Universidad de las Ciencias Informáticas
Facultad 6

Título: **Sistema para la Gestión de Información de Ciencia Tecnología, Innovación y Postgrado del Centro GEYSED.**

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

AUTORES: CYNDI LORENZO JORGE / NUVIA ROQUE FUENTES
TUTOR: Msc. IVÁN PÉREZ MALLEA

Declaración de Auditoria

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Cyndi Lorenzo Jorge **Nuvia Roque Fuentes**

_____

Tutor: MSC. Iván Pérez Mallea

Datos de Contacto

Autor: Cyndi Lorenzo Jorge

Universidad de las Ciencias Informáticas

Email: cyndi@uci.cu

Autor: Nuvia Roque Fuentes

Universidad de las Ciencias Informáticas

Email: nuviarf@uci.cu

Tutor: Iván Pérez Mallea

Universidad de las Ciencias Informáticas

Email: mallea@uci.cu

Ocupación Actual: Profesor**.**

Ingeniero en la Especialidad de Mecánica 1997.

Master en Informática Aplicada 2000.

Categoría Docente: Profesor Auxiliar**.**

Resumen

En el presente trabajo se presenta una solución para la gestión de la información de ciencia, tecnología, innovación y postgrado para el Centro de Geoinformática y Señales Digitales (GEYSED). La existencia de dificultades e ineficiencias en el manejo, seguimiento y control de la información y las tareas asociadas a los profesionales de este centro, pues todo el proceso se realiza empleando documentos MS Excel y Word, surge la necesidad de crear un sistema que agrupe la información referente a cada uno de los profesionales del Centro, (plan de trabajo, maestrías, doctorados, evidencias de Ciencia, Tecnología e Innovación (CTI), etc.) en un mismo lugar facilitando el trabajo, evitando errores en el manejo de la información, y ahorrando tiempo y recursos a la universidad. Para lograr el cumplimiento del objetivo de la investigación, se llevó a cabo un estudio de los sistemas homólogos existentes a nivel internacional, en el país y en la propia universidad, además se analizaron las herramientas y metodologías a utilizar, así como la definición de los artefactos a partir de la metodología XP para el desarrollo de software.

Palabras Clave: gestión, sistema, evidencias, postgrado.

Índice

Índice de tablas

Índice de figuras

Introducción

En la sociedad actual donde términos como “sociedad del conocimiento”, “sociedad de la información”, o “gestión del conocimiento” son cada día más comunes, en gran medida gracias al desarrollo de las Tecnologías de Información y Comunicaciones, es imposible hablar de desarrollo social sin hablar de ciencia e innovación. Desde la década de los 80 del siglo anterior, incluso antes, se habla de indicadores para medir la gestión del conocimiento incluso han aparecido procesos y sistemas para medir los niveles de conocimiento, de ahí que se escuche hablar por ejemplo de cienciometría. La cienciometría estudia los aspectos cuantitativos de la ciencia como disciplina o actividad económica, forma parte de la sociología de la ciencia y encuentra aplicación en el establecimiento de las políticas científicas, donde incluye entre otras las de publicación; emplea, al igual que otras disciplinas, técnicas métricas para la evaluación de la ciencia (tanto a las ciencias naturales como a las sociales), y examina el desarrollo de las políticas científicas de países y organizaciones. (Spinak, 1998)

La educación ocupa un lugar importante en estos sistemas de medición del conocimiento social. Las universidades, encargadas de la formación continua de profesionales, son un punto importante, conformando *rankings* a niveles regionales y mundiales de las mismas en función de estos indicadores cienciométricos.

Incrementar la proyección social de los egresados de las universidades cubanas a través de la actividad de postgrado e investigación, constituye una de las principales premisas de la educación superior como parte de sus acciones estratégicas camino a la formación de profesionales de excelencia, que respondan a las demandas actuales de desarrollo social. La enseñanza superior cubana, desarrolla más de 500 programas de maestría y especialidades para distintas ramas de la ciencia. Más de medio millón de profesionales participan en actividades de postgrado anualmente en Cuba, donde además se gradúan alrededor de 600 doctores en igual periodo. (Jiménez & Barrios, 2012)

Resulta significativo subrayar la participación internacional de numerosos aspirantes a maestrías y doctorados en las diferentes áreas del postgrado cubano. Donde han sido formados hasta el año 2011 más

de 1 500 másteres extranjeros de 71 países y alrededor de 840 doctores de 62 naciones (fundamentalmente de México, Colombia, Brasil y Venezuela). (Jiménez & Barrios, 2012)

Para los diferentes programas de postgrado en Cuba, existen varias modalidades o formas organizativas como cursos, entrenamientos, diplomados y programas académicos como maestrías y especialidades. Otras formas de superación son la auto-preparación, la conferencia especializada, el seminario, el taller, el debate científico y otras que complementan y posibilitan el estudio y la divulgación de los avances del conocimiento, la ciencia, la tecnología y el arte. Los programas correspondientes a la superación profesional son proyectados y ejecutados por centros de educación superior. (MES, 2004) La investigación es una actividad fundamental en el modelo de educación superior cubano, como está definido en los lineamientos de la política educacional aprobada en 1976 con la creación del Ministerio de Educación Superior y donde se expresa que no hay verdadera educación superior sin actividad de investigación. (Investigaciones, 2013)

La Universidad de las Ciencias Informáticas (UCI), a pesar de ser uno de los centros de educación superior más joven de Cuba, posee un amplio sistema articulado de investigación, postgrado e innovación convirtiéndose en el principal centro de la superación de profesionales de la rama en el país. (Postgrado, 2012) Debido a la composición de sus graduados la UCI tiene presencia en todas las regiones del país y en casi todos los organismos que componen los órganos del estado cubano, razón que genera un compromiso indisoluble entre estos profesionales, más de 10 000, y su casa de altos estudios. En la UCI, desde su fundación, también han recibido superación profesional más de 350 extranjeros concentrados fundamentalmente en las Escuelas de Invierno y Verano que se realizan en febrero y julio respectivamente y en estos momentos cursan por primera vez estudios de postgrado académico un grupo de angolanos y un ecuatoriano. (Postgrado, 2013)

Anualmente en la UCI se trazan un grupo de objetivos e indicadores a nivel de universidad que posibiliten la superación continua de sus graduados y el aumento de los niveles científico profesional de su claustro para así permitir a la universidad situarse en lugares decorosos en los *rankings* cienciométricos, elevando el prestigio nacional e internacional del centro. (Investigaciones, 2012) Normalmente estos indicadores se tratan de manera cuantitativa a nivel de universidad y se desglosan entre las diferentes áreas de la misma, suele suceder que los indicadores de ciencia e innovación y los de postgrado se manejen de manera

separada y se les exija su cumplimiento a las áreas por dos direcciones de la universidad, la Dirección de Investigaciones y la Dirección de Formación Postgraduada.

En las áreas estos indicadores son centrados por los vice-decanatos de investigación y postgrado en las facultades o los subdirectores de investigación y postgrado en los centros. Esta estructura genera una situación donde la persona que atiende esta esfera en las áreas recibe pedidos de información de dos lugares diferentes además de ser el encargado de atender otros aspectos de formación del claustro que no se incluyen en estas áreas como puede ser la categorización docente. Para cada una de las áreas relacionadas con el entorno académico-profesional de la universidad esta descentralización de los indicadores de ciencia, innovación y postgrado implica tener un control estricto de las tareas que se convierten de simples números a acciones concretas en los planes de trabajo de los profesionales del área, pudiendo así dar respuestas ágiles y puntuales a los diferentes organismos superiores que así lo requieren.

Entre dichas áreas se encuentra la Subdirección de Investigación y Postgrado del Centro de Geoinformática y Señales Digitales (GEYSED), en la cual a pesar de darle gran importancia a estas actividades, existen dificultades e ineficiencias en el manejo, seguimiento y control de la información y las tareas asociadas a los profesionales ya que todo el proceso se realiza empleando documentos MS Excel y Word. Esta condición de trabajo provoca que las planificaciones sean engorrosas y que la proyección del plan de trabajo de los profesionales se vea entorpecida por la gestión manual que requiere la consulta de varias páginas tabuladas para generar dicha información en un momento determinado. Además se hace lenta y difícil la toma de decisiones debido a que la información no se encuentra unificada para cada persona, provocando retrasos en la subdirección, aumentando el margen de error, acarreando gastos monetarios y de tiempo, y pérdida de prestigio tanto para el centro como para la universidad influyendo negativamente en la formación de profesionales altamente calificados en la rama de la Informática.

En nuestra universidad existen algunas soluciones informáticas que gestionan algunos de estos procesos pero solo de manera cuantitativa ((Rodríguez Odales & Ortíz López, 2008), (Alvarez Muñoz & Soto Pérez, 2011), (González Soriano & Martínez Consuegra, 2010)), sin lograr que estos números se reflejen en el plan de trabajo o de resultados del profesional y ninguna de estas soluciones enfoca de manera integral los indicadores de ciencia e innovación con los de postgrado.

Derivado de la situación problémica expuesta se puede arribar al siguiente **problema a resolver** ¿Cómo contribuir a la gestión de información sobre ciencia, tecnología e innovación del Centro GEYSED de la Facultad 6 para elevar el control del cumplimiento de los objetivos de trabajo del área?

En función de resolver el problema anterior se definen como **objeto de estudio:** los procesos de gestión de información sobre ciencia, tecnología e innovación, centrándose en el **campo de acción** de los sistemas de gestión de información de Ciencia, Tecnología e Innovación. Además se propone como **objetivo general** desarrollar un Sistema para la Gestión de Información de Ciencia, Tecnología e Innovación y Postgrado que contribuya a elevar el control del cumplimiento de los objetivos de trabajo del área.

Para dar cumplimiento a este objetivo se tienen los siguientes **objetivos específicos**:

1. Realizar un estudio del estado del arte sobre los sistemas de gestión de documentación para ciencia tecnología e innovación.
2. Diseñar el sistema para la gestión de la documentación de la Subdirección de “Investigación y Postgrado” del centro GEYSED.
3. Implementar el sistema para la gestión de la documentación de la Subdirección de “Investigación y Postgrado” del centro GEYSED.
4. Validar el sistema implementado.

Para desarrollar los objetivos específicos se planifican las siguientes **tareas de la investigación**:

1. Caracterización de determinados sistemas, tanto nacionales como internacionales, para la conformación del estado del arte.
2. Selección de las tecnologías para el desarrollo del sistema.
3. Descripción de la metodología a definir para el sistema.
4. Identificación de los requisitos funcionales y no funcionales de los procesos de gestión de CTI.
5. Realización del diseño de la solución.
6. Implementación del sistema de planificación de tareas de superación.
7. Validación de la solución planteada.

Para una mejor presentación y comprensión de la información inherente a esta investigación, se propone la siguiente estructura del documento:

Capítulo 1: Fundamentación teórica

En este capítulo se hace un análisis de los principales conceptos asociados al dominio del objeto de estudio, así como las soluciones existentes y el uso de las tecnologías en el desarrollo de la propuesta.

Capítulo 2: Características del sistema

En este capítulo se traza como objetivo principal el proceso de construcción de la solución, en aras de obtener como resultado los artefactos que propone la metodología de desarrollo que se identifique y seleccione para esta investigación.

Capítulo 3: Implementación y Validación de la solución propuesta

Se abarcará en este capítulo todo lo concerniente a la implementación del sistema. Se realizan además pruebas a la aplicación, en pos de validar y garantizar su calidad.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se exponen los principales conceptos afines al problema en cuestión que aborda el presente trabajo de diploma. Además se brinda información acerca de los indicadores tanto de postgrado como de ciencia e innovación así como los referidos a los procesos de categorización docente, indicadores que forman parte de la esencia misma de este trabajo. Se brinda información sobre el estado del arte y otros sistemas similares desarrollados fuera y dentro de la UCI. Por otra parte se ofrecen detalles sobre la metodología y las tecnologías que darán soporte a la solución de la investigación desarrollada en este trabajo de diploma.

1.2 Marco conceptual

La composición del claustro universitario es uno de los indicadores cienciométricos que prestigian las universidades a nivel mundial. La cantidad de doctores, masters y profesores e investigadores con categorías principales son aspectos que influyen considerablemente en las evaluaciones de calidad de una institución universitaria.

En el caso de la Universidad de las Ciencias Informáticas, en su génesis tuvo que asistirse de profesores provenientes de otras instituciones y universidades que prestaran sus servicios para poder comenzar la actividad docente y de producción de software. Todos esos profesionales contaban con categorías docentes principales como (Profesor Titular, Profesor Auxiliar, Asistente e Instructor).

Con el paso del tiempo y el incremento del estudiantado a razón de 2000 estudiantes por año aproximadamente, se hizo necesario aumentar el universo profesoral con profesionales recién graduados y cuyos perfiles estuviesen relacionados con la carrera que se imparte en la universidad. Luego de los primeros 4 años de fundada la universidad, la composición del claustro alcanzó cifras de casi el 70% de profesores en categorías inferiores como instructor y asistente y solo el 30% en categorías superiores, los niveles de doctores no alcanzaban ni el 10%. (Postgrado, 2013) Por ello, su categorización docente, su desarrollo científico y superación profesional se convertirían en los procesos de mayor prioridad en la universidad para garantizar la calidad de los resultados en los entornos académico y productivo.

A continuación se detallan los conceptos más importantes de estos procesos, utilizando como base el Reglamento de Postgrado del Ministerio de Educación Superior (MES, 2004) y la Resolución 128 del 2006 para la Categorización Docente (MES, 2006).

1.2.1 Categorización docente

El proceso de categorización docente, da comienzo a la conformación y actualización del expediente del docente aspirante, labor en la que interviene el departamento de cuadros de la dirección de desarrollo del capital humano de la universidad. El aspirante, es el responsable de que en su expediente esté registrada y documentada la trayectoria de su desempeño laboral, académico y científico. (MES, 2006)
El proceso de categorización se divide en dos grupos principales: las categorías docentes principales y las categorías docentes especiales.

Figura 1 Categorías docentes

También existe la categoría o condición de Profesor Consultante que se ofrece a profesores titulares con una vasta experiencia.

Según la resolución 128 del Ministerio de Educación Superior del año 2006, y vigente en la actualidad, en sus artículos del 62 al 64 plantea: Todo docente universitario que posea alguna categoría docente principal o complementaria, está en la obligación de mantener la adecuada preparación y actualización científico pedagógica que el ejercicio de sus funciones requiere. Cada cinco años el docente universitario es sometido

a proceso de ratificación de su categoría docente. De no concurrir a dicho proceso la categoría docente que posea queda sin efecto. Los docentes que ostenten la condición de Profesor Consultante están eximidos de la ratificación de su categoría docente. El proceso de ratificación se realizará anualmente en el período de enero a abril, ambos inclusive, y al mismo concurrirán los docentes que hayan cumplido 5 años desde el otorgamiento o ratificación de la categoría docente o que los cumplan en el período habilitado para el proceso de ratificación. (MES, 2006)

A pesar de que la ratificación de la categoría docente es responsabilidad del profesor es importante para los directivos tener un control estricto de en qué períodos algún profesional de su claustro debe ratificar/cambiar su categoría docente permitiéndole así organizar sus planes de trabajo e incorporando a los mismos tareas relacionadas con esa actividad.

1.2.2 Educación de postgrado

La educación de postgrado es una de las líneas principales en las que se centra el trabajo de la educación superior en Cuba, representando el nivel más alto del sistema de educación superior, enmarcado en la formación permanente de los graduados universitarios, concurriendo no solo los procesos de enseñanza y aprendizaje, sino también otros como investigación, innovación, creación artística, etc. La importancia de la educación de postgrado se evidencia en la necesidad de la superación a lo largo de la vida profesional, también en la gestión del aprendizaje y su socialización en la construcción del conocimiento, es la fuerza social transformadora que promueve el desarrollo sostenible de la sociedad.

Las formas organizativas principales de la superación profesional en la educación de postgrado son el curso, el entrenamiento y el diplomado, de estos 3, solo el diplomado en conjunto con los programas académicos son nombrados programas de postgrado. (MES, 2004)

Los programas de postgrado son sustentados por un sistema acumulativo de créditos académicos que facilita la flexibilidad organizativa de los planes de estudio, la transferencia de docentes, y la homologación de estudios realizados entre diversas instituciones.

El crédito académico es una unidad de expresión cuantitativa y cualitativa que valora los resultados alcanzados teniendo en cuenta la profundidad, el volumen y la intensidad del trabajo que realiza el estudiante para lograr las metas trazadas en los programas. Los mismos se otorgan al considerar cumplidos los objetivos de las actividades planificadas. Los créditos pueden ser obligatorios, opcionales y libres, de acuerdo con los objetivos y la estrategia de formación. Tanto los créditos opcionales, como los libres, se obtendrán bajo la orientación del comité académico.

- Obligatorios: son aquellos que se exigen a todos los estudiantes, sin distinción.
- Opcionales: son seleccionados por los alumnos dentro del propio programa matriculado, a partir de un conjunto de ofertas.
- Libres: son los que se obtienen en cursos y entrenamientos de posgrado fuera del programa en el que está matriculado el profesional.

Los programas de postgrado, por otra parte, tienen asociados un grupo de cursos, que no son más que la organización de un conjunto de contenidos que abordan resultados de investigación relevantes o asuntos trascendentes con el propósito de complementar o actualizar los conocimientos de los profesionales que los reciben.

Otra forma organizativa de superación es el entrenamiento, el cual posibilita la formación básica y especializada de los graduados universitarios, particularmente en la adquisición de habilidades y destrezas en la asimilación e introducción de nuevos procedimientos y tecnologías.

Por último se encuentra el diplomado como otra de las formas organizativas; el mismo tiene como objetivo la especialización en un área particular del desempeño y propicia la adquisición de conocimientos y habilidades académicas, científicas y/o profesionales en cualquier etapa del desarrollo de un graduado universitario, de acuerdo con las necesidades de su formación profesional o cultural.

Apoyándose en el conocimiento de los principales conceptos anteriormente abarcados, pertenecientes a los procesos de categorización docente y educación de postgrado, se realiza la planificación de las tareas de superación de los profesionales.

1.2.3 Principales indicadores de ciencia e innovación

Hasta ahora hemos hablado de indicadores de ciencia, tecnología e innovación pero en lo concreto: ¿Cuáles son estos famosos indicadores cienciométricos? En un artículo publicado hace algunos años en Scientometrics, Vinkler presenta una tabla comparativa con 46 indicadores simples y compuestos (Spinak, 1998). Los indicadores pueden dividirse en 2 grandes grupos:

- Indicadores de publicación: miden la calidad e impacto de las publicaciones científicas.
- Indicadores de citación: miden la cantidad e impacto de las vinculaciones o relaciones entre las publicaciones científicas.

Los indicadores cienciométricos solo miden una parte del trabajo de CTI que se realiza en la educación superior cubana. Los principales indicadores de los grupos señalados anteriormente se recogen en Scimago (SCImago, 2010). Entre los principales indicadores de CTI establecido por el MES (Investigaciones, 2012) están:

- Premios obtenidos por la institución a diferentes niveles.
- Publicaciones realizadas por los investigadores de acuerdo con los niveles establecidos por el MES.
- Trabajos Presentados a eventos nacionales o internacionales.
- Capacitación realizada en las diversas áreas del conocimiento (Postgrado)
- Proyectos Innovación y Desarrollo
- Patentes y Registros
- Resultados de investigación Introducidos en la práctica.
- Empleo de Estudiantes en la realización de trabajos de I+D
- Indicadores Scimago para medir la calidad, el impacto y el índice de citación de las publicaciones.

1.2.4 Planificación

La planificación, de manera general, es el proceso de toma de decisiones para lograr un objetivo deseado, teniendo en cuenta la situación actual y factores del medio ambiente que pueden influir en el logro del

resultado esperado. Una de las variantes de la planificación, es la planificación estratégica, la misma consiste en un patrón de decisiones coherente, unificador e integrado, que determina y revela el propósito de un centro u organización en términos de objetivos a largo plazo, programas de acción y prioridades para la asignación de recursos. Para llevar a cabo la planificación estratégica se tienen en cuenta las siguientes fases. (David, 2003)

- Visión.
- Análisis del entorno y escenarios futuros.
- Misión.
- Análisis interno de la organización.
- Definición de objetivos y metas.
- Identificación e implementación de factores estratégicos.

Aplicando estas fases a los procesos de superación postgraduada, la visión sería la apreciación idealizada de lo que se desea lograr, en otras palabras la obtención de una planificación exitosa, que permita un mejor control de las tareas de superación postgraduada.

La segunda fase, es el análisis del entorno y escenarios futuros. Se debe tener en cuenta identificar y prever los cambios que se produzcan en la realidad actual o el comportamiento futuro, esos cambios pueden verse reflejados en la composición del claustro a través de los años en cuanto a categoría docente, títulos académicos, grados científicos, etc. Esta variación influiría directamente en la futura planificación de las tareas.

Otra fase importante, es la de definición de objetivos y metas, los objetivos son los resultados a largo plazo y las metas los resultados a corto plazo. Los objetivos se establecen en términos generales y amplios, por ejemplo, un objetivo sería lograr la planificación de las tareas pertenecientes a los programas docentes de postgrado y una meta sería el desglose de los cursos aún no concluidos por los profesionales que cursan los programas de superación. Por último, la fase de implementación de factores estratégicos se centra en construir planes y asignar recursos. O sea, hacer operativos los objetivos y metas obtenidos del análisis de la situación. Todos estos conceptos referentes a la categorización docente, la educación de postgrado y la

planificación científica, de una manera u otra, ayudan a una mejor comprensión de los procesos del negocio y contribuyen a formar bases para sustentar la investigación de los sistemas que puedan brindar una posible solución al problema que se plantea en el presente trabajo de diploma.

1.3 Tendencias y paradigmas actuales

La educación postgraduada, la gestión de indicadores de ciencia e innovación y la categorización docente de profesionales son procesos que complementan la gestión universitaria, representando una pequeña parte de los procesos que conforman el medio ambiente en las universidades. Los sistemas de gestión universitarios abarcan los más disimiles procesos y están representados desde servicios universitarios como la gestión de matrículas, cursos, consultas de plan de estudio y cronograma de evaluaciones hasta los servicios de índole financiero; así como desde el análisis de la situación corporativa de la universidad o la gestión de los recursos humanos y económicos.

1.3.1 Sistemas a nivel internacional

Desde el ámbito internacional, los sistemas de gestión universitaria son ampliamente empleados por diferentes instituciones de gran prestigio, un ejemplo de ello es el **UNIVERSITAS XXI**. Esta solución integral de gestión fue creada por el proyecto español de las universidades públicas de Alcalá, Carlos III de Madrid, Castilla-La Mancha, Rey Juan Carlos, Salamanca, Valladolid y del Grupo Santander, el cual es conocido como la Oficina de Cooperación Universitaria (OCU). Dicha solución es un Planificador de Recursos Empresariales (ERP) para las universidades, que consta de módulos que cubren algunos procesos de la gestión universitaria tales como la gestión académica, los recursos humanos, el ámbito económico, las investigaciones, la inteligencia institucional y la comunicación entre los usuarios del sistema mediante SMS, E-Mail y *Twitter*. De estos procesos que informatiza de alguna manera, solo las investigaciones y la gestión académica presentan algún tipo de relación con la superación postgraduada. El módulo de investigación aborda el tratamiento de la información relativa a la actualidad investigadora y la producción científico-técnica del personal investigador, además automatiza el control de proyectos, tramitación de patentes, gestión de contratos y gestión de grupos de investigación. Por otra parte el módulo académico cubre la gestión de becas nacionales e internacionales, organización de prácticas en empresas, definición de planes

de estudio y planificación de la docencia. La caracterización de estos módulos, demuestra de manera concreta la ausencia de alguna forma de planificación de tareas de superación para la formación postgraduada (Oficina de Cooperación Universitaria Española (OCU), 2014)

Otro de los sistemas de gestión universitaria investigados fue **CLASS**. El cual es un software creado por la empresa INNOVASOFT.SA que lo comercializa. Esta empresa tiene su sede en la casa matriz de San José, Costa Rica y se especializa en la gestión académica y contable. Por su parte CLASS define como principal propósito: optimizar los diferentes procesos entre el área de servicios universitarios y el área financiera. Con una integración entre sus modulaciones, pretende una autoadministración con poca intervención de usuarios, basada en herramientas de plataforma de auto-gestión para estudiantes, profesores y catedráticos por medio de Internet.

Entre los beneficios más importantes que presenta este sistema se encuentran los siguientes:

1. Aplicaciones universitarias-financieras integradas bajo un solo proveedor de software.
2. Centralización de información de multicompañías y sedes en un único servidor de datos para análisis y toma de decisiones.
3. Analizador de la situación corporativa de la universidad. Plataforma de pagos (conexión en línea con la red local bancaria y pago con tarjeta).
4. Descongestionamiento de las funciones del departamento de registro y matrícula, fomentando el autoservicio de los usuarios (estudiantes y profesores) por medio de Internet.

El descongestionamiento de las funciones del departamento de registro y matrícula, contribuye a la informatización del proceso de matriculación en cursos, permitiendo controlar los cursos a los que están asociados tanto un estudiante en su formación como un profesional en su superación de postgrado. Sin embargo, el sistema como tal no cuenta con un módulo centrado en la planificación de tareas de superación postgraduada para profesionales, lo que dificulta su adaptabilidad a las especificidades de esta investigación.

Además de los sistemas descritos inicialmente se tiene el **Sistema de autogestión de alumnos SIUGuaraní,** que es un software argentino que tiene como finalidad, brindar a los alumnos de la Universidad Nacional del Noreste (UNNE) y sus Unidades Académicas una herramienta que les permita acceder a través de Internet y realizar acciones como: inscripción a exámenes y cursos, reinscripción a carrera, consulta de créditos, consulta de inscripciones, consulta de plan de estudios e historia académica, consulta de

cronograma de evaluaciones parciales, notas de evaluaciones parciales, materias regulares, agenda de clases, solicitud de certificados, actualización de datos censales y recepción de mensajes (UNNE, 2014). Esta posibilidad de acceder a la información y realizar algunas gestiones administrativas evitando traslados innecesarios de la persona, contribuye a agilizar los procesos administrativos en general, en un ambiente más seguro y mejorando la calidad de la información obtenida. De las funcionalidades de este sistema, solamente consulta de créditos y la historia académica, presentan elementos que tributan a la formación de postgrado, pero no son suficientemente abarcadores para cumplir con el objetivo de planificar tareas para la superación postgraduada de los profesionales.

1.3.2 Sistemas a nivel nacional

En las universidades cubanas el sistema de gestión universitaria SIGENU desarrollado por el Instituto Superior Politécnico José Antonio Echeverría (ISPJAE) es el más difundido, este centra los procesos sustantivos de la vida universitaria. El módulo de postgrado del SIGENU no está desarrollado por los autores originales sino por un grupo de la Universidad Central de las Villas Marta Abreu (Rodríguez, 2013). Al igual que otros sistemas de gestión universitaria su información es más cuantitativa y se pueden gestionar la inscripción de actividades de postgrado, la matrícula en las mismas, la gestión de certificados, etc. Este sistema no incluye indicadores de Ciencia e innovación y no permite que un profesional vea en un solo lugar reflejadas todas las tareas de su plan de trabajo o de resultados.

Luego de explorar el escenario internacional y nacional, resulta necesario centrarse en algunos trabajos desarrollados en nuestra propia institución, específicamente en el sistema de gestión universitaria que se encuentra en desarrollo en el Centro de Informatización Universitaria (CENIA) y no es más que un ERP Universitario denominado **Gestión Universitaria UCI**. Este incluye varios subsistemas, agrupados en las siguientes áreas de proceso: Pregrado, Postgrado, Producción, Investigación, Ingreso, Ubicación Laboral, Laboratorio, Residencia, Extensión Universitaria, Cooperación Internacional, Biblioteca y Teleformación. (González Soriano & Martínez Consuegra, 2010)

Específicamente el módulo de postgrado, está pensado para gestionar los procesos de formación postgraduada con el objetivo de garantizar dichas actividades para el claustro de profesores del centro y contribuir a la superación de los profesionales. Aún con la futura liberación de este módulo, el sistema de

Gestión Universitaria UCI, no contaría con la opción de planificar tareas a los profesionales, en función de las metas a cumplir para terminar los programas de superación postgraduada o en función de los objetivos a superar según la categoría docente avalada.

También existen varias tesis de grado que implementan sistemas de postgrado, como ((Rodríguez Odales & Ortíz López, 2008), (Alvarez Muñoz & Soto Pérez, 2011), (González Soriano & Martínez Consuegra, 2010), etc.). Entre estos trabajos es importante destacar el sistema “Xendero” desarrollado en la facultad 3 (Quiles López, 2012) el cual es un sistema desktop desarrollado en Java que permite incipientemente combinar los indicadores de postgrado, los de ciencia e innovación y las categorías docentes, logrando tener en un solo sistema el control de los mismos. Este sistema tiene como principal deficiencia el vínculo de los indicadores con los planes de trabajo de los profesionales y además que al ser una aplicación desktop solo está disponible para las personas que gestionan la información y los usuarios no son capaces de poder ver su plan de resultado ni lo que se espera de el en relación al postgrado, la ciencia y la innovación, tampoco es posible ver la planificación de escenarios a largo plazo.

1.4 Valoración de los sistemas estudiados

El estudio de los sistemas anteriores, contribuyó con el establecimiento de las bases del conocimiento para la realización y complementación del proceso de desarrollo de software en respuesta al problema planteado. Para ello se pretende utilizar la experiencia adquirida en función de una mejor comunicación con el cliente, permitiendo sugerir elementos que puedan convertirse en nuevas funcionalidades atractivas para el mismo y que serán integradas a la solución. De igual manera, se pudo verificar que ninguno de los sistemas gestiona las funcionalidades referidas en los indicadores de Ciencia e Innovación y postgrado en vínculo directo al plan de resultados o trabajo de un profesional, en dependencia de la categoría docente o en función de las metas a lograr para vencer los programas académicos de superación postgraduada, permitiendo así aumentar el control y la toma de decisiones por la autoridades competentes.

1.5 Metodología y tecnologías

Terminado el análisis de los sistemas de gestión universitaria, el siguiente paso es la selección de la metodología y las tecnologías que brindarán soporte a la solución, por lo que en los segmentos desarrollados a continuación se describen las principales características inherentes al tema de este epígrafe.

1.5.1 Metodologías de desarrollo de software

En el proceso de desarrollo de software se hace imprescindible el uso de una metodología de desarrollo para llevar a cabo un control total del producto en cuestión. Una metodología está formada por fases, cada una de las cuales se puede desglosar en sub-fases, que guiarán a los desarrolladores de sistemas a seleccionar las técnicas propicias en cada parte del proyecto además de planificarlo, gestionarlo, controlarlo y evaluarlo. Una metodología refiere una representación que permite facilitar la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes implicadas en la construcción del sistema. Tienen como finalidad mostrar un conjunto de técnicas tradicionales y modernas de modelado de sistemas que permitan desplegar software de calidad, incluyendo heurísticas de construcción y criterios de comparación de modelos de sistemas.

Básicamente, las metodologías de desarrollo de software suelen dividirse en dos grupos: metodologías pesadas (tradicionales) y metodologías ágiles. (Letelier Torres, Sánchez Palma, & Molina Marco, 1997)

Metodologías pesadas. Proceso Unificado de Desarrollo

Las metodologías pesadas están fundamentadas en normas procedentes de estándares seguidos por el entorno de desarrollo, brindan por lo general cierta resistencia a los cambios que puedan ocasionarse durante el ciclo de desarrollo del producto, son procesos muy bien controlados con numerosas políticas y normas a seguir. En este conjunto se encuentra la metodología: Proceso Unificado de Desarrollo RUP, en inglés (*Rational Unified Process*), en la actualidad la más utilizada de esta alternativa. La misma se caracteriza por ser dirigida por casos de uso, estar centrada en la arquitectura y ser iterativa e incremental. Entre sus principios se encuentran adaptar el proceso a la necesidad del cliente, equilibrar prioridades, demostrar valor iterativamente, impulsar la colaboración entre equipos, elevar el nivel de abstracción y enfocarse en la calidad. Sin embargo RUP presenta algunos inconvenientes para su aplicación en algunos proyectos como son: límite de tiempo demasiado ajustado, la documentación al ser tan exhaustiva, hace mucho énfasis en la planificación, no siempre se puede llevar a cabo la implementación del software de acuerdo a su planificación, pues aparecen cambios los cuales solo pueden ser percibidos en el momento de la codificación.

Metodologías ágiles. Programación Extrema

Por otro lado las metodologías ágiles están basadas en heurísticas provenientes de prácticas de producción de código, y están especialmente preparadas para sufrir cambios durante el proyecto, son un proceso menos controlado y con pocos principios, por lo que desarrollan software en cortos períodos de tiempo y exigen poca documentación. Entre ellas se puede encontrar la metodología Programación Extrema (XP, en inglés *eXtreme Programming*).

La XP fue concebida por Kent Beck, como un proceso de creación de software diferente al convencional. En palabras de Beck: "XP es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software" (Beck, 2000). XP es una de las metodologías ágiles para el desarrollo de software más exitosas de la actualidad. Se utiliza en proyectos con pequeños equipos de desarrollo y con corto plazo de entrega. Se basa en la retroalimentación entre el cliente y el equipo de desarrollo, buena comunicación entre los participantes y simplicidad en las soluciones implementadas. Consiste en una programación rápida, cuya particularidad es tener como miembro del equipo al usuario final. Es adecuada para proyectos con requisitos imprecisos, muy cambiantes, y donde existe un alto riesgo técnico (Aguilar Ramos, 2005).

En XP se persigue la idea de la programación en dúos, dada las ventajas que promete ésta respecto a la creación del código, pues se pueden impedir errores y malos diseños al controlar cada línea de código y decisión de diseño instantáneamente. La interacción entre ambos desarrolladores puede generar discusiones que lleven a mejores estructuras y algoritmos, aumentando la calidad del software. La XP incluye buenas prácticas de desarrollo como son el Desarrollo Guiado por Test (TDD, en inglés Test *Development Driver*) (Jeffries, Hendrickson, & Anderson, 2000) e Integración Continua (Marches & otros, 2002).

1.5.2 Lenguaje de modelado

El lenguaje de modelado visual se usa para detallar, concebir, construir y evidenciar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Con ellos es posible diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Entre los más conocidos están: UML, MOF, Ecore, entre otros.

UML

Uno de los lenguajes de modelado visual más conocidos es el Lenguaje de Modelado Unificado (UML, en inglés *Unified Model Language*), el cual contiene conceptos semánticos, notación y principios generales.

Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que contengan generadores de código así como, generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Procura dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. Es una consolidación de varias de las notaciones y conceptos más usados en la modelación orientada a objetos.

Es importante recalcar que UML no es una guía para realizar el análisis y diseño orientado a objetos, no es un proceso, sino un lenguaje que permite la modelación de sistemas, con tecnología orientada a objetos. Desde el año 1995, UML es un estándar aprobado por la ISO como ISO/IEC 19501:2005 *Information technology*; está respaldado por el *Object Management Group* (OMG) (Jacobson, Booch, & Rumbaugh, 2000).

1.5.3 Herramienta CASE

CASE (*Computer Aided Software Engineering*). Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de Software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de software.

Las herramientas CASE se pueden clasificar teniendo en cuenta los siguientes parámetros:

- Las plataformas que soportan.
- las fases del ciclo de vida del desarrollo de sistemas que cubren.
- La arquitectura de las aplicaciones que producen.
- Su funcionalidad.

Visual Paradigm

Visual Paradigm es una de las herramientas UML CASE del mercado, considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelente facilidades de interoperabilidad con otras aplicaciones. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones. Fue creada para el ciclo vital completo del desarrollo de software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación. Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de las clases. Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros. (Pressman, 2005)

Características

- Producto de calidad.
- Soporta aplicaciones Web.
- Maneja varios idiomas**.**
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Se integra con las siguientes herramientas Java:
    - Eclipse/IBM WebSphere
    - . Jbuilder.
    - NetBeans IDE.
    - Oracle Jdeveloper.
    - BEA Weblogic.

Ventajas

- Apoya lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Genera modelos VP-UML instantáneamente a partir de código binario .Net.
- Generación de documentación en formatos HTML y PDF.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

1.5.4 Tecnologías para el Desarrollo Web del lado del Cliente.

Las tecnologías del lado del cliente son un conjunto de lenguajes que tienen su mayor protagonismo en la confección de páginas dinámicas, en estas páginas, toda la carga del procesamiento de los efectos y funcionalidades la soporta el navegador.

Java Script

El lenguaje de programación Java Script se utiliza en la creación de programas encargados de realizar acciones dentro del ámbito de una página web. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas, así como las interactividades con el usuario. Técnicamente es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con Java Script se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Actualmente es utilizado por millones de páginas web para validar

formularios, crear cookies, detectar navegadores y mejorar el diseño, su fácil aprendizaje hace de él un lenguaje muy demandado. (Flanagan & Ferguson, 2002)

Symfony 2.5.4

Symfony es un *framework* diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony fue diseñado para ajustarse a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Unix estándares).
- Independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de Propel, permiten cambiar con facilidad de SGBD en cualquier fase del proyecto.
- Utiliza programación orientada a objetos, de ahí que sea imprescindible PHP 5.
- Sencillo de usar en la mayoría de casos, aunque es preferible para el desarrollo de grandes aplicaciones Web que para pequeños proyectos.
- Aunque utiliza MVC (Modelo Vista Controlador), tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de base de datos, el controlador frontal y las acciones.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicación empresarial y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.

- Una potente línea de comandos que facilitan generación de código, lo cual contribuye a ahorrar tiempo de trabajo. (Eguiluz, 2012)

HTML 5

Lenguaje de Marcado de Hipertexto (HTML, en inglés *HyperText Markup Language*, es utilizado para la creación de páginas Web estáticas. Esta versión incorpora algunas etiquetas nuevas pensadas para hacer que la estructura de la página Web sea más lógica y funcional. El enfoque general ha cambiado bastante respecto a versiones anteriores de HTML, añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad. También se tiene en cuenta que muchas páginas Web actuales son dinámicas, pareciéndose más a aplicaciones que a documentos. Algo básico es que HTML5 está definido en base al DOM (la representación interna de una Web con la que trabaja un navegador), dejando de lado la representación real, definiendo a la vez un estándar HTML y XHTML (Lubbers, Albers, & Salim, 2010)

CSS 3

Las hojas de estilo en cascada, conocidos como CSS se define según como "…un lenguaje de hojas de estilo creado para controlar la presentación de los documentos...", y se utilizan para darle forma y aplicar diseño o presentación al contenido o estructura HTML y XML que forma una página o aplicación web.

CSS es la mejor forma de separar contenidos de presentación y es imprescindible para la creación de páginas web complejas ya que permite realizar cambios a múltiples elementos dentro del código, agilizando considerablemente así el proceso de cambios.

En los últimos años el código CSS ha evolucionado hasta su tercera revisión (CSS 3), la cual incluye propiedades que permiten alcanzar efectos complejos, facilitando la aplicación, edición y actualización de formato en todo un sitio web. Se centra en la apariencia final del documento, por lo que constituye un recurso muy útil para los márgenes, espaciado diseñadores de páginas Web. Gracias a esta tecnología es más fácil especificar entre líneas, colores para el texto y el fondo, tamaño y estilo de las fuentes, y otros detalles**.**

AJAX *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML)

Es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones. Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (*scripting language*) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML. (Garrett, 2005)

1.5.5 Tecnologías de desarrollo web del lado del servidor.

Las tecnologías de desarrollo web del lado del servidor no dependen del navegador ya que son interpretadas y ejecutadas por el servidor. Se consideran de mayor importancia ya que son las que garantizan el intercambio de datos en ambos sentidos, entre el navegador del cliente y el servidor Web, y a su vez las que soportan la funcionalidad del sitio.

Con el rápido desarrollo de internet y el uso cada vez más frecuente de los sitios web en todo el mundo se ha disparado un interés extraordinario en el diseño e implementación de soluciones flexibles y robustas. Estos diseños tienen una aplicación muy amplia en la transmisión y administración de la información y numerosas empresas los han adoptado como principio para implementar toda la modelación del negocio que desarrollan. Esto ha repercutido en un auge del comercio electrónico, la mercadotecnia informática, y por supuesto, del desarrollo de aplicaciones que funcionen en el ambiente agradable y asequible de la Internet. (Wilson, 1993)

Hoy, existen muchas tecnologías que se pueden usar para desarrollar sitios Web dinámicos usando la filosofía Cliente/Servidor. Aquí todo gira alrededor de un navegador cliente en la máquina del usuario que envía peticiones a un servidor Web encargado de darles curso a una gran cantidad de estas peticiones incluso concurrentemente.

PHP (por sus siglas en inglés *HipertextPreprocessor*)

Es un lenguaje del lado del servidor para la generación de HTML. El lenguaje PHP es gratuito e independiente de la plataforma, rápido, con una gran librería de funciones y mucha documentación. El tiempo de aprendizaje de PHP es muy corto gracias a su potencial y simplicidad. Las principales características de PHP son: rapidez, puede ser utilizado en diversos sistemas operativos, servidores web y la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Ha evolucionado como lenguaje orientado a objetos permitiendo incluir nuevas funcionalidades, mejoras, y reutilización de código en sitios web. (Group., 2011)

PHP tiene las cuatro grandes características necesarias para ser un potente lenguaje de scripts: velocidad, estabilidad, simplicidad y seguridad**.**

Velocidad: PHP no requiere demasiados recursos de sistema. Por esta razón no crea demoras en la máquina.

Estabilidad: Con el respaldo de una increíble comunidad de programadores y usuarios es mucho más difícil para los errores sobrevivir. Se utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.

Simplicidad: PHP permite a los programadores generar código en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.

Seguridad: PHP provee diferentes niveles de seguridad, estos pueden ser configurados desde el archivo.

1.5.6 Sistema Gestor de Bases de Datos

Un sistema gestor de base de datos (SGBD) es el programa o conjunto de programas que gestionan y mantienen consistentes los datos almacenados en la base de datos. Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad. (Bachman)

PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y de software libre. (PostgreSQL. Sobre PostgreSQL)

Características:

- Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.
- Amplia variedad de tipos nativos: PostgreSQL provee nativamente soporte para:
- Números de precisión arbitraria.
- Texto de largo ilimitado
- Figuras geométricas (con una variedad de funciones asociadas).
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- *Arrays*.

1.5.7 Entorno de desarrollo integrado (IDE)

En el análisis de la bibliografía consultada se coincide en que un Entorno de Desarrollo Integrado o IDE (*Integrated Development Environment*) es un programa informático que agrupa diversas herramientas de programación para facilitar la tarea al programador y obtener mayor rapidez en el desarrollo. En otras palabras, es un sistema que facilita el trabajo del desarrollador de software, integrando sólidamente la edición orientada al lenguaje, la compilación o interpretación, la depuración, las medidas de rendimiento, la incorporación de las fuentes a un sistema de control de fuentes, etc., normalmente de forma modular. (Rehman, 2002)

Un buen IDE debe incluir las siguientes características:

- Multiplataforma.
- Soporte para diversos lenguajes de programación.
- Integración con sistemas de control de versiones.
- Reconocimiento de sintaxis.
- Extensiones y componentes para el IDE.
- Integración con *framework* populares.
- Depurador.
- Importar y exportar proyectos.
- Múltiples idiomas.
- Manual de usuarios y ayuda.

NetBeans 8

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, aunque permite programar en diferentes lenguajes. NetBeans es una base sólida para desarrollar aplicaciones complejas con un enfoque modular y posee características como la extensibilidad y la escalabilidad.

Se selecciona NetBeans en su versión 8 pues ofrece un excelente entorno y proporciona un rendimiento superior para los desarrolladores de PHP, proporcionando editores y herramientas integrales para sus tecnologías relacionadas. Este IDE es extensible, permitiendo a través de *plugings*, editar código escrito en los lenguajes XML, HTML, PHP y Java Script, debido a que es extensible. También posee un gran soporte para la edición en HTML5, además tiene un excelente balance entre una interfaz con múltiples opciones y un aceptable completamiento de código. (Domínguez-Dorado, 2005)

A continuación se resaltan otras de las características de esta herramienta:

- Interfaz de usuario para la personalización de la aplicación.
- *Framework* para la creación de interfaces de usuario.
- Internacionalización.
- Ayudas del sistema.
- Rendimiento óptimo en tiempo de ejecución y optimización de recursos.

Las ventajas de usar NetBeans son muchas, a continuación se mencionan solo las más importantes:

- Tener el respaldo de una empresa tan grande y seria como es Oracle.

- Poder usar las licencias *open source* para realizar mejoras futuras.
- Tener un respaldo online por parte de otros programadores.
- Tener un IDE que soporta el desarrollo de todos los tipos de aplicación Java.

Para el desarrollo de la propuesta de solución se selecciona NetBeans 8 debido a las características anteriormente mencionadas.

1.5.8 Servidor web

Un servidor web es un programa que se ejecuta continuamente en un ordenador, manteniéndose a la espera de peticiones por parte de un cliente (un navegador de internet) y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error.

Para su correcta ejecución, una aplicación en PHP y *Symfony* necesita alojarse en un servidor web. El servidor web que utilizaremos es el Apache.

Apache 2

El servidor Apache es un software de código abierto para plataformas Unix (BSD, GNU/Linux), Microsoft Windows, Macintosh y otras. Apache es un potente servidor web de software libre cuyo objetivo es servir o suministrar páginas web a los clientes o navegadores que las solicitan.

Entre las principales características de este potente servidor web, se destacan las siguientes:

- Apache se ha concentrado en la escalabilidad, en la seguridad y en el rendimiento.
- Es un servidor web altamente configurable.
- Plataforma de servidores web de código fuente abierto.
- Trabaja con diversas tecnologías como PHP, *mod_perl, servlets* de Java.

Se seleccionó como servidor web Apache debido a que es una tecnología gratuita, de código abierto, coincidiendo con Mohamed J. Kabir cuando refiere que "…es la plataforma de servidores Web de código fuente abierto más poderosa en el mundo…" (Kabir, 2003).

Después de vistas todas las herramientas y tecnologías existentes se decidió utilizar XP como metodología de desarrollo, dado que esta es una metodología ágil y simple, está orientada a la programación en dúos, y se cuenta con poco tiempo para la realización de la solución. Para el lenguaje de modelado se usó UML pues resulta muy útil en los procesos de desarrollo iterativo, además de ser uno de los lenguajes de modelado más usados, lo que hace que cuente con muchísima documentación. Como herramienta CASE se determinó el uso del Visual Paradigm en su versión 8, ya que además de ser muy completa y fácil de usar, presenta licencia gratuita, y con esta herramienta se cubren perfectamente los diagramas necesarios para la modelación de la solución. Como lenguajes de programación del lado del cliente se usará Java Script, html 5 y CSS3. Como lenguaje de programación del lado del servidor se usará PHP, pues además de ser gratuito, es independiente de plataforma, y posee una gran librería y documentación. Se seleccionó Symfony 2 como *framework* de desarrollo, este está basado en Modelo Vista Controlador, es flexible, y de código abierto. Como IDE se eligió NetBeans 8 que es una herramienta sin restricciones de uso, e independiente de plataforma, se determinó usar PostgreSQL por ser confiable, software libre y de código abierto y como servidor web Apache versión 2.

1.6 Conclusiones

En este capítulo se dió cumplimiento al primero de los objetivos específicos realizando un estudio que permitió la identificación objetiva de la situación actual en torno a la existencia de sistemas de gestión universitaria tanto nacionales como internacionales, donde se evidenció la falta de correspondencia entre esas soluciones y las especificidades del negocio inherente a esta investigación. Por lo que justifica la intención del presente trabajo de llevar a cabo el desarrollo de una solución para el problema planteado. De igual manera se investigó sobre los principales conceptos que se manejan, logrando un mejor entendimiento del negocio y la comunicación con el cliente. Importante también resaltar la selección de las tecnologías y la metodología de desarrollo a emplear, facilitando sentar las bases para el futuro desarrollo del software.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA.

2.1 Introducción

En este capítulo se muestra la propuesta y descripción del Sistema de Gestión de Ciencia, Tecnología, Innovación y Postgrado. También se hace referencia a las fases de Exploración y Planificación, las cuales son propias de la metodología de desarrollo que se utilizó, en la que se confeccionan la lista de reservas de productos y las historias de usuario (HU) que resultan de gran significación, puesto que son la representación de las necesidades del usuario, encargadas de la especificación de los requerimientos del sistema. En este capítulo quedarán definidos los requisitos no funcionales y se lleva a cabo la realización del plan de entrega, donde se indican las historias de usuario que se crearán para cada versión de la aplicación propuesta, así como las fechas en las que se publicarán dichas versiones. Se confecciona el plan de iteraciones donde se muestran las HU a ejecutar en cada iteración según su prioridad en el negocio.

2.2 Análisis de la gestión de ciencia tecnología innovación y postgrado en el centro GEYSED.

Actualmente la gestión de ciencia, tecnología, innovación y postgrado del centro GEYSED se maneja en una serie de documentos de distintos formatos, WORD, EXCEL, entre otros; lo que conlleva a no tener un control exacto de los avances que se han tenido en el curso ocasionando además, que la toma de decisiones se torne más lenta y compleja, provocando retrasos en la subdirección debido a la pérdida de tiempo que genera la consulta de cada uno de estos documentos por cada especialista con que cuenta dicho centro.

2.3 Objeto de automatización.

Debido a lo planteado anteriormente existe la necesidad de automatizar todos estos procesos que se realizan de manera manual, lo cual resultará en una mejora práctica, revertida en mayor fiabilidad, rendimiento y desenvoltura a la hora de ejecutar estos métodos, pues la forma en que se realizan actualmente no satisface las necesidades del centro. La finalidad de este trabajo es facilitar estas tareas que hoy se efectúan de forma manual, por lo que se automatizarán los procesos de gestión de ciencia, tecnología e innovación y todos los procesos de postgrado, así como los reportes que deben generar estos procesos.

2.4 Propuesta del sistema.

La aplicación WEB a desarrollar tiene como objetivo la gestión de todos los procesos de ciencia, tecnología e innovación, y todos los procesos de postgrado en la subdirección de Investigación y postgrado del Centro GEYSED que brinde facilidades para el análisis del cumplimiento de los diferentes indicadores permitiendo elevar el control sobre los mismos y por ende mejorar la toma de decisiones en la organización. Debe contar con un módulo para la generación de reportes, un módulo para la gestión de los profesionales donde incluya la confección de su plan de trabajo y la gestión de sus evidencias de CTI, un módulo para la gestión de proyectos que deberá incluir la asignación de proyectos a los profesionales, y otros módulos que gestionen proyecciones, las áreas y los laboratorios y los roles y los permisos en la aplicación.

2.5 Modelo de Dominio.

Debido a las características, grado de complejidad que se plantea y el conocimiento que se posee sobre el sistema se decide realizar el Modelo de Dominio o Modelo Conceptual como también se le conoce. El modelo de dominio puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. Es utilizado como un medio para comprender el sector de negocios al cual el sistema va a servir. Puede ser tomado como un punto de partida para el diseño del sistema. (Pressman, 2005)

Figura 2: Diagrama de Modelo de dominio.

2.5.1 Descripción de las entidades de Modelo de Dominio.

A continuación se muestra una tabla con la definición de las entidades que interactúan en el negocio y los principales conceptos que se tratan en el problema que se analiza.

Entidad	Descripción
Usuario	Representa la persona que interactúa con el sitio solicitando una respuesta.
Sitio Web	Es la entidad que recibe las peticiones del usuario y se encarga de procesar y devolver la solución.
LDAP	Representa al Servidor de *LDAP* usado para autenticar el usuario.
Plan de trabajo	Es la entidad que recoge toda la información de lo que debe realizar el profesional.
Reporte	Contiene la información recogida del plan de trabajo y de las proyecciones generales.
Proyecciones generales	Recoge toda la información proyectada por el Centro para los años posteriores.

Tabla 1: Descripción de las entidades que interactúan en el negocio.

2.6 Fase de exploración.

Esta es la primera fase de la metodología *XP*, en la cual, los clientes plantean a grandes rasgos las funcionalidades que son de interés para la elaboración del producto, transformándose las mismas en historias de usuario. Partiendo de la información obtenida, el equipo de desarrollo evaluará de forma general el tiempo de codificación, se familiarizará con las herramientas, tecnologías y prácticas que serán utilizadas en el proyecto, además, se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo para ello.

La fase de exploración toma de pocas semanas a pocos meses, dependiendo de la habilidad que tengan los programadores con las tecnologías seleccionadas. Es válido aclarar que las estimaciones realizadas en esta etapa son primarias, pues estas se basan en datos de alto nivel, los cuales pueden variar a medida que se analicen con mayor cuidado y detalle en las siguientes fases. (Letelier & Penadés)

2.7 Personal relacionado con el sistema.

Uno de los elementos más importantes que se deben considerar cuando se comienza el desarrollo de un sistema informático es la delimitación de la audiencia a la cual va dirigido el mismo, lo cual en algunas ocasiones, puede ser un personal relacionado con el sistema o alguien completamente ajeno a él. Debe especificarse que esta audiencia a su vez puede ser dividida en grupos atendiendo a sus necesidades.

Se define como persona relacionada con el sistema, a toda aquella que de una manera u otra interactúe con este, obteniendo un resultado de uno o varios procesos que se ejecutan en el mismo. También son considerados como personas relacionadas con el sistema, aquellas que se encuentran involucradas en dichos procesos, que participan en ellos, pero no obtienen ningún resultado de valor.

Personal	Descripción
Jefe de Centro	Tiene completo acceso al sistema y es quien puede realizar cambios en el mismo, posee el rol de Administrador.
Subdirector de Investigaciones y postgrado	Tiene completo acceso al sistema y es quien puede realizar cambios en el mismo, posee el rol de Administrador.
Jefe de Departamento	Tiene completo acceso al sistema y es quien puede realizar cambios en el mismo.

Profesionales del Centro	Los profesionales (profesores y especialistas) tienen acceso al sistema para consultar la información y para subir las evidencias de CTI, pero no pueden modificar ninguna información.

Tabla 2. Personal relacionado con el sistema.

2.8 Lista de reservas del producto

La lista de reservas del producto está formada por un conjunto de requisitos que representan las funcionalidades que el sistema debe cumplir para trabajar correctamente, siempre que estas sean las que verdaderamente desea el cliente. Es por ello que, confeccionar un listado de requisitos bien detallados y que posteriormente puedan ser probados, es de suma importancia.

R1: Autenticar usuario.

R2: Gestionar profesional.

R2.1: Adicionar profesional.

R2.2: Eliminar profesional.

R2.3: Modificar profesional

R2.4: Mostrar profesional.

R2.5: Listar profesionales.

R3: Gestionar plan de trabajo del profesional.

R3.1: Adicionar plan de trabajo del profesional.

R3.2: Eliminar plan de trabajo del profesional.

R3.3: Modificar plan de trabajo del profesional.

R3.4: Mostrar plan de trabajo del profesional.

R4: Gestionar evidencias de indicadores de CTI.

R4.1: Gestionar publicaciones.

R4.1.1: Adicionar publicaciones.

R4.1.2: Eliminar publicaciones.

R4.1.3: Modificar publicaciones.

R4.2: Gestionar participación en eventos.

R4.2.1: Adicionar participación en eventos.

R4.2.2: Eliminar participación en eventos.

R4.2.3: Modificar participación en eventos.

R4.2.4: Adicionar memorias en eventos.

R4.2.5: Eliminar memorias en eventos.

R4.2.6: Modificar memorias en eventos.

R4.3: Gestionar patentes y registros.

R4.3.1: Adicionar patentes y registros.

R4.3.2: Eliminar patentes y registros.

R4.3.3: Modificar patentes y registros.

R4.4: Gestionar evidencia de cursos de postgrado recibidos.

R4.4.1: Adicionar evidencia de cursos de postgrado recibidos.

R4.4.2: Eliminar evidencia de cursos de postgrado recibidos.

R4.4.3: Modificar evidencia de cursos de postgrado recibidos.

R4.5: Gestionar evidencia de cursos de postgrado impartidos.

R4.5.1: Adicionar evidencia de cursos de postgrado impartidos.

R4.5.2: Eliminar evidencia de cursos de postgrado impartidos.

R4.5.3: Modificar evidencia de cursos de postgrado impartidos.

R5: Gestionar escenario.

R5.1: Adicionar escenario.

R5.2: Eliminar escenario.

R5.3: Modificar escenario.

R5.4: Mostrar proyecciones del profesional.

R6: Gestionar maestría.

R6.1: Adicionar maestría.

R6.2: Eliminar maestría.

R6.3: Modificar maestría.

R6.4: Buscar maestría.

R6.5: Listar maestría.

R6.6: Asignar maestría.

R7: Gestionar cursos de posgrado.

R7.1: Adicionar cursos de postgrado.

R7.2: Eliminar cursos de postgrado.

R7.3: Modificar cursos de postgrado.

R7.4: Asignar cursos de postgrado.

R7.5: Mostrar cursos de postgrado.

R7.6: Listar cursos de postgrado.

R7.7: Buscar cursos de postgrado.

R8: Gestionar doctorado.

R8.1: Adicionar doctorado.

R8.2: Eliminar doctorado.

R8.3: Modificar doctorado.

R8.4: Buscar doctorado.

R8.5: Listar doctorado.

R8.6: Asignar doctorado.

R9: Gestionar diplomado.

R9.1: Adicionar diplomado.

R9.2: Eliminar diplomado.

R9.3: Modificar diplomado.

R9.4: Buscar diplomado.

R9.5: Listar diplomado.

R9.6: Asignar diplomado.

R10: Gestionar proyectos.

R10.1: Adicionar proyecto.

R10.2: Eliminar proyecto.

R10.3: Modificar proyecto.

R10.4: Listar proyecto.

R10.5: Buscar proyecto.

R11: Gestionar roles.

R11.1: Adicionar rol.

R11.2: Eliminar rol.

R11.3: Modificar rol.

R11.4: Asignar rol.

R12: Generar reporte de curso por fecha de inicio.
R13: Generar reporte de escenario en maestrías.
R14: Generar reporte de escenario en doctorados.
R15: Generar reporte de escenario en cambios de categoría.

2.9 Características no funcionales del sistema

Las características no funcionales del sistema son propiedades o cualidades que el producto debe tener, son cualidades del mismo que lo hacen atractivo, usable, rápido y confiable.

Requisitos de hardware: Teniendo como base los requisitos mínimos de hardware de las herramientas y tecnologías seleccionadas se plantea:

Para explotación por parte del cliente: PC Pentium IV, CPU Intel Celeron a 1.8 GHZ o superior, 512 MB de RAM o superior y 650 MB de espacio libre en disco.

Para explotación por parte del servidor: PC con CPU Intel Celeron a 1.8 GHZ o superior, 512 MB de RAM o superior y 40 GB de espacio en disco.

Requisitos de seguridad: la seguridad del sistema desarrollado está basada en niveles de acceso sobre las funcionalidades y la información manejada por el sistema. Los principios que determinan la seguridad en el sistema son los siguientes:

- La seguridad se establecerá por perfiles que serán asignados a los usuarios que interactúen con el sistema, garantizando de esta forma que la información almacenada solo sea modificada y/o visualizada por los usuarios autorizados.
- La seguridad de la base de datos será gestionada mediante la asignación de roles para mantener la integridad de los datos en función del nivel de acceso asignado. El sistema poseerá mecanismos de seguridad que garanticen los niveles de acceso de acuerdo al rol asignado por el usuario administrador.

Requisitos de usabilidad: El sistema será utilizado por personas que tienen conocimientos básicos en el manejo de las computadoras, en este caso los especialistas, el jefe de centro, el jefe de departamento y el

subdirector de investigación y postgrado del centro GEYSED de la Facultad 6. Utilizará el idioma español para los mensajes y textos de la interfaz. La navegabilidad no debe ser muy compleja, todas las funcionalidades deben ser rápidamente accesibles por el usuario. El sistema debe ser escalable para que al agregar nuevas funcionalidades no sean afectadas las que ya están funcionando.

Requisitos de confiabilidad: El sistema debe ser preciso con la información que maneja y le proporciona al usuario, haciendo énfasis en los resultados de los análisis que ejecutará, para evitar errores que puedan incidir negativamente en la toma de decisiones.

Requisitos de disponibilidad: El sistema debe estar disponible las 24 horas del día exceptuando solo los días licenciados para mantenimiento.

Requisitos de portabilidad: El sistema debe ser independiente de plataforma. Debe ejecutarse tanto en Microsoft Windows como en GNU/Linux.

Requisitos de apariencia o interfaz externa: El sistema deberá poseer una interfaz gráfica uniforme, estructurado por módulos que integran la aplicación, reuniendo todas las funcionalidades de la aplicación, a las que se pueden acceder mediante un menú desplegable, estos módulos y funcionalidades se encuentran en un panel ubicado a la izquierda en cada vista del sistema. La interfaz Incluirá solo la información y elementos necesarios que hagan intuitivo su uso. Deberá contar con una adecuada combinación de color y tipografía.

2.10 Historias de Usuario (HU)

Una historia de usuario es una descripción sencilla de una funcionalidad del software por la que el cliente va a pagar. Los detalles extra son adicionados cuando la historia de usuario es implementada, siendo estos procesos de requisitos simples y sencillos. En cualquier momento estas historias pueden romperse, remplazarse, unirse o dividirse. Entre las principales características que debe tener una buena historia de usuario se encuentran: la historia debe ser entendida por el cliente (representa un concepto y no una especificación detallada), cada historia debe devolver algún valor para el cliente, el tamaño de las historias de usuario debe ser tal que se puedan construir varias de ellas en una iteración (duración aproximada entre

1-3 semanas ideales), las historias deben ser independientes unas de otras, cada historia debe comprobarse (Stellman & Jennifer, 2014).

Respecto a la información contenida en la historia de usuario, existen varias plantillas sugeridas pero no hay un consenso al respecto. En muchos casos sólo se propone utilizar un nombre y una descripción, quizás además una estimación de esfuerzo en días. Cada característica importante debe establecerse como una historia de usuario. No hay que preocuparse si en un principio no se identifican todas las historias de usuario, puesto que el proceso de elaboración de las iteraciones conlleva una retroalimentación en sí mismo. Al principio de cada iteración estarán registrados los cambios en las historias de usuario y es a partir de ello que se planificará la siguiente. Las historias de usuario son desglosadas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración.

Se define por el cliente, la prioridad en el negocio y por el equipo, el riesgo en desarrollo de cada HU.

La prioridad en el negocio:

- **Alta:** Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.
- **Media:** Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
- **Baja:** Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

El riesgo en su desarrollo:

- **Alta:** Cuando en la implementación de las HU se consideran la posible existencia de errores que lleven la inoperatividad del código.
- **Media:** Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.
- **Baja:** Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que aporten problemas mayores para el desarrollo del proyecto.

A continuación se describen las Historias de Usuario con prioridad alta del sistema en cuestión:

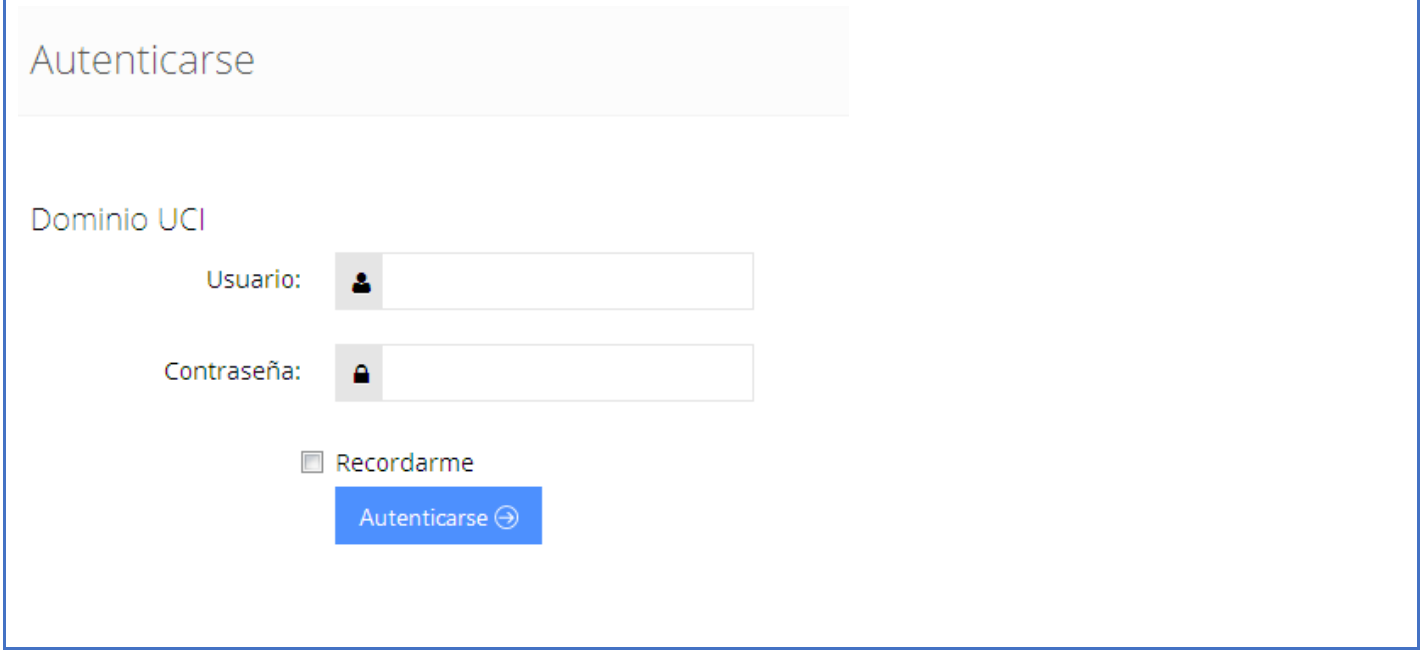
Historia de usuario	
Número: 1	**Usuario:** Todo el personal relacionado con el sistema.
Nombre de Historia de Usuario: Autenticar usuario.	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos de Estimación: 4	**Iteración Asignada:** 1
Programador(es) responsable(s): Cyndi Lorenzo, Nuvia Roque	
Descripción: Permite al usuario acceder al sistema mediante su usuario y contraseña del dominio UCI.	
Prototipo de interfaz: 	

Tabla 3: Historia de usuario no.1 Autenticar usuario.

Historia de usuario	
Número: 2	**Usuario:** Jefe de Centro, Subdirector de investigación y postgrado.
Nombre de Historia de Usuario: Gestionar Profesional.	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos de Estimación: 5	**Iteración Asignada:** 1
Programador(es) responsable(s): Cyndi Lorenzo, Nuvia Roque	
Descripción: Permite adicionar, eliminar, modificar a un profesional en el sistema. Permite también mostrar los datos del profesional y listar los profesionales.	
Observaciones: Los campos definidos serán: Nombre y Apellidos, solapín, usuario, área y proyecto.	
Prototipo de interfaz: 	

Tabla 4: Historia de usuario no. 2 Gestionar profesional.

Historia de usuario	
Número: 3	**Usuario:** Jefe de Centro, Subdirector de investigación y postgrado.
Nombre de Historia de Usuario: Gestionar plan de trabajo del profesional.	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos de Estimación: 5	**Iteración Asignada:** 1
Programador(es) responsable(s): Cyndi Lorenzo, Nuvia Roque	
Descripción: Permite adicionar, eliminar, modificar y mostrar las tareas del plan de trabajo de un profesional, por año.	
Observaciones:	
Prototipo de interfaz:	

Tabla 5: Historia de usuario no. 3 Gestionar plan de trabajo del profesional.

Historia de usuario	
Número: 4	**Usuario:** Jefe de Departamento, Subdirector de Investigación y postgrado
Nombre de Historia de Usuario: Gestionar evidencias de indicadores de CTI.	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Estimación: 8	**Iteración Asignada:** 1
Programador(es) responsable(s): Cyndi Lorenzo, Nuvia Roque	
Descripción: El usuario podrá adicionar, eliminar, modificar las evidencias de los indicadores de CTI.	
Observaciones: Esta HU se subdivide en cinco Historias de Usuario las cuales son: Gestionar publicaciones, Gestionar participación en eventos, Gestionar patentes y registros, Gestionar evidencia de curso de postgrado recibido y Gestionar evidencia de curso de postgrado impartido.	
Prototipo de interfaz:	

Tabla 6: Historia de usuario no.4 Gestionar evidencias de indicadores de CTI.

Historia de usuario	
Número: 5	**Usuario:** Subdirector de Investigación y postgrado.
Nombre de Historia de Usuario: Gestionar Escenario.	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos de Estimación: 5	**Iteración Asignada:** 1
Programador(es) responsable(s): Cyndi Lorenzo, Nuvia Roque	
Descripción: Posibilita adicionar, modificar, eliminar, y mostrar las proyecciones de un profesional, así como las proyecciones globales del Centro.	
Observaciones:	
Prototipo de interfaz:	

Tabla 7: Historia de usuario no.5 Gestionar Escenario.

Historia de usuario	
Número: 11	**Usuario:** Jefe de Centro, Subdirector de Investigación y postgrado.
Nombre de Historia de Usuario: Gestionar rol.	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos de Estimación: 4	**Iteración Asignada:** 1
Programador(es) responsable(s): Cyndi Lorenzo, Nuvia Roque	
Descripción: Permite al usuario adicionar, eliminar, modificar y asignar algún rol específico para cada usuario del sistema.	
Observaciones:	

Tabla 8: Historia de usuario no.6 Gestionar rol.

Para la duración de las semanas que se tuvieron en cuenta en las estimaciones de las historias de usuario anteriores, es necesario aclarar que una semana laboral equivale a los 5 días laborables de la misma. Generalmente se hacen cálculos erróneos que estiman los tiempos de desarrollo basándose en los 7 días de la semana.

Mes de ejemplo						
Domingo	**Lunes**	**Martes**	**Miércoles**	**Jueves**	**Viernes**	**Sábado**
1	2	3	4	5	6	**7**
8	9	10	11	12	13	**14**
15	16	17	18	**19**	20	**21**
22	23	24	25	26	27	**28**
29	30	31				

Días feriados
Fin de semana
Próximo mes
Días laborales

Tabla 9: Análisis de los días laborables de un mes de ejemplo.

2.11 Planificación.

El propósito general de la fase de planificación es que los clientes y desarrolladores se orienten en qué historias de usuario deben estar listas para la primera liberación. Esta es la fase en que los programadores alcanzan la estimación del esfuerzo necesario para la elaboración de las historias de usuario. La medida para calcular dicho esfuerzo es el punto y por lo general, un punto equivale a una semana ideal. En el caso particular de la presente investigación se mantendrá durante todo el documento como valor de un punto, un día ideal debido a las características del equipo de desarrollo y el entorno de trabajo. El tiempo ideal es aquel en el cual se escribe el código sin distracciones y con una dedicación a tiempo completo. Es en esta fase donde se obtiene un aproximado del cronograma de entrega de cada una de las liberaciones. Esta es una fase de muy corta duración, pero clave en el avance de la solución a desarrollar. (Beck, 2000)

2.11.1 Iteraciones.

Una iteración no es más que un mini-proyecto. Al finalizar una, se obtiene un resultado en software con un valor para el cliente. Pero no es hasta la última iteración que el cliente quedará totalmente satisfecho, ya que es la que finaliza y completa el producto acordado inicialmente.

Para organizar las iteraciones no es recomendable extenderlas en más de 1 mes laboral lo que sería generalmente 20 o 21 días. Los plazos de entrega y retroalimentación largos atentan contra el cumplimiento de los objetivos del cliente, sometidos a pequeños cambios de manera constante. Esto sucede, aun así, cuando un miembro de su equipo de trabajo se encuentre incluido en el de desarrollo. No hay mejor forma de evitar esto que la verificación del sistema en el entorno de despliegue del mismo. Las pruebas de los usuarios finales, las correcciones a las que se somete la aplicación en las revisiones y el análisis a partir de casos de prueba, son la mejor forma de verificar que el software avanza por el camino correcto.

Historias de Usuario	Puntos de estimación
1 Autenticar usuario.	4
2 Gestionar profesional.	5
3 Gestionar plan de trabajo del profesional.	5
4 Gestionar evidencias de indicadores de CTI.	8
5 Gestionar escenario.	5
6 Gestionar maestría.	4
7 Gestionar cursos de postgrado.	4
8 Gestionar doctorado.	3
9 Gestionar diplomado.	3
10 Gestionar proyecto.	3
11 Gestionar rol.	4
12 Generar reporte de curso por fecha de inicio.	3
13 Generar reporte de escenario en maestrías.	3
14 Generar reporte de escenario en doctorados.	3
15 Generar reporte de escenario en cambios de categoría.	3

Tabla 10: Estimación de esfuerzo por Historias de Usuario.

2.12 Plan de Iteraciones

Una vez descritas e identificadas las historias de usuario, además de estimado el esfuerzo necesario para el desarrollo de cada una de ellas, se procede a realizar la planificación de las etapas de implementación del sistema. Con este fin se define un plan que contiene las iteraciones a realizar y la relación de las historias

de usuario que serán implementadas y en qué orden para cada iteración, teniéndose en cuenta la duración de las mismas.

Una iteración es un conjunto de períodos de tiempo dentro de un proyecto, en el cual se produce una versión ejecutable del producto, estable, junto con cualquier otra documentación de soporte, instalación de scripts o similar, necesarios para usar esta liberación (Stellman & Jennifer, 2014). XP define varias iteraciones sobre el sistema antes de ser entregado, dichas iteraciones generalmente son numeradas consecutivamente y siguen una a otra de manera continua, para el presente trabajo se definieron tres iteraciones, divididas por la prioridad en el negocio.

Iteración 1

En esta primera iteración se llevará a cabo el desarrollo de las HU de la número 1 hasta la número 5 y también se implementará la HU número 11, pues son las de prioridad alta en el sistema. Al terminar la iteración esto representará un 40 % de la implementación de la aplicación.

Iteración 2

En la iteración 2 se llevará a cabo el desarrollo de las HU del número 12 hasta el número 15, estas historias son las de prioridad media en el sistema. Al terminar la iteración esto representará un 66.6 % de la implementación de la aplicación.

Iteración 3

En la iteración 3 se llevará a cabo el desarrollo de las HU del número 6 hasta la número10, que son HU de prioridad baja. Siendo esta la última iteración, se pretende entregar el producto acabado con todas las funcionalidades propuestas por el cliente. Al terminar la iteración esto representará el 100 % de la implementación de la aplicación.

2.13 Plan de entregas

El plan de entregas es el compromiso final del equipo de desarrollo con los clientes. Es una cuestión de vital importancia para el negocio entre ambas partes, ya que la entrega tardía o temprana de la solución, repercute notablemente en la economía y moral de todos los involucrados. La estimación es uno de los temas más complicados del desarrollo de un proyecto de software y es por ello que resulta de vital

importancia tener bien claros los requerimientos del cliente, el estilo de trabajo del equipo de desarrollo y el tiempo con que dispone el cliente para tener en sus manos la solución.

Nombre del Sistema	Final de la Iteración 1 (27/11/2014)	Final de la Iteración 2 (15/12/2014)	Final de la Iteración 3 (16/1/2015)
Sistema de Gestión de Información de CTI y postgrado.	Versión 0.1	Versión 0.2	Versión 0.3

Tabla 11: Plan de entregas.

2.14 Conclusiones

En este capítulo se describió la propuesta del sistema a desarrollar. Se ejecutó un análisis del flujo de los procesos para entender lo que desea el cliente. Se identificaron las funcionalidades que el sistema debe cumplir, así como la descripción de las Historias de Usuario divididas en tres iteraciones. Se realizó además, la estimación del esfuerzo dedicado a la realización de cada una de ellas, en el orden en que se les dará cumplimiento según las necesidades del cliente y finalmente con la planificación y el plan de entrega se logró delimitar el ciclo de desarrollo del sistema. La planificación del trabajo y el desarrollo por iteraciones aportan limpieza y orden al proceso de implementación del sistema, ahorrando en recursos y tiempo, obteniéndose al final de la investigación un producto de software completamente funcional y que responde a las necesidades planteadas por el cliente al inicio de la presente investigación.

CAPÍTULO 3. DISEÑO, IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA.

3.1 Introducción.

XP no propone concisamente los artefactos a utilizar en la implementación, deja en manos del equipo de desarrollo la decisión de utilizar tantos tipos de diagrama UML como crean posible y así facilitar el proceso de desarrollo. En este capítulo como parte de la metodología XP se describe la fase de diseño, implementación y prueba. Se identifican y organizan las clases relevantes para las funcionalidades del sistema, las tareas de ingeniería generadas por cada historia de usuario, las tarjetas Clase-Responsabilidad-Colaborador (CRC) así como la arquitectura y los patrones arquitectónicos empleados en el desarrollo de la aplicación. Además se realizará la descripción de diseño de base de datos y se evaluará la calidad de la aplicación a través del proceso de pruebas utilizado.

3.2 Patrón y estilo de arquitectura.

Una arquitectura de software define la estructura del sistema, la cual adquiere gran importancia debido a que las representaciones de arquitecturas de software permiten la comunicación entre todas las partes interesadas en el desarrollo de cómputo; constituye un modelo comprensible de cómo está estructurado el sistema y como trabajan juntos sus componentes. (Jacobson, Booch, & Rumbaugh, 2000). Los patrones y estilos son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios.

3.2.1 Arquitectura cliente-servidor.

En el mundo de Protocolo de Control de Transmisión y Protocolo de Internet (*Transmission Control Protocol / Internet Protocol*) TCP/IP las comunicaciones entre computadoras se rigen básicamente por lo que se llama modelo Cliente-Servidor. Este término fue usado por primera vez en 1980 para referirse a las PC en red. Esta arquitectura se divide en dos partes claramente diferenciadas: servidor y clientes. Normalmente el servidor es una PC bastante potente que actúa de depósito de datos y funciona como un SGBD. Por otro lado los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor. Ambas partes deben estar conectadas entre sí mediante una red.

Esta tecnología proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso, en múltiples plataformas.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio). Es el proceso encargado de atender a múltiples clientes que hacen peticiones. (Pressman, 2005).

Considerando la lógica de la aplicación que se desea desarrollar y analizando que el modelo cliente- servidor provee usabilidad, flexibilidad, interoperabilidad, escalabilidad en las comunicaciones, uso de entornos multiplataforma y recursos heterogéneos, se decidió utilizar este estilo arquitectónico para el desarrollo de la aplicación.

Figura 3: Arquitectura Cliente – Servidor.

Los patrones constituyen soluciones a problemas recurrentes que ocurren en el entorno. Luego de llegar a la solución se encapsulan todas las variables y factores de la misma, lo cual conforma una guía para resolver una y otra vez el mismo problema. Entre sus características se encuentran: describir el problema de forma sencilla, describir el contexto en el que ocurre, puntualizar los pasos a seguir, hacer énfasis en los puntos fuertes y débiles de la solución, referir otros patrones asociados, entre otras. En la presente investigación se abordan los patrones arquitectónicos y los patrones de diseño que son utilizados para conformar el diseño de la aplicación propuesta.

Los patrones arquitectónicos son patrones de software que se encargan de definir la estructura de un sistema. Estos a su vez se componen de subsistemas con sus responsabilidades, también poseen una serie de directivas para organizar los diferentes componentes del mismo sistema, con el objetivo de facilitar las tareas del diseño. Un patrón arquitectónico se enfoca en dar solución a un problema en específico y abarca

solo parte de la arquitectura. Aun cuando un patrón de este tipo brinda una imagen general de un sistema, él no es una arquitectura como tal. Es por esto que un patrón arquitectónico es un concepto que captura elementos esenciales de una arquitectura de software. (Gamma, Helm, Jonhson, & Vli, 2002)

3.2.2 Patrón modelo vista controlador (MVC).

El patrón Modelo Vista Controlador es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones WEB ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema de forma simple y sencilla. (Bahit, 2007)

El MVC fue introducido inicialmente en la comunidad de desarrolladores de Smalltalk-80. MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto utiliza las siguientes abstracciones:

- Modelo (*Model*): encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- Vista (*View*): muestra la información al usuario. Obtiene los datos del modelo. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- Controlador (*Controller*): Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsaciones del ratón, pulsaciones de teclas, entre otros. Los eventos son traducidos a solicitudes de servicio (***service requests*** en el texto original) para el modelo o la vista.
  El usuario interactúa con el sistema a través de los controladores.

El MVC es utilizado en múltiples *frameworks* como:

- *Java Swing*.
- *Java Enterprise Edition* (J2 EE).
- *XForms* (Formato XML estándar del W3C para la especificación de un modelo de proceso de datos XML e interfaces de usuarios como formularios WEB).
- GTK+ (escrito en C, *toolkit* creado por *Gnome* para construir aplicaciones graficas inicialmente para el sistema X Window).
- ASP.NET MVC *Frameworks* (Microsoft).
- Google WEB *Toolkit* (GWT para crear aplicaciones Ajax con Java).

- Apache *Struts* (*Framework* para aplicaciones WEB J2EE).
- *Ruby on Rails* (*Framework* para aplicaciones WEB con Ruby).

Figura 4: Modelo Vista Controlador.

3.2.3 Patrones de diseño

Un patrón de diseño es la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Este resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. Son descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un determinado contexto. Dentro de los más conocidos se encuentran los patrones de Asignación de Responsabilidades (GRASP) y los patrones de la Banda de los Cuatro (GOF). (Gamma, Helm, Jonhson, & Vli, 2002)

Patrones GRASP

Los Patrones Generales de Asignación de Responsabilidades de Software (GRASP por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, de forma tal que se pueda diseñar software orientado a objetos. Los mismos no introducen ideas novedosas, sino que son la mera codificación de los principios básicos más usados. Los patrones GRASP están compuestos por:

Experto, Creador, Bajo Acoplamiento, Alta Cohesión, Controlador, Fabricación Pura, entre otros. (Brown, Malveau, & McCormick, 1998) Muchos de estos principios básicos se tuvieron muy vigentes durante la planificación del diseño de los componentes de la aplicación propuesta.

Experto: se encarga de asignar la responsabilidad a la clase que cuenta con la información necesaria para efectuar la tarea que tiene encomendada.

Creador: tiene en cuenta para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto solo pueda ser creada por el objeto que contiene la información necesaria. El uso de este patrón permite crear las dependencias mínimas necesarias entre las clases, lo cual favorece al mantenimiento del sistema. Este patrón se evidencia en el sistema a la hora de crear un formulario, cuando el controlador va a hacer uso de un formulario.

Bajo acoplamiento: Brinda como solución asignar responsabilidades de manera que las clases no dependan fuertemente de otras. Ofrece como beneficio que son fáciles de entender por separadas, fáciles de reutilizar y no se afectan por cambios de otros componentes. Dicho patrón se tiene en cuenta debido a la importancia de realizar un diseño de clases independientes que soporten los cambios. El uso de este patrón en la aplicación se evidencia en todas las clases controladoras.

Alta cohesión: Propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de los límites manejables, para evadir un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan un bajo acoplamiento, soporta mayor capacidad de reutilización. . El uso de este patrón en la aplicación se evidencia en todas las clases controladoras.

Fabricación pura: La fabricación pura se da en las clases que no representan un ente u objeto real del dominio del problema, sino que se ha creado con toda intención para disminuir el acoplamiento, aumentar la cohesión y potenciar la reutilización del código. En la aplicación se evidencia cuando se accede a uno de los métodos del repositorio.

Patrones GOF

Los patrones de la Banda de los Cuatro (*Gang of Four* GoF), se encuentran agrupados en tres divisiones en dependencia de su responsabilidad, estas son: comportamiento, creacionales y estructurales.

Dentro de los patrones GOF, utilizados en el diseño de la presente investigación se encuentra el patrón creacional Fábrica (*Factory)* que involucra algún tipo de factoría o fábrica de objetos. Los objetos de fabricación tienen la función de crear instancias de otras clases. Además tienen la responsabilidad y el conocimiento necesario para encapsular la forma que en que se crean determinados tipos de objetos en una aplicación. Específicamente existen distintos tipos de patrones de fabricación, tales como: *simple factory, factory method y abstract factory*. (Gamma, Helm, Jonhson, & Vli, 2002)

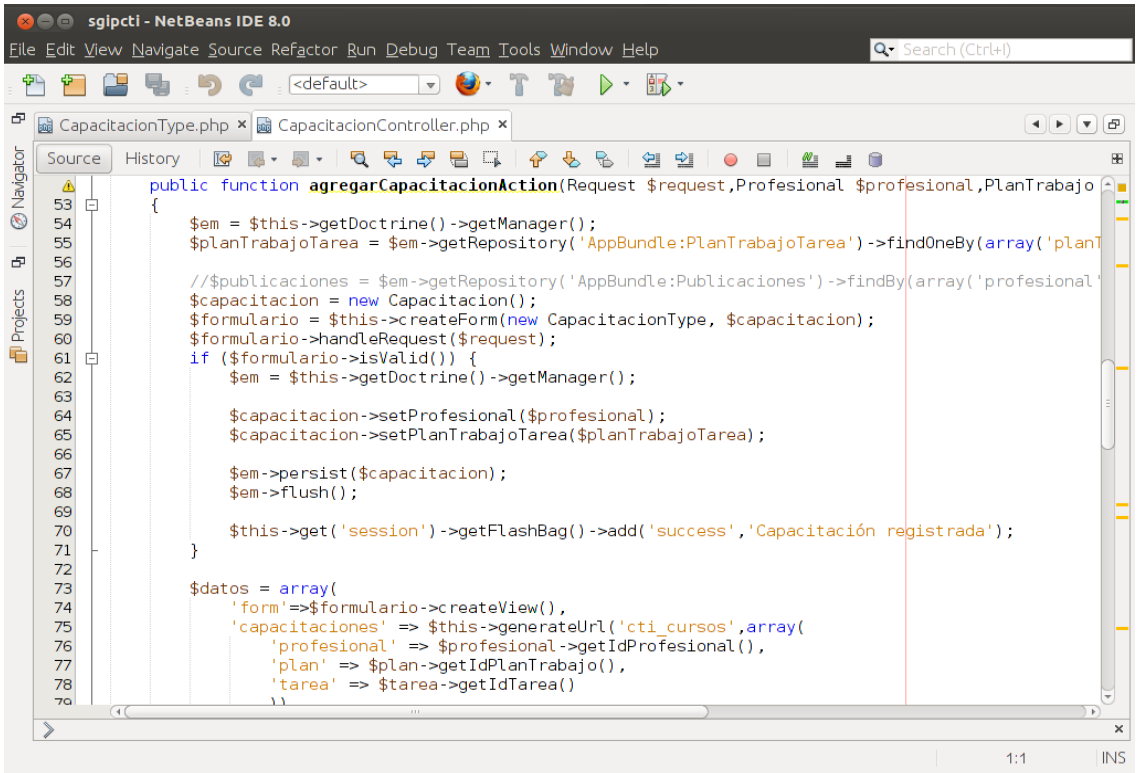


Figura 5: Patrón Fábrica.

El patrón Composición (*Composite)* se usó para la creación de vistas compuestas. Combina objetos en estructuras de árbol para representar jerarquías de parte-todo, permite tratar a cada vista compuesta igual

que a una vista normal. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos. En la aplicación se evidencia al crear campos de un formulario.

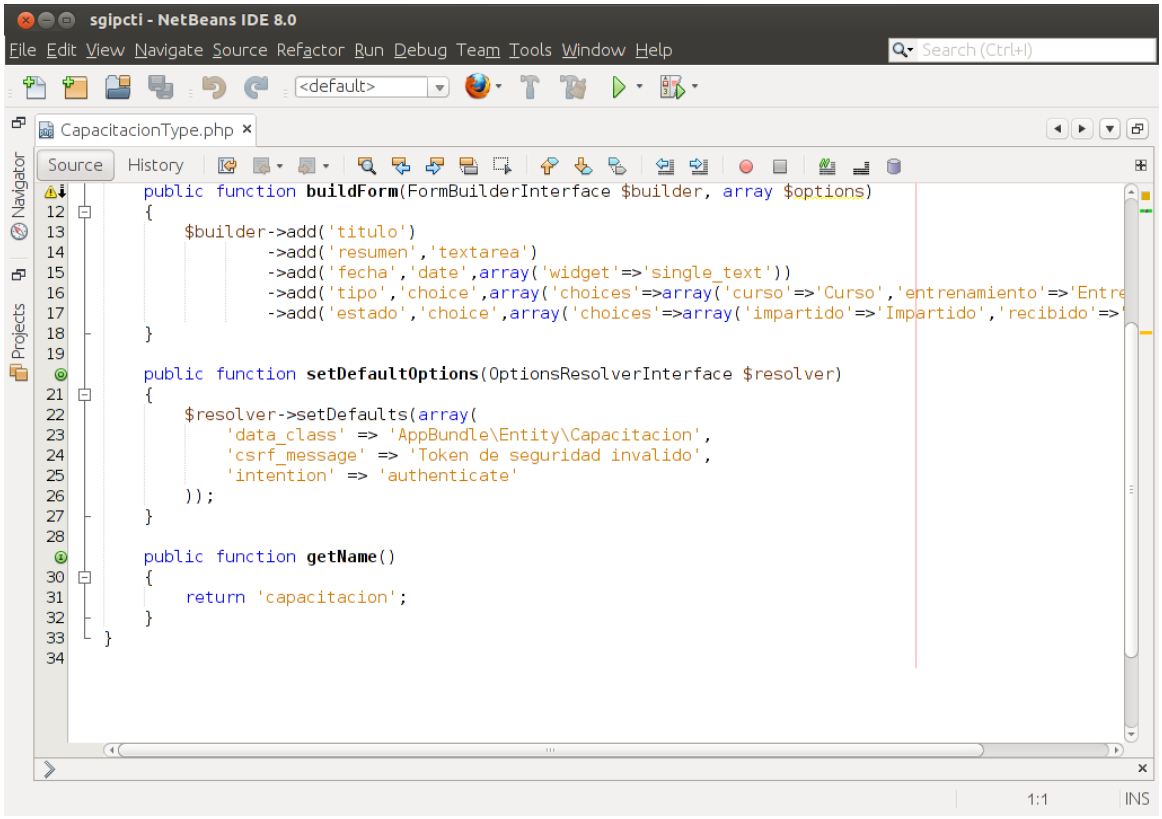


Figura 6: Patrón Composición.

Otro de los patrones empleados en la solución es el patrón Decorador (*Decorator*) el cual añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad. En la aplicación se evidencia en el boostrap del *framework* al momento de utilizar el entorno de desarrollo o de publicación en la configuración.

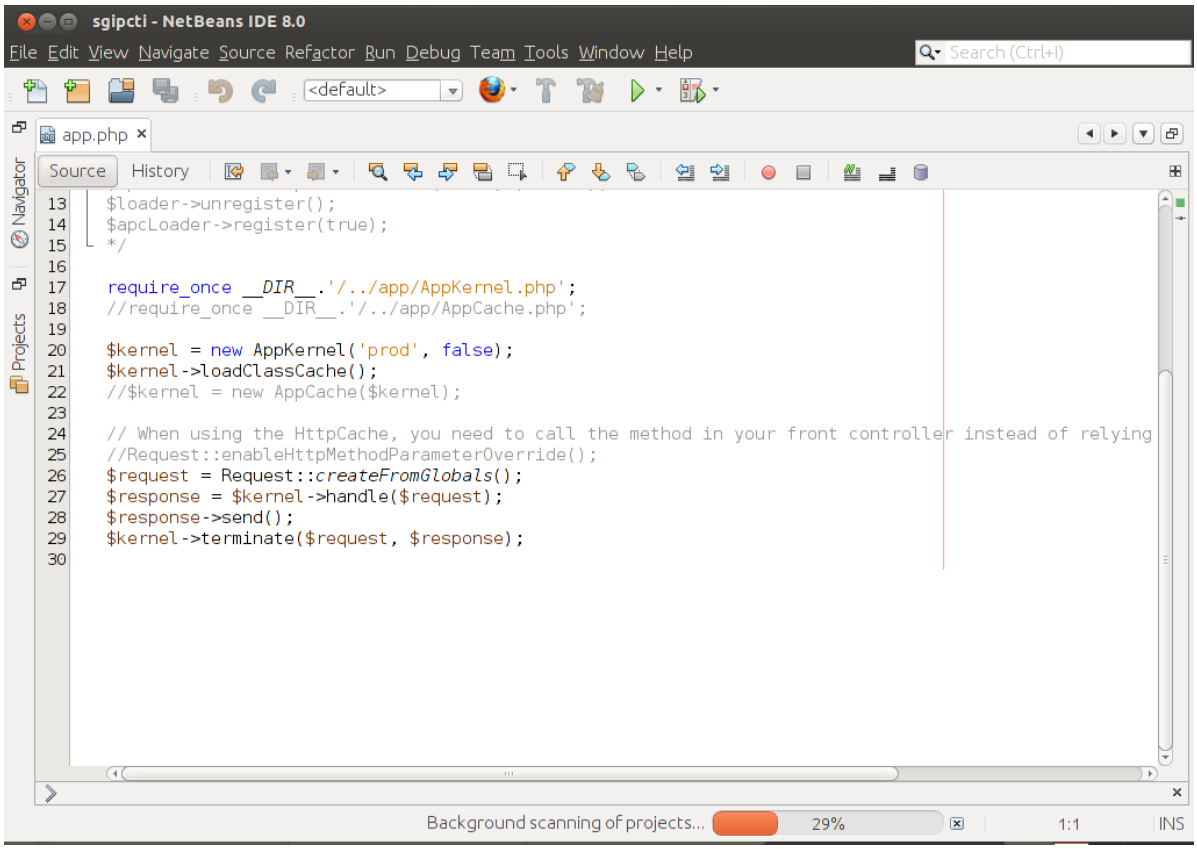


Figura 7: Patrón Decorador.

El patrón Adaptador (*Adapter*) se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda, su propósito es convertir la interfaz de una clase en otra interfaz que el cliente espera. El Adaptador permite a las clases trabajar juntas, lo que de otra manera no podría hacerlo debido a sus interfaces incompatibles. Se utilizó en la configuración al crear un formulario.

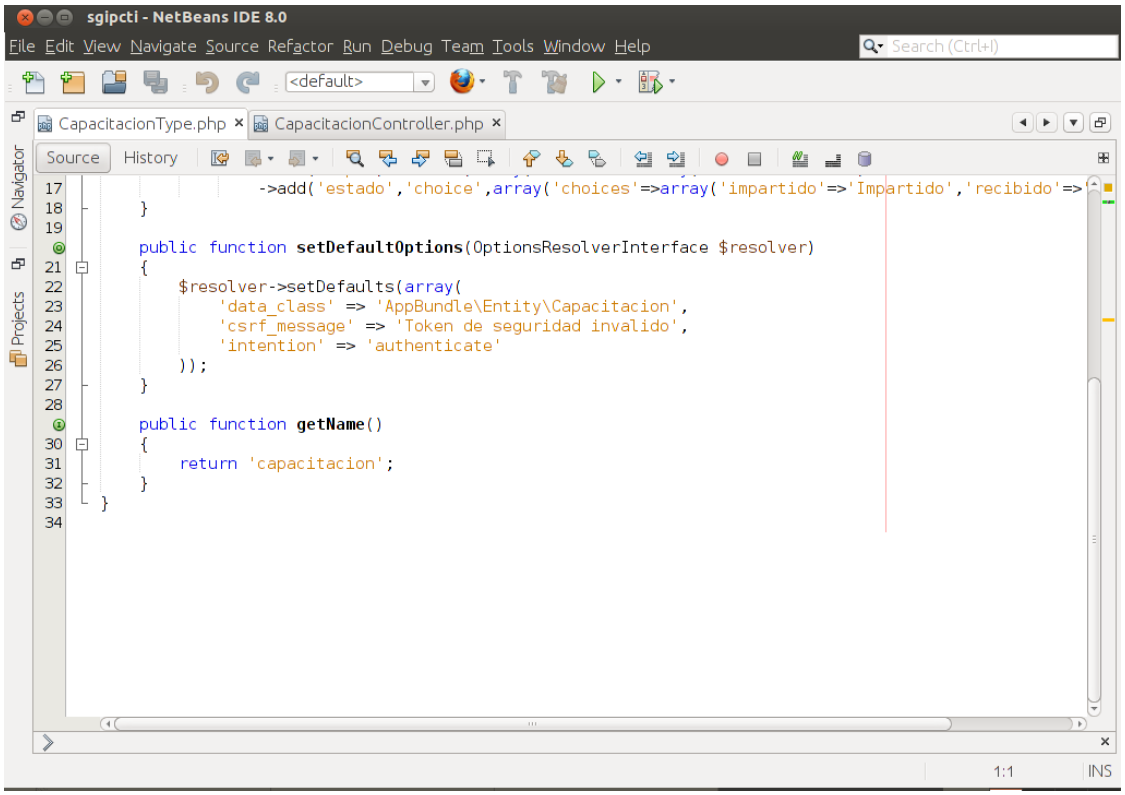


Figura 8: Patrón Adaptador.

3.3 Diseño del sistema.

Para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar la metodología XP utiliza ciertas técnicas como las tarjetas Clase, Responsabilidad y Colaboración (CRC).

3.3.1 Tarjetas CRC

Las tarjetas CRC (Clase Responsabilidad- Colaborador) son confeccionadas durante la etapa de diseño de la metodología de desarrollo XP. La misma constituye un modelo simple que tiene como objetivo desarrollar una representación organizada de las clases. Cada tarjeta representa una clase, de cada una se describe las responsabilidades que tiene y las clases colaboradoras que se relacionan con la misma.

Una responsabilidad es cualquier cosa que la clase sabe o hace. Los colaboradores son aquellas clases que se requieren para que una clase reciba la información necesaria para completar una responsabilidad. (Beck, 2000) A continuación se describen tres tarjetas CRC, el resto se encuentran visibles en los Anexos.

Clase: Login	
Responsabilidad	**Colaboración**
Verificar que los datos del usuario entrado existan en la Base de Datos. Con los datos entrados del usuario crear una sesión que persista durante el ciclo de vida de la aplicación.	Rol; Usuario.

Tabla 12: Tarjeta CRC: Clase *Login*.

Clase: Profesional	
Responsabilidad	**Colaboración**
Se encarga de la gestión del profesional.	Área; Proyecto; Plan de Trabajo; Grado Científico; Capacitación; Diplomado; Categoría Docente; Premio; Patente; Publicación; Evento; Cantidades; Asistente proyección; Auxiliar proyección; Doctor proyección; Instructor proyección; Master proyección; Titular proyección.

Tabla 13: Tarjeta CRC: Clase Profesional.

Clase: Proyección	
Responsabilidad	**Colaboración**
Se encarga de gestionar los datos de las proyecciones. Genera los reportes relacionados con las proyecciones.	Asistente proyección; Auxiliar proyección; Doctor proyección; Instructor proyección; Master proyección; Titular proyección; Cantidades;

Tabla 14: Tarjeta CRC Clase Proyección.

3. 3.2 Diagrama de Clases del Diseño.

A pesar de que la metodología XP no define diagramas de clases del diseño, se realizaron algunos diagramas para un mejor entendimiento del funcionamiento del sistema.

Un diagrama de clases del diseño es un tipo de diagrama estático que describe la estructura de un sistema, mostrando sus clases, atributos y las relaciones entre ellos. Estos diagramas son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro. Describen gráficamente las especificaciones de las clases de software y las interfaces. (Pressman, 2005)

La figura muestra el diagrama de clases del diseño correspondiente a Autenticar Usuario. En él se definen las clases que se utilizan para la autenticación de los usuarios, entre las que se encuentran la clase controladora Inicio Controller. Entre otras cosas se modela el formulario que visualizará el cliente y las relaciones entre los elementos.

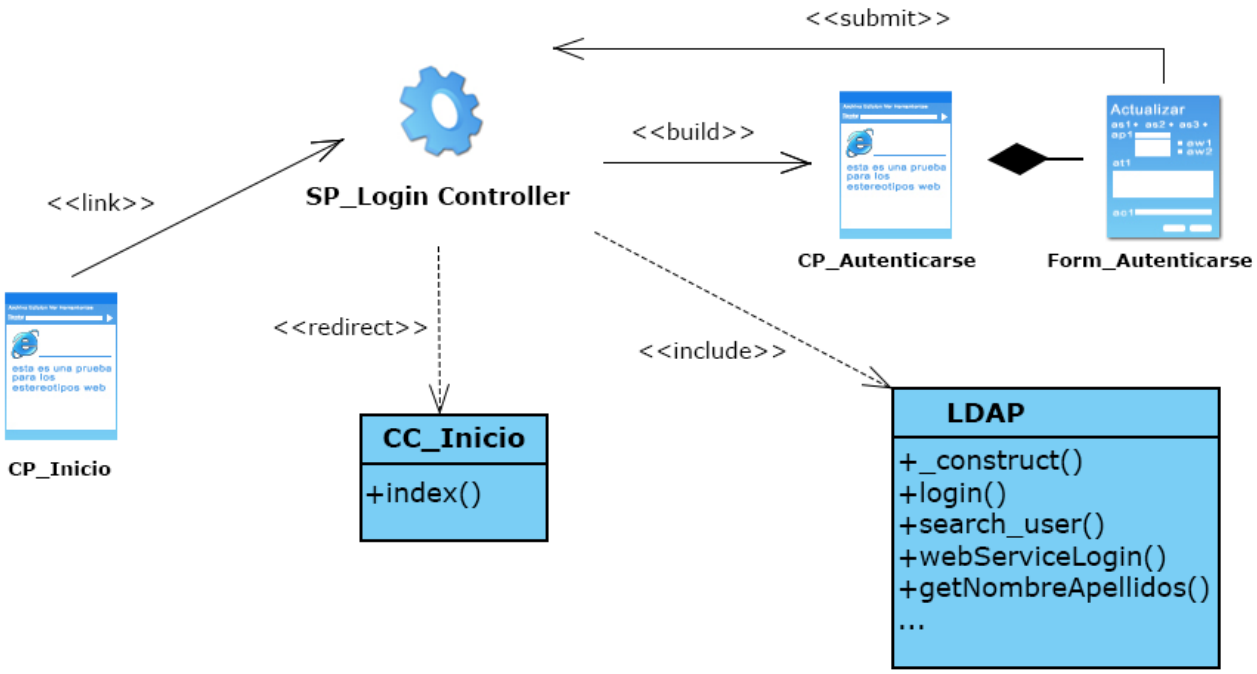


Figura 9: Diagrama de clases del diseño Autenticar Usuario.

En la figura se muestra una representación gráfica de la estructura del RF Plan de Trabajo, la que permite visualizar de forma sencilla las clases que intervienen, dígase CC_Profesional como una de las clases controladoras, las paginas servidor y cliente que interactúan con el usuario y la aplicación. En este diagrama

de clases de diseño se evidencia claramente el uso que hace Symfony del patrón arquitectónico Modelo-Vista-Controlador. El modelo es la clase de acceso a datos CAD_Profesional que incluye a la clase Profesional Repository que funciona como una consulta especializada a la base de datos haciendo uso del repositorio principal que tiene Doctrine, un mapeador de objetos relacional que tiene incorporado Symfony, por otro lado están las páginas clientes que no son más que las vistan compuestas por los formularios. Y las controladoras que manejan todo el funcionamiento de la aplicación pidiendo a las clases de acceso a datos que muestre en las vistas la información solicitada por el usuario.

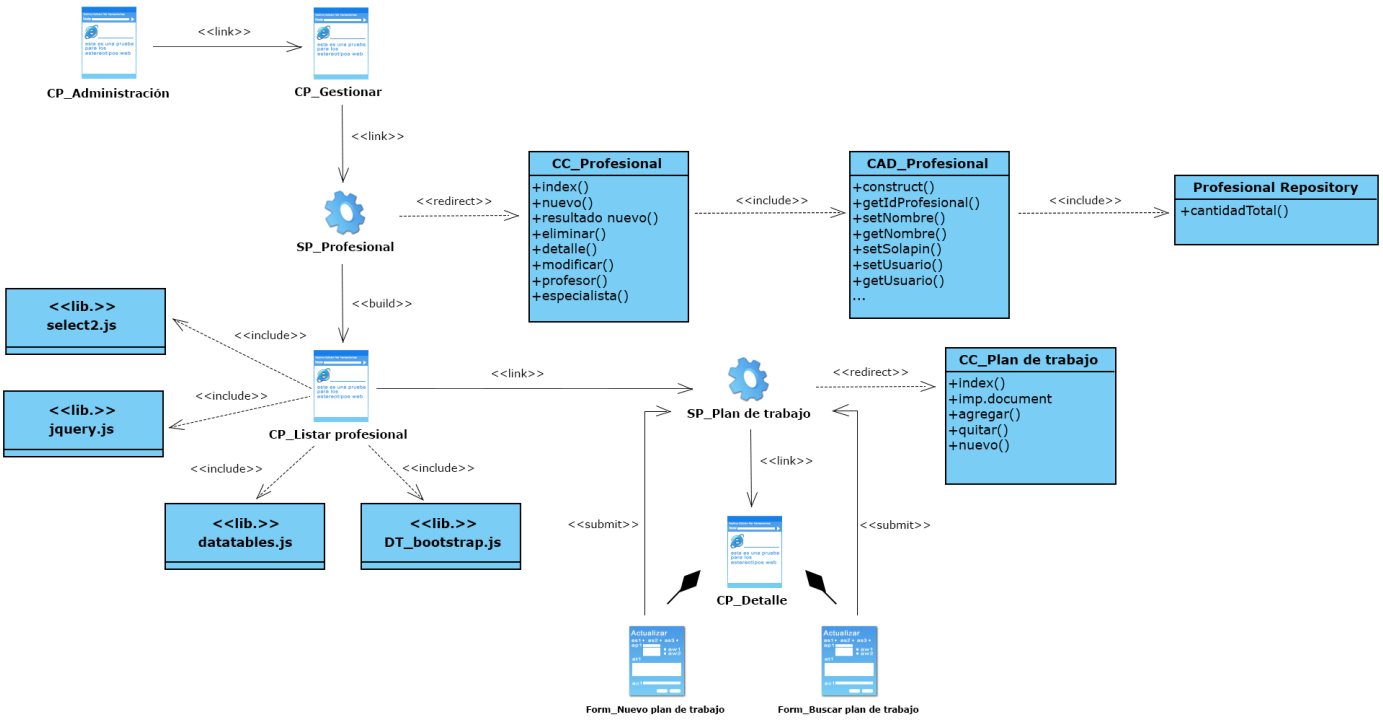


Figura 10: Diagrama de clases de diseño Plan de trabajo.

3.4 Diseño de la Base de Datos.

En cualquier sistema en el que se gestione información, la base de datos desempeña un papel fundamental. El diseño de la misma es un proceso esencial para el desarrollo de sistemas de este tipo, pues la misma

permite el almacenamiento de la información de forma coherente y organizada, evitando así perdidas e inconsistencias, de esta manera la recuperación de la información se puede realizar de forma rápida y flexible. A continuación se muestra el modelo físico de la base de datos que se utilizó, dicho diagrama se presenta fraccionado por módulos, debido al tamaño de la base de datos manejada:

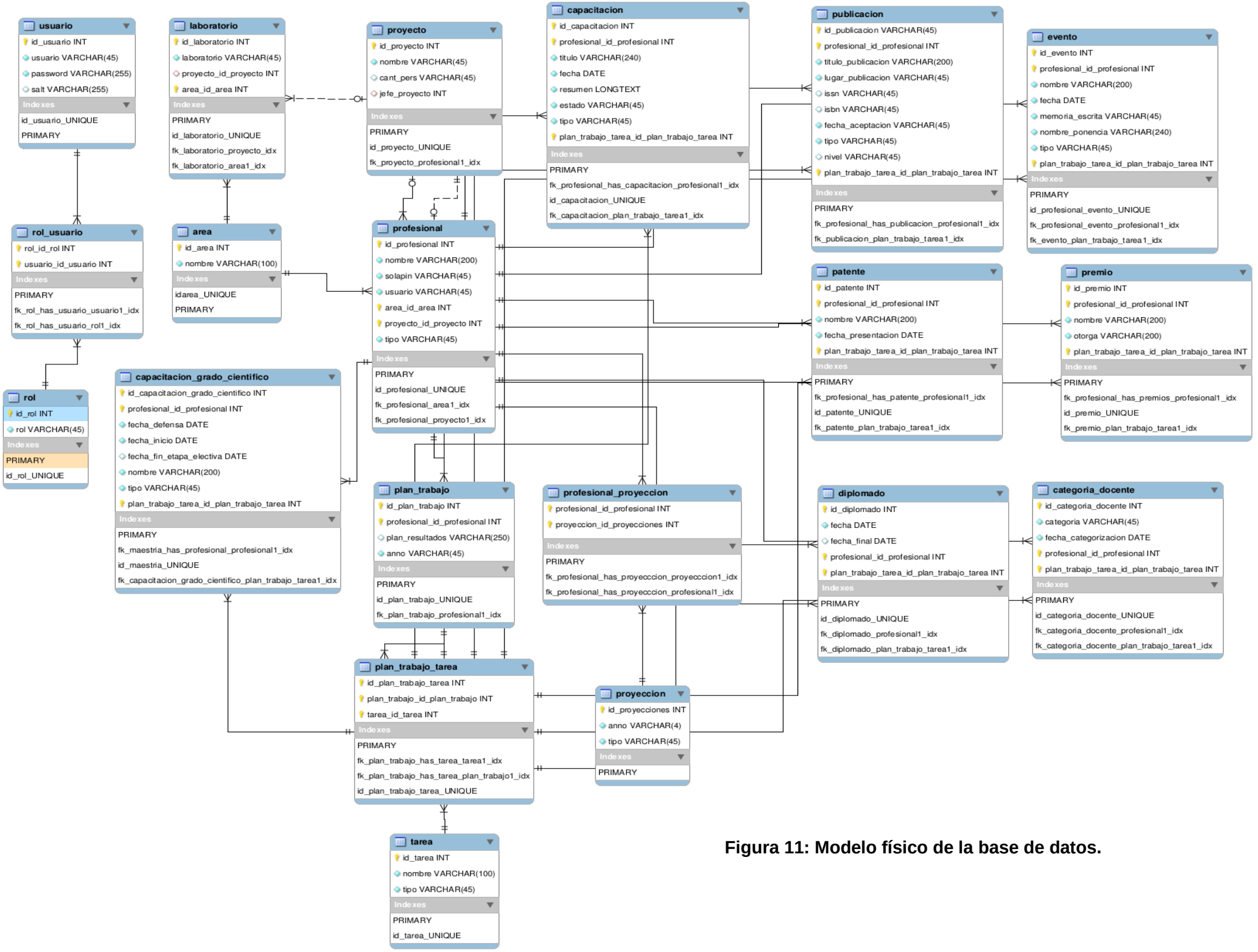


Figura 11: Modelo físico de la base de datos.

3.5 Implementación del sistema:

Para describir las tareas llevadas a cabo en la fase de implementación, se emplea un lenguaje técnico, el cual, no necesariamente debe ser entendible por el cliente. Dichas tareas son asignadas al equipo o programador responsable, normalmente la codificación se lleva a cabo por una pareja de programadores. Esta labor se lleva a cabo con el objetivo de detallar mejor las historias de usuario, lo cual facilita el entendimiento en el proceso de implementación. Cada historia de usuario puede contener una o más tareas de ingeniería, explicando de forma general las acciones que se realizan en la misma.

3.5.1 Tareas de Ingeniería.

Las tareas de ingeniería son escritas por el equipo de desarrollo a partir de las Historias de Usuario elaboradas por el cliente, brindando un detalle más profundo para realizar una implementación de las mismas y estimando un tiempo más cercano a la realidad para cada una de ellas. (Beck, 2000)

A continuación son presentadas dos de las tareas definidas para el desarrollo de la presente investigación, el resto de las tareas se encuentran en los anexos del presente documento.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:**1
Nombre Tarea: Autenticar usuario	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 4
Fecha Inicio: 14-1-2015	**Fecha Fin:** 18-1-2015
Programador(es) responsable(s): Cyndi Lorenzo Jorge, Nuvia Roque Fuentes	
Descripción: Una vez ejecutada la aplicación se muestra la presentación y una ventana de autenticación donde el usuario o el profesional deben escribir el usuario y la contraseña del dominio uci de acceso al sistema.	

Tabla 15: Tarea de Ingeniería no.1. Autenticar usuario.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Adicionar Profesional	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 19-1-2015	**Fecha Fin:** 19-1-2015
Programador(es) responsable(s): Cyndi Lorenzo Jorge, Nuvia Roque Fuentes	
Descripción: A la izquierda se encuentra el menú donde seleccionaremos listar profesionales mediante lo cual se mostrara un listado de los profesionales registrados en el sistema encima de dicho listado se visualiza un botón para agregar un nuevo profesional, este nos llevara a una página con un formulario donde se insertarán los datos del nuevo profesional.	

Tabla16: Tarea no.2 Adicionar profesional.

3.6 Diagrama de Despliegue.

El diagrama de despliegue muestra cómo interactúan los diferentes nodos que componen el sistema. En este caso el usuario realiza una petición mediante la PC_Cliente al servidor web que ejecuta las funciones que tiene implementado el sistema, accede a la base de datos, ya sea para buscar, adicionar, modificar o eliminar los datos existentes y envía respuestas de vuelta al cliente. A continuación se muestra este diagrama.

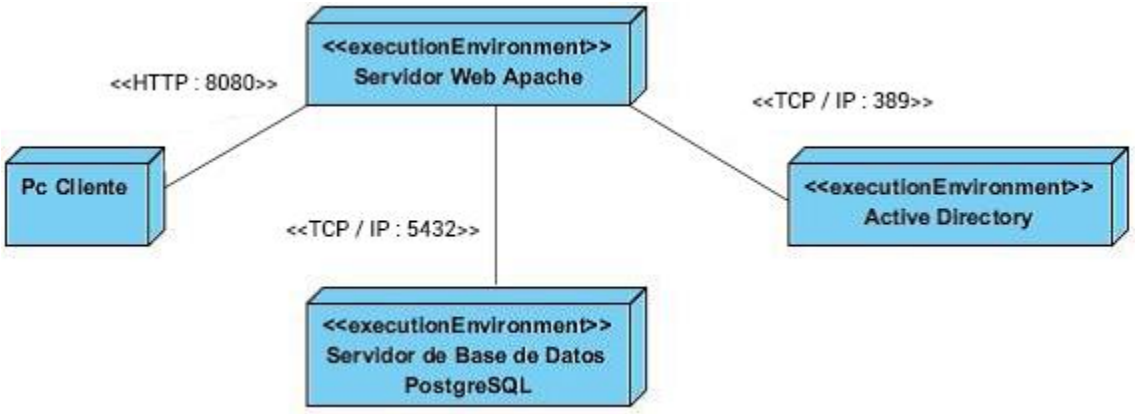


Figura 12: Diagrama de Despliegue.

Nombre del nodo: Pc_Cliente.

Este nodo hace referencia a cualquier estación de trabajo (Pc_Cliente) desde la cual se accede al sistema vía web. El servidor es el que se encarga de procesar todas las peticiones del usuario, por lo que los requisitos de hardware en este caso son mínimos.

Nombre del nodo: Servidor Web Apache.

Este nodo hace referencia al servidor de aplicaciones web Apache en el cual se aloja la aplicación, encargado de manejar todas las peticiones de los clientes.

Nombre del nodo: Servidor de Base de Datos.

Este nodo hace referencia al servidor PostgreSQL que dará soporte a la información necesaria del sistema.

Nombre del nodo: LDAP UCI. El sistema utiliza este nodo para la autenticación de los usuarios.

3.7 Diagrama de Componentes

Los diagramas de componentes muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos proporcionan y utilizan a través de las interfaces.

Describen los elementos físicos, muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos de código fuente, archivos, binarios, bibliotecas cargadas dinámicamente o ejecutables. Las relaciones de dependencia se utilizan para indicar que un componente se refiere a los servicios ofrecidos por otro componente. (Pressman, 2005)

La figura representa una vista de la estructura de la implementación de la aplicación web., con todos los paquetes, las relaciones de dependencias que se establecen entre los componentes, librerías utilizadas, así como cada una de las capas del patrón MVC integrado al *framework* Symfony.

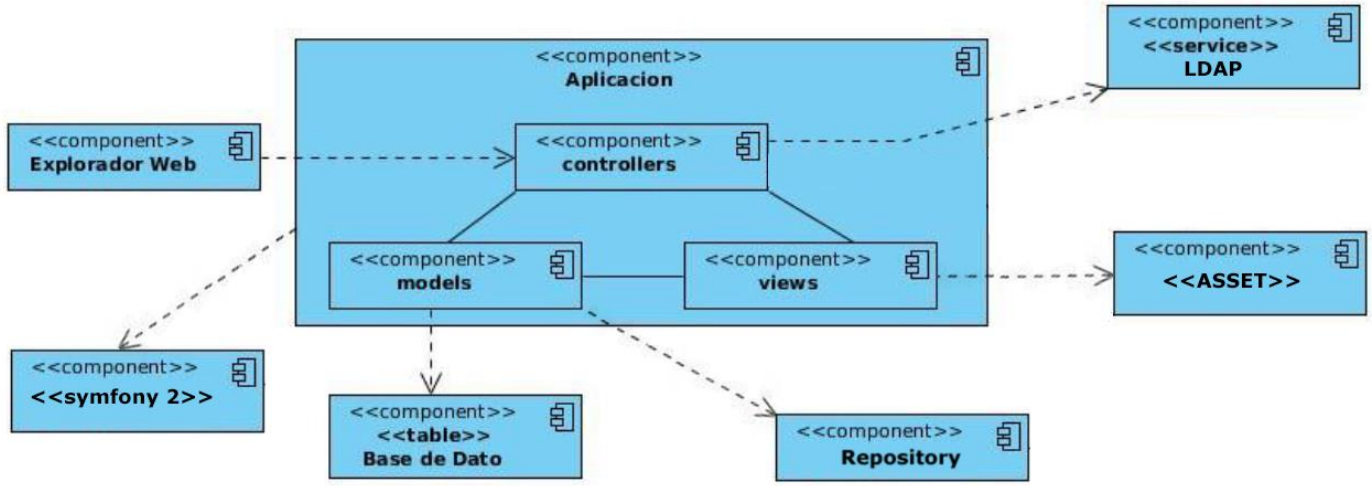


Figura 13: Diagrama de componentes.

3.8 Pruebas

La fase de las pruebas es una de las etapas fundamentales del desarrollo de una aplicación. El objetivo de cada una de las pruebas no es el de prevenir errores, sino de detectarlos basándose en técnicas y estrategias empleadas en cada una de las pruebas. (Beck, 2000) La metodología XP propone para validar las necesidades de los usuarios y dirigir la implementación las pruebas unitarias y las de aceptación.

Como ya se ha señalado anteriormente existen disimiles estrategias de pruebas, el estudio de este trabajo se ha centrado específicamente en la investigación de las pruebas correspondientes a la metodología de desarrollo de software empleada en este trabajo de diploma. XP divide las pruebas del sistema en dos

grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores y las pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

3.8.1 Pruebas unitarias

Una prueba unitaria es la verificación de un módulo (unidad de código) determinado dentro de un sistema. Son llevadas a cabo por los programadores encargados de cada módulo asegurando que un determinado modulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema. Las pruebas unitarias son una de las piedras angulares de XP. Estas pruebas deben ser definidas antes de realizar el código. Los programadores deben realizar estas pruebas cuando la interfaz de un método de la aplicación no es clara, la implementación es complicada, para probar entradas y condiciones inusuales, luego de modificar algo. Estas deben contemplar cada módulo del sistema que pueda generar fallas. En XP los programadores deben escribir las pruebas unitarias para cada módulo. Los casos de este tipo de pruebas no hay que escribirlos para todos los módulos, sino destacar en aquellos que exista la posibilidad de fallar.

Para las pruebas unitarias se usó el *PHPUnit*, el cual es uno de los principales *frameworks* de pruebas unitarias en PHP, que permite realizar las pruebas pertinentes al código, verificando que el funcionamiento de las aplicaciones PHP es el deseado encontrando bugs y errores que una vez solucionados mejorarán la calidad del desarrollo web. (Eguiluz, 2012)

3.8.2 Pruebas de aceptación.

Las pruebas de aceptación son creadas a partir de las HU. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Estas pruebas se enfocan en las características generales y las funcionalidades del sistema, son las encargadas de comprobar que las funcionalidades desarrolladas sean las esperadas por el cliente. Luego de haber superado las pruebas de aceptación podrá considerarse que la aplicación esta apta para el uso y despliegue dentro del proyecto. Las pruebas de aceptación propuestas a realizarse se encuentran divididas en las siguientes secciones para una mayor organización:

- Clases Válidas**:** describe cada uno de los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de verificar si se obtiene el resultado esperado.
- Clases No válidas: describe cada uno de los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta cada una de las posibles entradas no válidas que hace el usuario con el objetivo de verificar si se obtiene el resultado esperado y cómo responde el sistema.
- Resultado Esperado: se describe el resultado que se espera ya sea para entradas válidas o no válidas.
- Resultado de la Prueba: se describe el resultado que se obtiene.
- Observaciones: algún señalamiento o advertencia que sea necesario hacerle a la sección que se está probando.

A continuación se muestran dos casos de prueba correspondientes a dos de las funcionalidades del sistema, el resto de las pruebas de aceptación pueden ser visualizadas en los Anexos.

Clases Válidas	Clases No válidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El usuario en la interfaz inicial del sistema introduce los siguientes datos: usuario y contraseña. Presiona la opción Entrar que envía los datos para la autenticación. Luego accede en el sistema.		El sistema verifica que los datos entrados sean válidos y en caso de serlo permite el acceso al sistema.	Satisfactorio.	
	El usuario accede a la interfaz inicial del sistema y presiona la opción Entrar con los datos de usuario y contraseña	El sistema muestra un mensaje de error.	Satisfactorio.	

	incorrectos.			
	El usuario accede a la interfaz inicial del sistema y presiona la opción Entrar con los campos vacíos.	El sistema muestra un mensaje de error.	Satisfactorio.	

Tabla 16: Caso de prueba Autenticar usuario.

Clases Válidas	Clases No válidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El usuario accede a la interfaz de Gestionar profesional. Selecciona la opción Agregar profesional y entra los siguientes datos: usuario, nombre, apellidos, solapín, área y proyecto.		El sistema verifica que la entrada es válida e inserta los datos de un nuevo usuario.	Satisfactorio.	
	El usuario accede a la interfaz Gestionar profesional. Selecciona la opción Agregar profesional y luego presiona Aceptar aún con campos vacíos.	El sistema señala un error en el campo vacío.	Satisfactorio.	

	El usuario accede a la interfaz Gestionar profesional. Selecciona la opción Agregar profesional y luego presiona Aceptar con campos incorrectos.	El sistema señala un error en el campo.		

Tabla 17: Caso de prueba Gestionar profesional.

3.8.3 Registro de no conformidades

Uno de los detalles a no pasar por alto al momento de realizar las pruebas son las no conformidades, las cuales se traducen en los errores encontrados y funcionalidades no deseadas por el cliente, detectadas en cada iteración de pruebas. Al final de cada iteración se le muestra al cliente una versión funcional del software de forma que pueda detectar aquellas no conformidades que serán corregidas al inicio de la subsiguiente iteración. La presente investigación está dividida en tres iteraciones, a continuación se muestran las no conformidades encontradas en cada una de ellas:

Iteración	Descripción
Primera	1. Errores de falta de ortografía en los mensajes de la pantalla de autenticación. 2. No se especifica en el mensaje mostrado al usuario los errores que tiene el formulario al autenticarse. 3. Permite insertar números en el campo de nombre de usuario. 4. Retardo en la carga de las pestañas. 5. La aplicación no muestra los % en los gráficos de pastel. 6. La aplicación no puede autenticarse por el LDAP. 7. El usuario autenticado se puede eliminar a sí mismo. 8. En la aplicación hay texto que aparece en idioma inglés. 9. La aplicación no busca plan de trabajo por año, sino que muestra todos los planes de trabajo existentes asociados a un profesional.

Segunda	1. Error en la generación de reportes de maestrías. 2. Pérdida de datos. 3. Errores de concordancia en los mensajes. 4. Mensajes de error poco claros. 5. El formulario de publicación no actualiza y se queda cargando indefinidamente. 6. No permite cambiar la categoría de profesional a profesor o especialista.
Tercera	No se encuentran no conformidades en el sistema.

Tabla 18: Registro de no conformidades.

Como parte de la metodología XP, las no conformidades encontradas en cada iteración son las primeras tareas a resolver de la iteración siguiente, siendo el cliente el encargado de ordenarlas por prioridad. Algunas de ellas al no ser críticas, son arrastradas a la siguiente iteración. Llevando a cabo este proceso se logran minimizar los niveles de aceptación de errores. De esta manera quedaron resueltas las no conformidades detectadas en la aplicación desarrollada.

3.9 Conclusiones parciales.

El diseño arquitectónico es la base para el óptimo desarrollo de un sistema, un deficiente diseño acarrea que el proceso de ejecutar actualizaciones y mejoras al mismo supone una pérdida considerable de tiempo y otros recursos. Por este motivo la presente investigación respalda su diseño sobre patrones muy bien establecidos, los cuales fueron escogidos en dependencia de su función, tenida en cuenta a la hora de estructurar las distintas capas que componen el producto.

La metodología propuesta presenta sus métodos propios para la confección de las tareas de diseño, usando para este fin las tarjetas CRC con el objetivo de especificar las responsabilidades de cada clase presente en la lógica de negocio. La sencillez de estas tarjetas posibilita una actualización rápida en caso de cambios en la concepción del diseño. Con el objetivo de mostrar una idea de cómo fue elaborada la persistencia de la información manejada por la aplicación, se presentó el diagrama entidad relación de la base de datos utilizada.

Por otra parte las tareas de implementación ejecutadas sirvieron como soporte organizativo en el desarrollo de la codificación del software propuesto. Las mismas posibilitaron distribuir de forma organizada la programación entre los miembros del equipo de desarrollo.

Por último las pruebas permitieron establecer en tiempo los errores introducidos en la aplicación. Una de las ventajas que provee XP al efectuar las pruebas, es la posibilidad de que una vez expuesto el avance de la aplicación al cliente, este puede identificar las inconformidades existentes, las cuales el equipo de desarrollo corrige de inmediato.

Conclusiones

Se culmina la presente investigación logrando cumplir los objetivos trazados con técnicas y metodologías para cada uno de ellos. En vista de lograr la mejor aproximación a estos se puede arribar a los siguientes resultados:

- El análisis del ambiente nacional e internacional, permitió reconocer la no existencia de una solución informática que diese soporte a las necesidades planteadas por el cliente.
- La modelación y construcción de la solución propuesta viabiliza el proceso de planificación de tareas de superación de los profesionales del centro GEYSED.
- La solución es confiable en los resultados que brinda, la implementación web permite que la misma sea accedida desde cualquier lugar, independientemente de los sistemas operativos en las máquinas clientes.
- Las pruebas realizadas a la solución, validan la estabilidad y confiabilidad del sistema implementado.
- El sistema desarrollado, representa una notable mejora en el proceso de planificación de los planes de resultado de los profesionales lo que permite dar un mejor seguimiento y control del cumplimiento de los objetivos de trabajo del área reflejados en dichos planes.

Recomendaciones

Como resultado del proceso de investigación y realización de la presente investigación se sugieren aspectos que serían recomendables a tener en cuenta para el futuro perfeccionamiento del sistema:

- Adicionar un módulo que permita alertar a los responsables sobre situaciones de incumplimiento ante tareas planificadas, permitiendo que el control sea proactivo.
- Implementar un módulo de seguimiento que notifique a los usuarios cuando falta una evidencia o esté incompleta una información de CTI.

Referencias bibliográficas

Bachman, C. (s.f.). The programmer as navigator. *Communications of the ACM, 16*. doi:10.1145/355611.362534

Aguilar Ramos, C. (2005). *Aplicación de Conceptos de Gestión de Proyectos y Gestión de Riesgos en el Desarrollo de Productos Nuevos en el Campo de la Tecnología de Información.* Tesis de Maestría, Universidad de Puerto Rico. Recuperado el 20 de noviembre de 2014, de http://catedragc.mes.edu.cu/download/Tesis de Maestria/Ingeniera Industrial - Internacionales/CatherineAguilarRamos.pdf

Alvarez Muñoz, E., & Soto Pérez, J. A. (2011). *Aplicación web para la gestión de información de Investigación y Postgrado de la Facultad 4.* Trabajo de diploma, Universidad de las Ciencias Informáticas, Facultad 4, Habana. Recuperado el 3 de 11 de 2014, de http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_04591_11

Bahit, E. (2007). *El paradigma de la Programación Orientada a Objetos en PHP con el patrón arquitectónico MVC.* bubok. Obtenido de http://www.bubok.es/libros/205199/POO-y-MVC-en-PHP

Beck, K. y. (2000). *Planning xtreme Programming.* Addison-Wesley.

Brown, W. J., Malveau, R. C., & McCormick, H. W. (1998). *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis* (1st ed.). Recuperado el 10 de 12 de 2014

David, F. R. (2003). *Conceptos de administración estratégica.* Mexico: Pearson Educación.

Domínguez-Dorado, M. (Noviembre de 2005). NetBeans IDE. La alternativa a Eclipse. *Todo Programación*(13), Págs. 32-34. Recuperado el 4 de 11 de 2014

Eguiluz, J. (2012). *Desarrollo Web Ágil con Symfony2.* Recuperado el 4 de 11 de 2014

Flanagan, D., & Ferguson, P. (2002). *JavaScript: The Definitive Guide* (4.ª edición ed.). Recuperado el 1 de 11 de 2014

Gamma, E., Helm, R., Jonhson, R., & Vli, J. (2002). *Patrones de Diseño: elementos de software orientados a objetos reutilizables.* (Acebal, Trad.) Addison Wesley.

Garrett, J. J. (18 de 2 de 2005). *Adaptive Path*. Recuperado el 12 de 2 de 2015, de Ajax: A New Approach to Web Applications: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

González Soriano, G., & Martínez Consuegra, G. (2010). *Gestion Posgraduada : modulo Cursos de Posgrados : rol de Analista.* Trabajo de diploma, Universidad de las Ciencias Informáticas, Facultad 1, Habana. Recuperado el 3 de 11 de 2014, de http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_03171_10

Group., T. P. (2011). *PHP: Hipertext Preprocesor*. Recuperado el 1 de 11 de 2014, de PHP: Hipertext Preprocesor: http://php.net/manual/es/intro-whatis.php.

Investigaciones. (2012). *Indicadores de Medición de la Actividad de Ciencia y Técnica para los Centros de Educación Superior.* Documento Institucional, Universidad de las Ciencias Informáticas, Dirección de Investigaciones, Habana. Recuperado el 14 de octubre de 2014, de https://investigaciones.uci.cu/downloads.php?cat_id=2&download_id=5

Investigaciones. (2013). *Política Científica de la Universidad de las Ciencias Informáticas.* Documento Institucional, Universidad de las Ciencias Informáticas, Dirección de Investigaciones, Habana. Recuperado el 14 de Octubre de 2014, de https://investigaciones.uci.cu/downloads.php?cat_id=18&download_id=38

Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *El lenguaje unificado de modelado - Manual de Referencia.* (A. Weslley, Ed.)

Jeffries, R., Hendrickson, C., & Anderson, A. (2000). *Extreme Programming Installed.* Addison-Wesley Professional.

Jiménez, M. M., & Barrios, M. (15 de 2 de 2012). Se requiere un nuevo contrato social. *Juventud Rebelde*. Recuperado el 20 de 1 de 2015, de http://www.juventudrebelde.cu/cuba/2012-02-15/se-requiere-un-nuevo-contrato-social/

Kabir, M. J. (2003). *La biblia del Servidor Apache 2.* Anaya Multimedia.

Letelier Torres, P., Sánchez Palma, P., & Molina Marco, A. (1997). *Metodología y tecnología de la programación.* España: Universidad Politécnica de Valencia. Servicio de publicaciones. Recuperado el 25 de 10 de 2014

Letelier, P., & Penadés, M. (s.f.). *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Obtenido de http://www.cyta.com.ar/ta0502/v5n2a1.htm

Lubbers, P., Albers, B., & Salim, F. (2010). *Pro HTML5 Programming.* Springer Science+Business Media. Obtenido de http://sunshine.prod.uci.cu/book/50490b540571746019000034/

Marches, M., & otros. (2002). *Extreme Programming Perspectivas.* Addison Wesley.

MES. (2004). *Reglamento de Postgrado República de Cuba. Res. No. 132.* Habana: Ministerio de Educación Superior.

MES. (2006). *Reglamento para la Aplicación de las Categorías Docentes de la Educación Superior. Res. No. 128.* Ministerio de Educación Superior, Habana.

Oficina de Cooperación Universitaria Española (OCU). (20 de noviembre de 2014). Recuperado el 20 de noviembre de 2014, de http://www.ocu.es/portal/page/portal/inicio/sofware_gestion_universitaria

Postgrado. (2012). *Estrategia de Postgrado de la Universidad de las Ciencias Informáticas.* Documento Institucional, Universidad de las Ciencias Informáticas, Dirección de Formación Postgraduada, Habana.

Postgrado. (2013). *Balance Postgrado 2013.* Documento Institucional, Universidad de las Ciencias Informáticas, Dirección de Formación Postgraduada, Habana.

PostgreSQL. Sobre PostgreSQL. (s.f.). Recuperado el 1 de 11 de 2014, de PostgreSQL. Sobre PostgreSQL: www.postgresql.org.es/sobre_postgresql

Pressman, R. (2005). *Ingeniería del software: un enfoque práctico* (5 ed.). (J. E. Murrieta, Trad.) McGraw-Hill.

Quiles López, Á. (2012). *Desarrollo del sistema de planificación de tareas de superación Xendero.* Trabajo de diploma, Universidad de las Ciencias Informáticas, Facultad 3, Habana. Recuperado el 3 de noviembre de 2014, de http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_05480_12

Rehman, C. (2002). *The Linux Development Platform: Configuring, Using and Maintaining a Complete Programming Environment.*

Rodríguez Odales, L., & Ortíz López, L. S. (2008). *Control de la Actividad de Postgrado de la Facultad 4.* Trabajo de Diploma, Universidad de las Ciencias Informáticas, Facultad 4, Habana. Recuperado el 3 de 11 de 2014, de http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_1439_08

Rodríguez, A. (2013). *Sistema de Posgrado.* Editorial Academica Española.

SCImago. (2010). *Ranking Iberoamericano SIR 2010.* SCImago Research Group, SCImago Institution Ranking. Recuperado el 5 de noviembre de 2014, de https://investigaciones.uci.cu/downloads.php?cat_id=2&download_id=53

Spinak, E. (1998). Indicadores cienciometricos. *Ciencia de la Información, 27*(2). Recuperado el 19 de junio de 2014, de http://www.scielo.br/scielo.php?pid=S0100-19651998000200006&script=sci_arttext

Stellman, A., & Jennifer, G. (2014). *Learning Agile.* O'Reilly Media, Inc. Obtenido de http://sunshine.prod.uci.cu/book/54c155300571740399000196/

UNNE. (20 de noviembre de 2014). *http://guarani.unne.edu.ar/*. (U. N. Nordeste, Productor) Obtenido de http://guarani.unne.edu.ar/

Wilson, L. (1993). *Comparative Programming Languages* (Second Edition ed.). Addison-Wesley. Recuperado el 1 de 11 de 2014