Universidad de las Ciencias Informáticas

Facultad 3

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

Título: Sistema para la gestión de los procesos administrativos de CEIGE. Módulo: Gestión de los procesos del Departamento de Práctica Profesional.

Autora: Célida Bagarotti Abreu

Tutoras: Ing. Olga Yarisbel Rojas Grass

Ing. Aneyvis Hernández Chinea

La Habana, 2017

Año 59 de la Revolución

DECLARACIÓN DE AUTORÍA

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

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

_____

Célida Bagarotti Abreu

Firma de la autora

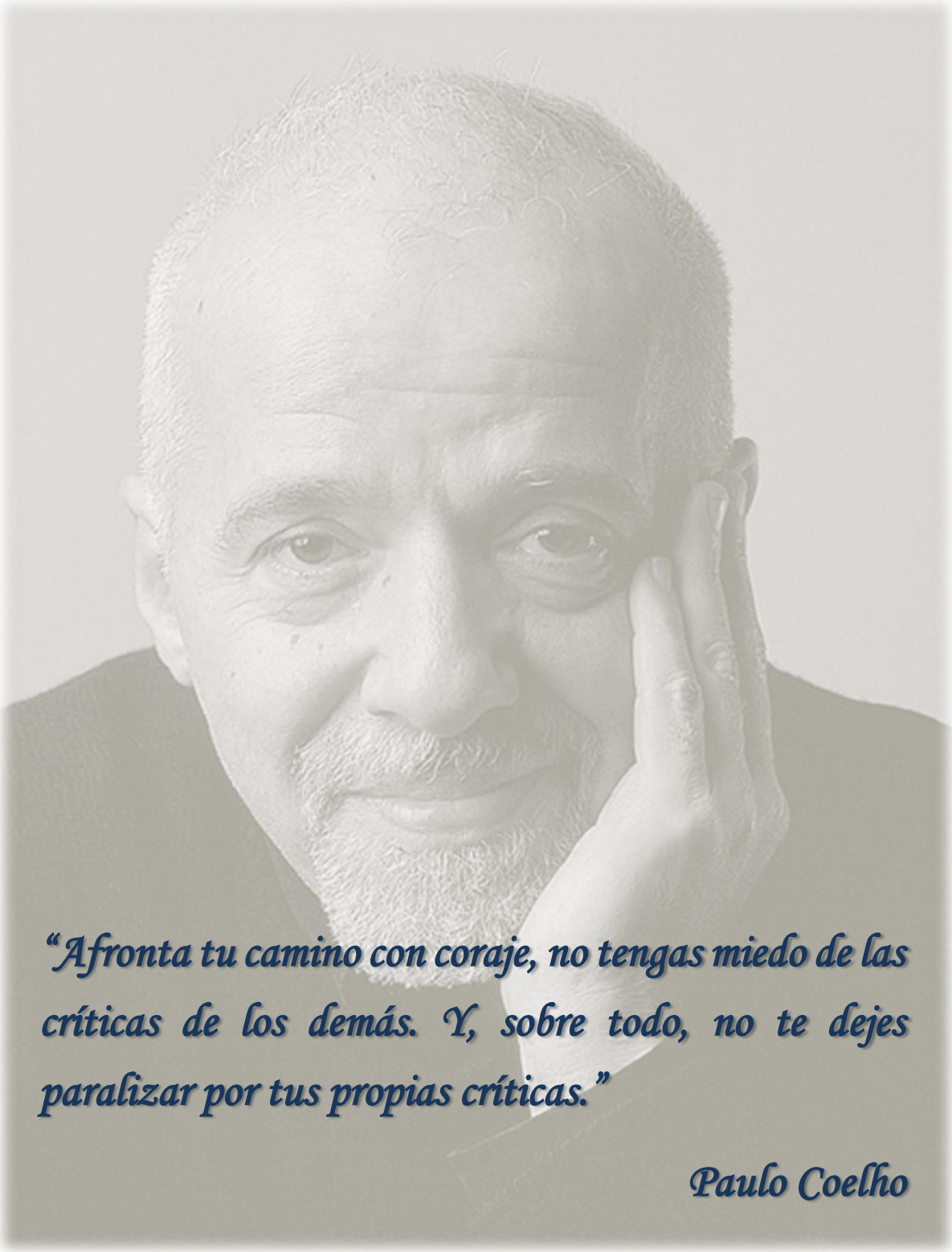
_____ _____

Ing. Olga Yarisbel Rojas Grass

Firma de la Tutora

Ing. Aneyvis Hernández Chinea

Firma de la Tutora


"Afronta tu camino con coraje, no tengas miedo de las críticas de los demás. Y, sobre todo, no te dejes paralizar por tus propias críticas."
Paulo Coelho

Luego de tantos años de esfuerzo y de sacrificio, hoy culmino una etapa muy importante en mi vida, que no hubiese sido posible sin el apoyo y la comprensión de mis seres queridos, amigos y familia. Le agradezco por todo:

A la persona más importante que tengo en esta vida, a esa que siempre me demostró que yo si podía, aun cuando yo misma no creía en mí. Ese ángel que desde el momento que vine a este mundo ha estado conmigo y me ha enseñado que en esta vida nunca es tarde para lograr lo que se quiere: A ti mi MAMI te agradezco por todo.

A mi sustento y mi alegría cuando más deprimida estaba, a ese que desde lejos siempre me cuida y está pendiente de mis pasos. Al que siempre tiene un chiste cuando le cuento que me pasa algo malo: A mi PADRE bello, muchas gracias.

A mi hermanita Ángela, por esas noches de risa que siempre tenemos, por ayudarme a estudiar para mis pruebas hasta tarde en la noche, por estar pendiente de mí y por darme ánimos siempre.

A mi querida abuela Antonia, por ser mi pilar fundamental durante mi carrera, por sus consejos siempre oportunos, por su ánimo y su fortaleza. Si hoy estoy aquí, es por ti mi abue.

A mi tía Odalys, que desde lejos siempre estuvo cerca de todos mis pasos, pendiente de mis resultados y de mis actividades, dándome su ayuda como podía y queriendo estar aquí conmigo para ayudarme.

A mi prima Dianellys por sus llamadas casi todas las noches queriendo saber siempre todo lo que me sucedía y haciéndome reír con sus cuentos.

A mis tíos Jorge Luis y Gelza por ser mis padres durante este tiempo, por enseñarme tantas cosas de la vida y por siempre darme un poquito de su conocimiento para ser una persona mejor.

A mi primo Amed que ha sido mi hermano, ayudándome a rebasar los obstáculos más difíciles en este tiempo y siempre queriéndome ayudar de más. Muchas gracias mi primo.

A mis abuelos paternos por su cariño incondicional y por su preocupación.

A mi novio por ser mi apoyo, mi alegría, mi fuerza y por darme siempre un poco de su optimismo. Por demostrarme que yo sí puedo y por ayudarme aun cuando no tenía tiempo para sí mismo. Muchas gracias mi amor. Te amo.

A mi tutora Olga, por estar conmigo durante todo este proceso, por siempre tener un tiempo para ayudarme, por todos sus consejos y críticas que me hicieron una mejor persona. Por ser más que mi tutora, por ser mi amiga.

A mi tutora Aneyvis por sus correcciones y sus consejos. Muchas Gracias.

A Víctor y Joseito por aguantarme tantos días y por ser como mi familia en esta última etapa.

A una persona que ha sido muy importante para mí en esta universidad, a mi amiga Rosalina, por darme la oportunidad de conocer una faceta mía que no sabía que existía, por dejarme ser parte de su vida, por sus consejos, sus palabras siempre alentadoras, por tenerme siempre en cuenta, muchas gracias. Te quiero mucho.

A la profesora Dailién, por ayudarme cuando más lo necesité, por sus consejos, por su apoyo y por estar siempre pendiente de mí.

A Yoandry por ayudarme siempre en todo, por enseñarme tanto y por compartir tantos momentos buenos y de estrés para lograr este resultado.

A mis compañeras de apartamento Claudia, Yanet, Dalilis, Kenia y Yara por todos los momentos de alegría y de estrés que pasamos juntas.

A mis compañeros de aula por formar parte de este periodo tan importante de mi vida.

A Pavel por sus consejos, sus correcciones y por siempre darme un poquito de su amor.

A mis profesores que ayudaron a mi formación y son el principal motivo por el que estoy hoy aquí.

A los miembros del tribunal y a la oponente por sus recomendaciones oportunas y por ayudarme a convertirme en una profesional.

A todos, muchas gracias.

A mi abuelo Jorge que siempre me cuida desde arriba.

A toda mi familia por su apoyo incondicional siempre.

A Ivón, por el ánimo que siempre me dio y por su alegría.

La práctica laboral investigativa está fundamentada en el desarrollo de la disciplina de Práctica Profesional que constituye la Disciplina Principal Integradora. La misma está compuesta por cinco asignaturas que se imparten en el ciclo profesional en tercero, cuarto y quinto año, donde los estudiantes están incorporados a proyectos productivos. El Centro de Informatización de Entidades pertenece a la Facultad 3 de la Universidad de las Ciencias Informáticas y en él, el Departamento de Práctica Profesional, que tiene como objetivo realizar el control y seguimiento de los estudiantes pertenecientes al centro mientras cursan las asignaturas de la disciplina de Práctica Profesional. Este proceso genera mucha información que se encuentra aislada, lo que dificulta ofrecer de forma rápida y oportuna cualquier dato o reporte referente al mismo. Por tanto, el objetivo de la investigación es desarrollar un módulo para la gestión de los procesos de control y seguimiento del Departamento de Práctica Profesional del Centro de Informatización de Entidades. Para guiar el proceso de desarrollo del sistema se utilizó la metodología de software Variación de AUP para la universidad y en la implementación se emplearon tecnologías y herramientas de software libre. El módulo elaborado permite realizar el control y seguimiento de los estudiantes en los Proyecto de Investigación y Desarrollo.

Palabras claves: control, proyecto de investigación y desarrollo, seguimiento.

Introducción

En los últimos años la informática ha alcanzado un alto nivel en los diferentes sectores de la sociedad, desempeñando un papel protagónico en esta nueva revolución de avance tecnológico que se está desarrollando. Paulatinamente, se ha ido sumergiendo en cada sector facilitando el desempeño de los mismos, aplicándoles a los problemas existentes soluciones informáticas. El uso de sistemas web es una de las soluciones factibles, que se desarrollan con el objetivo de mejorar el funcionamiento de los procesos del entorno. En la medida que ocurre esto en el ámbito universal se puede afirmar que la web ha evolucionado a grandes pasos, esta ha dejado de ser un medio para la publicación de información y contenidos, para convertirse en una plataforma de diseño y desarrollo de aplicaciones informáticas distribuidas. (Balmaseda, 2008).

Cuba también forma parte de este avance tecnológico y como evidencia de ello fue creada la Universidad de las Ciencias Informáticas (UCI), estructurada por seis facultades. La peculiaridad de la carrera de Ingeniería en Ciencias Informáticas, creada por el Comandante Fidel Castro, es que, a partir del tercer año de estudio la docencia está vinculada a la producción de software.

La vinculación docencia-producción se evidencia a través de la disciplina Práctica Profesional (PP) que juega distintos roles en cada uno de los ciclos que componen la actividad formativa de la universidad: el ciclo básico y el ciclo profesional. En el primero tiene como objetivo crear las habilidades básicas que debe desarrollar el Ingeniero en Ciencias Informáticas. Durante el ciclo profesional, el estudiante se inserta en los proyectos productivos, en los que adquiere competencias técnicas a partir de los roles que desempeña en los equipos de proyecto. Las asignaturas que se definen en la disciplina de PP en el plan de estudio de Ingeniero en Ciencias Informáticas están definidas como Proyecto de Investigación y Desarrollo (PID). ((UCI), 2014)

En la Facultad 3 de la universidad están ubicados con dos centros de producción, uno de ellos es el Centro de Informatización de Entidades (CEIGE), el cual está estructurado por varias áreas. Una de estas es el Departamento de Práctica Profesional donde se realiza el control y seguimiento de los estudiantes vinculados a proyectos, donde reciben las asignaturas de la disciplina de PP en el ciclo profesional. En estos momentos todo el contenido se encuentra registrado en varias hojas de cálculo, que debido a la cantidad de información almacenada son muy extensas y se dificulta el trabajo a la hora de buscar datos referentes a algún estudiante específico. No se cuenta con un sistema de reportes, que es la mayor necesidad que tiene hoy el colectivo de profesores del Departamento, carencia que les hace engorroso el trabajo. Toda la actualización de los datos se realiza de forma manual y es necesaria la creación de

nuevos documentos cada vez que se necesita la información general de algún estudiante, para luego exportarlo a formato PDF.

Luego de analizada la problemática anterior se plantea como **problema a resolver**: ¿Cómo contribuir a mejorar la gestión de los procesos de seguimiento y control de los estudiantes vinculados a la PID en el Departamento de Práctica Profesional de CEIGE?

El **objeto de estudio** estará centrado en la gestión de los procesos de control y seguimiento de la Práctica Profesional siendo el **campo de acción** la gestión del proceso docente de las asignaturas de Proyecto de Investigación y Desarrollo en el Centro de Informatización de Entidades.

El **objetivo general** de la investigación es desarrollar un módulo para la gestión de los procesos del Departamento de Práctica Profesional de CEIGE, lo que se desglosa en los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación mediante el estudio y el análisis de los principales referentes teóricos acerca del control y seguimiento de los estudiantes universitarios en la práctica profesional.
2. Realizar el análisis e implementación de un módulo que gestione los procesos de control y seguimiento de los estudiantes en la Práctica Profesional de CEIGE.
3. Validar el objetivo de la investigación mediante la aplicación de técnicas, métricas y pruebas.

Con el fin de dar solución al problema planteado y alcanzar el objetivo trazado se proponen las siguientes **tareas de investigación**:

- Análisis de soluciones similares existentes.
- Estudio y selección de las técnicas, herramientas y metodologías a emplear en el desarrollo de la solución.
- Redacción del diseño teórico metodológico de la investigación.
- Descripción de la propuesta de solución, los actores que interactuaran con el sistema, así como los requerimientos funcionales y no funcionales a implementar en la aplicación.
- Análisis y diseño de la propuesta de solución mediante la confección de los artefactos definidos en la metodología de desarrollo seleccionada.
- Estudio de patrones arquitectónicos y de diseño a emplear en el desarrollo del software.
- Implementar el módulo para la gestión de los procesos de Práctica Profesional de CEIGE.
- Estudio sobre los niveles y métodos de prueba que se pueden aplicar para la validación del software.

- Validación funcional de la solución mediante las pruebas diseñadas y documentación de los resultados.

Como **idea a defender** se plantea que: Si se desarrolla un módulo para la gestión de los procesos del Departamento de Práctica Profesional de CEIGE, se mejora la gestión de los procesos de control y seguimiento de los estudiantes vinculados a la asignatura de PID.

Para desarrollar la investigación se emplearon los siguientes **métodos de investigación científica**:

Métodos teóricos:

- **Histórico-lógico**: mediante el estudio del estado del arte de Sistemas de gestión de la información académica, se podrá conocer la evolución histórica y el desarrollo actual de los mismos.
- **Analítico-Sintético**: facilita el estudio de las fuentes bibliográficas utilizadas en la investigación, con el objetivo de elaborar el marco teórico, identificar los elementos relacionados con la solución propuesta y brindar un análisis de los sistemas relacionados al campo de acción.
- **Modelación**: muestra de forma abstracta la solución ya que muestra los diferentes elementos que la componen a través de modelos y diagramas.

Método empírico

- **Entrevista:** se realiza para obtener información sobre las deficiencias que existen hoy en el Departamento de PP y para saber los elementos a tener en cuenta en el desarrollo de la solución.

El presente documento está estructurado en los siguientes capítulos:

Capítulo 1: Fundamentación Teórica: En este capítulo el lector podrá encontrar información sobre los principales conceptos abordados en la investigación. Se realiza un estudio a diferentes sistemas, que posibilitan ampliar el conocimiento en cuanto a las funcionalidades que contienen, permitiendo una mejor toma de decisiones en el diseño y la implementación del módulo. Se define la metodología de desarrollo de software, las herramientas y las tecnologías a utilizar.

Capítulo 2: Propuesta solución: En el capítulo se presenta la propuesta de solución al problema de la investigación. Se obtiene los productos de trabajo resultantes del desarrollo de las actividades ejecutadas en las disciplinas de modelado del negocio, requisitos, análisis y diseño que se definen en la metodología.

Capítulo 3: Implementación y pruebas: En este capítulo se realiza la implementación de la solución. Se aplican las pruebas para validar el correcto funcionamiento de las funcionalidades del sistema desarrollado y se analizan los resultados de las pruebas aplicadas.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se enuncian los conceptos fundamentales relacionados con la práctica profesional, para favorecer la comprensión de la investigación. Se realiza un estudio del estado del arte de sistemas que gestionan información relacionada al control y seguimiento de los estudiantes durante el ciclo profesional. Además, se define la metodología y se analizan las características de las herramientas y tecnologías a utilizar en el desarrollo de la solución.

1.2 Conceptos fundamentales asociados a la problemática

La **práctica profesional** es una disciplina que permite que el estudiante fortalezca y concrete sus competencias profesionales para desarrollarlas en la escuela y el aula. Además, promover en él, una actitud reflexiva y crítica que le permita replantear su docencia utilizando con pertinencia los conocimientos teórico-metodológicos y técnicos que ha adquirido a través de los cursos que componen el currículo, así como los que él mismo se ha procurado a partir de la búsqueda e interés para profundizar y ofrecer respuestas a las exigencias que la práctica le plantea , con la finalidad de tener mejores resultados en cada una de sus intervenciones. (Superior, 2012)

La **práctica profesional**, suele constituirse como el primer paso de un estudiante o de un recién graduado en el mercado laboral. Se trata de una etapa que combina cuestiones típicas de un empleo (la necesidad de alcanzar un cierto grado de productividad, la obligación de acatar las órdenes de un superior) con elementos vinculados a la formación y al aprendizaje. (Julián Pérez Porto, Ana Gardey, 2015)

Luego de analizadas las definiciones anteriores se decide trabajar con el primer criterio planteado, debido a que es el que mejor se ajusta al entorno en que se desarrolla el negocio.

La actividad que realiza el profesor en su función de **tutor**, es encauzada al desarrollo integral desde el punto de vista, intelectual, profesional y humano de los estudiantes universitarios. Así se proyecta en función de que adquieran no sólo saberes, sino además competencias, para el desarrollo de su proceso de aprendizaje a lo largo de la carrera y durante su ejercicio profesional. (Nieto, 2005)

Según (Valladares, 2016) se puede definir **tutor** como el educador profesional que asume la responsabilidad de la formación integral de profesionales en formación.

Por las características del segundo concepto planteado, se decide trabajar con el mismo debido a que es el que mejor se ajusta al entorno universitario.

1.3 Estudio de sistemas informáticos

En el proceso de búsqueda de sistemas informáticos existentes con propósitos similares a los deseados se definieron los siguientes criterios a evaluar: sistemas basados en tecnología web, centralización de la información (se desea que toda la información referente al estudiante que se encuentra recibiendo la PID esté almacenada en un mismo sitio), ubicación de la práctica profesional o laboral, registro de evaluación y la relación entre estudiantes y tutores. Estos criterios permitieron identificar sistemas informáticos con características similares a las necesidades del Departamento de PP, permitiendo conocer el objetivo y las funcionalidades que poseen los mismos. A continuación, se relacionan los sistemas que por sus características forman parte del estudio.

PRISMA

PRISMA es un sistema de información para gestionar los estudios de toda la Universidad Politécnica de Catalunya (UPC) en Barcelona. Actualmente PRISMA pertenece al área de Docencia que apoya la gestión de los estudios de la UPC mediante el desarrollo de sistemas de información y proporciona atención y formación a sus usuarios. Algunas de las ventajas con las que cuenta PRISMA son:

- Facilita las tareas del día a día de la gestión académica de los centros de la UPC.
- Cuenta con un único archivo de datos.
- Apoya determinados procesos de gestión de planes de estudio, estudiantes y sus evaluaciones, proyectos de final de la carrera.
- Sistema de análisis de información directiva (SAID), orientado a apoyar los procesos de planificación y evaluación de la actividad académica (información para la toma de decisiones) (García, 2014)

Entre los inconvenientes que se identificaron de este sistema está que es un sistema de información en internet, por lo que su uso debe hacerse en línea. El sistema está desarrollado sobre software privativo, uno de los elementos que lo corrobora es que la base de datos que utiliza es Oracle, por tanto, en cuanto a soberanía tecnológica no se debe utilizar. Entre los procesos de gestión que realiza no tiene concebida la ubicación del estudiante que realiza la práctica profesional, como tampoco refleja que tutor lo atiende en este proceso.

Gestión de Prácticas Profesionales de la Universidad de Bío Bío

La Universidad del Bío-Bío es una institución de educación superior chilena, de carácter estatal, ubicada en la Región del Biobío, con sedes en Concepción y Chillán. El objetivo de este sistema es implementar un Sistema de Información de apoyo en la gestión del proceso de prácticas profesionales en la Universidad del Bío-Bío que se desarrollan en el entorno laboral. El sistema brinda, como principales beneficios: (Bio)

1. Ser una plataforma educativa para el desarrollo de la Práctica Profesional.
2. Mantener información centralizada sobre la actividad de prácticas.
3. Propiciar acercamiento con el entorno laboral.
4. Contribuir al desarrollo de competencias profesionales en los estudiantes.
5. Realizar un seguimiento académico de las prácticas.

Entre los requisitos funcionales con que cuenta están:

Gestión de Prácticas Laborales.

- Planificación

– programa de la práctica, periodos de práctica, escala evaluación, inscripción, aprobación.

- Ejecución y Control

– prácticas, descripción, participantes, objetivos, tareas, entregables.

- Evaluación

– roles de evaluación, evaluación del informe, evaluación de exposición, evaluación desempeño, evaluaciones parciales, evaluación final.

Este sistema no refleja el puesto de trabajo en el que debe estar ubicado el estudiante. Este sistema solo se toma como referencia para estudiar las funcionalidades que contiene, que en su mayoría responden al proceso de seguimiento y control de la Práctica Profesional.

Sistema de Gestión Universitaria

El Sistema de Gestión Universitaria se desarrolló en la Universidad de Ciencias Informáticas con el objetivo de centralizar la información académica de los estudiantes. Algunos de los servicios que brinda este sistema son: (UCI, 2015)

– Calendario docente.

– Horario docente.

– Balance de carga de las asignaturas.

– Planes de estudio de la carrera de cada estudiante.

- Trámites docentes.
- Registro de asignaturas.
- Resumen de evaluaciones.
- Permite realizar solicitudes de asignaturas no lectivas.
- Permite consultar las asignaturas cursadas con sus evaluaciones y calcula el índice académico.
- Genera varios reportes referentes a las evaluaciones de cada una de las asignaturas y calcula los porcientos de promoción y calidad.

El sistema no refleja la ubicación de los estudiantes en su PID en cuanto a departamento y proyecto al que pertenecen, así como los tutores que los atienden en el tema que desarrollan como parte de la asignatura. Al no manejarse esta información no permite hacer reportes que garanticen los elementos mencionados anteriormente. El sistema refleja las notas de los estudiantes por brigadas, lo que es un inconveniente para el cliente debido a que no todos los estudiantes de un grupo pertenecen a un mismo centro y esto dificulta el trabajo en el momento que se decide calcular los porcientos de promoción y calidad de los estudiantes.

GESTACAD

En la Universidad de Matanzas Camilo Cienfuegos se encuentra implementado el Sistema de Gestión Docente (GESTACAD). En la actualidad el sistema es capaz de realizar acciones y brindar algunos reportes, los cuales son fruto de los requisitos funcionales del sistema recogidos durante la investigación, que fueron surgiendo a medida que se conocía una necesidad de algún futuro usuario directo del sistema, ya sea estudiante o trabajador (incluyendo el administrador del sistema), como son (Delgado, 2013):

- Búsqueda de un alumno. [Brinda la ubicación según el horario docente].
- Listado de estudiantes por grupo.
- Reportes dinámicos de la información existente
- Reporte de notas por asignatura y grupo [examen final, extraordinario, especial, premio].
- Tabla con los resultados docentes de un grupo en un semestre.
- Reporte de los resultados académicos de un estudiante en toda su carrera. [Hoja de Rendimiento].
- Actas de exámenes de las diferentes asignaturas.
- Registro de características de un grupo de estudiantes.

El sistema no cubre las necesidades que tiene hoy el departamento de PP debido a que no refleja por quién está siendo tutorado el estudiante y tampoco muestra la ubicación de los mismos en la Práctica Profesional.

1.3.1 Análisis realizado a los sistemas informáticos

A partir de los criterios definidos en el epígrafe 1.3 y la revisión bibliográfica realizada a los sistemas anteriores, se pudo conocer el estado actual de las aplicaciones informáticas que permiten la gestión académica relacionadas a la PP. Los resultados de este estudio se resumen en la Tabla 1.

Tabla 1 Sistemas similares. Elaboración propia

Criterios de búsqueda	**Sistemas estudiados**			
	PRISMA	**Entorno virtual de Prácticas Profesionales**	**Sistema de gestión universitaria**	**GESTACAD**
Relación tutor-estudiante	NO	SI	NO	NO
Registro de evaluación de los estudiantes	SI	SI	SI	SI
Centralización de la información	SI	SI	SI	SI
Sistema basado en tecnología web	SI	SI	SI	SI
Ubicación de la Práctica Profesional	NO	NO	NO	NO

El estudio realizado permitió definir un conjunto de ventajas y limitaciones, mostradas en la Tabla 2, que serán tenidos en cuenta para el diseño de la propuesta de solución.

Tabla 2 Análisis de sistemas. Elaboración propia.

Sistemas	**Ventajas**	**Limitaciones**
PRISMA	Permite la gestión de estudiantes, de sus evaluaciones y de los proyectos de fin de carrera.	No refleja la ubicación de los estudiantes en la Práctica Profesional. Es un sistema privativo y su uso debe hacerse en línea.

Gestión de Prácticas Profesionales de la Universidad de Bío Bío	Contiene entre sus procesos distintas formas de evaluación como: roles de evaluación, evaluación del informe y evaluación de la exposición.	No refleja la ubicación de los estudiantes en la Práctica Profesional.
Sistema de Gestión Universitaria	El sistema de permisos a los usuarios está orientado a roles. Genera varios reportes referentes a las evaluaciones de cada una de las asignaturas y calcula los porcientos de promoción y calidad.	No presenta la relación entre los estudiantes y sus tutores y no refleja la ubicación de la PP.
GESTCAD	Se le da la posibilidad al usuario de seleccionar los campos de datos que desea obtener en el reporte así como el título de este y las condiciones que debe cumplir la información a mostrar.	No presenta la relación entre los estudiantes y sus tutores y no refleja la ubicación de la PP.

A partir del estudio realizado anteriormente y las necesidades del cliente, se opta por el desarrollo de un módulo que permita la gestión de los procesos del departamento de PP del centro, además de que este deberá ser integrado al Sistema para la gestión de los procesos administrativos de CEIGE. Las ventajas que se presentaron anteriormente constituyen una fuente inicial para la definición de los requisitos.

1.4 Metodología de desarrollo de software

Una metodología de desarrollo de software “es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo. Es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema informático” (INTECO, 2009).

Las metodologías de desarrollo de software “son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto, pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales” (Méndez, 2010).

Metodología de desarrollo de software Variación de AUP para la UCI

En el desarrollo de la aplicación se utilizará la metodología de desarrollo de software Variación de AUP para la UCI. Esta una variante realizada por la Universidad de las Ciencias Informáticas a la metodología ágil AUP y está definida por el centro de estudios como documento rector de la actividad productiva.

Fases de Variación de AUP para la UCI

La metodología Variación de AUP para la UCI está compuesta por tres fases, (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad, las cuales contienen las características de las cuatro fases (Inicio, Elaboración, Construcción y Transición) propuestas en AUP. Las características de las fases de la metodología de la universidad son (Sánchez, 2014):

Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización del cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o entregado al cliente junto con la documentación. Además, en esta transición se capacita a los usuarios finales sobre la utilización de la aplicación.

Cierre: En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Disciplinas de Variación de AUP para la UCI

AUP propone 7 disciplinas: Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno; para el ciclo de vida de los proyectos de la UCI se decidió contar con 8 disciplinas, pero a un nivel más atómico que el definido en AUP, ellas son: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas interna, Pruebas de liberación, Pruebas de Aceptación y Despliegue. (Sánchez, 2014)

El desarrollo del sistema de gestión de los procesos del Departamento de Práctica Profesional de CEIGE, se centrará en la fase de Ejecución de la metodología. Además, se generarán los productos de trabajo

definidos para las disciplinas modelado del negocio, requisitos, análisis y diseño, implementación, pruebas internas, pruebas de liberación y de aceptación.

Escenario de la metodología a utilizar

Se decide utilizar el escenario número tres de la metodología debido a que existen roles que intervienen y que pudieran llamarse en otro momento actores del sistema. Existen entradas y salidas, actividades que se ejecutan y se puede realizar la definición de conceptos y sus relaciones a partir del negocio. Por ende, se evidencia la existencia de procesos de negocio y por ello se utiliza este escenario.

Figura 1 Escenario 3 de la metodología AUP-UCI (Sánchez, 2014)

1.5 Herramientas, lenguajes y tecnologías

Lenguaje de modelado de software

El Lenguaje Unificado de Modelado 2.0 (UML, Unified Modeling Language) es un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y pretende unificar diferentes técnicas de modelado.

UML incluye conceptos semánticos, notación, y principios generales de un sistema además de que permite representar decisiones de implementación y entorno. Contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite dividir grandes sistemas en piezas de trabajo más simples. (Ivar, y otros, 2002)

Herramientas para el modelado del sistema

Visual Paradigm 8.0 para UML es una herramienta CASE que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Permite integrarse con otras aplicaciones, como

herramientas ofimáticas, lo cual aumenta la productividad. Brinda la posibilidad de generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software así como obtener diversos informes a partir de la información introducida en la herramienta. (Domingo, y otros, 2012)

Esta herramienta se selecciona para el modelado del sistema por la importancia del uso del software libre en Cuba, la existencia de una licencia otorgada a la universidad para su utilización y las características que posee.

Lenguajes de programación

HTML5 (HyperText Markup Language, Lenguaje de Marcado de Hipertextos) "define una estructura básica y un código HTML para la definición de contenido de una página web, como texto, imágenes, entre otros" y se basa en la referenciación por hipertextos o enlaces entre páginas (Pérez, 2010).

HTML5 no es una nueva versión del antiguo lenguaje de etiquetas, ni siquiera una mejora de lenguaje, sino un nuevo concepto para la construcción de sitios web y aplicaciones donde se combinan las tecnologías HTML, para la estructura y organización de las páginas; CSS, para presentar el contenido a través de diferentes estilos de diseño y Javascript para proveer determinadas funcionalidades útiles para la web (Gauchat, 2012).

CSS3 (Cascading Style Sheets, Hojas de Estilo en Cascada) es un "lenguaje para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. CSS permite separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas". (…) "Mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes" (Pérez, 2010).

Se utiliza para definir el aspecto de todos los contenidos, como: el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista (Pérez, 2010). Esta forma de descripción de estilos ofrece a los desarrolladores el control sobre el estilo y el formato de sus documentos. Funciona a base de reglas para las declaraciones sobre el estilo de uno o más elementos.

PHP 5.5 (HyperText Preprocessor, Preprocesador de Hipertexto) es un lenguaje de programación de uso general de código abierto y del lado del servidor, resulta útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos, permite la generación dinámica de contenidos en un servidor web y puede ser embebido en páginas HTML. Contiene técnicas de programación orientada a objetos y posee bibliotecas con funciones predefinidas. Es un lenguaje multiplataforma con capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, tiene capacidad de

expandir su potencial utilizando la mayoría de los módulos, llamados extensiones (L. Welling; Laura Thomson, 2010).

Javascript es un lenguaje de programación web en el lado del cliente que se utiliza principalmente para crear páginas web dinámicas. Es un lenguaje interpretado, por lo que no es necesario compilar los programas para ejecutarlos, es decir, los programas escritos con Javascript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Pérez, 2010).

"Está diseñado para el desarrollo de aplicaciones cliente-servidor a través de Internet" (Maza, 2012). Permite incorporar funciones para el control de la información. Evita cargas en las transferencias de datos entre el cliente y el servidor ya que el navegador de usuario es el responsable de asumir toda la carga de procesamiento (Gauchat, 2012).

Herramientas y tecnologías

PhpStorm 9.0 es un Entorno de Desarrollo Integrado (IDE) compatible con PHP desde su versión 5.3 hasta la 7.0, proporciona prevención de errores, mejor autocompletado y refactorización de código, depuración de configuración cero, y un editor HTML, CSS, JavaScript. El IDE ofrece completamiento inteligente de código, resaltado de sintaxis, configuración extendida del formato de código, comprobación de errores, plegado de código, soporta las mezclas de idiomas, entre otros (JetBrains, 2001). Esta herramienta se seleccionó para el desarrollo de la propuesta de solución ya que responde a la política de soberanía tecnológica de la universidad.

Marco de trabajo

Symfony 2 es un marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web. Esta plataforma de desarrollo garantiza la separación de responsabilidades, es decir, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación. Esto permite mantener el código organizado y que la aplicación evolucione, evitando mezclar llamadas a la base de datos, etiquetas HTML y el código de lógica del negocio en un mismo archivo (Pacheco, 2011).

"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" (Bastidas, 2012).

Symfony 2 utiliza el lenguaje de programación PHP. Posee una arquitectura interna completamente desacoplada, lo que permite reemplazar o eliminar aquellas partes que no se ajustan en un proyecto de desarrollo de aplicaciones web (Pérez, 2014). Emplea el patrón de diseño MVC (Modelo – Vista – Controlador) para separar las distintas partes que forman un sistema (Yunaysy Ortiz Batista, 2011)

Servidor de aplicaciones web

Apache 2.0 es un servidor web de tecnología de código abierto para uso comercial. Opera en diferentes sistemas operativos y es configurable. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor ya que puede configurarse para que ejecute un determinado script (archivo de órdenes de procesamiento) cuando ocurra un error en concreto. Posee una alta configurabilidad en la creación y gestión de los registros, posibilitando la creación de ficheros de log, de este modo puedes tener un mayor control sobre lo que sucede en el servidor (Saco, 2014).

Servidor de base de datos

PostgreSQL 9.3 es un sistema de gestión de bases de datos objeto-relacional, multiusuario, centralizado y de propósito general. Es considerado como el sistema gestor de bases de datos de código abierto más avanzado del mundo. Permite el control de concurrencia multiversión, gestión de transacciones y puntos de salvas (Alvarez, y otros, 2010).

Las herramientas, lenguajes y tecnologías mencionadas anteriormente se seleccionaron para el desarrollo de la solución ya que responden a la política de soberanía tecnológica de la universidad.

1.6 Patrón Arquitectónico Modelo Vista Controlador

Los patrones arquitectónicos son patrones de software que ofrecen soluciones a problemas de arquitectura de software, adaptabilidad a requerimientos cambiantes, rendimiento, modularidad y acoplamiento. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes de un sistema. La solución que plantean es la creación de patrones de llamadas entre objetos y el empaquetado de funcionalidades (Venete, 2011). El patrón MVC utilizado para definir la arquitectura del sistema separa los datos de la aplicación (modelo), la interfaz de usuario (vista), y la lógica de negocio (controlador) en tres componentes distintos.

Modelo: Es la capa donde se trabaja con los datos, por tanto, contiene mecanismos para acceder a la información y también para actualizar su estado. La información se tiene almacenada en la base de datos, por lo que en los modelos se tienen todas las funciones que accederán a las tablas y harán los correspondientes *seleccionar, actualizar, insertar*, entre otros. (Alvares, 2014)

Vista: Las vistas, como su nombre hace entender, contienen el código de la aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que permitirá reenderizar los estados de la aplicación en HTML. En las vistas nada más se tienen los códigos HTML y PHP. (Alvares, 2014)

Controlador: Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información. Es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de la aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo. (Alvares, 2014)

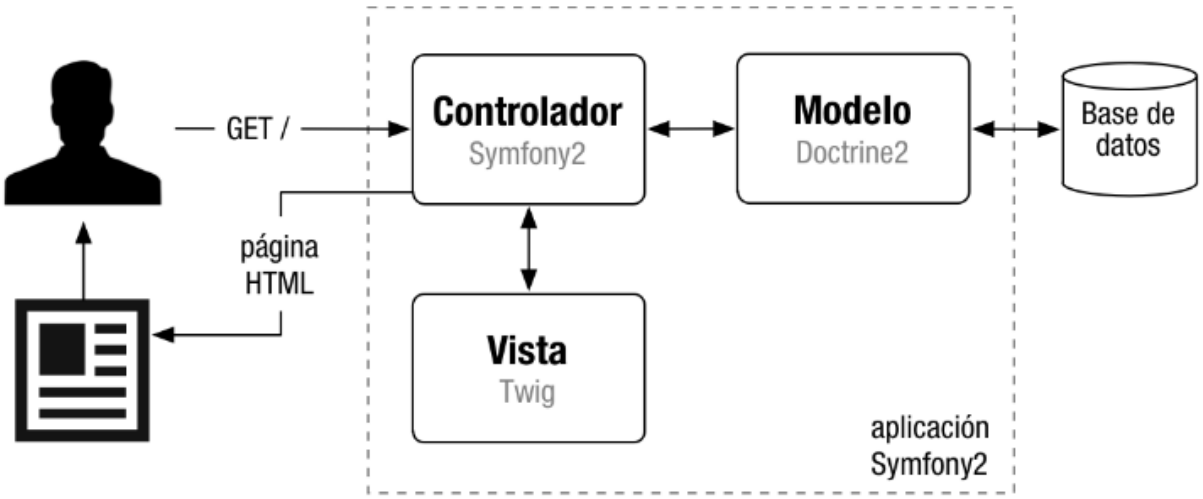


Figura 2: Arquitectura Modelo Vista Controlador. (Eguiluz, 2013)

1.7 Métricas para la validación del diseño

Las métricas de diseño de software permiten medir de forma cuantitativa la calidad de los atributos internos del software. Esto proporciona una vía para evaluar la calidad durante el desarrollo del sistema. A nivel de componentes se concentran en las características internas de los componentes del software con medidas que ayudan a juzgar la calidad de un diseño a nivel de componente. En su libro acerca de métricas, Lorenz y Kidd dividen las métricas basadas en clase en cuatro amplias categorías; cada una tiene una relación en el diseño en el nivel de componentes: tamaño, herencia, internos y externos. Las métricas orientadas a tamaño para una clase de diseño se enfocan en conteos de atributos y operaciones para una clase individual y en valores promedio para el sistema como un todo. Las métricas basadas en herencia se enfocan en la forma en la que las operaciones se reutilizan a lo largo de la jerarquía de clases. (Pressman, 2010). En este caso se aplicarán Tamaño Operacional de Clases y la Relación entre Clases. La métrica Tamaño Operacional de Clases (TOC) está dado por el número de métodos u operaciones (de instancia privada y heredada) que están encapsulados dentro o por una clase. Evalúa los siguientes atributos de calidad mostrados en la Tabla 3.

Tabla 3: Atributos de calidad que evalúa TOC

Atributo	Modo en que lo afecta
Responsabilidad.	Un aumento del TOC provoca una mayor responsabilidad asignada a la clase.
Complejidad de implementación.	Un aumento del TOC provoca un aumento de la complejidad de implementación de la clase.
Reutilización.	Aumento del TOC provoca disminución del grado de reutilización de la clase.

Para evaluar los atributos se emplean los siguientes criterios y categorías de evaluación mostrados en la Tabla 4.

Tabla 4: Criterios y categorías de evaluación de TOC

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

La métrica Relación entre clases(RC) se define con el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad.

Tabla 5: Atributos de calidad evaluados por la métrica RC

Atributo	Modo en que lo afecta
Acoplamiento	Un aumento de las relaciones entre clases provoca aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento de las relaciones entre clases provoca aumento de la complejidad de mantenimiento de la clase.
Reutilización	Un aumento de las relaciones entre clases provoca disminución del grado de reutilización de la clase.

Cantidad de pruebas	Un aumento de las relaciones entre clases provoca aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Para la evaluación de dichos atributos de calidad, se definen los siguientes criterios y categorías de evaluación.

Tabla 6: Criterios y categorías de evaluación de RC

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad Mantenimiento	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

1.8 Pruebas de software

Las pruebas de software se pueden definir como el proceso de verificar el comportamiento de una programa ante un conjunto de casos de prueba, contra el comportamiento esperado (Pressman, 2005). Por tanto, las pruebas de software son un conjunto de actividades dentro del desarrollo de software que podrán ser ejecutadas en cualquier momento en el proceso de desarrollo. Estas se realizan con el fin de detectar posibles errores de implementación, calidad o usabilidad de un programa.

Los niveles de pruebas que se pueden aplicar a un software son descritos a continuación:

- ✓ **Pruebas de unitarias:** son usadas para comprobar el correcto funcionamiento de un módulo de código. Con esta prueba se comprueba que cada módulo funcione correctamente por separado.

- ✓ **Pruebas de integración:** en estas pruebas se combinan módulos individuales de forma tal que seas probados como un grupo. Se usan para detectar errores de interfaces y relaciones entre componentes.
- ✓ **Pruebas de sistema:** con estas pruebas se trata de asegurar la correcta navegación dentro del sistema, ingreso, procesamiento y recuperación de datos. Son utilizadas para conocer las fallas funcionales del sistema y el cumplimiento del desarrollo de los requisitos definidos con el cliente.
- ✓ **Pruebas de aceptación:** son desarrolladas y ejecutadas con el cliente para determinar si el producto satisface sus criterios o requisitos. Estas son las únicas pruebas que se realizan directamente con el cliente, a diferencia de las restantes.

1.9 Conclusiones parciales del capítulo

El estudio realizado a los diferentes sistemas proporcionó un conjunto de funcionalidades a tener en cuenta en el desarrollo del módulo informático para el control y seguimiento de los estudiantes en las asignaturas de PID. Este módulo se desarrollará con tecnologías y herramientas de código abierto garantizando la soberanía tecnológica y se utilizará la metodología Variación de AUP para la UCI para guiar el proceso de desarrollo. Además, se estudiaron las métricas de validación del diseño y los diferentes métodos de pruebas que serán aplicadas en la validación de la solución.

Capítulo 2: Propuesta de solución

2.1 Introducción

El presente capítulo contiene la información relacionada con el análisis y el diseño de la propuesta de solución. Se describen los procedimientos llevados a cabo para la construcción de la aplicación, utilizando la metodología de desarrollo de software seleccionada para el desarrollo de la investigación. Se evidencian las necesidades del cliente, los requisitos que el sistema debe cumplir y se define el comportamiento del mismo, así como las restricciones del diseño, concebidas para la construcción de la aplicación.

2.2 Proceso de negocio

En el departamento de PP de CEIGE se reciben a los estudiantes una vez que inician en el ciclo profesional, recibiendo las asignaturas de PID III hasta la PID VII. Se les realiza una distribución por las diferentes áreas, asignándoles el tema de investigación a desarrollar y los tutores. Además, los estudiantes son atendidos por un profesor que, de conjunto con los tutores, elabora los planes de formación y los evalúan. Un elemento a tener en cuenta es la ubicación del estudiante en el puesto de trabajo, ya sea en el propio proyecto al que está vinculado o en un laboratorio de producción del centro. A partir de lo antes mencionado el Jefe de departamento mantiene el control y seguimiento de los estudiantes en las asignaturas de PID.

2.2.1 Modelo conceptual

El Modelo conceptual es una representación visual de los objetos y conceptos (o partes esenciales) que se requieren en la modelación de un problema determinado y las relaciones que subsisten entre ellos (Guerra, 2016). Los principales conceptos y relaciones de la solución se muestran en la Figura 3. El estudiante es el actor principal de la Práctica Profesional. Cada uno de ellos pertenece a un proyecto, puede o no ser de la propia facultad, en el que se le asigna un tema de investigación, (diferente dependiendo del año académico, para los estudiantes de quinto año se corresponde con el tema de tesis). Los trabajadores que se encuentran relacionados directamente con los estudiantes pueden ser profesores, especialistas o recién graduados en adiestramiento (RGA), en el que uno o más de ellos será el tutor de uno o varios estudiantes los que a su vez tendrán asignados puestos de trabajo para el desarrollo de sus actividades.

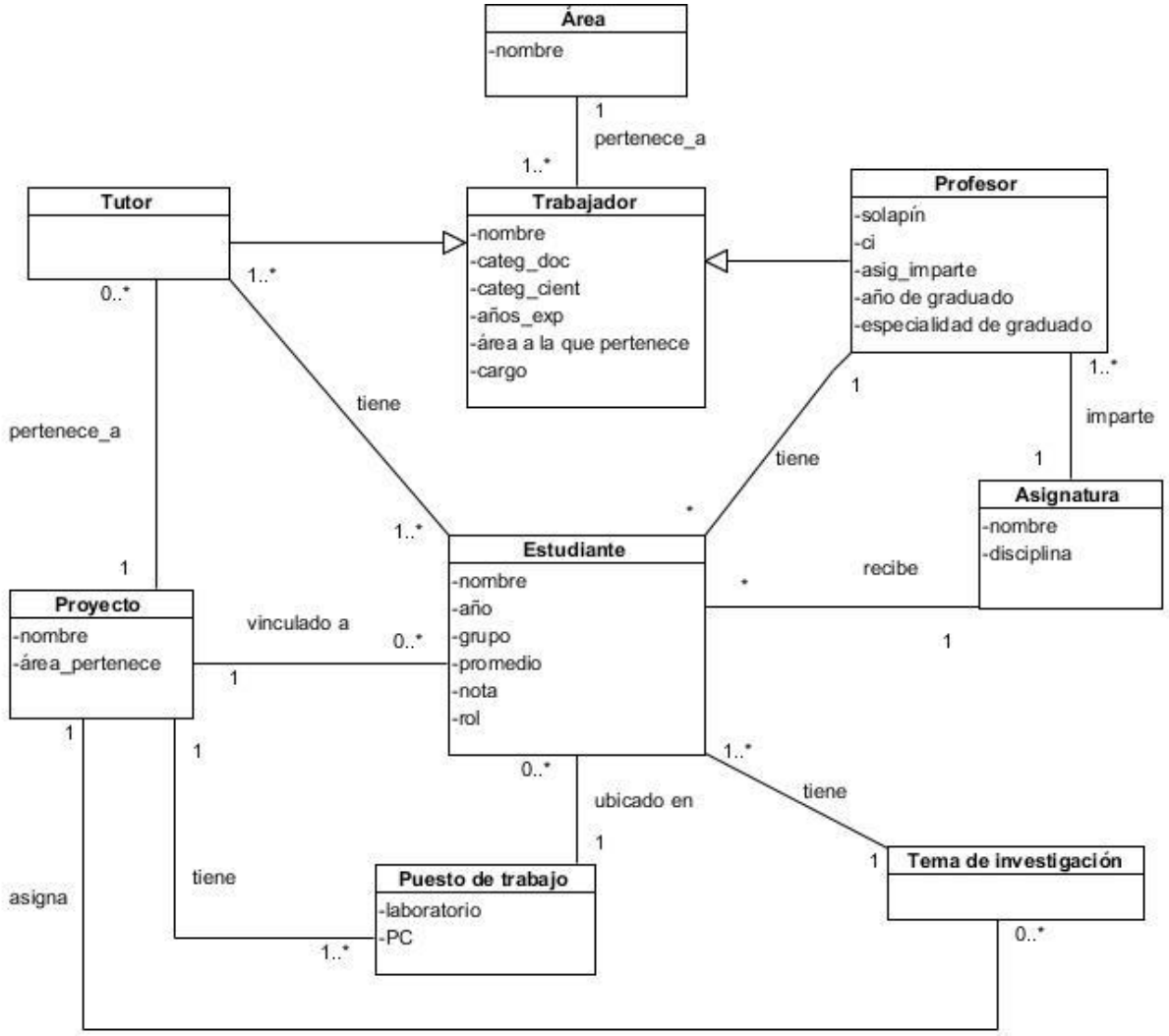


Figura 3: Modelo conceptual. Elaboración propia

El modelo conceptual elaborado se utiliza como base para la definición de las funcionalidades y restricciones que debe cumplir el sistema. A partir de este modelo y el estudio de sistemas realizado en el capítulo anterior se definen los requisitos que deben cumplir con las necesidades del departamento de PP y que serán abordados en la disciplina Requisitos según la metodología de desarrollo.

2.3 Requisitos

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir y comprende la administración de los requisitos funcionales y no funcionales del producto (Sánchez, 2014). Permite mejorar el entendimiento, análisis, validación, especificación y gestión de las necesidades, funcionalidades y restricciones que debe cumplir un sistema. A continuación, se presentan las actividades desarrolladas en esta disciplina, así como los productos de trabajo elaborados.

2.3.1 Fuentes para la obtención de requisitos y técnicas de identificación de requisitos

Las fuentes de obtención de requisitos a consultar para la definición de los requisitos del sistema se relacionan a continuación:

- ✓ Soluciones informáticas existentes con propósitos similares a los deseados.
- ✓ Modelo conceptual.
- ✓ Jefe de Departamento de PP en la facultad 3.

2.3.2 Técnicas de obtención de requisitos

Las técnicas de obtención de requisitos de software permiten identificar las necesidades de negocio de clientes y usuarios. Son mecanismos que se utilizan para recolectar la información necesaria en la obtención de los requisitos funcionales y no funcionales de una aplicación (Pressman, 2010). Existen diferentes técnicas de obtención de requisitos dentro de las que se encuentran: entrevistas, encuestas y tormentas de ideas (Sommerville, 2011).

Entrevista: es una técnica ampliamente utilizada para la obtención de requisitos. Se basa en la acción de desarrollar una conversación con los clientes con el objetivo de obtener una comprensión general de lo que realizan. Las entrevistas pueden ser de dos tipos: cerradas, donde los entrevistados responden a un conjunto predeterminado de preguntas y abiertas, donde no hay un programa de preguntas predefinido (Sommerville, 2011).

Se aplicaron entrevistas al Jefe de Departamento de PP y a tres profesoras del colectivo, que evidenciaron las principales insuficiencias que existen en el departamento. Entre estas, las inconsistencias que pueden existir cuando se actualiza la información referente a los estudiantes, la búsqueda de datos para la realización de informes se hace más lenta y se pudiera cometer algún error. Se dificulta el trabajo a la hora de buscar datos referentes a algún estudiante específico, así como poder determinar la promoción y la calidad de estos en los diferentes años. Se concluyó la necesidad de una mejor organización de la información referente a la PID y que esté disponible en un sistema al que se pueda acceder desde diferentes lugares de la universidad.

Análisis documental: es una de las técnicas más comunes para el levantamiento de requisitos, varios tipos de documentación, como manuales y reportes, pueden proporcionar al analista información valiosa con respecto a las organizaciones y a sus operaciones. La documentación difícilmente refleja la forma en que realmente se desarrollan las actividades, o donde se encuentra el poder de la toma de decisiones. Sin embargo, puede ser de gran importancia para introducir al analista al dominio de operación y el vocabulario que utiliza (Guerra, 2016).

Se estudiaron documentos que reflejan el desarrollo de las actividades de la PP en diferentes lugares, que permitió adquirir conocimiento sobre las acciones que se realizan.

Tormenta de ideas: consiste en reunir a los diferentes encargados de los procesos de negocio para que en una reunión informal generen ideas sobre los diferentes problemas que cada uno en su área posea. Como resultado de estas sesiones se obtiene la mayor cantidad de soluciones posibles desde el punto de vista de cada uno de los involucrados en los procesos de la organización y puede generar salidas que beneficien a la organización (Martínez Guerrero, 2010).

Se realizaron varias reuniones y encuentros con el colectivo de profesores del departamento donde se expusieron las principales deficiencias y carencias que existen en estos momentos. Como resultado se obtuvieron un conjunto de ideas que permitieron definir un listado inicial de los requisitos funcionales.

Construcción de prototipos: los prototipos suelen consistir en vistas reducidas de la aplicación a desarrollar. Permite que el usuario cuente con una visión general de lo que desea y que puedan mejorar las especificaciones de los requerimientos (Sommerville, 2011).

2.3.3 Requisitos funcionales

Los requisitos funcionales de un software describen lo que este debe hacer. Dependen del tipo de software que se desarrolle, de los posibles usuarios del software y de las necesidades que posea el cliente. Son las declaraciones de los servicios que el sistema debe proporcionar, de la manera en que debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones determinadas (Sommerville, 2011). La Tabla 7 muestra los requisitos funcionales definidos para el desarrollo de la aplicación que se propone y se han especificado a partir de las características y funcionalidades definidas con el cliente.

Tabla 7: Requisitos funcionales

Agrupación de requisitos	RF	Nombre	Complejidad
Gestionar estudiante	RF 1	Insertar estudiante	Baja
	RF 2	Modificar estudiante	Baja
	RF 3	Listar estudiante	Baja
	RF 4	Desactivar estudiante	Alta
Gestionar tutor	RF 5	Insertar tutor	Baja
	RF 6	Modificar tutor	Baja
	RF 7	Listar tutor	Media
	RF 8	Eliminar tutor	Baja

Gestionar asignatura	RF 9	Insertar asignatura	Baja
	RF 10	Modificar asignatura	Baja
	RF 11	Listar asignatura	Baja
	RF 12	Eliminar asignatura	Baja
Gestionar proyecto	RF 13	Insertar proyecto	Baja
	RF 14	Modificar proyecto	Baja
	RF 15	Listar proyecto	Baja
	RF 16	Eliminar proyecto	Baja
Gestionar profesor	RF 17	Insertar profesor	Baja
	RF 18	Modificar profesor	Baja
	RF 19	Listar profesor	Media
	RF 20	Eliminar profesor	Baja
Gestionar área	RF 21	Insertar área	Baja
	RF 22	Modificar área	Baja
	RF 23	Listar área	Baja
	RF 24	Eliminar área	Baja
Gestionar tema de tesis	RF 25	Insertar tema de tesis	Baja
	RF 26	Modificar tema de tesis	Baja
	RF 27	Listar tema de tesis	Baja
	RF 28	Eliminar tema de tesis	Baja
Promoción y Calidad	RF 29	Calcular promoción de un año	Alta
	RF 30	Calcular calidad de un año	Alta
Reporte tutor-estudiante	RF 31	Exportar listado de estudiantes por tutor	Media
Reporte estudiante-profesor	RF 32	Exportar el listado de estudiante por profesor	Alta
Reporte estudiante-puesto de trabajo	RF 33	Exportar listado de estudiante por puesto de trabajo	Alta
Reporte de estudiantes por área	RF 34	Exportar listado de estudiante por área	Alta
Reporte de estudiantes por proyecto	RF 35	Exportar listado de estudiante por proyecto	Alta
	RF 36	Insertar Usuario	Baja
	RF 37	Editar Usuario	Baja
	RF 38	Eliminar Usuario	Baja

Gestionar usuario	RF 39	Listar Usuarios	Baja
	RF 40	Asignar rol a Usuario	Alta
Autenticar usuario	RF 41	Autenticar usuario	Alta

Descripción de requisito Modificar estudiante

La descripción de requisitos describe con mayor detalle, lo que este debe proporcionar en el sistema. Contiene los pasos o las actividades que se deberán realizar para darle cumplimiento al requisito, así como las restricciones a las que está sujeto y las posibles respuestas del sistema. Es una herramienta ampliamente utilizada en el análisis y diseño de un sistema informático (Sommerville, 2011). Seguidamente se muestra la descripción del requisito funcional Modificar estudiante.

Tabla 8: Descripción de requisito funcional Modificar estudiante

Precondiciones	Debe existir al menos un estudiante en el sistema.		
Flujo de eventos			
Flujo básico Modificar estudiante			
1.	El profesor selecciona la opción Listado de estudiantes del menú Gestionar estudiante en la PP		
2.	El sistema muestra un listado con los años académicos de los estudiantes 3ro,4to,5to		
3.	El profesor selecciona el año del estudiante que desea modificar		
4.	El sistema muestra un listado con los estudiantes del año seleccionado		
5.	El profesor selecciona el estudiante a modificar		
6.	El sistema muestra el formulario del estudiante seleccionado		
7.	El profesor modifica los datos que desea		
8.	Selecciona la opción Aceptar		
9.	El sistema valida los datos introducidos		
Pos-condiciones			
1.	Se ha modificado un estudiante		
Flujos alternativos			
Flujo alternativo 9 a. Datos del formulario incorrectos			
1.	El sistema muestra un notificación de error		
2.	El profesor introduce nuevamente los datos		
3.	Volver al paso 8 del flujo básico		
Pos-condiciones			
•	N/A		
Validaciones			
•	Se validan los datos según lo establecido en el Modelo conceptual		
Conceptos	**Estudiante**	Visibles en la interfaz: Nombre Año Grupo Rol Tema de Investigación/Tema de tesis Puesto de trabajo	

		Profesor Área a la que pertenece Asignatura Tutor Promedio Estado Utilizados internamente: Id	
Requisitos especiales	N/A.		
Asuntos pendientes	N/A		

Prototipo de interfaz de usuario del requisito funcional Modificar estudiante

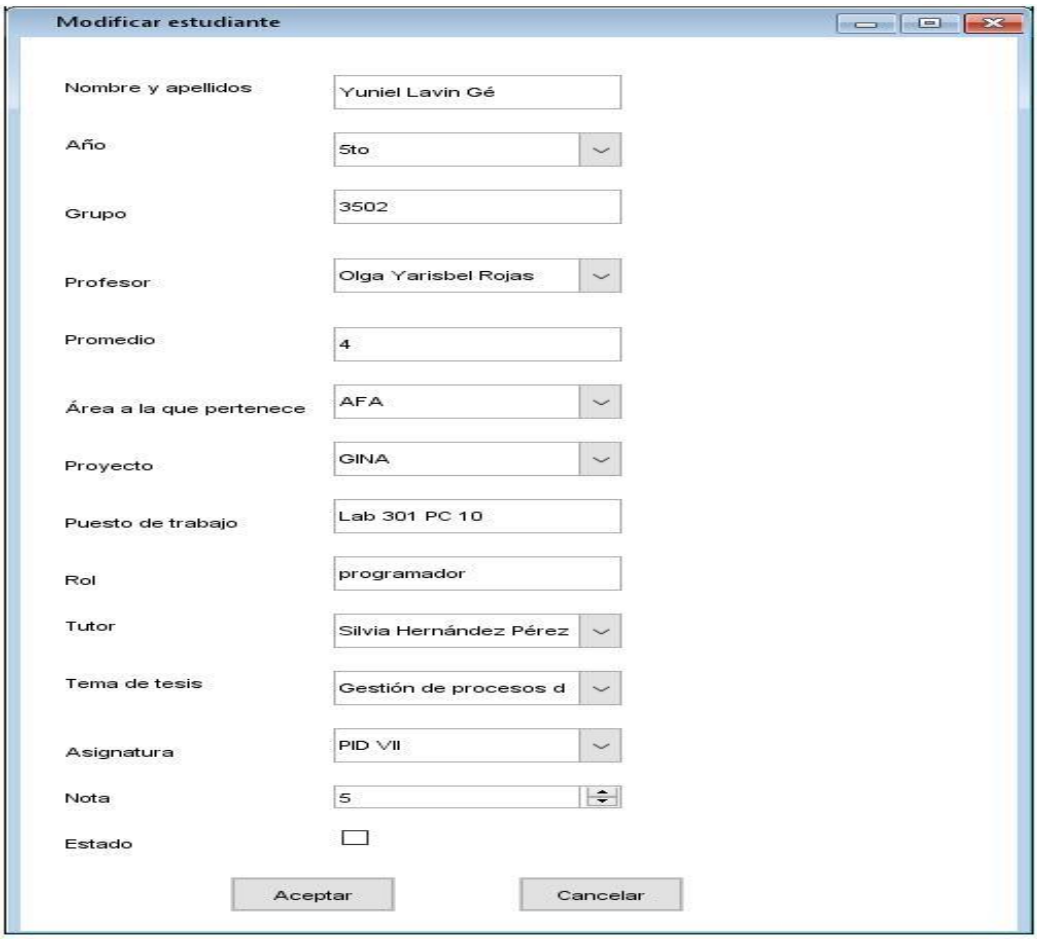


Figura 4: Prototipo de interfaz de usuario. Requisito funcional Modificar estudiante

2.3.4 Requisitos no funcionales

Los requisitos no funcionales son restricciones de las funciones o servicios ofrecidos por el sistema. No se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este. Definen restricciones sobre el mismo y el proceso de desarrollo de software (Sommerville, 2011). Los requisitos no funcionales definidos son:

RNF. 1 Usabilidad

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

RNF. 2 Confiabilidad

RNF. 2.1 Ante el fallo de una funcionalidad del sistema, el resto de las funcionalidades que no dependen de esta deberán seguir funcionando.

RNF. 2.2 El sistema concederá acceso a cada usuario autenticado solo a las funcionalidades que le estén permitidas de acuerdo a la configuración del sistema.

RNF. 3 Mantenibilidad

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

RNF. 4 Portabilidad

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

RNF.5 Hardware

RNF. 5.1 La estación de trabajo debe contar como mínimo con un procesador Pentium IV, RAM de 512 MB, disco duro de 500 GB y una tarjeta de red.

RNF.6 Software

RNF. 6.1 Se recomienda que estaciones de trabajo posean un navegador web Mozilla Firefox 34.0 o una versión superior.

RNF. 6.2 El servidor de aplicaciones web es Apache 2.0.

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

2.3.5 Validación de requisitos

El objetivo principal de la validación requisitos es comprobar que los requisitos funcionales y no funcionales definidos para el sistema se corresponden con las necesidades del negocio del cliente y usuarios, obteniendo su aprobación y permitiendo generar una línea base de los requisitos en la actividad (MADEJA, 2015). Existen diferentes técnicas para la validación de requisitos de software como:

- **Técnica de prototipado**: el prototipado de interfaz de usuario es una técnica de representación aproximada de la interfaz de usuario de un sistema software que permite a clientes y usuarios entender fácilmente la propuesta de solución para resolver sus problemas de negocio (MADEJA, 2015).
- **Revisiones Técnicas Formales:** es el filtro más efectivo desde el punto de vista de garantía de calidad. El objetivo fundamental de estas revisiones es encontrar errores durante el proceso de desarrollo de software. Se deben realizar desde el principio del proyecto para evitar la propagación de los errores a las etapas siguientes del proceso de software (Pressman, 2010).

La validación de los requisitos de software se desarrolló a través de la técnica de prototipado. En este proceso se diseñaron los prototipos de interfaces y se presentaron a diferentes profesores del Departamento de PP de CEIGE, incluyendo al jefe de departamento para su valoración. A lo largo de la realización del proyecto se realizaron pruebas funcionales del sistema, tanto por parte del cliente como del desarrollador, para detectar los posibles errores y evitar su propagación. Esto permitió que el cliente tuviera una noción de la propuesta de solución y se pudieran hacer las correcciones necesarias antes del diseño e implementación de las funcionalidades del sistema.

2.3.6 Administración de requisitos

La administración de requisitos es el proceso de comprender y controlar los cambios en los requisitos del sistema. Incluye las actividades de planificación de la administración de requisitos, en la que se identifican los requisitos cambiantes y los no cambiantes, se establecen políticas y procedimientos para el seguimiento de la relación entre los requisitos y de estos con el sistema. Además abarca la gestión de cambio en la que se analiza los cambios propuestos en los requisitos y se evalúa el impacto que provoca (Sommerville, 2011).

En la propuesta de solución se realizaron las matrices de trazabilidad para la administración de los requisitos del sistema desarrollado, permitiendo seguir la evolución de los requisitos durante el ciclo de vida del proyecto. Para ello se definieron los siguientes elementos de seguimiento para la trazabilidad de requisitos:

- Modelo conceptual
- Requisitos
- Modelo de datos
- Diagrama de clases del diseño
- Diagrama de componentes

La matriz de trazabilidad de los requisitos funcionales del sistema realizada fue Modelo Conceptual-Requisitos, la cual se muestra a continuación:

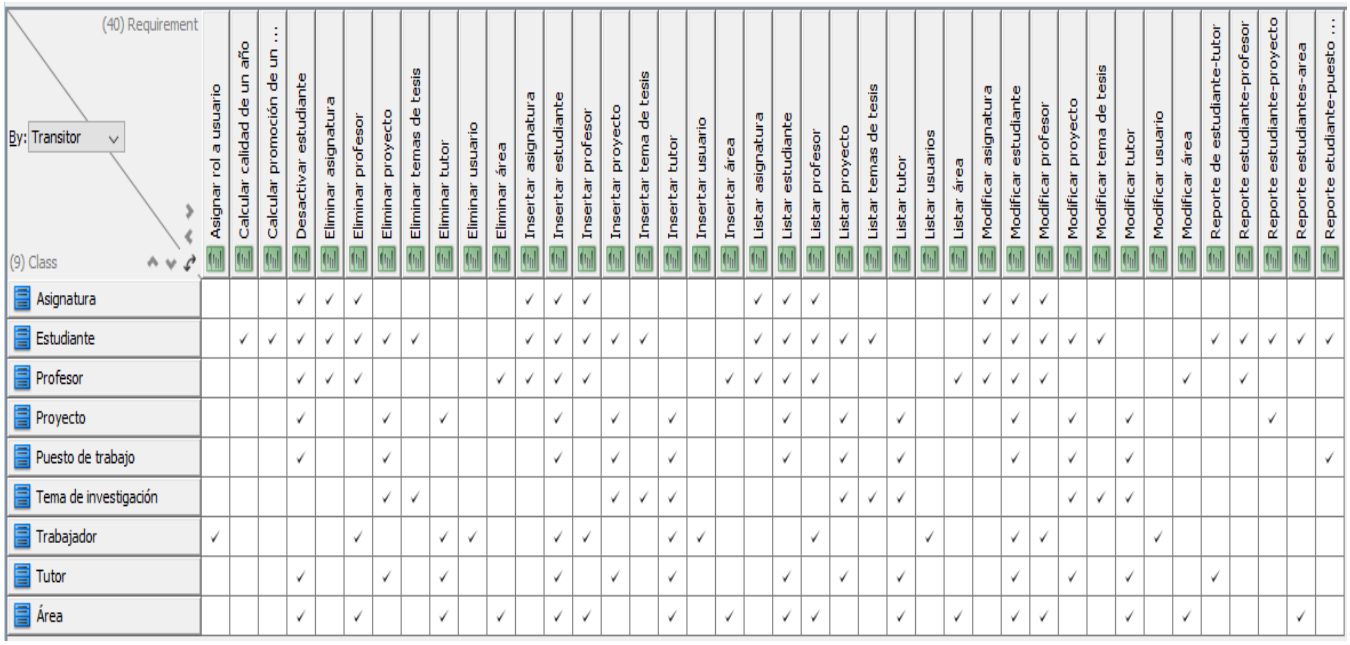


Figura 5: Matriz de trazabilidad modelo conceptual-requisitos

2.4 Modelo de diseño

El diseño es la forma exacta en la que un requisito del cliente se puede convertir en un sistema o producto de software terminado. Crea una representación o modelo del software de forma detallada sobre la estructura de datos, la arquitectura, las interfaces y los componentes del software que son necesarios para implementar el sistema e influye directamente en la calidad del producto informático (Pressman, 2010).

2.4.1 Diseño arquitectónico

La arquitectura de software de un sistema se refiere a sus diferentes componentes, las interacciones entre ellos y la configuración del mismo. El diseño arquitectónico es una actividad que permite estructurar de forma general al sistema y su objetivo es tener una visión clara del software a construir. En él se identifican los subsistemas en los que se divide una aplicación y se establece un marco de control y comunicación entre ellos (Mora, 2011). En el diseño arquitectónico del sistema propuesto se utiliza el patrón Modelo – Vista – Controlador (MVC).

Seguidamente se representa la separación de las clases de la aplicación según el marco de trabajo Symfony 2. En ella se visualiza la utilización del patrón MVC ya que se encuentran separadas las clases controladoras, las clases del modelo y las clases que forman parte de la vista en diferentes carpetas, tal como indica el patrón arquitectónico empleado.

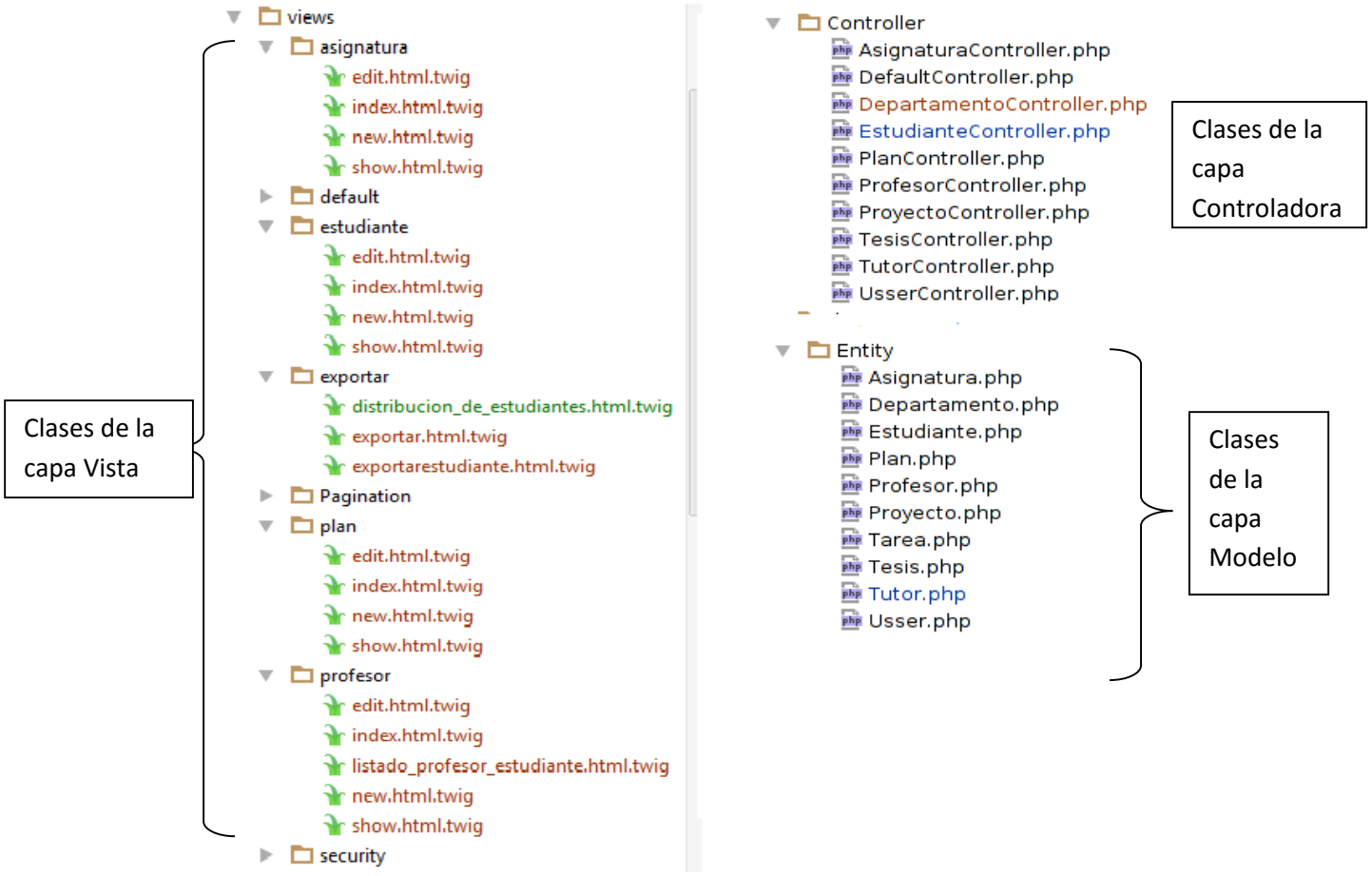


Figura 6: Aplicación del patrón MVC

2.4.2 Diagrama de clases del diseño

El diagrama de clases del diseño es el resultado de un refinamiento del diagrama de clases del análisis, y muestra la estructura del sistema a un nivel de detalles de diseño que permitirá la implementación de dichas clases (Pressman, 2010). La Figura a continuación muestra el diagrama de clases de diseño (o estereotipos web) del requisito Modificar estudiante.

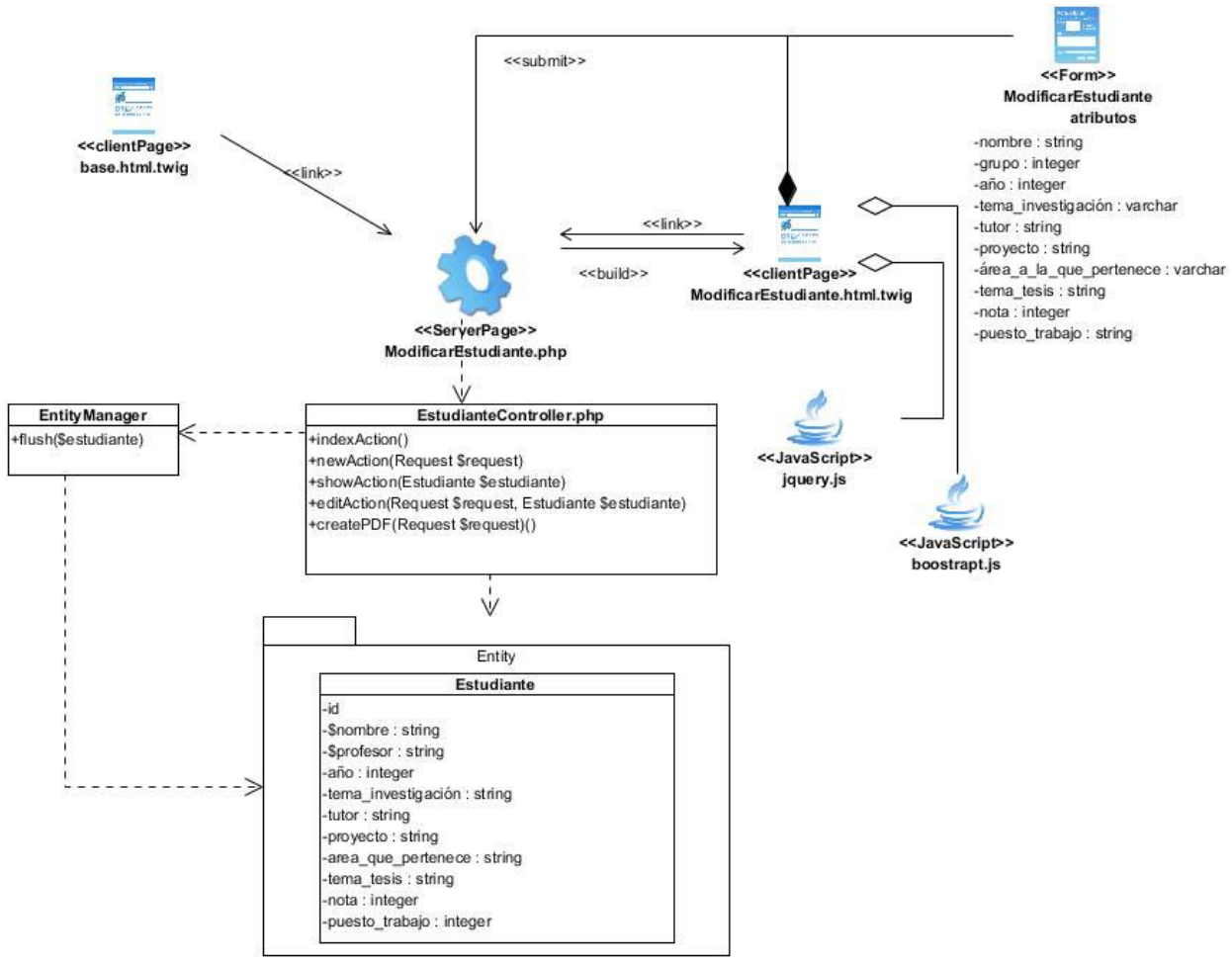


Figura 7: Diagrama de clases del diseño del requisito Modificar estudiante

El diagrama de clases del diseño está compuesto por diferentes tipos de clases y las relaciones existentes entre ellas:

- Clases de interfaz: Definen todas las abstracciones necesarias para que el usuario interactúe con la aplicación. En el diagrama se representan las páginas clientes: *base.html.twig*, *GestionarEstudiante.html.twig* y *AdicionarEstudiante.html.twig*.
- Clase controladora: permite encapsular las funcionalidades necesarias para interactuar con las clases interfaces, las clases entidades y de acceso a los datos. Agrupa la lógica del negocio ya que responde a las peticiones HTTP provenientes del cliente e invoca peticiones a la base de datos. En este caso es la clase *GestionarEstudianteController.php.*
- Clase de acceso a datos: Representada en el diagrama por la clase EntityManager, es la que permite la comunicación entre la clase controladora y la base de datos.

2.4.3 Patrones del diseño de software

Los patrones de diseño son una herramienta de apoyo en la búsqueda de soluciones a una serie de problemas comunes que se presentan en el desarrollo de software. Facilitan la reutilización y la capacidad de expansión del software, reducen la complejidad del código y del acoplamiento y facilitan el mantenimiento (Salazar, Montenegro y Rodríguez 2012). En el presente trabajo se utilizan los patrones GRASP y GoF, con el objetivo de describir los principios fundamentales del diseño de objetos ya que cada uno describe un problema, una solución y los beneficios de implementación.

Patrones GoF

Los patrones GoF (The Gang of Four, o La banda de los cuatro), conocidos así por las cuatro personas que lo propusieron, son patrones de diseño de software que se utilizan para proveer soluciones comprobadas a problemas comunes que se presentan en el desarrollo de software. Se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Carlos A. Guerrero, Johanna M. Suárez y Luz E. Gutiérrez, 2013). En el diseño de la solución se utilizó el patrón Decorador, contenido dentro de la categoría de los patrones estructurales de GoF. El propósito de este patrón es añadir funcionalidades adicionales a una clase dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia. Aporta una mayor flexibilidad que la herencia estática, permitiendo añadir una funcionalidad dos o más veces (Politécnica de Madrid, 2013). Este patrón se implementa en la clase *View* de Symfony, que es utilizada en la creación de plantillas para las páginas *html.twig*.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}Inicio{% endblock %}</title>
        <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" />
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta content="width=device-width, initial-scale=1" name="viewport">
        <meta content="" name="description">
        <meta content="" name="author">

        {% block stylesheet %}
            <!-- BEGIN GLOBAL MANDATORY STYLES -->
            <link href="{{ asset('bundles/app/plugins/font-awesome/css/font-awesome.min.css') }}" rel="stylesheet" type="text/css">
            <link href="{{ asset('bundles/app/plugins/bootstrap/css/bootstrap.min.css') }}" rel="stylesheet" type="text/css">
            <link href="{{ asset('bundles/app/plugins/uniform/css/uniform.default.css') }}" rel="stylesheet" type="text/css">
            <!-- END GLOBAL MANDATORY STYLES -->
            <!-- BEGIN THEME STYLES -->
            <link href="{{ asset('bundles/app/css/style-metronic.css') }}" rel="stylesheet" type="text/css">
            <link href="{{ asset('bundles/app/css/style.css') }}" rel="stylesheet" type="text/css">
            <link href="{{ asset('bundles/app/css/style-responsive.css') }}" rel="stylesheet" type="text/css">
            <link href="{{ asset('bundles/app/css/plugins.css') }}" rel="stylesheet" type="text/css">
            {#<link href="{{ asset('bundles/app/css/pages/tasks.css') }}" rel="stylesheet" type="text/css">#}
            <link href="{{ asset('bundles/app/css/themes/default.css') }}" rel="stylesheet" type="text/css" id="style_color">
            {#<link href="{{ asset('bundles/app/css/print.css') }}" rel="stylesheet" type="text/css" media="print">#}
            <link href="{{ asset('bundles/app/css/custom.css') }}" rel="stylesheet" type="text/css">
            <!-- END THEME STYLES -->
        {% endblock %}
```

Figura 8: Fragmento de código de la clase Base.html.twig

```
{% extends 'base.html.twig' %}
{% block stylesheet %}
    {{ parent() }}

    <link rel="stylesheet" type="text/css" href="{{ asset('bundles/app/css/select2.css') }}">
    <link rel="stylesheet" type="text/css" href="{{ asset('bundles/app/css/select2-metronic.css') }}">
    <link rel="stylesheet" type="text/css" href="{{ asset('bundles/app/css/bootstrap-wysihtml5.css') }
    <link rel="stylesheet" type="text/css" href="{{ asset('bundles/app/css/bootstrap-markdown.min.css'
{% endblock %}
{% block body %}

    {{ form_start(form) }}
```

Figura 9: Ejemplo del uso del patrón decorador

Otros patrones GoF implementados en el marco de trabajo Symfony 2 son los siguientes:

Creacionales

- Fábrica: Permite crear diferentes familias de objetos. En Symfony 2 se emplea para la creación de objetos que pueden ser utilizados como servicios para otras clases, indicándole al contenedor de servicios que llame a un método del objeto creado y no directamente una instancia del objeto (Politécnica de Madrid, 2013).

```php
/**
 * Listado profesor Estudiante
 * @Route("/obtener/profEst/{id}", name="profe_est")
 */
public function obtenerEstProfe($id)
{
    $em = $this->getDoctrine()->getManager();
    $estudiantes = $em->getRepository( className: 'AppBundle:Estudiante')->findAll();
    $aux=[];

    foreach ($estudiantes as $estudiante ) {
        if($estudiante->getProfesor()!=null)
        if($estudiante->getProfesor()->getId()==$id)
            $aux[$estudiante->getNombre()]=$estudiante;
   }
    $profes = $em->getRepository( className: 'AppBundle:Profesor')->findAll();

    return $this->render( view: 'profesor/listado_profesor_estudiante.html.twig', array(
        'profesores' => $aux,
        'nro' => 1,
        'listado_profe' => $profes
    ));
}
```

Figura 10: Utilización del patrón Fábrica en la clase ProfesorController.php

De Comportamiento

- Observador: Define una dependencia “uno a muchos” entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente (Politécnica de Madrid, 2013) .

El desarrollador de la aplicación no visualiza la utilización de estos patrones ya que son implementados y utilizados internamente en el marco de trabajo Symfony 2.

GRASP

Los patrones GRASP (General Responsibility Assignment Software Patterns o Patrones Generales de Software para la Asignación de Responsabilidades) son patrones generales de software para asignación de responsabilidades y estos fomentan una serie de buenas prácticas para el diseño del software (Pérez, 2012).

Los principales patrones de GRASP son:

- Experto: asigna la responsabilidad al experto en información. Se observa el uso de este patrón en todas las clases a utilizar en la solución ya que cada clase conoce su información y es la encargada de implementar las funcionalidades que brindan información de las mismas. (Usaola, 2012) En la

siguiente figura se muestra la clase Estudiante, la cual contiene toda la información referente a los estudiantes.

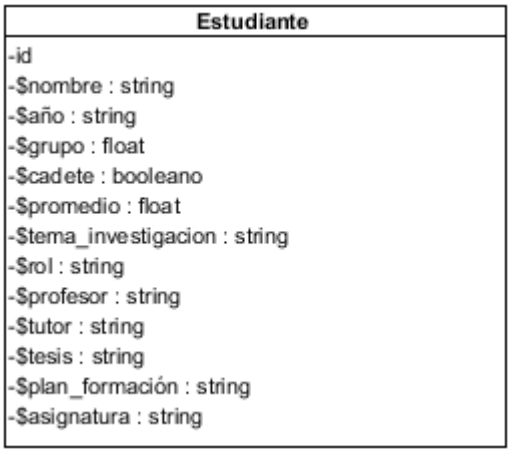


Figura 11: Clase Estudiante

- Creador: designa la clase que debe ser responsable de crear una instancia de otra. (Usaola, 2012) Este patrón es utilizado en las clases controladoras, en las cuales se crean objetos de las clases entidades para su manipulación.
- Bajo acoplamiento: este patrón expresa que entre las clases deberán existir pocas ataduras, es decir, estas estarán lo menos relacionadas posible de forma tal que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases. Permite crear clases más independientes y más reutilizables. (Usaola, 2012)
  Este patrón se evidencia en la poca atadura que existen entre las clases entidades.
- Alta cohesión: asigna responsabilidades de manera que la cohesión se mantenga alta, de este modo se hace más fácil la comprensión, fácil reutilización, fácil mantenibilidad y mayor resistencia a los cambios. (Usaola, 2012)
  Este patrón se evidencia en el controlador EstudianteController.php en el que si se realiza un cambio en alguno de sus atributos también se deben modificar los métodos relacionados con los mismos.
- Controlador: asigna responsabilidades al “controlador”, que será una clase que represente al sistema completo, una parte activa del mundo real que desencadena de tal evento, representa un manejador artificial de eventos (Usaola, 2012). Como ejemplo se muestra la clase EstudianteController.php.

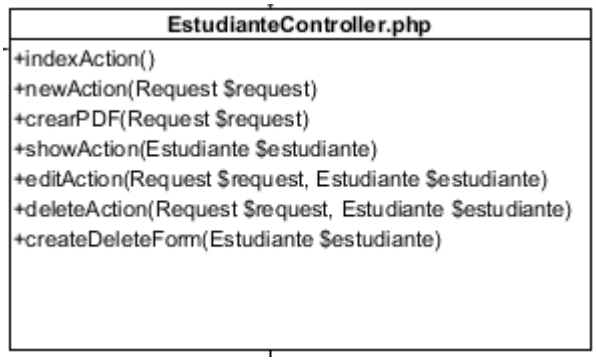


Figura 12: Clase controladora EstudianteController.php

2.4.4 Modelo de datos

Un modelo de datos describe los datos que apoyan los procesos de una organización y refleja con exactitud cómo serán guardados los datos en la base de datos. Contiene cada una de las tablas de la aplicación, así como sus atributos y relaciones. El modelo se encuentra normalizado en Tercera Forma Normal, de las seis formas normales existentes para la normalización de las bases de datos relacionales, por lo que todos los atributos de las tablas del modelo, dependen únicamente de la llave primaria.

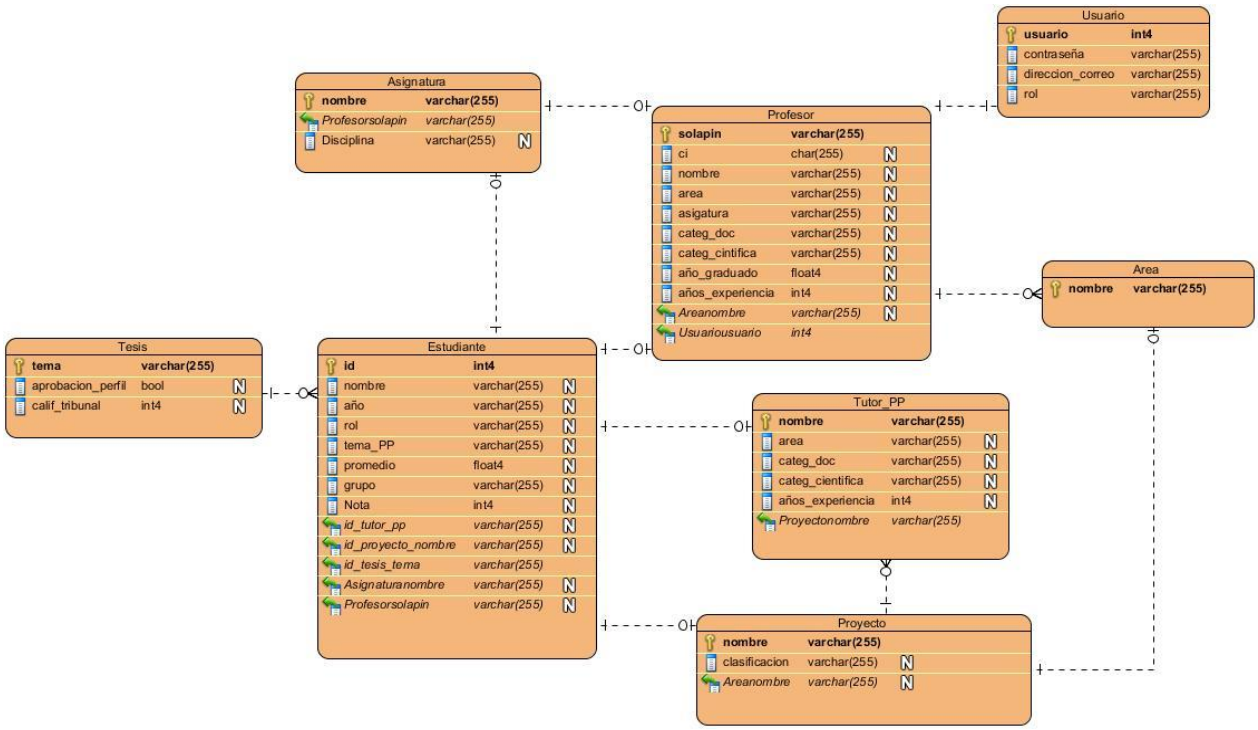


Figura 13: Modelo de datos

2.5 Conclusiones parciales del capítulo

Se elaboraron los productos de trabajo relacionados con las disciplinas de Modelado de negocio, Requisitos y Análisis y diseño de la metodología, sirviendo estos como guía para la implementación del sistema. El estudio de los sistemas y las entrevistas realizadas permitieron definir 41 requisitos funcionales y 9 requisitos no funcionales. La definición del modelo de datos relacional permitió un mejor entendimiento y organización de la información. La aplicación de los patrones del diseño y arquitectónicos permitieron crear una estructura común y conocida para varios programadores, lo que facilitará un mejor mantenimiento y reutilización del sistema.

Capítulo 3: Implementación y Pruebas

3.1 Introducción

En el presente capítulo se describe el modelo de implementación de la solución propuesta, así como el estándar de codificación utilizado. Por último, se realizan las pruebas funcionales que permiten comprobar el correcto funcionamiento del sistema. Luego se realizan las pruebas de aceptación con el cliente para verificar que el sistema desarrollado cumple con las necesidades planteadas inicialmente.

3.2 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Este modelo representa también la organización de los componentes de acuerdo con los mecanismos de estructuración y modulación disponibles en el entorno de implementación y en el lenguaje de programación empleado y cómo dependen los componentes unos de otros (Paz, 2016).

3.2.1 Diagrama de componentes

El diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean éstos de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo (UNAD, 2015).

La Figura 14 presenta el diagrama de componentes de la solución, en él se observan los componentes o unidades físicas de implementación con interfaces bien definidas, que son utilizados como parte reemplazable de un sistema e incorporan la implementación de ciertas clases del diseño del software y las relaciones existentes entre ellos (Ivar, y otros, 2002).

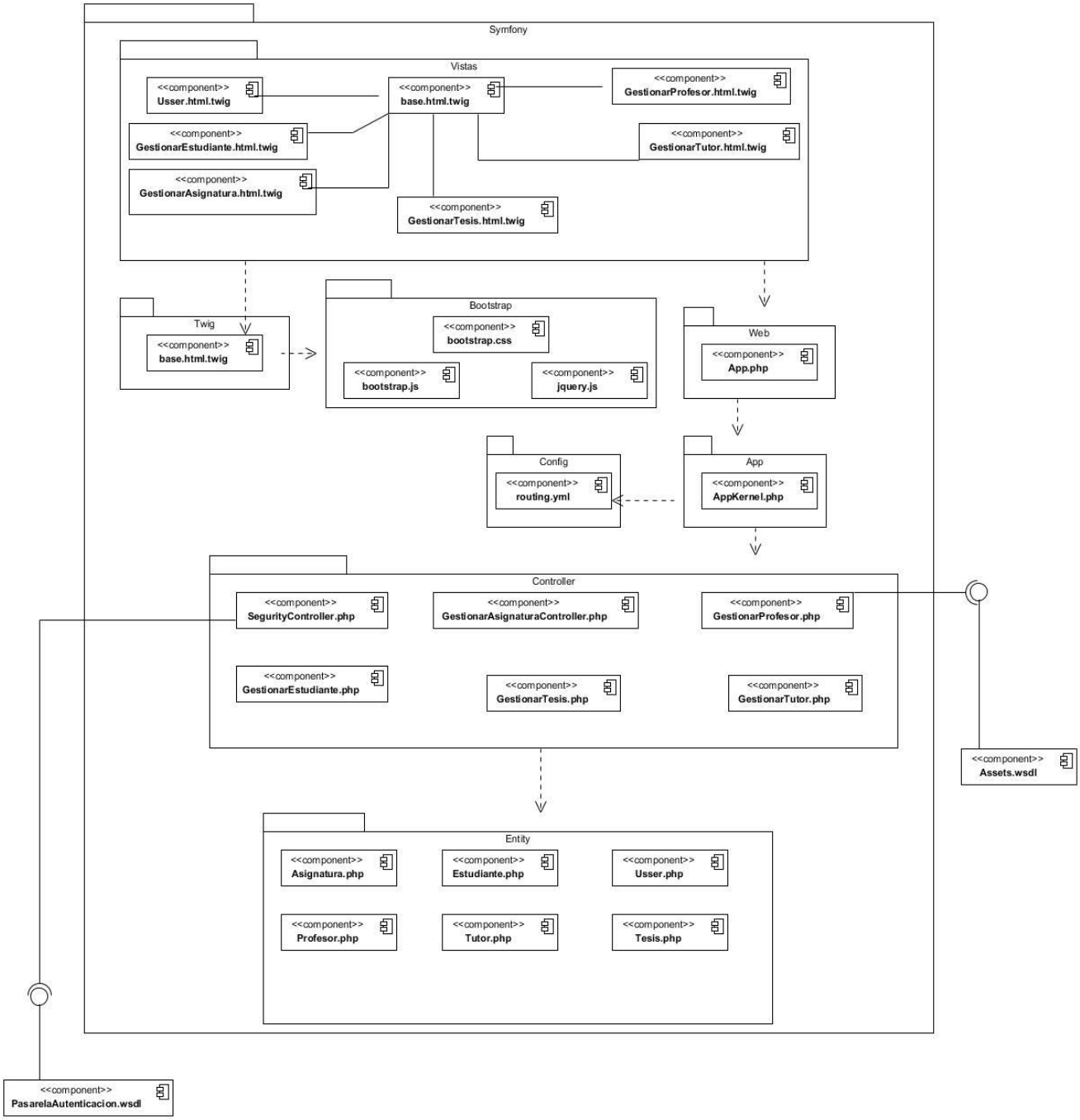


Figura 14: Diagrama de componentes. Elaboración propia

Los componentes contenidos en el diagrama anterior son:

- Vistas: representan las páginas clientes que permiten y facilitan la comunicación de los usuarios con el sistema.
- *base.html.twig*: es la plantilla base para la organización de las páginas *html.twig.*
- *bootstrap.css*: archivos de estilos de formato visual (CSS) de las páginas *html.twig.*

- *bootstrap.js y jquery.js:* archivos de validación de Javascript (JS).
- *App.php:* componente que representa al controlador frontal o punto de acceso a la aplicación.
- *AppKernel.php:* representa a la clase responsable de cargar el archivo que gestiona las rutas de las clases que contienen las funcionalidades que responden a las peticiones del cliente y con las clases controladoras.
- *routing.yml:* archivo que contiene las rutas de las funcionalidades del sistema.
- *Controller*: clases controladoras que gestionan el flujo de información en el sistema.
- *Entity*: representan a las clases entidades que contienen la información persistente del sistema.
- *PasarelaAutenticacion.wsdl:* permite la autenticación y obtención de los datos de los usuarios del sistema, en este caso mediante un usuario y su contraseña del dominio UCI.
- *Assets.wsdl*: permite la obtención de los datos referente a los profesores como el solapín y el número de carné de identidad.

3.2.2 Interfaces del módulo Gestión de los procesos de PP

La interfaz de usuario es el medio de comunicación entre el usuario y el sistema y debe cumplir con las tres reglas siguientes: dar control al usuario, reducir la carga de memoria del usuario y debe ser consistente. El cumplimiento de estas reglas permite desarrollar interfaces flexibles, en las que el usuario no realice acciones innecesarias y no deseadas, así como la interrupción de una secuencia de acciones en el instante requerido por el usuario. Además, deben desglosar la información de forma progresiva a través de una estructura organizada, orientar al usuario en las tareas a desarrollar y que esas tareas se realicen en el contexto adecuado (Pressman, 2010). En la Figura 15, se muestra la interfaz de usuario con la distribución de los estudiantes en la PP, la cual permite realizar un reporte en PDF.

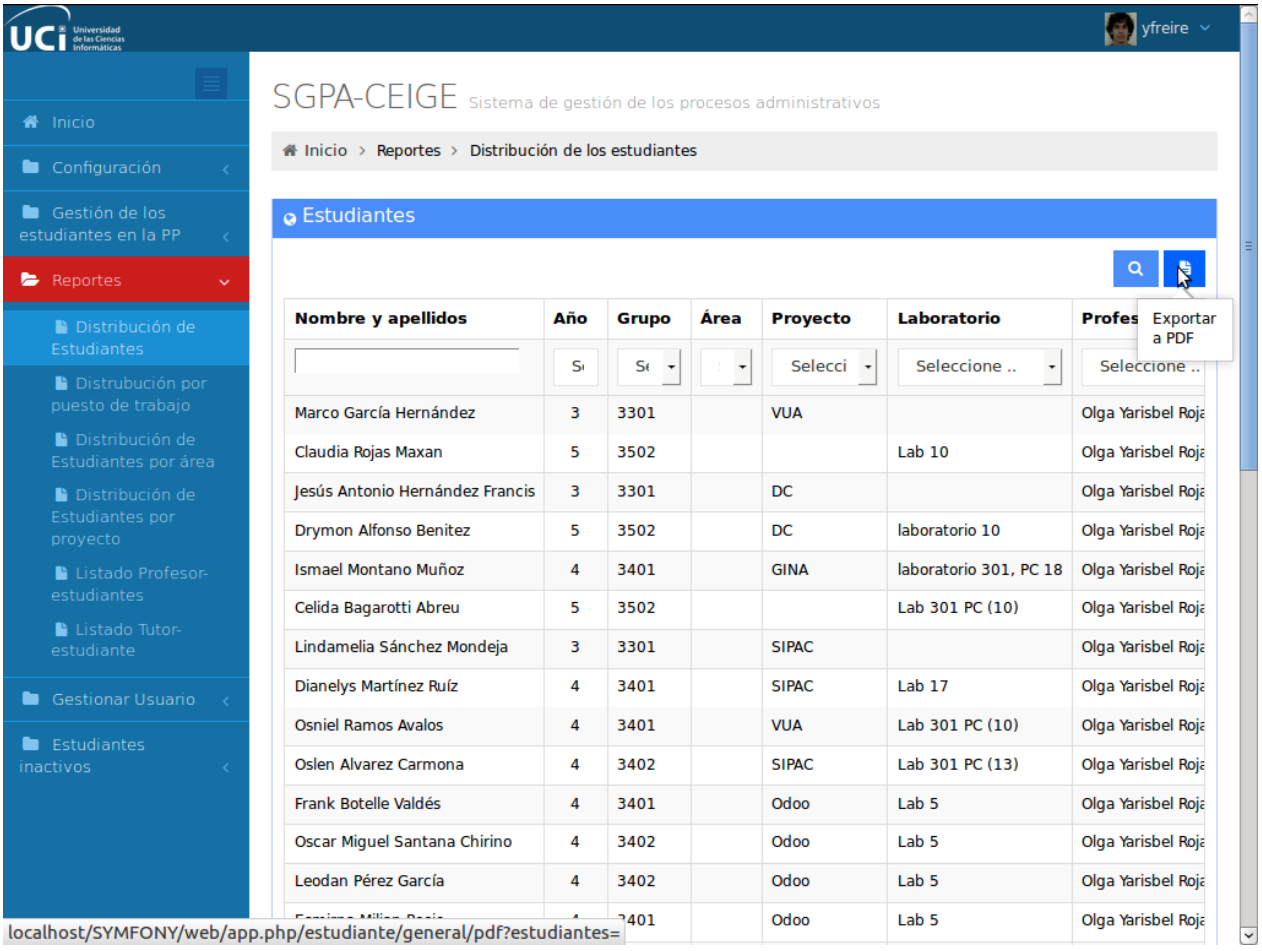


Figura 15 Interfaz con la distribución de los estudiantes en la PP

3.3 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes de hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software. (Guadarrama, 2013)

A continuación, se muestra el diagrama de despliegue de la solución implementada. En la computadora cliente se accede al sistema mediante el protocolo HTTPS. El sistema se encuentra instalado en un servidor web y maneja las peticiones que realiza el cliente. La información del sistema, de las acciones ejecutadas y del negocio es almacenada en la base de datos y mediante el protocolo HTTPS se realiza el consumo de los servicios necesarios para el correcto desarrollo del sistema, como el PasarelaAutenticacion.wsdl y el servicio Asset.wsdl.

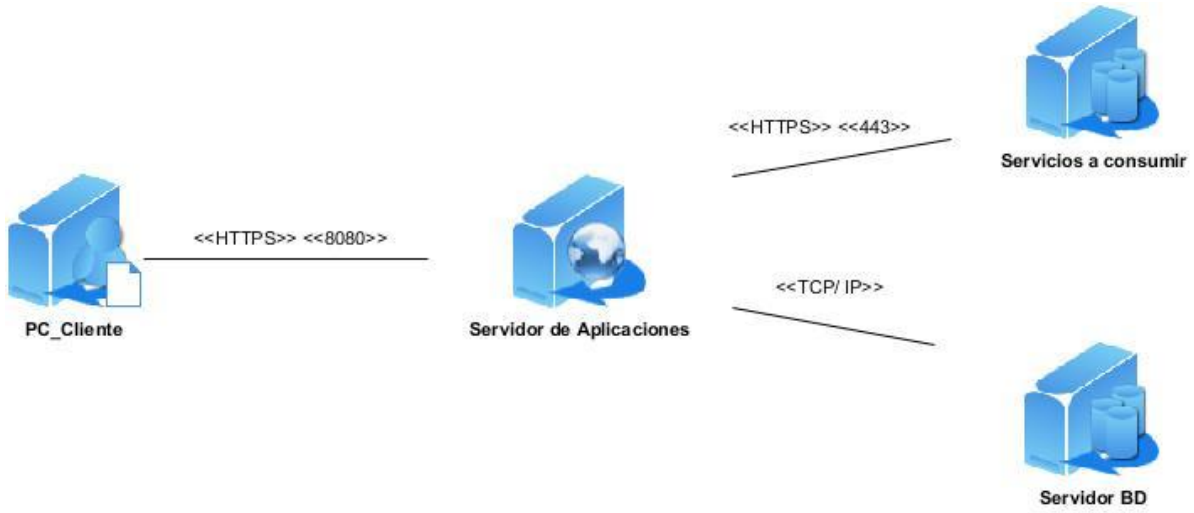


Figura 16: Diagrama de despliegue

3.4 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. El uso de estándares de codificación permite lograr un código legible y reutilizable, de tal forma que se pueda aumentar su mantenibilidad a lo largo del tiempo. El marco de trabajo Symfony2 emplea una estructura de código de la que a continuación se describen las empleadas en la solución. (Alfonso, 2012).

Estructura del código

- Añadir un solo espacio alrededor de los operadores (==, &&).
- Añadir una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último.
- Añadir una línea en blanco antes de las declaraciones *return*, a menos que el valor devuelto solo sea dentro de un grupo de declaraciones (tal como una declaración *if*).
- Declarar las propiedades de clase antes que los métodos.
- Usa llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.
- Usar paréntesis cuando se instancie una clase a pesar del número de argumentos que su constructor posea.

Nomenclaturas utilizadas

Para las nomenclaturas de las clases se emplea la notación CamelCase en su variante lowerCamelCase, que se aplica a frases o palabras compuestas, para esta variante la primera palabra siempre inicia en minúsculas.

- Nomenclatura de los controladores: los controladores después del nombre llevan el sufijo Controller. Por ejemplo, EstudianteController.
- Nomenclatura de las variables: El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación lowerCamelCasing y comenzando con un prefijo según el tipo de datos. Ejemplo:

$nombre y $temaInvestigacion.

- Usa guiones bajos para nombres de opción y nombres de parámetro.
- Sufija las interfaces con *Interface*.
- Sufija las excepciones con *Exception*.
- Utiliza caracteres alfanuméricos y guiones bajos para los nombres de archivo.

3.5 Aplicación de las métricas de diseño de software

A partir de las métricas que proponen Lorenz y Kidd se utilizarán las siguientes, para el análisis del correcto funcionamiento del diseño realizado.

3.5.1 Tamaño operacional de las clases (TOC)

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 94% de las clases empleadas en el sistema poseen evaluaciones positivas de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización). A continuación, se muestran los resultados obtenidos en las: Figura 17, Figura 18 y Figura 19.

.

Figura 17: Resultados del análisis de responsabilidad

Figura 18: Resultado del análisis de complejidad

Figura 19: Resultados del análisis de reutilización

En la siguiente tabla se muestra con mayor nivel de detalles los resultados de la aplicación de la métrica TOC.

Tabla 9: Resultados de aplicar TOC

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
Asignatura	5	Baja	Baja	Alta
Area	14	Media	Media	Media
Estudiante	28	Alta	Alta	Baja
Profesor	20	Media	Media	Media
Proyecto	10	Baja	Baja	Alta
Tesis	9	Baja	Baja	Alta
Tutor	16	Media	Media	Media
Usuario	16	Media	Media	Media
AsignaturaController	6	Baja	Baja	Alta
AreaController	6	Baja	Baja	Alta
EstudianteController	18	Media	Media	Media
ProfesorController	11	Baja	Baja	Alta
ProyectoController	6	Baja	Baja	Alta
TesisController	6	Baja	Baja	Alta
TutorController	9	Baja	Baja	Alta
UsuarioController	6	Baja	Baja	Alta

3.5.2 Relaciones entre clases (RC)

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 81% de las clases empleadas posee menos de 3 dependencias de otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización). A continuación, se muestran los resultados obtenidos en las: Figura 20, Figura 21, Figura 22 y Figura 23.

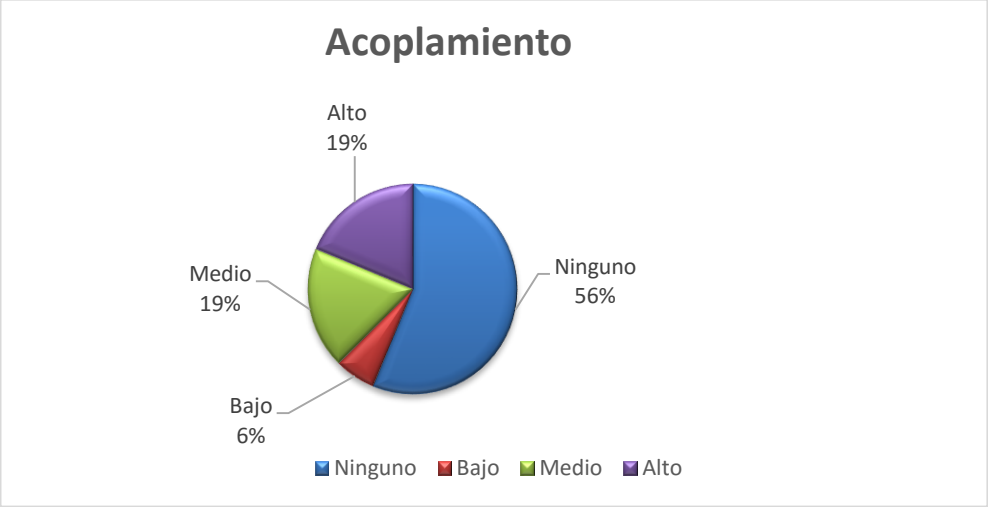


Figura 20: Resultado de análisis de acoplamiento

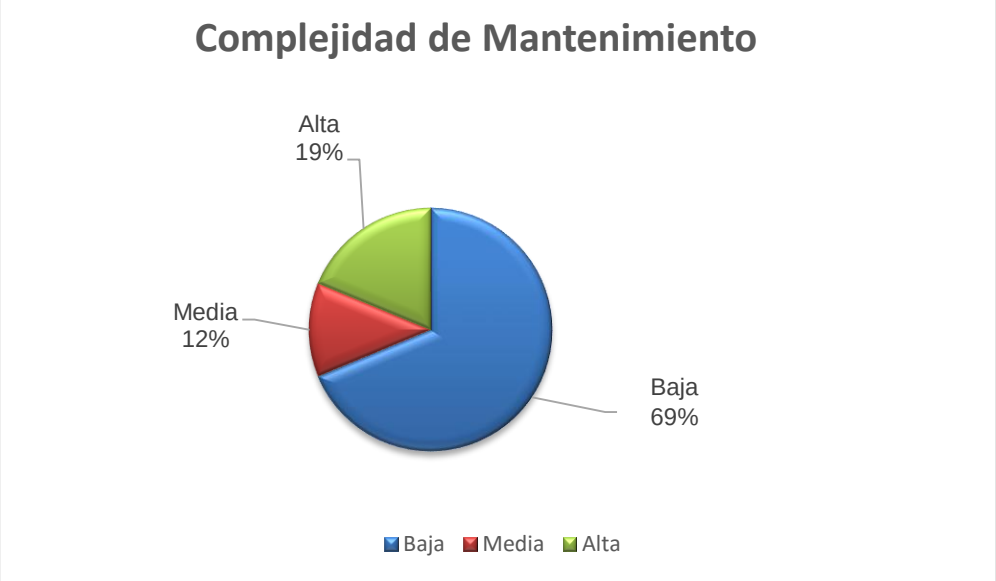


Figura 21: Resultado del análisis de complejidad de mantenimiento

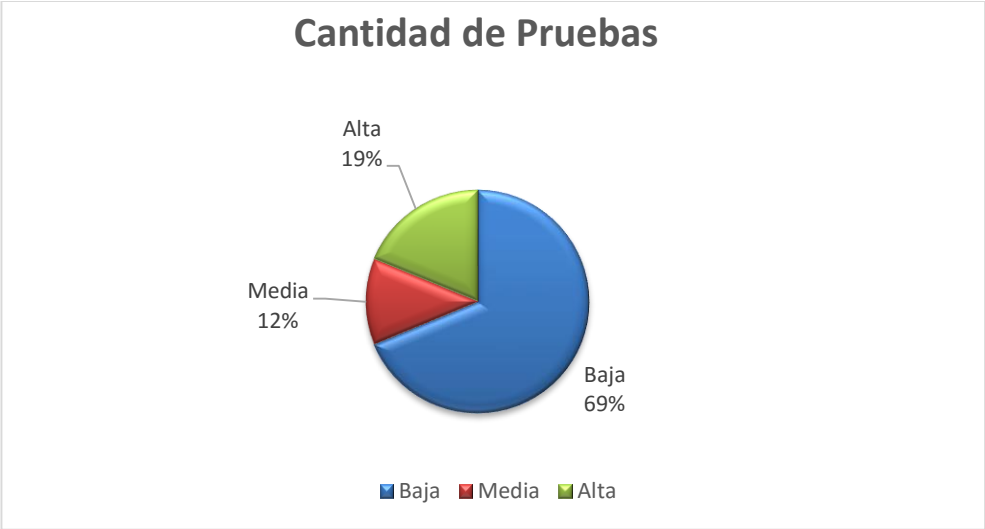


Figura 22: Resultado del análisis de cantidad de pruebas

Figura 23: Resultado del análisis de reutilización

En la siguiente tabla se muestra con mayor nivel de detalles los resultados de la aplicación de la métrica RC.

Tabla 10: Resultados de aplicar RC

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
Asignatura	2	Medio	Media	Media	Media
Area	2	Medio	Media	Media	Media
Estudiante	5	Alto	Alta	Baja	Alta
Profesor	3	Alto	Alta	Baja	Alta
Proyecto	3	Alto	Alta	Baja	Alta
Tesis	1	Bajo	Baja	Alta	Baja
Tutor	2	Medio	Baja	Media	Media
Usuario	0	Ninguno	Baja	Alta	Baja
AsignaturaController	0	Ninguno	Baja	Alta	Baja
AreaController	0	Ninguno	Baja	Alta	Baja
EstudianteController	0	Ninguno	Baja	Alta	Baja
ProfesorController	0	Ninguno	Baja	Alta	Baja
ProyectoController	0	Ninguno	Baja	Alta	Baja
TesisController	0	Ninguno	Baja	Alta	Baja
TutorController	0	Ninguno	Baja	Alta	Baja
UsuarioController	0	Ninguno	Baja	Alta	Baja

3.6 Aplicación de las pruebas de software

3.6.1 Pruebas de caja blanca

La prueba de caja blanca 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 estos métodos el ingeniero de software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas; ejecuten todos los bucles en sus límites y con sus limitaciones operacionales; y ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2005).

Las técnicas usadas para ejecutar dichas pruebas son las descritas a continuación:

✓ **Técnica del camino básico:** permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

- ✓ **Prueba de condición:** ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.
- ✓ **Prueba de Flujo de Datos:** se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.
- ✓ **Prueba de Bucles:** se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución todos los bucles en sus límites operacionales.

Con el objetivo de valorar la calidad con la que se llevó a cabo la implementación, fue necesario aplicar una de las pruebas descritas anteriormente: **la técnica de camino básico.**

Para ello fue necesario seguir los siguientes pasos:

- ✓ A partir del diseño o del código fuente, dibujar el grafo de flujo asociado.
- ✓ Calcular la complejidad ciclomática del grafo.
- ✓ Determinar el conjunto básico de caminos independientes.
- ✓ Preparar los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Para dar cumplimiento a los anteriores pasos básicos se enumeran cada una de las sentencias de código de la funcionalidad *modificarTutor(),* que es la encargada de modificar los datos del tutor seleccionado.

```
public function editAction(Request $request, Tutor $tutor)
{

    $deleteForm = $this->createDeleteForm($tutor);
    $editForm = $this->createForm( type: 'AppBundle\Form\TutorType', $tutor);
    $editForm->handleRequest($request);

    if ($editForm->isSubmitted() && $editForm->isValid()) {
        $this->getDoctrine()->getManager()->flush();

        $session=new Session();
        $session->getFlashBag()->add( type: 'correcto', message: 'El tutor ha sido editado satisfactoriamente');



        return $this->redirectToRoute( route: 'tutor_index');
    }

    return $this->render( view: 'tutor/edit.html.twig', array(
        'tutor' => $tutor,
        'edit_form' => $editForm->createView(),
        'delete_form' => $deleteForm->createView(),
    ));
}
```

1 2 3 4 5

Figura 24: Fragmento del código del método modificarTutor()

Una vez obtenidas las sentencias se construye el grafo, quedando de la forma mostrada en la Figura 25.

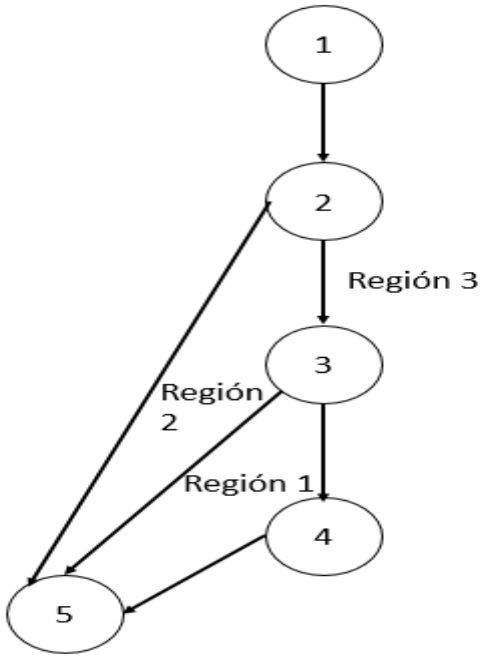


Figura 25 Grafo de flujo asociado al método modificarTutor()

La complejidad ciclomática es la métrica de software con que se define la cantidad de caminos independientes de cada una de las funcionalidades del programa y provee el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez (Pressman, 2005). Dicho cálculo es necesario efectuarlo mediante tres vías o fórmulas de manera tal que quede justificado el resultado, siendo el mismo en cada caso:

Se calcula de la siguiente manera:

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

V(G)=6-5+2

V(G)= 3

2. V (G) = P + 1 siendo “P” la cantidad de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 2 + 1

V (G) = 3

3. V (G) = R siendo “R” la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más

V (G) = 3

El cálculo efectuado mediante las fórmulas antes presentadas muestra una complejidad ciclomática de valor tres, de manera que existen tres posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

Camino básico 1: 1-2-3-4-5.

Camino básico 2: 1-2-5.

Camino básico 2: 1-2-3-5.

Luego se definen los casos de prueba para comprobar la ejecución de cada camino del conjunto básico. En el diseño de los casos de prueba se deben especificar los siguientes elementos:

Descripción: contiene una descripción sobre las restricciones de los datos de entrada que debe tener el caso de prueba.

Condición de ejecución: se especifican los parámetros que debe poseer el caso de prueba para que se cumpla una condición deseada como respuesta del funcionamiento del procedimiento.

Entrada: se muestran los parámetros de entrada al procedimiento.

Resultados esperados: se explica el resultado esperado de la ejecución del procedimiento.
La tabla muestra el diseño de caso de prueba para el camino 1 del conjunto básico de caminos linealmente independientes, correspondiente a la funcionalidad modificarTutor(). En este camino se prueba la modificación de los datos de un tutor de forma satisfactoria.

Tabla 11 Diseño de caso de prueba para el camino 1

Diseño de caso de prueba para el camino 1	
Descripción	Los datos de entrada son válidos y deben cumplir con el formato especificado.
Condición de ejecución	Campos válidos: El campo Nombre permite la entrada del nombre y los apellidos del tutor y es una cadena de caracteres no nula. El campo Cargo es un campo de selección y es no nulo. El campo Área a la que pertenece es un campo de selección y es no nulo. El campo Categoría docente es un campo de selección y es no nulo. El campo Categoría científica es un campo de selección y es no nulo. El campo Años de experiencia es un campo numérico, entero y puede ser nulo.
Entrada	Nombre: Olga Yarisbel Rojas Grass Cargo: Profesor Área a la que pertenece: PP Categoría docente: Auxiliar Categoría científica: Máster Años de experiencia: 6
Resultados esperados	Se modifican los datos del tutor satisfactoriamente.

A partir de la aplicación del caso de prueba expuesto anteriormente se evidenció que el flujo de trabajo de las funcionalidades es correcto, ya que se comprobó que cada sentencia es ejecutada al menos una vez, cumpliéndose así las condiciones de la prueba y el resultado esperado es satisfactorio.

3.6.2 Pruebas de caja negra

Las pruebas de caja negra se centran en los requisitos funcionales. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2005). El tipo de prueba a aplicar es la partición equivalente. Es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (Pressman, 2005).

En el caso del Módulo para gestionar los procesos del Departamento de PP, se diseñó un caso de prueba por cada requisito funcional. A continuación, se muestran los resultados de las pruebas realizadas.

Tabla 12: No conformidades detectadas por iteración.

No conformidades	Aplicación	Ortografía
Primera iteración	18	4
Segunda iteración	7	2
Tercera iteración	_	_

Las pruebas se realizaron por un especialista del grupo de calidad de CEIGE de la Facultad 3. Se realizaron tres iteraciones, en la primera se encontraron 22 no conformidades, en la segunda iteración nueve y en la tercera iteración no se encontraron no conformidades. El sistema desarrollado fue liberado y para ello se emitió el acta de liberación. En la Figura 25 se presenta un gráfico con los resultados de las pruebas de liberación.

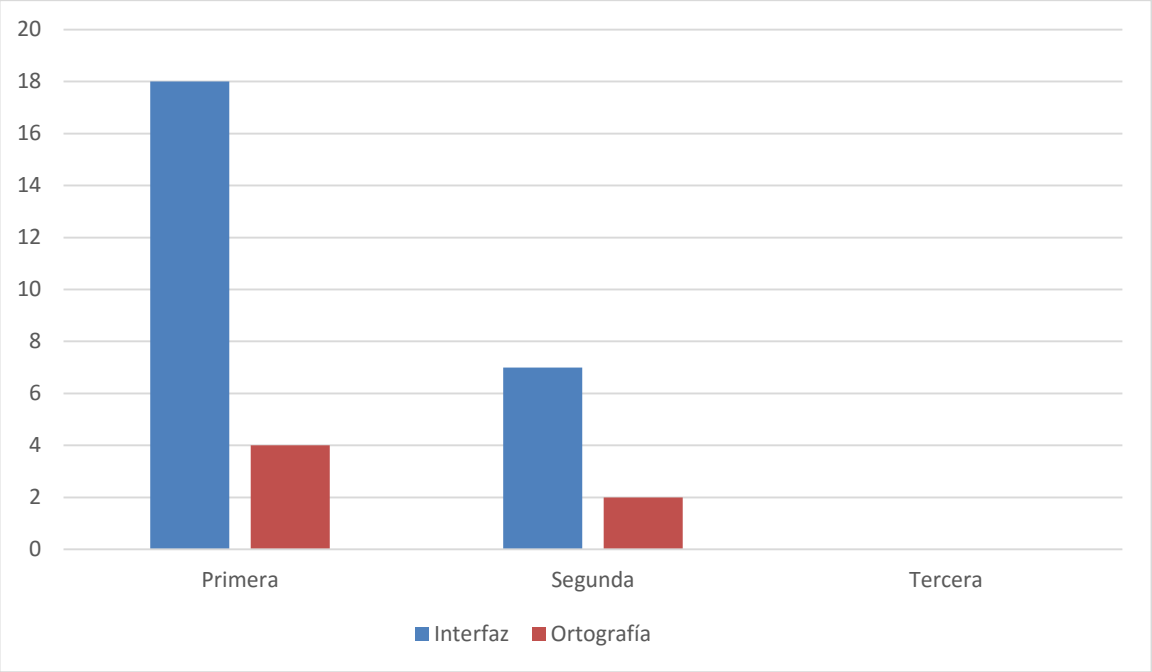


Figura 26 Resultado de las pruebas funcionales

3.7 Aplicación y resultados de las pruebas de aceptación

Las pruebas de aceptación fueron realizadas por el Jefe de departamento de PP de CEIGE y un profesor del departamento. Para ello se realizaron las siguientes actividades:

1. Revisión y aprobación de los requisitos funcionales y no funcionales por el cliente.
2. Realización de pruebas exploratorias por parte del cliente al sistema desarrollado, para validar el cumplimiento de los requisitos aprobados.

Una vez validadas todas las funcionalidades del sistema, el cliente emitió el acta de aceptación que puede ser consultada en el Anexo 1.

3.8 Conclusiones parciales del capítulo

El empleo de métricas de software validó el diseño propuesto en el capítulo anterior y demostró que el componente cuenta con un diseño robusto. El uso de estándares de codificación aseguró una homogeneidad en la nomenclatura de las clases y métodos, que facilita su comprensión a otros programadores. La práctica de pruebas de caja blanca y caja negra comprobó el cumplimiento de los requisitos funcionales y la correcta ejecución del código.

Conclusiones Generales

La investigación realizada cumple con los objetivos planteados y se arriba a las siguientes conclusiones:

- El análisis de los referentes teóricos y el estudio realizado a los diferentes sistemas proporcionó un conjunto de funcionalidades a tener en cuenta en el desarrollo del módulo informático para el control y seguimiento de los estudiantes en las asignaturas de PID.
- Se obtuvo un módulo que permite el control y seguimiento de los estudiantes en la Práctica Profesional de CEIGE, cumpliendo con los requisitos definidos por el cliente.
- El empleo de patrones de diseño e implementación y la validación empleando métricas y pruebas demostró que se obtuvo una solución adaptable, mantenible y reutilizable.

Recomendaciones

Integrar el módulo al Sistema para la gestión de los procesos administrativos de CEIGE.

Se recomienda como material de estudio en caso de que se añadan estas funcionalidades al Sistema de Gestión Universitaria.

(UCI), Universidad de las Ciencias Informáticas. 2014. *Plan de estudios D. Ingenieria en Ciencias Informáticas.* La Habana : Ministerio de Educación Superior, 2014.

Alfonso, Damián Pérez. 2012. *Normas y estándares de codificación.* La Habana : Universidad de las Ciencias Informáticas., 2012.

Alvares, Miguel Angel. 2014. desarrolloWeb.com. [En línea] 2 de enero de 2014. [Citado el: 9 de abril de 2017.] https://desarrolloweb.com/articulos/que-es-mvc.html.

Alvarez, A. Cristiá, y otros. 2010. *Guía para la personalización de PostgreSQL 8.4. Centro de Tecnologías de Gestión de Datos (DATEC).* La Habana, Cuba : Universidad de las Ciencias Informáticas, 2010.

Andreu, R. Ricart J, E. Y Valor. 1991. *Estrategia y Sistemas de información.* Madrid : s.n., 1991.

Balmaseda, Osbel Castro. 2008. *Portal para la gestion de la informacion de la facultad 7.* Habana : Universidad de las Ciencias Informáticas, 2008.

Bastidas, C.V Mlina. 2012. Sistema de gestión de pedidos y proformas dinámicas por internet para la empresa. JIMEMOR CIA.LTDA utilizando Symfony in Trabajo de diploma para optar por el título de Ingeniería en Sistemas Computacionales. [En línea] Universidad Técnica del Norte, Ibarra, Ecuador, 2012. [Citado el: 18 de 1 de 2017.] http://repositorio.utn.edu.ec/bitstream/123456789/1814/1/Tesis%20formato%20Pdf.pdf..

Bio, convenio de desempeño de la universidad del Bio. Entorno Virtual de Practicas Profesionales. [En línea] [Citado el: 2016 de Diciembre de 13.] http://convenio.ubiobio.cl/biblioteca/pdf/sistema_de_informacion_para_la_gestion.

Carlos Reynoso, Nicolás Kicillof. 2004. *Estilos y patrones en la estrategia de arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

Concepto definicion ABC. [En línea] [Citado el: 13 de 4 de 2017.] https://www.definicionabc.com/general/estudiante.php.

2014. ConceptoDefinicion.de. [En línea] Diciembre de 2014. http://conceptodefinicion.de/estudiante/.

Cuadernos de Educación y Desarrollo. **Navarro, María Fernanda Ramírez. 2009.** 2, Guadalajara, México : eumed.net, 2009, Vol. 1.

Delgado, Lic. Yanoski Calderín. 2013. *GESTACAD.Sistema para la Gestión Académica.* Universidad Matanzas ´´Camilo Cienfuegos´´.Facultad de Informática. : s.n., 2013.

desarrollo, Karen Tremaria: Especialista de capacitación y. 2009. *Aprende a diseñar tu plan de formación.* 2009.

Domingo, Irene, Rius, G. y Cuenca, L. 2012. *Una revisión sobre el estado del arte en herramientas de modelado basado en UML.* Vigo.España : 6th International Conference on Industrial Engineering and Industrial Management. VI Congreso de Ingeniería de Organización, 2012.

Eguiluz, Javier. 2013. *Desarrollo web ágil con Symfony2.* 2013.

2017. Escuela Nacional de lenguas, linguísticas y traduccion ENALLT. [En línea] UNAM, 12 de 5 de 2017. [Citado el: 20 de 5 de 2017.] http://cele.unam.mx/index.php?categoria=10&subcategoria=74.

García, Nuria Cuesta. 2014. inLab FIB. [En línea] Universidad Politécnica de Catalunya, 2014. [Citado el: 8 de diciembre de 2016.] http://inlab.fib.upc.edu/es/prisma-sistema-de-gestion-academica-de-la-upc.

Gauchat, J.D. 2012. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona, España : Marcombo Ediciones Técnicas., 2012. ISBN: 978-84-267-1782-5..

Guadarrama, Jorge. 2013. *Modelo de implementación.* 2013.

Guerra, César Arturo. 2016. Software Guru.Seccion Requerimientos. [En línea] 2016. [Citado el: 8 de marzo de 2017.] https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

Iñigo, Dayron Cruz. 2007. *Sistema de Gestión de Información de la Facultad 8. Módulo para la gestión de información docente.* La Habana : s.n., 2007.

INTECO. 2009. *Ingeniería de software:metodologías y ciclos de vida.* España : Instituto Nacional de Tecnologías de la Comunicación.Laboratorio Nacional de Calidad de software, 2009.

Ivar, Jacobson y Grady, Booch. 2002. *El Lenguaje Unificado de Modelado. Manual de referencias.* Madrid. España : Pearson Educacion, 2002.

JetBrains. 2001. [En línea] 2001. [Citado el: 6 de marzo de 2017.] www.jetbrains.com/phpstorm/features.

Julián Pérez Porto, Ana Gardey. 2015. Definición de. [En línea] 2015. [Citado el: 20 de 4 de 2017.] http://definicion.de/practica-profesional/).

L. Welling; Laura Thomson. 2010. Desarrollo Web con PHP y MySQL. [En línea] Anaya Multimedia, 2010. [Citado el: 15 de 1 de 2017.] http://www.researchgate.net.

MADEJA. 2015. Especificación de Requisitos del Sistema. Marco de Desarrollo de la Junta de Andalucía. [En línea] Anda Lucía. España, 2015. [Citado el: 15 de enero de 2017.] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/407.

—. 2015. Marco de Desarrollo de la Junta de Andalucía. [En línea] 2015. [Citado el: 10 de febrero de 2017.] http://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/185.

Madrid, Universidad Politécnica de. 2013. Patrones del "Gang of Four". *Universidad Politécnica de Madrid.* [En línea] 2013. [Citado el: 16 de abril de 2017.] http://www.pdfbook-s.com/gang-of-four/1/.

Martínez Guerrero, J.M., y Silva Delgado, Camilo Andrés. 2010. Guía Metodológica. Levantamiento y análisis de Requerimientos de Software con base en procesos de negocio. [En línea] 2010. [Citado el: 15 de enero de 2017.] http://pegasus.javeriana.edu.co/~CIS1010IS06/descargas/Anexos/Marco%20Teorico/Anexo%205.%20Gu%C3%ADa%20Metodol%C3%B3gica%20para%20el%20Levantamiento%20y%20An%C3%A1lisis%20de%20Requerimientos%20de%20Software%20en%20base%20a%20Procesos%20de%20Negocio.pdf..

Maza, M.Á.Sánchez. 2012. *JavaScript.* s.l. : IC Editorial, 2012. ISBN: 978-849-5733184..

Méndez, A. birruetas. 2010. Metodologías de desarrollo de software. *Instituto Tecnológico Superior de Apatzingán.* [En línea] 2010. [Citado el: 2016 de Diciembre de 13.] http://www.itsapatzingan.edu.mx.

Mora, Juan Tahuiton. 2011. *Arquitectura de software para aplicaciones web.* México DF : Centro de investigación y de estudios avanzados del instituto politécnico nacionl, 2011.

Muñoz, Vicente Javier Ortega. Direccion Academica. [En línea] [Citado el: 8 de Diciembre de 2016.] http://diracademica.manizales.unal.edu.co/index.php/procesos-liderados/sistema-de-informacion-academica-sia.

2015. Netbeans. *Netbeans.* [En línea] 2015. [Citado el: 18 de 1 de 2017.] http://netbeans.org.

Nieto, Narciso García. 2005. *La tutoría universitaria ante el proceso de armonización europea.* s.l. : Revista de Educación Superior, 2005.

Pacheco, N. 2011. *Manual de Symfony 2.* 2011.

Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. **C.A Guerrero, Johanna M.Suárez y Luz E. Gutiérrez. 2013.** 3, La Serena.Colombia : Información Tecnológica, 2013, Vol. 24. ISSN 0718-0764.

2011. *Patrones GoF.* Colombia : s.n., 2011.

Paz, Pavel E. Barzaga de la. 2016. *Componente para la transformación y.* La Habana : Universidad de las Ciencias Informáticas, 2016.

2005. PBWorks. [En línea] 13 de Julio de 2005. [Citado el: 10 de Marzo de 2017.] http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos.

Pérez, J. Eguíluz. 2014. *Desarrollo web ágil con Symfony 2.* 2014.

—. 2010. *Introducción a CSS.* 2010.

—. 2010. *Introducción a JavaScrript.* 2010.

—. 2010. *Introducción a XHTML.* 2010.

Pérez, Master en Ciencias Sergio Luis Pérez. *Fundamentos de Ingeniería de Software(Patrones de Diseño).* Cuajimalpa. Mexico D.F : s.n.

Pressman, R.S. 2010. *Software engineering: a practitioner's approach.* New York,EUA : McGraw-Hil, 2010. ISBN 978-0-07-337597.

Pressman, Roger S. 2005. *Ingeniería del Software. Un enfoque práctico.* La Habana : Felix Varela, 2005.

Revista Cubana de Informática Médica. **Diana Margarita Rey Kaba, Lilia Ester Rodréguez Chávez. 2011.** 2, La Habana : RCMI, 2011, Vol. 3.

Saco, G. González-Vallés. 2014. *Una Introducción a APACHE.* 2014.

Sánchez, T. Rodríguez. 2014. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana.Cuba : Universidad de las Ciencias Informáticas, 2014.

—. 2014. *Metodología de desarrollo para la actividad productiva en la UCI.* 2014.

Sommerville, I. 2011. *Ingeniería de software.* España : Pearson, 2011. ISBN: 9786073206044.

Superior, Miniserio de Educación. 2014. *Plan de Estudios ¨D¨ Ingeniería en Ciencias Informáticas.* La Habana : UCI, 2014.

Superior, Subsecretaría de Educación Superior.Dirección General de Educación. 2012. *Práctica Profesional.* s.l. : Secretaría de educación pública, 2012.

Trasobares, Alejandro Hernández. 2015. *Los sistemas de información: Evolucion y desarrollo.* Zaragoza : s.n., 2015.

UCI. 2015. Sistema de Gestion Universitaria. [En línea] 2015. [Citado el: 9 de Diciembre de 2016.] https://akademos.uci.cu.

UNAD. 2015. Lección 29. Diagramas de componentes. [En línea] 2015. [Citado el: 2 de abril de 2017.] http://datateca.unad.edu.co/contenidos/200609/exeuml/leccin_29_diagramas_de_componentes.html.

UNcuyo. *Facultad de Filosofía y Letras.* [En línea] [Citado el: 9 de Diciembre de 2016.] http://ffyl.uncuyo.edu.ar/upload/instructivo-guarani.pdf.

UNCuyo, CICUNC. [En línea] [Citado el: 8 de Diciembre de 2016.] http://ffyl.uncuyo.edu.ar/que-es-y-para-que-sirve-el-siu-guarani.

Usaola, Macario Polo. 2012. Patrones GRASP. [En línea] 2012. [Citado el: 23 de abril de 2017.] http://www.inf-cr.uclm.es/www/mpolo/asig/0304/0102/patronesgrasp.pdf.

Valenciano, Jaime de Pablo. *La elaborción de un plan de formación e innovación.*

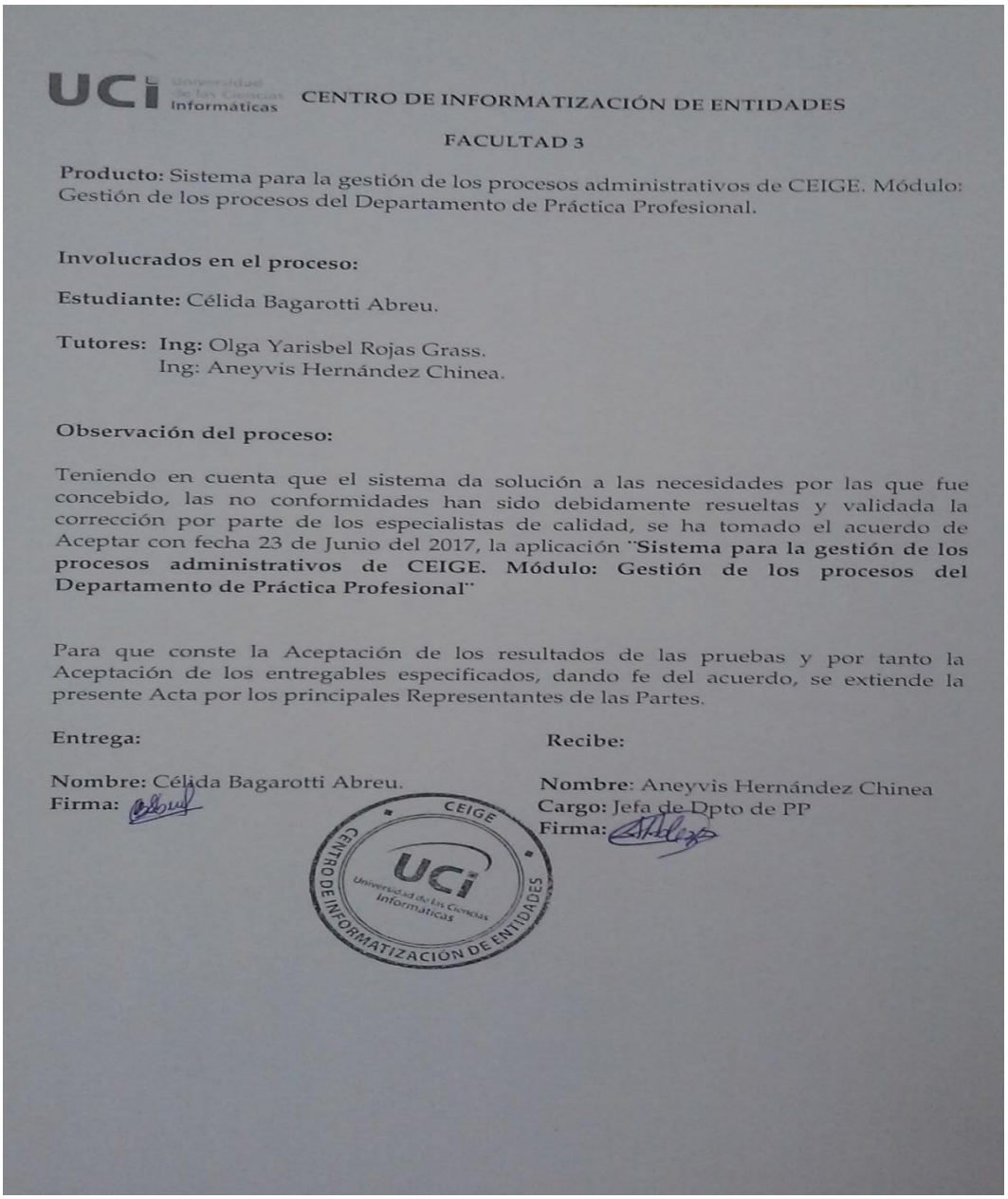
Valladares, Dra Adalia Lisett Rojas. 2016. *Apuntes sobre el trabajo de tutoría en la formación del profesional en la educación superior.* República Dominicana : s.n., 2016. SSN 2308-1953.

Venete, A. 2011. Introducción a los Patrones de Arquitectura. [En línea] 2011. [Citado el: 8 de abril de 2017.] http://mahara.uji.es/artefact/file/download.php?file=54534&view=4648..

W3C. [En línea] [Citado el: 15 de 1 de 2017.] http://www.w3c.es/.

Yunaysy Ortiz Batista. 2011. Repositorio Digital. [En línea] Universidad de la Ciencias Informáticas,La Habana,Cuba, 2011. https://repositorio_institucional.uci.cu/jspui/bitstream/ident/8016/3/TM_04853_11.pdf.

Anexo 1. Acta de aceptación del cliente.



UCI Universidad de las Ciencias Informáticas

CENTRO DE INFORMATIZACIÓN DE ENTIDADES

FACULTAD 3

Producto: Sistema para la gestión de los procesos administrativos de CEIGE. Módulo: Gestión de los procesos del Departamento de Práctica Profesional.

Involucrados en el proceso:

Estudiante: Célida Bagarotti Abreu.

Tutores: **Ing:** Olga Yarisbel Rojas Grass.
Ing: Aneyvis Hernández Chinea.

Observación del proceso:

Teniendo en cuenta que el sistema da solución a las necesidades por las que fue concebido, las no conformidades han sido debidamente resueltas y validada la corrección por parte de los especialistas de calidad, se ha tomado el acuerdo de Aceptar con fecha 23 de Junio del 2017, la aplicación "**Sistema para la gestión de los procesos administrativos de CEIGE. Módulo: Gestión de los procesos del Departamento de Práctica Profesional**"

Para que conste la Aceptación de los resultados de las pruebas y por tanto la Aceptación de los entregables especificados, dando fe del acuerdo, se extiende la presente Acta por los principales Representantes de las Partes.

Entrega:

Nombre: Célida Bagarotti Abreu.
Firma:

Recibe:

Nombre: Aneyvis Hernández Chinea
Cargo: Jefa de Dpto de PP
Firma:

CEIGE
UCI Universidad de las Ciencias Informáticas
CENTRO DE INFORMATIZACIÓN DE ENTIDADES

Anexo 2. Interfaz de usuario de Modificar estudiante.