Universidad de las Ciencias Informáticas

Facultad 3

Título: Módulos Posgrado y Categoría docente para el Sistema de gestión de información de Investigación y Posgrado del Centro de Gobierno Electrónico.

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

Autor: Yoe Reyes Redondo.

Tutores:

MSc. Isabel González Flores.

Ing. Cesar Manuel Cruzata De la cruz.

Cotutor:

Ing. Yadira García García.

La Habana, junio del 2017

DECLARACIÓN DE AUTORÍA

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

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

Yoe Reyes Redondo

_____

Firma del Autor

MSc. Isabel González Flores

_____

Firma del tutor

Ing. Cesar Manuel Cruzata De la cruz

_____

Firma del tutor

Ing. Yadira García García

_____

Firma del cotutor

AGRADECIMIENTOS

Muchas son las personas que de una forma u otra han incidido en quien me he convertido. En primer lugar, a mi mamá Ana Delis Redondo Osorio y a mi papá Dioel Reyes Viquillón, quienes, a su forma, han sabido educarme. A mi mima, por ser esa madre que todos quisieran, por estar siempre conmigo, apoyar mis decisiones en todo momento y por todas las veces que me llamó para, al menos, escuchar mi voz diciéndole que no me llamara tanto. A mi papá por siempre estar pendiente de mis avances, por cada consejo que me da, por tratar de enseñarme su visión de la vida y por comprenderme cada día más. A los dos por todo el esfuerzo y sacrificio que han realizado, no solo durante estos cinco años, sino desde que nací para que no me faltara nada. Espero algún día poder retribuirles todo el cariño y dedicación que me han brindado.

A mi abuela María por todas esas tardes que me cuidó para que mi mamá fuera a la universidad.

A mi hermano Oscar Mustelier, de quien he aprendido mucho, porque siempre me ha tenido presente en su vida, cuidándome en todo momento y ayudándome en lo que estuviera a su alcance. En especial, por traer a mi vida a Ale mi sobrino, a quien quiero como estoy seguro que querré a mis futuros hijos y que tanta alegría me da desde que nació.

A Yarilis Castillo, el amor de mi vida, mi esposa, mami, Yara o simplemente Castillo. Por llegar a mi vida y darle vueltas a gusto, por amarme, respetarme y hacerme cada día el hombre más feliz del mundo. Tú eres y siempre serás la única en mi corazón. Ansío el momento de empezar nuestra familia juntos. Te amo.

A mi abuela Agustina a quen, aunque no veo hace años, quiero mucho y a mi abuelo Eloy a quien hubiese querido ver ya siendo ingeniero.

A mis segundas madres: Odalis, Marbelis y Angelina, porque siempre me han tratado y cuidado como un hijo más. Las quiero a todas.

A Yanet Más por como se ha comportado conmigo desde que se integró a mi familia, espero que sepas que te considero una hermana.

A mi prima Maidelis por siempre preocuparse y quererme, a su hijo Lariel porque lo vengo cuidando desde que nació y es como un hermanito para mí.

A Abraham por ser ese otro hermano con quien hablar de cosas de nuestra edad y por siempre apoyarme.

A toda la familia de mi esposa por recibirme como un miembro más, en especial a Magdalena por ayudarme a mejorar mis habilidades en el dominó.

A todos mis amigos y compañeros de aula que me han ido apoyando estos cinco años de carrera. A los de mi apartamento Yairon, Dailet, El Flaco, Gabriel, Ernesto, El Teacher.

A los del grupo Generación: Trimiño, Rauly y Oscar.

A los compañeros y amigos que hice como miembro del consejo FEU: Ariel, Raciel, Polanco, Anel, Jorge, Victor y Martha.

A todos los profes que tuve, en especial a: Rosalina, Dariela, Odette, Yoan, Yalice, Rainer, Maigret, Yanet, Ailenys, Lieen, Yadira, Hilda y Pascual.

A mis tutores Isabel y Cesar por comprenderme y ayudarme a formar esta investigación que tantos dolores de cabeza dio. A mi cotutora Yadira, por toda la ayuda que en tan poco tiempo me brindó.

Al profe Yordanis que desde un inicio siempre me brindó su ayuda y apoyo.

A los profes Nemury, Eilys, Yanet y Dariela que me apoyaron y ayudaron con sus intervenciones en cada taller y fuera de estos.

A todos muchas gracias por ser parte de este capítulo de mi vida que culmina y ayudarme con sus enseñanzas a ser mejor.

DEDICATORIA

A mi mamá, mi papá y mi esposa.

RESUMEN

El presente trabajo propone una solución a las dificultades que presenta el Centro de Gobierno Electrónico para obtener la información asociada al área de Posgrado y a la gestión de las categorías docentes de forma rápida y oportuna. Este surge a partir de la necesidad de proporcionar al Sistema de gestión de la información de Investigación y Posgrado de ese centro, desarrollado en el curso 2015-2016, un conjunto de funcionalidades necesarias para disminuir las dificultades respecto a la gestión de posgrado y categorías docentes del centro. Estas funcionalidades, encapsuladas en dos módulos: Posgrado y Categoría docente, permiten gestionar las necesidades de capacitación y las evidencias, las categorías docentes, automatizar la alerta sobre la necesidad de cambio de categoría docente, entre otras. Se utiliza un enfoque ágil, definiéndose como metodología de desarrollo de software la Programación Extrema. Las tecnologías y herramientas definidas facilitaron el desarrollo y a la vez garantizaron la soberanía tecnológica por la que apuesta el país. Las pruebas y técnicas utilizadas para verificar y validar la solución permitieron comprobar el adecuado diseño e implementación de la misma, así como la satisfacción de los usuarios del sistema y el cumplimiento del objetivo de la investigación.

Palabras claves: categorías docentes, evidencias, necesidades de capacitación, posgrado

TABLA DE CONTENIDO

AGRADECIMIENTOS I
DEDICATORIA III
RESUMEN IV
INTRODUCCIÓN 1
CAPÍTULO 1: Fundamentación Teórica 6
1.1 Proceso de Posgrado 6
1.2 Proceso de categoría docente 7
1.3 Sistemas informáticos existentes 7
1.4 Metodología de desarrollo de software 10
1.5 Ambiente de trabajo 13
1.5.1 Lenguajes de programación 13
1.5.2 Marco de trabajo 15
1.5.3 ORM 17
1.5.4 Motor de plantillas 17
1.5.5 Marco de trabajo de CSS 18
1.5.6 Marco de trabajo de JavaScript 18
1.5.7 Herramientas 19
1.5.8 Servidor HTTP 22
1.6 Patrones de diseño 22
1.6.1 Patrones GRASP 22
1.6.2 Patrones GoF 23
1.6.3 Patrón de Llave Subrogada 23
1.7 Verificación del diseño 23
1.8 Estrategia de pruebas 25
1.8.1 Niveles de prueba 25
1.8.2 Métodos de prueba 26
1.9 Validación de la solución 28
1.9.1 Técnica de ladov 28
1.10 Conclusiones del capítulo 29
CAPÍTULO 2: Propuesta de solución 31
2.1 Introducción 31
2.2 Fase I: Exploración 31
2.2.1 Historias de usuario 31
2.2.2 Requisitos no funcionales 32
2.3 Fase II: Planificación de la entrega 34
2.3.1 Estimación de esfuerzos por HU 34

TABLA DE CONTENIDO

2.4 Fase III: Iteraciones 34
2.4.1 Plan de iteraciones 35
2.4.2 Plan de entregas 35
2.4.3 Modelo de datos 36
2.4.4 Tarjetas Clase - Responsabilidad - Colaborador 37
2.4.5 Arquitectura de software 38
2.4.6 Patrones arquitectónicos 41
2.4.7 Patrones de diseño 42
2.4.8 Tareas de ingeniería 44
2.5 Verificación del diseño 45
2.6 Estándar de codificación 48
2.7 Descripción de la solución 48
2.8 Conclusiones del capítulo 49
CAPÍTULO 3 Pruebas de la solución 50
3.1 Introducción 50
3.2 Fase IV: Producción 50
3.2.1 Resultados del método de caja blanca 50
3.2.2 Resultados del método de caja negra 52
3.2.3 Pruebas de aceptación 53
3.2.4 Validación de la solución 54
3.3 Conclusiones parciales 58
CONCLUSIONES GENERALES 59
RECOMENDACIONES 60
BIBLIOGRAFÍA 61

ÍNDICE DE FIGURAS

Figura 1 Modelo de datos del módulo Posgrado (Elaboración propia) 37
Figura 2 Estructura propuesta por Symfony (Elaboración propia) 39
Figura 3 Arquitectura de GINFOR (Elaboración propia) 39
Figura 4 Patrón MVC en GINFOR (Elaboración propia) 41
Figura 5 Ejemplo de patrón experto en la clase *TbOfertaPosgrado* (Elaboración propia) 42
Figura 6 Ejemplo de patrón creador en la clase controladora *TbOfertaPosgradoController* (Elaboración propia) 43
Figura 7 Ejemplo del patrón decorador en la plantilla *diagnostico_necesidades.html.twig* (Elaboración propia) 44
Figura 8 Uso del patrón Llave subrogada (Elaboración propia) 44
Figura 9 Representación de los resultados de la métrica TOC (Elaboración propia) 46
Figura 10 Representación de los resultados de la métrica RC (Elaboración propia) 48
Figura 11 Método *ObtenerSolicitudesSinEvidencia* (Elaboración propia) 50
Figura 12 Grafo de flujo del método *ObtenerSolicitudesSinEvidencia* (Elaboración propia) 51
Figura 13 Resultados de aplicar el método de caja negra (Elaboración propia) 53
Figura 14 Nivel de satisfacción de los usuarios (Elaboración propia) 55

ÍNDICE DE TABLAS

Tabla 1 Comparación entre los sistemas y los indicadores de soberanía tecnológica (Elaboración propia) 8
Tabla 2 Relación entre Akademos y los indicadores del problema de la investigación (Elaboración propia) 9
Tabla 3 Comparación entre los tipos de metodologías (INTECO, 2009) 10
Tabla 4 Comparación entre las metodologías ágiles XP, SCRUM y AUP-UCI (Mendes Calo, y otros, 2009) (Elaboración propia) 12
Tabla 5 Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC (Lorenz, y otros, 1994) 24
Tabla 6 Rangos de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC (Lorenz, y otros, 1994) 25
Tabla 7 Cuadro lógico de ladov (Kuzmina, 1970) 28
Tabla 8 Índice de satisfacción de ladov (Kuzmina, 1970) 29
Tabla 9 HU Gestionar necesidades de capacitación (Elaboración propia) 32
Tabla 10 Puntos de estimación por HU (Elaboración propia) 34
Tabla 11 Plan de iteraciones (Elaboración propia) 35
Tabla 12 Plan de entregas (Elaboración propia) 35
Tabla 13 Tarjeta CRC *TbPersonaOfertaPosgrado* (Elaboración propia) 37
Tabla 14 Tarea de ingeniería 2 (Elaboración propia) 44
Tabla 15 Tarea de ingeniería 3 (Elaboración propia) 45
Tabla 16 Tarea de ingeniería 4 (Elaboración propia) 45
Tabla 17 Métrica TOC (Elaboración propia) 45
Tabla 18 Métrica RC (Elaboración propia) 47
Tabla 19 Relación Módulo-Funcionalidad-Rol (Elaboración propia) 49
Tabla 20 Caminos por donde el flujo puede circular (Elaboración propia) 51
Tabla 21 Caso de prueba para el camino básico 1 (Elaboración propia) 51
Tabla 22 Caso de prueba para el camino básico 2 (Elaboración propia) 52
Tabla 23 Caso de prueba para el camino básico 3 (Elaboración propia) 52
Tabla 24 Caso de prueba de aceptación para la HU Gestionar necesidades de capacitación (Elaboración propia) 53
Tabla 25 Cuadro lógico de ladov para medir la satisfacción de los usuarios (Elaboración propia) 54
Tabla 26 Comparación de necesidades del cliente antes y después del uso del sistema (Elaboración propia) 55

INTRODUCCIÓN

La universidad cubana es una institución que tiene como misión principal la formación de futuros profesionales altamente calificados en la rama que cursen. A su vez ofrece posibilidades de superación a sus graduados, conocidas como posgrado o formación posgraduada, que según Manzo Rodriguez y otros (2006) permite la renovación de conocimientos y habilidades profesionales al nivel de los avances científicos, la adquisición de nuevos conocimientos no recibidos en los estudios precedentes y la profundización a un nivel superior de los conocidos.

La educación de posgrado es una de las direcciones principales de trabajo de la educación superior en Cuba y el nivel más alto del sistema de educación superior, dirigido a promover la educación permanente de los graduados universitarios. La misma puede recibirse tanto en territorio nacional como en una universidad extranjera gracias a estancias o becas otorgadas con ese objetivo, permitiendo así la retroalimentación de técnicas, conocimientos y tecnologías (Manzo Rodriguez, y otros, 2006).

En Cuba las universidades tienen un plan de posgrado sólido, que va desde cursos hasta eventos científicos y concursos, contribuyendo en la preparación y superación de los egresados. La Universidad de las Ciencias Informáticas (UCI) no se queda atrás en este tema, organizando las ediciones de verano e invierno de las Escuelas Internacionales de Posgrado y eventos como: entrenamientos, talleres y diplomados complementadas con otras formas de superación académica que se realizan son las maestrías, especialidades y doctorados.

Anualmente en la UCI se identifican los objetivos de trabajo que guían los procesos sustantivos de la universidad. Estos se desagregan jerárquicamente para su cumplimiento por facultades, departamentos, centros productivos y demás áreas de la universidad. Se organizan en cinco áreas de resultado clave (ARC), siendo la segunda la encargada de satisfacer las necesidades de formación de posgrado en el área de las ciencias informáticas (UCI, 2017).

Los resultados de cada centro productivo y departamento tributan a la facultad a la que pertenecen, los de la universidad se obtienen añadiendo los de cada facultad y las restantes áreas. Estos resultados son revisados y analizados de manera trimestral, con un corte semestral y un cierre anual, para finalmente generar un informe llamado Balance de los indicadores de CTI y Posgrado, donde se especifica el estado de cumplimiento de los objetivos de trabajo del año.

El claustro de la UCI está compuesto por profesores y especialistas que tienen amplias y diversas necesidades de formación posgraduada. Para contribuir a lograr la excelencia académica de la universidad, es un imperativo elevar el nivel científico de su claustro y ello constituye un gran reto.

Además de la formación posgraduada, los profesionales pueden categorizarse o cambiar de categoría docente. La categoría docente es una forma para la organización del trabajo del personal docente en los centros de educación superior que se divide en (Consejo de Estado, 2016):

- Una categoría docente transitoria: Instructor.
- Tres categorías docentes principales: Profesor Titular, Profesor Auxiliar y Profesor Asistente.

- Una categoría docente complementaria: Auxiliar Técnico de la Docencia.
- Dos categorías docentes especiales: Profesor Emérito y Profesor Invitado.
- Una condición docente especial: Profesor Consultante.

Inicialmente los profesionales de la universidad pueden optar por la categoría transitoria Instructor y luego hacer un cambio de categoría a Asistente, Auxiliar y Titular siguiendo ese orden. Esta información de categorización y cambios de categoría docente de los trabajadores debe ser actualizada constantemente, en dependencia de la realización de los ejercicios y requisitos correspondientes a cada categoría (Consejo de Estado, 2016).

Para mejorar la gestión del ARC "Ciencia, Tecnología e Innovación" en CEGEL, durante el curso escolar 2015-2016, se desarrolló un trabajo de diploma que dio lugar a la primera versión del Sistema de gestión de la información de Investigación y Posgrado de CEGEL (GINFOR). Sin embargo, la solución desarrollada no contempla la gestión de la información asociada al área de posgrado y las categorías docentes, así como tampoco la vinculación de estos procesos con el plan de resultados de los trabajadores del centro.

A partir del análisis documental y las entrevistas realizadas al asesor de Investigación y Posgrado, se pudo comprender el funcionamiento y las características de la gestión del área de Posgrado y las categorías docentes en el Centro de Gobierno Electrónico (CEGEL), adjunto a la facultad 3 de la UCI, se evidenciaron dificultades para llevar a cabo el seguimiento y control del cumplimiento del plan de posgrado y de categoría docente de sus profesionales, teniendo en cuenta los compromisos recogidos en el plan individual de resultados de cada trabajador, generado a partir de los objetivos de trabajo del área. Por ejemplo, la información que se genera no se encuentra accesible para ser consultada y actualizada por los trabajadores del centro.

Mediante la realización de la entrevista al asesor de Investigación y Posgrado se pudo, además, determinar que las actividades de posgrado generan un conjunto importante de evidencias, estas son enviadas por correo electrónico, generalmente, tienen un tamaño superior a un megabyte y en ocasiones no son recibidas cuando el buzón del asesor de Investigación y Posgrado está lleno o próximo a llenarse. Los dispositivos de almacenamiento externo en los que también se entregan, deben ser revisados para detectar la existencia de virus informáticos, la duración de esta tarea varía en función de la capacidad y cantidad de archivos de cada dispositivo.

Los repositorios utilizados con anterioridad para almacenar las evidencias de Posgrado han presentado problemas de seguridad, pues cada persona podía manipular la información de cualquier individuo, así como colocar la evidencia en un espacio inadecuado. Además, no se revisaban antes de ser almacenadas.

El asesor de Investigación y Posgrado utiliza un sistema de carpetas para organizar los tipos de evidencias, lo que puede dar lugar a que una evidencia se guarde en un espacio que no le corresponde, haciendo difícil su localización posteriormente. En la actualidad esta información es

registrada manualmente, dificultando la realización de un análisis de tendencias de años anteriores para la toma de decisiones.

Las deficiencias antes mencionadas hacen que la gestión de las evidencias de Posgrado se torne en un proceso engorroso, lento y poco confiable, dificultando la comprobación del cumplimiento de los planes de resultados y los objetivos de trabajo del centro. Esta situación afecta la obtención de información con rapidez y precisión con respecto a la identificación de los cursos que ha recibido cada trabajador, los que le restan por cumplir, así como el momento en el que cumple con las necesidades de formación fijadas en su plan de resultados.

En CEGEL la información asociada a la gestión de las categorías docentes de sus profesionales, como: categoría docente que poseen, fecha de categorización docente, fecha de realización de los ejercicios correspondientes y de revocación, no se encuentra registrada en un documento oficial que pueda ser consultado por directivos o trabajadores. Estos datos son necesarios para elaborar el informe de autoevaluación que tributa a la facultad en el proceso de acreditación institucional de la universidad. Además, se deben tener en cuenta para efectuar alertas tempranas para la realización de los ejercicios asociados a los cambios de categoría docente.

Lo descrito anteriormente se resume en que los reportes de cumplimiento de las necesidades de capacitación y del estado actual de los cambios de categoría docente de los trabajadores del centro no se realizan con inmediatez. La información no se encuentra estandarizada y su procesamiento de forma manual provoca demoras en el proceso de entrega del reporte o pérdida de calidad del mismo, por estar sujeto a errores. Se requiere de un seguimiento exhaustivo con el fin de realizar la alerta temprana del año en que los trabajadores deben cambiar de categoría docente y notificar que se presenten a los ejercicios complementarios para optar por la categoría docente de Asistente, Auxiliar o Titular.

Ante la problemática planteada se identifica el siguiente **problema de investigación:** ¿cómo obtener la información asociada al área de posgrado y a la gestión de las categorías docentes en CEGEL de forma rápida y oportuna[1]?

El **objeto de estudio** va dirigido al proceso de gestión de la información asociada al área de posgrado y a la gestión de las categorías docentes.

Se define como **objetivo general:** desarrollar los módulos Posgrado y Categoría docente del Sistema de gestión de información de Investigación y Posgrado de CEGEL, contribuyendo a obtener la información de forma rápida y oportuna.

Enmarcando el estudio en el **campo de acción:** informatización del proceso de gestión de la información asociada al área de posgrado y a la gestión de las categorías docentes.

Para el desarrollo de la investigación se definen los siguientes **objetivos específicos:**

[1] Oportuna: Que se hace o sucede en tiempo a propósito y cuando conviene (RAE, 2017), para contribuir a la toma de decisiones de los directivos de CEGEL.

- Elaborar el marco teórico referencial relacionado con el proceso de gestión de la información asociada al área de posgrado y a la gestión de las categorías docentes.
- Realizar el diseño de los módulos Posgrado y Categoría docente.
- Implementar los módulos Posgrado y Categoría docente.
- Verificar la solución propuesta a partir de métricas y pruebas definidas para la investigación.
- Validar la solución propuesta a partir de la técnica de Iadov y un estudio comparativo de las variables de la investigación.

Se plantea como **idea a defender:** con el desarrollo de los módulos Posgrado y Categoría docente del Sistema de gestión de información de Investigación y Posgrado de CEGEL se contribuirá a obtener información de forma rápida y oportuna de estas áreas.

Los métodos científicos utilizados en la investigación se presentan a continuación.

Métodos teóricos

Análisis y síntesis: se utiliza en la investigación para extraer los elementos más importantes de los documentos estudiados, realizando el análisis de las tecnologías, metodologías y herramientas a utilizar en el desarrollo del sistema, de esta manera fueron definidas las que contribuyen con el desarrollo de la investigación.

Análisis documental: se realizaron consultas a los documentos de CEGEL donde se registra la información asociada al posgrado y las categorías docentes de sus trabajadores, así como a la literatura especializada, con el objetivo de extraer la información necesaria para comprender el funcionamiento y las características de la gestión de estas áreas.

Histórico-lógico: se utiliza para realizar el estudio de las principales aplicaciones que se han desarrollado para la gestión de la información de Investigación y Posgrado.

Modelación: se utiliza para la realización de los diagramas necesarios en el proceso de desarrollo de software, haciendo una representación abstracta de la solución, facilitando así su creación.

Métodos empíricos:

Entrevistas: se aplica al personal consultado con el objetivo de obtener información sobre la gestión de Investigación y Posgrado en CEGEL y definir los indicadores que el cliente necesita sean cumplidos referentes a la investigación, intercambiando directamente con el personal involucrado en esta actividad.

Medición: se evidencia en la aplicación de métricas de calidad y la realización de pruebas que aseguren la calidad de los requisitos y artefactos generados en el proceso de desarrollo de software.

Estructura del documento

El presente documento está estructurado de la siguiente manera: resumen, introducción y tres capítulos de los cuales a continuación se resume su contenido. Además, cuenta con conclusiones generales, recomendaciones, bibliografía y anexos.

Capítulo 1: Fundamentación teórica, se reflejan los elementos teóricos referentes al objeto de estudio de la presente investigación, se analizan sistemas informáticos homólogos del ámbito nacional e internacional. Además, se presenta el análisis y selección de la metodología de desarrollo de software, del ambiente de trabajo y los patrones de diseño para dar solución al problema de la investigación, así como los elementos teóricos referentes a las métricas, las pruebas definidas para la presente investigación, así como la validación de la solución propuesta.

Capítulo 2: Propuesta de solución, se presentan las principales características de la propuesta de solución: funcionalidades del sistema, planificación de la entrega, plan de iteraciones, modelo de datos, arquitectura, patrones de diseño, verificación del diseño de clases y estándar de codificación.

Capítulo 3: Pruebas de la solución, se presenta la verificación y validación de la solución propuesta, aplicando la estrategia de pruebas definida para valorar el funcionamiento del sistema implementado y se validan las variables de la investigación.

CAPÍTULO 1: Fundamentación Teórica

Este capítulo trata los elementos teóricos obtenidos mediante el análisis documental, el estudio de sistemas homólogos existentes, la metodología de desarrollo de software, tecnologías, herramientas, lenguajes de programación y patrones de diseño. Además, se presentan las métricas y pruebas definidas para la investigación, así como la propuesta de validación de la solución.

1.1 Proceso de Posgrado

La flexibilidad en la adopción de formas organizativas y el rigor de la calidad de las ofertas, son características esenciales de la educación de posgrado. Las actividades de posgrado se desarrollan en diferentes modalidades de dedicación: tiempo completo o tiempo parcial y con diferentes grados de comparecencia: de forma presencial, semipresencial o a distancia. Las formas organizativas principales de la superación profesional son el curso, el entrenamiento y el diplomado. 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 (Consejo de Estado., 2004).

El curso y el entrenamiento posibilitan la formación básica y especializada de los graduados universitarios. A su vez, el diplomado 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 (Consejo de Estado., 2004).

El diplomado está compuesto por un sistema de cursos y/o entrenamientos y otras formas articuladas entre sí, que culmina con la realización y defensa de un trabajo ante un tribunal. Al concluir cualquier actividad de superación profesional de las formas definidas en los artículos precedentes, el estudiante recibe un certificado. Las evaluaciones se expresan con las calificaciones de Excelente (5), Bien (4), Aprobado (3) o Desaprobado (2). También se encuentran las formas de formación académica, que son la maestría, la especialidad y el doctorado. La calificación a otorgar en cada actividad comprendida en los programas de estudio de los programas de maestría y de especialidad de posgrado, será de 5, 4, 3 o 2 (Consejo de Estado., 2004).

Aplicándose lo mencionado anteriormente, cada año se identifican las necesidades de capacitación de los trabajadores de CEGEL y son enviadas a la Facultad, al finalizar el año se evalúa el grado de cumplimiento que se logró. Los jefes de departamento indican cuáles son las necesidades de capacitación que se deben priorizar, de acuerdo a los intereses del departamento y de CEGEL. En el caso de los recién graduados en adiestramiento se especifica en qué año deben recibir los cursos de posgrados obligatorios asociados al período de adiestramiento. Las opciones de posgrado que debe recibir cada trabajador quedan reflejadas en su respectivo plan anual de resultados.

1.2 Proceso de categoría docente

Las categorías docentes son una forma para la organización del trabajo del personal docente en los centros de educación superior. La categoría Transitoria de Instructor tiene un tiempo máximo de 5 años, transcurrido el cual, si no adquiere la categoría Principal de Profesor Asistente, pierde el derecho a continuar como profesor universitario. El rector puede autorizar, con carácter excepcional, la prórroga de este período por causas debidamente justificadas, tales como: licencia de maternidad, misión internacionalista u otras bien fundamentadas. El proceso de análisis para el otorgamiento de las categorías docentes se lleva a cabo por tribunales nombrados al efecto para la disciplina correspondiente, quienes evalúan a su vez el cumplimiento de los requisitos y los ejercicios establecidos (Consejo de Estado, 2016).

Es de interés en la presente investigación tener en cuenta como principales requisitos para el cambio de categoría docente la realización de los ejercicios de Problemas Sociales de la Ciencia y la Tecnología (PSCT) y de Lenguas Extranjeras (Consejo de Estado, 2016). El examen de PSCT, para los aspirantes a las categorías docentes principales Profesor Titular, Profesor Auxiliar y Profesor Asistente, consta de la presentación de un trabajo referativo escrito. En este trabajo el aspirante debe demostrar la capacidad de analizar su propia experiencia profesional, docente, científica, desde una perspectiva ética, política, epistemológica, social, que enfatice los nexos entre conocimiento, ciencia, tecnología, innovación y sociedad (MES., 2016).

Respecto a los exámenes de Lenguas Extranjeras, los aspirantes a las categorías docentes principales Profesor Titular y Profesor Auxiliar deben realizar el mismo tipo de examen, en el que deben demostrar el dominio de la lengua a examinar, en un nivel B1, Usuario independiente. Para obtener la categoría Docente Principal de Profesor Asistente y de la segunda lengua extranjera para Profesor Titular, el ejercicio consiste en la traducción al español de dos cuartillas de un material de la especialidad, que le permita la descripción de los acontecimientos, debiendo evaluarse la capacidad para leer y comprender artículos e informes relativos a problemas contemporáneos, de uso habitual y cotidiano (MES, 2016).

1.3 Sistemas informáticos existentes

En la actualidad existen sistemas informáticos que permiten la gestión de la información asociada al área de posgrado y a la gestión de las categorías docentes, en su mayoría desarrolladas a la medida, solo tienen en cuenta las características y restricciones propias de la institución a la que va dirigido el sistema.

En el ámbito internacional:

SIU-Araucano es un sistema de información estadística de alumnos de carreras de pregrado y posgrado para las universidades públicas y privadas argentinas (Sistema de Información Universitaria SIU, 2015). Permite la definición de la oferta académica, consulta y validación de los datos ingresados, impresión de reportes, gestión de la cantidad de estudiantes (nuevos inscritos y reinscritos) y de egresados de las ofertas académicas de pregrado, grado y posgrado (Dirección

General de Servicios Informáticos, 2013). SIU-Arauco es una aplicación web multiplataforma, sin embargo, gratuita solo para las universidades argentinas (Universidad Nacional de Formosa, 2014).

En el ámbito nacional:

El **Sistema Informático para la gestión de la formación de postgrado en los profesionales del municipio Mayarí (SICOP)** gestiona la formación de posgrado de los profesionales del municipio Mayarí y apoya la toma de decisiones de directivos del municipio con respecto a los siguientes aspectos (Otero Méndez, 2007):

- Necesidades de superación de la institución.
- Intereses personales de superación.
- Necesidades materiales para los niveles de superación planificados.
- Priorización de cursos.
- Determinación de matrículas teniendo en cuenta las necesidades.

SICOP se desarrolló usando la herramienta privativa Visual FoxPro versión 8.0 y solo se ejecuta sobre la plataforma de Windows.

A partir de la entrevista realizada al ingeniero Yasmany Tellez Collazo, Jefe de departamento de Desarrollo de la Dirección de Informatización de la UCI, que además es uno de los principales responsables del desarrollo del sistema **Akademos**, se pudieron conocer las características de este software que implementa funcionalidades de gestión de posgrado en la UCI. Este es un sistema de gestión universitaria que incluye la concepción de varias líneas de desarrollo agrupadas en las áreas: pregrado, posgrado, residencia, investigación, cooperación, teleformación, biblioteca, desarrollo, tecnologías, extensión, organizaciones y egresos. Para su desarrollo se utilizaron tecnologías libres.

Actualmente el sistema no dispone de las funcionalidades necesarias para resolver el problema de la presente investigación, pues no permite:

- El levantamiento de las necesidades de capacitación de los trabajadores de la universidad, ni su priorización.
- El registro de las evidencias de posgrado.
- Obtener reportes de todos los cursos que ha recibido cada trabajador, solamente muestra los que se recibieron desde el año 2015 en la UCI.
- Listar los cursos que debe recibir un trabajador según su plan anual de resultados.
- La gestión de categorías docentes.

En la tabla 1 se comparan los sistemas estudiados respecto a: carácter comercial, código abierto o multiplataforma, para determinar qué sistemas garantizan la soberanía tecnológica por la que apuesta Cuba.

Tabla 1 Comparación entre los sistemas y los indicadores de soberanía tecnológica (Elaboración propia)

Indicadores	SIU-Araucano	SICOP	Akademos

Código abierto	No	No	Sí
Carácter comercial	Sí	No	Sí
Multiplataforma	Sí	No	Sí

SIU-Araucano y SICOP no son sistemas de código abierto, por lo que no cumplen con el paradigma de soberanía tecnológica que sigue Cuba. Sin embargo, la gestión de cursos y notificaciones que realiza el primero, al igual que la identificación de necesidades y priorización de los cursos de acuerdo a las necesidades de superación de la institución del segundo, fueron tomadas en cuenta como funcionalidades a implementar en la presente investigación.

En cambio, Akademos es de código abierto, multiplataforma y tiene carácter comercial, aunque su uso en la UCI es libre. En la tabla 2 se presenta la relación entre Akademos y los indicadores referentes al problema de la investigación. Los mismos fueron identificados a partir de una entrevista realizada al asesor de Investigación y Posgrado de CEGEL, teniendo en cuenta las necesidades de informatización de los procesos de posgrado y categoría docente del centro.

Tabla 2 Relación entre Akademos y los indicadores del problema de la investigación (Elaboración propia)

Indicadores	Akademos
I1: Gestión de necesidades de capacitación.	Parcialmente
I2: Priorización de necesidades de capacitación.	No
I3: Notificaciones.	Sí
I4: Recepción de evidencias de posgrado recibidos dentro y fuera de la institución.	Parcialmente
I5: Listado de cursos, entrenamientos y diplomados cursados dentro y fuera de la UCI.	Parcialmente
I6: Control de tipos de posgrado (curso, entrenamiento, capacitación, diplomado, maestría y doctorado).	Sí
I7: Gestión de categoría docente.	No

Akademos solo permite la gestión de los indicadores I3, I6, I1, I4 e I5, los tres últimos de forma parcial, que permitieron identificar características y funcionalidades necesarias para resolver el problema de la investigación. Debido a que los sistemas informáticos estudiados no resuelven la situación problemática que se plantea en la presente investigación, se opta por la incorporación a GINFOR de los módulos Posgrado y Categoría docente para dar solución a la situación problemática existente.

1.4 Metodología de desarrollo de software

Según el autor Piattini Velthuis (1996) las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores en su labor. Estas rigen el proceso de desarrollo de software, con el propósito de hacerlo más predecible y eficiente, su principal objetivo es aumentar la calidad del resultado del desarrollo en cada una de sus fases. De acuerdo a sus características y los objetivos que persiguen se pueden clasificar en tradicionales y ágiles (Pressman, 2010), (Shore, y otros, 2008) y (Rivadeneira Molina, 2012).

- **Metodologías tradicionales o robustas:** centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto. Presentan altos costes al implementar un cambio y tienen una falta de flexibilidad en proyectos donde el entorno es volátil. Las metodologías tradicionales se focalizan en la planificación, control del proyecto, especificación de requisitos y en el modelado (INTECO, 2009). Su amplia utilización se debe a que son eficaces y necesarias para proyectos grandes que requieren mucho tiempo y recursos, donde la organización juega un papel fundamental para su desarrollo. Algunos ejemplos de metodologías robustas son: *Microsoft Solution Framework* (MSF) y *Rational Unified Process* (RUP).
- **Metodologías ágiles o ligeras:** son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son altamente colaborativos y se adaptan mejor a los cambios, al igual que las entregas constantes al cliente y la retroalimentación por parte de él. Tanto el producto como el proceso son mejorados frecuentemente (Navarro Cadavid, y otros, 2013). Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan (INTECO, 2009).

A continuación, se muestra una tabla comparativa entre los tipos de metodologías antes mencionados.

Tabla 3 Comparación entre los tipos de metodologías (INTECO, 2009)

Criterios	Metodologías ágiles	Metodologías tradicionales
Cambios en los requisitos	Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Control del proyecto	Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
Interacción con el cliente	El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.

Integrantes	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Generar artefactos	Pocos artefactos.	Muchos artefactos.
Roles de proyecto	Pocos roles.	Muchos roles.
Arquitectura de software	Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Debido a que se cuenta con un solo desarrollador, la investigación está dirigida a satisfacer al cliente mediante la entrega continua de software funcional manteniendo una constante comunicación entre el cliente y el equipo de desarrollo, además, teniendo en cuenta que los módulos a desarrollar no poseen un gran número de funcionalidades, se opta por utilizar una metodología ágil para su desarrollo.

Algunas de las metodologías ágiles son: SCRUM, Proceso Unificado Ágil versión UCI (AUP-UCI por sus siglas en inglés) y Programación Extrema (XP por sus siglas en inglés).

- **SCRUM**: es un marco de trabajo basado en la teoría de control de procesos empíricos. Emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. No es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos (Schwaber, y otros, 2013). Está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales (Proyectos ágiles, 2015).
- **Proceso Unificado Ágil variación para la UCI (AUP-UCI):** es una versión simplificada del RUP. Describe de manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. A su vez, la variación AUP-UCI es una adaptación de AUP para ajustarse al ciclo de vida definido para la actividad productiva de la UCI; utilizando como guía para aplicar buenas prácticas el modelo CMMI-DEV v1.3. De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes tres fases de AUP en una sola, llamada Ejecución y se agrega la fase de Cierre. Esta versión define cuatro escenarios para la disciplina de Requisitos, donde el cuarto escenario está diseñado para proyectos ágiles (Rodríguez Sánchez, 2015).
- **Programación extrema (XP):** es una metodología ágil para pequeños y medianos equipos desarrollando software cuando los requerimientos son ambiguos o rápidamente cambiantes. A diferencia de los procesos tradicionales para desarrollar software, XP asume el cambio

como algo natural. En XP se realiza el software que el cliente solicita y necesita, en el momento que lo precisa, alentando a los programadores a responder a los requerimientos cambiantes que plantea el cliente en cualquier momento. Esto es posible porque está diseñado para adaptarse en forma inmediata a los cambios, con bajos costos asociados, en cualquier etapa del ciclo de vida que consta de 6 fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte (Beck, 2005). XP utiliza, las que han demostrado ser, las mejores prácticas para desarrollar software, llevándolas al extremo. Estas son: equipo completo, planificación, test del cliente, versiones pequeñas, integración continua, propiedad colectiva, codificación estandarizada, metáforas, ritmo sostenible, diseño simple, programación en parejas, desarrollo guiado por las pruebas automáticas y refactorización (García Rodriguez, 2015).

A continuación se presenta una tabla comparativa entre las metodologías antes mencionadas (Mendes Calo, y otros, 2009).

Tabla 4 Comparación entre las metodologías ágiles XP, SCRUM y AUP-UCI (Mendes Calo, y otros, 2009) (Elaboración propia)

Criterios	XP	SCRUM	AUP-UCI
Fechas de entrega	Las iteraciones de entrega son de una a tres semanas.	Las iteraciones de entrega son de dos a cuatro semanas.	Indefinido
Valorar la interacción del equipo por sobre el proceso	Los miembros de XP trabajan en dúos.	Los miembros de SCRUM trabajan de forma independiente.	Indefinido
Valorar la respuesta al cambio	Permite introducir cambios en la iteración en curso.	Permite la evolución y el cambio, pero no es recomendable en la iteración en curso.	Permite introducir cambios en la iteración en curso.
Valorar el seguimiento de un plan	Define un plan detallado para cada iteración, que puede ser modificado.	Define un plan detallado de iteraciones, no acepta cambios durante una iteración.	Define un plan detallado para cada iteración, que puede ser modificado.
Valorar el desarrollo de software que funcione	Generar entregable con prueba satisfactoria e integrada con el resto	Generar entregable con prueba satisfactoria al finalizar cada iteración.	Generar entregable con prueba satisfactoria e integrada con el resto

	de las funciones al finalizar cada iteración.		de las funciones al finalizar cada iteración.

Debido a que se ajusta a las características de la investigación y del equipo de trabajo, compuesto por un desarrollador, en constante colaboración con el cliente, admitiendo cambios de última hora en las iteraciones, con el objetivo de favorecer las entregas rápidas de software funcional en un periodo que no excede los seis meses y la documentación que se genera es la mínima necesaria, se opta por el uso de un enfoque de desarrollo ágil, escogiéndose XP como metodología de desarrollo, ya que esta ofrece 13 prácticas que permitirán o aumentarán las posibilidades de éxito del desarrollo en el tiempo estimado con las características antes mencionadas.

1.5 Ambiente de trabajo

Como los módulos Posgrado y Categoría docente se integran a GINFOR, se utilizaron las herramientas y lenguajes definidos para el desarrollo de este sistema en su primera versión los que se presentan a continuación.

1.5.1 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar instrucciones que pueden ser llevadas a cabo por un ordenador. Son usados para crear programas que controlen el comportamiento físico y lógico de una máquina, expresar algoritmos con precisión o como modo de comunicación humana. Permiten especificar de manera precisa sobre qué datos debe operar una computadora, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo una gran cantidad de opciones posibles. Todo esto, a través de un lenguaje que intenta ser relativamente próximo al lenguaje humano o natural (Suárez, 2015). Uno de los lenguajes de programación más utilizados es PHP.

Preprocesador de Hipertexto (PHP): es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo de aplicaciones web dinámicas y que puede ser embebido en HTML[2]. El código fuente escrito en PHP es invisible al navegador y al usuario, ya que es el servidor quien se encarga de ejecutar el código y enviar su resultado HTML al navegador. Presenta una extrema simplicidad para el principiante (PHP, 2015). Según Suárez (2015) presenta una mayor capacidad de conexión con la mayoría de los gestores de base de datos que se utilizan en la actualidad. Tiene la capacidad de expandir su potencial utilizando una enorme cantidad de módulos. Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. GINFOR utiliza PHP en su versión 5.5.9, debido a su potencial para el trabajo en el entorno web.

[2] HTML: Lenguaje de marcado de hipertexto

Lenguaje de Marcado de Hipertexto 5 (HTML5): provee básicamente tres características: estructura, estilo y funcionalidad. Es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML 5: HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y Javascript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales (Gauchat, 2012). Álvarez (2009) explica que incluyen novedades significativas en diversos ámbitos. No sólo se trata de incorporar nuevas etiquetas o eliminar otras, sino que supone mejoras en áreas que hasta ahora quedaban fuera del lenguaje y para las que se necesitaba utilizar otras tecnologías, entre ellas se encuentran:

- Estructura del cuerpo: la mayoría de las webs tienen un formato común, formado por elementos como cabecera, pie y navegadores. HTML 5 permite agrupar todas estas partes de una web en nuevas etiquetas que representarán cada uno de las partes típicas de una página.
- Bases de datos locales: el navegador permitirá el uso de una base de datos local, con la que se podrá trabajar en una página web por medio del cliente y a través de una interfaz de programación de aplicaciones (API), lo que permitirá la creación de aplicaciones web que funcionen sin necesidad de estar conectados a internet.
- Fin de las etiquetas de presentación: todas las etiquetas que tienen que ver con la presentación del documento, es decir, que modifican estilos de la página, serán eliminadas. La responsabilidad de definir el aspecto de una web correrá a cargo únicamente de CSS.

JavaScript: es un lenguaje de programación que se utiliza principalmente del lado del cliente, implementado como parte de un navegador web. Técnicamente JavaScript es un lenguaje de programación interpretado por lo que no es necesario compilar el código para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Además está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, estaciones de trabajo y sobre arquitecturas distintas o con sistemas operativos diversos (Eguiluz, 2015).

Según MDN (2016), este es un lenguaje de script multiplataforma que amplía las ventajas del HTML al trabajar en entornos web, aporta dinamismo y velocidad, permitiendo realizar mejoras en la interfaces de usuario. Es pequeño y ligero, está diseñado para una fácil incrustación en otros productos y aplicaciones. Es mucho más liberado que el Java, ya que no tiene que declarar todas las variables, clases y métodos. No debe preocuparse si sus métodos son públicos, privados o protegidos y no tiene que implementar sus interfaces. Los tipos de variables, parámetros y funciones de retorno no son explícitamente definidos.

Hojas de Estilo en Cascada 3 (CSS3): es un lenguaje que trabaja junto a HTML para proveer estilos visuales a los elementos del documento como tamaño, color, fondo y borde (Gauchat, 2012).

Ofrece nuevas posibilidades para crear impacto con sus diseños. Permite el uso de hojas de estilo más diversas para variedades de ocasiones, entre otras características que harán la aplicación más amigable para los usuarios. La novedad más importante que aporta CSS3, de cara a los desarrolladores de webs, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos, que a menudo complicaban el código de las webs. Entre las principales características de CSS 3 se encuentran: bordes redondeados, con degradados, imágenes; cajas con sombra; múltiples columnas; texto con sombra y cajas redimensionales. También incluye mejoras en el uso de la opacidad (pudiéndose fijar a una caja, imagen o texto), selectores y personalizado de fuentes (CSS3, 2015).

HTML5, JavaScript y CSS3 son utilizados en conjunto en GINFOR para mejorar el trabajo en entornos web, permitiendo crear interfaces amigables para los usuarios del sistema, por lo tanto, se utilizan en el desarrollo de los módulos Posgrado y Categoría docente con el mismo propósito.

1.5.2 Marco de trabajo

En el desarrollo de software, un marco de trabajo es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto (Gutiérrez, 2016). A continuación, se presentan algunas de las características que se tienen en cuenta para elegir un marco de trabajo (Gutiérrez, 2016).

- **Persistencia de datos:** la mayoría de los marcos de trabajo brindan soporte para el trabajo con múltiples Sistemas Gestores de Base de Datos (SGBD), principalmente MySQL, PostgreSQL y Oracle. Una condición que potencia el uso del marco de trabajo es precisamente el hecho de que soporte un mayor número de SGBD y que permita al desarrollador abstraerse del gestor dotándolo de una API consistente y unificada, donde sin modificar el código fuente pueda estar interactuando con cualquier gestor de los que el marco de trabajo soporta. Además, es habitual que los marcos de trabajo contengan herramientas que posibiliten el mapeo de tuplas a objetos, lo que se conoce como ORM (Mapeo de Objetos Relacionales) (Gutiérrez, 2016).
- **Motores de plantillas:** un marco de trabajo puede brindar uno o más motores o sistemas de plantilla, que puede ser tan simple como el reemplazo de palabras claves, o un poco más robusto como para generar automáticamente contenido como campos de un formulario. El contenido puede provenir directamente de una base de datos. La sintaxis de la plantilla varía según el marco de trabajo, pero debe diferenciarse del HTML y de las variables a expandir. Los sistemas de plantillas ayudan a separar la lógica de negocio de la de presentación, lo que es siempre considerado una buena práctica (Gutiérrez, 2016).

- **Gestión de usuarios:** a diferencia de los sitios estáticos, en los que cualquier usuario es considerado por igual como anónimo, en las aplicaciones web se requiere restringir el acceso a sus contenidos, es por ello que se considera una característica prominente de los marcos de trabajo que posean cuentas de usuario genéricas y extensibles, de manera que estos puedan registrarse, acceder a los contenidos acorde con su nivel de privilegio y cambiar sus contraseñas de acceso (Gutiérrez, 2016).
- **Trabajo con caché[3]:** para mejorar el rendimiento de un sistema web, los desarrolladores a menudo se auxilian de la caché para almacenar cierto contenido de uso muy frecuente, de modo que este no tenga que generarse ante cada petición de la página. Los marcos de trabajo brindan un mecanismo para almacenar dicho contenido, ya sea en la base de datos o en el sistema de ficheros. El uso de la caché reduce considerablemente el consumo de ancho de banda y la recarga del servidor (Gutiérrez, 2016).
- **Seguridad:** en ocasiones las páginas web se conciben para estar accesibles a usuarios autenticados, de ahí que los entornos de trabajo permitan verificar y solicitar autenticación antes de conceder el acceso a un determinado recurso o contenido, característica que combinan con la gestión de usuarios (Gutiérrez, 2016).

Symfony es un marco de trabajo de PHP para el desarrollo de aplicaciones web rápidas y seguras. Su código, y el de todos los componentes y bibliotecas que incluye se publican bajo licencia MIT *(Massachusetts Institute of Technology)* de software libre. Cuenta con un sitio online en el cual se puede encontrar una gran cantidad de documentación, de forma gratuita. Está diseñado para explotar las potencialidades de PHP 5.3, lo cual supone mejoras en el rendimiento del marco de trabajo. Su arquitectura está totalmente desacoplada, esto permite eliminar aquellos componentes que no se necesiten para el desarrollo. Está diseñado para que se ajuste a los siguientes requisitos (symfony.es, 2016):

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Preparado para aplicaciones empresariales y adaptables 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 y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.

[3] Caché: conjunto de datos duplicados de otros originales, con la propiedad de que los datos originales son costosos de acceder, normalmente en tiempo, respecto a la copia en el cache (Avendano, 1981).

Symfony en su versión 2.6 fue utilizado como marco de trabajo en el desarrollo de GINFOR por lo que se continuará con el uso del mismo. Este reúne las mejores prácticas de desarrollo web e integra bibliotecas y herramientas que aportan fortaleza al producto final. Las herramientas de generación de código, la interfaz de línea de comando para la instalación del sistema desarrollado y otras tareas comunes automatizadas, convierten a Symfony en un potente marco de trabajo para PHP (symfony.es, 2016).

1.5.3 ORM

ORM es un componente de software que permite trabajar con los datos persistentes como si ellos fueran parte de una base de datos orientada a objetos (tuProgramacion, 2015). Los ORM son elegidos en dependencia del marco de trabajo y del lenguaje de programación que se utilice. Doctrine es el ORM que utiliza Symfony por defecto y fue el utilizado en el desarrollo de GINFOR. Dentro de sus características están (Doctrine, 2016):

- Permite todas las operaciones habituales de CRUD (Crear, Modificar, Eliminar y Actualizar).
- Crear un nuevo registro para actualizar las existentes.
- Generar las clases PHP.
- Se basa en YAML[4].
- Apoya diferentes motores de bases de datos.
- Contiene la validación de datos en los modelos y apoya a la herencia simple.
- Es portable, lo que le permite integrarse fácilmente a cualquier proyecto.
- Tiene un lenguaje de consulta propio para ayudar a la extracción de los objetos de la base de datos que entiende las relaciones entre clases.
- Es aconsejado por los creadores de Symfony 2 (Eguiluz, 2016).

Por lo antes expuesto, se utiliza Doctrine 2 como ORM de Symfony

1.5.4 Motor de plantillas

El uso de un motor de plantillas les permite a los desarrolladores separar el código PHP del código HTML, lo cual hace más sencillo el desarrollo de la interfaz del sistema. Debido a esto, para el desarrollo de los módulos se continuará con el uso de Twig como motor de plantilla, pues junto al hecho de ser rápido y eficiente, se le atribuye la condición de haber sido creado por el propio Fabien Potencier, autor y máximo responsable de Symfony, quien recomienda a Twig para el uso del marco de trabajo. Twig se caracteriza fundamentalmente por ser (Potencier, 2011):

- **Rápido:** compila las plantillas hasta código PHP regular optimizado. El costo general en comparación con código PHP regular se ha reducido al mínimo.

[4] YAML es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl y otros (BEN-KIKI, y otros, 2005).

- **Seguro:** tiene un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable. Esto permite utilizar Twig como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla.
- **Flexible:** es alimentado por flexibles analizadores léxico y sintáctico. Esto permite al desarrollador definir sus propias etiquetas y filtros personalizados, así como crear su propio DSL (Lenguaje Específico del Dominio).

Para la generación de las vistas, se utiliza Twig en su versión 1.0, puesto que los creadores de Symfony 2 recomiendan su utilización para la creación de las plantillas de la aplicación. Además, ofrece la herencia entre plantillas, que permite crear una estructura base que contenga todos los elementos comunes del sitio y define los bloques que las plantillas descendientes pueden sustituir, garantizando así organización en el código y facilidad de mantenimiento.

1.5.5 Marco de trabajo de CSS

Bootstrap: es un marco de trabajo desarrollado para crear interfaces de aplicaciones web y fomentar la consistencia a través de herramientas internas. Los diseños implementados con el marco son intuitivos, fáciles y ligeros, lo que propicia una mejora notable para adaptarse a casi cualquier dispositivo como: teléfonos y televisores inteligentes, tabletas y computadoras. Utiliza CSS3 y JQuery permitiendo un mejor manejo de eventos y animaciones en la aplicación (Thornton, y otros, 2012). Este marco de trabajo presenta las siguientes ventajas (Borillo, 2012):

- Ofrece una serie de plantillas CSS y ficheros JavaScript que facilitan la integración del marco de trabajo de forma sencilla y potente en los proyectos web.
- Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como tabletas y móviles a distintas escalas y resoluciones.
- Se integra perfectamente con las principales bibliotecas de JavaScript, por ejemplo, JQuery.
- Ofrece un diseño sólido usando estándares como CSS3/HTML5.
- Es un marco de trabajo ligero que se integra de forma sencilla con el proyecto actual.
- Funciona con todos los navegadores, incluido Internet Explorer usando HTML para que reconozca las etiquetas HTML5.
- Dispone de distintas capas redefinidos con estructuras fijas a 940 píxeles de distintas columnas o diseños fluidos.

GINFOR utiliza Bootstrap en su versión 3.0 como marco de trabajo para la creación de interfaces, por lo que se continuará el uso del mismo en el desarrollo de las interfaces de los módulos de la investigación.

1.5.6 Marco de trabajo de JavaScript

JQuery es una biblioteca JavaScript rápida, pequeña y rica en funciones. Hace manipulación y recorrido de documentos HTML, control de eventos, efectos y animaciones personalizadas, selección de elementos del Modelo de Objetos del Dominio (por sus siglas en inglés DOM) y

manipulación de la hoja de estilos CSS, con una interfaz de programación de aplicaciones (API) que funciona a través de muchos navegadores. Brinda varias utilidades como obtener información del navegador, operar con vectores y funciones para rutinas comunes (The Apache Software Foundation, 2016).

JQuery en su versión 1.8 como biblioteca Javascript, ofrece una infraestructura que facilita la creación de aplicaciones complejas del lado del cliente. A la vez ayuda en la creación de interfaces de usuario, efectos dinámicos y en el uso de Ajax, permite la abstracción casi total del uso de este. Por otra parte, garantiza la creación de todo tipo de funcionalidades en el lado cliente de manera sencilla, y favorece en el rendimiento por el manejo rápido de propiedades y CSS.

Debido a las posibilidades que este brinda, se utiliza Jquery para la creación de funcionalidades sencillas del lado del cliente que favorezcan la rápida respuesta de los módulos al no tener que esperar una respuesta desde el servidor.

1.5.7 Herramientas

Lenguaje Unificado de Modelado

Según Wesley y otros (2000) el Lenguaje Unificado de Modelado (UML) permite modelar, construir y documentar los elementos que forman un producto de software que responde a un enfoque orientado a objetos. UML no es un lenguaje de programación sino un lenguaje de propósito general para el modelado orientado a objetos y también puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes. Intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común que cualquier desarrollador será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama. Dentro de las ventajas que proporciona este lenguaje de modelado se destacan las siguientes (Wesley, y otros, 2000):

- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyendo así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo.
- Cubre las cuestiones relacionadas con el tamaño propio de los sistemas complejos y críticos.
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

- UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

Herramienta CASE

Visual Paradigm para UML: es una herramienta muy completa, fácil de usar y con soporte multiplataforma. Su uso es sencillo para la creación de todo tipo de diagramas en UML, para los que dispone de un número considerable de estereotipos, permitiendo mayor entendimiento de los mismos (INEI, 2015). Según lo especificado en Visual Paradigm (2000), este permite visualizar el flujo central detallado de cada proceso mediante diagramas, posibilitando la obtención de los definidos por la metodología escogida, entre ellos el diagrama de clase y el modelo de datos. Algunas características que presenta son (Visual Paradigm, 2000):

- Navegación intuitiva entre la escritura del código y su visualización.
- Generador de informes en formato PDF/HTML.
- Documentación automática.
- Generación de bases de datos: transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Interoperabilidad con modelos UML.
- Ingeniería inversa de bases de datos: desde SGBD existentes a diagramas de Entidad-Relación.

Se opta por utilizar la herramienta CASE Visual Paradigm 5.0 para UML 2.1 en el modelado del modelo de datos ya que sigue un estándar de modelación reconocido en el mundo.

Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado (IDE) es un programa informático compuesto por un conjunto de herramientas para programar en un lenguaje de programación o varios, donde se puede encontrar como mínimo un editor, compilador, interprete y depurador de uno o varios lenguajes de programación. Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones (Regalut, 2012).

PhpStorm: cuenta con un editor de código rico para PHP que entiende su código y estructura profundamente, soporta PHP 5.3, 5.4, 5.5 y 5.6 para los proyectos modernos y antiguos. El IDE proporciona finalización de código inteligente, resaltado de sintaxis, la configuración de formato de código extendido, sobre la marcha de la comprobación de errores, plegado de código, integración con sistemas de control de versiones y apoya las mezclas de idiomas. Posee gran disponibilidad de complementos, transfiriendo diversas características a marcos específicos, como finalización de código, la navegación, tipos de inferencia y otras mejoras prácticas para varios marcos de PHP

(Symfony, Drupal, Magento). Utiliza el mismo entorno de desarrollo integrado en Windows, Mac OS X y Linux con su clave única (JetBrains, 2016).

Se opta por el uso de esta herramienta en su versión 8.0.2 por las características antes mencionadas, su buen rendimiento en múltiples plataformas, consumo racional del hardware de la computadora. Además, permite el completamiento de código, una mejor integración con los marcos de trabajo para el desarrollo web que existen en la actualidad y con sistemas de control de versión como Git, VCS y Subversion.

Sistema Gestor de Base de Datos

Un SGBD consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. El objetivo primordial de un SGBD es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la base de datos. De igual forma es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, además proporciona acceso controlado a la misma (Fernandez, 2009).

PostgreSQL: es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia Distribución de Software Berkeley. Según Stinson (2001) permite que mientras un proceso escriba en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último que se ejecutó. Esta estrategia es superior al uso de bloqueos por tabla o por filas comunes en otras bases, eliminando la necesidad del uso de bloqueos explícitos. Utiliza un modelo cliente-servidor y multiproceso en vez de multihilo para garantizar la estabilidad del sistema. La aplicación de escritorio pgAdmin III se utiliza como interfaz gráfica para hacer más fácil la administración del usuario con la base de datos. Se puede ejecutar en varias plataformas como: Linux (todas las distribuciones recientes), Windows (Win2000 SP4 y posterior), FreeBSD, OpenBSD, NetBSD, Mac OS X, AIX, HP / UX, IRIX, Solaris, Tru64 Unix y UnixWare. Otros sistemas Unix también pueden funcionar, pero no están siendo probados actualmente (PostgreSQL, 2016).

Las características más importantes a destacar en la versión 9.3 son (PostgreSQL, 2016):

- Conectores de datos foráneos modificables.
- Sumas de verificación de páginas de datos.
- Conmutación por error rápido.
- Métodos constructores y extractores adicionales para JSON[5].
- Vistas actualizables automáticas.
- *pg_dump*[6] en paralelo para acelerar el respaldo de bases de datos muy grandes.
- Procesos en segundo plano definidos por el usuario.

[5] JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos (BRAY, 2014).

[6] p*g_dump*, funcionalidad que permite extraer una base de datos de PostgreSQL hacia un archivo como copia de seguridad (PostgreSQL, 2016).

Para el desarrollo de GINFOR se utilizó PostgreSQL en su versión 9.3 para la gestión de la base de datos, por lo tanto, se utiliza en la investigación actual con ese fin, ya que este SGBD es soportado por un gran número de plataformas, además brinda estabilidad y seguridad al usuario.

1.5.8 Servidor HTTP

Apache es un servidor HTTP de código abierto para sistemas operativos modernos, incluyendo UNIX y Windows NT. Proporciona un servidor seguro, eficiente y extensible que proporciona servicios HTTP en sincronización con los estándares HTTP actuales. Mejora el almacenamiento en caché y soporta grandes cantidades de información (The Apache Software Foundation, 2016). Presenta, entre otras características, mensajes de error altamente configurables, es flexible y de diseño modular, presenta bases de datos de autenticación y negociado de contenido, tiene una amplia aceptación en la red: es el servidor HTTP más popular ya que funciona en casi todos los sistemas operativos (Comparison of Different Web Servers, 2016).

Se opta por el uso de esta herramienta ya que la arquitectura de este servidor es modular y las principales metas de su diseño son: velocidad, simplicidad, multiplataforma y facilidad de desarrollo distribuido garantizando una buena plataforma como base para la solución de la presente investigación.

1.6 Patrones de diseño

Los patrones de diseño son herramientas que proveen facilidades para crear un software reutilizable de buena calidad. Cada patrón describe un problema que ocurre repetidamente en el entorno y describe el núcleo de la solución a ese problema, de tal forma que ésta pueda ser usada un millón de veces, sin hacer el mismo trabajo dos veces (Asenjo González, y otros, 2003). A continuación, se mencionan los patrones utilizados en la investigación

1.6.1 Patrones GRASP

- **Experto:** el patrón experto en información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Este permite conservar el encapsulamiento, ya que los objetos se valen de su propia información para llevar a cabo las tareas. El comportamiento se distribuye entre clases que cuentan con la información requerida, alentando con ello definiciones de clase sencillas y más cohesivas que son más fáciles de comprender y mantener. Así se brinda soporte a una alta cohesión (Buono, 2012).
- **Creador:** el patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización (Larman, 1999).

- **Controlador:** el patrón controlador es un objeto que no pertenece a la interfaz de usuario, es un manejador artificial de todos los eventos del sistema que define el método de su operación. A menudo genera un bajo acoplamiento y permite un mayor potencial de los componentes reutilizables (Visconti, y otros, 2004). Larman (1999) plantea que los objetos externos de conexión (los objetos ventana y pequeñas aplicaciones) y la capa de presentación no deberían tener la responsabilidad de llevar a cabo los eventos del sistema.
- **Bajo acoplamiento:** consiste en tener las clases lo menos relacionadas entre sí, para en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en el resto de las clases. Esta característica permite potenciar la reutilización y disminuye la dependencia entre las clases (USM, 2015).
- **Alta cohesión:** en la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (Larman, 1999) y (Visconti, y otros, 2004).

1.6.2 Patrones GoF

- **Decorador:** permite añadir responsabilidades extra a objetos concretos de manera dinámica. Proporciona una alternativa flexible a la herencia estática para extender funcionalidad. Evita que las clases más altas en la jerarquía estén demasiado cargadas de funcionalidad y sean complejas (Sánchez, 2010).

1.6.3 Patrón de Llave Subrogada

Este patrón es utilizado cuando se desea generar una llave primaria única para cada entidad. Generalmente en los modelos de datos se utilizan números enteros en columnas auto-incrementales o en este caso identificadores del tipo UUID[7]. Este patrón es muy utilizado por las entidades que integran el modelo de datos, ya que representan en su mayoría identificadores del tipo UUID (Rankins, y otros, 2000).

El uso de los patrones de diseño permite estandarizar el modo en que se realiza el diseño de los módulos. Los patrones GRASP permiten obtener un sistema bien diseñado respecto a las responsabilidades y la estructura de las clases del sistema. De igual forma el patrón de diseño GOF Decorador, facilita la generación de vistas utilizando Twig y el patrón de base de datos llave subrogada permite identificar una instancia con un identificador único.

1.7 Verificación del diseño

Para evaluar la calidad del diseño se opta por el uso de las métricas orientada a clases: Tamaño operacional de clase (TOC) y Relaciones entre clases (RC).

[7] Un UUID (Universally Unique Identifiers) es un entero de 128 bits representado en notación hexadecimal que se puede utilizar siempre que se requiera de un identificador único.

TOC: está dada por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad (Lorenz, y otros, 1994):

- **Responsabilidad:** un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- **Complejidad de implementación:** un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- **Reutilización:** un aumento del TOC implica una disminución del grado de reutilización de la clase.

A continuación, se muestran los criterios a tener en cuenta para aplicar la métrica TOC, donde CP se refiere a la cantidad de procedimientos:

Tabla 5 Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC (Lorenz, y otros, 1994)

Atributo de calidad	Categoría	Criterio
Responsabilidad	Baja	CP< =Promedio
	Media	Promedio<=CP<=2* Promedio
	Alta	CP> 2* Promedio
Complejidad de implementación	Baja	CP< =Promedio
	Media	Promedio<=CP<=2* Promedio
	Alta	CP> 2* Promedio
Reutilización	Baja	CP> 2*Promedio
	Media	Promedio<=CP<=2* Promedio
	Alta	CP<= Promedio

RC: está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad (Lorenz, y otros, 1994):

- **Acoplamiento:** un aumento de las RC implica un aumento del acoplamiento de la clase.
- **Complejidad de mantenimiento:** un aumento de las RC implica un aumento de la complejidad del mantenimiento de la clase.
- **Reutilización:** un aumento de las RC implica una disminución en el grado de reutilización de la clase.
- **Cantidad de pruebas:** un aumento de las RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

A continuación, se muestran en la siguiente tabla los criterios a tener en cuenta para aplicar la métrica RC, donde CRU se refiere a la cantidad de relaciones de uso.

Tabla 6 Rangos de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC (Lorenz, y otros, 1994)

Atributo de calidad	Categoría	Criterio
Acoplamiento	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	CRU<= Promedio
	Media	Promedio<=CRU<=2* Promedio
	Alta	CRU> 2* Promedio
Reutilización	Baja	CRU>2* Promedio
	Media	Promedio<=CRU<= 2* Promedio
	Alta	CRU<= Promedio
Cantidad de pruebas	Baja	CRU<= Promedio
	Media	Promedio<=CRU<=2* Promedio
	Alta	CRU> 2* Promedio

La aplicación de las métricas RC y TOC permite verificar la calidad del diseño de las clases de los módulos Posgrado y Categoría docente, respecto a la cantidad de relaciones entre clases y a la cantidad de funcionalidades que presentan cada una.

1.8 Estrategia de pruebas

La estrategia de pruebas de software describe el enfoque y los objetivos generales de las actividades de prueba. Incluye los niveles de prueba, el tipo de prueba a ser ejecutada y los casos de prueba diseñados para lograr los objetivos (Pressman, 2005).

1.8.1 Niveles de prueba

Los niveles de prueba son aplicados en diferentes escenarios o niveles de trabajo (Pressman, 2010):

- Prueba de unidad: se enfoca en cada componente de manera individual, garantizando su adecuado funcionamiento como unidad. Utiliza mucho de las técnicas de prueba que ejercitan rutas específicas en una estructura de control de componentes para asegurar una cobertura completa y la máxima detección de errores (Pressman, 2010).
- Prueba de integración: aborda conflictos asociados con los problemas duales de verificación y construcción de programas. Durante la integración, se usan más las técnicas de diseños de casos de prueba que se enfocan en entradas y salidas, aunque también pueden usarse técnicas que ejercitan rutas de programa específicas para asegurar la cobertura de las principales rutas de control (Pressman, 2010).

- Prueba de sistema: verifica que todos los elementos del sistema (por ejemplo, hardware, personal, base de datos) se mezclan de manera adecuada y que se logra el funcionamiento y rendimiento global del sistema (Pressman, 2010).
- Prueba de aceptación: las pruebas de aceptación o pruebas del cliente son especificadas por el cliente y se centran en las características y funcionalidades generales del sistema, visibles y revisables por parte del cliente. Las pruebas de aceptación se derivan de las historias de usuario que se han implementado como parte de la liberación del software (Pressman, 2010).

Según explica Pressman (2005) la estrategia de prueba a seguir posee un enfoque incremental, inicia con la prueba de unidades individuales del programa, sigue con pruebas diseñadas para facilitar la integración de las unidades, y culmina con las pruebas que se realizan sobre el sistema construido y las pruebas de aceptación para la liberación del mismo.

1.8.2 Métodos de prueba

Método de caja blanca

Pressman (2010) plantea que se basa en un examen cercano al detalle procedimental. Se prueban las rutas lógicas del software y la colaboración entre componentes, al proporcionar casos de pruebas que ejerciten conjuntos específicos de condiciones, bucles o ambos. Al emplearlos el ingeniero del software podrá derivar casos de prueba que:

- Garanticen que todas las rutas independientes dentro del módulo se han ejercido al menos una vez.
- Ejerciten los lados verdadero y falso de todas las decisiones lógicas.
- Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
- Ejerciten estructuras de datos internos para asegurar su validez.

En el presente trabajo se utiliza la técnica de flujo básico. A continuación, se describen los pasos a seguir al aplicar la técnica (Pressman, 2010)**:**

1. **Notación del grafo de flujo:** usando el código como base, se realiza la representación del grafo de flujo (grafo del programa) mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo: los nodos que representa una o más sentencias procedimentales, las aristas representan flujo de control y son análogas a las flechas del diagrama de flujo y las regiones son áreas delimitadas por aristas y nodos (Pressman, 2010).
2. **Complejidad ciclomática:** es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa y propone que se realice al menos un caso de prueba por cada camino independiente, de manera que se asegure la ejecución de

cada sentencia como mínimo una vez. La complejidad ciclomática V (G), se calcula de tres formas:

- V (G) = Número de regiones del grafo de flujo.
- V (G) = A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
- V (G) = P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.

3. **Determinar un conjunto básico de caminos linealmente independientes:** el valor de V (G) coincide con el número de caminos linealmente independientes de la estructura de control del programa.
4. **Obtención de casos de prueba:** se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico.

Se utiliza esta técnica porque permite obtener una medida de la complejidad lógica de un diseño procedimental y usarla como guía en la definición de un conjunto básico de caminos de ejecución. Además, permite la comprobación de los resultados que se deben obtener al recorrer cada camino de la funcionalidad que se prueba.

Método de caja negra

Según Pressman (2010) se concentra en las historias de usuario del sistema, permite al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todas las historias de usuario de un programa. Trata de encontrar errores en las siguientes categorías:

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

Para desarrollar el método de caja negra existen varias técnicas, entre ellas están (Pressman, 2010):

- **Técnica de la partición de equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Técnica del análisis de valores límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Técnica de grafos de causa-efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Dentro del método de caja negra la técnica de la partición de equivalencia es una de las más efectivas, pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. Se dirige a la definición de casos

de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba que hay que desarrollar (Pressman, 2010).

A partir del análisis realizado se establece como estrategia de pruebas: aplicar los niveles de prueba de unidad, de sistema y de aceptación, utilizando del método de caja blanca la técnica de flujo básico y del método de caja negra la técnica de partición de equivalencia y el análisis de valores límites.

1.9 Validación de la solución

Con el fin de conocer el nivel de satisfacción del cliente respecto a la solución y a su vez validar que esta resuelve el problema de la investigación se aplicará la técnica de ladov y un estudio comparativo de las necesidades del cliente respecto al antes y después del uso del sistema.

1.9.1 Técnica de ladov

La técnica ladov permite medir la satisfacción del cliente con un producto, se compone de cinco preguntas claves: tres cerradas y dos abiertas, las cuales se reformulan en la investigación para valorar el grado de satisfacción de los clientes sobre un tema en específico. Una vez establecidas las preguntas se conforma el cuadro lógico de ladov como se observa en la tabla 7 y el número resultante de la interrelación de las tres preguntas, indica la posición de los sujetos en la escala de satisfacción. La escala de satisfacción está dada por los criterios (Kuzmina, 1970):

- Máxima satisfacción
- Más satisfecho que insatisfecho
- No definida
- Más insatisfecho que satisfecho
- Máxima insatisfacción
- Contradictoria

Tabla 7 Cuadro lógico de ladov (Kuzmina, 1970)

Pregunta cerrada 3	Pregunta cerrada 1								
	No			No sé			Sí		
	Pregunta cerrada 2								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	6
No sé qué decir	2	3	6	3	3	3	6	3	4

Si un encuestado responde a la pregunta uno "no", se va a la zona izquierda del cuadro, debajo de la pregunta uno, donde aparece “no”. Si a la pregunta dos responde "no sé" se busca el "no sé" que aparece debajo del “no” anterior. Si a la pregunta tres responde: "Me disgusta más de lo que me gusta" entonces se busca en las filas, a la izquierda, la casilla donde aparece esa respuesta y se busca el punto donde se interceptan la fila "Me disgusta más de lo que me gusta" con la columna "no sé". El resultado de dicho encuestado es "3", que equivale a "satisfacción no definida". Así se procede con cada usuario de la muestra, en dependencia de sus respuestas. De esta forma se van clasificando en las seis categorías antes mencionadas (Kuzmina, 1970).

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y - 1 de la siguiente forma (Kuzmina, 1970):

Tabla 8 Índice de satisfacción de Iadov (Kuzmina, 1970)

Índice de satisfacción	Escala
Máxima satisfacción	+1
Más satisfecho que insatisfecho	0,5
No definido y contradictorio	0
Más insatisfecho que satisfecho	-0,5
Máxima insatisfacción	-1

La satisfacción grupal (ISG) se calcula por la siguiente fórmula (Kuzmina, 1970):

$$ISG = \frac{A(+1) + B(+0{,}5) + C(0) + D(-0{,}5) + E(-1)}{N}$$

Donde:

- A representa el número de sujetos con índice individual 1.
- B representa el número de sujetos con índice individual 2.
- C representa el número de sujetos con índice individual 3 o 6.
- D representa el número de sujetos con índice individual 4.
- E representa el número de sujetos con índice individual 5.
- N representa el número total de sujetos del grupo.

Esta técnica se utiliza para validar que la solución propuesta resuelve el problema de esta investigación y permite medir la satisfacción de los clientes con un producto.

1.10 Conclusiones del capítulo

El estudio de los procesos de posgrado y categoría docente, así como de los sistemas que los gestionan, evidenció la necesidad de desarrollar los módulos Posgrado y Categoría docente de GINFOR. La metodología de desarrollo de software, los lenguajes de programación, el marco de trabajo, las herramientas y los patrones de diseño facilitan el desarrollo de los módulos a desarrollar. La verificación del diseño, la estrategia de pruebas y la validación de la solución, permiten identificar

deficiencias en el diseño y en la implementación de las funcionalidades de los módulos, con el objetivo de solventarlas y obtener un producto que contribuya a solucionar el problema de la investigación.

CAPÍTULO 2: Propuesta de solución

2.1 Introducción

En este capítulo se describen los aspectos fundamentales de la propuesta de solución, de acuerdo a lo planteado en el capítulo anterior y se desarrollan las fases de la metodología de desarrollo XP. Además, se presenta el modelo de datos, la arquitectura de la propuesta de solución, la verificación del diseño, el estándar de codificación y la descripción de la solución.

2.2 Fase I: Exploración

Según Letelier Torres (2003) los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema.

2.2.1 Historias de usuario

Las historias de usuario (HU) sirven para registrar los requerimientos de los clientes y son utilizadas para crear las pruebas de aceptación y realizar la estimación de cada una de las iteraciones durante la fase de planificación como se plantea en (Calabria, y otros, 2003). Describen brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales en base a lo que se estima necesario para el sistema. El tratamiento de las HU es muy dinámico y flexible. Cada una es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Los datos que deben ser llenados en una HU son los siguientes (Grupo ISSI, 2003):

- **Número:** identificador de la HU.
- **Nombre:** nombre que identifica a la HU.
- **Usuario:** involucrados en la ejecución de la HU.
- **Iteración asignada:** iteración en que se implementará la HU.
- **Prioridad en el negocio:** prioridad de la HU con respecto al resto de las HU (alta, media o baja).
- **Riesgo en desarrollo:** riesgo en la implementación de la HU (alto, medio o bajo)
- **Puntos estimados:** estima el esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, siendo utilizados, generalmente, de uno a tres puntos.
- **Puntos reales:** resultado del esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, siendo utilizados, generalmente, de uno a tres puntos.
- **Descripción:** descripción sintetizada de la HU.
- **Observaciones:** información adicional.

Los módulos a desarrollar contemplan 11 HU listadas a continuación:

- Gestionar nomenclador
- Gestionar necesidades de capacitación
- Disponer sobre necesidades de capacitación del trabajador
- Buscar necesidades de capacitación
- Registrar evidencia de posgrado
- Disponer sobre evidencia de posgrado
- Mostrar estado actual de posgrado
- Gestionar categoría docente de trabajadores
- Notificar convocatoria de posgrado
- Generar reportes de categoría docente
- Generar reportes de posgrado

En la tabla 9 se presenta la HU "Gestionar necesidades de capacitación" ya que es una de las principales funcionalidades del sistema, las restantes se encuentran en el Anexo #2.

Tabla 9 HU Gestionar necesidades de capacitación (Elaboración propia)

Historia de usuario	
Número: 2	**Nombre:** Gestionar necesidades de capacitación
Usuario: trabajador	**Iteración asignada:** 1
Prioridad en negocio: alta	**Puntos estimados:** 1
Riesgo en desarrollo: alto	**Puntos reales:** 1
Descripción: cada trabajador podrá seleccionar los cursos de posgrado, diplomados y entrenamientos de los que se ofertan para cubrir sus necesidades de capacitación. El sistema permitirá cancelar una solicitud realizada, después de que haya sido seleccionada.	
Observaciones: • Se mostrará una opción de búsqueda para facilitar el filtrado de los cursos de posgrado, diplomados y entrenamientos. • Se registrarán las solicitudes realizadas y se notificará al jefe de departamento por correo electrónico. Esta notificación también se mostrará en el sistema. • Al finalizar se muestra una interfaz en la que se listan las solicitudes realizadas por el trabajador y su estado (Pendiente, Por cumplir, Cumplido), mientras el jefe de departamento no disponga sobre cada solicitud su estado será "Pendiente".	

2.2.2 Requisitos no funcionales

Los requisitos no funcionales detallan las propiedades o cualidades que el producto debe tener, aumentándole funcionalidad al sistema, pues hacen al producto atractivo, fácil de usar, rápido y confiable. A continuación, se presentan los requisitos no funcionales definidos para GINFOR.

Requisito de usabilidad

RNF1: el sistema podrá ser utilizado por usuarios que tengan experiencia básica en informática y en la gestión de actividades de posgrado y de las categorías docentes. La aplicación web debe poseer un diseño adaptativo, a fin de garantizar la adecuada visualización en múltiples computadores personales, dispositivos tableta y teléfonos inteligentes.

Requisito de soporte

RNF2: se mantendrá un sistema de codificación estándar siguiendo las normas establecidas.

Requisitos de interfaz

RNF3: el sistema debe ofrecer una interfaz amigable y fácil de operar, la tipografía debe ser uniforme, de un tamaño adecuado y con un contraste que resalte los textos. Todas las interfaces deben estar en idioma español.

Requisitos de portabilidad

RNF4: el sistema debe desarrollarse sin basarse en características propias de algún sistema operativo, para lograr un producto multiplataforma.

Requisitos de seguridad

Autenticación

RNF5: los usuarios se autenticarán haciendo uso del servicio LDAP.

Confiabilidad

RNF6: la información manejada por el sistema estará protegida de accesos no autorizados.

Integridad

RNF7: la información que es manejada por el sistema permanecerá inalterada, a menos que sea modificada por el personal autorizado.

Disponibilidad

RNF8: solo tendrá acceso al sistema el personal autorizado.

Requisitos de ambiente de ejecución

Software

RNF9

- Navegador con soporte para HTML 5 y CSS 3.

Hardware en el cliente

RNF10

- 512 MB de memoria RAM.
- Como mínimo un microprocesador a 1.0 GHz.
- Conexión de red.

Hardware en el servidor

RNF11

- 1 GB de memoria RAM.

- Un procesador Intel Core 2 Duo.
- Disco duro de 500GB.

2.3 Fase II: Planificación de la entrega

En esta fase los clientes establecen la prioridad de las HU en correspondencia con las necesidades inmediatas para luego asignarlas, por el orden de relevancia, a las iteraciones planificadas. A partir de las HU se realiza la estimación del esfuerzo que se requiere por parte del equipo para su posterior implementación (Calabria, y otros, 2003).

Las estimaciones de esfuerzo asociadas a la implementación de las HU la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. El tiempo de implementación de una HU generalmente es de uno a tres puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las HU que fueron terminadas en la última iteración (Calabria, y otros, 2003).

2.3.1 Estimación de esfuerzos por HU

Para el desarrollo de la solución, se realizó la estimación de esfuerzo para cada una de las HU, como se puede apreciar en la tabla 10:

Tabla 10 Puntos de estimación por HU (Elaboración propia)

No	HU	Estimación (semana)
1	Gestionar nomenclador	0,5
2	Gestionar necesidades de capacitación	1
3	Disponer sobre necesidades de capacitación del trabajador	1
4	Buscar necesidades de capacitación	0,5
5	Registrar evidencia de posgrado	2
6	Disponer sobre evidencia de posgrado	1
7	Mostrar estado actual de posgrado	0,5
8	Gestionar categoría docente de trabajadores	1
9	Notificar convocatoria de posgrado	2
10	Generar reportes de categoría docente	0,25
11	Generar reportes de posgrado	0,25
TOTAL		10

2.4 Fase III: Iteraciones

Esta fase incluye varias iteraciones del sistema antes de la primera entrega. El calendario es dividido en un número de iteraciones de tal manera que cada iteración tome de una a cuatro semanas de implementación. En la primera iteración se crea un sistema que abarca los aspectos más importantes de la arquitectura global, esto se logra seleccionando las HU que hagan referencia a la

construcción de la estructura de todo el sistema. El cliente decide que HU van a ser implementadas para cada iteración. Además, se realizan las pruebas funcionales, realizadas por el cliente, al final de cada iteración. Al finalizar la última iteración el sistema está listo para ser puesto en producción (Calabria, y otros, 2003).

2.4.1 Plan de iteraciones

En el plan de iteraciones se especifican las HU a implementar en cada iteración del sistema, estableciéndose 4 iteraciones para la realización del sistema en coordinación con el cliente.

- **Iteración 1:** tiene como objetivo la implementación de las HU 1, 2, 3 y 4.
- **Iteración 2:** tiene como objetivo la implementación de las HU 5, 6 y 7.
- **Iteración 3:** tiene como objetivo la implementación de la HU 8.
- **Iteración 4:** tiene como objetivo la implementación de las HU 9, 10 y 11.

En la tabla 11 se muestran las 4 iteraciones, las HU que incluyen y su duración estimada.

Tabla 11 Plan de iteraciones (Elaboración propia)

Iteraciones	HU	Duración (semanas)
1	HU1 Gestionar nomenclador HU2 Gestionar necesidades de capacitación HU3 Disponer sobre necesidades de capacitación del trabajador HU4 Buscar necesidades de capacitación	3
2	HU5 Registrar evidencia de posgrado HU6 Disponer sobre evidencia de posgrado HU7 Mostrar estado actual de posgrado	3,5
3	HU8 Gestionar categoría docente de trabajadores	1
4	HU9 Notificar convocatoria de posgrado HU10 Generar reportes de categoría docente HU11 Generar reportes de posgrado	2,5

2.4.2 Plan de entregas

Dado el plan de iteraciones realizado en el epígrafe anterior, el plan de entregas acordado con el cliente se resume en la tabla 12:

Tabla 12 Plan de entregas (Elaboración propia)

No	Historia de usuario	Estimación (semana)	Fecha (inicio-fin)
1	Gestionar nomenclador	0,5	6/2/17-8/2/17

2	Gestionar necesidades de capacitación	1	9/2/17-15/2/17
3	Disponer sobre necesidades de capacitación del trabajador	1	16/2/17-23/2/17
4	Buscar necesidades de capacitación	0,5	24/2/17-26/2/17
5	Registrar evidencia de posgrado	2	27/2/17-12/3/17
6	Disponer sobre evidencia de posgrado	1	13/3/17-19/3/17
7	Mostrar estado actual de posgrado	0,5	20/3/17-21/3/17
8	Gestionar categoría docente de trabajadores	1	22/3/17-28/3/17
9	Notificar convocatoria de posgrado	2	1/4/17-14/4/17
10	Generar reportes de categoría docente	0,25	14/4/17-15/4/17
11	Generar reportes de posgrado	0,25	15/4/17-16/4/17

2.4.3 Modelo de datos

El modelo de datos es un conjunto de conceptos, reglas y convenciones que permiten describir y manipular los datos de la parcela de un cierto mundo real que se desea almacenar en la base de datos (Piattini, y otros, 1999).

El modelo de datos correspondiente a los módulos a desarrollar utiliza llaves subrogadas para una mejor manipulación de los datos y tablas nomencladoras para almacenar información. Consta de 11 tablas, de estas 4 son los nomencladores.

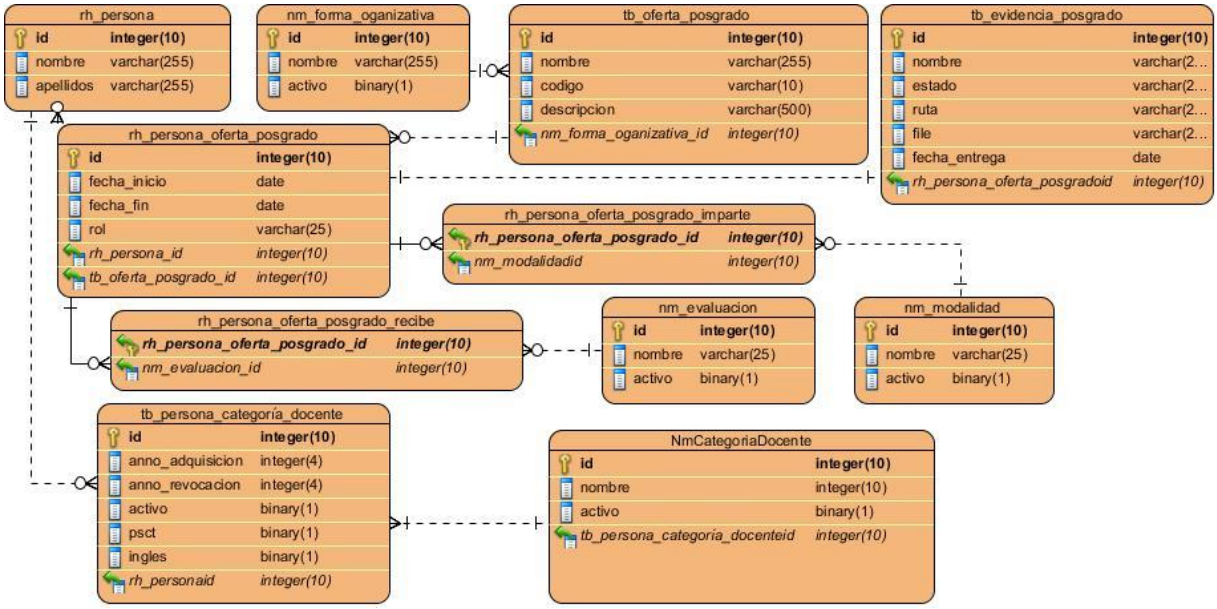

Figura 1 Modelo de datos del módulo Posgrado (Elaboración propia)

2.4.4 Tarjetas Clase - Responsabilidad - Colaborador

Las tarjetas Clase - Responsabilidad - Colaborador (CRC) representan una entidad del sistema, a la cual asignar responsabilidades y colaboraciones. El formato físico de las tarjetas CRC facilita la interacción entre los participantes del proyecto, en sesiones en las que se aplican técnicas de grupos, como tormenta de ideas o juego de roles, y se ejecutan escenarios a partir de la especificación de requisitos e historias de usuarios. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones (Casas, y otros, 2009).

En la metodología XP el proceso de diseño es iterativo. Las tarjetas CRC no se crean en un mismo tiempo, se van desarrollando en la medida que se ejecuten las iteraciones del sistema y se le añaden responsabilidades y colaboradores según sea necesario. Las tarjetas CRC están compuestas por tres campos:

- **Clase:** nombre de la clase que se está modelando.
- **Responsabilidades:** descripción del propósito de la clase.
- **Colaboradores:** clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

En la tabla 13 se muestra la estructura de la tarjeta CRC de la clase *TbPersonaOfertaPosgrado* ya que es una de las clases principales del sistema, las restantes se encuentran en el Anexo #3.

Tabla 13 Tarjeta CRC *TbPersonaOfertaPosgrado* (Elaboración propia)

Clase *TbPersonaOfertaPosgrado*	
Responsabilidad	**Colaboración**
Gestionar las solicitudes de posgrado	*TbOfertaPosgrado* *Persona*

	TbEvidenciaPosgrado

2.4.5 Arquitectura de software

Según Pressman (2010), la arquitectura de software es la representación que capacita al ingeniero del software para: analizar la efectividad del diseño para la consecución de los requisitos fijados, considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el diseño es relativamente fácil, y reducir los riesgos asociados a la construcción del software.

Para el desarrollo de la primera versión de GINFOR se utilizó una arquitectura en capas, utilizando como patrón arquitectónico Modelo-Vista-Controlador (MVC), por tanto, se continúa el uso de esta arquitectura de software y patrón arquitectónico en el desarrollo de los módulos Posgrado y Categoría docente.

Arquitectura en capas

En la arquitectura en capas como organización jerárquica, cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas (Garlan, y otros, 1993). El uso de arquitecturas en capas, explícitas o implícitas, es frecuente en la actualidad y entre las principales ventajas que brinda se encuentran:

- Soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- Admite naturalmente optimizaciones y refinamientos.
- Proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas (Pérez Ávila, 2010).

El sistema GINFOR posee una arquitectura en capas, compuesta por las capas Vista, Controlador, Modelo y Datos.

El marco de trabajo Symfony 2 propone una estructura de paquetes determinando en qué directorio están alojadas las clases correspondientes a la vista, al modelo y al controlador, como se muestra en la figura 2.

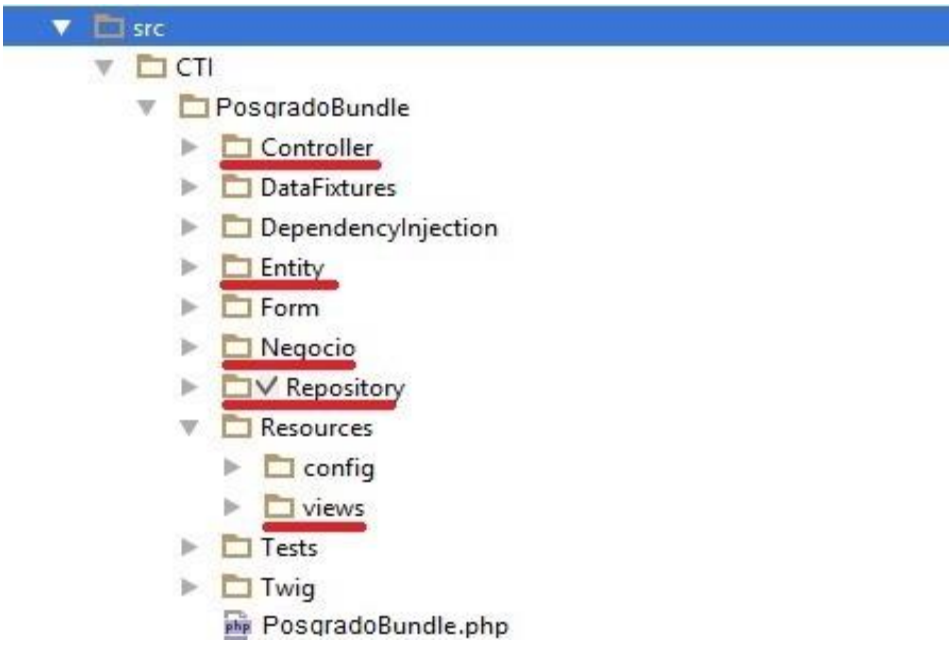

Figura 2 Estructura propuesta por Symfony (Elaboración propia)

En la figura 3 se muestra la arquitectura del sistema y la ubicación de los componentes en cada capa.

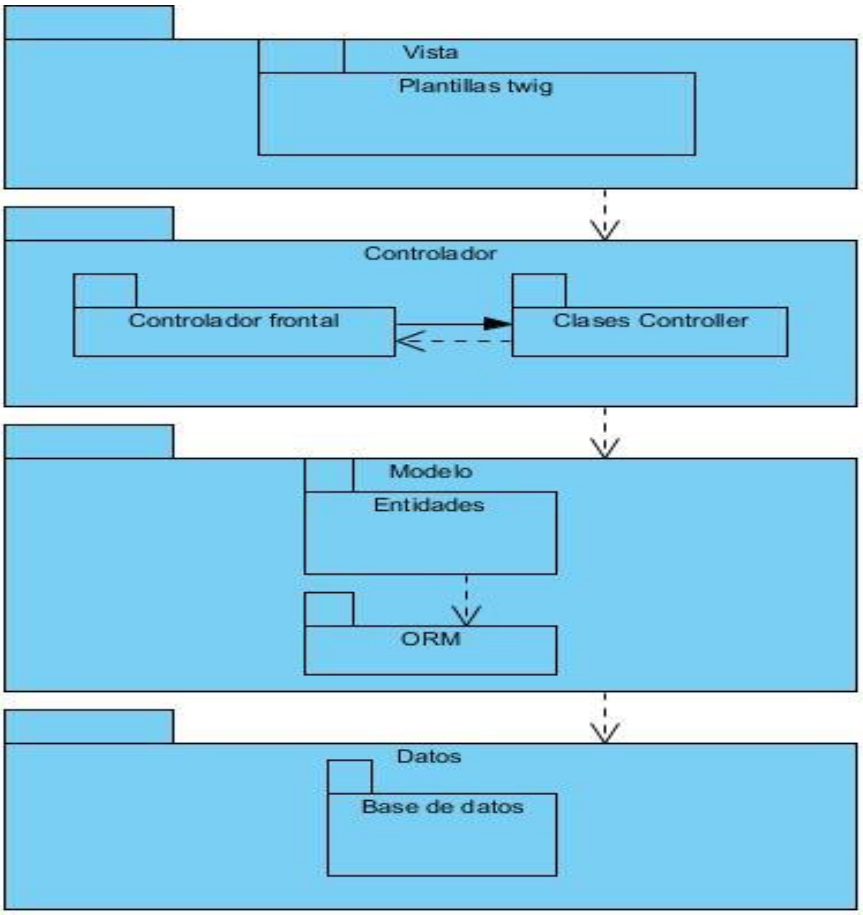

Figura 3 Arquitectura de GINFOR (Elaboración propia)

Vista

Se encarga de gestionar los datos de entrada y salida mediante las interfaces de usuario, esta capa contiene los componentes o páginas con los que el usuario interactúa. En ella se visualizan los datos procesados mediante el navegador web utilizando tecnologías como JavaScript, HTML5, CSS3, JQuery y Twitter Bootstrap, así como las plantillas generadas por el motor de plantillas Twig. En la estructura propuesta por Symfony2 las vistas se encuentran en el directorio *src\CTI\PosgradoBundle\Resources\views*.

Controlador

Es la capa intermedia entre la vista y el modelo, es la encargada de responder a las acciones del usuario e invocar cambios en el modelo o generar la vista apropiada, dependiendo de las peticiones del usuario. En Symfony 2 esta capa queda evidenciada a través del controlador frontal, mediante los archivos *app.php* para entornos de producción y *app_dev.php* para entornos de desarrollo, que se conciben para la ejecución de la aplicación en el servidor de producción y para el uso de los programadores respectivamente, pues les provee de información útil en todas las páginas.

Los archivos de enrutamiento también forman parte de esta capa intermedia los cuales permite determinar qué controlador está asociado con la página que se solicita. En la estructura propuesta por Symfony las clases controladores se encuentran en el directorio *src\CTI\PosgradoBundle\Controller*.

Modelo

Esta capa está compuesta por las subcapas acceso a datos y abstracción de datos. La subcapa de acceso a datos se encarga del manejo de la lógica de negocio, las clases que componen esta capa son las gestoras y los repositorios, ubicadas en *src\CTI\PosgradoBundle\Entity* y *src\CTI\PosgradoBundle\Repository* respectivamente. Las clases gestoras reciben la información obtenida de la vista por medio de las controladoras y posteriormente se encargan de tratar dicha información en conjunto con las clases repositorios. Las clases repositorios contienen un conjunto de consultas para acceder y realizar acciones sobre los datos de tablas específicas de la base de datos.

En esta capa también se encuentra el ORM Doctrine que posibilita la separación de la aplicación respecto al sistema gestor de base de datos mediante su lenguaje propio de consultas DQL. La subcapa de abstracción de datos contiene las entidades, las cuales son una representación de las tablas de la base de datos mapeadas por el ORM Doctrine, de ahí que constituyen una abstracción de los datos reales almacenados en la base de datos. Las clases entidades se encuentran en *src\CTI\PosgradoBundle\Entity*.

Datos

En esta capa se encuentra el gestor de base de datos PostgreSQL y en este un conjunto de tablas, secuencias y funciones que permiten persistir la información con la que trabaja la aplicación y gestionar su almacenamiento.

Elementos transversales

Son un conjunto de facilidades y mejoras que permiten que la arquitectura sea más flexible y adaptable. Entre estas mejoras se encuentran el contenedor de servicios de Symfony 2, la gestión de seguridad, tratamiento de excepciones, caché, configuración y validaciones.

2.4.6 Patrones arquitectónicos

Los patrones arquitectónicos son imprescindibles para la construcción y diseño de un sistema de software. Estos ofrecen la organización estructural de la aplicación (Eguiluz, 2016).

Modelo-Vista-Controlador

El patrón arquitectónico Modelo Vista Controlador (MVC) divide una aplicación interactiva en tres componentes: el modelo, las vistas y los controladores. Estos dos últimos componentes, en conjunto, forman la interfaz del usuario y manejan las peticiones del mismo, asegurando la consistencia entre esta y el modelo. La siguiente figura muestra cómo se aplica en el sistema este patrón, utilizando el flujo que se genera a partir de la necesidad de incluir o editar una oferta de posgrado:

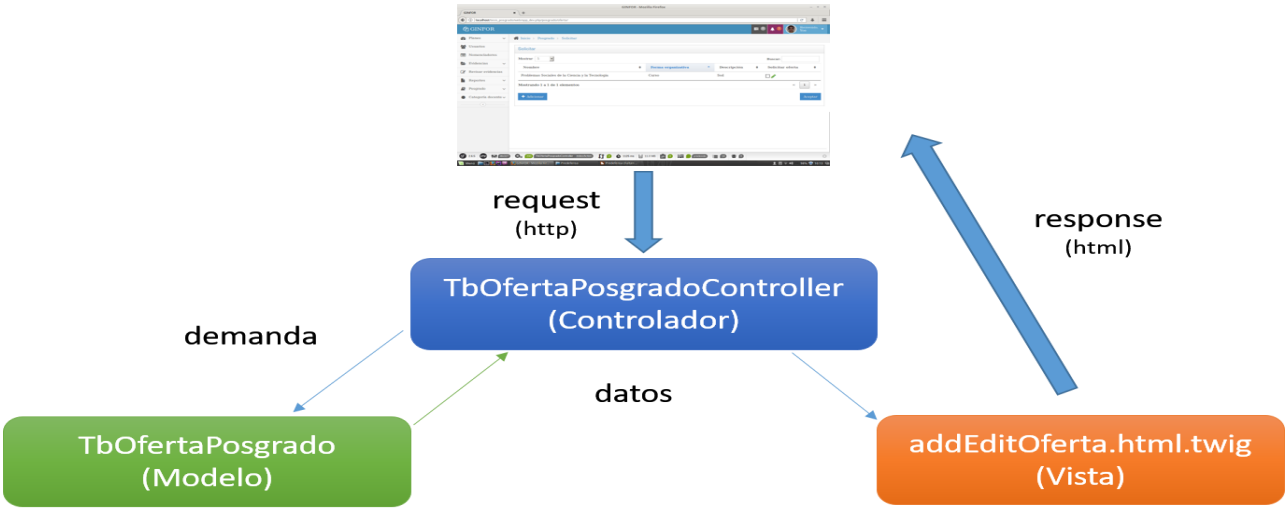

Figura 4 Patrón MVC en GINFOR (Elaboración propia)

En general:

- El modelo representa la información con la que trabaja o su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Por ejemplo, si una misma aplicación debe ejecutarse tanto en un navegador estándar como en un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la

lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (Eguiluz, 2016).

2.4.7 Patrones de diseño

A continuación, se muestra la aplicación de los patrones de diseño en la propuesta de solución.

Patrones GRASP

Se presentan a continuación los patrones GRASP utilizados durante el desarrollo de la investigación, para cada uno se muestra un ejemplo de aplicación en la propuesta de solución:

Experto

El patrón experto indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (bajo acoplamiento). Es empleado en todas las clases entidades que contiene el sistema, en la figura 5 se muestra la clase entidad *TbOfertaPosgrado* que contiene toda la información necesaria para crear un objeto de ese tipo.

```php
class TbOfertaPosgrado
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="nombre", type="string", length=255)
     */
    private $nombre;

    /**
     * @var string
     *
     * @ORM\Column(name="codigo", type="string", length=10)
     */
    private $codigo;
```

Figura 5 Ejemplo de patrón experto en la clase *TbOfertaPosgrado* (Elaboración propia)

Creador

Este patrón se evidencia en las clases Controladoras o *Controller*, por ejemplo, la clase *TbOfertaPosgradoController* ya que es la responsable de la construcción de los formularios que se visualizarán en las vistas referentes a la creación o edición de ofertas de posgrado. En la figura 6 se muestra un ejemplo de clase controladora que refleja el patrón creador.

```php
class TbOfertaPosgradoController extends Controller
{

    /**
     * Lists all TbOfertaPosgrado entities.
     *
     * @Route("/", name="posgrado_oferta")
     * @Method("GET")
     * @Template()
     */
    public function indexAction()
    {
        $em = $this->getDoctrine()->getManager();

        $entities = $em->getRepository('PosgradoBundle:TbOfertaPosgrado')->findAll();

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

Figura 6 Ejemplo de patrón creador en la clase controladora *TbOfertaPosgradoController* (Elaboración propia)

Controlador

Se evidencia su uso en las clases controladoras y en el controlador frontal de Symfony 2. Las peticiones realizadas por el usuario son manejadas por el controlador frontal, único punto de entrada a la aplicación en un entorno determinado, que se localiza en el directorio *CTI/web/app.php*. Además, en todas las clases controladoras está presente y sirve de intermediario entre las interfaces y los algoritmos que las implementan. Las clases controladoras se encuentran en el directorio *src\CTI\PosgradoBundle\Controller.*

Alta cohesión

Se evidencia cuando se desean mostrar las ofertas de posgrado. La acción de mostrar las ofertas es responsabilidad de *TbOfertaPosgradoController*, pero la clase que tiene los datos es *TbOfertaPosgrado*. De esta manera se observa la relación que debe existir entre ambas clases, ya que la primera solo se encarga de controlar la información referente a los datos que devuelve la clase *TbOfertaPosgrado*.

Bajo acoplamiento

Este patrón está presente dentro del marco de trabajo Symfony 2 en las clases que implementan la lógica del negocio y de acceso a datos que se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista, lo que proporciona que la dependencia en este caso sea baja. Como existe poca dependencia entre esas clases se permite una mayor reutilización.

Patrones GoF

A continuación, se presenta el patrón GoF utilizado durante el desarrollo de la investigación:

Decorador

Patrón de tipo estructura, a nivel de objetos. Añade responsabilidades adicionales a un objeto dinámicamente. Se utiliza para las vistas y la plantilla global que decora el contenido de las mismas. Este patrón se utiliza de manera implícita en el marco de trabajo Symfony2 con el motor de plantillas Twig, que proporciona la herencia entre plantillas para la creación de una plantilla base que contiene

los elementos comunes del sitio web y define los bloques que se van a rediseñar en las plantillas que hereden de esta. Esto proporciona una forma flexible de introducir o eliminar funcionalidades a las plantillas del sistema. En la figura 7 se aprecia una parte de la plantilla DE*diagnostico_necesidades.html.twig* que hereda de *base.html.twig* y a la vez redefine los bloques título y ruta.

```
{% extends '@Seguridad/Default/base.html.twig' %}

{% block titulo %}
    Diagnóstico
{% endblock %}

{% block ruta ...%}

{% block contenido %}
<div class="widget-main">
```

Figura 7 Ejemplo del patrón decorador en la plantilla *diagnostico_necesidades.html.twig* (Elaboración propia)

Patrón de llave subrogada

En la siguiente figura se muestra un ejemplo del patrón llave subrogada donde a la clase *rh_persona_oferta_posgrado* se le asigna un identificador o llave única (id).

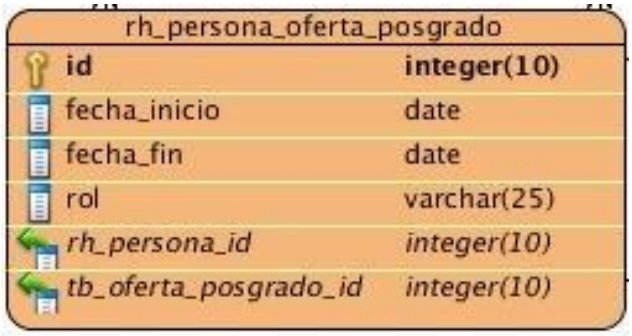

Figura 8 Uso del patrón Llave subrogada (Elaboración propia)

2.4.8 Tareas de ingeniería

Una vez identificadas las HU, los programadores proceden a descomponer cada una en tareas específicas, las denominadas tareas de programación que están escritas técnicamente y que darán solución a la HU correspondiente. En las siguientes tablas se muestran las tareas de ingeniería correspondientes a Gestionar necesidades de capacitación, una de las principales HU definidas las restantes se encuentran en el Anexo #4.

Tabla 14 Tarea de ingeniería 2 (Elaboración propia)

Tarea de ingeniería	
Número tarea: 2	**Número HU**: 2
Nombre tarea: Crear interfaz gestionar las necesidades de capacitación	
Tipo tarea: Desarrollo	**Puntos estimados**: 2 días
Fecha inicio: 9/02/17	**Fecha fin**: 10/2/17

Programador responsable: Yoe Reyes Redondo
Descripción: esta tarea facilita la creación de una interfaz para que un trabajador establezca sus necesidades de capacitación.

Tabla 15 Tarea de ingeniería 3 (Elaboración propia)

Tarea de ingeniería	
Número tarea: 3	**Número HU:** 2
Nombre tarea: Notificar a jefe de departamento sobre solicitudes	
Tipo tarea: Desarrollo	**Puntos estimados:** 1 día
Fecha inicio: 11/02/17	**Fecha fin:** 11/2/17
Programador responsable: Yoe Reyes Redondo	
Descripción: esta tarea facilita la notificación mediante correo y el sistema al jefe de departamento sobre las solicitudes realizadas por un trabajador	

Tabla 16 Tarea de ingeniería 4 (Elaboración propia)

Tarea de ingeniería	
Número tarea: 4	**Número HU:** 2
Nombre tarea: Mostrar estado de las solicitudes realizadas por un trabajador	
Tipo tarea: Desarrollo	**Puntos estimados:** 1 día
Fecha inicio: 12/02/17	**Fecha fin:** 12/2/17
Programador responsable: Yoe Reyes Redondo	
Descripción: esta tarea facilita la creación de una interfaz para que un trabajador observe el estado (Solicitado, Por cumplir o Cumplido) en que se encuentran sus solicitudes.	

2.5 Verificación del diseño

Para la evaluación de la calidad del diseño propuesto se hace uso de las métricas Tamaño operacional de clase (TOC) y Relaciones entre clases (RC) propuestas por Lorenz y otros (1994).

TOC: permite medir los atributos de calidad responsabilidad, complejidad de implementación y reutilización de las clases del diseño. La responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que a mayor responsabilidad y complejidad de implementación de una clase, menor será su nivel de reutilización (Lorenz, y otros, 1994). Las clases del sistema evaluadas en los atributos de calidad mencionados se muestran en la tabla 17.

Tabla 17 Métrica TOC (Elaboración propia)

Clase	Cantidad de procedimientos	Responsabilidad	Complejidad de implementación	Reutilización
NmCategoriaEvaluacion	10	Baja	Baja	Alta
NmFormaOrganizativa	10	Baja	Baja	Alta

NmModalidad	10	Baja	Baja	Alta
NmCategoriaDocente	10	Baja	Baja	Alta
TbEvidenciaPosgrado	29	Alta	Alta	Baja
TbOfertaPosgrado	15	Media	Media	Media
TbPersonaOfertaPosgrado	15	Media	Media	Media
TbPersonaOfertaPosgradoImpartir	2	Baja	Baja	Alta
TbPersonaOfertaPosgradoRecibir	2	Baja	Baja	Alta
TbPersonaOfertaPosgradoRecibirRepository	1	Baja	Baja	Alta
TbEvidenciaPosgrado	29	Alta	Alta	Baja
TbPersonaCategoriaDocente	13	Media	Media	Media

En la figura 9 se representan gráficamente los resultados de la aplicación de la métrica TOC.

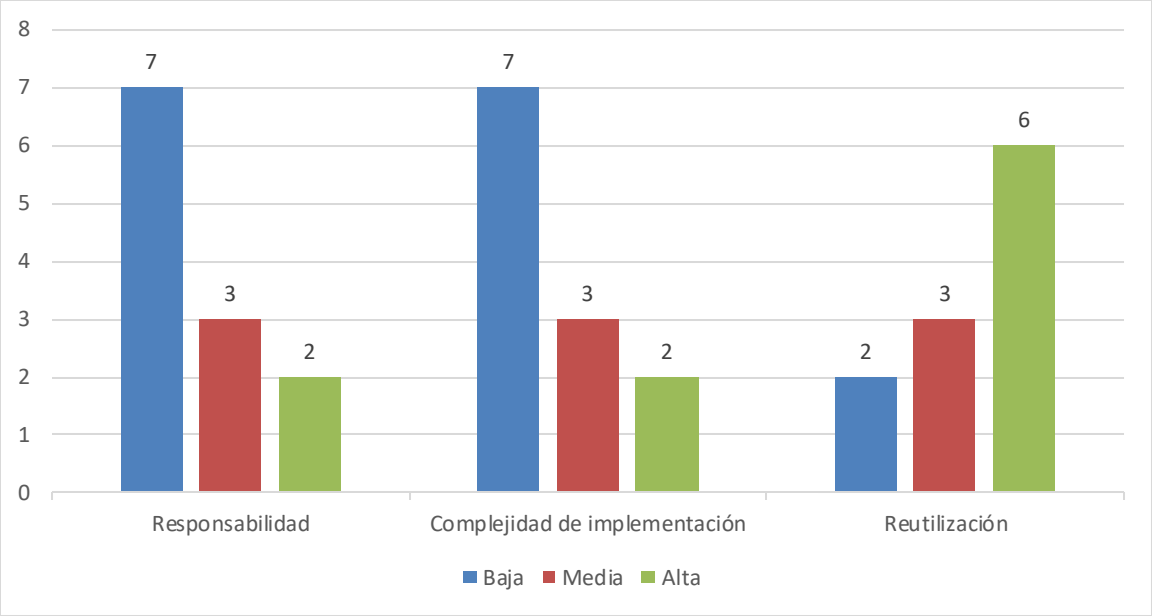

Figura 9 Representación de los resultados de la métrica TOC (Elaboración propia)

Como resultado de la aplicación de la métrica TOC se evidencia que las clases del sistema poseen una baja responsabilidad, baja complejidad de implementación y alta reutilización, por lo que el diseño de las clases en cuanto a cantidad de funcionalidades es satisfactorio.

RC: está dada por el número de relaciones de uso de una clase con otra. Permite evaluar los atributos de calidad: acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas de unidad necesarias de cada clase, teniendo en cuenta las relaciones existentes entre ellas (Lorenz, y otros, 1994). Las clases del sistema son evaluadas en los atributos de calidad mencionados en la tabla 18.

Tabla 18 Métrica RC (Elaboración propia)

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de pruebas
NmCategoriaDocente	1	Baja	Baja	Alta	Baja
NmCategoriaEvaluacion	1	Baja	Baja	Alta	Baja
NmFormaOrganizativa	1	Baja	Baja	Alta	Baja
NmModalidad	1	Baja	Baja	Alta	Baja
TbEvidenciaPosgrado	2	Media	Media	Media	Media
TbOfertaPosgrado	2	Media	Media	Media	Media
TbPersonaOfertaPosgrado	3	Alta	Alta	Baja	Alta
TbPersonaOfertaPosgradoImpartir	1	Baja	Baja	Alta	Baja
TbPersonaOfertaPosgradoRecibir	1	Baja	Baja	Alta	Baja
TbPersonaOfertaPosgradoRecibirRepository	1	Baja	Baja	Alta	Baja
TbEvidenciaPosgrado	1	Baja	Baja	Alta	Baja
TbPersonaCategoriaDocente	2	Media	Media	Media	Media

En la figura 10 se representan gráficamente los resultados de la aplicación de la métrica TOC.

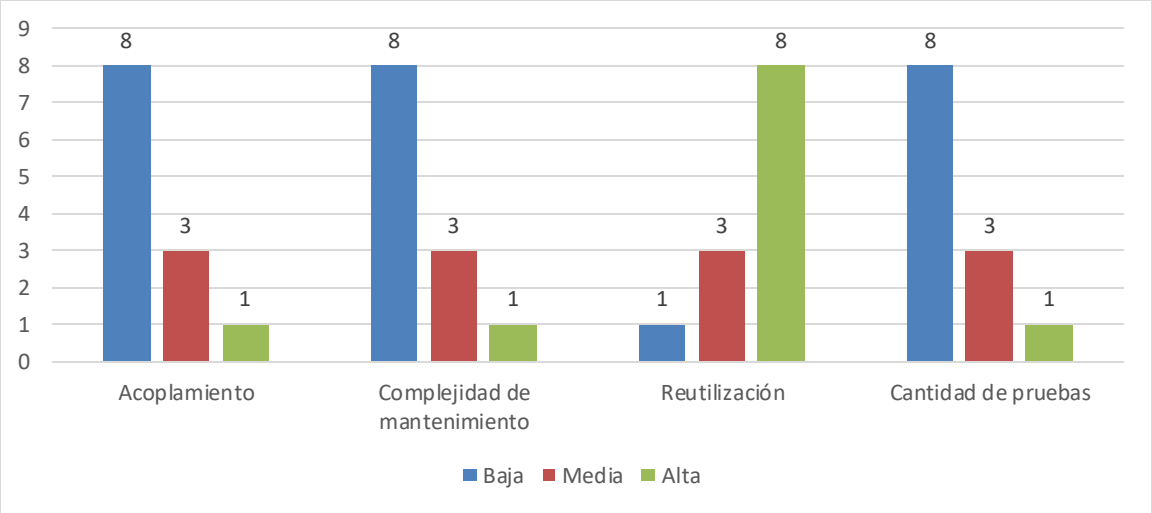

Figura 10 Representación de los resultados de la métrica RC (Elaboración propia)

Como resultado de la aplicación de la métrica RC se evidencia que las clases del sistema poseen un bajo acoplamiento, baja complejidad de mantenimiento, alta reutilización y una baja cantidad de pruebas, por lo que el diseño de las clases en cuanto a cantidad de relaciones de cada una en el sistema es satisfactorio.

2.6 Estándar de codificación

Los estándares de codificación son pautas que se utilizan en la escritura del código fuente, las mismas aseguran que todos los programadores del proyecto mantengan un vocabulario común, además permiten tener un código entendible y organizado. Son empleados para asegurar la unificación en el código. Le provee una guía para el encargado del mantenimiento o actualización del sistema, con código claro y bien documentado (ZANINOTTO, y otros, 2007).

En el desarrollo de la aplicación se decide usar el estándar de codificación *CamelCase*, específicamente la variante *UpperCamelCase* para los nombres de las clases y *lowerCamelCase* para los nombres de los métodos y variables. La Figura 6 Ejemplo de patrón creador en la clase controladora *TbOfertaPosgradoController* (Elaboración propia) muestra un ejemplo de la aplicación de este estándar de codificación en la clase *TbOfertaPosgradoController*.

Se decide seguir las siguientes normas:

General

- Se exceptúan el uso de las tildes y la letra ñ, la que será sustituida por nn.
- Se usarán nombres claros y libres de ambigüedades. Identación.
- El código del sistema será identado por tabulaciones en lugar de hacer uso de espacios en blanco.

Clases

- El nombre de las clases siempre comenzará con mayúscula. En caso de ser una palabra compuesta, cada una de las palabras comenzarán también de la misma forma.
- Intentar mantener los nombres de las clases descriptivos y simples. Usar palabras completas, evitar acrónimos y abreviaturas.

Nombre de variables y métodos

- No se utilizarán nombres de variables que puedan ser ambiguos.
- Los nombres de las variables booleanas deben ser positivos.
- La primera letra usada para los nombres de las variables y métodos es minúscula, en caso de que el nombre sea compuesto empieza con minúscula y el comienzo de la otra palabra es mayúscula.

2.7 Descripción de la solución

Los dos módulos propuestos constituyen una herramienta para la gestión de la información asociada al área de Posgrado y a la gestión de las categorías docentes. Contempla varios tipos de usuarios:

administrador, director, subdirector, asesor de Investigación y Posgrado, jefe de departamento y trabajador en ese orden jerárquico. La siguiente tabla relaciona los módulos Posgrado y Categoría docente con la funcionalidad y el rol que la utiliza.

Tabla 19 Relación Módulo-Funcionalidad-Rol (Elaboración propia)

Módulo	Funcionalidad	Rol
Posgrado	Gestionar necesidades de capacitación	Trabajador
	Disponer sobre necesidades de capacitación del trabajador	Jefe de departamento
	Buscar necesidades de capacitación	Jefe de departamento
	Registrar evidencia de posgrado	Baja
	Disponer sobre evidencia de posgrado	Asesor de Investigación y Posgrado
	Mostrar estado actual de posgrado	Trabajador
	Notificar convocatoria de posgrado	Asesor de Investigación y Posgrado
	Generar reportes de posgrado	Asesor de Investigación y Posgrado
Categoría docente	Gestionar categoría docente de trabajadores	Asesor de Investigación y Posgrado
	Generar reportes de categoría docente	Asesor de Investigación y Posgrado

Solamente tendrán acceso al sistema los trabajadores que pertenecen a CEGEL, después de registrarse con el usuario y contraseña que utilizan para acceder a los servicios telemáticos de la UCI.

2.8 Conclusiones del capítulo

Las HU establecidas por el cliente permitieron identificar las funcionalidades del sistema a desarrollar. La prioridad para el negocio de cada una permitió realizar la estimación del esfuerzo para su desarrollo en cuatro iteraciones. La creación de tarjetas CRC permitió añadir responsabilidades y colaboración a las clases. La aplicación de los patrones y el uso de un mismo estándar de codificación facilitaron la reutilización del código y una mejora en el nivel de complejidad de las clases. El uso del modelo arquitectónico definido permitió analizar el sistema desde distintos puntos de vista y la aplicación de la metodología XP posibilitó reducir el tiempo de desarrollo estimado ante cambios de último momento.

CAPÍTULO 3 Pruebas de la solución

3.1 Introducción

En este capítulo se realiza la validación de la solución propuesta mediante las pruebas definidas en la estrategia de pruebas, así como la validación de las variables de la investigación.

3.2 Fase IV: Producción

En esta fase el producto se pone en producción y se realizan todas las pruebas al sistema. Este momento es donde se afinan los detalles del sistema debido a que se tiene un gran conocimiento del diseño. Además, se dispone del hardware donde se va a ejecutar el sistema. Durante esta fase se debe ir más despacio a medida que es desarrollado el software. Esto no significa que el desarrollo se detenga pero si es de considerar, que el riesgo se vuelve más importante a medida que los cambios afectan la planificación de entrega del proyecto (Pressman, 2010).

3.2.1 Resultados del método de caja blanca

A continuación, se muestra el resultado de aplicar, del método de caja blanca, la técnica del camino básico al método *ObtenerSolicitudesSinEvidencia*, ya que es uno de los métodos principales para la gestión de evidencias, de la clase controladora *TbEvidenciaPosgradoController*, que se encarga de devolver las solicitudes que aún no tienen evidencia.

```
private function ObtenerSolicitudesSinEvidencia(){
    $em = $this->getDoctrine()->getManager();
    $user = $this->getUser()->getPersona()->getId();
    $solicitudes = $em->getRepository('PosgradoBundle:TbPersonaOfertaPosgrado')->findBy(array(
        'persona' => $user,
        'estado' => 'Por cumplir'));
    $res = array();
    foreach ($solicitudes as $solicitud) {
        $evidencia = $em->getRepository('PosgradoBundle:TbEvidenciaPosgrado')->findOneBy(array(
            'personaOfertaPosgrado' => $solicitud));
        if ($evidencia){
            $evidencia = null;
        }
        else {
            $res[$solicitud->getId()] = $solicitud;
            $evidencia = null;
        }
    }
    return $res;
}
```

Figura 11 Método *ObtenerSolicitudesSinEvidencia* (Elaboración propia)

En la figura 12 se muestra el grafo de flujo asociado al método *ObtenerSolicitudesSinEvidencia.*

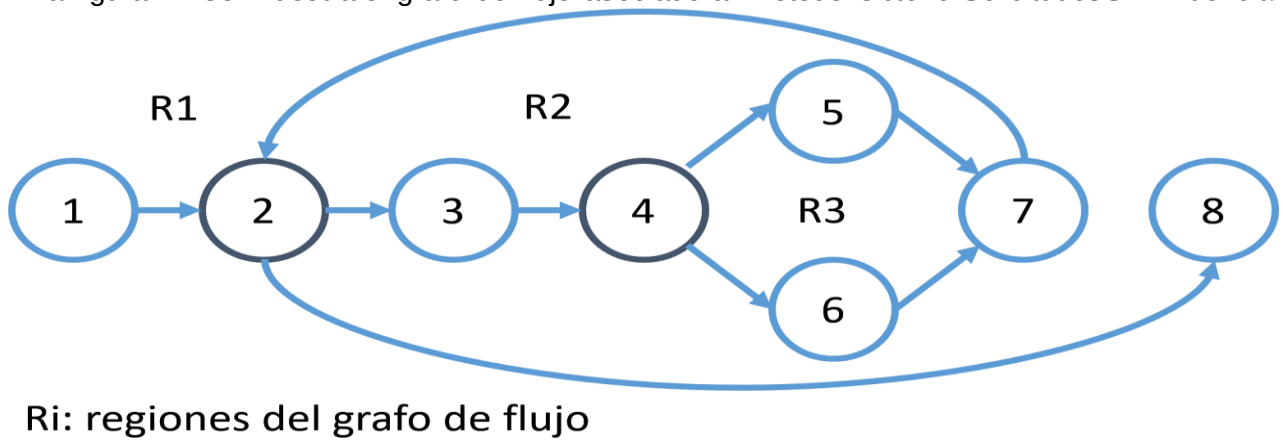

Figura 12 Grafo de flujo del método *ObtenerSolicitudesSinEvidencia* (Elaboración propia)

Tomando como referencia al grafo dirigido del flujo se procede a calcular la complejidad ciclomática empleando las tres variantes definidas.

Existen tres regiones.

V (G) = 9 aristas - 8 nodos + 2 = 3

V (G) = 2 predicado + 1 = 3

La complejidad ciclomática es igual a tres, lo que significa que existen tres posibles caminos linealmente independientes y representa el mínimo número de casos de prueba para el algoritmo. A continuación, se muestran los caminos existentes.

Tabla 20 Caminos por donde el flujo puede circular (Elaboración propia)

Número	Camino
1	1-2-3-4-5-7-2-8
2	1-2-3-4-6-7-2-8
3	1-2-8

El próximo paso es ejecutar los casos de pruebas para cada camino y se compara con los resultados esperados, verificando que las instrucciones se ejecuten por lo menos una vez. A continuación, se muestran los casos de prueba para los caminos básicos identificados.

Tabla 21 Caso de prueba para el camino básico 1 (Elaboración propia)

Caso de prueba para el camino básico 1	
Descripción	Comprueba que existan solicitudes sin evidencia y devuelve, en caso de existir, un listado con las solicitudes.
Condiciones de	Que no existan solicitudes sin evidencia.

ejecución	
Entrada	Ninguna
Resultado esperado	Devuelve un listado vacío.

Tabla 22 Caso de prueba para el camino básico 2 (Elaboración propia)

Caso de prueba para el camino básico 2	
Descripción	Comprueba que existan solicitudes sin evidencia y devuelve, en caso de existir, un listado con las solicitudes.
Condiciones de ejecución	Que exista una solicitud sin evidencia.
Entrada	Ninguna
Resultado esperado	Devuelve un arreglo con una solicitud sin evidencia.

Tabla 23 Caso de prueba para el camino básico 3 (Elaboración propia)

Caso de prueba para el camino básico 3	
Descripción	Comprueba que existan solicitudes sin evidencia y devuelve, en caso de existir, un listado con las solicitudes.
Condiciones de ejecución	Que no existan solicitudes en estado “Por cumplir”.
Entrada	Ninguna
Resultado esperado	Devuelve un listado vacío.

Después de ejecutar los casos de prueba diseñados se comprobó que cada sentencia del código se ejecuta al menos una vez, teniendo en cuenta todas las condiciones lógicas en sus variantes verdaderas y falsas, además los resultados fueron satisfactorios.

3.2.2 Resultados del método de caja negra

En la figura 13 se muestran los resultados del método de caja negra, ejecutado por un especialista del equipo de calidad de CEGEL en tres iteraciones, utilizando las técnicas de partición de equivalencia y análisis de valores límites.

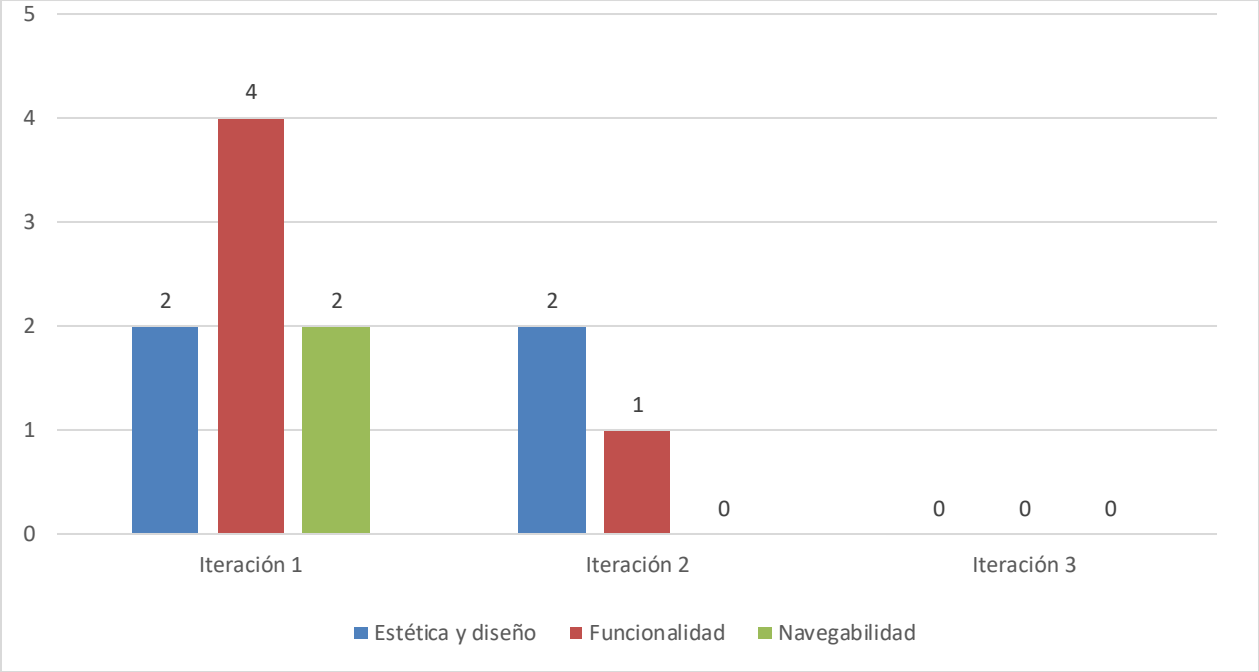

Figura 13 Resultados de aplicar el método de caja negra (Elaboración propia)

En la primera iteración se detectaron un total de ocho no conformidades (NC): dos de Estética y diseño, 4 de Funcionalidad y dos de Navegabilidad, siendo resueltas de forma rápida, dando paso a la segunda iteración donde se detectaron tres NC: dos de Estética y diseño y una de Aplicación, que de igual manera fueron resueltas. En la tercera iteración y final no se encontraron NC, emitiéndose el acta de liberación que se muestra en el Anexo #1.

3.2.3 Pruebas de aceptación

Las pruebas de aceptación se derivan de las historias de usuario que se han implementado como parte de la liberación del software para verificar que este cumple con lo especificado por el cliente (Pressman, 2010). A continuación, se muestra el caso de prueba de aceptación diseñado para la HU Gestionar necesidades de capacitación debido a que es una funcionalidad básica del sistema.

Tabla 24 Caso de prueba de aceptación para la HU Gestionar necesidades de capacitación (Elaboración propia)

Caso de prueba de aceptación	
Código: CPA-02	**No. Historia de usuario: 2**
Nombre: Gestionar necesidades de capacitación	
Descripción: el usuario puede seleccionar o deseleccionar una o varias necesidades de capacitación de posgrado de las ofertadas.	
Condiciones de ejecución: el usuario debe estar autenticado en el sistema.	
Entrada/Pasos de ejecución: el usuario debe seleccionar la necesidad o necesidades de capacitación que desea solicitar o deseleccionar las que quiere eliminar y luego aceptar los cambios.	

Resultado esperado: se le debe enviar a la página de resumen de posgrado del usuario y mostrarle una notificación para indicar que la acción fue realizada con éxito. Además el jefe de su departamento debe recibir una notificación sobre las solicitudes de su trabajador.
Evaluación de la prueba: Prueba satisfactoria.

El director de CEGEL y el asesor de Investigación y Posgrado realizaron las pruebas de aceptación del sistema, teniendo en cuenta los casos de prueba definidos. Los resultados obtenidos fueron satisfactorios emitiéndose el acta de aceptación que se muestra en el Anexo #6.

3.2.4 Validación de la solución

Para realizar la validación de la solución utilizando la técnica de ladov se aplicó la encuesta que se encuentra en el Anexo #5 a 10 trabajadores de CEGEL. La tabla representa el cuadro lógico de ladov modificado con las preguntas cerradas para la valoración por parte de los usuarios.

Tabla 25 Cuadro lógico de ladov para medir la satisfacción de los usuarios (Elaboración propia)

¿Le satisface el sistema desarrollado para acceder a la información que necesita y agilizar su trabajo?	¿Considera que el sistema es complejo y difícil de entender?								
	No			No sé			Sí		
	Si pudiera elegir entre usar o no el software desarrollado para realizar su trabajo ¿lo usaría?								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	6
No sé qué decir	2	3	6	3	3	3	6	3	4

Dada las respuestas obtenidas en la encuesta el cálculo de ISG quedó de la siguiente forma:

$$ISG = \frac{8(1) + 1(0{,}5) + 1(0) + 0(-0{,}5) + 0(-1)}{10}$$

El valor obtenido del ISG fue de 0,85, lo que refleja la aceptación de la propuesta, un reconocimiento a su utilidad, en tanto los usuarios han emitido criterios donde evidencian su satisfacción por la contribución del sistema a la obtención de información asociada al área de Posgrado y a la gestión de las categorías docentes.

Figura 14 Nivel de satisfacción de los usuarios (Elaboración propia)

La técnica de ladov contempla además dos preguntas complementarias de carácter abierto. Estas permiten profundizar en las causas que originan los diferentes niveles de satisfacción y son las siguientes.

- ¿Qué otra funcionalidad crees que sería de utilidad?
- ¿Qué recomendaría para mejorar la aplicación?

Las respuestas dadas reflejan sugerencias de utilidad para la presente y futuras investigaciones. A continuación, se listan algunas de ellas:

- Desarrollar una aplicación para dispositivos móviles que permita a un trabajador realizar la digitalización de sus evidencias de posgrado y almacenarlas en el sistema automáticamente, sin tener que acceder a este desde el navegador.
- Utilizar herramientas de flujo de trabajo (*workflow*) que permitan modificar la lógica del sistema.

Por otra parte, a partir de entrevistas realizadas a trabajadores de CEGEL y del Departamento de Capacitación y Desarrollo de la Dirección de Recursos Humanos de la UCI se realizó una comparación de las necesidades del cliente en cuanto al comportamiento de las variables dependientes de la investigación: rápida y oportuna, enfocada en el antes y después del uso del sistema. Estas necesidades son las mismas a partir de las cuáles fueron definidos los indicadores utilizados en el estudio de los sistemas homólogos. Los resultados obtenidos fueron los siguientes:

Tabla 26 Comparación de necesidades del cliente antes y después del uso del sistema (Elaboración propia)

Necesidades del cliente	Antes	Después

Gestión de necesidades de capacitación.	- El asesor de Investigación y Posgrado debe ver a cada trabajador personalmente para conocer las solicitudes de capacitación del mismo, lo que provoca demoras en el proceso.	- Cada trabajador realiza las solicitudes de capacitación que estime directamente en el sistema, disminuyendo la demora del proceso.
	- El trabajador no tiene un listado con los nombres de los cursos en el momento de realizar sus solicitudes.	- El trabajador tiene un listado con las ofertas disponibles de forma oportuna.
Priorización de necesidades de capacitación.	- Proceso manual, donde el Jefe de departamento recibe el documento de necesidades de capacitación de sus trabajadores solo cuando el asesor de Investigación y Posgrado lo termina.	- El Jefe de departamento puede definir oportuna y rápidamente quién ya realizó sus solicitudes y exigir administrativamente a los que faltan, que lo hagan.
	- El asesor, muestra el documento de necesidades a cada Jefe de departamento quienes deciden que cursos aceptar, priorizar o rechazar.	- El Jefe de departamento puede, directamente en el sistema, priorizar, aceptar o rechazar las solicitudes de necesidades de sus departamentos mientras van siendo realizadas, contribuyendo a la oportunidad y rapidez de esta información.
Notificaciones	- El trabajador solo se entera de las necesidades de capacitación aceptadas y las priorizadas cuando se realiza todo el proceso, se crea el plan de resultados y se discute con él.	- El trabajador se entera de forma rápida y oportuna de la disposición de cada necesidad de capacitación en el momento en que estas son revisadas por el encargado mediante notificaciones del sistema y el correo electrónico.

	- El asesor notifica personalmente o vía correo electrónico a todos los que deben cambiar o ratificar la categoría docente en el año vigente y los requisitos que le faltan para esto, proceso lento y engorroso debido a la información que se debe tener en cuenta.	- El sistema notifica de forma automática a todos los trabajadores, cuya categoría activa vence en el año vigente, sobre la necesidad del cambio de categoría y los requisitos que le faltan por cumplir, cada 3 meses 2 veces por mes, iniciando en Enero.
Recepción de evidencias de posgrado recibidos dentro y fuera de la institución.	- Las evidencias las recibe el asesor de Investigación y Posgrado, que se encarga de guardarlas donde corresponda, dependiendo la inmediatez de este proceso de la carga de trabajo que tenga.	- El trabajador se encarga de subir sus evidencias, que al momento pueden ser revisadas por el asesor de Investigación y Posgrado, proporcionando rapidez al proceso.
	- Los metadatos de las evidencias (curso, fechas, evaluación, etc.) no son gestionados, por lo que para conocerlos el asesor nombra de forma manual cada evidencia.	- El trabajador al subir una evidencia debe definir los metadatos de la misma, permitiendo su localización de forma rápida y oportuna.
Listado de cursos o diplomados cursados dentro y fuera de la institución.	- Los trabajadores no tienen acceso rápido y oportuno a un resumen que contenga la información de todos los cursos que han recibido.	- Los trabajadores pueden ver los cursos que han recibido de forma rápida y oportuna.
	- El director de centro, el subdirector y los jefes de departamento deben consultar al asesor de Investigación y Posgrado para conocer los cursos	- El director de centro, el subdirector y los jefes de departamento pueden consultar la información de los cursos que ha solicitado o recibido cada trabajador de forma rápida y oportuna.

	que ha solicitado o recibido cada trabajador.	
Gestión de categoría docente.	- La actualización de las categorías se hace utilizando un documento de texto que no guarda un historial de las anteriores categorías docentes.	- El sistema muestra un resumen de las categorías docentes de cada trabajador, cuando se categorizó y los cambios de categoría que realizó.

Los resultados obtenidos permiten afirmar que el desarrollo de los módulos Posgrado y Categoría docente de GINFOR contribuye a obtener información de forma rápida y oportuna de estas áreas en CEGEL.

3.3 Conclusiones parciales

Al utilizar la técnica “camino básico” del método de caja blanca, aplicada a los métodos críticos de los módulos, se comprobó la operatividad de estos y a la vez del sistema. La técnica de la partición de equivalencia del método de caja negra aplicada a las HU permitió detectar un conjunto de no conformidades que fueron resueltas en tres iteraciones, obteniéndose la solución esperada por el cliente y el acta de liberación emitida por un especialista del grupo de calidad de CEGEL.

La prueba de aceptación permitió verificar que el software cumple con las necesidades del cliente, obteniéndose finalmente el acta de aceptación.

La técnica de ladov permitió conocer el alto grado de satisfacción de los usuarios y el estudio comparativo permitió validar las variables de la investigación.

CONCLUSIONES GENERALES

La investigación evidenció la necesidad de añadir los módulos Posgrado y Categoría docente al Sistema de gestión de la información de Investigación y Posgrado de CEGEL.

Mediante el diseño e implementación de las funcionalidades identificadas se obtuvo un sistema que permite la gestión de la información del área Posgrado y la gestión de las categorías docentes contribuyendo a obtener información de forma rápida y oportuna asociada a estas áreas en CEGEL.

Se verificó y validó el sistema a partir de métricas, pruebas y técnicas permitiendo comprobar el adecuado diseño del sistema, el cumplimento de los requisitos acordados con el cliente y el alto grado de satisfacción de los usuarios.

Con la aplicación del estudio comparativo se evidenció que el sistema GINFOR permite la gestión rápida y oportuna de la información del área Posgrado y de las categorías docentes en CEGEL.

RECOMENDACIONES

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

- Desarrollar una aplicación para dispositivos móviles que permita a un trabajador realizar la digitalización de sus evidencias de posgrado y almacenarlas en el sistema automáticamente, sin tener que acceder a este desde el navegador.
- Utilizar herramientas de flujo de trabajo (*workflow*) que permitan modificar la lógica del sistema.
- Mejorar la versión web para dispositivos móviles.
- Extender su uso a otras entidades o centros que presenten la mismas dificultes.

BIBLIOGRAFÍA

Álvarez, Miguel Ángel. 2009. desarrolloweb.com. *desarrolloweb.com.* [En línea] 14 de Octubre de 2009. [Citado el: 22 de Febrero de 2017.] http://www.desarrolloweb.com/articulos/que-es-html5.html.

Asenjo González, Diego Andrés y Ríos Peña, Alejandro. 2003. *Patrones de Diseño.* Habana : s.n., 2003.

Avendano, Bibiana. 1981. Transmission control protocol. 1981.

Beck, Kent. 2005. *Extreme Programming Explained.* Segunda. 2005.

BEN-KIKI, Oren, EVANS, Clark y INGERSON, Brian. 2005. *YAML Ain't Markup Language (YAML™) Version 1.1.* s.l. : Tech Rep, 2005.

Borillo, Ricardo. 2012. genbetadev. [En línea] 2012. [Citado el: 9 de Diciembre de 2016.] http://www.genbetadev.com/frameworks/bootstrap.

BRAY, Tim. 2014. *The javascript object notation (json) data interchange format.* s.l. : Internet Engineering Task Force (IETF), 2014. 2070-1721.

Buono, Enrique. 2012. Patrones fundamentales en el Diseño Orientado a Objetos (DOO). *Red Colaborativa Postgrado UCV.* [En línea] 2012. [Citado el: 25 de 3 de 2015.] http://kuainasi.ciens.ucv.ve/red_educativa/blogs/42.

Calabria, Luis y Píriz, Pablo. 2003. *Metodología XP.* Universidad ORT Uruguay : Editorial: Universidad ORT, 2003.

Casas, Sandra y Reinaga, Héctor. 2009. *Aspectos Tempranos: un enfoque basado en Tarjetas CRC.* Argentina : Sandra Casas y Héctor Reinaga, 2009.

Comparison of Different Web Servers. **GOEL, Amita, BANSAL, Nishtha y GUPTA, Shreeya. 2016.** 12, 2016, Imperial Journal of Interdisciplinary Research, Vol. II.

Consejo de Estado. 2016. *Resolución No. 85/16 Reglamento para la Aplicación de las Categorías Docentes de la Educación Superior.* Ministerio de Educación Superior. Habana : s.n., 2016. pág. 18, Resolución.

Consejo de Estado. 2004. *Resolución No. 132/04 Reglamento de Educación de Posgrado de la República de Cuba.* Ministerio de Educación Superior. Habana : s.n., 2004. pág. 2, Resolución.

CSS3. 2015. CSS3.com. *CSS3.com.* [En línea] CSS3.com, Mayo de 2015. [Citado el: 22 de Febrero de 2015.] http://www.css3.com/.

Direccion General de Servicios Informáticos. 2013. DGSI. *Direccion General de Servicios Informáticos.* [En línea] Universidad Nacional de Formosa, Secretaría General Académica, 20 de Abril de 2013. [Citado el: 8 de Diciembre de 2016.] http://siu.unf.edu.ar/index.php.

Doctrine. 2016. doctrine. [En línea] 2016. [Citado el: 8 de Diciembre de 2016.] http://docs.doctrineproject.org/projects/doctrine-orm/en/latest/.

Eguiluz, Javier. 2016. Libros web. Symfony, la guia definitiva. [En línea] 2016. [Citado el: 8 de Dicembre de 2016.] http://librosweb.es/libro/symfony_2_2/.

Eguiluz, Javier. 2015. Libros web: JavaScript. [En línea] 2015. [Citado el: 1 de 3 de 2015.] http://librosweb.es/libro/javascript/..

Fernandez, H. 2009. Slideshare: Sistema de gestión de Base de Datos. [En línea] 2009. [Citado el: 9 de Diciembre de 2016.] http://www.slideshare.net/hugofern/sistema-gestin-de-bases-de-datos.

García Rodriguez, Manuel José. 2015. *Estudio comparativo entre las metodologías ágiles y las metodologías tradicionales para la gestión de proyectos software.* Departamento de Explotación y Prospección de Minas, Universidad de Oviedo. s.l. : Universidad de Oviedo, 2015. pág. 115, Tesis de Maestría.

Garlan, David y Shaw, Mary. 1993. *Advances in software eingineering and knowledge engineering.* 1993.

Gauchat, Juan Diego. 2012. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona, España : Marcombo, 2012. ISBN: 978-84-267-1770-2.

Grupo ISSI. 2003. *Metodologías Ágiles en el Desarrollo de Software.* Alicante : s.n., 2003.

Gutiérrez, Javier. 2016. Lenguajes y Sistemas Informáticos, Universidad de Sevilla. [En línea] 2016. [Citado el: 16 de 11 de 2016.] www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

INEI. 2015. academia. *Herramientas Case.El mejor soporte para el proceso de desarrollo de software.* [En línea] 2015. [Citado el: 12 de 4 de 2015.] http://www.academia.edu/4513393/Libro_HERRAMIENTAS_CASE.

INTECO. 2009. *Ingeniería del software: Metodologías y ciclos de vida.* España : INTECO, 2009. ISBN: 9788478290963.

JetBrains. 2016. JetBrains s.r.o. *JetBrains.* [En línea] 2016. [Citado el: 3 de 12 de 2016.] https://www.jetbrains.com/phpstorm/features/.

Kuzmina, N. V. 1970. *Metódicas investigativas de la actividad pedagógica.* Moscú : Editorial Leningrado, 1970.

Larman, Craig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : PRENTICE-HALL, 1999.

Letelier Torres, Patricio y Otros. 2003. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Universidad Politécnica de Valencia (UPV) : Grupo ISSI, 2003.

Lorenz, Mark y Kidd, Jeff. 1994. *Object-oriented software metrics: a practical guide.* New Jersey : Prentice-Hall, Inc., 1994.

Manzo Rodriguez, Lidia, Rivera Michelena, C. Natacha y Rodriguez Orozco, Alain R. 2006. La educación de posgrado y su repercusión en la formación del profesional iberoamericano. *Educ Med Super.* [En línea] 2006. [Citado el: 12 de 10 de 2016.] http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-21412006000300009. ISSN 0864-2141.

MDN. 2016. Mozilla Developer Network. *Mozilla Developer Network.* [En línea] © 2005-2016 Mozilla Developer Network and individual contributors, 2016. [Citado el: 11 de 11 de 2016.]

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Obsolete_Pages/Guía_JavaScript_1.5/Concepto_de_JavaScript#toc.

Mendes Calo, Karla, Estevez, Elsa y Fillottrani, Pablo. 2009. *Un Framework para Evaluación de Metodologías Ágiles.* Argentina.Universidad Nacional del Sur : EdiUNS, 2009.

MES. 2016. *Instrucción No. 3 /16.* Ministerio de Educación Superior. Habana : s.n., 2016.

MES. 2016. *Instrucción No. 4 /16 (Copia Corregida).* Ministerio de Educación. Habana : s.n., 2016.

Navarro Cadavid, Andrés y Fernández Martínez, Juan Daniel & Morales Vélez, Jonathan. 2013. *Revisión de metodologías ágiles para el desarrollo de software.* Colombia : Editorial:"Universidad ICESI", 2013. Vol. 11, No. 2.

Otero Méndez, Ángel Juan. 2007. *Sistema Informático Para La Gestión De La Formación De Postgrado En Los Profesionales Del Municipio Mayarí.* Moa : Instituto Superior Minero Metalúrgico, MES, 2007. pág. 128, Tesis.

Pérez Ávila, Rodolfo. 2010. *Arquitectura de Software Vista de Integración.* La Habana : s.n., 2010.

PHP. 2015. PHP. *PHP.* [En línea] The PHP Group, 2015. [Citado el: 16 de 11 de 2016.] http://php.net/manual/es/security.intro.php.

Piattini Velthuis, Mario G. 1996. *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión Rama.* Madrid : RA-MA EDITORIAL, 1996.

Piattini, Miguel A y E, Marcos. 1999. *Diseño de base de Datos Relacionales. Ra-Ma.* 1999.

PostgreSQL. 2016. The PostgreSQL Global Development Group. *PostgreSQL.* [En línea] 2016. [Citado el: 4 de 12 de 2016.] http://www.postgresql.org.

Potencier, Favien. 2011. *Manual de Twig.* 2011.

Pressman, Roger. 2005. Capitulo 13. Estrategias de pruebas. [aut. libro] Roger S. Pressman. *Ingeniería del Software. Un enfoque práctico.* Sexta. USA : McGraw-Hil, 2005.

Pressman, Roger S. 2010. *Ingeniería de Software. Un enfoque práctico.* New York : Mac Graw Hill. Higher Education, 2010. ISBN 978-0-07-337597-7.

Proyectos ágiles. 2015. proyectosagiles.org. *proyectosagiles.org.* [En línea] Creative Commons by-sa, 2015. [Citado el: 15 de 11 de 2016.] http://www.proyectosagiles.org/que-es-scrum.

RAE. 2017. Real Academia Española. Diccionario Usual. [En línea] Real Academia Española, 2017. [Citado el: 1 de Junio de 2017.]

Rankins, R., Bertucci, P. y Jensen, P. 2000. *Microsoft SQL Server 2000 Unleashed.* EUA : Sams Publishing, 2000.

Regalut. 2012. Desarrollo Móvil Multiplataforma. *Desarrollo Móvil Multiplataforma.* [En línea] 2012. [Citado el: 9 de Dicembre de 2016.] http://desarrollomovilmultiplataforma.blogspot.com/2012/08/aspectos-teoricosentorno-de-desarrollo.html.

Rivadeneira Molina, Silvia Gabriela. 2012. Revista de informes científicos y técnicos de la Universidad Nacional de la Patagonia Austral. *Metodologías ágiles enfocadas al modelado de requerimientos.* [En línea] Mayo de 2012. [Citado el: 9 de 11 de 2016.] http://ict.unpa.edu.ar/files/ICT-UNPA-57-2013.pdf. ISSN: 1852 - 4516.

Rodríguez Sánchez, Tamara. 2015. *Programa de mejora: Metodología de desarrollo para la Actividad productiva de la UCI.* UCI. La Habana : s.n., 2015. Informe.

Sánchez, Esther Guerra. 2010. Esther Guerra. *Escuela Politécnica Superior.* [En línea] 2010. [Citado el: 2 de Mayo de 2015.] http://arantxa.ii.uam.es/~eguerra/docencia/0809/10%20Decorator.pdf.

Schwaber, Ken y Sutherland, Jeff. 2013. *La Guía de Scrum.* s.l. : PLANETA, 2013. ISBN: 9788408135326.

Shore, James y Warden, Shane. 2008. *The art of agile development.* USA : O'Reilly Media, 2008. 1ra edición, ISBN-13: 978-0-596-52767-9, ISBN-10: 0-596-52767-5.

Sistema de Información Universitaria SIU. 2015. SIU ARAUCANO: Módulo de Estadística de Alumnos. *Sistema de Información Universitaria SIU.* [En línea] 2015. [Citado el: 8 de Diciembre de 2016.] http://www.siu.edu.ar/siu-araucano/.

Stinson, Barry. 2001. *PostgreSQL: Essential Reference.* s.l. : s.l. : Sams Publishing, 2001. ISBN: 0752064711216.

Suárez, María Lorena. 2015. Competencias en TIC: Colección de Fascículos Digitales. *Cuaderno 1: Introducción a la programación y sus lenguajes.* [En línea] 2015. [Citado el: 10 de 11 de 2016.] http://competenciastic.educ.ar/pdf/lenguajes_de_programacion_1.pdf.

Suárez, María Lorena. 2015. Competencias en TIC: Colección de Fascículos Digitales. *Cuaderno 4: Lenguajes del lado del servidor y lenguajes del lado del usuario.* [En línea] 2015. [Citado el: 10 de 11 de 2016.] http://escritorioalumnos.educ.ar/datos/recursos/lenguajes_de_programacion_4.pdf.

symfony.es. 2016. Symfony.es. *Symfony.es.* [En línea] 2016. [Citado el: 1 de 11 de 2016.] http://symfony.es/que-es-symfony.

The Apache Software Foundation. 2016. Apache. *Apache Software Foundation.* [En línea] 2016. [Citado el: 9 de Dicembre de 2016.] www.apache.org.

Thornton, Jacob y Otto, Mark. 2012. Bootstrap. *Bootstrap.* [En línea] 2012. [Citado el: 8 de 12 de 2014.] http://getbootstrap.com/2.3.2/index.html.

tuProgramacion. 2015. tuprogramacion. *tuprogramacion.* [En línea] 2015. [Citado el: 8 de Diciembre de 2016.] http://www.tuprogramacion.com/glosario/que-es-un-orm/.

UCI. 2017. *Objetivos de trabajo 2017.* La Habana : Universidad de Ciencias Informáticas, 2017.

Universidad Nacional de Formosa. 2014. Se realizó el taller anual 2014 del SIU. *Universidad Nacional de Formosa.* [En línea] 20 de Octubre de 2014. [Citado el: 8 de Diciembre de 2016.] http://www.unf.edu.ar/se-realizo-el-taller-anual-2014-del-siu/.

USM. 2015. Universidad Técnica Federico Santa Maria. [En línea] Sitio web administrado por la Dirección General de Comunicaciones, 2015. [Citado el: 19 de 5 de 2015.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08.

Visconti, Marcello y Astudillo, Hernán. 2004. Patrones. *Fundamento de Ingeniería de Software.* Chile : Universidad Técnica Federico Santa Maria, 2004.

Visual Paradigm. 2000. Visual Paradigm International. *Visual Paradigm.* [En línea] 2000. [Citado el: 4 de 12 de 2014.] http://www.visual-paradigm.com.

Wesley, Addison, y otros. 2000. *El Lenguaje Unificado de Modelado. Manual de Referencia.* 2000.

ZANINOTTO, François y POTENCIER, Fabien. 2007. *The definitive guide to Symfony.* s.l. : Apress, 2007.