Facultad 3

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

Título: Sistema de Gestión para la Certificación de Roles 2.0.

Autores

Felinda Rosabel León Mendoza

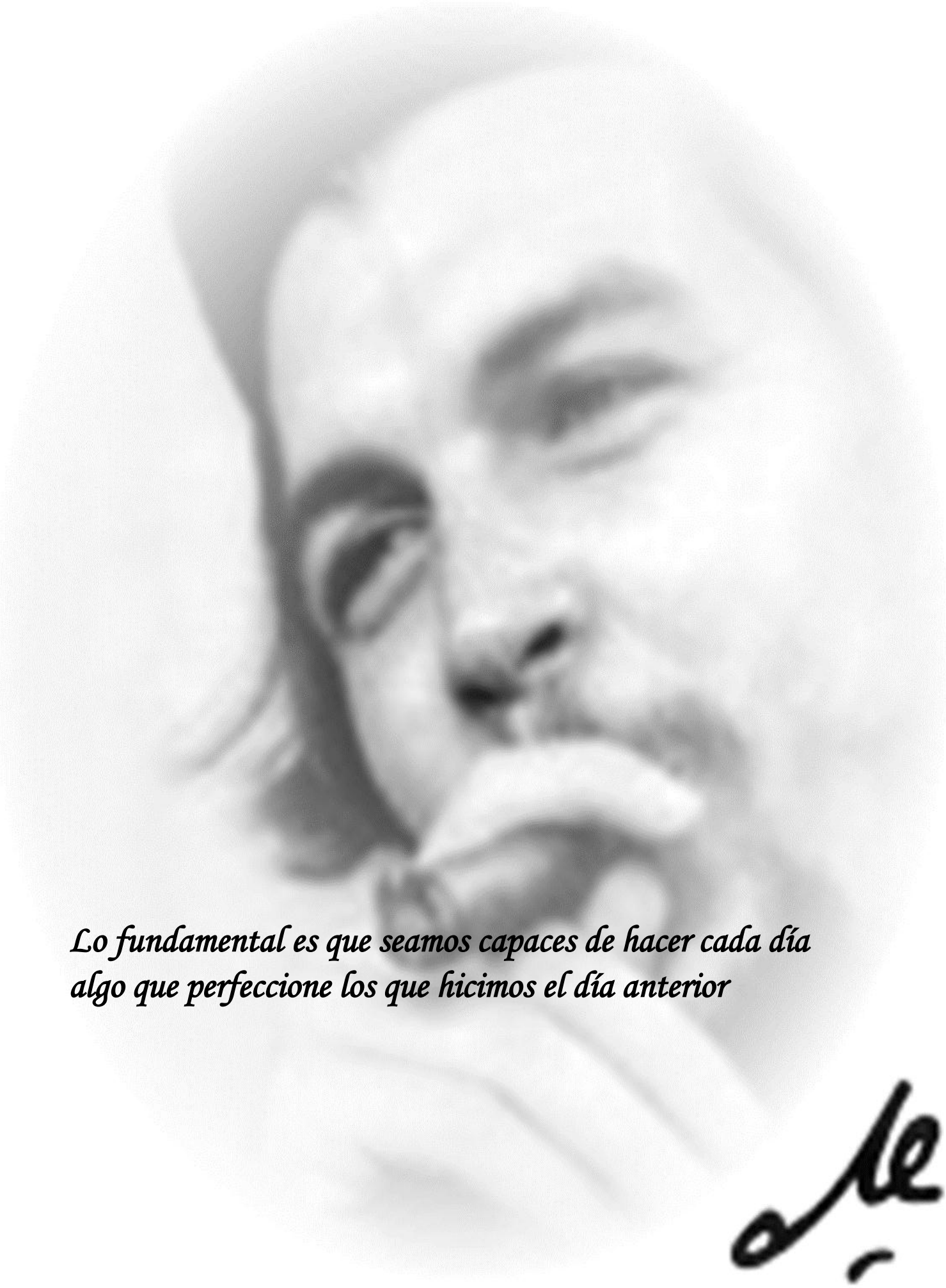
Eliodanis Maceo Rosales

Tutores

Ing. Luis Manuel Borges Rivero

Msc. Yoan Martínez Márquez

La Habana, junio de 2015



Lo fundamental es que seamos capaces de hacer cada día algo que perfeccione los que hicimos el día anterior

Felinda Rosabel León Mendoza

Para llevar a cabo mi sueño fueron muchas las personas que influyeron a lo largo de la carrera

Quiero agradecer en primer lugar a mi mamá Lina, por ser mi guía y ejemplo a seguir, por ser la mejor madre del mundo y apoyarme en todas mis decisiones, además de ser mi madre, ser mi amiga y consejera y por estar siempre ahí para llorar junto conmigo cuando salía mal en las pruebas, por celebrar mis logros, por curarme cuando estaba enferma y también por alarme las orejas cuando la sacaba del paso.

A mi hermana a Anabel, por ser la persona que me empujó a estar hoy aquí y lograr mi sueño, por ser la mejor amiga que he tenido en esta universidad y que ha estado conmigo todos los días de mi vida, por ser mi brazo derecho, por ayudarme en todo y por pasar malas noches estudiando para las pruebas, también por ser la más mandona y ayudarme a ser un poco más organizada.

A mi papá Alejandro por el apoyo que me ha brindado, que a pesar de la distancia su apoyo ha sido incondicional.

A mi tía Sara que aunque hoy no se encuentra aquí, ha sido mi segunda madre, y siempre me ha ayudado y apoyado en todo momento, y porque sé que siempre puedo contar ella, tía te quiero mucho.

A mi prima Sailen por contagiarme con sus novelas coreanas y ser una distracción en los momentos de estrés.

A mi novio Yasmany, por brindarme los momentos más felices de mi vida, por ser mi amigo y consejero y por estar siempre para mí, tati Te I love You.

A mis suegros, por acogerme en su familia como una hija y por el afecto que siempre me han dado.

A todas las amistades con las que he compartido durante estos 5 años, en especial a Sandy, Cesar, Carlos, Arian, Yiselly, Betty, Yasmany y Yordan y también a mis amigas de chismes Diana, Lorena, Gretel y Yaineris

A mi compañero de tesis Maceo por aguantarme todo este tiempo y trabajar siempre juntos para que el resultado final tuviera un resultado exitoso.

A mis tutores Yoan y Luis Manuel por apoyarnos y aconsejarnos y sobre todo por tener paciencia y sacarnos de dudas cuando pensábamos que las cosas eran imposibles de hacer.

A todos los profesores que han contribuido a mi formación como profesional.

Al tribunal de tesis por su preocupación y paciencia, especialmente a la profe Haydee.

En general, a todas aquellas personas que de una forma u otra me ayudaron con la realización de esta investigación.

Eliodanis Maceo Rosales

Primeramente quisiera agradecerle a mi madre Aglay por ser tan especial, por ser mi guía y la persona más importante de mi vida.

A mi padre Laersy, por los consejos y ayuda oportuna.

A mi hermana Xiomarita, que siempre puedo contar contigo y puedes tener la seguridad de que siempre estaré presente para ti.

A mis tías, Nancy, Mirtha, María y en especial a mi tío Armando por estar siempre que te necesité y ser un padre para mí.

A mis primos Reiner, Marita, Diane y en especial a mi primo Diosnel por ser siempre mi ejemplo a seguir.

A mi compañera de tesis Rosabel por estar siempre que la necesité y trabajar juntos para llegar a esta meta.

A mis tutores Yoan y Luis por su ayuda en la elaboración del trabajo de diploma.

A mi novia Laura por apoyarme en todo este tiempo que hemos estado juntos y por comprenderme en los momentos más difíciles.

A todos los amigos con los que he compartido momentos inolvidables, en especial Charly, Adrian, Jorgito, Cesar, Carlos, Yasmany, Yiselly, Las Jimaguas, Diana, Arian, Beatriz, han sido para mí una familia más.

A todas las personas que de una forma u otra colaboraron en el desarrollo de este trabajo.

Felinda Rosabel León Mendoza

Dedico este trabajo a mi mamá por todo el esfuerzo y sacrificio y por ser la persona más especial en mi vida.

A mi hermana por su apoyo y ayuda en estos cinco años de estudio.

Eliodanis Maceo Rosales

Dedico este trabajo a mi mamá por ser la persona más especial en mi vida.

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

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

Felinda Rosabel León Mendoza

_____

Firma del autor

Eliodanis Maceo Rosales

_____

Firma del autor

Msc Yoan Martínez Márquez

_____

Firma del tutor

Ing. Luis Manuel Borges Rivero

_____

Firma del tutor

RESUMEN

El presente trabajo de diploma está dedicado al desarrollo del Sistema de Gestión para la Certificación de Roles en el Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI). El mismo tiene como objetivo contribuir a la mejora del proceso de Certificación de Roles a partir de perfeccionar la ponderación establecida por competencia en cada rol a certificar; así como la conformación y funcionamiento del tribunal.

Se realiza la revisión bibliográfica de la temática a tratar, con el fin de identificar los conceptos principales de la investigación, esta revisión aportó los criterios para selección de la metodología, lenguajes y herramientas usadas como apoyo para darle solución al problema planteado. Además, se realiza un análisis de los sistemas informáticos que miden el proceso de certificación de roles, tanto a nivel nacional como internacional, demostrándose que estos no resuelven los problemas de dicho proceso en la Universidad de la Ciencias Informáticas. Se define la arquitectura del sistema y un conjunto de artefactos necesarios para lograr la implementación. Para evitar inconsistencias durante el desarrollo y cumplir con las expectativas del cliente se validan los requisitos y el diseño. Por último se realizan pruebas al sistema que verifican su correcto funcionamiento y se demuestra que la solución propuesta resuelve las necesidades del proceso de certificación de roles en CEGEL.

Palabras Claves: certificación, ponderación, tribunal, roles

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

El creciente desarrollo tecnológico, unido al acelerado ritmo de la generación de información y las influencias de las comunicaciones en el entorno social, hacen que nuestra sociedad esté en constante cambio. En este contexto, se exige de un profesional competente que dé solución a los nuevos retos que estos tiempos imponen.

Las competencias que un estudiante puede desarrollar, han pasado a ser de interés en las universidades para sus diseños curriculares. Esto permite lograr el profesional idóneo en determinado puesto, con capacidad de comunicarse, trabajar en colectivo y manejar determinadas aptitudes, destrezas, habilidades y conocimientos. De esta manera el estudiante puede resolver situaciones nuevas, generalizar aprendizajes y minimizar el impacto de la rápida obsolescencia de los conocimientos.

Estas competencias están asociadas al desempeño de roles, mediante tareas que generan artefactos a evaluar. Las mismas se concretan en evidencias que aportan la información a gestionar por los directivos, siendo la Informática una de las ramas de la ciencia que ha extendido su aplicación mediante sistemas que gestionan la información en diferentes procesos. La informática reúne muchas de las técnicas que el hombre ha desarrollado, con el objetivo de potenciar sus capacidades de pensamiento, memoria y comunicación. Su área de aplicación abarca la gestión de negocios, el almacenamiento de información, el control de procesos, las comunicaciones, el transporte, la medicina y muchos otros sectores.

Cuba se encuentra inmersa en un proceso de transformaciones tecnológicas, para lo cual se han creado diversas instituciones a nivel nacional que favorecen el crecimiento del país en esta esfera. Una de estas instituciones es la Universidad de las Ciencias Informáticas (UCI), fundada por el Comandante Fidel Castro en el año 2002, con la misión de formar profesionales competentes y comprometidos con la Revolución y contribuir a la informatización de la sociedad cubana mediante el desarrollo de productos y servicios informáticos. Posee una infraestructura productiva, formada por diferentes centros de desarrollo de software que brindan soluciones informáticas a diferentes problemas existentes en disímiles esferas de la sociedad.

En estos centros los estudiantes se vinculan a las actividades de desarrollo de software, papel fundamental para el desempeño de diferentes roles[1] productivos desde las asignaturas de Producción, Investigación y Desarrollo (PID) correspondientes a la disciplina de Práctica Profesional. En el transcurso de la carrera los estudiantes deben desarrollar un sistema de habilidades y competencias establecidas dentro del plan de formación, elaborado por el

[1] Roles: Funciones que algunas personas cumplen en determinado momento, una responsabilidad profesional, conducta esperada según el nivel social y cultural (Real Academia Española, 2015).

profesor de la asignatura y el tutor del Proyecto de Investigación y Desarrollo. El plan de formación está compuesto por macro tareas que se subdividen en tareas, a las que deben dar solución mediante la generación de artefactos que servirán de evidencias en el desarrollo de las competencias. Es precisamente esta información la que aporta los elementos necesarios para que se determine si un estudiante es candidato a certificar uno o varios roles.

Estudiantes pertenecientes al Centro de Gobierno Electrónico de la UCI desarrollaron la Tesis de Diploma: "Sistema de Gestión para la Certificación de Roles (SGCR)" en el curso 2013-2014. Esta solución abordó lo engorroso del proceso de generación, clasificación y almacenamiento de las evidencias, lo cual contribuye a que el estudiante y el profesor PID tengan las evidencias que soporten las tareas realizadas. Después de desarrollar el SGCR, se aplicó una encuesta (Anexo 5) a 29 integrantes de los tribunales de certificación de roles en la universidad, con el objetivo de medir la evaluación de los estudiantes de acuerdo a las evidencias alcanzadas para un rol a certificar, así como valorar la gestión de los tribunales de certificación de roles. La encuesta aplicada arrojó resultados negativos para el proceso de certificación de roles, debido a que la solución implementada no cuenta con una adecuada ponderación en el proceso de certificación de roles, de forma que permita determinar los estudiantes potenciales a certificar uno o varios roles. En este mismo sentido se planteó por la Subdirección de Formación del Centro la inclusión de la gestión del tribunal, desde su conformación, revisión de las evidencias, emisión de valoraciones y la obtención de reportes del proceso, por parte del nivel de dirección al que se subordina este proceso de certificación.

Se han identificado los siguientes elementos que limitan el desarrollo del proceso de certificación de roles:

- ✓ La ponderación de las evidencias establecida por la universidad, para las competencias de cada rol, se implementaron en hojas de cálculo que se encuentran dispersas y ofrecen bajos niveles de seguridad.
- ✓ La imposibilidad de determinar los estudiantes potenciales a certificar uno o varios roles.
- ✓ Las variables que son sustanciales a la conformación y funcionamiento del tribunal para la certificación de roles se omiten o contemplan subjetivamente.

Teniendo en cuenta la problemática explicada anteriormente se plantea el siguiente **problema a resolver:** ¿Cómo gestionar el proceso de certificación de roles, de manera que contribuya a perfeccionar la ponderación establecida por competencias, en cada rol a certificar, así como la conformación y funcionamiento del tribunal?

Se plantea como **objeto de estudio** el desarrollo de sistemas de gestión académica basados en evidencias, siendo el **campo de acción** la ponderación de los sistemas de gestión académica basados en evidencias.

Para dar solución al problema propuesto se traza el siguiente **objetivo general:** Desarrollar una segunda versión del sistema informático para gestionar la certificación de roles, que contribuya a perfeccionar la ponderación establecida por competencia en cada rol a certificar, así como la conformación y funcionamiento del tribunal.

Del objetivo general se desglosan los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación mediante el estudio de los referentes teóricos de la ponderación para la gestión académica basada en evidencias, metodología de desarrollo, técnicas y herramientas de software.
- ✓ Realizar el análisis y diseño del Sistema para la Gestión de la Certificación de Roles 2.0, mediante la metodología de desarrollo de software seleccionada.
- ✓ Implementar el Sistema para la Gestión de la Certificación de Roles 2.0, mediante las herramientas y tecnologías seleccionadas.
- ✓ Valorar la solución propuesta mediante pruebas de software para dar cumplimiento a los objetivos de la investigación.

Como guía a seguir para dar cumplimiento a los objetivos específicos se definen las siguientes **tareas de la investigación** para llevar a cabo el Sistema de Gestión para la Certificación de Roles.

- ✓ Estudio de los referentes teóricos de la ponderación de los sistemas de gestión académica basados en evidencias.
- ✓ Selección de la metodología, tecnologías y herramientas a utilizar en el desarrollo del sistema, para gestionar la certificación de roles.
- ✓ Análisis y especificación de los requisitos funcionales y no funcionales del sistema.
- ✓ Realizar modelo del diseño para la ponderación establecida por competencia en cada rol a certificar, así como al proceso de conformación del tribunal.
- ✓ Implementación de la ponderación establecida por competencia en cada rol a certificar así como del proceso de conformación del tribunal
- ✓ Ejecución y verificación mediante pruebas correspondiente a los métodos de caja negra y blanca al código fuente obtenido en la implementación de la ponderación establecida por competencia en cada rol a certificar, así como del proceso de conformación del tribunal.

Planteándose como **idea a defender**: Desarrollando una segunda versión del Sistema de Gestión para la Certificación de Roles, se perfeccionará la ponderación establecida por competencia, así como el funcionamiento y la conformación del tribunal.

Como métodos científicos de la investigación se emplearon los que a continuación se describen:

Métodos Teóricos:

Histórico-lógico: se emplea para analizar la trayectoria y evolución del diseño, patrones de diseño y metodologías de desarrollo de software. Este método se utilizó durante la realización del estudio de la revisión bibliográfica, permitiendo evaluar el desarrollo de los marcos de trabajo existentes, la evolución de sistemas y de las herramientas que pudieron ser usadas para la elaboración de la solución propuesta (León, y otros, 2011).

Analítico-sintético: permite el procesamiento de la información y arribar a conclusiones prácticas y teóricas de la investigación. Mediante este método se realizó un análisis de los documentos, las publicaciones, bibliografías y en general toda la información relacionada con la certificación de roles y el empleo de sistemas de gestión académica basados en evidencias. Se realizó un estudio sobre las tendencias actuales, posibilitando hacer una síntesis de estas principales fuentes y llegar a conclusiones parciales sobre el tema en estudio (León, y otros, 2011).

Métodos Empíricos:

Técnica de Entrevista: permite obtener información valiosa sobre las necesidades del cliente para identificar los requisitos en el desarrollo del software. Se utilizó en encuentros sostenidos con la Jefa del Departamento Docente de Ingeniería de Software de la Facultad 3 y con la asesora de las asignaturas de la disciplina de Práctica Profesional del Departamento Metodológico Central. Se persiguió como objetivo recopilar información referente al proceso de certificación de roles en la Universidad de las Ciencias Informáticas que contribuyera al entendimiento de este negocio, así como a la validación de las funcionalidades identificadas para el desarrollo de la solución (León, y otros, 2011).

Encuesta: se aplicó una encuesta a profesores y especialistas que forman parte de los tribunales de certificación con el objetivo de obtener la información necesaria para el desarrollo de la presente investigación.

Método Matemático-Estadístico:

Método de los factores ponderados: mide la importancia relativa que tienen los objetivos para el decisor. La puntuación total para cada alternativa se calcula mediante la suma de las

puntuaciones para cada factor ponderado, según su importancia relativa. El método se calcula utilizando la siguiente fórmula:

$$S_j = \sum_{i=1}^{m} W_i \cdot F_{ij}$$

Dónde:

Sj: Puntuación global de cada alternativa j
Wi: Es el peso ponderado de cada factor i
Fij: Es la puntuación de las alternativas j por cada uno de los factores i.

Este método es utilizado en la asignación y cálculo de la importancia relativa en cada uno de los niveles del proceso de certificación de roles, entiéndase por nivel de certificación alcanzado: evaluación de las competencias, evaluación de las evidencias y evaluación de las tareas. Fue desarrollado por los profesores Eugene Benge y sus colaboradores Samuel L. H. Burk y Edward N en el año 1941, a petición de la Philadelphia Rapid Transit Company de Estados Unidos con el objetivo de realizar un método, para la ordenación de los salarios horarios de los trabajadores de la compañía, debido a las insuficiencias que presentaba la aplicación del método de puntos (Buenas Tareas, 2011).

Estructura del documento

Capítulo 1. Fundamentación de la ponderación del Sistema de Gestión para la Certificación de Roles 2.0. Ofrece el análisis de la ponderación utilizada en los sistemas de gestión para la certificación de roles desarrollados en la actualidad, además del estudio de los lenguajes y herramientas utilizadas como apoyo, para darle solución al problema planteado.

Capítulo 2. Planificación, diseño e implementación de la propuesta de solución. Se especifican los requisitos funcionales y no funcionales, teniendo en cuenta las restricciones o políticas a cumplir por el sistema. Además, se realiza la descripción del comportamiento del sistema a través de las Historias de Usuario. Como parte del diseño se define el patrón de arquitectura y se identifican los patrones de diseño que se van a utilizar, posteriormente se confeccionan las Tarjetas CRC (Clase-Responsabilidad-Colaborador) que contienen el nombre de las clases, las responsabilidades y sus colaboradores. Luego se elaboran las Tareas de Ingeniería y se determinan los estándares de codificación para llevar a cabo la implementación del sistema**.**

Capítulo 3. Validación de la propuesta de solución. En este último capítulo, se utilizan las pruebas de verificación y validación, para ello se utilizan los métodos de pruebas de caja blanca y de caja negra, además se realizan las pruebas de aceptación para confirmar si las Historias de Usuario satisfacen las necesidades del cliente.

CAPÍTULO 1: Fundamentación de la ponderación del Sistema de Gestión para la Certificación de Roles 2.0

1.1 Introducción

En el presente capítulo se lleva a cabo la descripción de los conceptos fundamentales que sustentan la investigación y que pueden resultar de difícil comprensión. Además, se realiza un estudio bibliográfico de los sistemas de gestión para la certificación de roles que existen en el contexto actual, para determinar las características, procesos y funcionalidades asociadas a los mismos. También se hace un análisis de las tecnologías actuales y las principales herramientas que pudieran adecuarse en la construcción del sistema a desarrollar.

La presente investigación se encuentra enmarcada en un contexto estudiantil, lo cual trae consigo el uso de procedimientos de acuerdo a las políticas y reglamentos de la universidad. Estudiantes pertenecientes a la UCI realizaron el Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles, que permite gestionar las evidencias y desempeño, que tienen los estudiantes en su labor de PID. Dicho portafolio requiere un mejor funcionamiento de sus procesos, pues no tiene en cuenta la conformación de los tribunales de certificación de roles y no emite correctamente la evaluación de los estudiantes, ya que la aplicación anterior, tiene en cuenta la cantidad de competencias cumplidas para los estudiantes y no la importancia relativa que tiene cada una, para el rol a certificar. Es por ello que la nueva versión del portafolio debe contar con un método estadístico matemático, que mejore la ponderación establecida por competencias, además de evidenciarse la conformación de los tribunales de certificación.

A continuación se relacionan las principales definiciones que han sido estudiadas, con el objetivo de comprender las actividades que se realizan en los procesos de Certificación de Roles y Conformación de los Tribunales

1.2 Conceptos Fundamentales

- ✓ **Ponderar:** atribuir un peso a un elemento de un conjunto, con el fin de obtener la media ponderada (Real Academia Española, 2014).
- ✓ **Método de ponderación:** se basa en la priorización de las variables, en función de la elaboración de matrices y del cálculo de algunos valores que en forma consecuente indicarán que variables son los más importantes según su influencia hacia el resto y según su dependencia del resto (Universidad Peruana de los Andes, 2010).
- ✓ **Competencia:** la Asociación Nacional de Universidades e Instituciones de Educación Superior (ANUIES) define competencias como: El conjunto de conocimientos, habilidades y destrezas, tanto específicas como transversales, que debe reunir un

titulado para satisfacer plenamente las necesidades sociales (Desarrollo de Competencias, 2012).

- ✓ **Tribunal de certificación:** especialistas en el rol que valorarán a los candidatos que se presenten al proceso de certificación (Universidad de Panamá, 2012).
- ✓ **Competencia Laboral:** aptitudes, conocimientos y destrezas necesarias para cumplir exitosamente las actividades que componen una función laboral, según estándares definidos por el sector productivo (Sistema Nacional De Certificación De Competencias Laborales, 2008).
- ✓ **Evaluación de Competencias**: es un proceso de verificación del desempeño laboral de una persona, contra una unidad de competencia laboral, previamente acreditada (Sistema Nacional De Certificación De Competencias Laborales, 2008).
- ✓ **Certificación:** corresponde al proceso de reconocimiento formal, por una entidad independiente, de las competencias laborales demostradas por un individuo en el proceso de evaluación (Sistema Nacional De Certificación De Competencias Laborales, 2008).

Una vez relacionados los conceptos fundamentales relativos al negocio de los procesos Certificación de Roles y Conformación del Tribunal, es preciso conocer las características de los mismos para comprender la necesidad de contribuir a su mejora, a continuación se resume la situación actual de estos procesos.

1.3 Escenario actual de los procesos

En el proceso de Certificación de Roles están involucrados estudiantes y profesores de la universidad. Comienza al inicio de cada semestre donde se elabora el plan de formación de conjunto al estudiante, tutor de PID y profesor de PID. La certificación de rol es opcional y depende de la voluntad del estudiante. Cada plan de formación tiene reflejado un conjunto de competencias que el estudiante tiene que desarrollar para lograr certificar un rol. Para ello la universidad cuenta con una hoja de cálculo, de apoyo al proceso de certificación de roles, que le es envidado a cada estudiante que optó por certificar un rol.

Una vez completada la información, estas hojas de cálculo emiten un posible nivel de certificación, pero este nivel no siempre es correcto. Existe inestabilidad en las fórmulas, ya que al actualizarse las competencias definidas por la universidad para certificar un rol, las mismas no son modificadas en estas hojas de cálculo. El documento tiene muy bajos niveles de seguridad, pues cualquier persona que tenga mínimos conocimientos de excel puede cambiar las fórmulas e incluso eliminarlas.

Otra limitación es que los escenarios son evaluados por el profesor o tutor de PID, quienes le transmiten subjetividad adicional a este proceso de evaluación de desarrollo de las

competencias para cada rol. Además, el profesor tiene que evaluar gran cantidad de escenarios, por lo que la evaluación de estos, se torna un proceso exhaustivo, que pudiera no resultar justo con el estudiante. El sistema actual puede arrojar diferentes resultados en dependencia de los evaluadores, aun teniendo las mismas evidencias. Se considera que la evaluación de los escenarios debe ser estandarizada mediante la ponderación que se establece para el resto de los niveles de la certificación de roles.

El proceso de Conformación del Tribunal se lleva a cabo por el Departamento Metodológico Central (DMC) de conjunto con la Vicerrectoría de Producción de la Universidad, estas dos áreas, son las máximas responsables de conformar el tribunal de certificación. Actualmente, el tribunal es conformado por la asesora metodológica central, donde los individuos que integran estos tribunales son seleccionados por apreciación, y no por las características que deben cumplir para formar parte de estos tribunales de certificación.

Cada tribunal debe estar formado como mínimo por tres personas, que suelen ser especialistas o profesores que tienen experiencia en cada uno de los roles a certificar. Los estudiantes que cumplan con las condiciones necesarias para certificar un rol, deben presentar ante el tribunal de certificación, las evidencias del trabajo en el rol, además de realizar una presentación, donde se expongan las actividades y tareas desarrolladas asociadas el desempeño del rol que desea certificar, así como los elementos teóricos, herramientas, métodos, procedimiento llevado a cabo, resultados y novedad de las soluciones obtenidas.

Durante la presentación, el tribunal comprobará el conocimiento teórico del estudiante en el rol que desea certificar. En este ejercicio debe participar el tutor del estudiante u otro profesor o especialista del proyecto, que de fe del desempeño del estudiante en el rol. Una vez culminado el ejercicio, el tribunal decidirá si otorga o no la certificación al estudiante. En caso de otorgarla, los niveles de certificación que puede alcanzar el estudiante son: básico, intermedio y avanzado. A los estudiantes que certifiquen un rol se les entregará un certificado que refleje la certificación del rol correspondiente, el cual se adicionará a su currículum estudiantil.

Una vez presentado el estudiante ante el tribunal de certificación, las evidencias que el mismo expone, pueden ser la base para optar por el mérito a la investigación científica, pero no se almacena la información relacionada con la presentación del mismo, ni tampoco las valoraciones, y demás informaciones que fueron emitidas por el tribunal y que se contemplaron para emitir un nivel de certificación.

1.4 La ponderación de los sistemas similares para la gestión académica basada en evidencias

A continuación se establece el análisis de los sistemas similares en el marco internacional, entre los cuales destacan, los siguientes sistemas informáticos de referencia:

1. **El Portafolio Electrónico como estrategia de evaluación formativa del idioma Inglés en el nivel secundario de México:** el portafolio electrónico (ePi) presenta una experiencia de innovación en el aula para motivar a estudiantes de tercer grado de secundaria a, elaborar, almacenar sus tareas en formato digital, evaluar su propio aprendizaje (auto-evaluación) a través de rúbricas (Anexo 1) y tutorías del profesor. Una rúbrica o matriz de evaluación, es un cuadro de doble entrada en el cual se expresa de forma explícita, en el eje vertical (cabezas de filas) los aspectos que se evaluarán y ofrece información de la calidad de la tarea y en el eje horizontal (cabezas de columnas,) los cuantificadores (10, 9, 8) o calificativos (excelente, bien, regular, mal) que se asignarán a los diferentes niveles de logro. En las celdas de intersección entre categorías a evaluar y calificadores, se expresa qué características tendrá la tarea para merecer la calificación correspondiente. El protafolio que se plantea está compuesto por cuatro tareas fundamentales a las cuales los profesores responsables de impartir la asignatura de Idioma Inglés le asignaron un coeficiente de importancia, siendo la tercera la más significativa, seguida por la primera, la cuarta y la segunda respectivamente. Luego, con estos criterios establecidos les fue sencillo establecer la nota final. Esta forma de evaluación aporta a los resultados confiabilidad y exactitud, por lo que los autores de la presente investigación la consideran pertinente para su utilización. Por otra parte, el portafolio analizado fue diseñado para que una competencia del estudiante estuviera sustentada exclusivamente por una tarea, siendo esto la principal desventaja para la implementación de la solución propuesta, ya que las tareas establecidas en el plan de formación del estudiante de la UCI que pudieran sustentar una competencia, se denominan macro tareas, las cuales se tienen que dividir en tareas más sencillas para poder medir el desempeño real de los estudiantes (Suárez, 2014).
2. **Portafolio electrónico, desarrollo de competencias profesionales en la red Universidad de Barcelona (UAB) España:** expone el sentido y las características de un portafolio electrónico aplicado en un contexto universitario de enseñanza en línea. Este portafolio como instrumento de evaluación, propicia que el profesor principal emita un juicio evaluativo basado en la observación, la subjetividad y en el juicio profesional, apoyándose en matrices de evaluación para cada una de las tareas. En

este sistema informático, es el profesorado de la Universidad Autónoma de Barcelona quien dicta los factores de importancia en las tareas, aquí los usuarios del portafolio pueden elegir cuales evidencias desean presentar, para en función de los criterios establecidos por la institución, ver si pueden llegar a alcanzar alguna competencia. El portafolio electrónico que se plantea, es capaz de vincular a un profesor y a un estudiante facilitando la retroalimentación, pero disminuyendo la capacidad de involucrar a más individuos. Está diseñado para estudiantes de la carrera de psicopedagogía que tiene un objeto de la profesión diferente al de la carrera en ciencias informáticas, descartando la posibilidad de usarlo para la resolución de la problemática planteada por la tipología y el contexto de las tareas que desempeñan (Barret, 2000).

3. **El desarrollo de la práctica reflexiva sobre el quehacer docente, apoyada en el uso de un portafolio digital, en el marco de un programa de formación para académicos de la Universidad Centroamericana de Nicaragua:** analiza el nivel de calidad reflexiva de los profesores universitarios y la mejora de sus prácticas educativas. Se introduce el uso del portafolio digital, herramienta para el registro de evidencias de un proceso de aprendizaje y que puede facilitar la reflexión sistemática, acerca de la práctica docente. Con los criterios de ponderación previamente definidos por la institución y acordados con los usuarios del portafolio, tres jueces expertos son los encargados de evaluar el avance de las competencias de los profesores. Se emplea una rúbrica construida por ellos, con cinco categorías definidas a partir de los objetivos a alcanzar en el estudio, con el fin de medir el nivel de reflexión de cada profesor. Los resultados obtenidos muestran que el proceso reflexivo es directamente proporcional a aspectos relacionados con la mejora de la docencia, la planificación de clases y de la interacción con los alumnos. El estudio parte del interés de valorar la práctica reflexiva de los profesores universitarios, mediada y gestionada a través de un portafolio digital, en el ámbito de la formación docente, imposibilitando el uso de esta práctica para involucrar a más de un individuo en el proceso. Este último aspecto limita su uso en el contexto de la presente investigación (Rodriguez, 2013).

La solución informática que en mayor medida coincide con este tipo de aplicaciones fue desarrollada la UCI.

1. **Integración de los instrumentos de evaluación para la gestión de las evidencias en la plataforma educativa Zera:** expone la integración de los instrumentos de evaluación que han sido identificados en la plataforma Zera, brindándole al estudiante una potente herramienta de aval donde se almacenen las evidencias de aprendizaje, logros y habilidades alcanzadas por el estudiante durante su vida estudiantil. No

contempla método alguno que permita realizar un análisis evaluativo del cúmulo de evidencias que un estudiante es capaz de generar para certificar un rol informático (Pérez, 2012).

2. **Portafolio para la gestión de evidencias del proceso de certificación de roles (SGCR):** solución informática personalizada para la generación, almacenamiento, clasificación y revisión de las evidencias, elaboradas por los estudiantes en su labor docente productiva. No cuenta con la ponderación establecida por competencias ni tampoco con el proceso de conformación del tribunal (Carlos, y otros, 2014).
3. **Implementación del Portafolio Digital de la Universidad de las Ciencias Informáticas (IPD):** se crea una solución informática, que permite almacenar de manera segura la información de las personas de la universidad en un único expediente, retroalimentándose de todas las evidencias documentales que se generan, asociadas a una persona en un centro de estudios, la cual maneja toda la documentación relacionada con las personas de una institución. Sin embargo, su aprovechamiento en la práctica se encuentra limitado por su enfoque de repositorio que no contempla la evaluación de los artefactos para su constitución en evidencias (Benito, y otros, 2011).

A continuación se ilustra una tabla comparativa en la que se muestran los sistemas informáticos encontrados y los criterios necesarios para la implementación de la solución. Se puede apreciar que ninguno es flexible ante cambios, solo los sistemas nacionales se acoplan al dominio de aplicación y solamente los internacionales poseen una forma de evaluación para medir el desempeño de los estudiantes a través de rúbricas, las cuales basan su funcionamiento en el método de los factores ponderados.

Tabla 1: Tabla comparativa de los sistemas informáticos encontrados.

Solución	Dominio de aplicación	Método de evaluación	Flexibilidad ante cambios
1 ePi	Nivel secundaria básica	Rúbrica	No
2 UAB	Contexto universitario de enseñanza en línea	Rúbrica	No
3 Portafolio Nicaragua	Enmarcado en un ámbito académico universitario	Rúbrica	No
4 Zera	UCI	_____	No
5 ADPDE	UCI	_____	No
6 IPD	UCI	_____	No

Luego de analizar los sistemas informáticos basados en evidencias, se llega a la conclusión de que estos no cumplen con los criterios necesarios para resolver la problemática planteada, el desempeño de las competencias de los estudiantes es medido a través de rúbricas, estas poseen implícitamente un factor de ponderación para los criterios medidos en las tareas (también conocido como método de los factores ponderados), los cuales son previamente definidos por la institución o responsables de la acreditación y acordados con los usuarios del portafolio. Por lo que se propone implementar dicho método en el sistema a desarrollar, así como adicionar las funcionalidades necesarias para gestionar los tribunales de certificación de roles, de forma que contribuya a la solución de la problemática planteada.

1.5 Selección de la metodología, las herramientas, y las tecnologías a utilizar para el desarrollo del Sistema de Gestión para la Certificación de Roles 2.0

1.5.1 Selección de la metodología

La selección de la metodología de desarrollo apropiada es esencial para contribuir al éxito del desarrollo de la solución informática.

"La metodología de desarrollo de software se encarga de elaborar estrategias; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega. Su principal objetivo es elevar la calidad del software a través de un mayor control sobre el proceso" (Sommervile, 2005).

Las metodologías de desarrollo se pueden dividir en dos grupos, de acuerdo a sus características y los objetivos que persiguen, Ágiles y Tradicionales (Velthunis, 2002).

Metodologías tradicionales o pesadas: hacen mayor énfasis en la planificación y control del proyecto, en la especificación precisa de requisitos y modelado, estableciendo estrictamente las actividades involucradas, los roles definidos, los artefactos que se deben producir, las herramientas y la documentación usada (AUP (Proceso Unificado Ágil) RUP (Rational Unified Procces), MSF (Microsoft Solution Framework), Win-Win Spiral Model, Iconix).

Metodologías ágiles o ligeras: están orientadas a la generación de código con ciclos muy cortos de desarrollo manteniendo un proceso incremental, son capaces de permitir cambios en los requisitos de último momento, además el equipo de desarrollo mantiene una comunicación constante con el cliente (Extreme Programming (XP), SCRUM, Crystal Clear, Feature-Driven Development (FDD), Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD)).

Después de analizar los dos grupos, se definió el uso de una metodología ligera escogiéndose la programación extrema (XP, por sus siglas en inglés, Extreme Programming desarrollada

por Kent Beck en el año 2001) porque se centra en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. Además el equipo de desarrollo presenta solo dos integrantes y la planificación del cronograma de ejecución establece poco tiempo para darle respuesta al problema en cuestión. Ofrece la posibilidad de cambiar los requisitos en cualquier momento de la vida de un proyecto, ya que es adaptable a cambios. El cliente se convierte en miembro del equipo y decide que se implementa, además de saber el estado real y el progreso del proyecto. Puede añadir, cambiar o quitar requerimientos en cualquier momento. Se basa en una retroalimentación continuada entre el cliente y el equipo de desarrollo con una comunicación fluida entre todos los participantes, como clave para el éxito en el desarrollo de software.

1.5.2 Marco de Trabajo

Uno de los pasos más importantes en el desarrollo de un software es la selección del marco de trabajo. Un marco de trabajo (framework[2] en inglés) simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona una estructura al código fuente, forzando al desarrollador a crear código más legible. Facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas (Potencier, y otros, 2008).

Symfony 2.1.7: es un marco de trabajo completo, diseñado para optimizar el desarrollo de aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases, encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Permite la ejecución automatizada de tareas, brinda buena documentación y además logra la implementación robusta del patrón Modelo Vista Controlador (MVC) con la integración de Doctrine[3]. Es compatible con la mayoría de los gestores de bases de datos y puede ser ejecutado en múltiples plataformas (Potencier, 2010).

Se selecciona como marco de trabajo Symfony 2.1.7, ya que el mismo ha sido diseñado para desarrollar y optimizar aplicaciones web, así como obtener el máximo rendimiento de PHP 5 y aprovechar sus características.

[2] Framework: Traducido al español se refiere a marco de trabajo. Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concreto, con base a la cual otro proyecto de software puede ser más organizado y desarrollado con mayor facilidad.

[3] Doctrine: Es una biblioteca para PHP que permite trabajar con un esquema de base de datos como si fuese un conjunto de objetos y no de tablas y registros.

Selección del entorno de desarrollo

Un Entorno de Desarrollo Integrado (IDE, según sus siglas en inglés), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic, entre otros. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto (García, 2013).

NetBeans 7.3: es un Entorno de Desarrollo Integrado, modular y de base estándar. Consiste en un IDE de código abierto y una plataforma de aplicación, que puede ser usada como una estructura de soporte general para compilar cualquier tipo de aplicación. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. NetBeans permite crear aplicaciones web con PHP 5, además de venir con soporte para Symfony (Díaz, y otros, 2007).

Se seleccionó como entorno de desarrollo NetBeans 7.3, porque es de código abierto y posee una plataforma de aplicaciones que permiten a los desarrolladores crear un sistema web, este soporta varias plataformas dentro de ellas PHP, además ofrece una selección variada de complementos (plugins, según su denominación en inglés) para optimizar el trabajo y es completamente compatible con Symfony.

1.5.3 Lenguajes de programación

Los lenguajes de programación permiten crear programas mediante un conjunto de instrucciones, reglas de sintaxis y operadores que un equipo debe ejecutar (Bonanata, 2013). A continuación se enuncian los lenguajes de programación empleados para el desarrollo de la solución.

Del lado del Servidor:

PHP 5.3: es un lenguaje de programación multiplataforma orientado a objetos y con una gran cantidad de bibliotecas de funciones. Permite la conexión con la mayoría de los gestores de base de datos, y posee capacidad para expandir su potencial utilizando módulos. No requiere definición de tipos de variables, ni manejo detallado de bajo nivel (Mehdi Achour y otros, 2014).

Este es el lenguaje empleado en la construcción de las páginas web de contenido dinámico. Permite el manejo eficiente de excepciones y el empleo de técnicas de programación orientada a objetos. Proporciona estabilidad a la aplicación ya que utiliza su propio sistema

de administración de recursos y dispone de un método de manejo de variables, conformando así un sistema robusto. Posibilita configurar diferentes niveles de seguridad para evitar ataques al código. En la UCI es una tendencia desarrollar aplicaciones web empleando este lenguaje, producto a las facilidades que brinda. Existe una comunidad considerable de desarrolladores formalizada, con basta documentación sobre la tecnología.

Twig 2.1: es un motor de plantillas, que posee un ambiente amigable para el diseñador y el desarrollador, permitiendo añadir funcionalidades a los entornos de plantillas. Posee una sintaxis corta y concisa, similar a la de otros lenguajes de programación. Además implementa un mecanismo de herencia de plantillas, para acelerar el rendimiento del sistema que se desarrolla (Pacheco, 2013).

Este lenguaje permite compilar las plantillas hasta código PHP regular optimizado, lo que proporciona rapidez durante la implementación. Garantiza la seguridad del código de las plantillas ya que posee un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable. Permite al desarrollador definir sus propias etiquetas y filtros personalizados, garantizando flexibilidad a la aplicación. Posibilita la depuración de las plantillas creadas, mediante mensajes de error con el nombre del archivo y el número de línea donde se produjo el problema (Pacheco, 2013).

Del lado del Cliente:

HTML 5.0: es un lenguaje para escritura de hipertexto, en otras palabras, documentos de texto estructurados, incluye enlaces (links, según su denominación en inglés) que conducen a otros documentos o a otras fuentes de información y permite la inclusión de información por formularios, entre otros. HTML5 se caracteriza por ofrecer una mejor estructura eliminando el uso excesivo de las etiquetas <*div*>, esta se emplea para definir un bloque de contenido o sección de la página; con el objetivo de que la web sea más coherente y comprensible.

CSS 3.0: es un lenguaje creado para controlar el aspecto o la presentación de los documentos electrónicos definidos con HTML[4], es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos .Es empleado para separar los contenidos y su presentación y es imprescindible para crear páginas web complejas ya que separa la definición de los contenidos y la descripción de su aspecto, presentando numerosas ventajas. Mejora la accesibilidad del documento, reduce la

[4] HTML: Lenguaje de Marcado de Hipertexto

complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes (Jeremy, 2014).

Bootstrap 3.0.1: es una de las bibliotecas más utilizadas y conocidas en la actualidad, desarrollada por Twiter, y pensado y diseñado para crear interfaces web. Los diseños creados con Bootstrap son simples, limpios e intuitivos, esto ofrece agilidad a la hora de cargar y al adaptarse a otros dispositivos. Una de las principales características que posee, es que permite crear interfaces web con CSS[5] y JavaScript que adaptan la interfaz dependiendo del tamaño del dispositivo en el que se visualice de forma nativa, en otras palabras, automáticamente se adapta al tamaño de un ordenador o de una Tablet sin que el usuario tenga que hacer nada, esto se denomina diseño adaptativo o Responsive Designy. Es una de las corrientes que marcan esta nueva era digital (Otto, 2014).

JavaScript 1.6: es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario (Polo, y otros, 2008).

1.5.4 Herramientas para base de datos

Sistema Gestor de Bases de Datos: los Sistemas de Gestión de Base de Datos (SGBD) se usan como interfaz entre las aplicaciones, los usuarios y las bases de datos. Tienen como propósito principal gestionar convenientemente la información a almacenar o recuperar, de manera sencilla y fiable para el usuario (Universidad de Sistemas De Información, 2003).

PostgreSQL 9.2: es un gestor de base de datos orientado a objetos (SGBDOO) muy conocido y usado en entornos de software libre. Está considerado como un SGBD de código abierto robusto. Posee uso de transacciones avanzadas, lenguaje procedimental, estabilidad y escalabilidad (Mora, y otros, 2014).

Se selecciona el Gestor de Base de Datos PostgreSQL porque presenta gran capacidad de almacenamiento y buena escalabilidad. Se ajusta al número de CPU (Unidad Central de Procesamiento) y a la cantidad de memoria disponible de forma óptima, soportando una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta.

[5] CSS: Hoja de estilo en cascada

1.5.5 Servidor web

Un servidor web o servidor HTTP[6], es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente, generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. Para la transmisión de todos estos datos generalmente se utiliza el protocolo HTTP o el protocolo HTTPS[7] (la versión cifrada y autenticada), para estas comunicaciones pertenecientes a la capa de aplicación del modelo OSI[8].

Apache 2.0

Apache está diseñado para ser un servidor web potente y flexible, continuamente actualizado y adaptado a los nuevos protocolos (HTTP) y puede funcionar en la más amplia variedad de plataformas y entornos. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular, el cual permite elegir las características del servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. Tiene capacidad para servir páginas tanto de contenido estático, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información (Kabir, 2003).

Es un software de libre distribución, que publica su código fuente, lo que permite que pueda ser modificarlo para colaborar en su desarrollo. Tiene interfaz con todos los sistemas de autenticación. Facilita la integración de complementos de los lenguajes de programación de páginas web dinámicas más comunes. Tiene integración en estándar del protocolo de seguridad SSL y provee interfaz a todas las bases de datos. (Kabir, 2003)

Apache es seleccionado por la necesidad de un servidor web compatible con la plataforma de desarrollo, que permita rapidez en la navegación y la seguridad e integridad de los datos con los que se trabajen. Además Apache permite configurar los módulos a utilizar en el servidor, logrando desechar elementos innecesarios para lograr mayor rapidez en la renderización de las páginas.

1.6 Patrones

1.6.1 Patrón arquitectónico

Un patrón de arquitectura de software es un esquema genérico, probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica

[6] HTTP: Protocolo de transferencia de hipertexto

[7] HTTPS: Protocolo seguro de transferencia de hipertexto

[8] OSI: Modelo de interconexión de sistemas abiertos

describiendo los componentes, con sus responsabilidades, relaciones y las formas en que colaboran (García, 2003).

Modelo-Vista-Controlador (MVC): es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control, en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML, el modelo es la capa de abstracción que se sitúa sobre el Sistema de Gestión de Base de Datos llamado Doctrine y la lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

La vista representa el modelo en un formato adecuado para interactuar con el usuario. Transforma el modelo en una página web, que permite al usuario interactuar con ella. Se encarga de la presentación visual de los datos captados por el modelo, para después mostrarlos al usuario. Los controladores reciben la entrada o sea, los eventos producidos por el usuario mediante el uso del teclado o el mouse a través de las vistas. Los eventos son traducidos para servir las demandas del modelo o las vistas. (García, 2003)

Se seleccionó el patrón MVC, ya que el mismo permite el desarrollo de aplicaciones web robustas, además de poder aplicar multilenguaje y distintos diseños de presentación sin que se altere la lógica del negocio. Proporciona reutilización de los componentes, facilidad para la realización de pruebas unitarias y simplicidad en el mantenimiento del sistema

1.6.2 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular (Wesley, y otros, 2001). La utilización de patrones de diseño, permite ahorrar grandes cantidades de tiempo en la construcción de software. El producto obtenido es más fácil de comprender, mantener y extender. Existen versiones ya implementadas de estas funcionalidades comunes (marcos de trabajo) que permiten centrarse en desarrollar sólo la funcionalidad específica requerida por cada aplicación y que además, dan mejor imagen de profesionalidad y calidad.

Los patrones de diseño se agrupan en dos grandes categorías: GRASP[9] y GoF[10]. Los primeros describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Dentro de este grupo de patrones se encuentran los siguientes: Experto, Creador, Bajo Acoplamiento, Controlador y Alta Cohesión. Los

[9] GRASP: Acrónimo de "General Responsibility Assignment Software Patterns" *en español Patrones generales de software para asignación de responsabilidades.*

[10] Gang of Four, en español Banda de Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.

patrones de diseño GoF son 23 y se clasifican según su propósito en Creacionales, Estructurales, Comportamiento y según su ámbito en Objeto y Clase.

En el capítulo 2 se muestra una explicación de los patrones de diseño que fueron empleados en la implementación de la solución.

1.7 Métricas

Las métricas son una medida cuantitativa que permiten lograr una visión profunda de la eficacia del proceso del software. Ayudan en la planificación, seguimiento y control de un proyecto de software y evalúan la calidad del producto (IEEE, 2008).

1.7.1 Métricas para requisitos

Estabilidad de los requisitos

El objetivo de esta métrica es medir la estabilidad de los requisitos para asegurar su adecuación antes de pasar al próximo flujo de trabajo. Se considera que los requisitos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación. (Revista vinculando, 2010).

Especificidad de los requisitos

El objetivo de esta métrica es cuantificar la especificidad o la ausencia de ambigüedad en la definición de los requisitos. Para calcular esta métrica deben contarse los requisitos que tuvieron igual interpretación por los revisores y compararlos con el total de requisitos definidos (Pressman, 2007).

Grado de validación de los requisitos

Los requisitos deben ser posibles de validar. La validación de los requisitos se realiza en consenso con el equipo de desarrollo, al contrastar lo que desea el cliente con la posibilidad real de implementarlo. El grado de validación de los requisitos mide la corrección en la definición de los requisitos (Revista vinculando, 2010).

1.7.2 Métricas para el diseño

Las métricas referentes al tamaño para las clases orientadas a objetos (OO) se centran en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema OO como un todo. El tamaño general de una clase puede medirse determinando las siguientes medidas:

- ✓ El total de operaciones (heredadas y privadas de la instancia), que se encapsulan dentro de la clase.
- ✓ El número de atributos (heredados y privados de la instancia), encapsulados por la clase.

Estos dos valores son sumados de acuerdo a la clase que se analiza y los resultados son tomados como Cantidad de Procedimientos (CP) que luego son comparados para determinar el Tamaño Operacional de Clases de cada clase (Fornaris, y otros, 2009).

Tabla 2: Clasificación de valores.

Clasificación	Valores
Pequeño	CP <= 20
Medio	CP > 20 y <= 30
Grande	CP > 30

Tamaño Operacional de Clase (TOC)

Consiste en medir el tamaño general de una clase tomando el valor de la cantidad de operaciones que están encapsuladas dentro de dicha clase (Lincke, y otros, 2009)

Si el resultado obtenido indica valores grandes, significa que la clase posee un alto grado de responsabilidad, debido a esto se reducirá la reutilización, se hará mucho más difícil la implementación y la realización de pruebas de dicha clase. Por tanto, mientras menor sea el valor para el TOC se hará más fácil la reutilización de dicha clase dentro del sistema.

Tabla 3: Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica TOC.

Atributo	**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

Relaciones entre Clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad (Desarrollo web, 2013)

Acoplamiento: un aumento del RC implica un aumento del acoplamiento de la clase.

Complejidad de mantenimiento: un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.

Cantidad de pruebas: un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 4: Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	Cantidad de relaciones de uso > 2
Complejidad del Mantenimiento	Baja	Cantidad de relaciones de uso <= Promedio
	Media	Promedio <= Cantidad de relaciones de uso <= 2* Promedio
	Alta	Cantidad de relaciones de uso > 2* Promedio
Cantidad de Pruebas	Baja	Cantidad de relaciones de uso <= Promedio
	Media	Promedio <= Cantidad de relaciones de uso <= 2* Promedio
	Alta	Cantidad de relaciones de uso > 2* Promedio

1.8 Pruebas

La (IEEE, 2015) define las pruebas de software como una actividad en la que un sistema o un componente es ejecutado bajo condiciones especificadas .El objetivo es diseñar una serie de casos de pruebas que tengan una alta probabilidad de encontrar errores. Para ello se aplican las técnicas de pruebas del software, las cuales facilitan una guía sistemática para diseñar pruebas que comprueben la lógica interna de los componentes de software y verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento (Pressman, 2005) .

1.8.1 Método de prueba de caja negra

Las pruebas de caja negra son aquellas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina

algunos aspectos del funcionamiento del sistema sin tener mucho en cuenta la estructura interna del software (Peña, 2010) .

Estas pruebas permiten encontrar:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a las Bases de Datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

Según (Pressman, 2005) para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

1. **Técnica de la Partición de Equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
2. **Técnica del Análisis de Valores Límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
3. **Técnica de Grafos de Causa-Efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

En este caso se escoge dentro del método de Caja Negra la técnica de la Partición de Equivalencia que es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de casos de prueba que hay que desarrollar.

1.8.2 Método de prueba de caja blanca

El método de prueba de caja blanca, denominado a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero de software puede obtener casos de prueba que: (1) garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; (2) ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; (3) ejecuten todos los ciclos en sus límites y con sus límites operacionales; y (4) ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2005) .

Para ejecutar este tipo de pruebas según (Pressman, 2005) se utiliza la **técnica de camino básico:** esta técnica permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

1.8.3 Pruebas de aceptación

La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o un especialista de la aplicación y es conducida a determinar cómo el sistema satisface sus criterios de aceptación, validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio. Está considerada como la fase final del proceso, para crear un producto confiable y apropiado para su uso (Pressman, 2005).

1.9 Conclusiones parciales

Al finalizar el presente capítulo se concluye que:

- ✓ El estudio de los diferentes sistemas para medir el desempeño de las personas en ámbitos académicos, demostró la necesidad de desarrollar una segunda versión del Sistema de Gestión para la Certificación de roles.
- ✓ Se opta por un enfoque de desarrollo ágil, seleccionando XP como metodología de desarrollo de software. Los lenguajes y herramientas de desarrollo seleccionados contribuyen a la implementación de la segunda versión del sistema mencionado.
- ✓ Las métricas y los tipos de pruebas seleccionados permiten identificar deficiencias en los requisitos, en el diseño y en la implementación de las funcionalidades del sistema, con el objetivo de solventarlas y obtener un sistema que contribuya a solucionar el problema planteado.

CAPÍTULO 2: Planificación, diseño e implementación de la propuesta de solución

2.1 Introducción

En el presente capítulo se hace una descripción de la solución propuesta, teniendo en cuenta las fases de Planificación, Diseño e Implementación del ciclo de vida que propone la metodología XP. Se identifican los requisitos funcionales y no funcionales que debe cumplir la solución a desarrollar, posteriormente se confeccionan las HU y demás artefactos definidos por la metodología. Se identifican y organizan las clases relevantes para las funcionalidades del sistema, así como los patrones arquitectónicos y de diseño utilizados para la realización del Portafolio.

2.2 Propuesta de solución

El sistema cumple con una serie de requisitos funcionales y no funcionales, los cuales contribuyen a mejorar la solución antes propuesta, garantizando así, que se perfeccione la ponderación establecida por competencia y se conformen los tribunales de certificación.

Mediante la utilización de la herramienta el profesor de la asignatura PID podrá elaborar un registro con los estudiantes que supervisará en el semestre, y desde el comienzo del período en curso, ir asignándoles tareas a los estudiantes que están bajo su responsabilidad. El estudiante puede acceder a las tareas que le son asignadas y subir para cada una de ellas los artefactos que la misma genera, y de esta manera ir conformando un expediente asociado a un rol que será candidato a certificar. En la medida que el profesor vaya revisando las tareas asignadas y los artefactos, la aplicación debe ser capaz de mostrar el posible nivel de certificación que puede alcanzar el estudiante según la complejidad, criticidad y evaluaciones de las tareas.

El subdirector de formación del Centro de desarrollo podrá gestionar los tribunales de certificación a partir de un registro de profesores, los cuales debe elegir según su experiencia, categoría docente, título académico, grado científico y especialidad. Los profesores seleccionados serán los que conformarán cada uno de los tribunales de certificación, según los roles a certificar. Una vez que son generados los tribunales de certificación, el sistema brinda la posibilidad de intercambiar cada uno de los miembros del mismo. Cada tribunal podrá almacenar las presentaciones y las recomendaciones que les fueron emitidas a los estudiantes que se presentaron a acreditar un rol, además de emitir el nivel de certificación alcanzado por el estudiante y el acta que acredita que certificó dicho rol. El almacenamiento de estas evidencias le brindará al estudiante la posibilidad de optar por el Mérito Científico.

2.2.1 Actores del sistema

Se definen como actores del sistema, a los estudiantes que se incorporan a la asignatura PID en el segundo semestre de tercer año en adelante y a los profesores y especialistas de la universidad. A continuación se definen los actores del sistema.

Tabla 5: Actores relacionados con el sistema y su descripción.

Actores relacionados con el sistema	Descripción
Estudiante	Se encarga de incorporar evidencias relacionadas con las tareas asignadas en el sistema, para su posterior procesamiento. Es el encargado de realizar una solicitud para certificar un rol.
Tutor PID	Se encarga de incorporar y evaluar tareas, las cuales validará el profesor PID antes de ser asignadas a un estudiante.
Profesor PID	Se encarga de subir el plan de formación correspondiente al estudiante que atiende. Es el encargado de validar cada una de las tareas asignadas a los estudiantes y evalúa cada tarea junto con la evidencia que le corresponde.
Administrador del sistema	Se encarga de administrar el sistema, con privilegios y credenciales para realizar cualquier cambio en la aplicación
Subdirector de formación	Se encarga de velar por el correcto funcionamiento del proceso a través de la aplicación, por lo que tendrá una credencial que le permitirá visualizar todos los niveles del sistema, sin que esto implique hacer modificaciones en el mismo. Define el comienzo y el fin de un semestre en curso, además de conformar los tribunales de certificación y asociar un profesor PID a un grupo de estudiantes.

2.3 Identificación de requisitos

La identificación de los requisitos permite tener una correcta visión del producto que se quiere construir. Estos elementos se clasifican en dos grupos: requisitos funcionales (RF) y no funcionales (RNF). La ingeniería de requisitos proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requisitos (Requisitos, 2012). A continuación se especifican las diferentes funcionalidades con las que debe cumplir el sistema.

Tabla 6: Funcionalidades del sistema.

Número	Funcionalidad	Prioridad para el cliente
1	Generar tribunal de certificación de roles	Alta
2	Intercambiar miembro de tribunal de certificación de roles	Alta
3	Intercambiar estudiante de tribunal de certificación de roles	Alta
4	Asociar estudiante a un profesor PID	Alta
5	Mostrar nivel de certificación alcanzado por el estudiante	Alta
6	Generar reporte de datos relevantes	Alta
7	Cambiar credencial	Alta
8	Generar expediente del estudiante para un rol	Alta
9	Adicionar rol a certificar	Media
10	Modificar rol a certificar	Media
11	Listar rol a certificar	Media
12	Adicionar competencia de un rol	Media
13	Modificar competencia de un rol	Media
14	Listar competencia de un rol	Media
15	Adicionar escenario por competencia	Media
16	Modificar escenario por competencia	Media
17	Listar escenario por competencia	Media
18	Editar integrante del tribunal de certificación de roles	Media
19	Clasificar integrante del tribunal de certificación de roles	Media
20	Listar tribunal de certificación de roles	Media
21	Emitir respuesta del tribunal de certificación de roles	Media

22	Alojar artefacto para una tarea	Media
23	Aprobar tarea asignada a estudiante por tutor PID	Media
24	Adicionar curso académico	Media
25	Modificar curso académico	Media
26	Listar curso académico	Media
27	Aceptar solicitud de certificación de roles	Media
28	Mostrar estudiantes certificados	Media
29	Importar plan de formación del estudiante	Media
30	Realizar solicitud de certificación de roles	Media
31	Mostrar registro histórico de tareas del estudiante	Media
32	Mostrar estudiantes por profesor PID	Media
33	Listar solicitudes de certificación de roles	Media
34	Clasificar tutor PID	Media
35	Derogar profesor PID	Media
36	Eliminar rol de certificación	Baja
37	Eliminar competencia de un rol de certificación	Baja
38	Eliminar escenario de una competencia	Baja
39	Mostrar tarea por expediente asignado a un rol	Baja
40	Rechazar tarea asignada a estudiante por tutor PID	Baja
41	Eliminar curso académico	Baja
42	Listar semestre de un curso académico	Baja
43	Rechazar solicitud de certificación de roles	Baja
44	Agregar tarea a un estudiante	Baja
45	Mostrar detalles del estudiante certificado	Baja

Para la obtención de los requisitos, se emplearon técnicas que permitieron hacer este proceso de forma más adecuada y segura. De las técnicas existentes para la captura de requisitos se empleó la entrevista (Anexo 6) y la encuesta (Anexo 2). Como resultado de la aplicación de estas técnicas se obtuvo la Lista de Reserva del Producto.

2.3.1 Lista de reserva del producto

Las listas de reserva del producto en una aplicación son de gran importancia ya que son las cualidades que todo sistema debe poseer para un correcto funcionamiento. A continuación se definen los requisitos no funcionales mediante la lista de reserva del producto:

Tabla 7: Lista de reserva del producto.

RNF1	**Confiabilidad**
	El sistema debe ser preciso con la información que maneja y le proporciona al usuario. Haciendo énfasis en los resultados de los análisis que ejecutará, para evitar errores que puedan incidir negativamente en la toma de decisiones.
RNF2	**Seguridad**
	La seguridad de la base de datos será gestionada mediante la asignación de credenciales para mantener la integridad de los datos en función del nivel de acceso asignado.
RNF3	**Software(Cliente)**
	Las PC clientes deben tener instalado un Navegador Web Mozilla Firefox o Google Chrome.
RNF4	**Software(Servidor)**
RNF4.1	El servidor debe poseer el Sistema Operativo GNU/Linux.
RNF4.2	Se debe contar con un Servidor web Apache versión 2.0 o superior.
RNF5	**Hardware(cliente)**
RNF5.1	Las PC clientes deben poseer requisitos Mínimos como: Procesador 1ghz, con 512 de RAM.
RNF5.2	Cada PC cliente debe tener Conexión de red.
RNF6	**Hardware(servidor)**
RNF6.1	Se debe disponer de un servidor con requisitos Mínimos como: Procesador Intel Pentium Core-2-Duo, con 2GB de RAM.
RNF6.2	Se debe disponer de un Disco duro de 380 GB o superior.
RNF6.3	El servidor debe tener Conexión de red

2.4 Planificación

El ciclo de vida de un proyecto realizado con la metodología XP inicia con la fase de planificación. En esta fase, los clientes definen las historias de usuarios. Al finalizar, el equipo

cuenta con suficiente material de trabajo para producir una primera entrega. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas y tecnologías que serán utilizadas en el proyecto.

Uno de los artefactos que se generan por la metodología de desarrollo XP para la especificación de requisitos del software y las características del sistema son las HU.

2.4.1 Historias de Usuario

La Historia de Usuario es la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es dinámico y flexible. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Estas deben proporcionar sólo el detalle suficiente como para poder hacer razonable la estimación de cuánto tiempo requiere su implementación. También se les asigna un número identificativo, una prioridad en el negocio (alta, media, baja) y la iteración en la que se implementará (Beck, 2001) Para una mejor comprensión de las HU referirse al Anexo 3:

A continuación se muestra una de las HU de prioridad Alta diseñadas para el desarrollo del sistema:

Tabla 8: HU Gestionar rol.

Historia de usuario	
Número:1	**Nombre Historia de Usuario:** Gestionar rol
Modificación de Historia de Usuario Número: ninguna	
Usuario: Administrador	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1 semana

Descripción: • Permite adicionar rol El sistema debe mostrar la opción de adicionar rol, donde el administrador debe llenar el campo descripción del rol, luego debe seleccionar el botón Aceptar para guardar el dato. • Permite modificar rol El sistema debe permitir modificar un rol, donde debe aparecer el mismo campo que cuando adicionó el rol, con la posibilidad de rectificar los errores en el indicador. • Permite eliminar rol El sistema debe permitir eliminar un rol, donde seleccionando la opción eliminar, el sistema muestra un cartel de confirmación de la petición, si selecciona el botón Aceptar se eliminará el rol. • Permite mostrar rol El sistema debe ser capaz de mostrar un rol, donde debe aparecer la descripción de cada rol. • Permite listar rol El sistema debe ser capaz de mostrar un listado de cada uno de los roles a certificar.
Observaciones: solo está autorizado a gestionar los roles, el Administrador del sistema.

En esta fase el cliente establece la prioridad que tendrá cada HU según sus necesidades más inmediatas, luego los programadores realizan una estimación del esfuerzo que se necesita para cada una de ellas. La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las HU y, asociadas a estas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas (release[11]). El cronograma fijado en la etapa de planeamiento se realiza en un número de iteraciones, cada una de ellas tarda de una a cuatro semanas de ejecución.

La planificación se realiza basándose en el tiempo o el alcance. Al planificar por tiempo, se multiplicó el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según el alcance, se dividió la suma de puntos de las Historias de Usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

2.4.2 Desarrollo de iteraciones

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta etapa, generando al final de cada una, un entregable funcional que implementa las HU asignadas a la iteración. Como las HU no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis,

[11] Release: Es un ciclo desde la entrevista con el usuario hasta la obtención de una solución (Beck, 2001).

recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo. Las iteraciones son también utilizadas para medir el progreso del proyecto.

Una vez definidas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se distribuyó la realización del sistema en tres iteraciones, las cuales se describen a continuación de manera más detallada:

Iteración I: esta iteración tiene como objetivo realizar las HU de prioridad Alta referente a generar los tribunales de certificación, asociar estudiantes, realizar método de evaluación del desempeño, generar reportes, expedientes y asignar credencial.

Iteración II: esta iteración tiene como objetivo realizar las HU de prioridad Media referentes a la adición y modificación de los roles, las competencias, los escenarios, los cursos, los semestres y la clasificación edición y emisión de respuestas del tribunal.

Iteración III: esta iteración tiene como objetivo realizar las HU de prioridad Baja las cuales son las referentes a eliminar rol, competencia, escenario, mostrar tareas por expediente, rechazar tarea, eliminar curso y eliminar notificación.

Después de realizada la estimación del esfuerzo y el plan de iteraciones, y continuando los pasos que propone XP, se crea el plan de duración de las iteraciones. Este tiene como objetivo fundamental mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una, según la prioridad asignada por el cliente.

Tabla 9: Plan de duración de las iteraciones.

Iteración	Historias de Usuario	Duración total de las Iteraciones (semanas)
Iteración I	HU(1-8)	5
Iteración II	HU(9-35)	4
Iteración III	HU(36-45)	2
Total	45	11

A continuación se presenta el plan de entregas ideado para la fase de implementación. Atendiendo al mismo se harán entregas del sistema al finalizar cada iteración en la fecha aproximada que se indica en la siguiente tabla:

Tabla 10: Duración del plan de iteraciones.

Iteración	Fecha de Entrega
Iteración I	16 de marzo de 2015
Iteración II	1 de mayo de 2015
Iteración III	18 de mayo de 2015

2.5 Diseño de la Solución

La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que realizarlo lo menos complicado posible, para conseguir un diseño entendible y de fácil implementación, que a la larga costará menos tiempo y esfuerzo desarrollar. Como parte de esta fase, se define el patrón arquitectónico a utilizar para solucionar problemas de arquitectura y se precisan los patrones de diseño que se van a emplear para remediar problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño.

La arquitectura del Sistema de Gestión para la Certificación de Roles V 2.0, está basada en el patrón arquitectónico MVC, el cual separa la lógica de negocio (el modelo), del controlador y la presentación (la vista), por lo que se consigue una mayor organización y un mantenimiento más sencillo de la aplicación.

2.5.1 Arquitectura del sistema

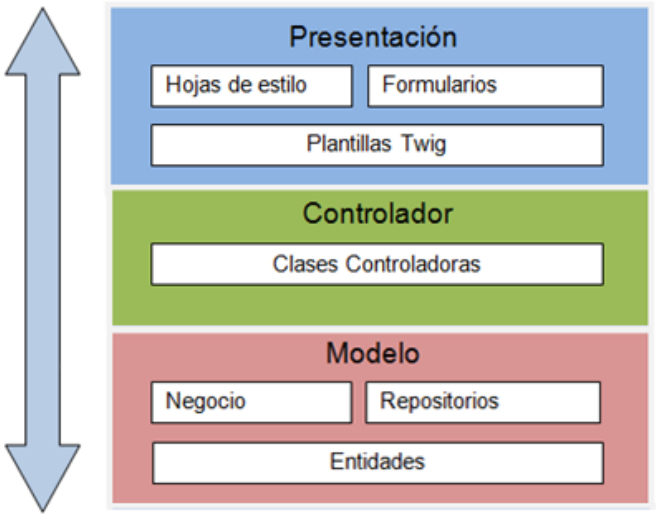


Figura. 1 Arquitectura del sistema

La arquitectura del sistema en el nivel superior, o sea la capa de presentación, encapsula las interfaces de usuario representadas por las clases twig, hojas de estilo, funciones JavaScript y formularios necesarios para la interacción con el cliente. La capa controlador contiene las clases controladoras que se encargan de dar respuesta a las peticiones realizadas por el usuario. La capa de negocio contiene las clases de negocio, que tienen como principal función separar la lógica de negocio del resto de la aplicación y obtener una mayor reutilización del

código. Esta capa se encarga de recibir las peticiones, procesar la información, hacer pedidos a las clases Repositorio y devolver respuestas a las clases Controladoras, las que a su vez las envían a las clases enmarcadas en la capa vista.

Inicialmente el usuario interactúa con la interfaz (Vista). El controlador recibe la petición de la acción solicitada y gestiona el evento. El controlador accede al modelo actualizándolo o buscando la información requerida. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario y mostrar los datos del modelo para generar la interfaz apropiada. (Ver Figura. 2 Patrón MVC)

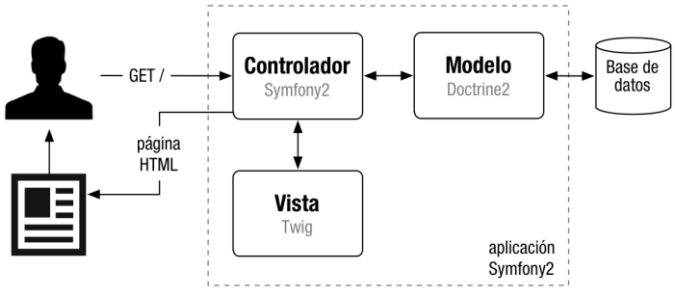


Figura. 2 Patrón MVC

A continuación se muestra la estructura de carpetas del proyecto, siguiendo el patrón arquitectónico MVC obtenido con la ayuda del marco de trabajo Symfony2.

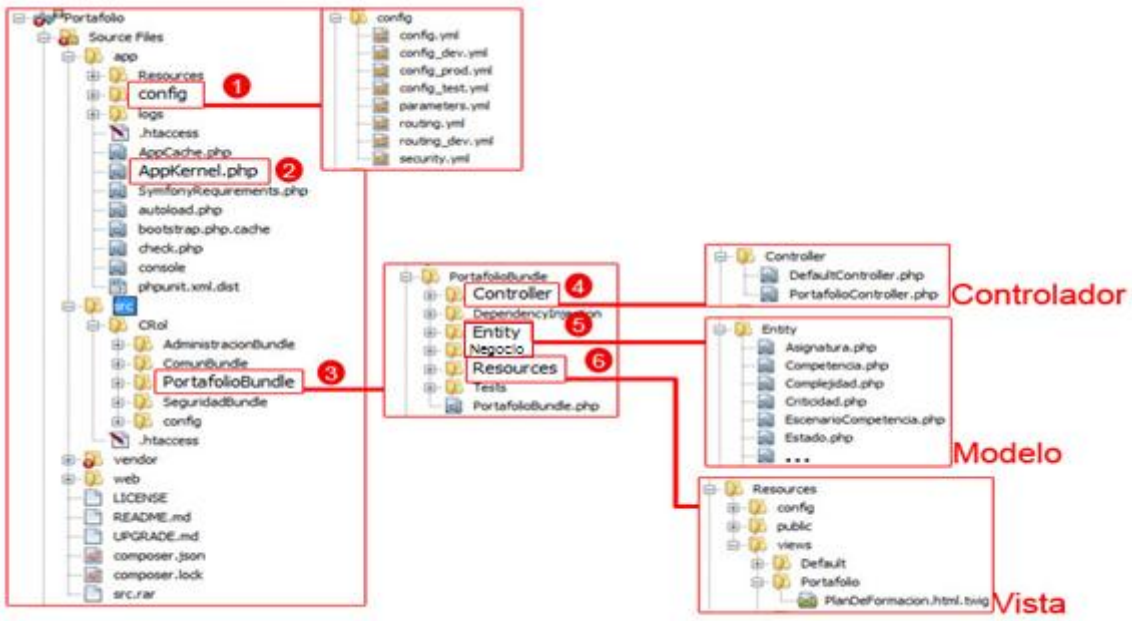


Figura. 3 Estructura de clases

Symfony 2 posee un controlador frontal el cual es el encargado de crear el kernel de la aplicación mediante una instancia de la clase **(2)** y es la responsable de toda la configuración.

Ofrece las rutas de los módulos o Bundles que el usuario necesita para satisfacer su necesidad y que se encuentran en la carpeta config **(1).** Es el núcleo de Symfony 2 y por tanto uno de los componentes fundamentales para su correcto funcionamiento. En la carpeta src se localiza el proyecto llamado Crol, nombre que se deriva de Certificar un Rol y que identifica al sistema creado. El sistema está organizado en cuatro bundles (módulos, según su denominación en inglés): Administración, Común, Portafolio y Seguridad, los cuales se pueden observar dentro de la estructura de clases mostrada anteriormente. Los módulos poseen una organización común, la cual se muestra en **(3)** dentro de la misma se localizan las carpetas con los componentes específicos de la arquitectura:
Controller **(4)**: posee los ficheros con los códigos de las clases Controladoras.
Entity **(5)**: en la carpeta Entity se encuentran las Entidades del Modelo y en la carpeta Negocio se encuentran las clases que contienen los métodos que acceden a los datos de la base de datos.
Resources **(6):** dentro de esta se encuentran las carpetas con los CSS, los JS y las TWIG que conforman la Vista.
A continuación se realiza una descripción de cada uno de los Módulos junto con una explicación de su funcionamiento y de sus funcionalidades.

Administración: este módulo contiene las funcionalidades necesarias para gestionar los usuarios que tendrán acceso al sistema. A través de esta vista se podrá gestionar un grupo de credenciales que permiten la accesibilidad al sistema, modificar los datos y rol de un usuario existente. El sistema permitirá listar o mostrar los usuarios registrados en el mismo. Contiene una funcionalidad para especificar la fecha del curso escolar, así como los semestres que pertenecen a dicho curso. En este módulo también se gestionan todos los datos que resultan básicos en un sistema, así como la gestión de asignaturas encargada de organizar las asignaturas por semestre en curso, la gestión de los roles a certificar, con cada una de las competencias específicas y los estados de competencias de estas. Además este módulo contiene las funcionalidades necesarias para gestionar los tribunales de certificación. Se podrá conformar cada uno de los tribunales en dependencia del rol a certificar, listar los profesores que formará parte del mismo y se podrá editar los parámetros que permitirán medir la función que debe desempeñar cada uno de los profesores que conformarán el tribunal. El tribunal podrá emitir recomendaciones y podrá subir cada una de las presentaciones expuestas por los estudiantes.

Seguridad: este módulo de seguridad se encarga de la autenticación de los usuarios en el sistema y de la gestión de las trazas. El equipo de desarrollo decidió gestionar esta funcionalidad independiente para lograr una seguridad de acuerdo a lo que plantean las

normas de Seguridad Informáticas en la UCI. El módulo autentica los usuarios del dominio uci.cu cuyos permisos valida el Módulo de Administración. La gestión de las trazas es un sistema que va registrando las operaciones que van realizando los usuarios en la aplicación, de esta manera se conoce qué usuario inserta, actualiza, elimina o simplemente entra en la aplicación y en qué momento lo hace.

Común: el módulo común gestiona las vistas y las entidades que son comunes en el sistema permitiendo la reutilización de código. Este módulo además, gestiona las notificaciones del sistema ayudando a la comunicación entre usuarios.

Portafolio: gestiona las entidades y funcionalidades del negocio permitiendo una independencia del código. Las clases de este módulo poseen responsabilidades específicas a cumplir, de acuerdo al negocio para la certificación de roles. A través de este módulo se podrá calcular el nivel alcanzado por un estudiante para ser clasificado como candidato a certificar uno o varios roles, utilizando para ello el método de factores ponderados. El sistema mostrará a los estudiantes un listado con las tareas asignadas, donde podrán alojar las evidencias para estas tareas, del mismo modo los profesores podrán visualizar a sus estudiantes, evaluar sus tareas y aceptar las solicitudes para la certificación.

2.5.2 Tarjetas Clases, Responsabilidad y Colaborador (CRC)

Las tarjetas CRC fueron las bases para la obtención del modelo entidad relación. Cada tarjeta se convirtió en objeto, sus responsabilidades en métodos públicos y sus colaboradores en llamados a otras clases. Las tarjetas CRC constituyen una primera aproximación a los objetos que luego se van a utilizar en el desarrollo del sistema. En XP el proceso de diseño es iterativo por lo que la creación de las tarjetas no es en un mismo tiempo, se crean según las iteraciones y se le añaden responsabilidades y colaboradores según se haga necesario (Casas, y otros, 2009).

Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. A continuación se muestra un ejemplo de tarjetas CRC confeccionadas durante la fase de diseño.

Tabla 11: Tarjeta CRC número 1.

RolController	
Responsabilidades • Adicionar rol • Modificar rol • Eliminar rol • Visualizar rol • Listar rol • Listado de roles por expediente	**Colaboradores** • Rol • Persona • Tipo Rol • Usuario

• Competencias por rol • Evaluar rol	

2.5.3 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 nuestro 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 (González, y otros, 2003). A continuación se muestran los patrones utilizados en la solución:

Patrones de diseño GRASP

- ✓ **Experto:** consiste en asignar una responsabilidad al experto en información, o sea, a la clase que cuenta con la información necesaria para cumplir la responsabilidad (Viscoti, y otros, 2004). El uso de este patrón se evidencia en las clases del negocio y del modelo, que poseen funciones concretas de acuerdo con la información que gestionan.
- ✓ **Creador:** el propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido (Viscoti, y otros, 2004). Este patrón se evidencia en la clase Tarea, la cual es la única que puede crear objetos de tipo Evidencia.
- ✓ **Bajo acoplamiento:** consiste en tener las clases lo menos relacionadas posible, ya que en caso de producirse una modificación en alguna de ellas, se tenga una repercusión mínima en las demás (Viscoti, y otros, 2004). Esta característica permitió potenciar la reutilización y disminuyó la dependencia entre las clases.
- ✓ **Alta cohesión:** se encuentra evidenciado en la implementación de las clases que forman parte de la capa del modelo; las cuales están formadas por diferentes funcionalidades que se encuentran estrechamente relacionadas.
- ✓ **Controlador:** un Controlador es un objeto no destinado al usuario que se encarga de manejar un evento del sistema. Es una clase que se encarga de controlar el flujo de eventos del sistema (Viscoti, y otros, 2004). Ejemplo de ello se evidencia en las clases de negocio, entre ellas la clase TareaGtr que maneja todas las peticiones relacionadas con las tareas.

Patrones de diseño GoF

Patrones Estructurales: Los patrones estructurales se ocupan, de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.

- ✓ **Fachada**: provee de una única interfaz para acceder a un sistema completo, que

actúa como único punto de acceso al mismo, y hace que éste sea más fácil de utilizar. Ejemplo de esto se evidencia en la clase PortafolioGtr que es la encargada del acceso a los datos de la entidad Tarea en la base de datos.

- ✓ **Decorador:** es aplicado a la generación de vistas, la solución que ofrece dicho patrón es añadir funcionalidades adicionales a las plantillas. Por ejemplo, añadir el menú y el pie de página a las plantillas que lo requieran, se trata de decorar las plantillas con elementos adicionales reutilizables.

2.6 Codificación de la solución

En esta fase se genera todo el código fuente necesario para satisfacer las HU definidas para la solución. Al inicio de cada iteración, se lleva a cabo una revisión del plan de iteraciones y se modifica de ser necesario. Todas las HU son traducidas en tareas de programación.

Para llevar a cabo la correcta implementación de las HU se deben definir por parte del equipo de desarrollo las Tareas de Ingeniería (TI) que se realizan en cada una de las iteraciones. Las TI también conocidas como tareas de implementación permiten a los desarrolladores obtener un nivel de detalle más avanzado que el que propicia las HU. A continuación se describen una las Tareas de Ingeniería pertenecientes a la primera iteración.

Tabla 12: HU Tarea de ingeniería Adicionar rol.

Tarea de Ingeniería	
Número Tarea:1	**Historia de Usuario:** 1, Adicionar rol.
Nombre Tarea: Implementar RF_Adicionar_ rol	
Tipo de Tarea: *Desarrollo*	**Puntos Estimados:** 1
Fecha inicio:09/02/2015	**Fecha Fin:** 9/02/2015
Programador Responsable: *Eliodanis Maceo Rosales, Felinda Rosabel León Mendoza.*	
Descripción: se debe implementar el requisito funcional Adicionar rol, el cual debe permitir al administrador registrar cada uno de los roles que conforma el plan de formación.	

Para una correcta comprensión y ejecución de la codificación resulta imprescindible el uso de estándares de codificación.

2.6.1 Estándares de codificación

Los estándares de codificación son aquellos que permiten entender de manera rápida y sencilla el código empleado en el desarrollo de un software. Garantizan el mantenimiento óptimo de dicho código por parte del programador (Calleja, y otros, 2009)). A continuación se

muestran algunas pautas del estándar definido por el equipo de desarrollo así como ejemplos de su uso.

1. Todas las nomenclaturas a utilizar se definen en idioma español.
2. Los identificadores para las variables y los parámetros se escriben con letras en minúsculas y en caso de ser un nombre compuesto se divide cada palabra con un guion bajo.

```
private $id_tribunal;
private $presidente;
```

3. En caso de que los métodos se nombren con una sola palabra, esta se escribe en minúsculas y en caso de ser un nombre compuesto, las palabras que lo conforman se escriben juntas, de la segunda en adelante se escriben con letra inicial mayúscula. Se emplea la notación Camello variante (LowerCamelCase).
4. El bundle del módulo Portafolio se escribe con el nombre Portafolio seguido de la palabra Bundle (PortafolioBundle).

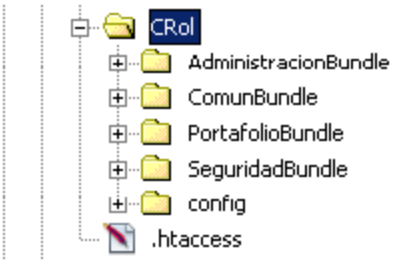


Figura. 4 Estructura de carpeta

5. Los nombres de las clases se escriben con la primera letra de cada palabra que lo compone en mayúscula, haciendo uso de la notación Camello, con la variante UpperCamelCase

```
class PersonaController extends BaseController {
```

6. Las clases formularios comienzan con el nombre del formulario según su función, seguido de la palabra Type (BuscarEstudianteType).

```
class BuscarEstudianteType extends AbstractType {
```

7. Las clases gestoras de negocio comienzan con el nombre del gestor seguido de Gtr(PortafolioGtr).

```
class PortafolioGtr extends BaseGtr {
```

8. Las funciones deben ser llamadas sin espacios entre el nombre de la función, el signo de paréntesis y el primer parámetro, espacios entre cada coma por parámetro y sin

espacios entre el último paréntesis, el signo de paréntesis cerrado y el signo de punto y coma.

```
epository('PortafolioBundle:Semestre')->findOneBy(array('anno'=>$estudiante->getAnno()));
tRepository('PortafolioBundle:Expediente')->findOneBy(array('portafolio'=>$portafolio, 'semestre'=>$semestre));
```

9. Se hizo uso de llaves para ganar en comprensión del código

```
if (!is_null($estudiante)) {
    $profesor->removeEstudiante($estudiante);
    $estudiante->removeProfesor($profesor);
    $this->salvarActualizar($profesor);
    $this->salvarActualizar($estudiante);
}
```

10. Cadenas de texto entre comillas: PHP tiene dos formas de poner las cadenas de texto. Con comillas simples y con comillas dobles. La diferencia es que si se usa comillas dobles y se coloca dentro del texto un nombre de variable, el compilador lo interpretará y reemplazará por su valor. Por ésta razón siempre se va a usar comillas simples a menos que se necesite hacer la interpolación de variables que permiten las dobles. Por supuesto hay casos especiales donde es mejor usar dobles comillas (Como cuando se usan caracteres de escape \ intensivamente).

```
$profesores = $this->em->getRepository('ComunBundle:Profesor')->findAll();

$html.="<h5>Cantidad de escenarios registrados:"
```

Los estándares de codificación permitieron establecer un estilo de programación homogéneo permitiendo que cualquier persona que consulte el código lo pueda entender en menos tiempo. A continuación se describe el método factores ponderados, el cual refleja el empleo de algunos de los estándares de codificación establecidos.

2.6.2 Descripción del método factores ponderados

El método de factores ponderados, en dependencia de las evidencias que el estudiante posea en su expediente, emite una evaluación que indica el posible nivel de certificación que este podría alcanzar. Inicialmente se evalúan los escenarios, luego se evalúan las competencias y por último el rol. El nivel de certificación se mide por 4 indicadores: no certificado, básico, intermedio y avanzado. El sistema muestra además el porciento de avance para el rol. A continuación se establecen los umbrales junto a su correspondiente codificación.

1. 0 al 59 No certificado
2. 60 al 70 Básico
3. 70 al 85 Intermedio
4. 85 al 100 Avanzado

Para cada uno de los umbrales se evidencia el nivel que hasta el momento tiene el estudiante, con una barra que muestra el progreso de las evaluaciones. Esta barra de progreso muestra varios colores, en dependencia de las evidencias que posea el estudiante en la aplicación. El umbral no certificado se muestra de color rojo, el básico se muestra de color naranja, el intermedio de color azul y el avanzado de color verde. A continuación se muestra una imagen de la codificación para estos umbrales y la barra de progreso.

```
if ($resultado >=0.85) {
    $nivel = $this->em->getRepository('PortafolioBundle:Nivel')->findOneBy(array('id_nivel' => Util::avanzado));
}
if ($resultado < 0.85 and $resultado > 0.7) {
    $nivel = $this->em->getRepository('PortafolioBundle:Nivel')->findOneBy(array('id_nivel' => Util::intermedio));
}
if ($resultado <= 0.7 and $resultado >= 0.6) {
    $nivel = $this->em->getRepository('PortafolioBundle:Nivel')->findOneBy(array('id_nivel' => Util::basico));
}
if ($resultado < 0.6) {
    $nivel = $this->em->getRepository('PortafolioBundle:Nivel')->findOneBy(array('id_nivel' => Util::no_certifica));
}
```

Figura. 5 Umbrales en el método factores ponderados

Figura. 6 Barra de progreso para el umbral avanzado

Figura. 7 Barra de progreso para el umbral intermedio

Figura. 8 Barra de progreso para el umbral básico

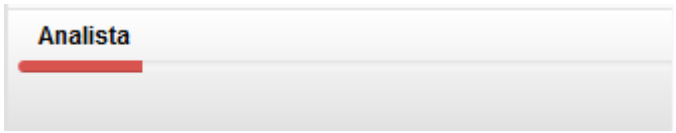


Figura. 9 Barra de progreso para el umbral no certificado

2.7 Conclusiones parciales

Al finalizar el presente capítulo se arriba a las siguientes conclusiones:

- ✓ Después de realizar el análisis del sistema, quedaron definidas las Historias de Usuario, proporcionando una comprensión detallada de las funcionalidades de la aplicación.

- ✓ La propuesta de arquitectura del sistema se sustenta en un conjunto de componentes reutilizables que tienen como base el patrón arquitectónico MVC, lo que conforma un sistema robusto y flexible a cambios.

- ✓ La obtención de los requisitos funcionales y no funcionales, permitió definir el comportamiento y restricciones del sistema para su implementación.
- ✓ El empleo de patrones de diseño garantizó una solución que tiene como premisa la reutilización de código durante la fase de implementación del software.

CAPÍTULO 3 Validación de la propuesta de solución

3.1 Introducción

En este último capítulo se verifica y valida el correcto funcionamiento del sistema, la verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica y en la validación se ejecutan un conjunto de actividades, que aseguran que el software construido se ajusta a los requisitos del cliente. Para la comprobación final del sistema se aplican técnicas de validación de requisitos y métricas de software, además se realizan pruebas unitarias y pruebas de aceptación.

3.2 Validación de los requisitos

La validación de los requisitos tiene como objetivo comprobar que estos son correctos. Esta fase debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación, con el costo que eso conlleva. Es muy importante asegurar la validez de los requisitos antes de comenzar el desarrollo del software. Para ello debe hacerse una comprobación de la correspondencia entre las descripciones iniciales y la definición de los requisitos realizada, para verificar que responden a lo que desea el usuario final (Cristiá, 2011). Para llevar a cabo este proceso, se aplicaron las siguientes técnicas de validación de requisitos:

- ✓ **Revisión de requisitos:** Se realizaron reuniones para localizar errores en el documento. Donde se agregaron requisitos y se modificaron otros.
- ✓ **Generación de casos de prueba de aceptación*:*** para validar los requisitos funcionales de la solución, se diseñaron casos de pruebas de aceptación para cada una de las Historias de Usuario.

3.2.1 Métrica para requisitos

Los requisitos del software son la base de las medidas de calidad. En la disciplina de requisitos se tuvo en cuenta la métrica para medir su estabilidad, especificidad y grado de validación. A continuación se aplican cada una de estas métricas.

Aplicación de la métrica estabilidad de requisitos:

La misma se calcula como:

$$ETR = \left[\frac{RT - RM}{RT}\right] \times 100$$

Donde:

1. ETR: valor de la estabilidad de los requisitos.

2. RT: total de requisitos definidos
3. RM: número de requisitos modificados, que se obtienen como la sumatoria de los requisitos insertados, modificados y eliminados

Esta métrica ofrece valores entre 0 y 100. El mejor valor de ETR es el más cercano a 100 porque mostrará que no se están realizando cambios sobre los requisitos, son estables y por tanto es confiable trabajar el análisis y diseño sobre ellos.

Teniendo en cuenta que se identificaron un total de 45 requisitos funcionales, de los cuales 4 resultaron modificados (1 por eliminación, 2 por modificación, 1 por inserción) se calcula:

ETR = [(45– 4) / 45 * 100]= 91,11

Como resultado se obtuvo un valor de 91,11. Dicha cifra demuestra que no se han realizado cambios significativos sobre los requisitos, son estables, y por tanto es confiable el análisis y diseño sobre ellos.

Aplicación de la métrica especificidad de los requisitos:

La misma se calcula como:

$$ER = {}^{n_{ui}}/_{n_r}$$

Donde:

1. ER: grado de especificidad de los requisitos.
2. nui: número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.
3. nr: cantidad de requisitos en una especificación.

El valor de esta métrica debe estar siempre entre 0 y 1. Mientras más cerca de 1 esté el valor de ER mayor será la consistencia de la interpretación de los revisores para cada requisito y menor será la ambigüedad en la especificación de los requisitos.

Q**1** = n**ui** /n**r** = 45 /45 = 1

Como resultado de la aplicación de la métrica se obtuvo un valor de 1. Dicha cifra demuestra que los requisitos son entendibles, específicos y que no poseen ambigüedades.

Aplicación de la métrica grado de validación:

La misma se calcula como:

$$VR = {}^{n_c}/_{(n_c + n_{nv})}$$

Donde:

1. VR: grado de validación de los requisitos.
2. nc: número de requisitos que se han validado como correctos.
3. nnv: número de requisitos no validados aún.

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de corrección en la definición de los requisitos.

Q**3** = n**c** / (n**c** + n**nv**) = 45/(45+ 0) = 1

La aplicación de esta métrica dio como resultado 1, por lo tanto se concluye que se realizó una correcta definición de los requisitos.

3.3 Validación del diseño

Para comprobar la calidad del diseño del sistema se emplearon las métricas: Relaciones entre Clases y Tamaño Operacional de Clase (TOC).

3.3.1 Relaciones entre clases

La métrica RC está dada por el número de relaciones de uso de una clase con otra. Permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase, teniendo en cuenta las relaciones existentes entre ellas.

- ✓ **Acoplamiento**: consiste en el grado de dependencia o interconexión de una clase o estructura de clases con otras, está muy ligada a la característica de Reutilización.
- ✓ **Complejidad del mantenimiento**: consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ✓ **Cantidad de pruebas**: un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Para determinar el grado de afectación de los atributos de calidad que mide la métrica RC es necesario determinar la Cantidad de Relaciones de Uso (CRU) que posee cada una de las clases a medir. Una vez determinada la CRU, se procede a calcular el promedio de las mismas y teniendo ambos valores, según los criterios expuestos en el capítulo 1, se determina la incidencia de los atributos de calidad en cada una de las clases.

La aplicación del instrumento de evaluación de la métrica RC para el número total de relaciones arrojó los resultados que se plasman en el gráfico de la siguiente figura.

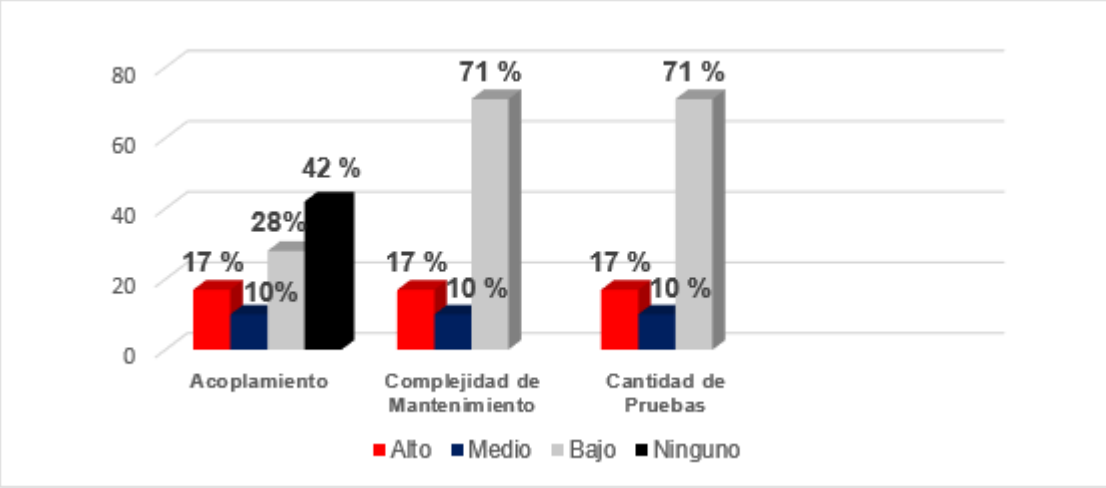


Figura. 10 Resultado de la métrica RC

Acoplamiento: según los resultados que se muestran, el (42%) de las clases no posee relaciones de uso por lo que la mayoría de las clases no tienen valores de acoplamiento, validando una realización correcta del diseño.

Complejidad de mantenimiento: según los resultados que se muestran en la figura anterior, el (71%) de las clases se comportan de forma satisfactoria pues, son de fácil soporte.

Cantidad de pruebas: luego de aplicar la métrica se obtuvo que el (71%) de las clases poseen un bajo grado de esfuerzo a la hora de realizar cambios, rectificaciones y pruebas de software.

Según lo analizado anteriormente, los valores de RC se comportan de forma satisfactoria siendo discretos en la mayoría de las clases. Este resultado implica una disminución del acoplamiento y mayor facilidad de mantenimiento de las mismas, además de ser factible el diseño realizado.

3.3.2 Tamaño operacional de clases TOC

Al aplicar la métrica TOC se tendrá en cuenta un conjunto de atributos de calidad que se relacionan a continuación:

- ✓ **Responsabilidad**: consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio.
- ✓ **Complejidad de implementación**: consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ✓ **Reutilización**: consiste en el grado de reutilización presente en una clase o estructura de clases, dentro de un diseño de software.

Para determinar el valor de los atributos de calidad, se debe determinar la cantidad de procedimientos que posee cada una de las clases a medir. Una vez determinado la CP se procede a calcular el promedio del mismo y, según los criterios expuestos en el capítulo 1, se determina la incidencia de los atributos de calidad en cada una de las operaciones de las clases.

La aplicación del instrumento de evaluación de la métrica TOC para el número total de operaciones arrojó los resultados que se plasman en el gráfico de la siguiente figura.

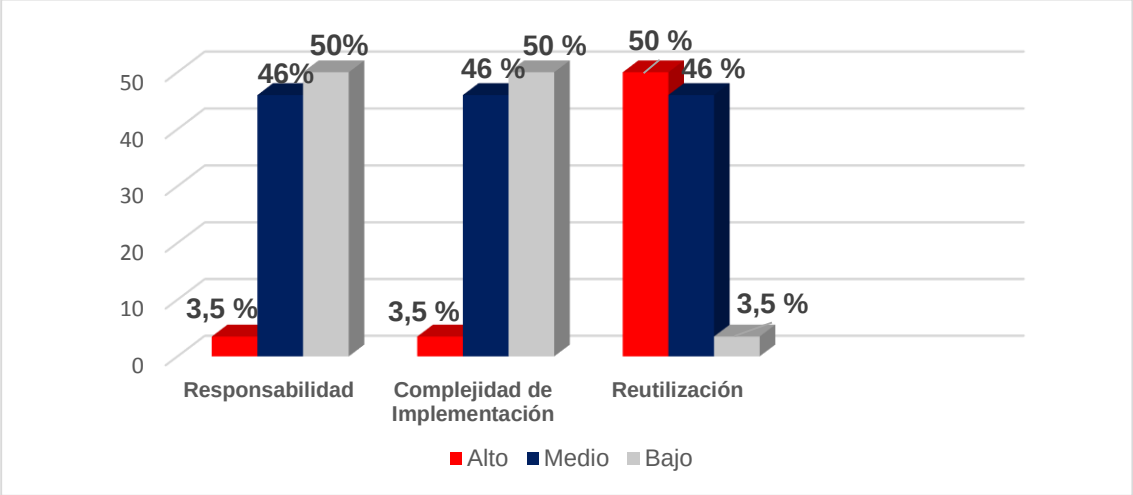


Figura. 11 Resultado de la métrica TOC

Responsabilidad: luego de aplicar la métrica se obtuvieron resultados satisfactorios que reflejan una responsabilidad baja con un valor del 50%.

Complejidad de Implementación: después de haberse realizado la medición de la métrica, arrojó también resultados positivos ya que la complejidad de las clases es baja en un 50 %.

Reutilización: se obtuvieron valores que según muestra la gráfica de la figura anterior se comporta en un nivel alto con un 50%.

Haciendo un análisis de los resultados obtenidos para los atributos de la métrica TOC se puede observar que el atributo reutilización cuenta con un porciento alto. Se demuestra que el componente cuenta con una elevada reutilización, baja responsabilidad y complejidad en el diseño propuesto. Por lo que se concluye que los resultados obtenidos en esta métrica son positivos.

3.4 Verificación del sistema

Las pruebas tienen como objetivo valorar y mejorar la calidad de los productos del trabajo generado durante el desarrollo y modificación del software. Según (Pressman, 2000) verificación *es el conjunto de actividades que aseguran que el software implemente*

correctamente una función específica, y la validación *es un conjunto diferente de actividades que aseguran que el software construido corresponde y satisface los requisitos del cliente.* Se tuvieron en cuenta los siguientes niveles de pruebas:

- ✓ Nivel de Unidad
- ✓ Nivel de Aceptación

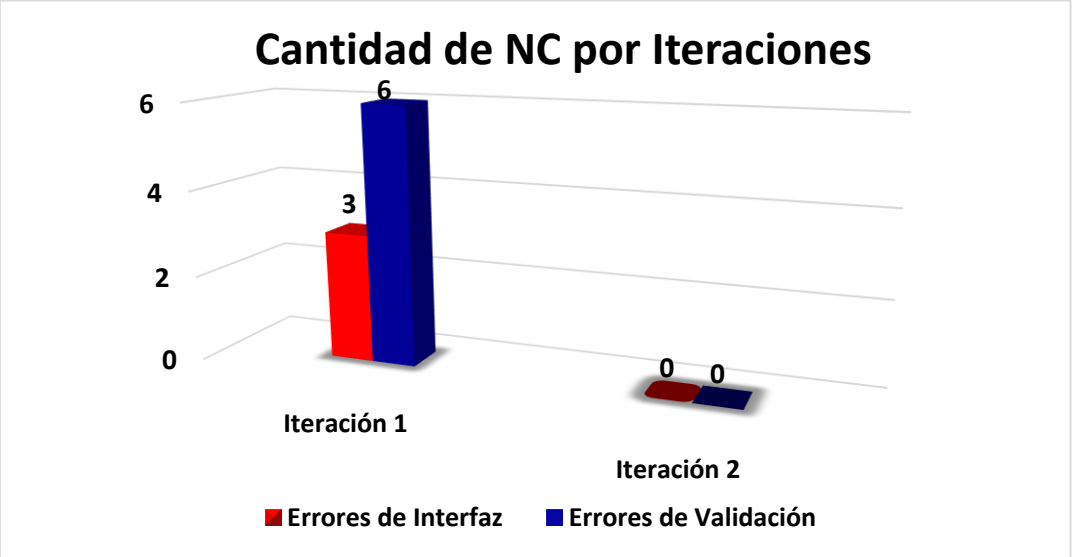
3.4.1 Nivel de unidad

Las pruebas unitarias son aplicadas para verificar que el software cumple los requisitos funcionales y también son empleadas para asegurar la calidad del código entregado. Además, son la mejor forma de detectar fallas tempranamente en el desarrollo y está demostrado que mientras más pronto se encuentren los errores, menos costará corregirlos. A este nivel se realizan las pruebas de funcionalidad, utilizando los métodos de prueba de Caja Blanca y Caja Negra para comprobar que tanto el código como la interfaz no contengan errores y se ejecutan adecuadamente.

3.4.1.1 Método de pruebas de caja negra

Para la realización de las pruebas de caja negra se empleó la técnica partición de equivalencia que garantizó efectividad al examinar los valores válidos e inválidos de las entradas existentes en el software. A continuación se brinda un ejemplo de un caso de prueba de uno de los requisitos que componen la funcionalidad gestionar rol, el resto de estos se incluye en el documento 01_Diseño de casos basados en requisitos.

Escenario	Descripción	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Adicionar rol con campos correctos.	Permite adicionar un rol con sus datos correspondientes.	V Analista	El sistema debe adicionar el rol en el panel central.	Seleccione la opción Adicionar Rol en el menú central superior.
EC 1.2 Adicionar rol con campos incompletos	El sistema no permite adicionar un rol porque existen campos obligatorios vacíos.	I	El sistema muestra un mensaje de !Error! Existen campos obligatorios vacíos.	
EC 1.3 Adicionar rol con campos incorrectos	El sistema no permite adicionar un rol porque existen campos incorrectos .	I 45h@2345f	El sistema no permite escribir números.	



Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Descripción	Campo de texto	No	El campo es obligatorio y está representado con * en el sistema, admite solo letras.

Figura. 12 Caso de prueba adicionar rol

Luego de aplicar las pruebas de caja negra por el Grupo de Calidad del Centro de Gobierno Electrónico (CEGEL) se obtuvieron los siguientes resultados:

Figura. 13 Cantidad de NC por iteraciones

Las no conformidades encontradas en la primera iteración estaban relacionadas con la validación de los datos que se introducen en el sistema y errores en las interfaces. Se pudo comprobar en la segunda iteración realizada que estos errores fueron corregidos y que la aplicación desarrollada funciona correctamente.

Después de concluidas las pruebas, el grupo de calidad CEGEL liberó la aplicación Sistema de Gestión para la Certificación de Roles v2.0, entregándose al equipo de desarrollado el Acta de Liberación Interna de Productos del Software en la que consta que la aplicación está apta para ser utilizada (Anexo 4).

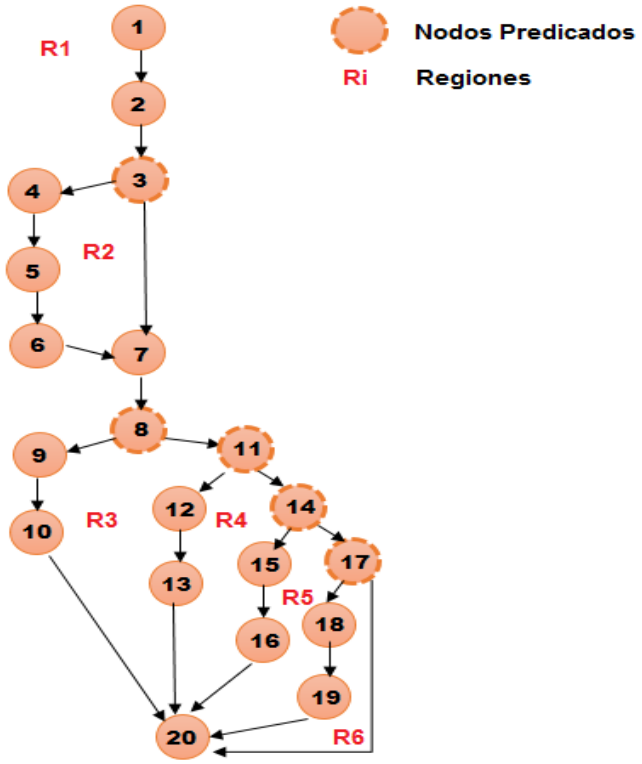
3.4.1.2 Método de prueba de caja blanca

Para aplicar el método de pruebas de caja blanca se empleó la técnica del camino básico. Esta permitió obtener una medida de la complejidad lógica para el diseño de los casos de pruebas y usar esta medida como guía para la definición de un conjunto básico de caminos de ejecución. Se tomó como ejemplo el método *evaluarCompetenciaPorEstudiante*, perteneciente a la clase *RolGtr.* Para ello se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

```
public function evaluarCompetenciaPorEstudiante($rol,$id_tipo_competencia, $id_usuario) {
    $tipo_competencia = $this->em->getRepository('PortafolioBundle:TipoCompetencia')->findOneBy(array('id_tipo_competencia' => $id_tipo_competencia));
    $tipo_escenarios = $tipo_competencia->getTiposEscenariosCompetencias();
    $competencia = $this->em->getRepository('PortafolioBundle:Competencia')->findOneBy(array('tipo_competencia' => $tipo_competencia, 'rol' => $rol));
1   $escenarios = $this->em->getRepository('PortafolioBundle:EscenarioCompetencia')->findBy(array('comptencia' => $competencia));
    $valor_escenarios = array();
    $i = 0;
    $valor = 0;
2   foreach ($escenarios as $escenario) {

3       if (in_array($escenario->getTipoEscenarioCompetencia()->getIdTipoEscenarioCompetencia(), $valor_escenarios) == false) {
4           $valor_escenarios[$i] = $escenario->getTipoEscenarioCompetencia()->getIdTipoEscenarioCompetencia();
5           $i = $i + 1;
6           $valor = $valor + $this->obtenerMaxAporteEscenario($escenario->getTipoEscenarioCompetencia()->getIdTipoEscenarioCompetencia(), $id_usuario)
        }
    }

7   $eval_comp = $valor / 5;

8   if ($eval_comp >=0.85) {
9       $nivel = $this->em->getRepository('PortafolioBundle:Nivel')->findOneBy(array('id_nivel' => Util::des_excelente));
10      $eval_comp=5;

    }
11  elseif ($eval_comp < 0.85 and $eval_comp >0.7) {
12      $nivel = $this->em->getRepository('PortafolioBundle:Nivel')->findOneBy(array('id_nivel' => Util::des_intermedio));
13      $eval_comp=4;

    }
14  elseif ($eval_comp <= 0.7 and $eval_comp >= 0.6) {
15      $nivel = $this->em->getRepository('PortafolioBundle:Nivel')->findOneBy(array('id_nivel' => Util::bajo_desarrollo));
16      $eval_comp=3;

    }
17  elseif ($eval_comp < 0.6) {
18      $nivel = $this->em->getRepository('PortafolioBundle:Nivel')->findOneBy(array('id_nivel' => Util::no_desarrollada));
19      $eval_comp=2;

    }
20  return $eval_comp;
}
```

Fig. 14 Código fuente del método evaluarCompetenciaPorEstudiante

A continuación se realiza el grafo de flujo partiendo del código tomado:



Luego de haber realizado la construcción del grafo de flujo se procede a calcular la complejidad ciclomática mediante la siguiente fórmula que se describe a continuación.

En la fórmula V (G) representa el valor del cálculo.

V (G) = (A - N) + 2

Donde **A** es el número de aristas y **N** es el número de nodos contenidos en el grafo.

V (G) = (24 - 20) + 2

V (G) = 6

El número de regiones del grafo es igual a la complejidad ciclomática.

El cálculo efectuado anteriormente dio como resultado que, la complejidad ciclomática es de 6, este valor indica que existen 6 posibles caminos por donde el flujo puede circular y determina el número de casos de prueba que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. A continuación se representan los caminos básicos por los que puede transitar el flujo:

- **Camino 1:** 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 -9 - 10 -20
- **Camino 2:** 1 - 2 - 3 - 7 - 8 - 9 - 10 - 20
- **Camino 3:** 1 - 2 - 3 - 7 - 8 - 11 - 12 - 13 - 20
- **Camino 4:** 1 - 2 - 3 - 7 - 8 - 11 - 14 - 15 - 16 - 20
- **Camino 5:** 1- 2 - 3 - 7 - 8 - 11 - 14 - 17 - 18 -19 -20
- **Camino 6:**1 - 2- 3 – 7 - 8 - 11 - 14 - 17 -20

Luego de establecidos los caminos básicos se procede a realizar los casos de prueba para cada uno de ellos, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino. A continuación se presenta un ejemplo de los 6 casos de pruebas realizados a esta funcionalidad.

Tabla 13: Caso de prueba para el camino 2.

Entrada	Que exista el rol a certificar con una competencia asociada y que el usuario esté registrado en el sistema.
Resultados esperados	Devuelve la evaluación de la competencia, dado por los valores 2, 3,4 y 5.
Condiciones	in_array($escenario->getTipoEscenarioCompetencia()->getIdTipoEscenarioCompetencia(), $valor_escenarios) == false $eval_comp >=0.85 $eval_comp < 0.85 and $eval_comp >0.7 $eval_comp <= 0.7 and $eval_comp >= 0.6 $eval_comp < 0.6

Caso de prueba para el Camino 2: una vez ejecutados todos los casos de pruebas obtenidos a través de la aplicación de la técnica camino básico, se concluye que los mismos fueron probados satisfactoriamente demostrando que el código generado no presenta ciclos infinitos y no existe código innecesario en el sistema desarrollado.

3.5 Validación del sistema

Con la validación del sistema se pretende comprobar que el software cumple las expectativas que el cliente espera. Para llevar a cabo la validación de la aplicación se aplicaron las pruebas de aceptación por cada Historia de Usuario definidas en la etapa de planificación.

3.5.1 Nivel de aceptación

Las pruebas de aceptación son destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente final. Estas pruebas aseguran el comportamiento del sistema y especifican los aspectos a probar cuando una Historia de Usuario ha sido implementada correctamente (Riva, 2014).

A cada uno de las Historias de Usuario se le realizaron pruebas de aceptación. Estas pruebas son reflejadas mediante casos de pruebas de aceptación, las cuales están conformadas por 7 parámetros, que dan información acerca de la prueba realizada. Los parámetros a medir son:

- ✓ **Código:** muestra un identificador para cada prueba realizada (normalmente se pone el nombre del componente seguido de las letras PA: pruebas de aceptación).
- ✓ **Nombre de la historia de usuario:** Indica el nombre de la prueba.
- ✓ **Descripción:** se describe cuál es la funcionalidad que se va a medir del componente al que se le esté realizando la prueba.
- ✓ **Condiciones de Ejecución:** indica cuáles son las condiciones que se tienen que cumplir para que el componente realice correctamente la funcionalidad que se va a medir.
- ✓ **Entrada / Pasos de ejecución:** indica los pasos a seguir para realizar la prueba.
- ✓ **Resultados esperados:** muestra cuál es el resultado que se obtendría de un correcto funcionamiento de la prueba.
- ✓ **Evaluación de la prueba:** indica el estado de la prueba si es satisfactoria o insatisfactoria.

A continuación se muestra un ejemplo de caso de prueba de aceptación.

Tabla 14: PA Importar plan de formación.

Caso de prueba de aceptación	
Código:PA6-HU06	**Historia de Usuario:** Adicionar curso académico.
Nombre: Adicionar curso académico.	
Descripción: Se registra un curso académico en el sistema.	
Condición de ejecución: El Subdirector de formación debe estar autenticado en la aplicación.	
Entrada/Pasos de ejecución: ✓ El usuario presiona el botón Adicionar curso. ✓ El usuario introduce la descripción del curso. ✓ El usuario introduce la fecha de inicio del curso. ✓ El usuario introduce la fecha de fin del curso. ✓ El usuario presiona el botón aceptar.	
Resultado esperado: Se registra el curso académico.	

Evaluación de la prueba: Prueba satisfactoria.

3.6 Validación de las variables

Para darle cumplimiento al objetivo planteado se aplicó la misma encuesta (Anexo 5) a los 29 integrantes de los tribunales de certificación a nivel UCI. Esta vez se obtuvieron resultados satisfactorios, demostrándose que:

- ✓ La evaluación que se obtiene para cada estudiante, utilizando como forma de evaluación el método de factores ponderados, es la más cercana a la realidad porque tiene en cuenta la importancia relativa para cada escenario y competencia en cada rol a certificar, además de ser extraído estos valores de las encuestas realizadas a los tribunales de certificación.
- ✓ La informatización del proceso de conformación del tribunal, permite realizar un proceso justo y flexible a cambios, ya que se tiene en cuenta las competencias profesionales de cada profesor para conformar el tribunal, además el sistema brinda la posibilidad de intercambiar cada uno de los integrantes que forman parte del mismo.

3.7 Conclusiones parciales

Al concluir el presente capítulo se evidencia que:

- ✓ Se aplicaron técnicas de validación de requisitos permitiendo asegurar la validez de los mismos en el proceso de desarrollo de software.
- ✓ La aplicación de las métricas para requisitos, arrojó como resultado que los mismos son estables y por tanto, es confiable el diseño efectuado sobre ellos.
- ✓ La aplicación de métricas de diseño, demostró que las clases del diseño poseen bajo acoplamiento, que existe además una baja responsabilidad y complejidad de implementación y una alta reutilización en el diseño propuesto.
- ✓ Se aplicaron pruebas unitarias utilizando los métodos de prueba de caja blanca y de caja negra lo que permitió verificar que el software cumple los requisitos funcionales.
- ✓ Se validaron las variables que forman parte del problema de la investigación, demostrando que el sistema desarrollado perfecciona la ponderación establecida por competencia en cada rol a certificar, así como la conformación y funcionamiento del tribunal.

Conclusiones Generales

Con la culminación de la presente investigación se concluye que:

- ✓ El estudio de los referentes teóricos y de los sistemas existentes contribuyó a profundizar en la fundamentación de la solución, así como la incorporación del método de factores ponderados, permitiendo perfeccionar la ponderación establecidas por competencias.
- ✓ Las tecnologías y herramientas seleccionadas, así como la utilización de XP como metodología de desarrollo, en correspondencia con los criterios de selección definidos, contribuyeron a la obtención de una solución informática para la problemática identificada.
- ✓ Con la utilización de patrones de diseño y patrones arquitectónicos, se logró implementar un sistema de acuerdo a los estándares y modelos utilizados en el desarrollo de software que responden a las necesidades del cliente.
- ✓ Mediante el diseño y aplicación de los casos de prueba se logró valorar los resultados y verificar que las funcionalidades cumplieran con las expectativas y necesidades del cliente.

Recomendaciones

- ✓ Desarrollar un módulo de reportes sobre los estados del proceso de certificación de roles, de manera que contribuya a la obtención de información pertinente y oportuna para la toma de decisiones correctivas, que aporten al mejoramiento de los resultados en la certificación de roles.
- ✓ Desarrollar un módulo para la ponderación del mérito científico de manera que, contribuya a identificar los estudiantes potenciales a obtener esta distinción.

Bibliografía

Barret, Helen. 2000. *Create your Own Electronic Portfolio: Using 2off-the-shelf software to showcase your own or student work. Learning and Leading with Technology.* 2000.

Beck, Kent. 2001. *Extreme Programming Explained. s.l. : s.l. : Embrace Change.* 2001. sl: Pearson Education.

Benito, Luis and Díaz, Martín. 2011. *Implementación del Portafolio Digital de la Universidad de las Ciencias Informáticas.* 2011.

Bonanata, Maximiliano. 2013. *Programación y algoritmos.* s.l. : MP ediciones S.A, 2013. p. Página 17.

Buenas Tareas. 2011. Buenas Tareas. *pequesmile25.* [Online] 11 18, 2011. [Cited: Febrero 21, 2015.] http://www.buenastareas.com/ensayos/Comparacion-De-Factores/3133339.html.

Calleja, M. A and Arias, Manuel. 2009. Carmen. Etándares de codificación. [Online] Mayo 7, 2009. [Cited: Abril 27, 2015.] http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf.

Carlos, Frank and Rafael, Carlos. 2014. Biblioteca. [Online] Junio 2014. [Cited: Marzo 21, 2015.] http://catalogoenlinea.uci.cu/.

Casas, Sandra and Reinaga, Héctor. 2009. *Aspectos tempranos: Un enfoque basado en tarjetas CRC.* Argentina : s.n., 2009.

Cristiá, Maximiliano. 2011. Introducción a la ingeniería de requerimientos. [Online] Agosto 2011. [Cited: Junio 4, 2015.] http://www.fceia.unr.edu.ar/~mcristia/publicaciones/ingreq-a.pdf.

Desarrollo de Competencias. 2012. DC. [Online] Febrero 4, 2012. http://web.archive.org/web/20120204001748/http://www.desarrollodecompetencias.com/que_es_desarrollo_por_competencias.

Desarrollo web, D. 2013. DSW. [Online] 2013. [Cited: Abril 24, 2015.] http://www.desarrolloweb.com/articulos-copyleft/articulo-metricas-de-software.html.

Díaz, Daniel, Marisol and Verónica López Guzmán. 2007. *Soluciones de software libre para aplicaciones de bases de datos.* 2007.

Fornaris, Sánchez, Maite and Rabí, Dayanis Alcantara. 2009. *Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas.* 2009.

García, Adolfo Arreola. 2003. Propuesta de una guía de métricas para evaluar el desarrollo de los sistemas de información geográfica. [Online] 2003. [Cited: Abril 18, 2015.]

http://vinculando.org/articulos/sociedad_america_latina/propuesta_guia_de_medidas_para_evaluacion_sistema_informacion.html.

García, Fernando. 2013. fergarciac. [Online] 2013. [Cited: Marzo 26, 2015.] https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

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

IEEE. 2015. [Online] 2015. http://www.computer.org/cms/Computer.org/Computer.org/s2esc/s2esc_pols/SP-06_Vocabulary_Objectives.html.

—. 2008. *Métricas del software.* 2008.

Jeremy. 2014. Características de Css3. . [Online] 2014. [Cited: Abril 10, 2015.] http://www.css3.com.

Kabir, M. 2003. *Anaya Multimedia, La Biblia del Servidor Apache 2.* 2003.

León, Hernández, et al. 2011. *El Proceso de Investigación Científica.* Ciudad de La Habana. : s.n., 2011.

Lincke, Lundberg and W., J. y Löwe. 2009. *Comparing software metrics tools. International Symposium on Software Testing and Analysis.* Berlin : s.n., 2009.

Mehdi Achour y otros. 2014. ¿Qué es PHP? Manual . [Online] Junio 7, 2014. [Cited: Abril 6, 2015.] http://www.php.net/manual/es/intro-whatis.php.

Mora, Pérez, Oscar and Marc, Gibert Ginestà. 2014. *Bases de datos en PostgreSQL.* 2014. s.l.: UOC.

Otto, Jacob Mark. 2014. [Online] 2014. [Cited: Abril 11, 2015.] http://getbootstrap.com/.

Pacheco, N. 2013. Manual de Twig. [Online] 2013. [Cited: Abril 3, 2015.] http://gitnacho.github.io/Twig/.

Peña, Juan Manuel Fernández. 2010. Pruebas de software. [Online] 2010. [Cited: Abril 25, 2015.] www.uv.mx/personal/jfernandez/files/2010/07/Cap3-Caminos.pdf.

Pérez, Leonardo Cabrera. 2012. *Integración de los instrumentos de evaluación para la gestión de las evidencias en la plataforma educativa Zera.* 2012.

Polo, Macario and Villafranca, Daniel. 2008. Introducción a las aplicaciones Web con Java. [Online] 2008. [Cited: Abril 20, 2015.] http://www.inf-cr.uclm.es/www/mpolo/asig/0708/tutorJavaWeb.pdf.

Potencier, Fabien and Zaninotto, François. 2008. *Symfony, la guía definitiva.* 2008.

Potencier, Fabien. 2010. *Symfony 2.1 el libro oficial.* 2010.

Pressman. 2000. *Ingeniería de software. Un enfoque práctico.* 2000.

Pressman, Roger S. 2007. *Ingeniería de Software. Un enfoque práctico. Pág. 384.* 2007.

—. 2005. *Ingeniería del Software. Un enfoque práctico 6ta edición.* Nueva York : McGraw-Hill, 2005. 0072853182.

Real Academia Española. 2014. Drae. [Online] Octubre 2014. [Cited: Febrero 18, 2015.] http://lema.rae.es/drae/srv/search?id=jadYhVT31DXX2vknwZ9Y.

—. 2015. wordreference. [Online] 2015. http://www.wordreference.com/definicion/rol.

Requisitos. 2012. Memorias dentro del desarrollo de software. Qué es el levantamiento de requisitos. [Online] Febrero 7, 2012. [Cited: Junio 4, 2015.] http://phigux.blogspot.com/2012/02/que-es-el-levantamiento-de.html.

Revista vinculando. 2010. Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica. [Online] Enero 4, 2010. [Cited: Abril 8, 2015.] http://vinculando.org/?s=m%C3%A9tricas+para+requisitos.

Riva, José. 2014. Actas de los Talleres de Ingeniería del Software y Base de Datos. [Online] 2014. [Cited: Mayo 2, 2015.] http://www.sistedes.es/TJISBD/Vol-3/No-4/PRIS09.pdf.

Rodriguez, Renata. 2013. *El desarrollo de la práctica reflexiva sobre el quehacer docente, apoyada en el uso de un portafolio digital, en el marco de un programa de formación para académicos de la Universidad Centroamericana de Nicaragua.* 2013.

Sistema Nacional De Certificación De Competencias Laborales, S. 2008. SNCCL. [Online] Junio 25, 2008. [Cited: Marzo 6, 2015.] http://www.fdf.cl/biblioteca/presentaciones/2008/01_productividad/descargas/01_Erika_Madariaga.pdf.

Sommervile, Ian. 2005. *Ingeniería de Software. Septima Edición.* 2005.

Suárez, Mery. 2014. gobiernodecanarias. [Online] 2014. [Cited: Marzo 13, 2015.] http://www3.gobiernodecanarias.org/medusa/ecoblog/esuasan/rubricas-o-matrices-de-evaluacion/.

Universidad de Panamá. 2012. Propuesta de metodología para la certificación de roles durante la formación del ingeniero en ciencias informáticas. [Online] Julio 23, 2012. [Cited: Marzo 5, 2015.] www.laccei.org/LACCEI2012-Panama/RefereedPapers/RP054.pdf.

Universidad de Sistemas De Información. 2003. *ASPECTOS TÉCNICOS Y LEGALES.* 2003.

Universidad Peruana de los Andes. 2010. UPLA. [Online] 2010. [Cited: Febrero 10, 2015.] www.planificacion.upla.edu.pe/.

Velthunis, Mario G Piattini. 2002. *Análisis y diseño de aplicaciones informáticas de gestión.* Madrid : s.n., 2002.

Viscoti, Astudillo Marcello and Hernán. 2004. Fundamentos de Ingeniería de Software. [Online] Septiembre 9, 2004. [Cited: Abril 26, 2015.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

Wesley, Gamma and E, Addison. 2001. *Desing Patterns.* 2001.