Universidad de las Ciencias Informáticas

Facultad 3

Título:

Sistema de control de asistencia y permanencia de los trabajadores de CEGEL.

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

Autor:

Juan Enidio Martínez Costa

Tutor(a):

Ing. Yelena Hernández Estrada

Co-tutor:

Hernan Antonio Sanchez Guzmán

Ciudad de la Habana, Junio del 2014

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Juan Enidio Martínez Costa

_____
Firma del Autor

Tutor: Yelena Hernández Estrada **Co-Tutor: Hernán Sánchez Guzmán**

_____ _____

Firma del Tutor Firma del Co-Tutor

AGRADECIMIENTOS

A Dios por estar siempre conmigo y ser esa persona, ese amigo, ese hermano que siempre va a estar contigo y nunca te va a abandonar.

A mis padres por darme el regalo más grande que es la vida; en especial a mi mamá por estar siempre conmigo y aguantarme todas mis pesadeces y malacrianzas.

A mi familia, a mi bisabuela y abuela por parte de madre, que gracias a Dios me ha permitido contar todavía con su presencia; y en especial a mi tía Niurka pues a pesar de estar en tierras lejanas se ha mantenido al tanto de la evolución de carrera, a mi tía Josefina, por ser todo lo que ha hecho por mí para que este sueño de hoy día fuera posible.

A mis amigos del preuniversitario, por los buenos y malos momentos que compartimos juntos durante esos tres años de estancia, en especial a Pablito; pues sin él no estaría hoy aquí.

A mis amigos de la iglesia, en especial a mi mejor amiga Reglita, por los buenos y malos momentos que hemos paso juntos, por ser más que una amiga, una hermana para mí.

A todas las personas que han transitado por el grupo, en especial a Leordan, Luis Manuel por todas las cosas por las que pasamos para hacer de este sueño una

realidad; y a Dayana Francia, por demostrarme su amistad en el poco tiempo que compartimos.

A la dermatóloga del hospital-pediátrico Juan Manuel Márquez amiga de mi mamá que me atendió con mucho amor y cariño durante todo el tiempo que estuve enfermo y gracias a la cual puedo hacer verdad este sueño en este año y no en otro.

A mis amigos y personas del barrio que aunque no saben o saben poco sobre mi carrera me incitan día a día a ser mejor en la misma.

A todas las personas que he conocido a lo largo de estos cinco años y con las cuales he tenido el placer de compartir y de las cuales he aprendido cosas.

A mi tutora Yelena por ser como una madre para mí y por estar en todo momento pendiente de mí y de la tesis, dedicándome en varias ocasiones parte de su tiempo libre.

A todas las personas que de una forma u otra han contribuido y formado parte de mi vida y de este camino largo y amargo.

A todos muchas gracias.

DEDICATORIA

Dedico este trabajo a mis padres, en especial a mi madre por estar siempre conmigo apoyándome en los buenos y malos momentos que he pasado en la vida, y por desempeñar por tantos años ese rol de madre y padre. A mi hermana y hermano; y a toda mi familia por su preocupación y apoyo incondicional; y en especial a mi tía Josefina, por estar siempre pendiente desde el inicio hasta el fin de esta etapa de mi vida y a todas las personas que me han apoyado y me han dado fuerzas para seguir adelante.

RESUMEN

Mantener un control sobre los horarios de entrada y salida de los trabajadores en una institución es complejo, más aún cuando es una gran cantidad de personal. Esta situación ocurre en el Centro de Gobierno Electrónico (CEGEL), donde además está presente la diversidad de laboratorios, pues su ubicación no es la misma para todos ellos.

Actualmente en CEGEL se regula la llegada de los trabajadores mediante un registro de firma, el cual se encuentra en el centro de desarrollo de producción. Este es un proceso que tiene vulnerabilidades, pues el registro no es suficiente para controlar los horarios del personal que labora en el centro.

Dada la importancia de los proyectos que le asignan al centro, el personal de cada uno de ellos no puede ser ubicado en un laboratorio, sino que ocupan muchas veces más de uno. Esta situación trae consigo que la dirección de los proyectos no esté en contacto permanente con algunos de sus trabajadores y por tanto no puedan comprobar si las personas están trabajando en tiempo real.

Para dar una solución acorde con el problema, fue necesario realizar un estudio de algunas soluciones de software relacionadas con el control del personal, en las cuales se utilizan recursos que contribuyen a resolver escenarios similares a los planteados con anterioridad.

La puesta en marcha de este sistema permitirá dar los primeros pasos para resolver los problemas existentes, disponiéndose de un software que permitirá:

-Llevar el control del personal.

-Consultar datos del personal.

-Generar reportes del comportamiento del personal.

Palabras claves: asistencia, firma, control, personal.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

TABLA DE CONTENIDO

INTRODUCCIÓN

El sistema para el control de la asistencia del personal de una empresa, es uno de los ejes fundamentales para alcanzar un funcionamiento global eficaz. Gracias al mismo se optimizan procesos, los cuales implican la toma de decisiones por parte de directivos en vista a elevar el rendimiento, la calidad y la efectividad de la institución.

Actualmente en CEGEL (Centro de Gobierno Electrónico), la entrada de los trabajadores se establece manualmente mediante el Registro de firma. Esto trae consigo que existan vulnerabilidades en el control de entrada y salida de los empleados, pudiendo ocurrir que trabajadores le firmen a otros, que se registren horarios indebidos y en caso de tardanza del personal, que se desconozcan las razones, ya que el actual mecanismo de control de asistencia no permite reflejar estas incidencias.

Teniendo en cuenta la magnitud, el alcance y la importancia de sus proyectos, este centro tiene desplegado sus recursos humanos por diferentes proyectos de producción, aún perteneciendo al mismo. Esto provoca que el control de las personas, por parte de sus jefes y superiores sea insuficiente, debido a que no se puede saber si el trabajador está en su puesto de trabajo en el horario establecido.

Los trabajadores o profesores que trabajan en proyecto, también pueden tener actividades docentes. Esta doble vinculación está reflejada en el Plan de Trabajo que se realiza los primeros días de cada mes, donde cada persona refleja sus actividades diarias. Las tareas planificadas justifican al trabajador a no presentarse al proyecto en determinado momento, son aquellas, que por su inmediatez, el trabajador no conoce con tiempo suficiente, las que no quedan reflejadas en el Plan de Trabajo y es un periodo que está ausente injustificadamente.

A raíz de lo anterior surge el siguiente **problema a resolver:** el actual mecanismo de control de asistencia y permanencia de los trabajadores de CEGEL tiene vulnerabilidades, lo que no posibilita obtener datos confiables para el análisis y toma de decisiones por los directivos.

El **objeto de estudio** se enmarca en los procesos de gestión de asistencia laboral y se define como **campo de acción** el proceso de gestión de asistencia laboral del personal de CEGEL.

Para dar solución a la problemática planteada se establece como **objetivo general** desarrollar un sistema para el control de asistencia y permanencia del personal de CEGEL que facilite obtener datos que permitan su análisis y la toma de decisiones de los directivos.

Se plantea como **idea a defender:** desarrollar un sistema para el control de asistencia y permanencia del personal de CEGEL posibilitará obtener datos confiables para el análisis y toma de decisiones por los directivos.

Objetivos específicos:

1. Elaborar el marco teórico y el estado del arte de la investigación para identificar buenas prácticas y adoptar una posición al respecto.
2. Elaborar el análisis y diseño de la propuesta de solución.
3. Implementar las funcionalidades del sistema para dar solución a los requisitos definidos.
4. Validar los resultados obtenidos mediante la realización de pruebas de software.

Para dar cumplimiento al objetivo propuesto se plantean las siguientes **tareas de investigación:**

1. Análisis de los sistemas de control de asistencia y permanencia a nivel mundial y en Cuba.
2. Identificación de las herramientas que se utilizarán en el desarrollo del sistema de control de asistencia y permanencia.
3. Modelación del negocio de los procesos de control de asistencia y permanencia.
4. Especificación de los requisitos de la solución de la gestión del control de asistencia y permanencia.
5. Diseño de la solución de software para la gestión del sistema de control de asistencia y permanencia.
6. Implementación de la solución para la gestión del sistema de control de asistencia y permanencia.

7. Validación de los artefactos obtenidos durante la investigación.

La realización de dichas tareas se sustenta en los siguientes **métodos de investigación**:

Métodos teóricos:

Histórico-Lógico: se usa con el objetivo de estudiar los sistemas existentes relacionados con el control y permanencia de las personas en las diversas instituciones, así como las metodologías que rigen el proceso de desarrollo del software, lenguajes de modelado, herramientas de desarrollo y tecnologías a utilizar.

Análisis sintético: se utiliza para extraer e identificar conceptos, características y otros elementos de la bibliografía consultada que posteriormente ayudan a establecer una propuesta adecuada a las necesidades del sistema. Luego del análisis de esta información es necesario organizarla y sintetizarla para la elaboración de una proposición que contenga una estructura apropiada con los elementos fundamentales del objeto estudiado.

Métodos empíricos:

Medición: se usa para medir la calidad de la especificación de los requisitos y el grado de ambigüedad, además de obtener una medida de la calidad del diseño para su validación.

Resultados esperados:

1. Un análisis de los principales conceptos y tendencias asociadas a los sistemas de control de asistencia y permanencia así como su aplicación en la actualidad.
2. Los artefactos que propone la metodología que puedan dar continuidad al trabajo que se presenta.
3. Un sistema que permita el control de asistencia y permanencia con las características que necesitan los directivos del centro.

El contenido de este trabajo consta de tres capítulos, definidos de la siguiente forma:

Capítulo 1. Fundamentación teórica. Incluye un estudio del estado del arte de las diferentes aplicaciones existentes en el mundo y en Cuba, así como de sus características y funcionalidades. Se muestran, además, algunos conceptos asociados al dominio del

problema, la metodología que guiará todo el proceso de desarrollo del software, el lenguaje de modelado empleado por la misma, la herramienta de Ingeniería del Software Asistida por Computadora (CASE por su nombre en inglés) y las tecnologías utilizadas durante el desarrollo de la aplicación.

Capítulo 2. Análisis y diseño de la propuesta de solución. Contiene las principales características de la aplicación definidas a partir del estudio del estado del arte realizado en el Capítulo 1. Muestra además los requisitos funcionales y no funcionales con los que deberá cumplir la aplicación. Expone los principales artefactos definidos por la metodología de desarrollo; así como elementos del diseño web y aspectos esenciales del proceso de implementación.

Capítulo 3. Validación. Se detallan las métricas y técnicas realizadas para la validación de los requisitos y el diseño de la aplicación, así como las pruebas realizadas para validar las historias de usuario y el sistema. Finalmente se expresan los resultados obtenidos de las métricas, técnicas y pruebas definidas para cada uno de los casos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se describen los elementos principales que fundamentan el contenido de este trabajo. Se realizará una descripción de varios sistemas que se utilizan en Cuba y el resto del mundo para realizar el control de personas, los cuales fueron estudiados para la elaboración del sistema que se desea obtener. Se detallarán aspectos importantes de la metodología de desarrollo de software, así como del lenguaje de modelado y la herramienta a utilizar en el sistema. Finalmente se dará una descripción de las diferentes tecnologías que serán empleadas durante todo el proceso de desarrollo del software.

1.1 Aplicación de Sistemas Informáticos en el mundo

Sistemas informáticos extranjeros

LantecQuickPass

Es un software diseñado sin complejidades de manejo. Contempla interfaces con periféricos de control como: lectores de huellas digitales, tarjetas magnéticas, identificaciones con código de barras y firmas digitales. Emite listados de gestión, históricos por fecha y horario, por personas, grupos o categorías. Asimismo genera reportes sobre agendas previamente cargadas, y compara lo real con lo planificado, destacando cambios y desvíos.

Es una solución simple que permite controlar los horarios de llegada, salida y horas trabajadas por recurso, así como sus vacaciones y ausencias. Gracias a la emisión de reportes, los responsables de controlar al personal o los mismos propietarios, pueden obtener un listado detallado de los movimientos de un sector, departamento o de la organización en su totalidad. (Lainado, 2011)

Con el estudio de este sistema se optó por la generación de reportes que ayuden a tener determinada información centralizada, ya sea por persona, roles o proyecto y enmarcado en un tiempo específico. Esta funcionalidad mejorara el conocimiento de los directivos de

cómo se comporta el personal y brinda un apoyo para la toma de decisiones.

Software de Control de Asistencia.

El Sistema de Control de Asistencia de IBIX es un software que permite llevar el registro automático del tiempo laborado e incidencias del personal. Confronta el registro de entrada y salida contra el turno definido del trabajador, realizando un cálculo preciso del tiempo extra, tiempo de labor en día de descanso y en día festivo, lo que permite poder elaborar la nómina.

Por sus características obliga el cumplimiento de la jornada de trabajo del personal y establece un control sobre el tiempo extra, el cual representa un alto costo en las empresas. Mejora el desempeño en el registro de entrada y salida de los trabajadores y productividad del personal de nóminas. (IBIX, 2011)

Una vez terminado de analizar el software de control y asistencia, se evidenció, lo importante que es para el centro, llevar un control de las horas extras laboradas por su personal. Esta característica, junto con el Plan de Trabajo, ayuda en la decisión a tomar con respecto al pago adicional realizado a los trabajadores.

ASV_Gimnasios

El software ASV_Gimnasios permite controlar las personas que asisten a estos centros deportivos. Dentro de sus funciones están: gestionar los datos necesarios de los servicios que se brindan, así como los datos de las personas que lo utilizan y restringir el acceso por persona, tiempo y fecha.

Tiene como características principales el control de asistencia y acceso para saber cuándo y cuántas veces accedió un socio al recinto, y poder denegar su acceso de forma manual o automática sin necesidad de operador. En cualquier momento se pueden obtener listados de incidencias. También proporciona información sobre los socios sin asistencias desde una determinada fecha y permite mostrar en gráficos los datos de interés para los directivos. (ASV-GIMNASIOS, 2010)

ASV_Gimnasios permitió enfocarse en el papel de directivo del centro, donde desde su puesto debe tomar una serie de decisiones basadas en números. A partir de esa valoración se optó por enriquecer al sistema, particularmente los reportes, de gráficas que mejoraran la perspectiva de los resultados que se obtuvieran en ellos.

Sistemas informáticos cubanos

Biomesys

Biomesys es un sistema que aprovecha las bondades de las tecnologías que aplican la biometría, registra los eventos de asistencia de una organización por medio de la identificación de los empleados y de la autenticación de su identidad mediante un sensor biométrico de huellas dactilares. A partir de la captura de identificaciones biométricas únicas, el sistema se convierte en un generador de datos altamente confiable por su bajo o casi nulo nivel de vulnerabilidad por la suplantación de identidad.

El sistema puede trabajar de forma conectada y desconectada. Incluye una aplicación web para la administración y obtención de informes estadísticos, documentos que facilitan a los decisores el control de la asistencia del personal y permite realizar el mantenimiento de los datos del personal. (DATYS Tecnologías y Sistemas, 2010)

Este sistema permitió que se tuviera en cuenta que trabajar sin conexión era un elemento a considerar. A partir de ello se puede eliminar el temor que se pierdan horas trabajadas por no poder el usuario acceder al sistema. También se tomó en cuenta utilizar servicios web que pudieran brindar los datos del trabajador y así no introducir errores en la base de datos.

GESPRO

GESPRO es un paquete para la Gestión de Proyectos desarrollado por la Universidad de las Ciencias Informáticas. Debido a la gran cantidad de funcionalidades que facilita, es tomado como propuesta de herramienta para la gestión de los proyectos de la universidad, teniendo en la actualidad un entorno para cada centro. La interacción con la herramienta es a través de la web, por lo que es necesario tener instalado algún navegador web (Mozilla, Opera, IExplorer).

GESPRO es una aplicación web desarrollada con el marco de trabajo Ruby on Rails, que tiene la ventaja de ser software libre. Además, cuenta con una serie de módulos que amplían sus funcionalidades. Entre las principales características se tiene: el soporte de múltiples proyectos, foros, seguimiento al tiempo, publicación de noticias, control de tareas y proyectos, integración con manejadores de configuración de código, gestión de riesgos,

además del control y seguimiento de los recursos humanos. (Empresariales(CDAE)., L.d.S.d.G.d.P.G.C.d.C.y.D.d.A, 2011)

Con la revisión de las funcionalidades del sistema, se comprobó que permite una gestión integral de los proyectos. Dentro de sus características está la toma de asistencia de los integrantes del proyecto, pero, por políticas de la universidad, dificulta llevar el control de las actividades extras al proyecto, lo que dificulta resolver uno de los problemas actuales del centro.

De forma general, la mayoría de estos sistemas no pueden ser utilizados pues no son de código abierto, imposibilitando modificar sus características en función de las necesidades propias. Varios de estos software utilizan para su acceso sistemas biométricos o código de barras, actualmente la primera técnica es imposible utilizarla, con la segunda hay importantes adelantos en la universidad pero por falta de tecnología no es viable su uso inmediato. Algunas herramientas como el GESPRO permiten la gestión de los proyectos, pero no se pueden incluir actividades ajenas al mismo, las cuales forman también parte de la vida laboral.

El estudio detallado de todos estos sistemas demostró que el desarrollo de una nueva aplicación es necesaria. Con la investigación de cada uno de los escenarios de los software anteriores, se reconocieron puntos en común que permiten la obtención posibles funcionalidades.

1.2 Metodología, lenguaje y herramientas de modelado para el desarrollo

Con el objetivo de guiar el proceso e incrementar la calidad y el rendimiento del trabajo, se hace necesaria la elección de una metodología de desarrollo de software, así como una herramienta y un lenguaje de modelado que permitan confeccionar los artefactos requeridos.

1.2.1 Metodología de desarrollo

Una metodología de desarrollo se refiere a un marco de trabajo que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas informáticos. Se clasifican en dos tipos principales, metodologías ágiles y robustas.

Las metodologías ágiles tienen como filosofía centrarse en el factor humano y el producto, haciendo al cliente partícipe del proceso de desarrollo, dando mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas, se recomienda su uso en proyectos pequeños.

Por su parte las metodologías robustas se centran principalmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estos tipos de metodologías han demostrado ser efectivas y necesarias para un gran número de proyectos, principalmente proyectos grandes. (Orosco Vaillant, 2010)

Hoy día existe diversidad en las características de los proyectos, por lo que surge la necesidad de seleccionar una metodología que guíe eficientemente el proceso hasta el resultado esperado.

Es muy común encontrar, la definición de criterios de evaluación para la valoración del ambiente donde se desarrolla un proyecto. Para la selección de la metodología se aplicó el método de Boehm, el cual va encaminado a determinar el enfoque del proyecto dado sus características. (Boeras Velásquez, Cabrera Barroso, Llano Castro, Sánchez Gonzalez, Oval Riveron, & Hernández Luque, 2012)

Método aplicado para la selección de la metodología: Estrella de Boehm y Turner

El método de Boehm y Turner plantea cinco criterios fundamentales mediante los que se valora el proyecto; estos son: tamaño del equipo, criticidad del producto, dinamismo de los cambios, cultura del equipo y personal con que se cuenta. Cada uno de esos criterios tiene elementos que lo discriminan y por tanto se tienen en cuenta a la hora de seleccionar uno u otro enfoque. Para la selección del valor que se ubicará en cada eje (uno para cada criterio) de la estrella se debe tener en cuenta el comportamiento de estos criterios en el proyecto. En lo sucesivo se describe cada uno:

Tamaño: este criterio se utiliza para representar el número de personas involucradas en el proyecto. Pueden tenerse en cuenta el nivel de complejidad que pueda presentarse en la comunicación entre los miembros del proyecto y los costos que pueden provocar cambios esperados.

Criticidad: se utiliza para evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto. Su evaluación puede ser cualitativa.

Dinamismo: Representa la rapidez con la que pueden estar cambiando los requerimientos del proyecto.

Personal: representa la proporción del personal con experiencia alta, media y baja.

Cultura: las organizaciones y las personas que relaciona el proyecto pueden depender de la confianza o de la relación contractual. Esto refleja el nivel de ceremonia necesario y aceptado: documentación, control, formalismo en las comunicaciones. (Boeras Velásquez, Cabrera Barroso, Llano Castro, Sánchez Gonzalez, Oval Riveron, & Hernández Luque, 2012)

En la tabla 1 se muestran los umbrales que establece el método.

Criterios / Enfoques	Ágil	Híbrido	Robusta
Tamaño	1 - 10 personas	11 - 100 personas	101-300 personas
Criticidad	Se afecta la utilidad	Se afectan bienes	Se afectan vidas
Dinamismo	30% - 100%	5% - 29%	0% - 4%
Personal	Novato 0% - 10% Especialista 30% - 100%	Novato 10% - 60% Especialista 30% - 100%	Novato 60% - 80% Especialista 30% - 100%
Cultura	Personas acostumbradas a trabajar con bajo nivel de ceremonia.	Personas acostumbradas a trabajar con poca documentación pero con procesos definidos y controlables.	Personas acostumbradas a procesos y políticas definidas y controlables.

Tabla 1: Umbrales establecidos por el método Bohem y Turner

Una vez aplicado el método de Boehm y Turner para la selección de la metodología de desarrollo a utilizar y calculados los cinco criterios que define la misma, los resultados que arrojaron las variables fueron los siguientes: el tamaño es de una persona, para la criticidad se define un bajo nivel de pérdida, para el dinamismo se prevé un 30% de

modificación de los requisitos, el personal es el elemento que más se aleja del centro pues se cuenta con una persona desarrollando, que no tiene todas las habilidades creadas y para la cultura 90% de adaptación por lo poco compleja que es la solución que permite modificaciones sin crear un caos. En la Figura 1 se muestra la estrella de Boehm y Turner con los criterios ya orientados hacia la metodología.

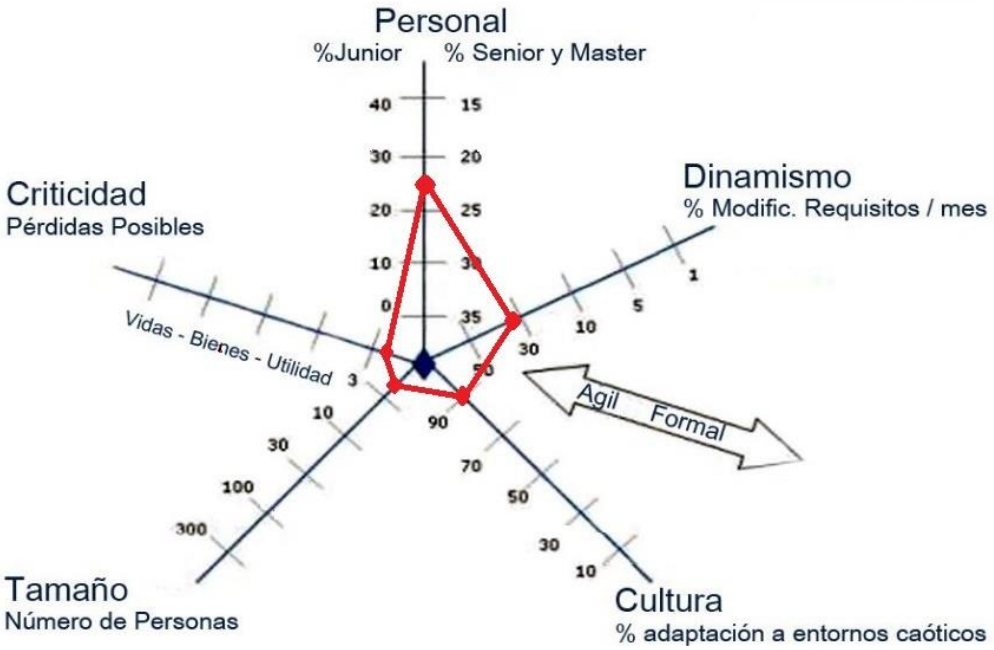

Figura 1: Representación de la estrella de Boehm y Turner

A raíz de los resultados obtenidos durante la aplicación del método de Boehm y Turner se propone el uso de una metodología con enfoque ágil.

El estudio de diversas bibliografías permitió conocer las características de algunas metodologías ágiles y resumir, en un cuadro comparativo, aquellos criterios que eran importante que se cumplieran para llevar el proceso a feliz término. En la Tabla 2 se presenta una comparación entre diferentes metodologías ágiles.

Metodologías / Criterios	Scrum	XP (Programación Extrema)	SXP
Duración de una iteración	Una semana o un mes	Aproximadamente 2 meses	Una semana o un mes (Romero, 2008)
Cambios en la iteraciones	Los equipos no permiten cambios	Los equipos son más susceptibles al	Los equipos no permiten cambios en

	en sus iteraciones.	cambio dentro de sus iteraciones.	sus iteraciones.
Prácticas de ingeniería	No prescribe práctica de ingeniería	Establece prácticas de ingeniería como el desarrollo basado en pruebas, el enfoque en pruebas automatizadas, la programación en parejas, diseño simple y refactorización. (Del Pino Valdarrama, 2005)	Establece prácticas de ingeniería como el desarrollo basado en pruebas, metáfora, la programación en parejas, diseño simple y refactorización. (Romero, 2008)
Prioridad de desarrollo	El equipo es quien determina la secuencia en la que se desarrollarán los elementos del *backlog*.	Los equipos trabajan en un orden de prioridad estricta determinado por el cliente.	Los equipos trabajan en un orden de prioridad estricta determinado por el cliente. (Romero, 2008)

Tabla 2: Comparación entre las metodologías ágiles.

Después de realizada la comparación entre las anteriores metodologías y basándose para la misma en los indicadores que demuestran su buen desempeño se propone como metodología a utilizar SXP. Es necesario aumentar los conocimientos que se tienen de la misma, es por ello que su estudio se profundiza a continuación.

SXP

La metodología ágil SXP es la unión de XP (Programación Extrema) y SCRUM. En la metodología se usa SCRUM para la planeación de los proyectos, dado que SCRUM en sí no es una metodología de análisis ni de diseño, es una metodología de gestión de trabajo. Para llevar a cabo el proceso de desarrollo se utiliza XP, con la idea principal de ir entregando versiones del producto que sean operativas, aunque no cuenten con las funcionalidades requeridas por el cliente final. (Orosco Vaillant, 2010)

SXP es un híbrido cubano de metodologías ágiles, que ofrece una estrategia tecnológica a partir de la introducción de procedimientos ágiles que permiten actualizar los procesos de software para el mejoramiento de la actividad productiva. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo y requisitos imprevistos, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados. Las entregas son frecuentes, y existe una refactorización continua, lo que permite mejorar el diseño cada vez que se le añada una nueva funcionalidad.

Consta de cuatro fases principales: Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto; Desarrollo, es donde se realiza la implementación del sistema hasta que esté listo para ser entregado; Entrega, puesta en marcha; y por último Mantenimiento, donde se realiza el soporte para el cliente. De cada una de ellas se despliegan siete flujos de trabajo: concepción inicial, captura de requisitos, diseño con metáforas, implantación, prueba, entrega de la documentación, soporte e investigación, el cual se utiliza por el equipo de desarrollo cuando sea necesario, es un flujo que se puede mover y utilizarlo en cualquier parte del ciclo de vida del proyecto.

De estos flujos se realizan numerosas actividades tales como el levantamiento de requisitos, diseño, implementación, planificación de las iteraciones, pruebas, así como la priorización y definición de artefactos como la Lista de reserva del producto y las Historias de usuario, además de las tareas necesarias para realizar las investigaciones para documentar todo el proceso. (Peñalver Romero, García de la Puente, & Meneses Abad, SXP, metodología de desarrollo de software, 2010)

1.2.2 Lenguaje de modelado

UML 2.0

Se utiliza UML pues es 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, y funciones del sistema.

Sirve para hacer modelos que permitan: visualizar cómo es un sistema o cómo se quiere que sea, especificar la estructura y/o comportamiento de los mismos, hacer plantillas que guíen su construcción y documentar las decisiones que se han tomado.

UML puede ser usado extensivamente en: recopilación de requerimientos, análisis de aplicaciones, diseño de sistemas, en pruebas, en implementación, en reingeniería y prácticamente en cualquier actividad de desarrollo que sea susceptible de ser modelada, además de ser útil en otros modelos tecnológicos ya que es independiente de lenguajes de programación o tecnología determinada.

Beneficios de UML:

- ✓ Mejores tiempos totales de desarrollo.
- ✓ Mejor soporte a la planeación y al control de proyectos.
- ✓ Mayor independencia del personal de desarrollo.
- ✓ Mayor soporte al cambio organizacional, comercial y tecnológico. (Zamudio, Ramírez, Álvarez, & Rodríguez, 2009)

Fundamentalmente facilitó al desarrollador visualizar los resultados de su trabajo en esquemas y diagramas estandarizados. Con UML se tuvo al alcance un estándar para escribir un "plano" del sistema, incluyendo aspectos conceptuales y concretos como las funciones del sistema, expresiones de lenguajes de programación y esquemas de bases de datos.

1.2.3 **Herramienta CASE**

Una vez que se conocen las necesidades y características del proyecto a desarrollar, llega el momento de seleccionar una herramienta de Ingeniería Asistida por Computadora (CASE, Computer Aided Software Engineering). Estas proporcionan ayuda al proceso del software automatizado en algunas de sus actividades como la ingeniería de requisitos, el diseño, el desarrollo de programas y las pruebas; además de brindar información acerca del desarrollo de software (Sommerville, 2005). Muchas son las herramientas CASE utilizadas dentro de las que se encuentran Visual Paradigm, ArgoUML y Rational Rose, por solo mencionar algunas.

Poseidon para UML

Es una de las herramientas CASE orientada a objetos, que cuenta con un amplio y completo grupo de diagramas para el modelamiento UML, tales como: Diagramas de estado, diagramas de clases, diagrama de paquetes, diagrama de caso de uso, diagrama de componentes, diagrama de actividades y diagramas de secuencia; además de contar con una interfaz de usuario que brinda comodidad y eficiencia. Desde el punto de vista del fabricante, Poseidon For UML proporciona estabilidad, escalabilidad, rendimiento, fiabilidad, personalización, al igual que califican su interfaz de usuario como la mejor de la industria.

También cuenta con algunas potentes funciones como la ingeniería de ida y vuelta y la generación de documentación que se han implementado de forma inteligente sin la carga común a otras herramientas UML.

Las desventajas con que cuenta es que la grabación de proyectos está limitada a ocho diagramas y no permite la integración con otras herramientas. (LÓPEZ ORTEGA & SANTA VILLA, 2012)

ArgoUML

Es una herramienta que contiene funciones avanzadas en las etapas de diseño y modelación de software. Presenta licencia comercial.

Como características fundamentales tiene que es modular y extensible. Soporta todas las especificaciones UML. Realiza los diagramas de apoyo a la Ingeniería de Software y aplica la ingeniería inversa a proyectos ya terminados. Se puede integrar con la herramienta AndroMDA.

Dentro de sus desventajas se encuentra que carece de soporte completo para algunos tipos de diagramas de secuencia y de colaboración. Asimismo los modelos creados una vez que se cierran no tienen posibilidad de edición. (Tigris.org Open Source Software Engineering Tools, 2009)

Visual Paradigm

Visual Paradigm es una herramienta de diseño de software, concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación

de todo tipo de diagramas. Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software.

Características:

1. Disponibilidad en múltiples plataformas (Windows, Linux).
2. Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
3. Disponibilidad de múltiples versiones, con diferentes especificaciones.
4. Compatibilidad entre ediciones.
5. Soporte de UML versión 2.1.
6. Importación y exportación de ficheros XML: esta característica permite seguir con el curso del diseño del software, pues define otra opción a parte de las opciones tradicionales (guardar y cargar) que presenta, esta y otras aplicaciones, para poder seguir trabajando con los modelados de los diagramas que se estén realizando. (Pressman, 2002)

Analizadas estas herramientas se decide utilizar Visual Paradigm, ya que genera todos los artefactos de la metodología seleccionada, cumpliendo con los estándares establecidos. Otra de las características por la que se decide usar Visual Paradigm es su disponibilidad en múltiples plataformas, ya que no obliga al usuario a desarrollar solo en el sistema operativo Windows, sino que está disponible en sistemas operativos como Windows, Linux, Unix. En comparación con las herramientas ArgoUML y Poseidon para UML, Visual Paradigm es una herramienta que permite integrarse con otras, sus modelos son editables y tiene soporte completo para los diagramas .Además de todo lo anterior expuesto, es la herramienta donde el desarrollador tiene más experiencia.

1.3 Tecnologías utilizadas para la solución del sistema

1.3.1 Marcos de trabajo

Los marcos de trabajo simplifican el desarrollo mediante la automatización de las tareas comunes, asimismo proporcionan estructura al código fuente, forzando al programador a

crear código más legible y fácil de mantener. La estructura de software intenta aliviar el exceso de carga asociado con actividades comunes usadas en desarrollos Web. Es por ello que el uso de un marco de trabajo se hace indispensable en el desarrollo de software, a continuación se hace un estudio para escoger con cuál trabajar.

Symfony Project

"Symfony es un marco de trabajo muy usado entre los usuarios y las empresas, ya que permite que los programadores sean mucho más productivos a la vez que crean código de calidad y facilidad de mantenimiento". (Eguíluz, 2010). Incluye su propio Mapeo Relacional de Objetos (ORM), llamados Propel y Doctrine. "ORM es una técnica empleada en la programación, para convertir datos entre sistemas incompatibles, como lo son las bases de datos relacionales y los lenguajes de programación. Esta conversión de datos entre los sistemas crea un efecto una base de datos virtual de objetos, que puede ser usada en el programa". (Amaya, 2009)

Utiliza el patrón arquitectónico Modelo-Vista-Controlador (MVC) para separar las distintas partes que forman una aplicación web. Este marco de trabajo simplifica al máximo el desarrollo de aplicaciones web profesionales, utilizando las mejores prácticas y los patrones de diseño más importantes. Este marco de trabajo se publica bajo licencia *Massachusetts Institute of Technology* (MIT), con la que se pueden desarrollar aplicaciones web comerciales, gratuitas y/o de software libre.

Además permite controlar hasta el último acceso a la información e incluye por defecto protección contra ataques. Su código fuente incluye más de 9.000 pruebas unitarias y funcionales. Es un marco de trabajo bien documentado, ya que ha publicado cinco libros y siempre actualizados. (Eguíluz, 2010)

Zend Framework (ZF)

Zend Framework es un marco de trabajo de código abierto para desarrollar aplicaciones y servicios web con PHP5 (*Hypertext Preprocessor)*. Zend Framework es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de Zend Framework es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado.

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de ZF conforman un potente y extensible marco de trabajo de aplicaciones web al combinarse. ZF ofrece un rendimiento eficiente y una excelente implementación MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios en Lenguaje de Marcado de Hipertexto (HTML), validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. (zend framework2, 2010)

Comparación entre los marcos de trabajos Symfony y ZF, el primero, a diferencia de su homólogo, posee su propio ORM, algo fundamental ya que *Hypertext Preprocessor* 5 (PHP 5), es un lenguaje orientado a objetos y para acceder de forma efectiva a la base de datos desde un contexto orientado a objetos, es necesaria una interfaz que traduzca la lógica de los objetos a la lógica relacional. “La principal ventaja que aporta el ORM es la reutilización, permitiendo llamar a los métodos de un objeto de datos desde varias partes de la aplicación e incluso desde diferentes aplicaciones. El patrón MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones**”.** (Potencier, 2008) Otra característica que se tuvo en cuenta al escoger Symfony es la amplia documentación, código en línea y consultores que existen en la web, también es el marco de trabajo utilizado en CEGEL, por lo que hay experiencia en su uso.

Otros marcos de trabajo que son necesarios para el desarrollo del sistema se describen a continuación:

JQuery 2.0

JQuery es una biblioteca de JavaScript rápida, pequeña y de fáciles funciones, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones, agregar interacción con la tecnología AJAX a páginas web y manipular el árbol DOM (Modelo de Objetos del Documento); que no es más que una interfaz de programación de aplicaciones para acceder, añadir y cambiar dinámicamente contenido estructurado en los documentos.

jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las

funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Dentro de sus características se encuentra la interactividad y modificaciones del árbol DOM, incluyendo la manipulación y soporte para CSS 1, 2 y 3. También el uso de la técnica de desarrollo web AJAX para crear aplicaciones interactivas. De manera general presenta varias utilidades como: obtener información del navegador y operar con objetos y vectores. (jQuery escribir menos, haces más, 2014)

El uso de esta biblioteca simplificó la implementación hacia el lenguaje Javascript, pues en ella se establecen funciones que ya están definidas, lo que ayudó a que se obtubieran métodos más rápidos y completos que los implementados con el lenguaje en su forma nativa. Es una de las bibliotecas utlizadas por el marco de trabajo boostrap en su implementación.

Bootstrap 2.6

Bootstrap es un marco de trabajo que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. La mayor ventaja es que se pueden crear interfaces que se adapten a los distintos navegadores apoyándose en un marco de trabajo potente con numerosos componentes webs, los cuales ahorran esfuerzo y tiempo.

Bootstrap se integra perfectamente con las principales bibliotecas Javascript, por ejemplo JQuery. Además ofrece un diseño sólido usando estándares como CSS3/HTML5 y funciona con todos los navegadores, incluido Internet Explorer. Es un marco de trabajo ligero que se integra de forma limpia en el proyecto actual y dispone de distintos niveles predefinidos con estructuras fijas a 940 píxeles de distintas columnas o diseños fluidos. (Rodríguez, 2012)

Este marco de trabajo es seleccionado por la amplia biblioteca de componentes de estilo predeterminado que contiene. Asimismo, otro objetivo de su uso está, en darle un aspecto visual agradable al proyecto sin dedicar mucho tiempo a este proceso, y que gracias a las facilidades ofrecidas por la herramienta se vuelve más sencillo de hacer.

Doctrine 2.14.1

Doctrine es un mapeador de objetos relacional (ORM) escrito en PHP que proporciona persistencia para objetos PHP. Está por encima de la capa de abstracción a la base de datos, una de sus características es la posibilidad de escribir consultas en las mismas a partir del tratamiento con objetos en PHP llamado DQL.

Contiene soporte para jerarquía (árbol de estructura) de datos, para los ganchos (métodos que pueden validar o modificar la base de datos de entrada y salida) y los eventos a la estructura lógica de negocio relacionado con ellos. Doctrine puede generar clases a partir de una base de datos creada en la cual, el programador, puede especificar relaciones y agregar funcionalidades comunes para las clases generadas. Se pueden escribir consultas a la base de datos a partir de la programación orientada a objetos, llamada DQL.

Objetos similares se pueden almacenar en la tabla de base de datos, con un tipo de columna que especifica el subtipo del objeto en particular, la subclase correcta siempre se devuelve cuando se hace una consulta. (doctrine, 2014)

Se empleó la biblioteca doctrine para la generación de las entidades del sistema, así como el modelo de datos de la aplicación, pues cuenta con comandos de consola que permiten realizar esta acción. Presenta funciones definidas que hacen más fácil el acceso hacia la base de datos, las cuales fueron utilizadas para realizar consultas sobre los datos.

1.3.2 Lenguajes de programación

“Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora”. (Bedoya, 2006) Los lenguajes de programación web han ido surgiendo según las tendencias y necesidades de las plataformas, intentando además facilitar el trabajo de los desarrolladores de aplicaciones. Estos se clasifican en lenguajes del lado cliente y del lado servidor.

La elección de Symfony Project como marco de trabajo a utilizar, hace evidente la utilización de PHP como lenguaje de programación web del lado del servidor. Por otra parte se utilizarán como lenguajes de programación del lado del cliente: HTML, Javascript y CSS.

La elección de estos lenguajes se realizó debido a que:

- ✓ Son lenguajes de programación web.
- ✓ Permiten facilidades de extensión como procedimientos y abstracción de datos.
- ✓ Presentan habilidades de integración e utilización de bibliotecas desarrolladas en otros lenguajes, como el uso de bibliotecas de java.
- ✓ Son los que mejor se adaptan con la descripción del sistema que se quiere desarrollar.

PHP5

PHP (Preprocesador de Hipertexto) es un lenguaje de código abierto, adecuado para el desarrollo web, que puede ser incrustado en HTML y centrado en programación de scripts del lado del servidor. El código es ejecutado en el servidor, generando HTML y enviándolo al cliente.

PHP puede usarse en todos los sistemas operativos entre los que se encuentran Windows y Linux y admite la mayoría de servidores web incluyendo Apache. Puede usar la programación por procedimientos, la programación orientada a objetos (POO) o una mezcla de ambos.

Entre sus capacidades se incluyen la creación de imágenes y ficheros PDF. También se puede generar textos como XHTML y ficheros XML. PHP puede autogenerar estos ficheros y guardarlos en el sistema de ficheros en vez de imprimirlos en pantalla, creando una caché en el lado del servidor para contenido dinámico.

Tiene soporte para una amplia gama de bases de datos, así como el intercambio de datos complejos entre todos los lenguajes de programación web. Permite la comunicación con otros servicios usando protocolos como LDAP e IMAP. Admite el procesamiento de texto, las cuales incluyen las expresiones regulares. (Achour, 2014)

JavaScript

JavaScript es un lenguaje interpretado que permite incluir macros en páginas web. Estas macros se ejecutan en las páginas web del ordenador del visitante, y no en el servidor, comprobando los datos que el usuario introduce en un formulario antes de enviarlos.

A su vez proporciona los medios para controlar las ventanas del navegador y el contenido que muestran, así como programar páginas dinámicas simples. Dentro de sus funciones se encuentra evitar depender del servidor web para cálculos sencillos y capturar los eventos generados por el usuario respondiendo a ellos sin salir a Internet.

Una de la característica de JavaScript que más simplifica la programación es que, aunque el lenguaje soporta cuatro tipos de datos, no es necesario declarar el tipo de las variables, argumentos de funciones ni valores de retorno de las funciones. El tipo de las variables cambia implícitamente cuando es necesario, lo que dificulta el desarrollo de programas complejos, pero ayuda a programar con rapidez macros sencillas. (E.T.S.I.Informática Dpto.L.C.C. U.M.A, 2013)

Este lenguaje es utilizado por el marco de trabajo bootstrap en su implementación. Se usa para la programación que se realiza del lado del cliente(páginas web), mostrando una vista interactiva para el usuario. Es utilizado por bibliotecas como jquery, los cuales ayudan que su implementación en las páginas sea más rápido y eficiente.

HTML5

HTML (Lenguaje Marcado de Hipertexto) es un lenguaje de hipertexto, que permite escribir texto de forma estructurada, y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento. Los navegadores se encargan de interpretar el código HTML de los documentos, y de mostrar a los usuarios las páginas web resultantes.

HTML5 está formado por módulos, cuyo grado de especificación está en niveles dispares. Por tanto, muchas de las características de HTML5 están ya listas para ser implementadas en un punto de desarrollo que se encuentra cercano al que finalmente será presentado.

Novedades de HTML5:

- Estructura del cuerpo: permite agrupar todos los fragmentos de una web en nuevas etiquetas que representarán cada una de las partes típicas de una página.

- Etiquetas para contenido específico: se utilizan etiquetas específicas para cada tipo de contenido en particular, como audio y video.

- Aplicaciones web Offline: existirá otro API para el trabajo con aplicaciones web, que se podrá desarrollar de modo que funcionen también en local y sin estar conectados a Internet. (htmlcinco, 2012))

El empleo de este lenguaje permitió estructurar el contenido de las vistas del sistema, pues a través de él se describe la estructura y el contenido en forma de texto. Este lenguaje es utilizado por el marco de trabajo bootstrap, biblioteca usada en el sistema para la decoración de las vistas.

CCS3

CSS (Hojas de Estilo en Cascadas) es un lenguaje de estilo que define la presentación de los documentos HTML. CSS abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo y posicionamiento avanzado. CSS está soportado por todos los navegadores y se usa para formatear el contenido previamente estructurado.

Beneficios de CCS:

- ✓ Control de la presentación de muchos documentos desde una única hoja de estilo.
- ✓ Aplicación de diferentes presentaciones a diferentes tipos de medios (pantalla, impresión). (HTML.net, 2010)

Con el uso de este lenguaje se obtienen en el sistema interfaces más agradables para el cliente. Permite disponer de un estilo único para las páginas de la aplicación, definiendo estilos para los diferentes componentes que conforman las vistas del sistema.

1.3.3 Herramientas

Gestor de Base de Datos PostgreSQL 9.3

PostgreSQL es un sistema de gestión de base de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. PostgreSQL utiliza un modelo cliente/servidor, así como multiprocesos en vez de multihilos, con el objetivo de garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

Entre sus características se encuentran: copias de seguridad, múltiples métodos de autenticación, completa documentación, está disponible para Linux y UNIX en todas sus variantes y Windows 32/ 64 bit. (Martínez, 2010)

PostgreSQL es un gestor de base datos que depende de aplicaciones gráficas para poder diseñar, manejar y gestionar la base de datos, la cual permite realizar desde simples consultas hasta las más complejas. Es por ello que se hace impresindible el uso de una herramienta que soporte este gestor y para eso se usará pgAdmin3, pues la misma se integra y funciona con postgreSQL.

Cliente para la administración de PostgreSQL

PgAdmin3 es una aplicación de diseño y manejo de bases de datos para su uso con PostgreSQL. Funciona sobre casi todas las plataformas. Se pueden escribir desde simples consultas SQL hasta la elaboración de bases de datos complejas. La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración. La aplicación incluye un editor de la sintaxis SQL y un editor de código del lado del servidor. Incorpora funcionalidades para realizar consultas, examinar su ejecución y trabajar con los datos. No requiere de controladores adicionales para comunicarse con la base de datos del servidor. (pgAdmin PosgreSQL Tools, 2010)

NetBeans 7.3

El NetBeans es una herramienta creada para escribir, compilar, depurar y ejecutar programas computacionales. Dentro de los lenguajes que soporta se encuentra PHP,

HTML, JavaScript, CSS, lenguajes utilizados para escribir el software que se propone. Es un producto de software libre y gratuito sin restricciones de uso, escrito completamente en Java usando la plataforma NetBeans. Permite integrar Symfony2 como marco de trabajo para proyectos realizados con el lenguaje PHP.

Características:

- ✓ Contiene editores, analizadores de código, y convertidores, los cuales hacen que se puedan actualizar las aplicaciones a utilizar.
- ✓ Los datos pueden ser visualizados desde diferentes vistas, pues permite la visión de múltiples ventanas de diferentes proyectos que hayan sido cargados y abiertos en la aplicación.
- ✓ Proporciona editores y herramientas integrales para los marcos de trabajo y las tecnologías relacionadas. (Netbeans, 2013)

Esta herramienta presenta un amplio soporte para varios leguajes de programación, entre los que se encuentran PHP, HTML, CCS y Javascript, asimismo varios marcos de trabajo utilizados para optimizar el uso de estos lenguajes, como symfony2 y jquery, permitiendo detectar los errores en estos últimos y corregirlos. Soporta las acciones que se realizan a través de comandos para el marco de trabajo symfony2, como son la creación de entidades y tablas de la base de datos, así como su creación y actualización.

Servidor web apache 2.0

Apache es un servidor que permite acceder a páginas web alojadas en un ordenador. El servidor Apache es usado por múltiples razones como su disponibilidad, facilidad de instalación, necesidad de pocos recursos de hardware, precio y disponibilidad del código fuente. Presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. Entre sus principales ventajas se encuentran que posee gran cantidad de extensiones para diversas tecnologías, además de una amplia documentación, es libre, modular y multiplataforma. (The Apache Software Foundation, 2012)

Debido a que el servidor estará implementado sobre la plataforma de software libre GNU/Linux, se decide utilizar como servidor: Apache en su version 2.0, por ser esta una

herramienta con software de código abierto multiplataforma. Es uno de lo servidores más potentes que existen actualmente en el mercado y presenta una amplia documentación.

Conclusiones parciales

En el capítulo se analizaron los conceptos fundamentales de desarrollo de software, a partir de los cuales se obtuvieron resultados que permiten continuar la investigación, llegándose a las siguientes conclusiones:

- El estudio de sistemas que siguen la misma línea temática, permitió obtener elementos base para orientar el desarrollo.
- A partir de los rasgos que definen el proyecto se estableció la metodología, la cual guiará todo el proceso de desarrollo del software.
- La investigación de tecnologías arrojó como resultado identificar aquellas que eran adecuadas para implementar un sistema acorde con las características deseadas.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN

Introducción

En este capítulo se realiza una descripción de la solución propuesta a través del Modelo del Dominio. Se detallará la propuesta del sistema así como los requisitos que debe cumplir los mismos (funcionales y no funcionales), los cuales permitirán obtener una concepción general del sistema, además de la descripción de los artefactos para el análisis y diseño que se generan durante el flujo de trabajo.

2.1 Propuesta de solución

Con el fin de dar solución a los problemas planteados en la introducción de este trabajo, se propone diseñar e implementar una aplicación web dinámica que permita gestionar toda la información referente al horario laboral de los trabajadores de CEGEL.

El proceso de firma procederá de la siguiente forma: cuando los trabajadores se autentiquen en la aplicación, automáticamente se guardará temporalmente la hora de entrada, así como la fecha, una vez que se desconecten, se registrará la hora de salida para ese trabajador.

Para comprobar la llegada de los usuarios al laboratorio de producción se designará por cada local de un usuario con rol chequeador, que será el encargado de comprobar si las personas que firmaron están físicamente en su puesto de trabajo. Una vez que este usuario chequee a los usuarios registrados, es cuando se guarda de forma permanente la hora de entrada. La aplicación será la encargada de avisarle a este usuario la hora para realizar esta actividad.

El software también brindará la opción de generar reportes sobre la asistencia de los trabajadores, lo cual ayudará a la toma de decisiones por parte de sus jefes y/o superiores. A su vez le brindará al trabajador la posibilidad de adicionar las actividades extras, las cuales por factores ajenos al personal no pudieron ser expresadas en el Plan de Trabajo. El sistema permitirá insertar, modificar, editar y eliminar usuarios, actividades, proyectos y horarios, así como generar reportes sobre la asistencia de los trabajadores.

Para la implementación y funcionamiento del sistema se utilizará el lenguaje PHP usando el marco de trabajo symfony2, el cual se implementará sobre el IDE de desarrollo Netbeans. Para el acceso a la base de datos se utilizará como gestor PostgreSQL, como lenguaje para la consulta de información DQL y como manager pgAdmin III.

2.2 Modelo del dominio

Dentro de las actividades más importantes determinadas en la metodología SXP, se encuentra la definición del modelo de historias de usuario del negocio, en el cual se hace una detallada descripción del negocio en cuestión. En los casos que el negocio no esté bien definido entre los clientes y los ejecutores del proyecto, asimismo los requisitos son cambiantes y el cliente no tiene certeza absoluta de lo que en realidad quiere; entonces es generado el llamado modelo de dominio. El negocio que se quiere modelar tiene todas las características para escoger el modelo de dominio. El mismo comienza con el cliente que, luego de iniciar sesión, se le inserta su información en la base de datos. El Administrador es el encargado de crear los horarios, los proyectos, las actividades. El Usuario a su vez también puede crear actividades, con la diferencia que el Administrador la puede crear para uno o varios usuarios, mientras que el Usuario la crea solo para él. En la aplicación se generan varios reportes sobre la información de los usuarios, lo cuales son mostrados al Administrador y al Usuario, al primero exponiendo los resultados de todos los usuarios y al segundo solamente sus datos.

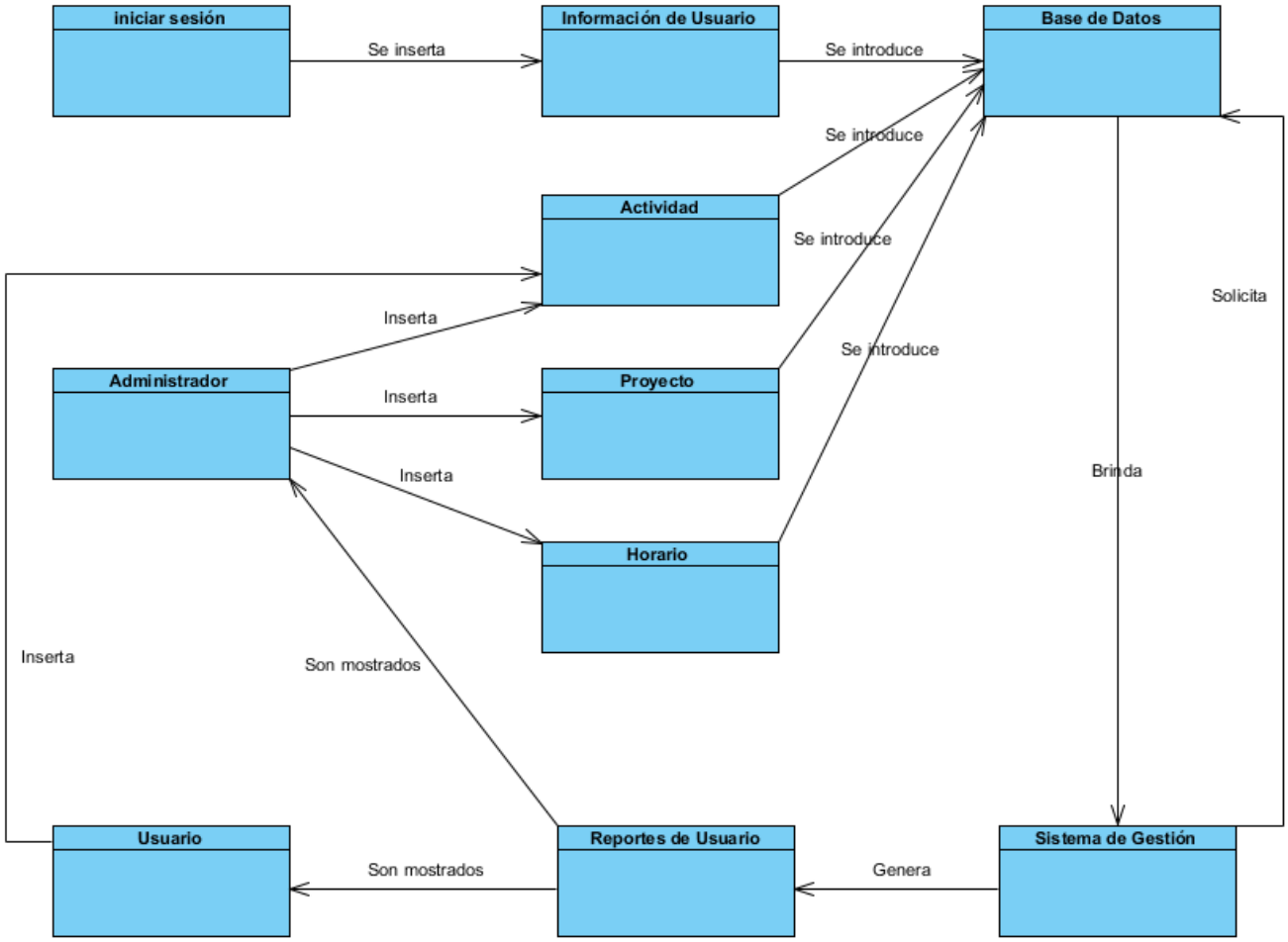

Figura 2: Modelo del dominio

2.3 Requisitos del software

Técnicas y criterios utilizados para el trabajo con los requisitos del sistema

En principio, parece bastante simple, investigar cómo los sistemas o productos se ajustan a las necesidades del negocio, y finalmente, cómo el sistema o producto va a ser utilizado en el día a día. La práctica ha demostrado que es un proceso complejo, que muchas veces forma un cuello de botella durante el desarrollo, pero que finalmente rinde frutos si es realizado correctamente.

Es por ello que surgen diferentes técnicas que ayudan a comprender el problema, proponer soluciones, negociar diferentes puntos de vista y finalmente especificar un conjunto básico de requisitos de la solución. A continuación se definen algunas técnicas y criterios que viabilizan este trabajo.

Técnicas de captura de requisitos

Algunas técnicas utilizadas para la captura de requisitos son las entrevistas, las reuniones con el cliente, los cuestionarios, la lluvia de ideas, el análisis de sistemas existentes, la arqueología de documentos y los prototipos. Para el levantamiento de requisitos del sistema se aplicaron las técnicas de reuniones, Análisis de sistemas existentes, lluvia de ideas y construcción de prototipos.

- ✓ **Reuniones:** se realizaron varias reuniones con el cliente con el objetivo de identificar la mayor cantidad de funcionalidades posibles que dieran soporte para el sistema que se quiere implementar.
- ✓ **Análisis de sistemas existentes:** se estudiaron varios sistemas relacionados con el control del personal en diversas instituciones con el objetivo de identificar funcionalidades que se pudieran adaptar al software que se quiere desarrollar.
- ✓ **Lluvia de ideas:** se realizaron varios debates por parte de los involucrados con el fin de llegar a un consenso sobre las funcionalidades con las que tendría que contar el sistema a desarrollar.
- ✓ **Construcción de prototipos:** se realizaron prototipos de las interfaces de la aplicación con el objetivo de diseñar lo más cercano posible lo que se quería lograr con el sistema a implementar.

Criterios utilizados para el trabajo con los requisitos

Los requisitos funcionales del sistema requieren de una complejidad y para esto fue necesario trabajar con una serie de criterios ya determinados en la ingeniería de software. A partir de una herramienta desarrollada en Excel, por el Programa de mejoras de CALISOFT, se evalúan estos indicadores y se establece una valoración final la cual corresponde a la complejidad de cada requisito funcional.

A continuación se muestra la Tabla 3 con los criterios que se tuvieron en cuenta para la actividad antes mencionada:

Criterios de Complejidad
Para la determinación de la complejidad de los requisitos se analizan individualmente los criterios siguientes, llegando al resultado de si el requisito es de complejidad Alta, Media o Baja. La clasificación de la complejidad permite estimar el esfuerzo de implementación del requisito y contribuye a la decisión sobre la inclusión en las etapas de desarrollo del software.
• **Complejidad por Interfaces:** El criterio interfaz se aplica a requisitos que presenten algún tipo de complejidad en su interacción con los siguientes elementos: • Humanas (formularios, informes) • Equipo (Tomógrafos, Rayos X); Ej API, drivers • Programación (Programas externos necesarios para apoyar el producto); Ej. Adicionar un nuevo usuario, usa el LDAP) • Comunicación (Protocolos de comunicación que serán utilizados); Ej. HL7
• **Diferentes comportamientos:** Un mismo requisito se comporta de manera diferente ante determinadas situaciones; Ej. Admisión de un paciente, puede ser normal, o por emergencia, en el primer caso recoge más información que en el segundo.
• **Formas de inicialización:** Un mismo requisito puede ser inicializado de diferentes formas, Ej. Ver detalles de una historia clínica, se inicializa cuando creas la historia clínica en una vista previa, y cuando se selecciona mostrar historia clínica.
• **Consultas a fuentes de almacenamientos:** Los requisitos pueden presentar diversidad en la cantidad y complejidad de la interacción con la fuente de datos. • Base de Datos • Ficheros • Otros
• **Restricciones de validación:** Complejidad de todas las validaciones que lleve un requisito, tanto las validaciones en el lado del cliente, como en el servidor.
• **Grado de reutilización:** Complejidad de un requisito, para poder ser reutilizado por otros.
• **Lógica de negocio:** Los requisitos pueden presentar diferentes niveles de complejidad para la implementación

Tabla 3: Criterios de complejidad

2.4 Especificación de los requisitos del software

Los requisitos son la descripción de los servicios y restricciones de un sistema de software, es decir, lo que el software debe hacer y bajo qué circunstancias debe hacerlo. (Somerville, 2008)

2.4.1 Requisitos funcionales

Los requisitos funcionales (RF) son aquellos que indican lo que debe hacer el producto, son las capacidades con las que debe cumplir. Aunque la metodología empleada no especifica la lista de requisitos como un artefacto, el equipo de trabajo se vio en la necesidad de ir documentándolos, a medida que surgían, dado que es un sistema del cual no se tenía referencia y que fue tomando forma a partir de las técnicas de captura de requisitos practicadas. En el sistema se obtuvieron en total 31 requisitos funcionales.

1. Autenticar usuario
2. Crear usuario
3. Modificar usuario
4. Eliminar usuario
5. Mostrar usuario
6. Crear actividad
7. Modificar actividad
8. Eliminar actividad
9. Mostrar actividad
10. Crear proyecto
11. Modificar proyecto.
12. Eliminar proyecto.
13. Mostrar proyecto.
14. Crear horario
15. Modificar horario
16. Eliminar horario
17. Mostrar horario
18. Registrar asistencia
19. Buscar asistencia
20. Reporte sobre las horas extras de los usuarios
21. Reporte sobre las tardanzas de los usuarios
22. Reporte sobre las horas trabajadas de los usuarios
23. Reporte sobre la asistencia de los usuarios
24. Reporte sobre las ausencias de lo usuarios

25. Gráfico sobre las horas extras de lo usuarios
16. Gráfico sobre las tardanzas de los usuarios
27. Gráfico sobre las horas trabajadas de los usuarios
28. Gráfico sobre la asistencia de los usuarios
29. Gráfico sobre las ausencias de lo usuarios
30. Exportar reporte en formato pdf
31. Avisar mediante mensaje al usuario chequeador

Seguidamente en la Tabla 4 se muestra la especificación de los requisitos Autenticar usuario. Este son de gran importancia, pues a través de él se realiza la restricción de la entrada de los usuarios al sistema. (Ver documento 0113_Especificación de Requisitos de Software 1.4).

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF_US_1	Autenticar usuario	Permite autenticar el usuario	Alta	Alta
	Prototipo			

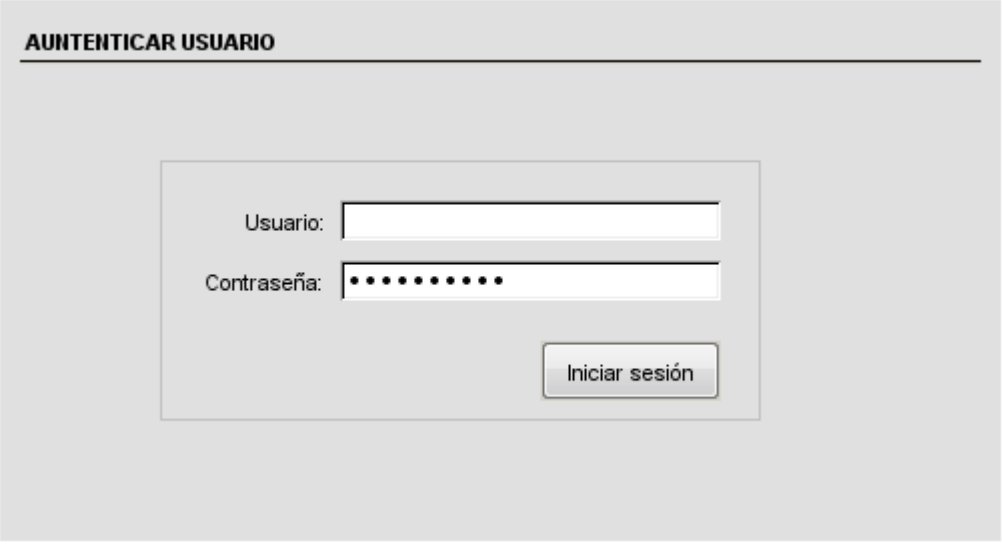

Figura.1.1 Autenticar usuario.

Campos	**Tipos de Datos**	**Reglas o Restricciones**
Usuario	Cadena	Obligatorio. El usuario tiene que estar registrado en el

		sistema
Contraseña	Cadena	Obligatorio. Admite letras, números y caracteres especiales
Observaciones	Solo se admiten 3 intentos fallidos para la contraseña, después quedará bloqueado dicho usuario.	

Tabla 4: Especificación del requisito funcional "Autenticar usuario"

2.4.2 Requisitos no funcionales

Los requisitos no funcionales (RnF) se refieren a las características o cualidades que debe tener el producto para hacerlo atractivo, usable, rápido, confiable y seguro. Para el desarrollo de la aplicación web se definieron 11 RnF, los que se clasificaron de usabilidad, confiabilidad, eficiencia, soporte, restricciones de diseño, de interfaz, de licencia, estándares aplicables, seguridad y requisitos legales de derecho de autor (Ver documento 0113_Especificación de Requisitos de Software 1.4).

Entre los requisitos no funcionales presentes en la aplicación sobresalen los de seguridad, por su importancia durante todo el ciclo de vida del sistema, con los cuales se puede garantizar una máxima seguridad en la aplicación frente a los diferentes ataques que se pueden originar durante el desarrollo de la misma. De estos requisitos no funcionales se pueden destacar los siguientes:

1. Restringir el acceso al sistema por un rango de IP.

   El acceso a la aplicación se realizará desde el lugar establecido para trabajar (locales de producción) y estará regulada por un rango de IP para cada uno de los proyectos.

2. Restringir el horario de acceso al sistema.

   El acceso a la aplicación se realizará dentro del horario establecido, en caso de horario no establecido se debe explicar las razones para acceder a la misma.

3. Restringir la entrada de una persona a partir de un rango de fecha.

   El acceso a la aplicación se realizará a partir del rango de fecha establecida.

2.5 Modelación del sistema

2.5.1 Lista de Reserva del Producto (LRP)

Es una lista priorizada que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requisitos sobre el producto. Sin embargo, suelen surgir los más importantes, que casi siempre son más que suficientes para una iteración.

Esta lista puede crecer y modificarse a medida que se obtiene más conocimiento acerca del producto y del cliente, con la restricción de que solo puede cambiarse entre iteración. El objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible, y para esto la lista debe acompañar los cambios en el entorno y el producto. (Peñalver Romero, 2008)

La presente lista fue pasando por varias iteraciones en las cuales los requisitos fueron cambiando hasta llegar a la tercera y última iteración, en la que se definieron los requisitos finales del sistema.

Prioridad	Ítem *	Descripción	Estimación	Estimado por
Muy Alta				
Alta				
	1	Autenticar usuario	5 días	Programador
	2	Crear usuario	5 días	Programador
	3	Modificar usuario	4 días	Programador
	4	Eliminar usuario	4 días	Programador
	5	Crear actividad	4 días	Programador
	6	Modificar actividad	4 días	Programador
	7	Eliminar actividad	4 días	Programador
	8	Crear proyecto	5 días	Programador
	9	Modificar proyecto	4 días	Programador
	10	Eliminar proyecto	4 días	Programador
	11	Crear horario	4 días	Programador
	12	Modificar horario	4 días	Programador
	13	Eliminar horario	4 días	Programador
	14	Registrar asistencia	5 días	Programador
	15	Buscar asistencia	5 días	Programador
Media				
	16	Mostrar usuario	3 días	Programador
	17	Mostrar actividad	3 días	Programador
	18	Mostrar proyecto	3 días	Programador

	19	Mostrar horario	3 días	Programador
	20	Reporte sobre las horas extras del usuario	3 días	Programador
	21	Reporte sobre las tardanzas de los usuarios	3 días	Programador
	22	Reporte sobre las horas trabajadas de los usuarios	3 días	Programador
	23	Reporte sobre la asistencia de los usuarios	3 días	Programador
	24	Reporte sobre las ausencias de los usuarios	3 días	Programador
	25	Gráfico sobre las horas extras de los usuarios	3 días	Programador
	26	Gráfico sobre las tardanzas de los usuarios	3 días	Programador
	27	Gráfico sobre las horas trabajadas de los usuarios	3 días	Programador
	28	Gráfico sobre las ausencias de los usuarios	3 días	Programador
	29	Gráfico sobre las horas extras de los usuarios	3 días	Programador
	30	Exportar reporte en formato pdf	2 días	Programador
	31	Enviar correo	2 días	Programador
Baja(Requisitos no funcionales)				
	32	Usabilidad		
	33	Confiabilidad		
	34	Eficiencia		
	35	Soporte		
	36	Restricciones de diseño		

Tabla 5: Lista de Reserva del Producto (LRP)

2.5.2 Historias de Usuario

Las historias de usuarios es el artefacto que utiliza la metodología SXP para especificar los requisitos del software y constituyen la base para las pruebas funcionales. Se definieron

cinco historias de usuario, a continuación se muestran las que están involucradas en la restricción de la entrada de los usuarios al sistema: Gestionar usuario y Gestionar proyecto.

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Gestionar usuario
Modificación de Historia de Usuario Número: ninguna	
Usuario: Juan Enidio Martínez Costa	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 3 semanas (21 días)
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2,42 semanas (17 días)
Descripción: Permite autenticar, crear, modificar, eliminar y visualizar los usuarios.	
Observaciones: El usuario se crea en el sistema desde la primera vez que el mismo se autentica en la aplicación con los siguientes datos: nombre, apellidos, usuario, contraseña, hora de entrada, fecha. Una vez registrado en el sistema, cada vez que acceda a la aplicación lo que se registra es la hora de entrada y la fecha, y cuando sale de la misma se registra la hora de salida. Para modificar, visualizar o eliminar un usuario, existirá una tabla donde se mostrarán todos los datos de los usuarios. Esta tabla tendrá una columna adicional, la cual contendrá una opción desplegable para poder seleccionar la opción a realizar sobre ese usuario, ya sea de modificar, visualizar o eliminar. En el caso que se elija la opción de modificar usuario aparecerá un formulario con los siguientes campos a modificar: rol en el proyecto, rol en el sistema.	
Prototipo de interfaz: Ver en el documento ERS del Proyecto Sistema de control del personal los siguientes requisitos funcionales: **RF_US_1** Autenticar usuario. **RF_US_2** Crear usuario. **RF_US_3** Modificar usuario.	

RF_US_4 Eliminar usuario.

RF_US_5 Visualizar usuario.

Tabla 6: Historia de Usuario “Gestionar usuario”

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Gestionar proyecto
Modificación de Historia de Usuario Número: ninguna	
Usuario: Juan Enidio Martínez Costa	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2,28 semanas (16 días)
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1,71 semanas (12 días)
Descripción: Permite crear, modificar, eliminar y visualizar proyectos.	
Observaciones: Para crear un proyecto en el sistema se mostrará un formulario con los siguientes campos: nombre del proyecto, laboratorio, integrantes. Para modificar, visualizar o eliminar un proyecto se mostrará una tabla donde estarán todos los datos de los proyectos. Esta tabla tendrá una columna adicional, la cual contendrá una opción desplegable para poder seleccionar la opción a realizar sobre ese proyecto, ya sea de modificar, visualizar o eliminar. En el caso que se elija la opción de modificar el proyecto, se mostrará un formulario con los mismos campos de crear proyecto.	
Prototipo de interfaz: Ver en el documento ERS del Proyecto Sistema de control del personal los siguientes requisitos funcionales: **RF_US_2** Crear proyecto. **RF_US_3** Modificar proyecto. **RF_US_4** Eliminar proyecto. **RF_US_5** Visualizar proyecto	

Tabla 7: Historia de Usuario “Gestionar proyecto”

2.5.3 Tareas de ingeniería

Es uno de los artefactos que genera la metodología SXP correspondiente en la fase de desarrollo al flujo de planificación, en el que se recogen las tareas por historia de usuario a realizar.

Tarea de Ingeniería	
Número Tarea: 1.1	**Número Historia de Usuario:** HU_1
Nombre Tarea: Implementación de la funcionalidad autenticar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 5 días
Fecha Inicio: 10/1/2014	**Fecha Fin:** 16/1/2014
Programador Responsable: Juan Enidio Martínez Costa	
Descripción: Esta tarea permitirá la autenticación del usuario.	

Tabla 8: Tarea de ingeniería “HU_1.1”

Tarea de Ingeniería	
Número Tarea: 1.2	**Número Historia de Usuario:** HU_1
Nombre Tarea: Implementación de la funcionalidad crear usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 5 días
Fecha Inicio: 1/1/2014	**Fecha Fin:** 7/1/2014
Programador Responsable: Juan Enidio Martínez Costa	
Descripción: Esta tarea permitirá la creación del usuario.	

Tabla 9: Tarea de ingeniería “HU_1.2”

2.5.4 Plan de Entregas

Es uno de los artefactos que genera la metodología SXP correspondiente, en la fase de desarrollo, al flujo de planificación. En él que se recogen las iteraciones a realizar con sus características, además del orden de las historias de usuario con su planificación estimada para ser implementadas. (Peñalver Romero, 2008)

Entrega	Descripción de la	Orden de la HU a	Duración total

	iteración	implementar	
Iteración 1	En esta iteración se desarrollarán las historias de usuario que tienen prioridad alta permitiendo al sistema la gestión de todo lo referente con los usuarios y sus actividades.	HU_1, HU_2, HU_3	4 semana
Iteración 2	En esta iteración se desarrollarán las historias de usuario que tienen prioridad baja permitiendo al sistema mostrar la información de los usuarios y sus actividades en forma de reporte.	HU_4	2 semana

Tabla 10: Plan de Entregas

2.6 Diseño del sistema

2.6.1 Diseño con metáforas

En SXP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella, explícitamente en el comienzo del proyecto, se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. La práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para

hablar sobre el dominio del problema. Este conjunto de nombres ayuda al diseño y métodos del sistema. (Peñalver Romero, 2008)

La metáfora definida para el sistema a desarrollar es: la implementación de un sistema para controlar la asistencia y permanencia de los trabajadores del centro CEGEL.

Basádonse en la metáfora definida, se diseña una solución de baja complejidad, funcional y de fácil implementación la cual incluye el diagrama de componentes.

Este diagrama muestra las dependencias lógicas entre los componentes del software, sean estos componentes: fuentes, binarios o ejecutables. Además permite modelar sistemas de software de cualquier tamaño y complejidad. En la Figura 3 se muestra el diagrama de componentes correspondiente a la propuesta de solución.

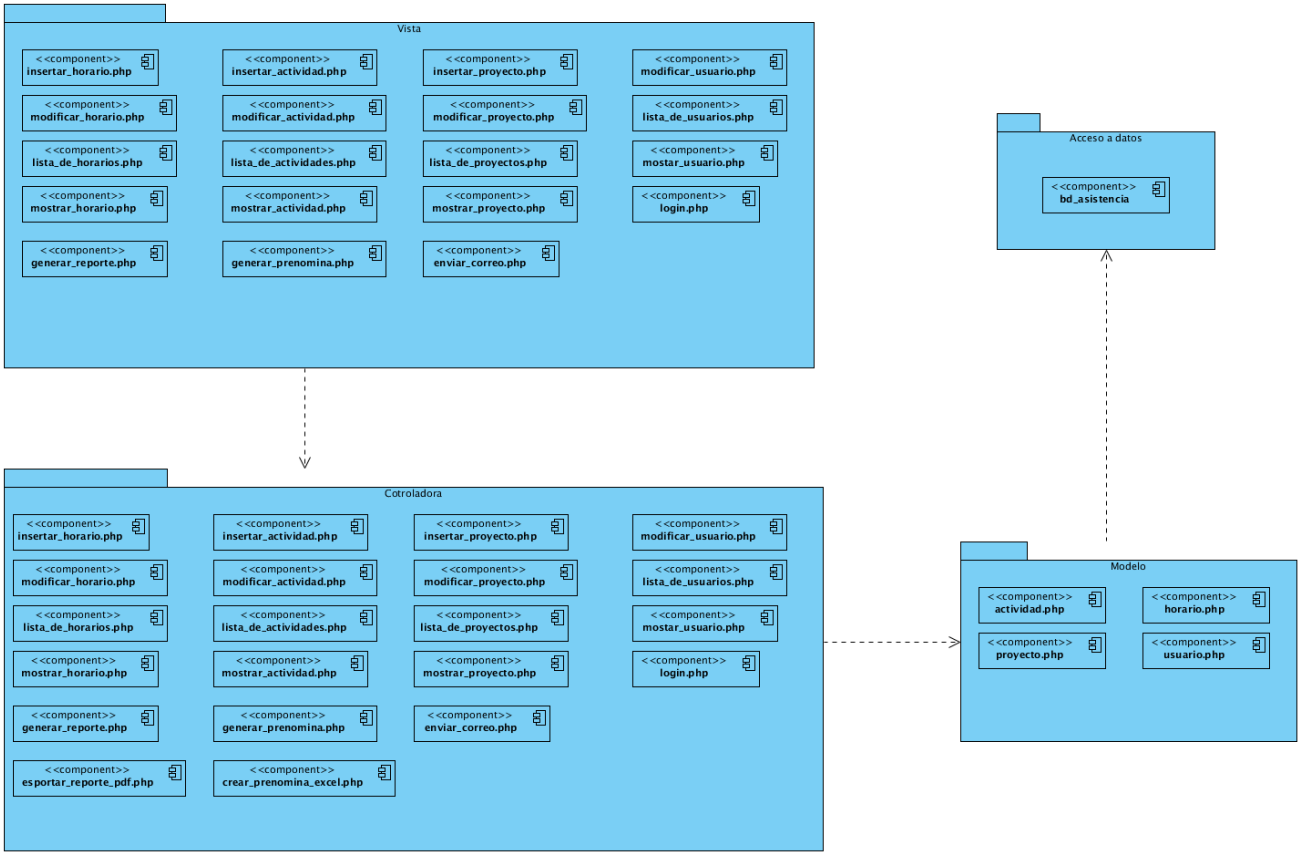

Figura 3: Diagrama de componentes

2.6.2 Estilos arquitectónicos utilizados

Arquitectura MVC (Modelo-Vista-Controlador)

El término MVC proviene de tres palabras que hoy en día se utilizan mucho dentro del ambiente de desarrollo de software: Modelo – Vista – Controlador. Esta arquitectura permite dividir las aplicaciones en tres capas:

- **Vista:** en la carpeta vista (view), localizada en en el directorio LdapBundle\Resources\view de la aplicación, es donde almacena el marco de trabajo symfony todas las vistas con las que contará el sistema, para eso se auxilia del motor de plantillas twig. Estas vistas tendrán la siguiente extensión “nombre.html.twig”. El sistema cuenta con esas vistas para las acciones que se realicen sobre entidades como los usuarios, los proyectos y las actividades

- **Modelo**: es el responsable de la conexión a la base de datos y la manipulación de los datos mismos. Esta capa está pensada para trabajar con los datos y obtenerlos, pero no mostrarlos, ya que la capa de presentación de datos es la Vista. En symfony para realizar estas acciones se utilizan las clases gestoras y los repositorios. En el sistema, se hace uso de las clases repositorios (EntityRepository), localizadas en el directorio LdapBundle\Entity. Estas tienen los métodos necesarios para su interacción con la base de datos de la aplicación.
- **Controlador:** su responsabilidad es procesar y mostrar los datos obtenidos por el Modelo. Es decir, este último trabaja de intermediario entre los otros dos, encargándose también de la lógica de negocio. Symfony se vale para realizar esta acción de las llamadas clases controladoras, las cuales se encuentran dentro de la carpeta Controller (controladora), localizada en el directorio LdapBundle\Controller, en la que se almacenan las clases controladoras del sistema. En ellas se van a encontrar las controladoras para el trabajo con las entidades como usuario, actividad y proyecto.

Arquitectura en capas

Se define el estilo en capas como una organización jerárquica, donde cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que

le brinda la inmediatamente inferior. Este estilo tiene algunos inconvenientes como es la recarga del nivel de aplicaciones. Es por ello que existe una fuerte y avanzada tendencia a adoptar la arquitectura en cuatro capas. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, solo se ataca al nivel requerido sin tener que revisar entre código mezclado.

Las cuatro capas que propone esta arquitectura son:

- ✓ Capa de presentación: es la que ve el usuario (hay quien la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.
- ✓ Capa de negocio: es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio), pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.
- ✓ Capa de datos: es donde residen los datos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes, de almacenamiento o recuperación de información, desde la capa de negocio.
- ✓ Capa de acceso a datos: es donde residen los ORM, los dispositvos y los servicios a través de los que acceden a la capa de datos, los cuales son motrados a la capa de negocio a través de la fachada de acceso a datos. (Departamento de Lenguajes y Sistemas Informáticos Universidad de Sevilla, 2008)

La arquitectura definida para la construcción del sistema es una arquitectura de cuatro capas, la cual está compuesta de la siguiente manera: la capa de datos, la capa de acceso a datos, la capa del negocio y la capa de presentación. Entre las capas de negocio y presentación se aplica el patrón Modelo-Vista-Controlador (MVC). En la Figura 5 se muestra la arquitectura de la aplicación.

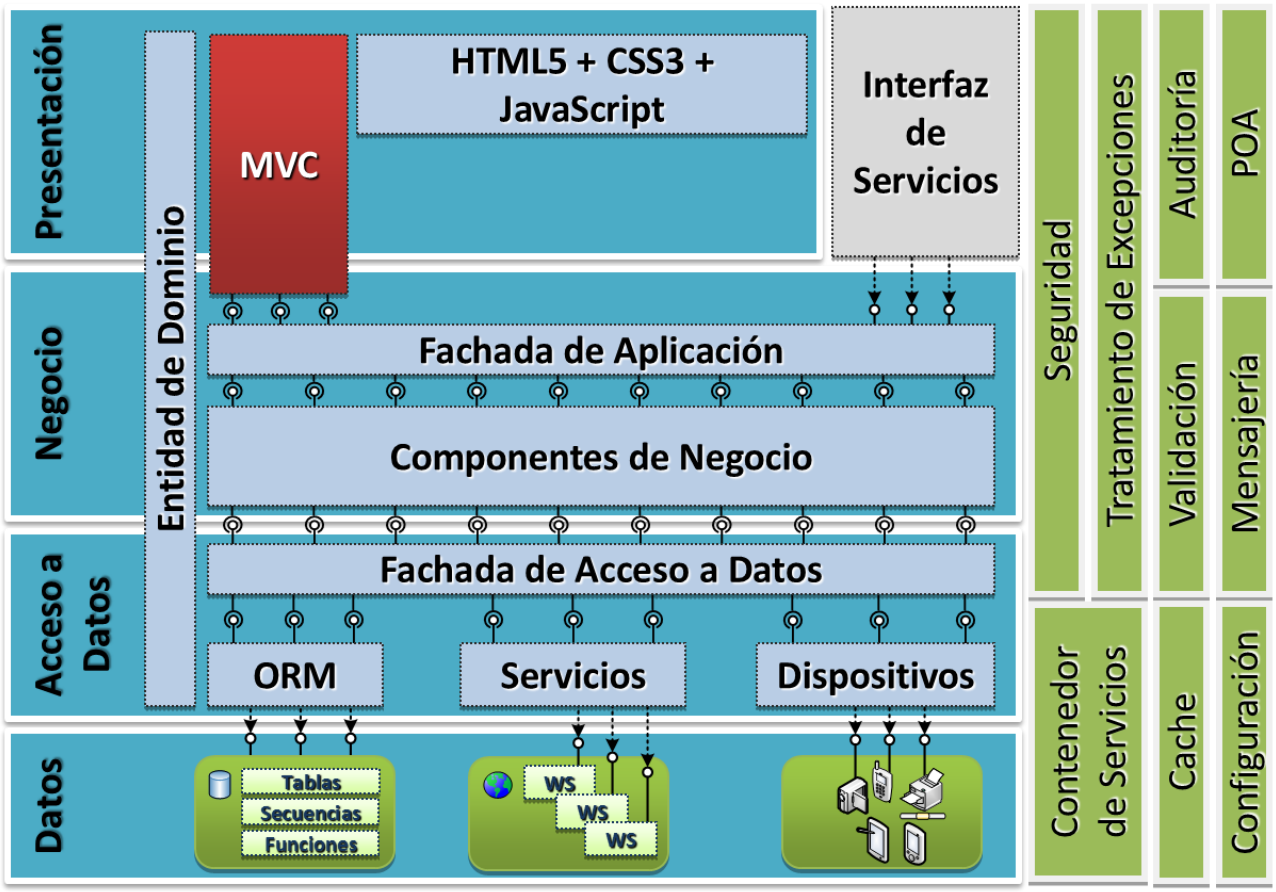

Figura 4: Arquitectura del sistema

2.6.3 Patrones de diseño

Un patrón es una descripción de un problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos. (Larman, 2011)

Patrones GRASP

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

Controlador: delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. No realiza mucho trabajo por sí mismo. Symfony utiliza este patrón para todas sus clases controladoras, el mismo contiene todos los métodos necesarios para controlar las actividades que se realicen. Este patrón se ve reflejado en las entidades controladoras.

En la Figura 6 se evidencia el uso de este patrón para la clase “ProyectoController.php”, en ella se encuentran métodos como newAction() y editAction(), encargados de crear y editar los proyectos respectivamente, delegando la función de crear los formularios para la misma a los métodos createCreateFrom y createEditForm respectivamente.

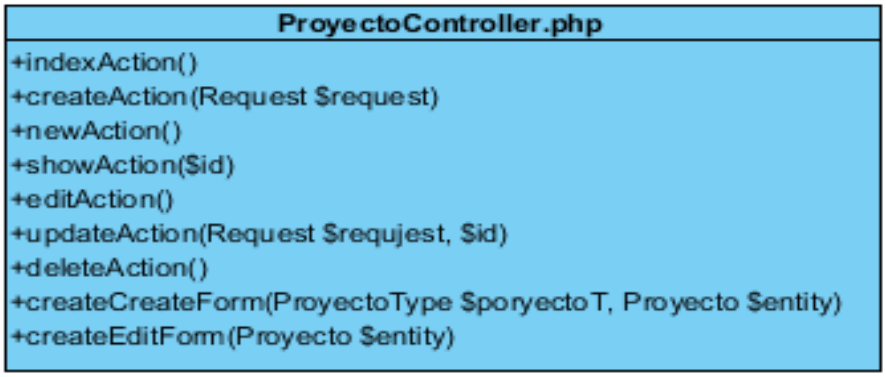

Figura 5: Clase controladora “ProyectoController.php”

Experto en Información: se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. El Experto expresa la "intuición" común de que los objetos hacen las cosas relacionadas con la información que tienen. Este patrón se ve reflejado en las entidades debido a que cada una de ellas es experta, pues contienen la información necesaria para cumplir con las responsabilidades que le fueron asignadas. En la Figura 7 se evidencia el uso de este patrón para la clase proyecto, la misma contiene datos como el nombre y el lab del proyecto, los cuales son necesarios para que el controlador a través de los métodos pertinentes (insertar, modificar, eliminar, visualizar y listar) pueda realizar las acciones necesarias sobre los mismos.

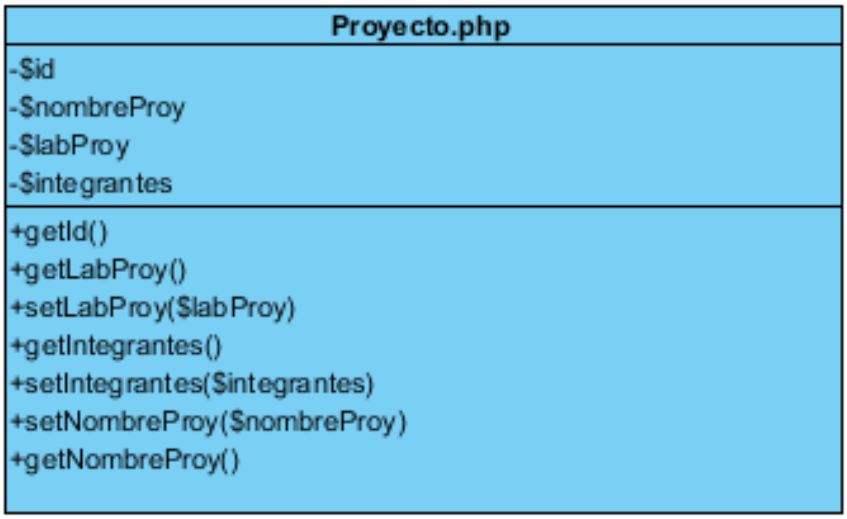

Figura 6: Clase experta en información "Proyecto.php"

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica de este patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Es el encargado de crear las instancias de los objetos para cada una de las funcionalidades de la aplicación. En la Figura 8 se evidencia el uso de este patrón para la clase “ProyectoController.php”, esta clase contiene los métodos createCreateForm() y createEditForm(), a los mismos se le asignan la responsabilidades de crear y editar un formulario de la instancia proyecto en la vista de la aplicación donde se vayan a mostrar estas acciones.

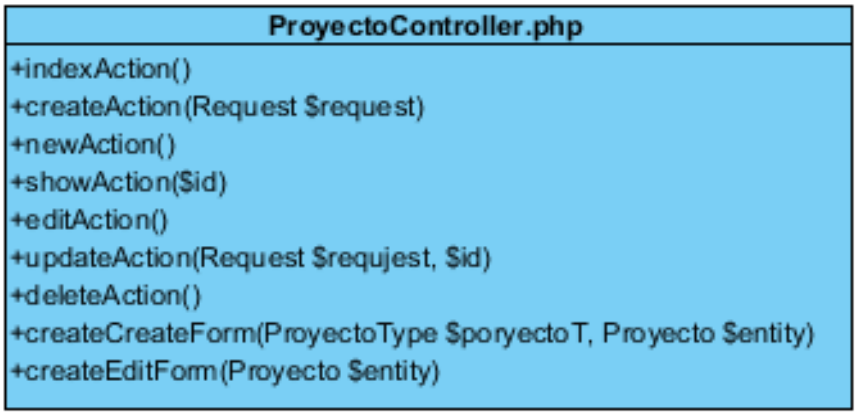

Figura 7: Clase creadora "ProyectoController.php" encargada de crear una instancia de proyecto

Alta Cohesión: es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño. Se evidencia debido a que a cada una de las clases se le asignaron responsabilidades de tal forma, que estén estrechamente relacionadas entre sí y no realicen un trabajo excesivo. En la Figura 9 se evidencia el uso de este patrón, pues la clase ProyectoController cuenta con los métodos necesarios para controlar las actividades sobre los proyectos. A su vez la clase Proyecto contiene la información necesaria para crear el Proyecto y la clase ProyectoRepository es la que se encarga de realizar las acciones en la base de datos sobre ese proyecto.

Bajo Acoplamiento: es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño. Proporciona un bajo acoplamiento en el diseño debido a que las clases existentes tienen asignadas responsabilidades de tal forma que estas no dependan en gran medida de otras, permitiendo de esta forma tener sistemas más robustos y de fácil mantenimiento. (Larman, 1999). En la Figura 9 se evidencia el uso de

este patrón pues estas clases solo hacen las funciones para las que fueron creadas. La clase ProyectoControler para controlar las acciones que se realicen sobre el proyecto, la clase ProyectoRepository es la encargada de gestionar las acciones hacia la base de datos y la clase Proyecto es la que tiene la información necesaria para que la clase ProyectoRepositoy pueda realizar las acciones sobre la base de datos.

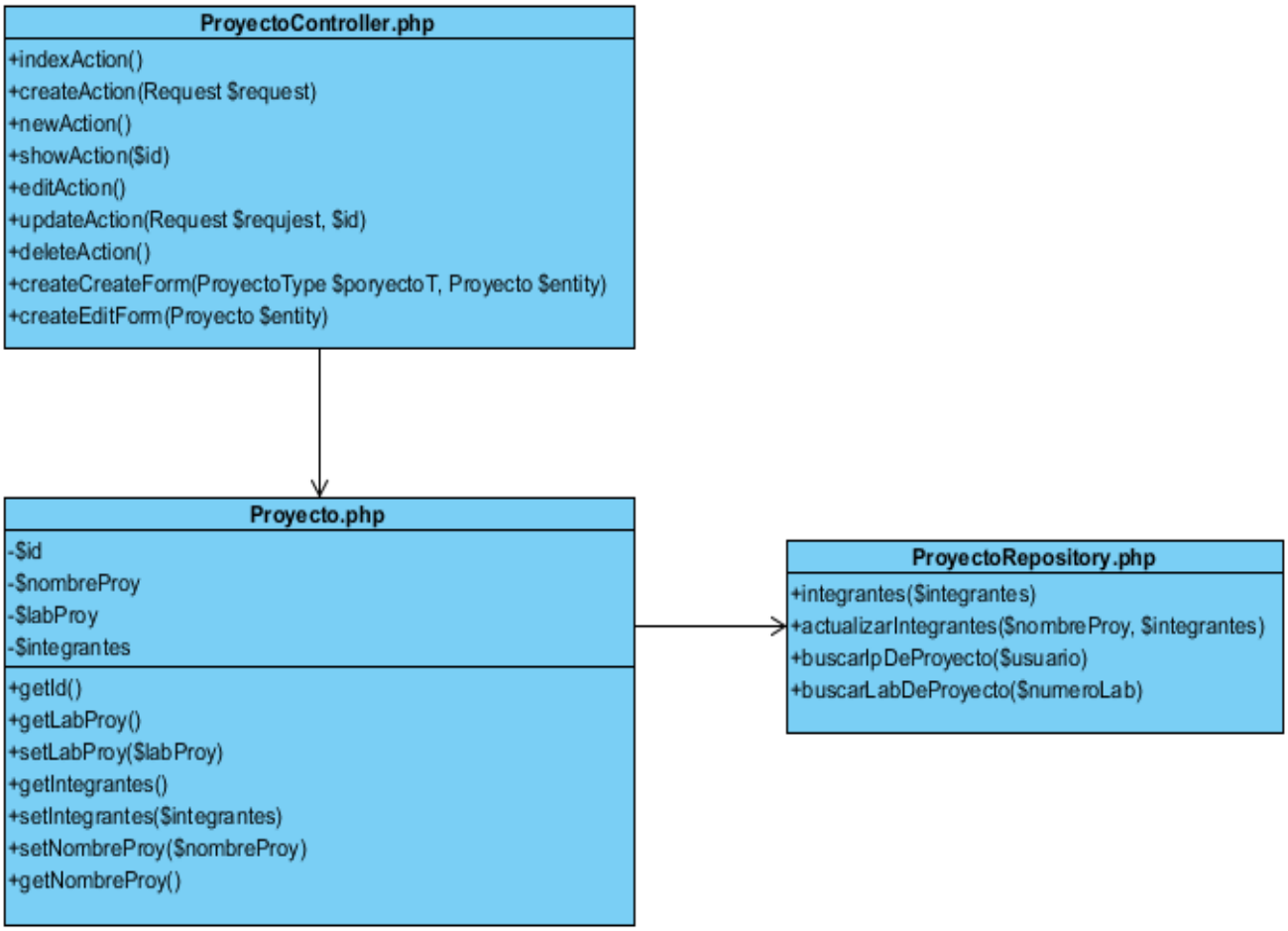

Figura 8: Presencia del patrón bajo acoplamiento y alta cohesión entre las entidades ProyectoController.php, Proyecto.php y ProyectoRepository.php

Patrones GoF

Decorador: añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. Es el archivo base.html.twig, que también se denomina plantilla global, localizado en el directorio Ldap\\app\Resoruces\views, el cual almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que

repetirlo en cada página. Para realizar esta actividad se auxilia de la herencia, así todas las plantillas que se definan para el sistema heredaran de la plantilla principal.

Fábrica abstracta: proporciona una interfaz para la creación de familias de objetos interdependientes o interrelacionados, sin especificar sus clases concretas. Cuando el marco de trabajo symfony2 necesita, por ejemplo, crear un nuevo objeto, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea. (Unidad Docente de Ingeniería del Software, 2000). Symfony lo implementa en muchos métodos que vienen definido con este marco de trabajo, uno de ellos es el metodo getRepository, el cual se encarga de realizar las consultas sobre las clases entidades hacia la base de datos de la aplicación. En el siguiente fragmento de código se puede observar el método getRepository, el cual se usa para acceder a todos los usuarios y proyectos que existen en la base de datos del sistema.

```
public function reportePdfAction() {

    $em = $this->getDoctrine()->getEntityManager();

    $usuarios = $em->getRepository('LdapBundle:Usuario')

        ->findAll();

    $proyectos = $em->getRepository('LdapBundle:Proyecto')

        ->findAll();

    return $this->render('LdapBundle:Reporte:reportePDF.html.twig', array('usuarios' => $usuarios, 'proyectos' => $proyectos));

  }
```

2.6.4 Estándar de codificación empleado

El sistema a desarrollar tributará a un mejor desempeño del control de la asistencia para los trabajadores del centro CEGEL, el mismo podrá ser actualizado y mejorado con el paso del tiempo por las personas que se encarguen del mantenimiento y el control del mismo. Para ello se hace factible el empleo de un estándar de código, pues las personas que lo actualicen o les realicen mejoras, no serán los mismos que un principio lo implementaron y

necesitan tener la mayor claridad posible de lo que realiza cada línea de código de la aplicación.

Durante el desarrollo de la aplicación se emplearon los estándares de codificación del Sistema de Informatización para la Gestión de los Tribunales Populares Cubanos (SITPC). En el mismo se resumen las siguientes reglas de codificación.

Identación

El contenido siempre se identará con tabs, nunca utilizando espacios en blanco.

Comentarios en las funciones

Todas las funciones deben tener un comentario, antes de su declaración, explicando qué hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben bastar para ello.

Nombres de variables

Los nombres deben ser descriptivos y concisos. No usar ni grandes frases ni pequeñas abreviaciones para las variables. Esto aplica para los nombres de variables, funciones, argumentos de funciones y clases. Los nombres de las variables y de las funciones inician con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula. Las constantes deben escribirse siempre en mayúsculas, y tanto estas como las variables globales, deben tener como prefijo el nombre de la clase a la que pertenecen.

Las Definiciones de la función

Los nombres de la función pueden contener solo carácteres alfanuméricos. Los nombres de la función siempre deben empezar en letras minúsculas. Cuando un nombre de la función consiste de más de una palabra, la primera letra de cada nueva palabra debe capitalizarse.

Llamadas a funciones

Deben llamarse las funciones sin los espacios entre el nombre de la función, el paréntesis de la apertura, y el primer parámetro; los espacios entre las comas y cada parámetro, y ningún espacial entre el último parámetro, el paréntesis del cierre, y el punto y coma.

Poner espacios entre signos

Si se tiene un signo y operador binario, se colocan espacios a ambos lados. Si se tiene un signo unario, se colocan espacios a uno de sus lados. Este elemento es algo muy sencillo que ayuda a la legibilidad del código.

Precedencia de operadores

Lo mejor es siempre usar paréntesis para estar seguro de la precedencia de los operadores.

```
$bool = (($i < 7) && (($j < 8) || ($k == 4)));

//Incluso mejor

$bool = ($i < 7 && ($j < 8 || $k == 4));
```

No utilizar variables sin inicializar

Si no se tiene control sobre el valor de una variable, se debe verificar que esté inicializada. Esto lo permite PHP de la siguiente manera:

```
if (isset($cliente) && $cliente == 5) ...
```

Pero solo se debe usar esta opción cuando no se tenga el control o no se esté seguro del valor que pueda tener la variable (como en variables que llegan por parámetro o por GET).

Conclusiones parciales

En el capítulo que concluye se presentaron elementos necesarios para lograr eficazmente la implementación del sistema, dentro de las tareas que se cumplieron se puede concluir que:

- Los requisitos definidos y especificados dan respuesta a las necesidades del cliente.
- Se documentaron una serie de artefactos que ayudaron a manejar la inestabilidad de los requisitos.
- La arquitectura definida brindó un estilo de programación eficiente, separando la lógica del negocio de la lógica del diseño.
- Se definieron los estándares de codificación a utilizar, lográndose así que el código quedase estructurado y legible, asegurándose de esta manera una mayor reutilización del mismo.

CAPÍTULO 3: VALIDACIÓN

Introducción

Actualmente el desarrollo de un software implica una serie de actividades en las cuales es común cometer errores, los cuales pueden comenzar desde el principio del proceso, con un planteamiento erróneo de los requisitos o durante los procesos de diseño e implementación. Debido a la incapacidad humana de comunicarse y realizar tareas de manera perfecta, es necesario llevar a cabo actividades que garanticen la calidad. Es por ello que el presente capítulo es dedicado a comprobar mediante métricas y técnicas los resultados obtenidos.

3.1 Validación de los requisitos

Con el objetivo de ratificar que los requisitos del software obtenidos definen el sistema que se desea, se llevó a cabo un proceso de validación de los mismos, para el cual se emplearon las técnicas de: matriz de trazabilidad, la métrica calidad de la especificación y la construcción de prototipos.

3.1.1 Construcción de prototipos

Los prototipos permiten la comunicación y participación entre el equipo de desarrollo y el cliente, haciendo de las especificaciones una herramienta fundamental para comprobar los requisitos del software.

La construcción de estos, es una alternativa para validar los requisitos funcionales que fueron capturados durante la fase de Planificación-Definición, definida por la metodología seleccionada. Para la construcción de la interfaz gráfica se hizo uso de la herramienta Visual Paradigm en su versión 8.0.

Con el resultado de esta actividad se obtuvo un diseño de la aplicación, lo cual permitió tener una visión preliminar de lo que se quería lograr con el sistema a desarrollar.

Se modelaron prototipos para todos los requisitos definidos en las historias de usuario, arrojando un total de 31 prototipos. En la Figura 10 se muestra el prototipo “Crear actividad del usuario”, perteneciente a la historia de usuario “Gestionar actividad”.

Figura 9: Crear actividad del usuario

3.1.2 Matriz de trazabilidad

La matriz de trazabilidad es una técnica que tiene como objetivo verificar si todas las partes del producto desarrollado se pueden identificar o relacionar con cada uno de los requisitos definidos en la fase de análisis, es decir, si a partir de cualquier elemento del software se puede llegar hasta los requisitos o viceversa.

Mediante la matriz de trazabilidad se pudo comprobar que los requisitos definidos en el sistema eran los mismos que estaban reflejados en las historias de usuario. A continuación se presenta la Tabla 11 correspondiente a la matriz de trazabilidad del sistema.

Requisitos / HU	Gestionar usuario	Gestionar actividad	Gestionar proyecto	Gestionar horario	Gestionar reporte	Gestionar asistencia
1	x					
2	x					

3	x					
4	x					
5	x					
6		x				
7		x				
8		x				
9		x				
10			x			
11			x			
12			x			
13			x			
14				x		
15				x		
16				x		
17				x		
18					x	
19					x	
20					x	
21					x	
22					x	
23					x	
24					x	

25					x	
26					x	
27					x	
28					x	
29					x	
30						x
31						x

Tabla 11: Matriz de trazabilidad

3.1.3 Métrica para calidad de la especificación

Para realizar la validación de los requisitos existe una lista de características que sugieren el uso de la misma: especificidad (ausencia de ambigüedad), corrección, comprensión, capacidad de verificación, consistencia externa e interna, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización. (Pressman, 2001)

Especificidad

Para llevar a cabo este proceso se tiene que: **Nr** representa el número de requisitos del sistema.

Nr = Nf+ Nnf

Nr = 31+ 11

Nr = 42

Donde **Nf** es el número de requisitos funcionales y **Nnf** es el número de requisitos no funcionales. Para determinar la especificidad (ausencia de ambigüedad) de los requisitos, se sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito.

Q = Nui/ Nr

Donde **Nui** es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas. El valor de **Q** a medida que se acerca a 1, va disminuyendo la ambigüedad de la especificación.

Con el objetivo de obtener la menor ambigüedad posible de los requisitos, de modo que satisfagan las necesidades de los clientes, se llevaron a cabo un total 2 revisiones, que fueron suficientes para alcanzar el resultado deseado.

En la primera revisión se identificaron algunos requisitos funcionales y no funcionales que presentaban problemas de redacción, de ortografía e incoherencia entre las ideas. Para un total de 42, los revisores tuvieron la misma interpretación para 36 de ellos.

Q = 36/ 42

Q = 0.86

En la segunda revisión, ya corregidos los errores identificados en la primera, los revisores tuvieron la misma interpretación para el total de requisitos. En la Figura 11 se muestra una gráfica resumen con los resultados obtenidos.

Q = 42/ 42

Q = 1

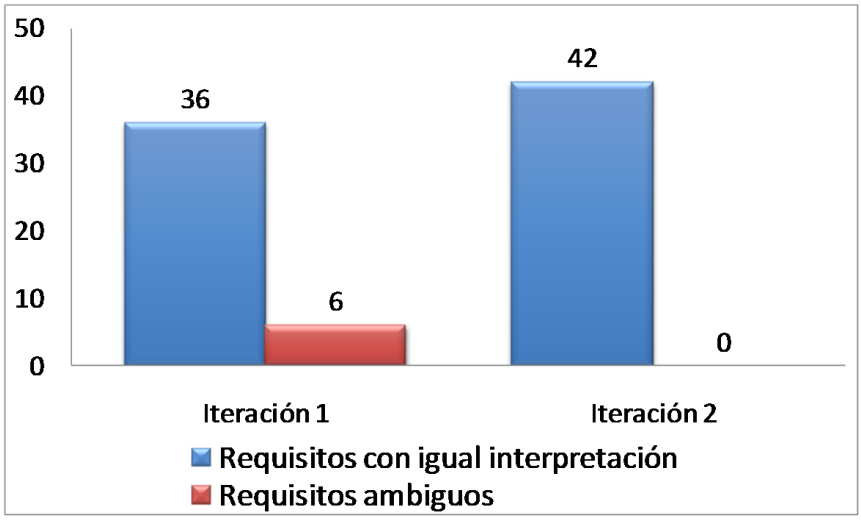

Figura 10: Gráfica de comparación entre la primera y la segunda revisión en la especificidad

Al concluir el estudio cuantitativo, realizado a los resultados obtenidos en cada una de las revisiones, se pudo evidenciar que los requisitos presentan un grado bajo de ambigüedad, lo que permite trabajar de forma más segura con ellos.

1.3 Validación de las Historias de Usuario

Generación de casos de prueba

Como parte del proceso de evaluación de los requisitos funcionales de la aplicación, fueron diseñados casos de prueba para cada historia de usuario, utilizando las plantillas de casos de prueba de aceptación generadas por el Programa de mejoras de CALISOFT. En ellas el desarrollador escribe las pruebas realizadas, según la historia de usuario seleccionada, para validar las funcionalidades del sistema y de esta forma saber si está apto para ser liberado. En la Tabla 12 se muestra el caso de prueba de aceptación correspondiente a la historia de usuario ¨Gestionar actividad¨.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU_1-1	**Nombre Historia de Usuario:** Gestionar actividad.
Nombre de la persona que realiza la prueba: Yelena Hernández Estrada	
Descripción de la Prueba: Se realizará la prueba a la funcionalidad crear actividad.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.	
Entrada / Pasos de ejecución: 1-Se selecciona la opción “Crear actividad”. 2-Se introducen los datos necesarios para crear la actividad: nombre, usuarios, hora de inicio, hora de fin y fecha.	
Resultado Esperado: 1-El sistema debe mostrar un formulario solicitando los datos necesarios para añadir la actividad. 2- El sistema debe mostrar un mensaje indicando si la acción se ha realizado correctamente o no, en caso de ser negativa la respuesta del sistema, debe especificar en el mensaje los motivos por los que no se pudo realizar la	

acción. El sistema debe mostrar el listado de todos las actividades existentes, si se realizó la acción satisfactoriamente se debe incluir en el listado la actividad añadida.
Evaluación de la Prueba: Satisfactoria

Tabla 12: Caso de prueba de aceptación “Crear actividad”.

1.4 Validación del diseño

Métrica Tamaño Operacional de Clase (TOC)

Para comprobar la calidad del diseño del sistema se empleó la métrica Tamaño Operacional de Clase (TOC). Esta métrica permite medir la responsabilidad, la complejidad de implementación y la reutilización de las clases del diseño. Es importante destacar que para esta métrica, la responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que a mayor responsabilidad y complejidad de implementación de una clase, menor será su nivel de reutilización.

Para la aplicación de esta métrica se deben de conocer los atributos a medir (responsabilidad, complejidad, reutilización) y las categorías (alta, media, baja) en las que se puede clasificar cada clase según los resultados obtenidos luego de aplicar el criterio de clasificación. En la Tabla 13 se muestran los atributos, las categorías y los criterios que se definen para la aplicación de esta métrica.

Atributos	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Tabla 13: Atributos, categorías y criterios definidos para la aplicación de la métrica tamaño operacional de clase.

La variable *Prom* definida en la tabla anterior, representa el promedio de las clases. Esta se obtiene a partir de la división de la cantidad de procedimientos de las clases entre el total de clases. En la tabla 14 se muestran los resultados obtenidos luego de aplicar la métrica a las clases del sistema.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
UsuarioController	8	Baja	Baja	Media
HorarioController	11	Media	Media	Media
ActividadController	11	Media	Media	Media
ProyectoController	15	Media	Media	Media
ReporteController	6	Baja	Baja	Alta
PrincipalController	3	Baja	Baja	Alta
DefaultController	7	Baja	Baja	Alta
Usuario	22	Alta	Alta	Baja
Horario	18	Baja	Baja	Media
Actividad	16	Baja	Baja	Media
Proyecto	9	Baja	Baja	Baja
Entrada	12	Media	Media	Media

Tabla 14: Resultados obtenidos por la métrica para las clases del sistema.

En las Figuras 12, 13 y 14 se muestran los resultados obtenidos en porciento (%) durante la aplicación de la métrica en cuanto a la responsabilidad, la complejidad y la reutilización de los procedimientos de las clases en el sistema.

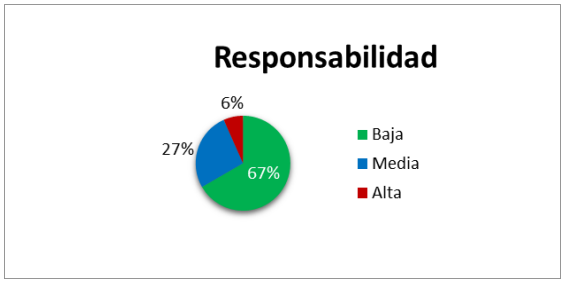

Figura 12: Representación en porciento (%) del nivel de responsabilidad de las clases

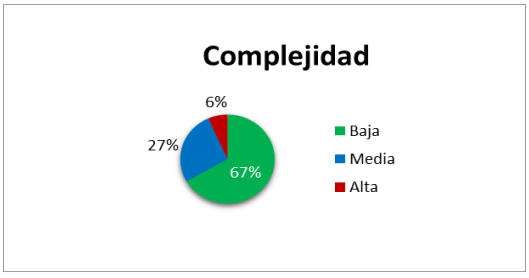

Figura 13: Representación en porciento (%) del nivel de complejidad de las clases

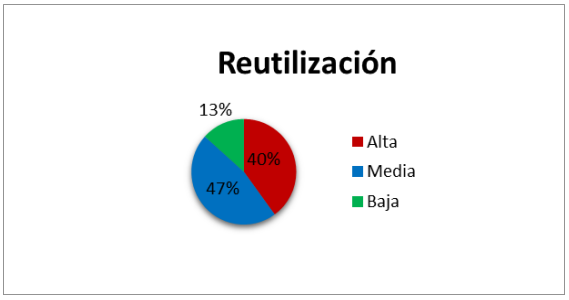

Figura 4: Representación en porciento (%) del nivel de reutilización de las clases

Luego de aplicada la métrica se observa que las clases de la aplicación no se encuentran sobrecargadas en cuanto a responsabilidades y el nivel de complejidad de las mismas no es muy alto, lo que favorece en gran medida la reutilización de estas.

1.5 Pruebas del sistema

Según (Pressman, 2001) las pruebas son un instrumento adecuado para determinar el estado de la calidad de un producto, en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos especificados. Estas no tienen el objeto de prevenir errores sino de detectarlos. Se efectúan sobre el trabajo realizado y se deben encaran con la intención de descubrir la mayor cantidad de errores posible.

Aunque no hay una clasificación oficial o formal acerca de los diversos tipos de pruebas de software, existen dos enfoques fundamentales:

- Pruebas de tipo caja negra: cuando una aplicación es probada usando su interfaz externa.
- Pruebas de tipo caja blanca: cuando una aplicación es probada desde dentro, usando su lógica aplicativa.

Pruebas de caja negra

Las pruebas de caja negra se centran en los requisitos funcionales del software e intenta encontrar errores de los siguientes tipos:

- Funciones incorrectas o inexistentes
- Errores relativos a las interfaces

- Errores en estructuras de datos o en accesos a bases de datos externas
- Errores debidos al rendimiento
- Errores de inicialización o terminación

Pruebas realizadas

Se realizó un plan de pruebas para las historias de usuario definidas en el sistema. A continuación se describen las pruebas y los resultados que arrojaron las mismas.

✓ Pruebas de carga y stress

Esta prueba se realizó con el objetivo de estudiar la velocidad de respuesta ante las peticiones de los usuarios, dependiendo de la cantidad de trabajo del sistema. Para ello se utilizó la herramienta Jmeter, esta herramienta permite comprobar la carga y stress para un número X de usuarios, simulando lo que pasaría en tiempo real para estos. En él se graba el flujo a probar y después se ejecuta para una cantidad definida de usuarios, dando un tiempo de respuesta para la misma.

En la Tabla 15 se muestran los resultados obtenidos después de aplicar la prueba a la funcionalidad actualizar usuario. El campo carga de trabajo se refiere a la cantidad de usuarios para los que se va a aplicar la prueba. Se tomaron 500 usuarios, pues el sistema tiene que soportar un gran número de usuarios simultáneamente.

ID del escenario	Escenarios de la sección	Carga de trabajo	Descripción	Resultado esperado	Resultado de la prueba
EC2-Actualizar usuario	EC 2.1 Usuario Para ello ejecutar clic en la pestaña “Usuario”.	500	Esta interfaz muestra una lista con todos los usuarios existentes en la base de datos del sistema.	La aplicación ha respondido a una velocidad menor que “27.5 seg”	15.9 seg
	EC 2.2 Lista de usuarios Para ello	500	Esta interfaz permite modificar los atributos del	La aplicación ha respondido a	15.9 seg

	ejecutar clic en la opción modificar perteneciente a el select “seleccione” del usuario a modificar.		usuario.	una velocidad menor que “27.5 seg”	

Tabla 15: Prueba de carga y stress realizada a la funcionalidad ¨Actualizar usuario¨

✓ Pruebas de seguridad y acceso

Esta prueba se realizó con el objetivo de verificar la seguridad a nivel de aplicación (que el usuario solo pueda acceder y modificar los datos que le correspondan) y a nivel de sistema (que solo puedan acceder a la aplicación los usuarios con permisos adecuados). Para la misma se comprobó el funcionamiento del sistema de autenticación de la aplicación para los distintos usuarios y roles. Además, se comprobó los permisos que tiene el usuario en el sistema hacia los cuales puede acceder, verificando en cada caso su aprobación en caso de contar con los permisos adecuados o su negación en caso de no contar con los permisos adecuados. En la Tabla 16 se muestra la prueba realizada y los resultados arrojados por la misma.

Nombre del Rol	**Nombre Escenario 2** **EC 1.2: Crear actividad** **Resultado esperado**	**Resultado de la prueba**
Usuario	Se muestra un formulario con los campos de la actividad a crear. Esta actividad se creara solo para ese usuario.	La prueba se cumplió satisfactoriamente, pues al usuario solo se le permite crear la actividad para él y no para otros usuarios.
Chequeador	Se muestra un formulario con los campos de la actividad a crear. Esta actividad se creara solo para ese usuario.	La prueba se cumplió satisfactoriamente, pues al usuario solo se le permite crear la actividad para él y no para otros usuarios.
Administrador	Se muestra un formulario con los campos de la actividad a crear. Este formulario tendrá además de las campos definidos otro campo adicionar en el cual el Administrador podrá insertar los usuarios para los	La prueba se cumplió satisfactoriamente, pues al usuario se le permite crear la actividad para él y para los usuarios que decida.

	cuales será la actividad.	
Descripción de la funcionalidad a probar	Permite la creación de la actividad.	

Tabla 16. Prueba realizada a la funcionalidad ¨Crear actividad¨

Resultados de las pruebas realizadas

Una vez aplicadas estas pruebas al software, se pudo comprobar que el sistema implementado cuenta con una buena seguridad, pues además de definir los roles necesarios para el acceso de los usuarios, se definen aquellas áreas o lugares dentro de la aplicación a los cuales tendrá acceso el usuario con los permisos necesarios para la misma. Además se cuenta con un sistema que puede soportar un gran número de usuarios conectados simultáneamente, factor fundamental para el buen desempeño de la aplicación, pues en el día deben acceder a la misma varias personas, las cuales pueden coincidir muchas a la misma hora en el sistema.

Pruebas de caja blanca

Para comprobar que las funciones internas de la aplicación se ejecutan correctamente, se realizaron pruebas al código de las principales funcionalidades que componen el sistema. Para ello se empleó la técnica de la prueba del camino básico, el cual consiste en garantizar que se pueda probar al menos una vez cada una de las sentencias del programa. Se parte de la obtención de la medida de la complejidad de un procedimiento o algoritmo y la obtención de un conjunto básico de caminos de ejecución de este, los cuales son utilizados para obtener los casos de prueba. Seguidamente se muestra el proceso de pruebas realizado a la funcionalidad principalAction() la cual se encarga de restringir la entrada de los usuarios dependiendo del lugar, así como de registrar el horario de entrada.

```
public function principalAction() {

    $em = $this->getDoctrine()->getEntityManager();  //1

    $datosUsuarioConectado = $this->get('security.context')->getToken()->getUser();  //1

    $usuario = $datosUsuarioConectado->getUserName();  //1
```

```
$correo = $datosUsuarioConectado->getRoles();  //1
$datos = new \SoapClient("https://autenticacion2.uci.cu/v6/PasarelaAutenticacionWS.wsdl");  //1
$datosUsuario = $datos->ObtenerPersonaDadoUsuario($usuario);  //1
$consulta = $em->getRepository('LdapBundle:Usuario')
    ->findOneBy(array('usuario' => '' . $usuario . ''));  //1
if (empty($consulta)) {  //2
    $idUsuario = new Usuario();  //3
    $idUsuario->setUsuario($usuario);  //3
    $idUsuario->setNombre($datosUsuario->Nombres);  //3
    $idUsuario->setApellidos($datosUsuario->Apellidos);  //3
    $idUsuario->setCorreo($datosUsuario->Correo);  //3
    $idUsuario->setCategoria(null);  //3
    $idUsuario->setProyecto(null);  //3
    $idUsuario->setRolProyecto(null);  //3
    $idUsuario->setRol(null);  //3
    $idUsuario->setCargo($datosUsuario->Cargo->NombreCargo);  //3
    $idUsuario->setIdExpediente($datosUsuario->IdExpediente);  //3
    $em->persist($idUsuario);  //3
    $em->flush();  //3
    $consulta1 = $em->getRepository('LdapBundle:Usuario')
        ->findOneBy(array('usuario' => '' . $usuario . ''));  //3
    $idEntrada = new Entrada();  //3
```

```
            $idEntrada->setHoraEnt(date("H:i:s"));  //3
            $idEntrada->setHoraSal(null);  //3
            $idEntrada->setFecha(date("d/m/Y"));  //3
            $idEntrada->setUsuario($consulta1->getId());  //3
            $em->persist($idEntrada);  //3
            $em->flush();  //3
        } else {
            $idEntrada = new Entrada(); //4
            $idEntrada->setHoraEnt(date("H:i:s"));  //4
            $idEntrada->setHoraSal(null);  //4
            $idEntrada->setFecha(date("d/m/Y"));  //4
            $idEntrada->setUsuario($consulta->getId());  //4
            $em->persist($idEntrada);   //4
            $em->flush();  //4
    }
            $consultaProy = $em->getRepository('LdapBundle:Proyecto')  //5
                    ->findOneBy(array('integrantes' => '' . $usuario . ''));
            if ($this->getRequest()->getClientIp() >= $consultaProy->getRangoIpIni() || $this->getRequest()->getClientIp() <= $consultaProy->getRangoIpFin()) {  //6
                return $this->render('LdapBundle:Error:error.html.twig');  //7
            }
            $consultaHorario = $em->getRepository('LdapBundle:Horario')
                    ->findOneBy(array('integrantes' => '' . $usuario . ''));  //8
```

```
        if ((date("H:i:s") >= $consultaHorario->getHoraEntMan() || date("H:i:s") <= $consultaHorario->getHoraSalMan())

            || (date("H:i:s") >= $consultaHorario->getHoraEntTar() || date("H:i:s") <= $consultaHorario->getHoraSalTar())) {  //9

            return $this->render('LdapBundle:Error:error.html.twig');  //10

        }

    }

    return $this->render('LdapBundle:Portada:portada.html.twig');  //11

}
```

Para obtener los casos de prueba a partir de la técnica seleccionada se debe construir el grafo de flujo correspondiente al código de la función, el mismo se muestra en la Figura 16.

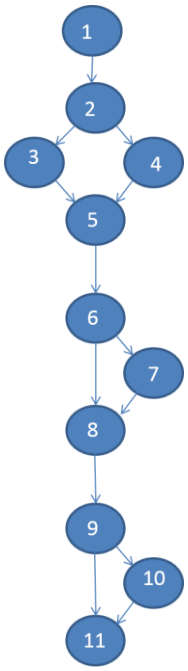

Figura 11: Grafo del flujo del código de la función PrincipalAction()

Luego se determina la complejidad ciclomática V (G) del grafo resultante, la cual es un indicador del número de caminos independientes que existen en un grafo, es decir, es cualquier camino dentro del código, que introduce como mínimo, un nuevo conjunto de sentencias de proceso o una nueva condición. La complejidad ciclomática puede ser calculada de 3 formas:

1. V (G) = a - n + 2, siendo *a* el número de arcos o aristas del grafo y *n* el número de nodos.
2. V (G) = r, siendo *r* el número de regiones cerradas del grafo.
3. V (G) = c + 1, siendo *c* el número de nodos de condición.

Realizando los cálculos correspondientes se obtiene para cualquiera de las variantes el siguiente resultado:

1. V (G) = a - n + 2	V (G) = r	V (G) = c + 1
V (G) = 13 – 11 + 2	V (G) = 4	V (G) = 3 + 1
V (G) = 4		V (G) = 4

Por lo que el conjunto de caminos básico sería:

Camino básico 1: 1-2-4-5-6-7-8-9-10-11

Camino básico 2: 1-2-3-5-6-7-8-9-10-11

Camino básico 3: 1-2-4-5-6-8-9-11

Camino básico 4: 1-2-3-5-6-8-9-11

Luego se definen los casos de prueba para cada uno de los caminos básicos obtenidos. En la tabla 17 y 18 se muestran los resultados de las pruebas aplicadas a los caminos básicos 3 y 4 respectivamente.

Descripción: se verificará que cuando el usuario se autentique en el sistema, acceda desde una ubicación permisible para la misma.		
Condiciones de ejecución:	**Entrada:**	**Resultados esperados:**
El usuario debe de estar creado en la base de datos y debe de tener asignado un proyecto.	Formulario de acceder al sistema.	Si el usuario no pertenece al laboratorio del proyecto desde el cual está accediendo a la aplicación, se le debe de re-direccionar a una página de error donde

		se le diga que está accediendo desde una ubicación no válida.
Resultado obtenido: Satisfactorio		

Tabla 17: Caso de prueba para el camino 3

Descripción: se verificará que cuando el usuario se autentique en el sistema, acceda en el horario permisible para la misma.		
Condiciones de ejecución:	**Entrada:**	**Resultados esperados:**
El usuario debe de estar creado en la base de datos.	Formulario de acceder al sistema.	Si la hora de entrada es distinta al horario de jornada laboral, se debe de especificar en un campo la actividad que se va a realizar.
Resultado obtenido: Satisfactorio		

Tabla 18: Caso de prueba para el camino 4

Conclusiones parciales

En el capítulo que concluye se expusieron las técnicas, herramientas y pruebas necesarias para validar la aplicación, dentro de las tareas que se cumplieron se puede concluir que:

- La métrica aplicada para validar el diseño permitió medir la responsabilidad, la complejidad de implementación y la reutilización de las clases, arrojando como resultado que las mismas no se encuentran sobrecargadas en cuanto a responsabilidades y que el nivel de complejidad no es muy alto, lo que favorece en gran medida la reutilización de estas.
- Las pruebas realizadas para validar las historias de usuario permitió comprobar que los flujos para cada una de ellas se desarrollaban de la forma en que estaban definidos.

CONCLUSIONES GENERALES

Con el fin de dar cumplimiento al objetivo general del presente trabajo se definieron las tareas de la investigación, con las que se arribó a las siguientes conclusiones:

- ✓ El análisis de los sistemas existentes relacionados con el campo de acción no cumplen los requisitos funcionales que se desean, lo que evidenció la necesidad de desarrollar una aplicación que cumpliera con ellos.
- ✓ El análisis de diferentes herramientas, tecnologías, lenguajes y metodologías permirtió seleccionar las adecuadas para el desarrollo de la aplicación, como fueron: la metodología SXP, el lenguaje PHP usando el marco de trabajo symfony2, como IDE de desarrollo Netbeans, como gestor de base de datos PostgreSQL, como lenguaje para la consulta de información DQL y como manager pgAdmin III.
- ✓ El uso de técnicas para realizar la captura de requisitos permitió identificar las funcionalidades que debe tener el sistema, realizando a partir de ellos el diseño de la propuesta de solución.
- ✓ Los diagramas y el modelo de diseño obtenidos fueron la base para la posterior implementación de una solución orientada a las especificaciones propias del software.
- ✓ Las métricas aplicadas a la propuesta de solución propiciaron una correcta validación de las funcionalidades y el diseño del software.
- ✓ Las pruebas realizadas permitieron comprobar las funcionalidades de la aplicación, arrojando resultados satisfactorios que validan la investigación.

RECOMENDACIONES

Después de concluido el trabajo se recomienda que:

- Además de tener en cuenta, para la permanencia, las actividades ajenas al proyecto, se mantenga un control de los procesos abiertos en la pc y tener reportes de los mismos.
- Integrar las actividades de docencia y extensión que se guardan en la aplicación, con aquellas que se registran en el GESPRO para tener un Plan de Trabajo completo.

Bibliografía

Achour, Mehdi. 2014. php. [En línea] 25 de abril de 2014. [Citado el: 7 de noviembre de 2013.] http://www.php.net/manual/es/intro-whatis.php.

Amaya, R. 2009. ¿Qué es ORM? [En línea] 2009. [Citado el: 11 de enero de 2014.] http://www.srbyte.com/2009/09/que-es-orm.html.

ASV-GIMNASIOS. 2010. ASV-SOFTWARE. [En línea] 2010. [Citado el: 2 de noviembre de 2013.] http://www.asvgimnasios.com.

Boeras Velásquez, M, y otros. 2012. *Aplicando el método de Boehm y Turner.* La Habana : s.n., 2012.

DATYS Tecnologías y Sistemas. 2010. DATYS Tecnologías y Sistemas. [En línea] 2010. [Citado el: 2 de noviembre de 2013.] htttp://www.datys.cu.

Departamento de Lenguajes y Sistemas Informáticos Universidad de Sevilla. 2008. Departamento de Lenguajes y Sistemas Informáticos Universidad de Sevilla. [En línea] 14 de febrero de 2008. [Citado el: 10 de noviembre de 2013.] http://www.lsi.us.es/docencia/get.php?id=1130.

doctrine. 2014. doctrine. [En línea] 2014. [Citado el: 8 de noviembre de 2013.] http://www.doctrine-project.org.

E.T.S.I.Informática Dpto.L.C.C. U.M.A. 2013. E.T.S.I.Informática Dpto.L.C.C. U.M.A. [En línea] 2013. [Citado el: 8 de noviembre de 2013.] http://www.lcc.uma.es/~eat/services/html-js/manual13.html.

Eguíluz, J. 2010. ¿Qué es Symfony? symfony.es. [En línea] 2010. [Citado el: 10 de enero de 2014.] http://www.symfony.es.

HTML.net. 2010. HTML.net. [En línea] 2010. [Citado el: 9 de noviembre de 2013.] http://es.html.net/tutorials/css/lesson1.php.

htmlcinco. 2012. htmlcinco. [En línea] 2012. [Citado el: 15 de enero de 2014.] www.htmlcinco.com.

IBIX. 2011. IBIX. [En línea] 26 de enero de 2011. [Citado el: 2 de noviembre de 2013.] http://www.ibix.com.

jQuery escribir menos, haces más. 2014. jQuery escribir menos, haces más. [En línea] 2014. [Citado el: 7 de noviembre de 2013.] http://jquery.com.

Lainado, Pablo. 2011. Lantec Servicios Informáticos. [En línea] 9 de septiembre de 2011. [Citado el: 2 de noviembre de 2013.] http://www.lantec.com.ar.

Larman, C. 1999. *UML y Patrones, Introducción al análisis y diseño orientado a objetos.* Mexico : Prentice Hall, 1999.

LÓPEZ ORTEGA, DANIEL y SANTA VILLA, JESSICA ANDREA. 2012. *ESTUDIO COMPARATIVO DE LAS HERRAMIENTAS CASE: STARUML, POSEIDON FOR UMLY ENTERPRISE ARCHITEST, PARA EL MODELADO DE DIAGRAMAS UML.* PEREIRA : s.n., 2012.

Martínez, Rafael. 2010. PostgreSQL-es Portal en español sobre PostgreSQL. [En línea] 2010. [Citado el: 7 de noviembre de 2013.] http://www.postgresql.org.es/sobre_postgresql.

Netbeans. 2013. Netbeans. [En línea] 2013. [Citado el: 9 de noviembre de 2013.] https://netbeans.org/features/index.html.

Orosco Vaillant, María E. 2010. *Informe de investigación.* Granma : s.n., 2010.

Peñalver Romero, Gladys Marsi. 2008. *MA-GMPR-UR2 Metodología ágil para proyectos de software libre.* Ciudad de La Habana : s.n., 2008.

Peñalver Romero, Gladys Marsi, García de la Puente, Sergio Jesús y Meneses Abad, Abel . 2010. *SXP, metodología de desarrollo de software.* Ciudad de la Habana : s.n., 2010.

pgAdmin PosgreSQL Tools. 2010. pgAdmin PosgreSQL Tools. [En línea] 2010. [Citado el: 8 de noviembre de 2013.] http://www.pgadmin.org.

Potencier, Fabien. 2008. Symfony 1.0, la guía definitiva. [En línea] 2008. [Citado el: 12 de enero de 2014.] http://www.librosWeb.es/symfony_1_0.

Pressman, S. 2001. *Ingeniería de Software. Un enfoque práctico.* España : s.n., 2001.

Rodríguez, Txema. 2012. GENBETA: dev desarrolllo y software. [En línea] 2012. [Citado el: 2 de noviembre de 2013.] http://www.genbetadev.com/frameworks/bootstrap.

Somerville, I. 2008. *Ingeniería de Software.* s.l. : Educacion Límitada de Personal, 2008.

Sommerville, I. 2005. *Ingeniería de Sotfware. Septima Edición.* Madrid, España : Pearson Educación. S. A., 2005. 84-7829-074-5., 2005.

The Apache Software Foundation. 2012. The Apache Software Foundation. [En línea] 2012. [Citado el: 9 de noviembre de 2013.] http://www.apache.org/foundation/.

Tigris.org Open Source Software Engineering Tools. 2009. Tigris.org Open Source Software Engineering Tools. [En línea] 2009. [Citado el: 2013 de noviembre de 10.] http://argouml.tigris.org/.

Unidad Docente de Ingeniería del Software. 2000. *Patrones del ¨Gang of Four¨.* Madrid, España : Facultad de Informática-Universidad Politécnica de Madrid, 2000.

Zamudio, Lennis , y otros. 2009. OoCities.org. [En línea] 2009. [Citado el: 10 de noviembre de 2013.] http://www.oocities.org/es/avrrinf/tabd/Foro/Foro_UML.htm.

zend framework2. 2010. zend framework2. [En línea] 2010. [Citado el: 13 de enero de 2014.] http://framework.zend.com.