local y de las incidencias que en estos se generen.
    - Un mecanismo que permita asignar actividades a los técnicos, así como sus evaluaciones.
    - Un mecanismo que permita evaluar el desempeño mensual de los técnicos en el cumplimiento de sus principales funciones.

- Tormenta de ideas: consistió en realizar reuniones con el cliente, donde como primer paso se sugirieron toda clase de ideas, acerca de los procesos que realizan los TCI, después de recopilar todas las ideas se realizó un análisis detallado de cada propuesta.
- Observación: permitió observar la forma en que se llevan a cabo los procesos realizados por los TCI y verificar que realmente se sigan todos los pasos especificados por el cliente, una vez que se realizó la entrevista.

A continuación, se definen los requisitos funcionales identificados luego de aplicar las técnicas anteriores.

Definición de los requisitos funcionales

Los requisitos funcionales describen lo que debe hacer el sistema. Estos requerimientos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactarlos. Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de como este se debe comportar en situaciones particulares (Pressman, 2010).

En la siguiente tabla se definen los requisitos funcionales identificados agrupados mediante el uso del patrón CRUD (Create-Read-Update-Delete).

	RF	Requisito Funcional	Complejidad
	RF 1	**Gestionar Usuario**	
1	RF1.1	Adicionar usuario	Media
2	RF1.2	Eliminar usuario	Media
3	RF1.3	Modificar usuario	Media
4	RF1.4	Listar usuario	Media
5	RF1.5	Asignar token	Media
6	RF1.6	Generar token	Alta
	RF 2	**Gestionar medio**	
7	RF2.1	Adicionar medio	Baja
8	RF2.2	Eliminar medio	Baja
9	RF2.3	Modificar medio	Baja
10	RF2.4	Listar medios	Baja
	RF 3	**Gestionar Tarea**	
11	RF3.1	Asignar tarea	Media
12	RF3.2	Evaluar tarea	Media

13	RF3.3	Listar tareas por técnicos	Baja
14	RF3.4	Reportar tarea cumplida	Baja
15	RF3.5	Eliminar tarea	Baja
	RF 4	**Gestionar reporte**	
16	RF4.1	Adicionar reporte	Media
17	RF4.2	Eliminar reporte	Media
18	RF4.3	Modificar reporte	Media
19	RF4.4	Listar reporte	Media
20	RF4.5	Buscar reporte por tipo	Media
	RF 5	**Gestionar evaluaciones**	
21	RF5.1	Proponer evaluación Mensual	Media
22	RF5.2	Listar evaluación	Baja
23	RF5.3	Listar evaluación por técnico	Media
24	RF5.4	Modificar evaluación	Media
25	RF5.5	Eliminar evaluación	baja
26	**RF 6**	**Autenticar Usuario**	Alta
	RF 7	**Adicionar Informe de Entrega y Recibo (IER)**	Alta
28	RF7.1	Adicionar IER	Alta
29	RF7.2	Listar IER del día	Alta
30	RF7.3	Listar historial de IER	Alta
	RF 8	**Gestionar incidencia**	
31	RF8.1	Adicionar incidencia	Alta
32	RF8.2	Listar incidencias pendientes	Baja
33	RF8.3	Listar todas las incidencias	Baja
	RF 9	**Gestionar Local**	
34	RF9.1	Adicionar local	Baja
35	RF9.2	Listar local	Baja
36	RF9.3	Modificar local	Media
37	RF9.4	Eliminar local	Baja
38	**RF 10**	**Registrar movimientos de medios**	Alta
	RF 11	**Gestionar Componente**	
39	RF11.1	Adicionar componente	Alta
40	RF11.2	Listar componente	Baja
41	RF11.3	Modificar componente	Media
42	RF11.4	Eliminar componente	Baja
43	**RF12**	Exportar IER a PDF	Alta
44	**RF13**	Exportar modelo de plan de trabajo a PDF	Alta
45	**RF14**	Exportar modelo de evaluaciones mensual	Alta

TABLA 4 REQUISITOS FUNCIONALES. FUENTE: ELABORACIÓN PROPIA.

Especificación de Requisito <Asignar Tarea>

La descripción de requisitos especifica con mayor detalle la acción que realiza el mismo en el sistema. Contiene los pasos o las actividades que se deberán realizar para darle cumplimiento al requisito, así como las restricciones a las que está sujeto y las posibles respuestas del sistema. Es una herramienta ampliamente utilizada en el análisis y diseño de un sistema informático (Pressman, 2010).

La siguiente tabla muestra la especificación de requisitos asignar tarea.

Precondiciones	Se debe estar logueado como administrador Debe existir a menos un técnico	
Flujo de eventos		
Flujo básico << Asignar Tarea >>		
1.	El usuario selecciona la opción asignar tarea del menú Técnico	
2.	El sistema muestra un listado de todas las tareas asignadas por meses a cada técnico	
3.	El usuario selecciona el botón asignar tarea	
4	El sistema muestra una ventana donde el usuario especificará el nombre, descripción y el técnico a al que se le asigna.	
5	El usuario especifica los datos y selecciona la opción aceptar	
6	El sistema agrega la tarea a la lista de tareas asignadas	
Pos-condiciones		
1.	Se muestra la lista de tareas actualizada	
Flujos alternativos		
Flujo alternativo		
•		
•		
Pos-condiciones		
•	N/A	
Validaciones		
•		
Conceptos	**Tarea**	**Visibles en la interfaz** Nombre de la tarea Descripción de la tarea Asignado a Asignado por Estado Fecha de cumplimiento

TABLA 5 ESPECIFICACIÓN DE REQUISITOS ASIGNAR TAREA. FUENTE: ELABORACIÓN PROPIA.

Validación de requisitos funcionales

Esta actividad se realizó con el objetivo de garantizar que los requisitos fueran correctos y cumplieran con las necesidades del cliente. Obteniendo en todo momento la conformidad del mismo, la misma se realizó mediante siguientes las técnicas:

Validación de requisitos mediante prototipos de interfaz de usuario: se presentaron los prototipos elaborados a los responsables, para corroborar que responden a las necesidades y aspiraciones identificadas en la obtención de los requisitos. Para ello, se desarrollaron varios escenarios posibles con el auxilio de juegos de datos, de forma tal que se visualizaron las diferentes funcionalidades que tendría el sistema. Teniendo como resultado que todos los requisitos identificados fueron aceptados. A continuación, se muestra un ejemplo de los prototipos utilizados. Prototipo asignar tarea.

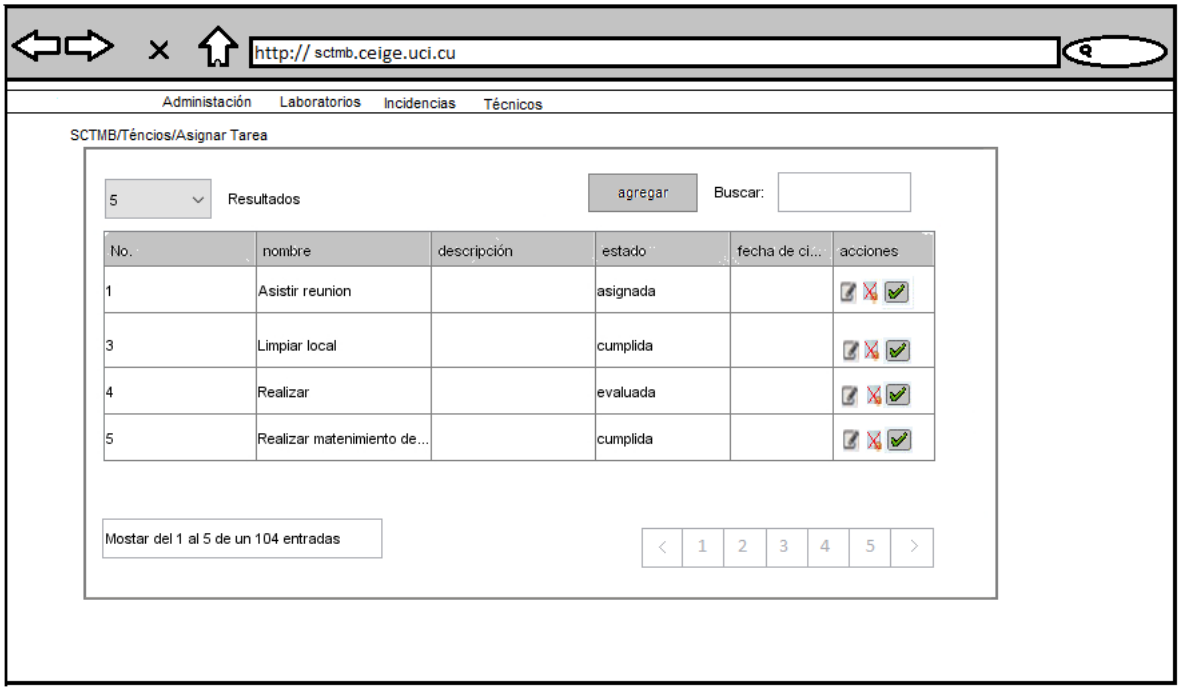

ILUSTRACIÓN 2 PROTOTIPO DE INTERFAZ DEL REQUISITO: ASIGNAR TAREA. FUENTE: ELABORACIÓN PROPIA.

Revisión técnica formal: este proceso se realizó en conjunto con el cliente para revisar detalladamente el buen funcionamiento del sistema. El objetivo fundamental

de estas revisiones es encontrar errores durante el proceso de desarrollo de software. Todos los errores u omisiones que surgieron se señalaron durante la revisión y se corrigieron satisfactoriamente.

Requisitos no funcionales

Los requisitos no funcionales (RNF) describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema; incluyen restricciones como el tiempo de respuesta, la precisión, recursos consumidos y seguridad (Pressman, 2010).

RNF. 1 Usabilidad

RNF. 1.1 El idioma de todas las interfaces de la aplicación será español. Los errores cometidos por el usuario les serán notificados. El sistema expondrá el menú general desde cualquiera de sus páginas.

RNF. 2 Confiabilidad

RNF. 2.1 El sistema estará disponible durante 24 horas, los 7 días de la semana, los 365 días del año.

RNF. 2.2 Ante el fallo de una funcionalidad del sistema, el resto de las funcionalidades que no dependen de esta deberán seguir funcionando.

RNF. 3 Mantenibilidad

RNF. 3.1 La codificación del sistema será estándar y las funcionalidades serán comentadas.

RNF. 4 Portabilidad

RNF. 4.1 El sistema deberá adaptarse al entorno operativo siempre y cuando este cumpla con las características de la aplicación y debe coexistir con otros sistemas sin dificultad.

RNF.5 Hardware

RNF. 5.1 Las estaciones de trabajo deberán poseer como mínimo un ordenador que sea capaz de ejecutar Firefox en cualquier versión.

RNF.6 Software

RNF. 6.1 Las estaciones de trabajo deberán poseer un navegador web. Mozilla Firefox 34.0 o una versión superior.

RNF. 6.2 El servidor de aplicaciones web deberá ser Apache 2.0 y el sistema operativo Linux o Windows,

RNF. 6.3 El servidor de base de datos deberá ser PostgreSQL 9.1.

Para que el desarrollo de un proyecto de software concluya con éxito, es importante que antes de comenzar a codificar lo que constituirá la solución, se tenga una completa y plena comprensión de los procesos relacionados con el software y un diseño correctamente elaborado para el entendimiento de los desarrolladores. En el siguiente epígrafe se describen los artefactos generados para la fase de análisis y diseño que propone la metodología, así como la evidencia del uso de los patrones de arquitectura y de diseño.

2.2 Análisis Y Diseño

El análisis permite definir en detalle el ámbito del software y se crean modelos de los requisitos de datos, flujo de información, flujo de control y del comportamiento operativo. Además, la etapa de diseño permite describir como el sistema va a satisfacer los requisitos. El resultado obtenido de la etapa de diseño facilitará la implementación posterior de la solución, pues proporciona la estructura básica y cómo, los diferentes componentes actúan y se relacionan entre sí (Pressman, 2010).

Descripción de la arquitectura

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC. El sistema a desarrollar respeta dicha arquitectura, siendo el controlador el encargado de gestionar el flujo de la petición realizada a través de la clase GTMlabController.php, pidiendo al modelo aquello que el usuario solicita (ejemplo MedioEntity.php), y devuelve como respuesta una representación del modelo a través de la vista (ejemplo GestionarMedio.html.twig). La siguiente imagen

ilustra la cooperación entre estos tres objetos dentro de la solución propuesta.

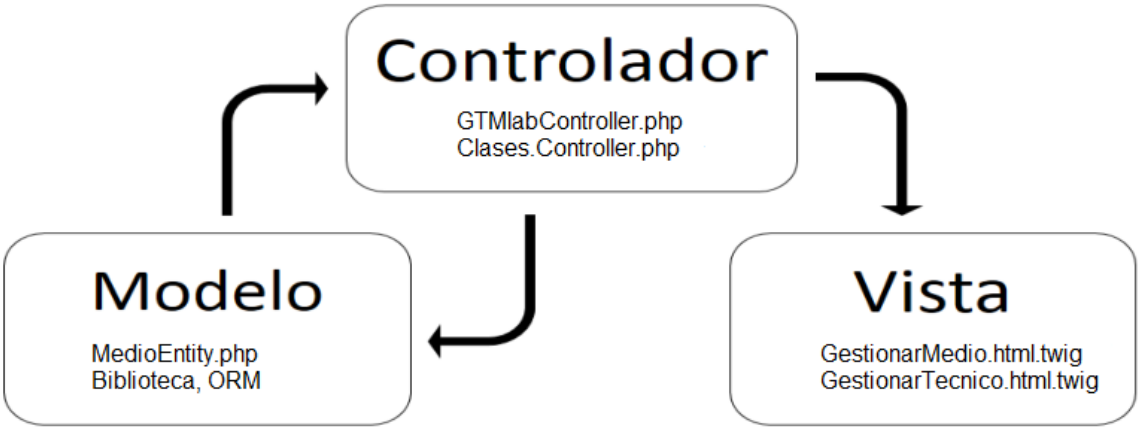

ILUSTRACIÓN 3 PATRÓN MODELO VISTA CONTROLADOR. FUENTE: ELABORACIÓN PROPIA.

El controlador

Un controlador es una función PHP que se encarga de obtener la información de la petición HTTPs y de generar y devolver la respuesta HTTPs (en forma de objeto de tipo "respuesta" (response) de Symfony2). El controlador contiene toda la lógica que la aplicación necesita para generar el contenido de la página. Sus componentes son: el controlador frontal, los filtros, las acciones y los objetos solicitud (request), respuesta (response) y sección (session) (Fabien, 2011). La siguiente imagen ilustra el funcionamiento de dichos componentes.

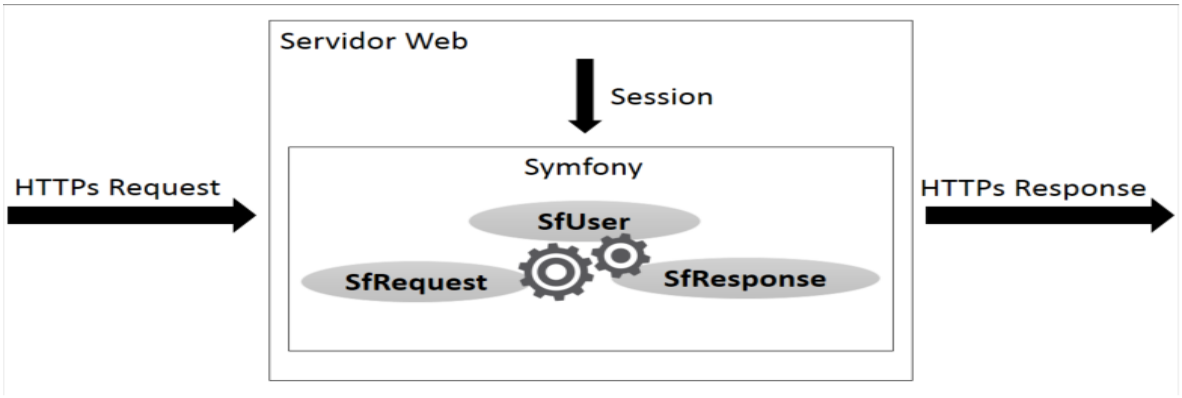

ILUSTRACIÓN 4 MODELO DE FUNCIONAMIENTO DE SYMFONY2 (CAPA CONTROLADORA). FUENTE: (FABIEN, 2011)

La vista

La generación de vistas en Symfony se realiza según lo establecido por el patrón de diseño denominado decorador. Este patrón responde a la necesidad de añadir dinámicamente funcionalidad a un objeto (Fabien, 2011). En Symfony dicho objeto sería la plantilla con la que se renderiza una determinada acción de algún módulo. La funcionalidad añadida dinámicamente sería el resto del documento HTML, definido en algunos de los ficheros alojados en el directorio apps/nombre_aplicacion/plantillas.

El modelo

La capa del modelo es la que se encarga de permitir el acceso a datos, logrando así una abstracción de la base de datos. Esta capa tiene entre sus componentes: el ORM (Mapeo relacional de objetos), los formularios, las extensiones propias que el programador realice de las clases del ORM y las clases y funciones propias que el programador construya para implementar la lógica de negocio (Fabien, 2011).

La relación de todos los componentes presentes en cada capa se muestra en la siguiente imagen.

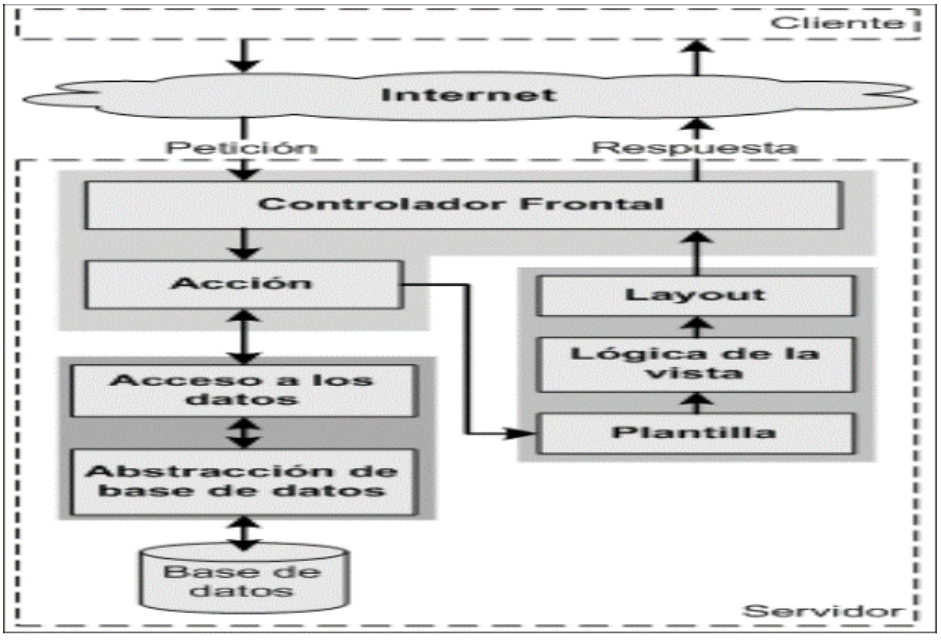

ILUSTRACIÓN 5 FLUJO DE TRABAJO DE SYMFONY. FUENTE: (FABIEN, 2011).

Diagrama de Clases del diseño

Las clases del diseño sirven para describir la estructura del sistema mostrando sus clases, atributos y las relaciones entre ellos. A continuación, se muestra un ejemplo de uno de los diagramas de clases del diseño creados.

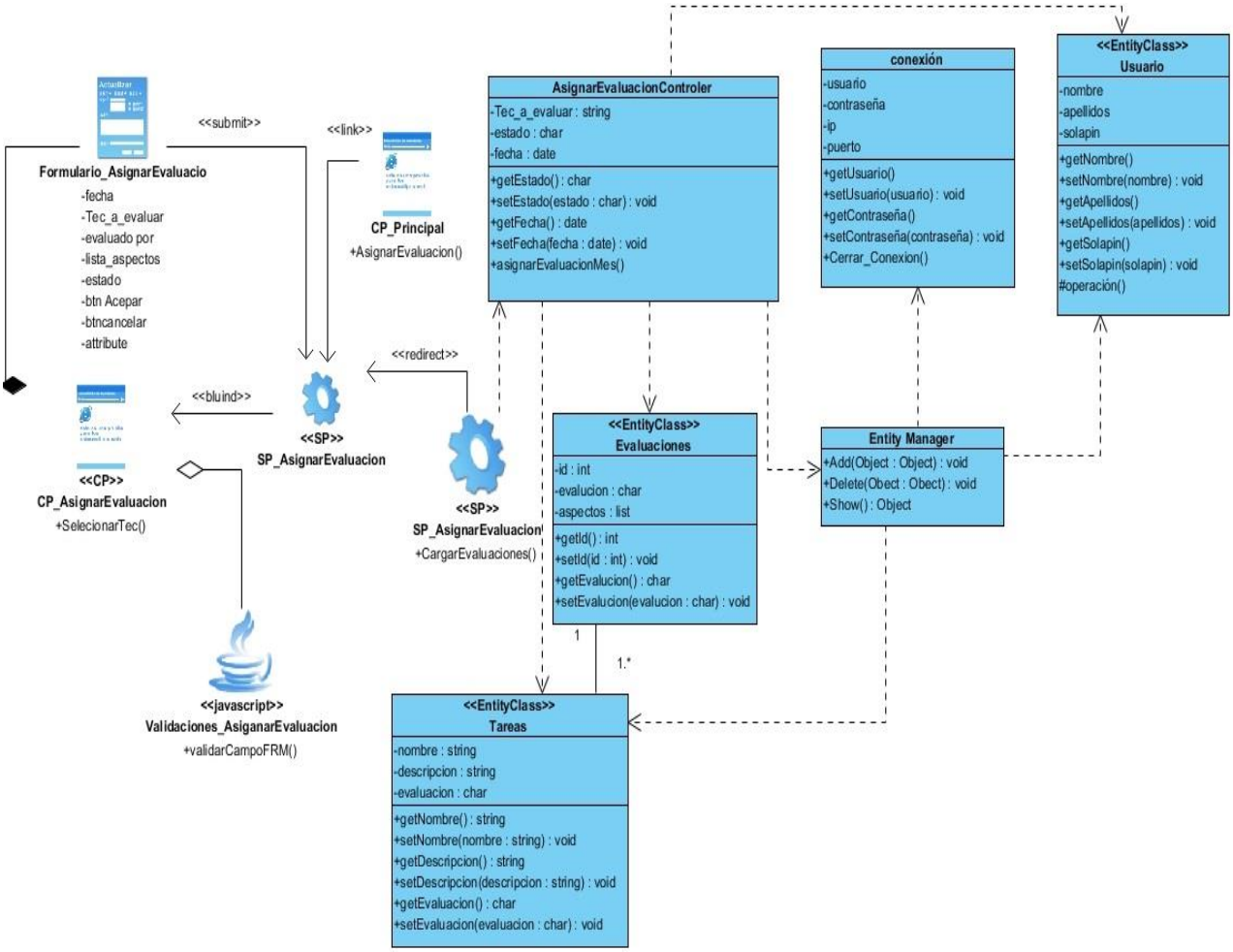

ILUSTRACIÓN 6 DIAGRAMA DE CLASES DEL DISEÑO DEL REQUISITO: ASIGNAR EVALUACIÓN. FUENTE: ELABORACIÓN PROPIA.

Los diagramas de clases del diseño generados muestran la utilización de los patrones de diseños presentes en las clases propuestas. A continuación, se describen el uso de los mismos para el diseño de las clases del sistema.

Utilización de los patrones de diseño

Un patrón de diseño define un esquema de refinamiento de los subsistemas o componentes dentro de un sistema, o las relaciones entre estos. Este describe una estructura común y recurrente de componentes interrelacionados, que resuelve un

problema general de diseño dentro de un contexto particular dividiéndose grupos: patrones GRASP y GoF.

Patrones GRASP:

Son patrones generales de software para asignar responsabilidades a objetos.

Los utilizados son:

Experto: este patrón consiste en asignar una responsabilidad al experto en información, es decir, a la clase que cuenta con la información necesaria para cumplir con la responsabilidad. Lo que permite reforzar el encapsulamiento y favorecer el bajo acoplamiento (Larman, 1999). Un ejemplo de uso de este patrón es en la clase TareaController.php la cual contiene toda la información de las tareas referente a las tareas permitiendo crear una nueva y manipular sus entidades.

```
 */
class TareaController extends baseController
{

    /**
     * Lists all Tarea entities.
     *
    public function indexAction()
    {
        $em = $this->getDoctrine()->getManager();

        $entities = $em->getRepository('TecnicosBundle:Tarea')->findAll();

        return array(
            'entities' => $entities,
        );
    }
```

ILUSTRACIÓN 7 USO DEL PATRÓN EXPERTO. FUENTE: ELABORACIÓN PROPIA.

Creador: es el responsable de la creación o instanciación de nuevos objetos o clases (Larman, 1999). Un ejemplo del uso de este patrón se puede encontrar en la clase TareaController la cual permite crear una nueva tarea en el sistema.

```
class TareaController extends baseController
{
 public function createAction(Request $request)
 {
     $entity = new Tarea();
     $form = $this->createCreateForm($entity);
     $form->handleRequest($request);
```

ILUSTRACIÓN 8 USO DEL PATRÓN CREADO. FUENTE: ELABORACIÓN PROPIA.

Bajo acoplamiento: pretende asignar una responsabilidad para mantener el bajo acoplamiento, es decir, el diseño de clases más independientes que no se relacionen con muchas otras y reduzcan el impacto de los cambios, siendo más reutilizables y acrecienten la oportunidad de una mayor productividad (Larman, 1999).

Alta cohesión: su objetivo es asignar una responsabilidad, de modo que la cohesión siga siendo alta. Las clases con alta cohesión se caracterizan por tener responsabilidades estrechamente relacionadas y no realizar un trabajo enorme. Una clase con alta cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. La utilización de este patrón se evidencia en las responsabilidades asignadas a las clases de tal manera que cada una se encargará solamente de realizar las funciones que estén en correspondencia con la responsabilidad que posea (Larman, 1999).

Patrones GoF:

Los patrones GOF según su propósito se clasifican en tres grupos. Los creacionales tratan la creación de instancias. Los estructurales tratan la relación entre clases, la combinación clases y la formación de estructuras de mayor complejidad. Mientras que los de comportamiento tratan la interacción y cooperación entre clases (Pressman, 2010). Los patrones a aplicar serán:

Decorador (Decorator): este patrón permite modificar, retirar o agregar responsabilidades a un objeto dinámicamente, o sea, permite agregarles funcionalidades a los objetos durante su ejecución. Esto trae consigo que no sea necesario la creación de clases compleja con mucho código, pues podemos usar distintas combinaciones de decoraciones para generar distintos comportamientos o

resultados (Pressman, 2010). Un ejemplo de uso es en la plantilla dashboard.html.twig donde se encuentra el código html y las referencias de los archivos ccs y javascript.

```
<html lang="en" class="no-js">
    <head>
        <meta charset="utf-8"/>
        <title> sctmb | {% block title %}{% endblock %}</title>
        <link href="{{ asset('bundles/dashboard/css/style-metronic.css') }}" rel="stylesheet" type="text/css"/>
        <link href="{{ asset('bundles/dashboard/css/style.css') }}" rel="stylesheet" type="text/css"/>
        <link href="{{ asset('bundles/dashboard/css/style-responsive.css') }}" rel="stylesheet" type="text/css"/>
        <link href="{{ asset('bundles/dashboard/css/plugins.css') }}" rel="stylesheet" type="text/css"/>
        <link href="{{ asset('bundles/dashboard/css/themes/default.css') }}" rel="stylesheet" type="text/css" id="style_color"/>
        <link href="{{ asset('bundles/dashboard/css/custom.css') }}" rel="stylesheet" type="text/css"/>
        {% block stylesheets %}{% endblock %}
        <link rel="icon" type="image/x-icon" href="{{ asset('bundles/dashboard/img/ceraf_ico.ico') }}" />
    </head>
```

ILUSTRACIÓN 9 USO DEL PATRÓN DECORADOR. FUENTE: ELABORACIÓN PROPIA.

Inyección de dependencia (Dependency injection): es un patrón que sugiere "inyectar" objetos a una clase, en vez de que la clase cree el objeto por sí solo. Esto permite que la clase que utilice el objeto no requiera especificar como crearlo, sino que otra clase se ocupará en crear este objeto según sus especificaciones y simplemente será agregado en la clase en cuestión (Pressman, 2010). Un ejemplo de uso de este patrón es en la clase UsuarioController.php. la cual usa un objeto role el cual no construye.

```
class Usuario extends BaseUser
{
    private $roles;
    public function addRole(\Comun\UsuarioBundle\Entity\Role $roles)
    {
        $this->roles[] = $roles;

        return $this;
    }
}
```

ILUSTRACIÓN 10 USO DEL PATRÓN INYECCIÓN DE DEPENDENCIA. FUENTE: ELABORACIÓN PROPIA.

El modelo de datos permite estructurar la base de datos, en cuanto a los tipos de datos presentes y la forma en que se relacionan entre sí. A continuación, se presenta el modelo de datos realizado para la presente investigación.

Modelo de base de datos

Se realizó el diseño de la base de datos mediante la utilización de la herramienta case Visual Paradin es su versión 8.0 la misma muestra las tablas que son utilizadas en la solución propuesta para el almacenamiento de la información, las cuales están normalizadas, cumpliendo con las normas establecidas para el diseño de bases de datos. La base de datos se encuentra en tercera forma normal pues sus atributos dependen directamente de la clave primaria, o sea no se relacionan a través de otros atributos, eliminando de esta manera la dependencia transitiva.

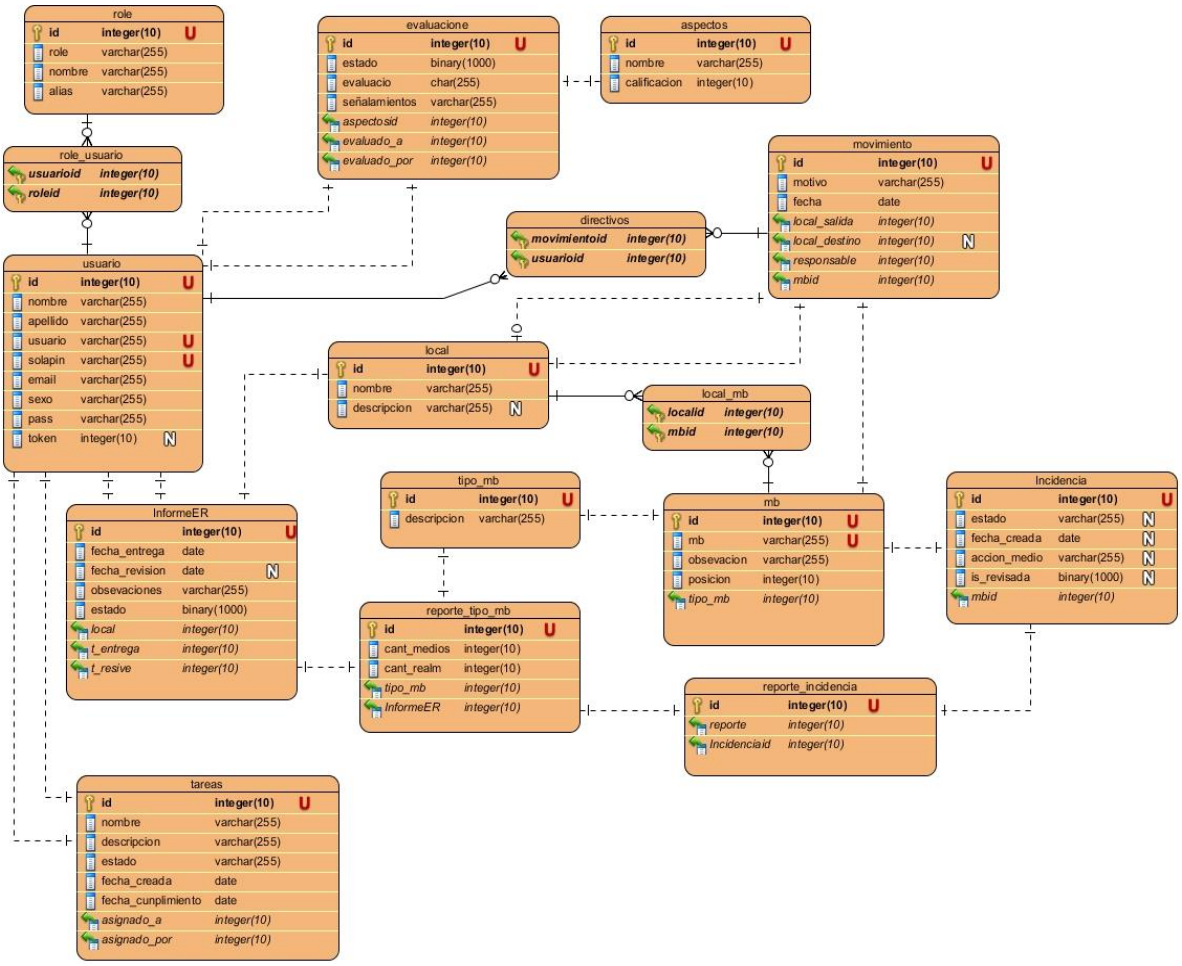

ILUSTRACIÓN 11 MODELO DE BASE DE DATOS. FUENTE: ELABORACIÓN PROPIA.

Descripción de las tablas de datos

mb: contiene la información del medio básico registrado permitiendo conocer el local al cual está asignado, en la posición que se encuentra y el estado del mismo.

usuario: contiene la información de los usuarios que tiene acceso a la aplicación los datos del mismo son obtenidos mediante el consumo de servicios de Ldap de la universidad.

local: contiene la información referente a los distintos locales con que cuenta el centro.

InformeER: contiene la información de los informes de entrega y recibo diarios de los locales, permitiendo conocer en qué fecha fue creado, el personal que realizó la entrega.

reporte_tipo_mb: contiene el reporte de medio según el tipo, con el que cuenta un InformeER de un local, permite conocer la cantidad de medios reales del local y la cantidad que se está entregando.

tipo_af: contiene la información referente de todos los tipos de medios con que cuenta el centro.

Incidencia: Contiene la información de todos los medios referentes a los reportes realizados por un informe de entrega-recibo, permitiendo conocer la fecha en que se generó la incidencia, el estado de los medios reportados o la acción que se realizó sobre el medio (extraído o en el local), además permite conocer el estado de la incidencia (resulta o no).

role: registra los roles que posee cada usuario registrado.

tarea: permite a los directivos asignar y controlar las tareas a los técnicos registrándose en la misma la fecha en la que fue asignada, el técnico al que se le asigna, así como su fecha de cumplimiento y evolución de la misma.

movimientos: permite llevar un registro de todos los movimientos de medios realizados por los directivos, registrando automáticamente los traslados permanentes de un local a otro de los medios, así como, cuando se les da de baja

a algún medio, el sistema registra el movimiento especificando su motivo (si es traslado o baja).

evaluaciones: contiene toda la información de las evaluaciones de los técnicos.

aspectos: contiene los aspectos tenidos en cuenta a la hora de adicionar una evaluación.

La aplicación de métricas para validar tanto el diseño como la propuesta de solución es fundamental para elevar la calidad de la misma y cumplir con las aspiraciones del cliente. A continuación, se definen las métricas utilizadas para evaluar el diseño propuesto.

Validación del diseño

Una métrica es un instrumento que permite evaluar el software al inicio del proceso, que cuantifica además un criterio y persigue comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado a nivel de proyecto. Con el fin de desarrollar un diseño robusto y sencillo, se realizó la validación del mismo utilizando las métricas Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC), ya que estas miden factores como: reutilización, encapsulamiento, complejidad y acoplamiento, siendo estos, elementos significativos en la programación orientada a objetos (POO) (Pressman, 2010).

TOC: las métricas orientadas a tamaño para una clase orientada a objetos (OO), está dado por el número de métodos asignados a una clase (tanto métodos heredados como métodos internos de dicha clase). Para esta métrica es importante que la reutilización sea inversamente proporcional a la complejidad de implementación y la responsabilidad. A mayor reutilización habrá menor complejidad y responsabilidad.

Se determinaron un total de 21 clases, con un promedio de procedimientos de 8,2 como se observa en los resultados siguientes.

Responsabilidad		
Atributo de calidad	Cantidad de clases	Promedio
Baja	11	55
Media	4	20
Alta	6	30
Complejidad		
Baja	11	55
Media	4	20
Alta	6	30
Reutilización		
Alta	17	85
Media	4	20
Baja	2	10

TABLA 6 CANTIDAD DE CLASES SEGÚN LOS ATRIBUTOS RESPONSABILIDAD, COMPLEJIDAD Y REUTILIZACIÓN. FUENTE: ELABORACIÓN PROPIA.

A continuación, se describen los atributos que posibilitan ejecutar la métrica TOC y los resultados obtenidos.

Atributo	Descripción
Responsabilidad	Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.
Complejidad	Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.
Reutilización	Grado de dificultad que tiene implementar un diseño de clases determinado.

TABLA 7 ATRIBUTOS DE LA MÉTRICA TOC. FUENTES: ELABORACIÓN PROPIA.

Para la evaluación de las clases fueron utilizados los umbrales para medir la responsabilidad, la complejidad de implementación y la reutilización que propone la métrica TOC.

	Categoria	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad imple	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

ILUSTRACIÓN 12 UMBRELA UTILIZADA POR LA MÉTRICA TOC. FUENTE: ELABORACIÓN PROPIA.

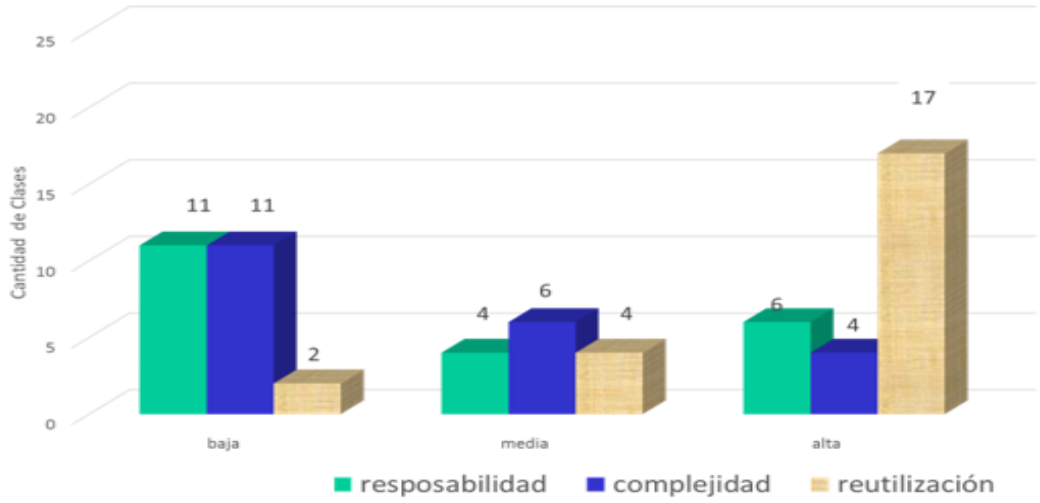

ILUSTRACIÓN 13 COMPORTAMIENTO DE LOS ATRIBUTOS DE CALIDAD DE LA MÉTRICA TOC. FUENTE: ELABORACIÓN PROPIA.

Luego de aplicar la métrica TOC se puede concluir que la propuesta de diseño cumple con los requisitos de calidad propuestos, teniendo como resultado que las clases contienen un porciento bajo de responsabilidad y complejidad, mientras que la reutilización es alta.

Relaciones entre clase (RC): está dado por el número de relaciones de uso de una clase con otra. La métrica evalúa el acoplamiento, la complejidad de mantenimiento, reutilización y la cantidad de pruebas que permiten medir la calidad de una clase. El procedimiento comienza al calcular el promedio de asociaciones de uso, siendo en este caso 1.05 y con esto se determina las categorías a la que corresponde cada

atributo de calidad. A continuación, se muestra el resultado de aplicar las métricas RC al diseño del sistema.

Acoplamiento		
Atributo de calidad	Cantidad de clases	Promedio
Ninguno	0	0
Bajo	21	105
Media	0	0
Alto	0	0
Complejidad de Mantenimiento		
Baja	20	100
Media	1	5
Alta	0	0
Cantidad de Pruebas		
Baja	20	100
Media	1	5
Alta	0	0
Reutilización		
Baja	0	0
Media	1	5
Alta	20	100

TABLA 8 CANTIDAD DE CLASES SEGÚN LOS ATRIBUTOS DE CALIDAD: ACOPLAMIENTO, COMPLEJIDAD DE MANTENIMIENTO, REUTILIZACIÓN Y CANTIDAD DE PRUEBAS PROPUESTA POR LA MÉTRICA RC. FUENTE: ELABORACIÓN PROPIA.

A continuación, se describen los atributos de calidad que posibilitan ejecutar la métrica y los resultados obtenidos al aplicar la misma.

Atributo	Descripción
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 9 MODO EN QUE SE AFECTAN LOS ATRIBUTOS DE CALIDAD. FUENTE: ELABORACIÓN PROPIA.

Para la evaluación de las clases fueron utilizados umbrales para el acoplamiento, complejidad de mantenimiento, la reutilización y cantidad de pruebas que propone la métrica RC.

	Categoria	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

	Categoria	Criterio
Complejidad Mant.	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

	Categoria	Criterio
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.

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

TABLA 10 UMBRELA UTILIZADAS POR LA MÉTRICA RC. FUENTE: ELABORACIÓN PROPIA.

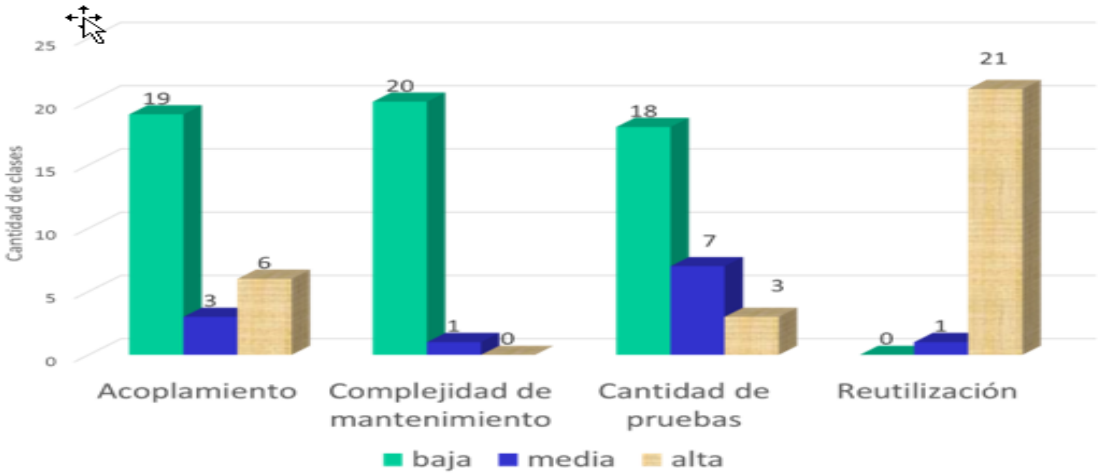

TABLA 11 COMPORTAMIENTO DE LOS ATRIBUTOS DE CALIDAD DE LA MÉTRICA RC. FUENTE: ELABORACIÓN PROPIA.

Los resultados después de aplicar la métrica de diseño RC y obtenidos los resultados de la evaluación del instrumento de medición de la métrica, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que aproximadamente el 95.68% de las clases empleadas poseen menos de tres dependencias de otras clases. Esto lleva a evaluaciones positivas de los atributos de calidad involucrados (bajo acoplamiento con 54.14%, baja complejidad de mantenimiento con un 44.85%, baja cantidad de pruebas con 67.85% y alta reutilización con un 96.85%). Favoreciendo de esta manera la reutilización de las clases, la baja complejidad de las mismas, así como la modificación e implantación del diseño propuesto.

Conclusiones del capítulo

- Se identificaron un total de 45 requisitos funcionales para identificar las principales funcionalidades con que contará la propuesta de solución.
- Se identificaron un total de 18 requisitos no funcionales los cuales proporcionaron a la solución atributos de calidad entre los que se encuentran usabilidad, confiabilidad, mantenibilidad entre otro.
- Se utilizaron los patrones de diseño experto, creador, bajo acoplamiento, alta cohesión y de arquitectura modelo-vista-controlador brindándole a la solución propuesta un mayor entendimiento y organización.

- A partir del uso de las técnicas de validación de requisitos; prototipo de interfaz y revisiones técnicas formales se garantizó que los requisitos estuvieran acordes con las necesidades y especificaciones del cliente.
- Con la aplicación de las métricas TOC y RC se constató que las clases del diseño presentan un bajo por ciento de responsabilidad, complejidad, acoplamiento, complejidad de mantenimiento y cantidad de pruebas mientras que su reutilización es alta.

Capítulo 3. Implementación y validación de la propuesta de Solución

Introducción

En el presente capítulo se describe el modelo de implementación de la solución propuesta y el estándar de codificación utilizado, se muestran los resultados de la aplicación de pruebas que aseguran la calidad del sistema, siguiendo como estrategia la realización de las pruebas propuestas en las disciplinas de pruebas internas y prueba de aceptación presentes en la metodología utilizada.

Para una mejor comprensión del proceso de desarrollo del sistema son descritos los diagramas propuestos por la metodología para la fase de implementación.

3.1 Implementación

En esta fase a partir de los resultados obtenidos del análisis y diseño se construye el sistema. El objetivo de la misma, es realizar las actividades necesarias para poner a disposición de los usuarios finales, el sistema desarrollado.

Seguidamente se describen los componentes en los cuales está estructurado el sistema desarrollado.

Diagrama de Componentes

El diagrama de componentes proporciona una visión física de la implementación del software. Muestra la organización de los componentes y sus relaciones. La siguiente imagen representa los componentes del sistema siguiendo la arquitectura Modelo-Vista-Controlador y su relación con el resto de los componentes utilizados. Cada capa es representada como un paquete, que abarca todos sus componentes internos y se muestra su relación con los servicios externos.

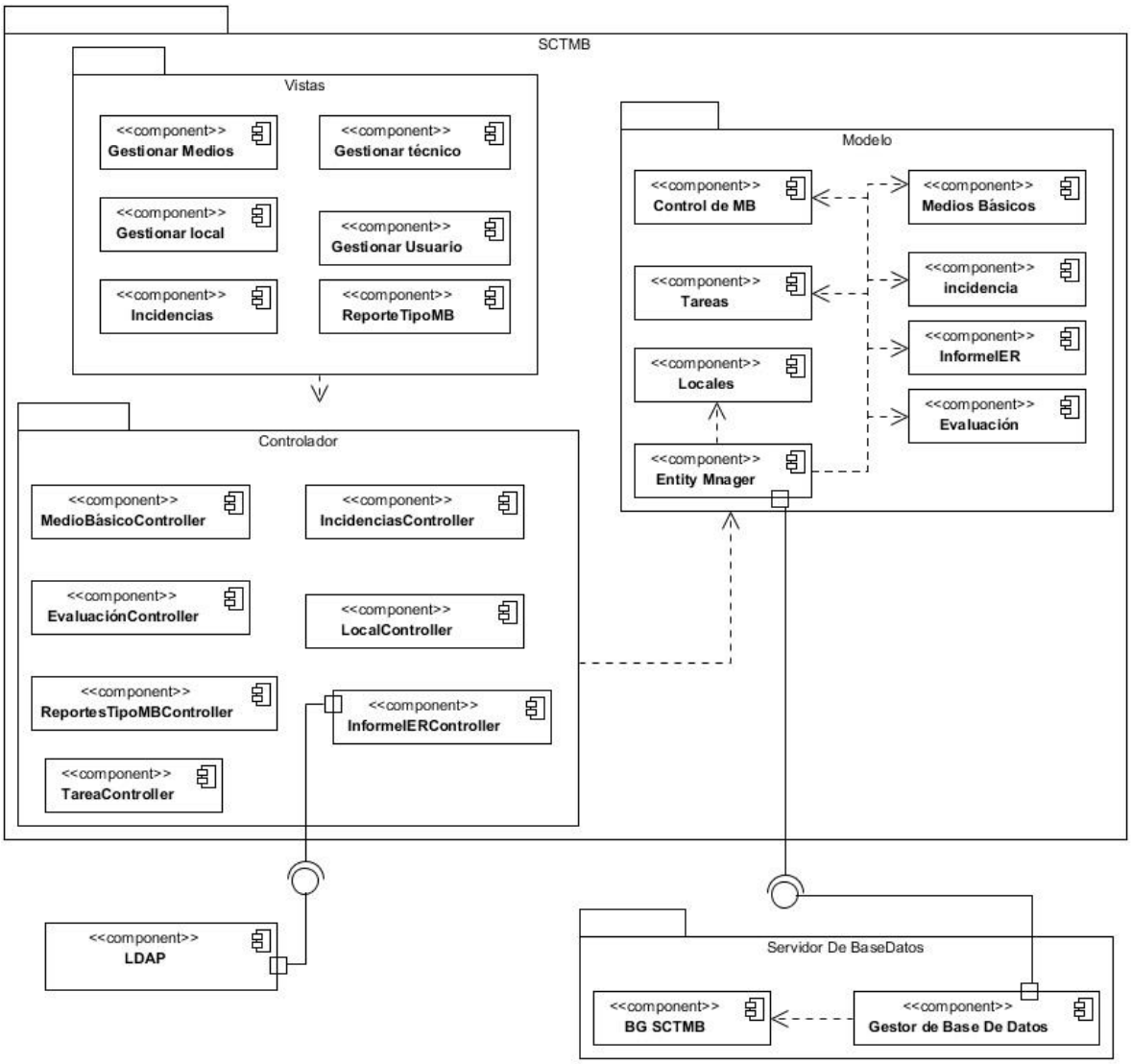

ILUSTRACIÓN 14 DIAGRAMA DE COMPONENTES. FUENTE: ELABORACIÓN PROPIA.

Diagrama de Despliegue

El diagrama de despliegue modela el hardware utilizado en la implementación del sistema y las relaciones físicas entre los componentes de hardware y software. Seguidamente se muestra el diagrama de despliegue (Quisbert Limachi, 2014).

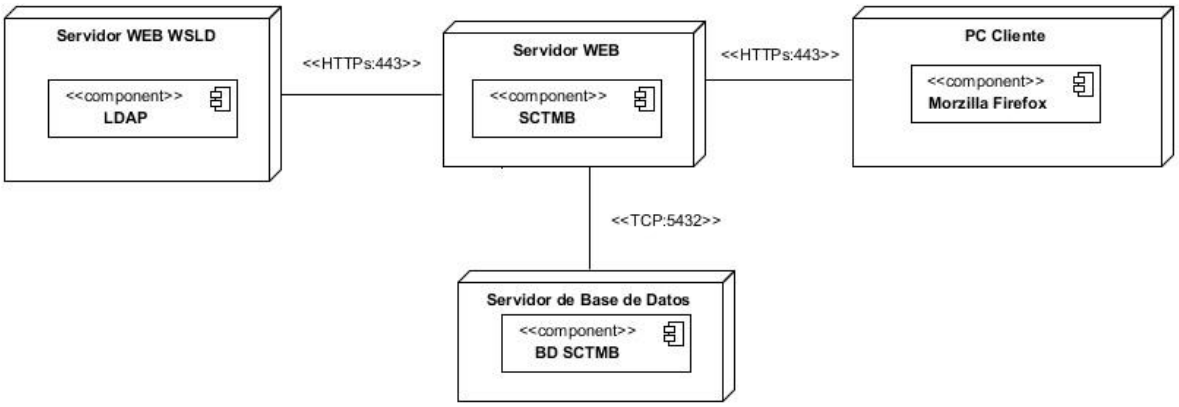

ILUSTRACIÓN 15 DIAGRAMA DE DESPLIEGUE. FUENTE: ELABORACIÓN PROPIA.

Descripción de los nodos físicos:

- PC Cliente: el nodo representa una PC cliente desde la que se podrá acceder al sistema por medio de un navegador web e interactuar con todas las funcionalidades que este brinda.
- Servidor Web: el nodo representa el servidor web donde está alojado el sistema.
- Servidor Web WSDL (Web Services Description Languaje.): el nodo representa el servidor de servicios web que ofrece la universidad. Entre los que ofrece se utiliza el servicio de Protocolo Ligero de Acceso a Directorios LDAP para la autenticación de los usuarios.
- Servidor de Base de Datos: el nodo representa el servidor de Base de Datos PostgresSQL en el que está alojada la base de datos.

Descripción de los protocolos utilizados:

HTTPS: protocolo para Transferencia de Hipertexto Seguro. Extensión del HTTP para la autenticación y encriptación de datos entre un servidor web y un navegador web.

TCP/IP: arquitectura definida por los protocolos, TCP (Protocolo de Control de Transmisión) y Protocolo de Internet (IP) (Transmission Control Protocol y Internet Protocol). Forma de comunicación básica que usa el Internet.

Para una mejor comprensión del código generado, se hace uso del estándar de codificación de Symfony, logrando de esta manera tener una nomenclatura en común para todas las clases y métodos.

Estándar de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Su uso de permite lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenibilidad a lo largo del tiempo (Pérez Alfonso, 2012).

Nomenclatura: los nombres de las clases comienzan con la primera letra en mayúscula y el resto con minúscula, en caso de ser una palabra compuesta, iniciará cada palabra con letra mayúscula.

Ejemplo: TareaController.php, EvaluacionController.php

Los nombres de los métodos y los atributos de las clases, comienzan con la primera letra en minúscula, en caso de que sea un nombre compuesto iniciarán cada palabra con letra mayúscula excepto la primera palabra. Adicionalmente le es agregado el sufijo Action propuesto por el marco de trabajo para identificar los métodos del controlador que responden a peticiones HTTPs.

Ejemplo: createAction, editarAction, actualizarAction.

Nomenclatura según el tipo de clase: las clases que se encuentran dentro del paquete Controller se nombran adicionándoles el nombre del controlador de Symfony2, del cual heredan al final del nombre de la clase (TareaController). El marco de trabajo se basa en esta técnica para identificar sus clases controladoras.

Ejemplo: MedioBasicoController.php

Las clases que se encuentran dentro del paquete Form se nombran adicionándoles como sufijo del nombre la palabra Type. El marco de trabajo propone esta técnica para un mejor entendimiento de sus clases.

Ejemplo: TaeraType, EvaluacionType

Normas de comentarios

Se debe comentar todo lo que se haga dentro del desarrollo, logrando establecer un código con todas las funciones descritas y así se pueda aumentar su mantenibilidad a lo largo del tiempo.

Comentarios de clases: antes de declarar una clase se brinda una descripción de esta, donde se explica el propósito de la misma y se escribe de la siguiente manera.

```
/**
 * Lists all Evaluacion entities.
 *controla todas las funcionalidades de las evalueaciones
 * actua como medioador entre los dato que gestiona la vista y el modelo
 * @Route("/", name="Evaluacion")
```

ILUSTRACIÓN 16 NOMENCLATURAS DE COMENTARIOS EN LAS CLASES DEL SISTEMA. FUENTE: ELABORACIÓN PROPIA.

Comentario en las funciones: los comentarios redactados al inicio de las funcionalidades describen el objetivo de la misma, así como los parámetros con que cuenta y el tipo de resultado que arroja. A continuación, se muestra un ejemplo.

```
 *
 * @Route("/", name="Evaluacion_create")
 * @Method("POST,GET")
 * @Template("TecnicosBundle:Evaluacion:new.html.twig")
 */
// crea un formulario para crear una evaluación en el sistema
```

ILUSTRACIÓN 17 NOMENCLATURA DE COMENTARIOS EN LAS FUNCIONES DEL SISTEMA. FUENTE: ELABORACIÓN PROPIA.

Tipos de mensajes utilizados:

Se definieron 3 tipos de mensajes para un mejor entendimiento entre el usuario y las acciones que este realiza al interactuar con el sistema. Estos mensajes son:

- Mensajes de error
- Mensajes de confirmación
- Mensajes de información

Mensajes de error: muestra al usuario que ha realizado una operación incorrecta, ejemplo:

- El usuario ya existe

- La suma de los peso de los componentes debe ser igual a 1
- Error en la operación realizada

Mensajes de confirmación: se utiliza para preguntarle al usuario si está seguro que desea realizar una acción ejemplo:

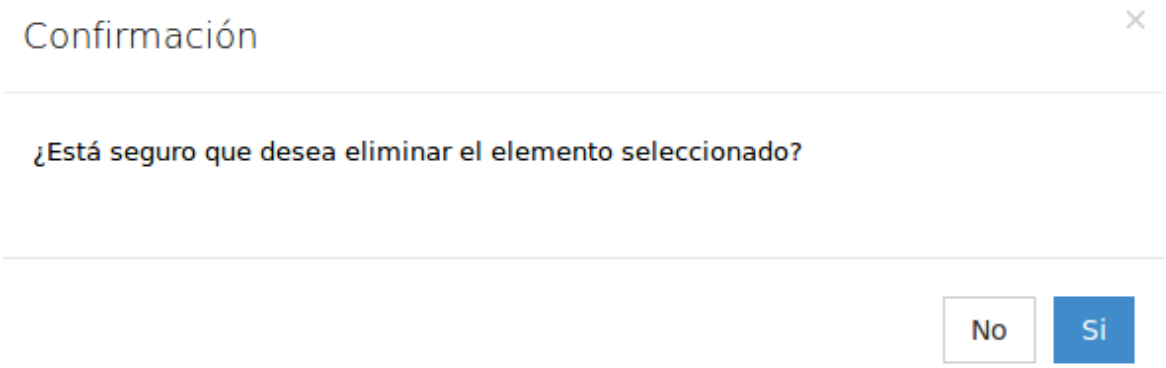

ILUSTRACIÓN 18 MENSAJES DE CONFIRMACIÓN DEL SISTEMA. FUENTE: ELABORACIÓN PROPIA.

Mensajes de información: se utiliza para brindarle información al usuario al realizar una operación correctamente ya sea de inserción, modificación o eliminación. Ejemplo:

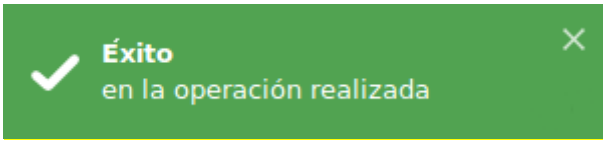

ILUSTRACIÓN 19 MENSAJES DE INFORMACIÓN DEL SISTEMA. FUENTE: ELABORACIÓN PROPIA.

Durante el proceso de desarrollo de software es necesario llevar un control estricto de las posibles salidas o resultados que arrojará el sistema, para ello se hace uso de un conjunto de pruebas. A continuación, se describen las mismas.

3.2 Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Para encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles. (ITM, 2015)

3.2.1 Pruebas de caja blanca

Las pruebas de caja blanca son una filosofía de diseño de caso de prueba que usan la estructura de control del diseño a nivel de componete para derivar los casos de

pruebas. Mediante estos métodos, el ingeniro de software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas; ejecuten todos los bucles en sus límites y con sus limitaciones operacionales; y ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2010).

La técnica a aplicar es la de camino básico, la cual permite al diseñador de caso de prueba obtener una medida de complejidad logica de un diseño de pricedimiento o usar esta como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Pressman, 2010).

Para obtener el conjunto de caminos independientes se construye el grafo de flujo asociado a una función y se calcula su complejidad, para este caso se tomó como ejemplo el método CrearTareaAction(), encargado de crear una nueva tarea en el sistema, mostrado a continuación. Primeramente, se enumeran las sentencias de código.

```
public function createAction(Request $request)
{
    $entity = new Tarea();
    $form = $this->createCreateForm($entity);   1
    $form->handleRequest($request);  2

    if ($form->isValid()) {   3
        $em = $this->getDoctrine()->getManager();      4
        $asignadoPor = $em->getRepository('UsuarioBundle:Usuario')->find(32);   5
        $entity->setEstado(1);
        $entity->setAsignadoPor($asignadoPor);
        $entity->setFechaCreacion(new \DateTime('now'));
        $entity->setEvaluacion("N/E");

        if ($form->get('finalizar')->isClicked()) { 6
            $entity->setEstado(2);
        }
        $em->persist($entity);   7

        $em->flush();
        $this->Alert(self::Success, array('en la operación realizada'));  8

        return $this->redirect($this->generateUrl('tarea'));  9
    } else {
        $this->Alert(self::Error, array('en la operación realizada'));  10
        return $this->redirect($this->generateUrl('tarea'));11
    }

    return new JsonResponse(array(  12
```

ILUSTRACIÓN 20 CÓDIGO DEL MÉTODO CREARTAREAACTION(). FUENTES: ELABORACIÓN PROPIA.

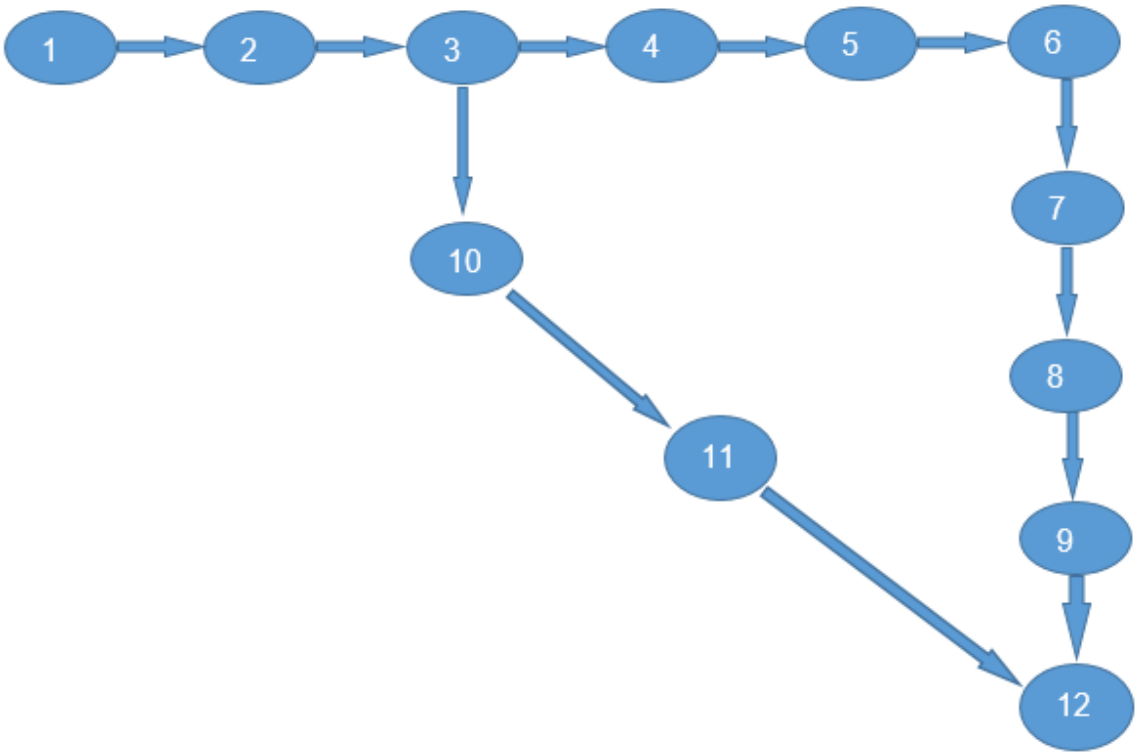

ILUSTRACIÓN 21 GRAFO DE FLUJO ASOCIADO AL MÉTODO CREARTAREAACTION(). FUENTE: ELABORACIÓN PROPIA.

La complejidad ciclomática es una medición de software con que se define la cantidad de caminos independientes del conjunto básico de un programa, brindando el número máximo de pruebas que se debe realizar para hacer que se ejecute cada sentencia al menos una vez (Pressman, 2010). La misma se calcula de la siguiente manera.

V(G)=E-N + 2 donde E es el número de aristas y N los nodos.

V(G)=12-12+2

V(G)=2

Para este caso se obtienen 2 posibles caminos independientes y cantidad de pruebas que se deben realizar para comprobar que las sentencias se ejecuten al menos una vez.

- Camino 1:1, 2, 3, 10, 11, 12.
- Camino 2: 1, 2, 3, 4, 5, 6, 7, 8, 9, 12.

Luego se elaboran los casos de prueba, quedando de la siguiente manera.

- Caso de prueba camino 1.

  Condición: si no se cumple N3

  Resultado esperado: el sistema envía un mensaje informando que no se puede adicionar la tarea.

- Caso de prueba camino 2.

  Condición: si se cumple N3

  Resultado esperado: el sistema adiciona la tarea correctamente.

Luego se ejecutan los casos de prueba para comparar los resultados obtenidos con los esperados, una vez comprobado que estos coinciden, se puede asegurar que todas las sentencias del método se han ejecutado al menos una vez.

3.2.2 Pruebas de caja negra

Las pruebas de caja negra o pruebas funcionales se realizan sobre la interfaz del software, comprobando las entradas y salidas de datos. Estas pruebas se realizan para comprobar que cada función es operativa, utilizando el artefacto diseño de caso de prueba, que tienen como objetivo introducir juegos de datos que ayuden a la ejecución de los casos y facilite que el sistema se ejecute en todas sus variantes (ITM, 2015). Con estas pruebas se intentaron encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a datos.
- Errores ortográficos y no conformidades.

La técnica utilizada para desarrollar las pruebas de Caja Negra fue partición equivalente. Es un método que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La misma está dirigida a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (Pressman, 2010). A continuación, se muestra un ejemplo de los DCP realizados. Ver el resto de los casos de prueba en el (anexo 3).

Caso de prueba: adicionar tipo de medio básico.

- Se debe autenticar en el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar la opción crear tipo de medio básico.

Caso de prueba: Adicionar tipo de medio básico.

Escenario	**Descripción**	**Nombre**	**Respuesta del sistema**	**Flujo central**
CP 1.1 Adicionar tipo medio básico Introduciendo datos válidos.	Se crea un nuevo tipo medio básico en el sistema.	V silla	El sistema adiciona el tipo de medio básico y muestra el mensaje de información: "Éxito en la operación realizada ".	-Se selecciona la opción Adicionar del menú tipo de medio básico. -El sistema muestra una ventana con los datos del tipo de medio básico. -El usuario introduce los datos. -El usuario selecciona el botón Adicionar. -El sistema muestra el mensaje de información: "Éxito en la operación realizada "
CP 1.1 Adicionar tipo medio básico Introduciendo datos inválidos.	Se crea un nuevo tipo medio básico en el sistema.	I vacío	El sistema muestra un mensaje de error debajo del campo: "Este capo es obligatorio ".	-Se selecciona la opción Adicionar del menú tipo de medio básico. -El sistema muestra una ventana con los datos del tipo de medio básico. -El usuario introduce los datos. -El usuario selecciona el botón Adicionar.

				-El sistema muestra el mensaje de error: “Este capo es obligatorio” y no se adicionan los datos al sistema.

TABLA 12 CASO DE PRUEBA: ADICIONAR TIPO DE MEDIO. FUENTE: ELABORACIÓN PROPIA.

En la primera iteración se probaron un total de 10 casos de pruebas de los cuales 6 arrojaron resultados satisfactorios para un 60% de efectividad, identificándose no conformidades tales como errores ortográficos y funciones incorrectas. Para la segunda iteración se realizaron correcciones a las 4 no conformidades pendientes de la iteración anterior y se probaron un total de 12 casos de prueba de los cuales 11 resultaron satisfactorios para un 91,6% de efectividad. Por último, se corrigieron los problemas pendientes en la iteración anterior y se repitieron nuevamente todas las pruebas resultando satisfactorios en un 100%. A continuación, se muestra el resultado de las pruebas funcionales para cada iteración.

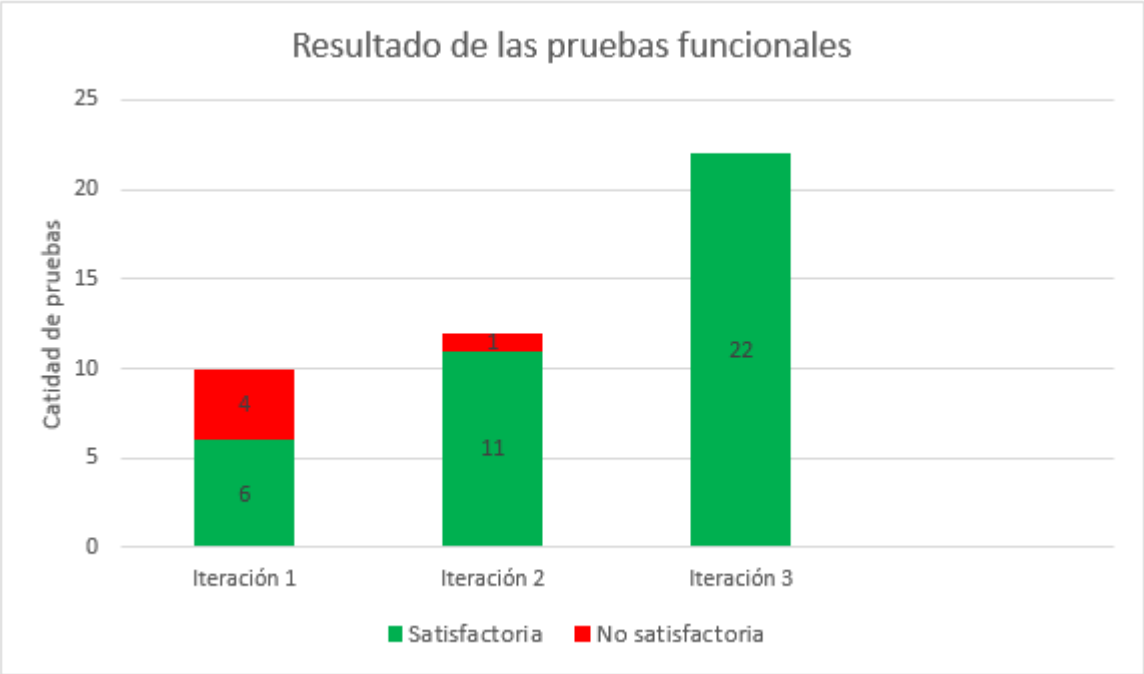

ILUSTRACIÓN 22 RESULTADO DE LAS PRUEBAS FUNCIONALES. FUENTE: ELABORACIÓN PROPIA.

3.3 Pruebas de aceptación del cliente

Para realizar las pruebas de aceptación con el cliente, se utilizó el mismo principio de las pruebas funcionales, permitiendo que sea el propio cliente el que realice dichas pruebas, utilizando juegos de datos en cada una de las interfaces seleccionadas. Se realizaron 3 iteraciones quedando reflejada el resultado de las mismas en la Ilustración 18. En la primera iteración, se realizaron pruebas a los módulos de entrega recibo de laboratorios, asignación de tareas y evaluaciones siendo identificados 9 no conformidades. En la segunda iteración se probaron todos los módulos de la primera sumándole el módulo de seguridad siendo identificados 4 no conformidades. En la última iteración se probó el sistema en su totalidad incluyendo los niveles de acceso, demostrando que el sistema creado se encuentra apto para ser usado.

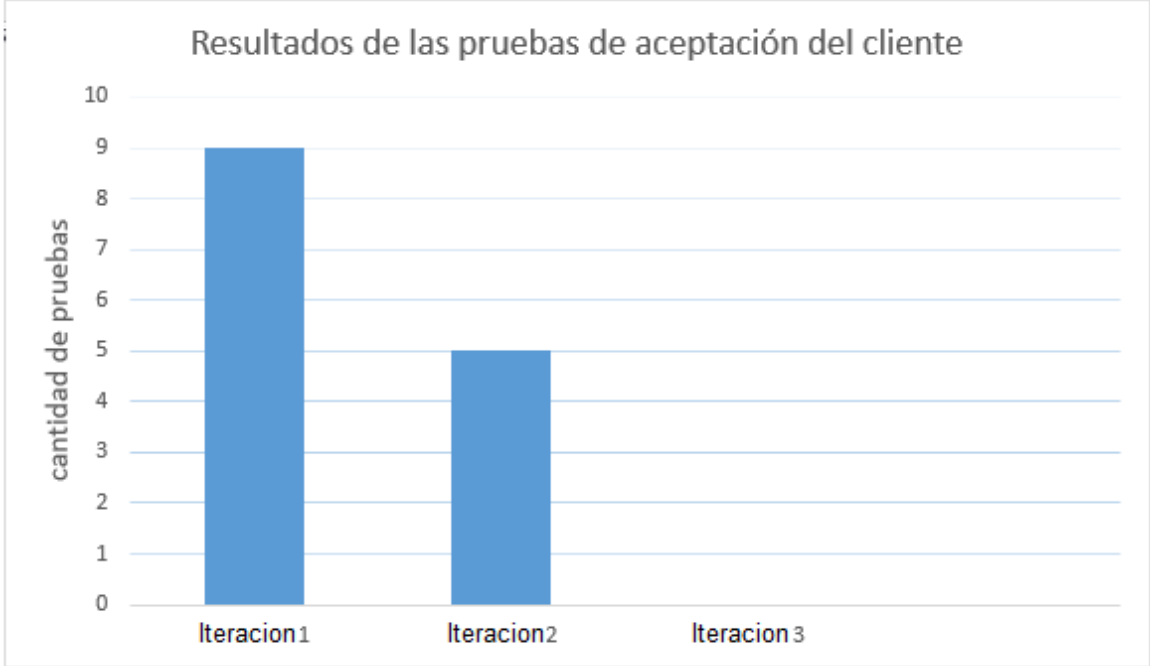

ILUSTRACIÓN 23 CANTIDAD DE NO CONFORMIDADES EN ENCONTRADAS POR ITERACIÓN. FUENTE: ELABORACIÓN PROPIA.

3.4 Resultados de la validación mediante criterios de expertos: Método Delphi.

Para validar la investigación se tuvo en cuenta los aspectos recogidos en la resolución 60/2011 sobre las normas del sistema de control interno. Esta norma

plantea que para evaluar el control se deben considerar cinco componentes, estos son: Ambiente de Control, Gestión y Prevención de Riesgos, Actividades de Control, Información y Comunicación y Supervisión y Monitoreo.

A partir del análisis de la población objeto de aplicación de la encuesta, se decidió aplicar un muestreo a conveniencia. Consiste en una técnica de muestreo no probabilístico donde los sujetos son seleccionados, dada la conveniente accesibilidad y proximidad de los sujetos para el investigador. Además, se consideró que las personas a encuestar no son expertos en el tema en cuestión, pero sí tienen un alto conocimiento empírico de los procesos relacionados con los TCI.

En correspondencia con estos elementos se diseñó un cuestionario orientado a obtener información sobre la valoración de los expertos, acerca de cómo la propuesta de solución implementada impacta en los cinco componentes (ver anexo 4). Se propone un cuestionario de un total de 11 preguntas de respuesta cerrada, facilitándose de esta forma el procesamiento y la cuantificación. La vía utilizada para la aplicación de la encuesta fue el cuestionario impreso.

Las respuestas a las preguntas están representadas en una escala de Likert. Es una escala psicométrica comúnmente utilizada en cuestionarios y se considera la escala de uso más amplio en encuestas para la investigación, principalmente en ciencias sociales. En cada pregunta del cuestionario se responde con el nivel de acuerdo o desacuerdo con una declaración, elemento o pregunta (ver anexo 4).

Para el procesamiento de la información obtenida en las encuestas se hizo corresponder un valor numérico a cada posible resultado de la escala utilizada. Los valores utilizados se muestran en la siguiente tabla:

Respuesta	Valor numérico
nada	1
muy poco	2
poco	3
bastante	4
mucho	5

Procedimiento para determinar el grado de consenso entre los expertos:

A partir de la tabla 13 se puede afirmar que existe un alto grado de concordancia en el criterio de los expertos en todas las preguntas realizadas. En todos los casos el coeficiente de variación muestra valores por debajo de 0.12 lo que sustenta la afirmación anterior. De este modo se puede concluir que, de acuerdo al consenso en el juicio de los expertos, la solución propuesta contribuye a elevar el control sobre los procesos desarrollados por los TCI. Destacan las preguntas 3 y 8 con un coeficiente de variación por debajo de 0.10.

Expertos	Preguntas										
	1	2	3	4	5	6	7	8	9	10	11
1	4	5	4	4	5	4	4	4	4	5	4
2	4	4	4	4	5	5	5	4	5	5	4
3	5	5	4	4	4	5	4	4	5	4	5
4	4	5	4	4	4	5	5	4	4	4	5
5	4	4	4	4	5	5	4	4	4	5	4
6	4	5	4	5	4	5	5	4	4	4	5
7	4	4	4	5	5	4	5	4	5	4	4
8	4	4	3	4	5	5	4	4	5	4	4
9	5	4	4	5	4	5	4	4	4	4	4
10	5	5	4	5	5	4	4	4	5	5	4
Cj media	4,30	4,50	3,90	4,40	4,60	4,70	4,40	4,00	4,50	4,40	4,30
Varianza	0,23	0,28	0,10	0,27	0,27	0,23	0,27	0,00	0,28	0,27	0,23
Coef. de Variación	0,11	0,12	0,08	0,12	0,11	0,10	0,12	0,00	0,12	0,12	0,11

TABLA 13 RESULTADO DE LA APLICACIÓN DEL MÉTODO DELPHI. FUENTE: ELABORACIÓN PROPIA.

El análisis de la tabla anterior permitió concluir lo siguiente por cada uno de las preguntas:

Preguntas	Valoración de los expertos
1. La solución contribuye a garantizar la integridad de los procesos.	Bastante
2. La solución contribuye a fortalecer el ambiente de control.	Mucho
3. La solución contribuye a identificar riegos e irregularidades de los procesos.	Bastante
4. La solución tributa a identificar objetivos de control.	Bastante
5. La solución contribuye a que los procesos se desarrollen de acuerdo a lo establecido.	Mucho
6. La solución contribuye al registro oportuno de las actividades.	Mucho
7. La solución apoya la toma de decisiones.	Bastante
8. El alto el nivel de seguridad de la información que se maneja en el sistema.	Bastante
9. La solución permite el monitoreo de los procesos.	Mucho

10. La solución contribuye a la detección de errores.	Bastante
11. La solución contribuye a elevar el control de los procesos.	Bastante

TABLA 14 EVALUACIÓN DE LAS PREGUNTAS DEL CUESTIONARIO UTILIZADO EN LA APLICACIÓN DEL MÉTODO DELPHI.

FUENTE: ELABORACIÓN PROPIA.

Conclusiones del capítulo

- Se implementó los requisitos funcionales en correspondencia con las descripciones realizadas obteniéndose una solución acorde a las especificaciones del cliente.
- La utilización de estándares de codificación en la implementación del sistema permitió estructurar y organizar el código de la propuesta de solución, logrando un lenguaje común y comprensible para todas las clases y métodos utilizados en el desarrollo del sistema.
- Al aplicar las pruebas de caja blanca mediante la técnica de camino básico permitió probar el código para corregir los errores detectados, obteniendo un código confiable que responde a los requisitos identificados.
- Se ejecutaron un total de 22 diseños de casos de pruebas, corrigiendo por iteraciones las no conformidades encontradas y repitiendo estas pruebas hasta lograr un estado del cien por ciento óptimo, logrando un alto nivel de calidad de la propuesta de solución.
- La validación de la investigación mediante el uso del método de criterio de experto arrojó que la propuesta de solución contribuye a elevar control sobre los procesos desarrollados por los TCI.

Conclusiones Generales

- Los procesos desarrollados por los TCI presentan insuficiencias que limitan un adecuado control de los mismos.
- El estudio realizado de los sistemas existentes relacionados con el tema de la presente investigación arrojó la necesidad de desarrollar una solución capaz de gestionar íntegramente los procesos desarrollados por los TCI.
- Se utilizaron los patrones de diseño; experto, creador, bajo acoplamiento, alta cohesión y de arquitectura modelo-vista-controlador brindándole a la solución propuesta un mayor entendimiento y organización.
- La utilización de estándares de codificación en la implementación del sistema permitió estructurar y organizar el código de la propuesta de solución, logrando un lenguaje común y comprensible para todas las clases y métodos utilizados en el desarrollo del sistema.
- A partir de la corrección de errores y no conformidades detectadas luego de realizar las pruebas de caja negra y caja blanca, se obtuvo un alto nivel de calidad en la propuesta de solución.
- La validación de la investigación mediante el uso del método de criterio de experto arrojó que la propuesta de solución contribuye a elevar control sobre los procesos desarrollados por los TCI.

Recomendaciones

- Hacer extensible el uso del módulo Técnicos de Laboratorio del sistema para la gestión de los procesos administrativos del centro de Informatización de Entidades para todos los centros de la universidad.
- Para una nueva versión del módulo, tener en cuenta los procesos no tratados en la presente investigación como son gestión del flujo de trabajo de los TCI y gestión de sus vacaciones.
- El sistema permita consumir la información de medios básicos del sistema ASSET.

Referencias Bibliográficas

Alcofin. (2012). from http://www.alcofin.com.mx/activos_fijos.html

Álvarez, M. (2006). Procesamiento y gestión digital de la información. *Revistas Científicas de América Latina, el Caribe, España y Portugal, 9*.

Augier, A. (2005). La gestión de la Información y el Conocimiento: desafíos de la Dirección Educacional Contemporánea.

Avila, K. (2015). ¿Qué es un Sistema Gestor de Bases de Datos o SGBD? , from http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/

Carrillo, I., Pérez, R., & Rodríguez, A. (2008). Metodología de desarrollo del software.

Cataldi, Z. (2000). *Metodología de diseño, desarrollo y evaluación de software educativo.*, Facultad de Informática. UNLP.

Cimo, F. (2015). *Bootstrap Programming Cookbook*.

De Luca, D., & De Luca, A. (2010). CC3 y HTML5. from http://html5.dwebapps.com/que-es-css3/

Eguiluz, J. (2015). Introducción a JavaScript. from http://librosweb.es/libro/javascript/capitulo_1.html

Fabien, R. (2011). Symfony 2.4, el libro oficial.

García Valdés, M., & Suérez Marín, M. (2013). El método Delphi para la consulta a expertos en la investigación científica. *Revista Cubana de Salud Pública*, 39(32) 253-267.

Geany. (2015). Geany.

ITM. (2015). Pruebas de Software. *Informática*.

Larman, C. (1999). *UML Y PATRONES*. Mexico: Prentice Hall.

López, P. (2014). Visual Paradigm for UML.

López Reyes, Y. (2017). Sistema de trabajo para los técnicos de laboratorio del centro CEIGE.

López, T. (2008). Selección de contenidos del Proyecto Estratégico de la UCI para el período 2008 – 2012. *Informática*.

Mar Cornelio, O. (2011). *Sistema de entrega de guardia para los laboratorios de la UCI.* Universidad de Ciencias Informáticas, La Habana.

Mateu, C. (2008). Desarrollo de aplicaciones web Retrieved from. from http://bibliotecadigital.org/jspui/handle/001/591

Méndez, G. (2009). *Ingeniería de Requisitos*. Universidad Complutense de Madrid.

Orichijuan. (2014). ¿Qué son los framework de desarrollo y por qué usarlos? . from http://www.actualidadgeek.org/2014/09/que-son-los-framework-de-desarrollo-y-por-que-usarlos/

Objetivos y Metas. *[Visitado el: 06 de noviembre de 2016.]* https://gespro.ceige.prod.uci.cu/

Pacheco, N. (2012). *Symfony2-es Release 2.0.15*. España.

Pantoja, Y. (2005). Introducción al Modelo Conceptual.

Pérez Alfonso, D. (2012). Normas y estándares de codificación. Universidad de las Ciencias aInformáticas.

Pressman, R. (2010). *Software Engineering A Practitioner's Approach* (7ma ed.). New York: McGraw-Hill.
Quisbert Limachi, N. (2014). ANALISIS Y DISEÑO DE SISTEMAS.
Rodríguez. (2015). *Asistente planificador para la asignación de actividades.* (ingeniero Ingeniero en Ciencias Informáticas), Universidad de Ciencias Informáticas La Habana.
Rodríguez, T. (2015). Metodología de desarrollo para la Actividad productiva de la UCI.
Rubio, F. J. G. (2006). Creación de sitios web con PHP5.
Saborit, A. M., & Alemán, J. J. (2015). *Sistema Informático para el control del proceso de entrega y recivo de activos fijos.* (ingeniero Tesis en opción al título de Ingeniero en Ciencias Informáticas), Universidad de las Cienias Informáticas, La Habana.
Sawyer, D. (2009). css.
Tinoco, O., Rosales, P., y Salas, J. (2010). Criterios de selección de metodologías de desarrollo de software.
Torres. (2015). La gestión de información y la gestión del conocimiento. *Revistas Científicas de América Latina, el Caribe, España y Portugal, 19*.
UCI. (2014). Modelo_del_Profesional. 37.
Vásquez, M. (2012). *Módulo de Juego Puzzle para el Generador de Aplicaciones J2ME para Dispositivos Móviles.* Universidad de las Ciencias Informáticas.
Win2PDF. (2015). Utilización del método Delphi en la pronosticación: una experiencia inicial.