Universidad de las Ciencias Informáticas

Facultad 15

Título: Desarrollo de un sistema para la gestión de evaluaciones docentes en la Disciplina de Programación

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Aracelis Ruiz Morrisel

Tutor: Ing. José Antonio Sánchez Imbert

Junio del 2010

¡Cuando un pueblo enérgico y viril llora, la injusticia tiembla!

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaro ser la 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 _____.

Aracelis Ruiz Morrisel

_____

Firma del Autor

Ing. José Antonio Sánchez Imbert

_____

Firma del Tutor

DATOS DE CONTACTO

José Antonio Sánchez Imbert: Ingeniero en Ciencias Informáticas.

Un año de experiencia

Adiestrado

Analista del proyecto SAGEB

Profesor de Introducción a la Programación

AGRADECIMIENTOS

Agradecerle a la Revolución por darme la posibilidad de estudiar y a Fidel por ser el gestor de esta maravillosa obra.

Al claustro de profesores de la UCI por haberme guiado paso a paso no solo en los conocimientos de la especialidad sino también a formarme como una joven integral.

A Ricardo por ayudarme a realizar esta tesis y por ser mi amigo. Sin ti no hubiera sido posible realizar esta tesis. Siempre te estaré agradecida.

A mi madre querida por guiarme, ayudarme y apoyarme siempre cuando más lo necesité y seguirme en todas las decisiones tomadas.

A mi familia por apoyarme en los momentos más difíciles y confiar en mí.

A la persona que ha estado a mi lado durante estos 5 largos años a Nela quien me ha guiado por el buen camino.

A mis amistades por estar siempre ahí en los malos y buenos momentos en esta gran casa.

DEDICATORIA

Dedico esta tesis:

A mi mamá quien ha sido mi ejemplo, mi guía, mi vida, mi razón de ser, gracias por existir y darme todo lo que soy.

A mi abuela Rubina quien me ha apoyado siempre. Te quiero de corazón.

A Nela por ser tan paciente y darme durante estos 5 años todo el apoyo y cariño.

A mi familia por ser tan especial y apoyarme siempre.

A mi hermana Ana Ivis.

A mi novio Yohan quien ha sido muy importante para mí. Te kieooooo

A mis amigos por ser tan especiales, a los nuevos, a los viejos, en fin a todos, muchas gracias.

En especial a mi hermano Rodolfo quien ha lidiado conmigo durante todo este tiempo, a José Carlos, a Walny,a Mayde, a Maydalis.

A todos gracias

RESUMEN

La gestión de evaluaciones es un proceso que puede ser realizado por diversas personas y de maneras muy variadas para obtener un resultado satisfactorio que garantice con eficiencia el éxito de la actividad. A través del control de los resultados docentes de los estudiantes se llevarán a cabo diversas diligencias, trámites, las cuales, conducirán al logro de un objetivo determinado, de un negocio o de un deseo que lleva largo tiempo en carpeta.

En la Universidad de las Ciencias Informáticas este proceso se realiza manualmente por los profesores o personas autorizadas a realizar dicha tarea pero no se tiene una aplicación que permita desarrollar el mismo detalladamente teniendo en cuenta todas las evaluaciones de los implicados, no sólo registrar las notas de los trabajos de controles sino en sentido general.

Este trabajo tiene como objetivo realizar una aplicación que le permita a los profesores gestionar todo el trabajo referente a las evaluaciones docentes de los estudiantes durante todo el curso en la Disciplina de Programación , llevando un control de las notas obtenidas, ya sean preguntas escritas, pruebas, seminarios, proyectos, tareas extra-clases, evaluaciones en el aula, criterio del profesor, que les permita emitir resúmenes del desempeño de sus alumnos en cuanto a resultados docentes, en fin todo el quehacer estudiantil.

Para el desarrollo de la aplicación se utilizará la herramienta Visual Paradigm, se hará uso de la metodología RUP, como gestor de base datos PostgreSQL con el fin de darle cumplimiento al objetivo propuesto.

Palabras Claves: Visual Paradigm, PostgreSQL, RUP, Symfony, Modelo Vista Controlador.

TABLA DE CONTENIDOS

Tabla de contenidos

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La Universidad de las Ciencias Informáticas, como cualquier otra universidad cubana, ha de valorar periódicamente en qué medida su actividad desarrolla y alcanza los objetivos y fines que se propone. Tal evaluación deberá suministrar la información necesaria para programar cambios y líneas de actuación que conduzcan a la mejora de su actividad docente, siendo esta finalidad, la de asegurar y mejorar la calidad de la actividad universitaria, el objetivo primordial que debe informar todo el proceso.

La necesidad de gestionar una determinada actividad o evento ha estado presente desde que surgió la humanidad. Esta se evidenció en la comunidad primitiva cuando sintieron la necesidad de buscar alimentos para la supervivencia. De este modo, todos luchando por un mismo fin fueron capaces de alimentarse, vincularse a la naturaleza y transformarla en su beneficio propio. Con el transcurso del tiempo el hombre fue evolucionando y le surgieron nuevas necesidades. Se iba desarrollando por lo que aumentaban sus conocimientos y preocupaciones. Cada vez adquiría más información sobre el mundo que lo rodeaba y debía manipularla de manera tal que le fuese útil.

En nuestros días el valor de la información se ha incrementado a medida que pasa el tiempo, de ella depende el desarrollo de cualquier entidad. Por lo que se hace necesario tener un control exquisito de la misma y gestionarla de la mejor manera posible. Los grandes volúmenes de información existentes deben ser almacenados, controlados y gestionados con el fin de posibilitar conocimientos a las personas que accedan a la información.

La Universidad de las Ciencias Informáticas que surgió al calor de la batalla de ideas como una obra de la Revolución Cubana, ha jugado un papel imprescindible en el quehacer de la gestión de la información. La informática en Cuba se encuentra en una etapa de avance y desarrollo que está inmersa en constantes cambios y transformaciones. La UCI está en la cúspide de dicha evolución debido a su perfil informático, manipulando así gran cantidad de información de forma automática.

El control de evaluaciones docentes es una actividad que se realiza desde el surgimiento de la enseñanza como elemento de verificación a lo largo de todo el proceso. La gestión de evaluaciones no es más que crear condiciones, construye escenarios adecuados, provee capacidades e instrumentos a los equipos de trabajo: Se sitúa en el terreno de la creación, el conflicto y las tensiones que constituyen el campo de la gestión. No hay un modelo único, ni una práctica única para la construcción diaria de la gestión. Existe una diversidad de variantes que pueden llevar a un mismo resultado mientras se garantice el despliegue de la creatividad y se permita que el hacer y sus logros se inserten, en el sentido del resultado esperado (Blejmar, 2005)

Introducción

La gestión de evaluaciones es un proceso que surgió con el objetivo de obtener un control del aprendizaje de los estudiantes. Apunta a los resultados dados por verificaciones a corto plazo (evaluaciones orales) o a largo plazo (pruebas escritas, estudios independientes). Es la vía que utiliza el profesor o personal autorizado para validar el nivel de conocimiento alcanzado por los estudiantes en cuestión.

En la UCI se realiza el proceso de gestión de las evaluaciones docentes a través de los profesores y así les posibilita a los estudiantes obtener sus resultados docentes. Este proceso lo realizan los trabajadores de forma directa cada día al impartir sus clases. Esta actividad tiene gran importancia para todos los involucrados. Es una vía de saber sus aspiraciones futuras y su proceso de mejora en cuanto a los resultados. Previamente una vez que tienen la información pueden tratar de mejorar los mismos.

Para los profesores realizar esta acción cada día resulta más agobiante. Debido a que deben estar constantemente entregando informes de los resultados de sus estudiantes en las diferentes evaluaciones a sus departamentos. Desarrollar resúmenes de las evaluaciones realizadas, tener un control sobre el estado de cada uno de sus alumnos, lo que puede traer consigo errores y pérdida de tiempo.

La Disciplina de Programación es realmente difícil para la mayoría de los estudiantes. Sus métodos de evaluación resultan complicados, presentan un mecanismo pre-establecido a través de fórmulas matemáticas que calculan dichas evaluaciones. Para el cálculo de las mismas recoge los resultados de todos los temas, sus respectivas evaluaciones y mediante dicha ecuación realizan los cálculos necesarios para tener un promedio de la nota final del evaluado. Sin embargo esta actividad no es tan sencilla como parece, debido a que los profesores tardan semanas en la realización del proceso.

No cuentan con una herramienta que les garantice recopilar las evaluaciones de los alumnos. La gestión de evaluaciones en la Disciplina se realiza de manera continua por lo que llevar un control diario de las mismas es bastante complicado. Para ello se utiliza el registro y un documento Excel con el desempeño de cada estudiante. Esto le genera al profesor dificultades a la hora de emitir un criterio sobre un determinado estudiante, pues debe buscar dónde se encuentra la información y revisarla para luego arribar a una conclusión. El educador se desgasta muchísimo y pierde tiempo en el proceso. La pérdida de información se evidencia en dicho proceso. Las evaluaciones son orientadas por vías informales tales como correos, carpetas compartidas y en ocasiones la información no le llega a todos los involucrados.

Introducción

Existe la falta de comunicación entre profesor y estudiantes pues no existe un sendero factible que permita dicha interacción. Las tareas desarrolladas por los alumnos deben ser entregadas al pedagogo mediante el correo o carpetas compartidas, quitándole espacio ya sea al buzón del destinatario como a la máquina de la persona en materia. En caso de existir problemas de conexión o el correo del profesor se encuentre lleno el estudiante pierde dicha evaluación.

En encuestas realizadas a profesores de la antigua facultad 4 de la Disciplina de Programación hacen referencia a la necesidad de una aplicación que les permita realizar la gestión de las evaluaciones docentes en detalle para esta disciplina. Cada profesor independiente realiza este proceso de diversas maneras, por lo que se hace muy difícil tener un control exquisito de cada una de las actividades realizadas cotidianamente en sus respectivas aulas. Todos concuerdan en la necesidad de una variante que les garantice controlar las evaluaciones docentes realizadas durante todo el curso, tales como: preguntas escritas, seminarios, trabajos de controles entre otras actividades.

Existen aplicaciones en nuestra universidad con funcionalidades similares pero no garantizan con exactitud la gestión de dichas evaluaciones.

Por lo antes expuesto tenemos como **Problema a resolver**: ¿Cómo gestionar las evaluaciones docentes en la Disciplina de Programación?

Teniendo en cuenta el problema antes mencionado se tiene como **Objeto de estudio:** Proceso de gestión de las evaluaciones docentes. Como **Campo de acción**: Gestión de evaluaciones docentes en la Disciplina de Programación en la Universidad de las Ciencias Informáticas.

Para darle solución al problema planteado se expone el siguiente **Objetivo general**: Desarrollar un sistema que facilite la gestión de evaluaciones docentes en la Disciplina de Programación en la Universidad de las Ciencias Informáticas.

Para darle cumplimiento al objetivo propuesto se realizarán **Las tareas** siguientes:

- Estudio de la metodología RUP.
- Estudio de la herramienta Visual Paradigm.
- Especificación y validación de los requisitos del sistema.
- Realización del diseño del sistema.
- Implementación del sistema.
- Prueba del sistema.

Introducción

El presente documento está compuesto por 4 capítulos:

En el **Capítulo 1 Fundamentación Teórica:** se abordarán los principales conceptos, definiciones y aspectos importantes en dicho trabajo. Se hará referencia al estado del arte de las funcionalidades del sistema que ofrecen en la Universidad y se abordará la ingeniería de requerimientos y las diferentes técnicas utilizadas para una correcta utilización de los requisitos funcionales de la aplicación. Se tratarán las herramientas CASE y la metodología a utilizar para dicho proceso. Se explicarán brevemente los patrones utilizados.

En el **Capítulo 2 Características del sistema:** se hará una caracterización detallada sobre la solución propuesta así como sus objetivos principales. Se realizará una breve descripción sobre los requisitos funcionales de la aplicación .además de definir los requerimientos no funcionales que representará el sistema.

En el **Capítulo 3 Diseño:** se modelarán las clases del diseño que se reflejan en el problema planteado de acuerdo a sus funcionalidades partiendo de los requisitos funcionales.

En el **Capítulo 4 Implementación y Prueba:** se implementarán a partir de las características del sistema las funcionalidades del mismo, así como los casos de uso más significativos para darle solución al problema planteado. Además se validará la implementación para comprobar si cumple con todas las funcionalidades que debe cumplir el sistema a desarrollar de acuerdo a los requisitos propuestos.

El documento cuenta además con las **Conclusiones** del trabajo, algunas **Recomendaciones** a tener en cuenta para el mejoramiento del sistema en dependencia de las necesidades que se presenten y que sea utilizada por los profesores que impartan la disciplina, las **Bibliografías,** un **Glosario de Términos** y un conjunto de **Anexos** un conjunto de pruebas que garantizan el correcto funcionamiento del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

INTRODUCCIÓN

El presente capítulo tratará acerca del estado del arte de la investigación, los conceptos más significativos, así como las herramientas CASE y las metodologías utilizadas para el desarrollo de la aplicación dándole cumplimiento a la problemática a planteada.

1. Evaluaciones

Una evaluación se le conceptúa normalmente como un juicio sistemático sobre el valor o mérito de algo: (RETO, 2008)

a) Por “algo” puede entenderse un proceso, una tecnología, un programa, etc. En nuestro caso, este “algo” es el desempeño docente, de manera que hablamos de un juicio integral sobre el valor o mérito del desempeño de la función docente.

b) “Sistemático” significa que no es un mero juicio espontáneo o intuitivo, sino, un ejercicio planificado/organizado que comporta el recurso a algún tipo de método.

Puede conceptualizarse como un proceso dinámico, continuo y sistemático, enfocado hacia los cambios de las conductas y rendimientos, mediante el cual verificamos los logros adquiridos en función de los objetivos propuestos. (Maccario)

La evaluación es una operación sistemática, integrada en la actividad educativa con el objetivo de conseguir su mejoramiento continuo, mediante el conocimiento lo más exacto posible del alumno en todos los aspectos de su personalidad, aportando una información ajustada sobre el proceso mismo y sobre todos los factores personales y ambientales que en esta inciden. Señala en qué medida el proceso educativo logra sus objetivos fundamentales y confronta los fijados con los realmente alcanzados. (Pila Teleña)

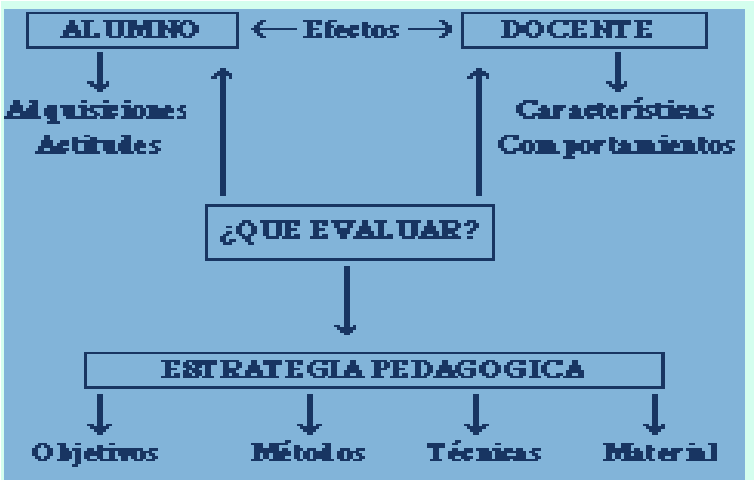

Figura 1. Diagrama de Evaluación

Evaluación implica comparación entre los objetivos impuestos a una actividad intencional y los resultados que produce. Es preciso evaluar no solamente los resultados, sino los objetivos, las condiciones, los medios, el sistema pedagógico y los diferentes medios de su puesta en acción. (Stufflebeam, 2002)

Esto supone:

Evaluación del contexto, determinar los objetivos, sus posibilidades, sus condiciones y medios de realización, lo que nos será de fundamental importancia al momento de elaborar la planificación.

Evaluación de las necesidades inherentes al proyecto (input), o sea la determinación de la puesta en práctica, de los recursos y de los medios.

Evaluación del proceso, estudio de los datos sobre los efectos que produjeron los métodos empleados, su progresión, sus dificultades y su comparación para tomar decisiones de ejecución.

Evaluación del producto, medición, interpretación, juicio acerca del cumplimiento de los objetivos, de la eficacia de la enseñanza, insuma evaluación de los resultados para tomar decisiones de reciclaje. (Stufflebeam, 2002)

1.1. Categorías de la Evaluación Docente (R. Tyler)

La Evaluación Predictiva o Inicial (Diagnóstica), se realiza para predecir un rendimiento o para determinar el nivel de aptitud previo al proceso educativo. Busca determinar cuáles son las características del alumno previo al desarrollo del programa, con el objetivo de ubicarlo en su nivel, clasificarlo y adecuar individualmente el nivel de partida del proceso educativo.

La Evaluación Formativa, es aquella que se realiza al finalizar cada tarea de aprendizaje y tiene por objetivo informar los logros obtenidos, y eventualmente, advertir dónde y en qué nivel existen dificultades de aprendizaje, permitiendo la búsqueda de nuevas estrategias educativas más exitosas. Aporta una retroalimentación permanente al desarrollo del programa educativo.

La Evaluación Sumativa, es aquella que tiene la estructura de un balance, realizada después de un período de aprendizaje en la finalización de un programa o curso.

1.2 Objetivos de las Evaluaciones Docentes

Calificar en función de un rendimiento, otorgar una certificación, determinar e informar sobre el nivel alcanzado a todos los niveles (alumnos, padres, institución, docentes, etc.).

1.3 Funciones de las Evaluaciones Docentes

- **Control:** permite controlar la presencia en el sistema y la superación de sus dispositivos de garantía.
- **Selección:** el sistema educativo va dejando fuera a quienes no superan las pruebas y va eligiendo a quienes son capaces de superarlas.
- **Comprobación:** permite saber si se han conseguido los objetivos propuestos, según una escala de valoraciones. La superación de las pruebas sirve de garantía social.
- **Clasificación:** como la evaluación tiene un referente doble (con los mínimos y con los demás estudiantes), los resultados permiten clasificar a los alumnos.
- **Acreditación:** la superación de los controles de la evaluación conduce a la acreditación económica y social. Esa acreditación tiene también una escala.
- **Jerarquización:** la evaluación encierra poder porque quien evalúa impone criterios, aplica pruebas y decide cuáles han de ser las pautas de corrección. Puede, incluso, negarse a compartirlas y discutirlas con los alumnos.
- **Diagnóstico:** la evaluación entendida como un proceso de análisis permite conocer cuáles son las ideas de los alumnos, los errores en los que tropiezan, las principales dificultades con las que se encuentran, los logros más importantes que han alcanzado.
- **Diálogo:** la evaluación puede convertirse en una plataforma de debate sobre la enseñanza.
- **Comprensión:** la evaluación es un fenómeno que facilita la comprensión de lo que sucede en el proceso de enseñanza y aprendizaje.

- **Retroalimentación:** la evaluación permite la reorientación del proceso de enseñanza y aprendizaje. No sólo en lo que se refiere al trabajo de los alumnos sino a la planificación de la enseñanza, a la modificación del contexto o a la manera de trabajar de los profesionales.
- **Aprendizaje:** la evaluación permite al profesor saber si es adecuada la metodología, si los contenidos son pertinentes, si el aprendizaje que se ha producido es significativo y relevante para los alumnos.
- **Formativa:** Se orienta a recoger datos del proceso de enseñanza-aprendizaje, con el objetivo de mejorarlo. A diferencia del modelo anterior, no es retrospectiva sino prospectiva, en tanto su preocupación se dirige a mejorar lo que queda por realizar. Si bien no desconoce a la evaluación Sumativa, lo que la distingue del modelo anterior, es el peso que se le da a este tipo de evaluación.

2. En el mundo

La gestión de evaluaciones docentes es una actividad que desde sus inicios ha sido realizada por los profesores. Estos se han encargado de desarrollarla de la manera más amena posible para tener un control de sus estudiantes. Con el paso del tiempo el volumen de evaluaciones aumentaba por lo que se hizo necesario la creación de aplicaciones que permitieran llevar un registro de estas. Para el desarrollo del control de evaluaciones las aplicaciones más utilizadas históricamente han sido las plataformas Moodle.

2.1. Moodle

¿Qué es Moodle? Modular Object-Oriented Dynamic Learning Environment (Entorno de Aprendizaje Dinámico Orientado a Objetos y Modular), lo que resulta fundamentalmente útil para programadores y profesores. Moodle es un conjunto de programas para la creación de cursos y sitios Web basados en Internet.

Fue creado por Martin Dougiamas, quien fue administrador de WebCT en la Universidad Tecnológica de Curtin. Basó su diseño en las ideas del constructivismo en pedagogía que afirman que el conocimiento se construye en la mente del estudiante en lugar de ser transmitido sin cambios a partir de libros o enseñanzas y en el aprendizaje colaborativo. Un profesor que opera desde este punto de vista crea un ambiente centrado en el estudiante que le ayuda a construir ese conocimiento con base en sus habilidades y conocimientos propios en lugar de simplemente publicar y transmitir la información que se considera que los estudiantes deben conocer.

1.1.1 Ventajas

Una de las características más atractivas de Moodle, que también aparece en otros gestores de contenido educativo, es la posibilidad de que los alumnos participen en la creación de glosarios, y en todas las lecciones se generan automáticamente enlaces a las palabras incluidas en estos.

Además, las Universidades podrán poner su Moodle local y así poder crear sus plataformas para cursos específicos en la misma universidad y dando la dirección respecto a Moodle, se moverá en su mismo idioma y podrán abrirse los cursos a los alumnos que se encuentren en cualquier parte del planeta.

1.1.2 Desventajas

Algunas actividades pueden ser un poco mecánicas, dependiendo mucho del diseño de la instrucción. Por estar basado en tecnología PHP, la configuración de un servidor con muchos usuarios debe ser cuidadosa para obtener el mejor desempeño. Mejorar su interfaz de una manera más sencilla. Hay desventajas asociadas a la seguridad, dependiendo en dónde se esté alojando la instalación de Moodle y cuáles sean las políticas de seguridad y la infraestructura tecnológica con la cual se cuente durante la instalación.

Existen también desventajas relacionadas con el soporte técnico. Al ser una plataforma de tecnología abierta y por lo tanto gratuita, no se incluyen servicios gratuitos de soporte, por lo que los costos de consultoría y soporte técnico están sujetos a firmas y entidades externas.

1.2 EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Para la gestión académica en la Universidad de las Ciencias Informáticas (UCI) se utilizan dos aplicaciones muy conocidas por todos en el centro. Ellas aportan informaciones referentes al desempeño docente de cada estudiante en cuanto a sus notas durante la carrera. Estas aplicaciones no realizan este proceso de manera detallada teniendo en cuenta toda la actividad evaluativa que se realiza en la Disciplina de Programación. El control de los resultados de los alumnos se hace mediante documentos Excel en los cuales se registran cada una de estas evaluaciones quitándoles tiempo a las personas que realizan este proceso. Muestra de estas aplicaciones tenemos el Entorno Virtual del Aprendizaje (EVA) y Akademos de las cuales se hará referencia a continuación.

1.2.1 EVA

El sitio Web *http://eva.uci.cu* del Entorno Virtual de Aprendizaje (EVA) de la Universidad de las Ciencias Informáticas (UCI) pone a disposición de la comunidad universitaria diversos servicios donde se podrá impartir o recibir formación tanto de pregrado como de postgrado haciendo un uso intensivo de las nuevas Tecnologías de la Información y las Comunicaciones. Es un espacio de apoyo al proceso de formación de la carrera de Ingeniería en Ciencias Informáticas, donde los profesores pueden implementar estrategias de enseñanza-aprendizaje complementarias a las clases presenciales, así como diseñar cursos semi-presenciales o totalmente a distancia, disponiendo los estudiantes de un poderoso medio en el cual pueden obtener, utilizar o compartir materiales didácticos. El mismo permite entre otras cuestiones:

- La resolución por parte del alumno, de las diversas actividades planteadas por los docentes y tutores.
- La consulta a través de chat, diálogos, foros, etc. sobre cualquiera de los temas que comprenden a las unidades del curso.
- El intercambio de ideas, aprendizajes y experiencias con compañeros que accedan al mismo curso.

Resumiendo entonces, el EVA pretende lograr una mayor comunicación entre quienes cursen la carrera de Ingeniería en Ciencias Informáticas en la UCI y el profesorado responsable, para que el proceso de enseñanza-aprendizaje se logre con la mayor eficacia posible. Dicha aplicación está encaminada a desarrollar una comunicación entre profesores y estudiantes.

1.2.2 AKADEMOS

El sistema de gestión académica de nuestra universidad, Akademos, pone a disposición este nuevo módulo Akademos 2006 desarrollado a partir de la necesidad de conocer las calificaciones obtenidas por un estudiante durante la carrera. Para su confección se tuvieron en cuenta las opiniones de estudiantes, profesores y cuadros de dirección de las facultades de la UCI.

A continuación se enumeran algunos aspectos que por su importancia se muestran detalladamente.

- El grupo de desarrollo de Akademos no se hace responsable por los posibles errores en los datos ya que la información que aquí se muestra es introducida en el sistema por los profesores y el personal de secretaría de cada facultad. En caso de errores debe dirigirse a la Secretaría Docente de la misma.
- El profesor podrá ver el registro de asistencias de sus estudiantes a clases, consultarlo y modificarlo.
- Cada profesor podrá consultar el registro de evaluaciones de sus estudiantes, así como las de cualquier otro. Además de poder registrarlas y modificarlas cuando lo necesite.

El profesor podrá filtrar las notas por años y semestres, así cómo conocer las evaluaciones parciales de las asignaturas que el estudiante está cursando en el semestre actual. Usando el vínculo en la parte superior usted podrá imprimir los cuadros de notas. Este sistema no es del todo eficiente en la gestión de evaluaciones docentes en la Disciplina de Programación debido a que no les permite a los profesores obtener resúmenes de evaluaciones, es decir reportes que contengan todo el desempeño docente.

3. Metodología de Desarrollo Utilizada

3.1 RUP

El Proceso Unificado de Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

- **Inicio**: el objetivo en esta etapa es determinar la visión del proyecto.

- **Elaboración:** en esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción**: en esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- **Transmisión**: el objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

Disciplina de Desarrollo

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente.

Disciplina de Soporte

- Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- Administrando el proyecto: Administrando horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.
- Distribución: Hacer todo lo necesario para la salida del proyecto

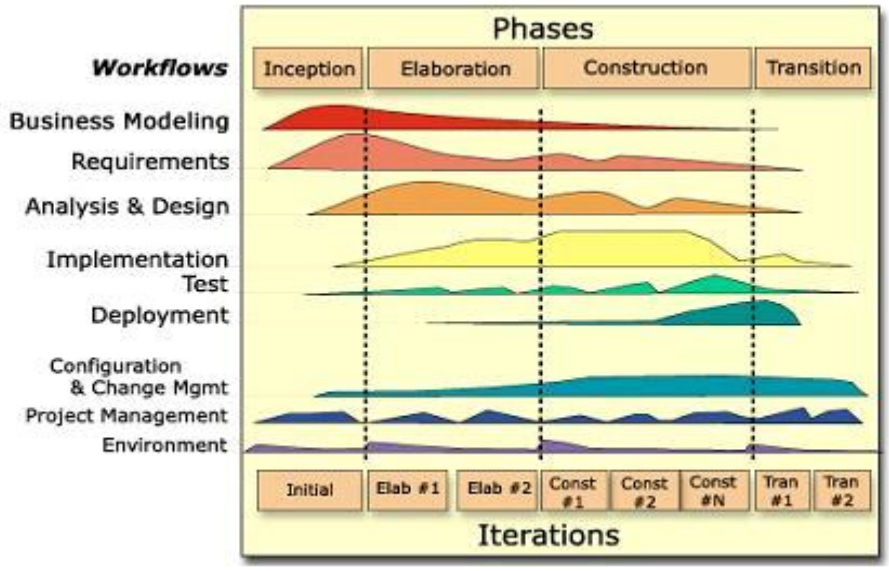

Figura 2 Fases e Iteraciones de la Metodología RUP

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

- **Actividades**: son los procesos que se llegan a determinar en cada iteración.
- **Trabajadores**: vienen hacer las personas o entes involucrados en cada proceso.
- **Artefactos**: un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Características de RUP

- **Dirigido por los casos de uso**: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso, es decir, cómo se llevan a cabo.

- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. Esta no sólo incluye las necesidades de los usuarios e inversores, sino también otros aspectos técnicos como el hardware, sistema operativo, sistema de gestión de base de datos, protocolos de red; con los que debe coexistir el sistema. RUP se desarrolla mediante iteraciones, comenzando por los casos de usos relevantes hasta llegar a un equilibrio entre funcionalidad y características técnicas desde el punto de vista de la arquitectura.
- **Iterativo e incremental: RUP** propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. La alta complejidad de los sistemas actuales hace que sea factible dividir el proceso de desarrollo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración y pueden o no representar un incremento en el grado de terminación del producto completo. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. En cada iteración los desarrolladores seleccionan un grupo de casos de uso, los cuales se diseñan, implementan y prueban. La planificación de iteraciones hace que se reduzcan los riesgos de los costes de un sólo incremento, no sacar al mercado un producto en el tiempo previsto, mantener la motivación del equipo pues puede ver avances claros a corto plazo y que el desarrollo pueda adaptarse a los cambios en los requisitos.

3.2. Lenguaje Unificado de Modelado

Por sus siglas en inglés, Unified Modeling Language (UML) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

En UML 2.0 hay 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, a veces es útil categorizarlos jerárquicamente.

Los **Diagramas de Estructura** enfatizan en los elementos que deben existir en el sistema modelado:

- Diagrama de clases
- Diagrama de componentes
- Diagrama de objetos
- Diagrama de estructura compuesta (UML 2.0)
- Diagrama de despliegue
- Diagrama de paquetes

Los **Diagramas de Comportamiento** enfatizan en lo que debe suceder en el sistema modelado:

- Diagrama de actividades
- Diagrama de casos de uso
- Diagrama de estados

Los **Diagramas de Interacción** son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia
- Diagrama de comunicación, que es una versión simplificada del Diagrama de colaboración (UML 1.x)
- Diagrama de tiempos (UML 2.0)
- Diagrama global de interacciones o Diagrama de vista de interacción (UML 2.0)

3.3. Patrones de diseño

- Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.
- Un patrón de diseño identifica: **clases**, **instancias**, **roles, colaboraciones** y la **distribución de responsabilidades**.

Un patrón de diseño es:

- Una solución estándar para un problema común de programación.
- Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- Un proyecto o estructura de implementación que logra una finalidad determinada.
- Un lenguaje de programación de alto nivel.
- Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- Conexiones entre componentes de programas.
- La forma de un diagrama de objeto o de un modelo de objeto.

Ventajas

- Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo.
- Por tanto, están basados en la recopilación del conocimiento de los expertos en desarrollo de software.
- Es una experiencia real, probada y que funciona. Es Historia y nos ayuda a no cometer los mismos errores.

El patrón Modelo Vista Controlador

Symfony está basado en un patrón clásico del diseño Web conocido como arquitectura modelo, vista, controlador (MVC), que está formado por tres niveles:

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

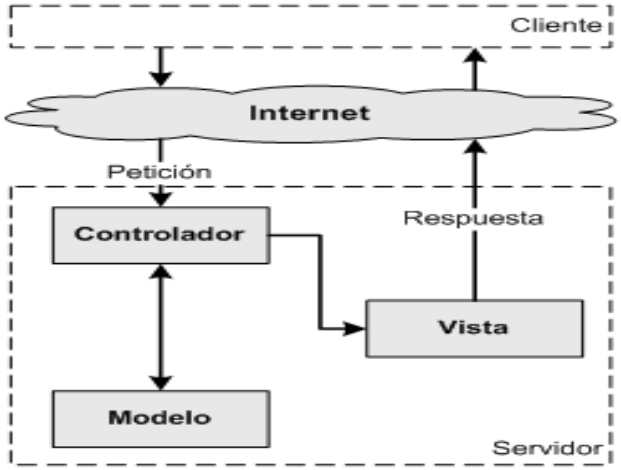

Figura 3 Modelo Vista Controlador

La arquitectura MVC proporciona grandes ventajas, como la organización del código, la reutilización, la flexibilidad y una programación mucho más entretenida.

Creador

En la clase Actions se encuentran las acciones definidas para la Gestión de Evaluaciones Docentes y se ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Actions es "creador" de dichas entidades.

Experto

Este es uno de los más utilizados, puesto que Doctrine es utilizada por Symfony para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Alta Cohesión

Symfony permite asignar responsabilidades con una alta cohesión, por ejemplo la clase Actions tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

Controlador

Todas las peticiones Web son manejadas por un sólo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Bajo Acoplamiento

La clase Action hereda solamente de sfActions para lograr un bajo acoplamiento de clases.

Singleton (Instancia única)

Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal hay una llamada a sfContext: getInstance(). En una acción, el método getContext(), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony

Abstract Factory (Fábrica abstracta)

Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre si y haciendo transparente el tipo de familia concreta que se está usando. Cuando el Framework necesita por ejemplo crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

Decorator (Envoltorio)

Añade funcionalidad a una clase, dinámicamente. El archivo layont.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla.

Composite (Objeto compuesto) Permite tratar objetos compuestos como si de un simple se tratase. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol .Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

3.4. Lenguajes

Es aquella actividad por la cual se crean programas para computadoras, tales programas pueden ser códigos fuentes interpretados (por ejemplo scripts en BASH) o códigos fuentes que serán compilados (como por ejemplo programas en C++) hacia lenguajes binarios y ejecutados desde el kernel del

sistema operativo. Es un proceso para convertir especificaciones generales de un sistema en instrucciones utilizables por la máquina, que produzcan los resultados deseados. Se le conoce también como desarrollo de software. (MELGAR VELIZ, 2009)

3.4.1. PHP 5

Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas.

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre. ()

3.4.2. Web 2.0

La Web 2.0 es la representación de la evolución de las aplicaciones tradicionales hacia aplicaciones Web enfocadas al usuario final. La Web 2.0 es una actitud y no precisamente una tecnología.

La Web 2.0 es la transición que se ha dado de aplicaciones tradicionales hacia aplicaciones que funcionan a través de la Web enfocada al usuario final. Se trata de aplicaciones que generen colaboración y de servicios que reemplacen las aplicaciones de escritorio.

4. Entorno de desarrollo y herramientas

4.1. Framework

Son diseñados con el objetivo de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

4.4.1. Symfony

Symfony es un completo Framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones Web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de

desarrollo de una aplicación Web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación Web. Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas (Unix, Linux, etc.) como en plataformas Windows.

4.4.2. Características de Symfony

Symfony se diseñó para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Unix estándares)
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la Web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Symfony puede ser completamente personalizado para cumplir con los requisitos de las empresas que disponen de sus propias políticas y reglas para la gestión de proyectos y la programación de aplicaciones. Por defecto incorpora varios entornos de desarrollo diferentes e incluye varias herramientas que permiten automatizar las tareas más comunes de la ingeniería del software:

- Las herramientas que generan automáticamente código han sido diseñadas para hacer prototipos de aplicaciones y para crear fácilmente la parte de gestión de las aplicaciones.

- El Framework de desarrollo de pruebas unitarias y funcionales proporciona las herramientas ideales para el desarrollo basado en pruebas (“test-driven development”).
- La barra de depuración Web simplifica la depuración de las aplicaciones, ya que muestra toda la información que los programadores necesitan sobre la página en la que están trabajando.
- La interfaz de línea de comandos automatiza la instalación de las aplicaciones entre servidores.
- Es posible realizar cambios de la configuración (sin necesidad de reiniciar el servidor).
- El completo sistema de log permite a los administradores acceder hasta el último detalle de las actividades que realiza la aplicación.

4.2. Gestor de Base Datos

4.2.1. PostgreSQL

Es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, Perl, Tcl y Python).

4.2.2. Características

- Alta concurrencia

Mediante un sistema denominado (MVCC Acceso Concurrente Multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

- Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)

- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

4.2.3. Funciones

Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientada a objetos o la programación funcional.

Los disparadores (triggers en inglés) son funciones enlazadas a operaciones sobre los datos.

Algunos de los lenguajes que se pueden usar son los siguientes:

- C.
- C++.
- Java PL/Java web.
- PL/Perl.
- plPHP.
- PL/Python.
- PL/Ruby.
- PL/sh.
- PL/Tcl.
- PL/Scheme.

PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta (query en inglés).

Las funciones pueden ser definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros DBMS, son muchas veces referidas como "procedimientos almacenados" (stored procedures en inglés).

4.2.4. Herramientas de administración

PgAdmin3

Entorno de escritorio visual.

PgAccess

Entorno de escritorio visual.

PhpPgAdmin

Entorno web.

PgSQL

Cliente de consola.

Database Master

Entorno de escritorio visual.

EMS SQL Manager for PostgreSQL

EMS SQL Manager para PostgreSQL es una herramienta de alto rendimiento para la administración y el desarrollo de bases de datos PostgreSQL. Funciona con cualquier versión de PostgreSQL hasta la 8.1 y soporta las últimas características de PostgreSQL incluyendo enumerados, búsqueda de texto, XML y tipos UUID de datos, el índice de PostgreSQL orden de clasificación clave, matrices de tipos de compuestos, y otros. SQL Manager para PostgreSQL ofrece herramientas, potentes base de datos tales como diseñador visual de base de datos para crear bases de datos PostgreSQL en pocos clics, Visual Query Builder para construir complejos de consultas de PostgreSQL, potente editor BLOB y muchas características más útiles para una eficiente administración de PostgreSQL. SQL Manager para PostgreSQL tiene una interfaz gráfica de usuario del estado de la técnica, con bien descritos asistente sistema, de modo claro en su uso que ni un principiante no debe confundirse con ella.

4.3. Herramientas CASE Utilizadas

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Sistema de software que intenta proporcionar ayuda automatizada a las actividades del proceso de software. Los sistemas CASE a menudo se utilizan como apoyo al método.

4.3.1. DBDesigner 4

DBDesigner 4 es un sistema de base de datos de diseño visual que integra el diseño de bases de datos, modelado, creación y mantenimiento en un único entorno sin fisuras. Combina características profesionales y una interfaz de usuario clara y sencilla para ofrecer la forma más eficiente para gestionar sus bases de datos.

Características de DBDesigner 4

DBDesigner 4 ofrece potentes funciones para crear un modelo visual de cualquier base de datos. A partir de un motor de ingeniería inversa para recuperar automáticamente un modelo de bases de datos existentes, herramienta de modelado extensa y editora a una función de sincronización que se le aplicarán cambios en el modelo de forma automática a la base de datos subyacente - que todo es parte de DBDesigner 4.

El modo de diseño frente a modo de consulta

DBDesigner 4 es compatible con dos interfaces de usuarios conmutables. El modo de diseño se utiliza para crear y mantener el modelo de bases de datos visuales. El modo de consulta se utiliza para trabajar con datos de la tabla y crear complejas instrucciones de consulta SQL para el uso en PHP u otro lenguaje de programación.

Plugins y de código abierto

Los modelos creados en DBDesigner 4 se almacenan en XML. Pueden ser modificados por los plugins de terceros que se lanzan desde DBDesigner 4 y otros productos de terceros. Porque DBDesigner 4 es un proyecto Open Source es fácil para los programas de desarrollo de nuevos plugins o ampliar DBDesigner 4 para adaptar las necesidades específicas.

4.3.2. Visual Paradigm

Es una herramienta CASE (Ingeniería de Software Asistida por Ordenador) profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Esta herramienta tiene una serie de características entre las que se destacan:

- Soporta la versión UML 2.1.
- Soporta Ingeniería Inversa.
- Soporta BPMN.

- Generación del modelo físico de datos.
- Documentación de la captura de requisitos.

4.4. Comparación de los dos ORMs más conocidos y utilizados

Los dos ORM para PHP más utilizados son sin lugar a dudas y como el título lo dice Doctrine y Propel.

Características:

- Ambos ORMs tienen bastantes características básicas similares, soportan cualquier operación usual en un CRUD (Create, Retrieve, Update and Delete), desde crear un nuevo registro o actualizar los registros existentes. Además, los dos pueden generar las clases PHP del modelo por vos, Propel basado en XML y Doctrine basado en YAML, y ambos soportan la generación de sus respectivos markups desde una base de datos existente. También, ambos soportan varios motores de bases de datos.
- Los dos soportan validación de datos en los modelos y relaciones entre modelos. Además, soportan herencia simple, aunque en Doctrine es conocida como una herencia concreta. Doctrine soporta otros dos tipos de herencia: Simple, donde todas las clases tienen las mismas columnas, y la herencia de agregación, donde almacena un valor adicional en la tabla, permitiendo la instanciación automática del tipo de modelo correcto cuando se realiza una consulta.
- Hasta ahora se ha evidenciado que ambos comparten casi todas las características, pero las siguientes son sólo características que Doctrine tiene.

Behaviors: Doctrine soporta la aplicación de varios "*behaviors*" a tus modelos, por ejemplo un modelo Timestampable automáticamente creará 2 columnas: *created_at* y *updated_at*, las cuales guardarán la fecha cuando un registro es creado y cuando un registro es actualizado respectivamente.

Searching: Doctrine tiene un motor de búsqueda *fulltext*.

Además, Doctrine soporta Data *fixtures* y migraciones, *caching, events, pagination, command line interface*... con lo que podríamos decir que en estas características avanzadas Doctrine supera a Propel.

Usabilidad

Documentación

Una de las cuestiones más importantes por supuesto es la documentación. Sin una buena documentación se hace difícil utilizar cualquier librería. Hasta el año pasado, la documentación de Propel era uno de sus principales problemas, y si bien han ido mejorándola aún le falta. Por el contrario, el equipo de Doctrine ha estado mejorando constantemente su ya superior documentación, y están trabajando en la creación de un libro. En cuanto a documentación es una clara victoria para Doctrine.

Usando las librerías

La primera tarea que tendrás con ambos ORMs será la de crear las clases del modelo. Doctrine te permite escribir un simple archivo YAML, o bien, código PHP si lo prefieres. Si usas YAML, Doctrine tiene algunos métodos que puedes llamar en tu propio código, o puedes descargar una interface de línea de comandos para construir tus modelos. La propuesta de Propel para crear modelos requiere escribir un XML, y además para construir tus modelos desde el XML, necesitas Phing. Personalmente, encuentro más difícil escribir XML que YAML, por eso prefiero escribir YAML.

Operaciones con la base de datos

Las operaciones básicas de un CRUD son bastante similares en ambos ORMs. Sin embargo, existe una gran diferencia en la manera en que las consultas más precisas son hechas.

Setear valores en las clases del modelo es un poco diferente para estos ORMs: Doctrine utiliza propiedades mágicas, mientras que Propel genera métodos para setear y obtener valores (setters y getters). Esta última característica le da a Propel la ventaja de la autocompletación en la mayoría de los IDE's (la última versión de NetBeans puede autocompletar las propiedades mágicas de Doctrine gracias al soporte de la anotación property de PHPDoc).

Como conclusión se puede decir que ambos ORMs son muy buenos, pero Doctrine es el mejor de los dos. Tiene mejor documentación, mejores características, y la comunidad está activa. Propel sigue creciendo, Doctrine crece mucho más rápido.

4.4.1. Doctrine

Es un **ORM para PHP** 5.2.3 y posterior. Además de todas las ventajas que conlleva un ORM, uno de sus puntos fuertes es su lenguaje **DQL** (Doctrine Query Language) inspirado en el HQL de Hibernate.

4.5. Conclusiones Parciales

Luego de un profundo estudio sobre las problemáticas existentes a la hora de realizar el proceso de gestión de evaluaciones docentes en la Disciplina de Programación en la Universidad de las Ciencias Informáticas se necesita tener en cuenta el uso de las TIC para hacer dicho proceso, mediante la utilización de las diferentes herramientas, metodologías y tecnologías que existen para lograr los objetivos propuestos. En el presente capítulo se realizó la fundamentación teórica para dar solución al problema planteado.

CAPÍTULO 2 CARACTERÍSTICAS DEL SISTEMA

INTRODUCCIÓN

En el presente capítulo se hará una descripción de la solución propuesta, así como sus características principales. Se realizará una comparación referente a la las aplicaciones existentes en nuestra universidad y la oferta en cuestión. Además se plantearán los objetivos de la aplicación y los procesos a automatizar.

1. Objetivos de la aplicación

La gestión de evaluaciones en la materia de Programación ha sido un proceso que desde el surgimiento de la Universidad de las Ciencias Informáticas está en constante transformación y cambios. Presenta una serie de objetivos a vencer los cuales cada día que pasan ocupan un grado de complejidad mayor, por lo que se hace necesario redefinir su procedimiento de enseñanza. Esto trae consigo que el control de sus resultados varíe en dependencia del programa de la disciplina. La solución propuesta tiene como objetivo desarrollar una aplicación que garantice con éxitos la gestión de evaluaciones docentes en la Disciplina de Programación a todos los profesores que imparten la misma. De manera tal que se pueda llevar un registro detallado de cada uno de los resultados obtenidos por los estudiantes. Poder en cualquier momento saber el estado de los evaluados y enviar resúmenes con el desenvolvimiento de los alumnos.

1.2 Propuesta del sistema

Se propone un sistema Web que garantice la gestión de las evaluaciones docentes en la Disciplina de Programación, que el profesor en cualquier momento tenga una control sobre el desempeño de cada estudiante. Se definió como patrón de diseño el estilo arquitectónico modelo-vista-controlador. El cual obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el framework. El código de la presentación se guarda en la vista, el código de manipulación de datos se guarda en el modelo y la lógica de procesamiento de las peticiones constituye el controlador. Aplicar este patrón resulta muy útil además de restrictivo. La aplicación ofrecerá la posibilidad de gestionar los resultados a partir de las tareas orientadas a los implicados. Las tareas serán asignadas a cada estudiante con un tiempo de entrega para desarrollarla, luego en dependencia de la respuesta enviada le será otorgada una evaluación que será registrada en el sistema.

1.3 Requerimientos

- La IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- Una representación documentada de una condición o capacidad.

1.4 Requerimientos Funcionales

Son capacidades o condiciones que el sistema debe cumplir.

1. Autenticarse
   1.1 Usuario
   1.2 Contraseña
2. Gestionar Profesores
   2.1 Insertar profesor
   2.2 Modificar profesor
   2.3 Eliminar profesor
3. Gestionar Usuarios
   3.1 Insertar usuarios
   3.2 Modificar usuarios
   3.3 Eliminar usuarios
4. Gestionar tareas.
   4.1 Insertar tarea
   4.2 Modificar tarea
   4.3 Eliminar tarea
5. Gestionar estudiantes
   5.1 Insertar estudiantes
   5.2 Modificar estudiantes

5.3 Eliminar estudiantes

6 Gestionar Roles

6.1 Insertar Roles

6.2 Modificar Roles

6.3 Eliminar Roles

7. Gestionar reportes

7.1 Insertar reportes

8. Realizar cálculos de notas finales

1.5 Requerimientos no funcionales

Son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

- **Requerimientos de Seguridad**

  Confidencialidad: La información manejada por el sistema está protegida de acceso no autorizado y divulgación.

  Integridad: la información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos.

  Disponibilidad: Significa que los usuarios autorizados se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

- **Requerimientos de Usabilidad:** El sistema debe ser fácil de utilizar para los usuarios autorizados y lo menos costoso posible.Facilidad de aprendizaje, flexibilidad y robustez.
- **Requerimientos de Soporte:** Abarcan todas las acciones a tomar una vez que se ha terminado el desarrollo del software con motivos de asistir a los clientes de este así como lograr su mejoramiento progresivo y evolución en el tiempo. Pueden incluir: Pruebas, Extensibilidad,

Adaptabilidad, Mantenimiento, Compatibilidad, Configuración, Servicios, Instalación, Internacionalización y Requerimientos de Portabilidad.

- **Requerimientos de Software:** Sistema Operativo Windows 95 o Superior; Linux, plataformas Unix, entre otras.
- **Requerimientos de Hardware**: se requiere disponer de una RAM de 512 o superior.
- **Apariencia o interfaz externa:** El sistema debe implementar una interfaz Web con un diseño sencillo, legible, interactivo.
- **Rendimiento:** El sistema debe ser eficiente, lograr un tiempo de respuesta mínimo, ya que de por si las aplicaciones Web son desconectadas y tienen que acceder al servidor en busca de cualquier información por lo que la velocidad de procesamiento es un parámetro fundamental.

1.6 Modelo de Casos de Usos del Sistema

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores.

1.7 Diagrama de Casos de Usos del Sistema

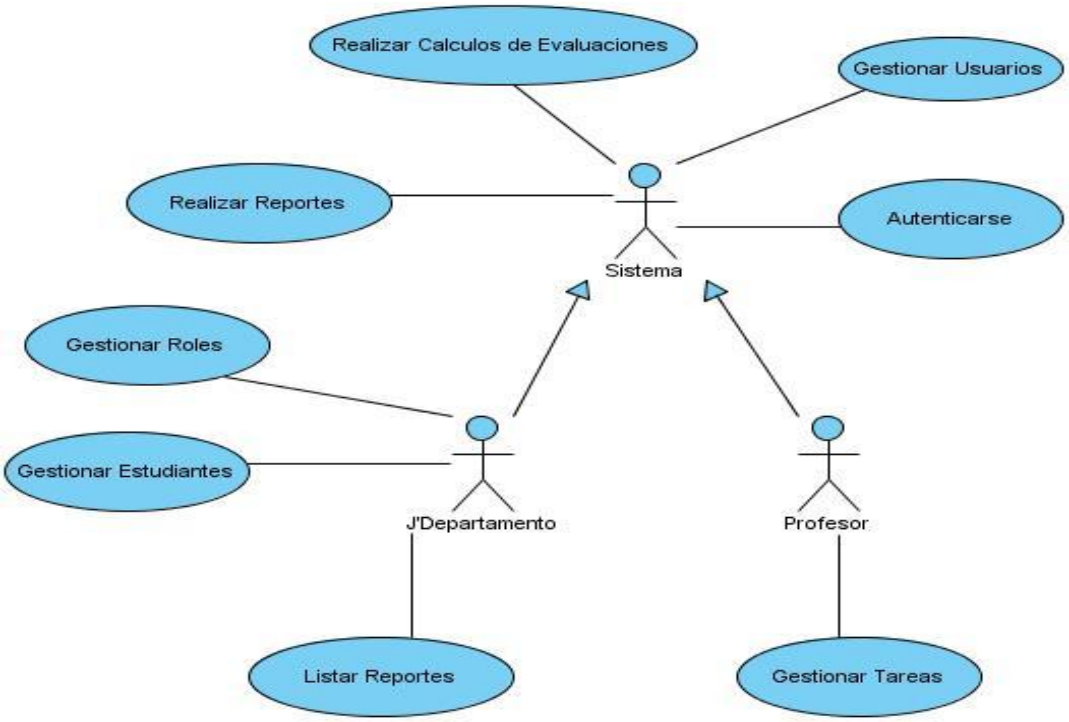

Figura 4 Diagrama de CU del Sistema

1.8 Definición de los actores del Sistema

Actores	Justificación
Profesor	Representa a la persona que realiza todo el proceso de gestión de evaluaciones docentes.
Administrador	Encargado de realizar la autenticación en el sistema.

Tabla 1. Definición de los actores del Sistema

1.9 Listado de Casos de Usos del Sistema. Breve descripción

CU1	Autenticarse
Actor	Administrador
Descripción	Todos los usuarios deben autenticarse al interactuar con la aplicación
Referencia	1

Tabla 2. Descripción del caso de uso Autenticarse

CU 2	Gestionar profesores
Actor	Administrador
Descripción	El administrador decide que acción va a realizar (insertar, eliminar o modificar) en caso de insertar o modificar llena los campos necesarios y en caso de eliminar selecciona la información deseada.
Referencia	2

Tabla 3. Descripción del caso de uso Gestionar profesores

CU 3	Gestionar Usuario
Actor	Administrador

Descripción	El administrador decide que acción va a realizar (insertar, eliminar o modificar) en caso de insertar o modificar llena los campos necesarios y en caso de eliminar selecciona la información deseada.
Referencia	3

Tabla 4. Descripción del caso de uso Gestionar Usuario

CU 4	Gestionar Tareas
Actor	Profesor
Descripción	El profesor escoge la opción deseada (insertar, modificar o eliminar) luego si lo que quiere es insertar o modificar llena los campos necesarios y si lo que desea es eliminar selecciona la información que necesita. Si desea asignar una nota llena el campo nota.
Referencia	5

Tabla 5. Descripción del caso de uso Gestionar Tareas

CU 5	Gestionar Estudiantes
Actor	Profesor(Jefe ‘Departamento)
Descripción	El profesor escoge la opción deseada (insertar, modificar o eliminar) luego si lo que quiere es insertar o modificar llena los campos necesarios y si lo que desea es eliminar selecciona la información que necesita
Referencia	6

Tabla 6. Descripción del caso de uso Gestionar Estudiantes

CU 6	Gestionar Roles
Actor	Profesor(Jefe’ Departamento)

Descripción	El Administrador escoge la opción deseada (insertar, modificar o eliminar) luego si lo que quiere es insertar o modificar llena los campos necesarios y si lo que desea es eliminar selecciona la información que necesita
REFERENCIA	8

Tabla 7. Descripción del caso de uso Gestionar Roles

CU 7	Gestionar Reportes
Actor	Administrador
Descripción	El administrador inserta el reporte de las evaluaciones de todos los estudiantes.
Referencia	7

Tabla 8. Descripción del caso de uso Interfaz Usuario

CU 8	Realizar Cálculos de Evaluaciones Finales
Actor	Administrador
Descripción	El administrador realiza el cálculo de las evaluaciones de todos los estudiantes.
Referencia	8

Tabla 9. Descripción del caso de uso Realizar Cálculos de Evaluaciones Finales

4.6. Conclusiones Parciales

El presente capítulo se demuestra la necesidad de realizar una aplicación que garantice la gestión de evaluaciones docentes en la Disciplina de Programación. Demostrándose a través de los requisitos funcionales del sistema cada una de las acciones que se realizarán y cómo el sistema propuesto cumple con todos esos requerimientos necesarios para el logro del objetivo propuesto Mediante las características de la solución que se propone se evidencia las grandes ventajas que aporta la aplicación para el control de los resultados docentes, ofreciéndole a los profesores una herramienta cómoda, fiable y viable para su desempeño.

CAPÍTULO 3: DISEÑO DEL SISTEMA

INTRODUCCIÓN

En el presente capítulo se realizará el diseño del sistema de manera tal que se muestre con claridad cómo quedará estructurada la aplicación en términos de componentes web. Además se abordarán los patrones de diseño utilizados.

1. Diseño

Es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. De hecho, cuando la precisión del diseño es muy grande, la implementación puede ser hecha por un generador automático de código.

1.1. Diagramas de clases del diseño

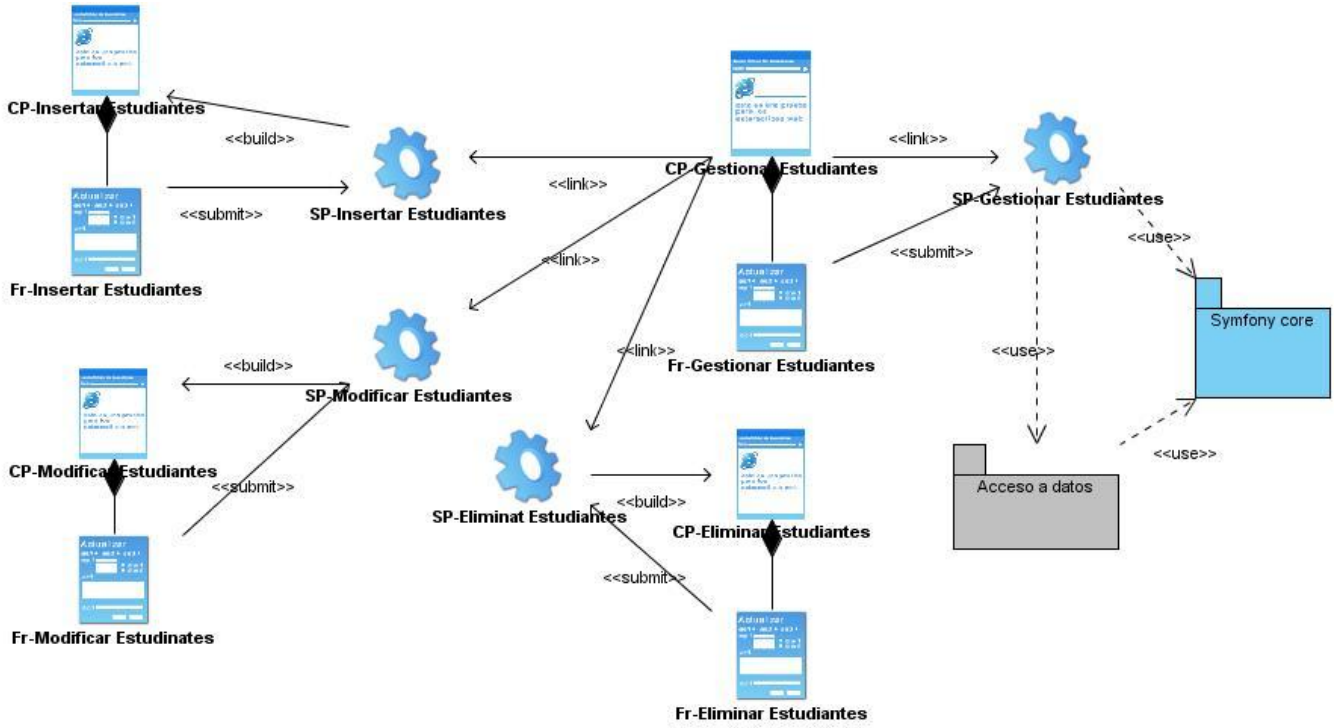

Figura 5 CU Gestionar Estudiantes

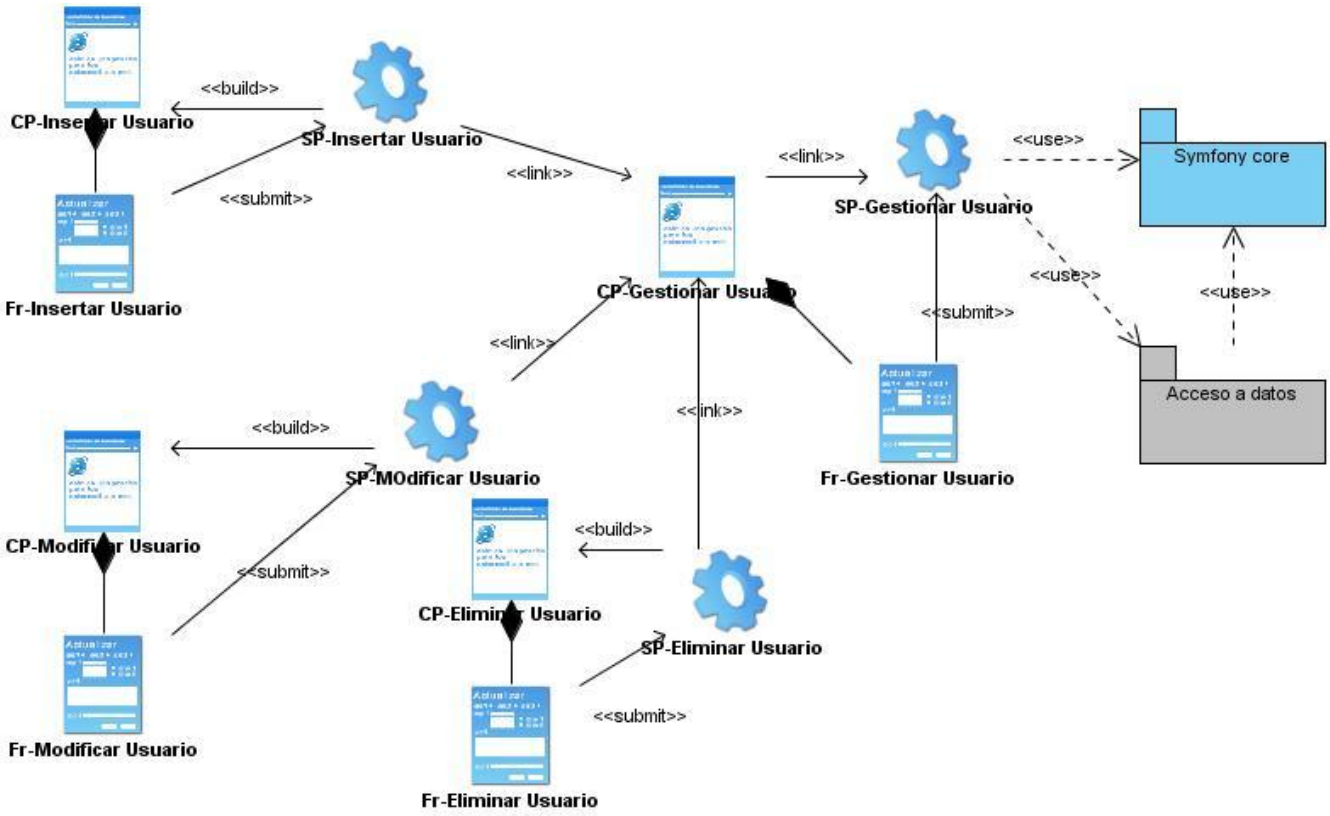

Figura 6 CU Gestionar Usuarios

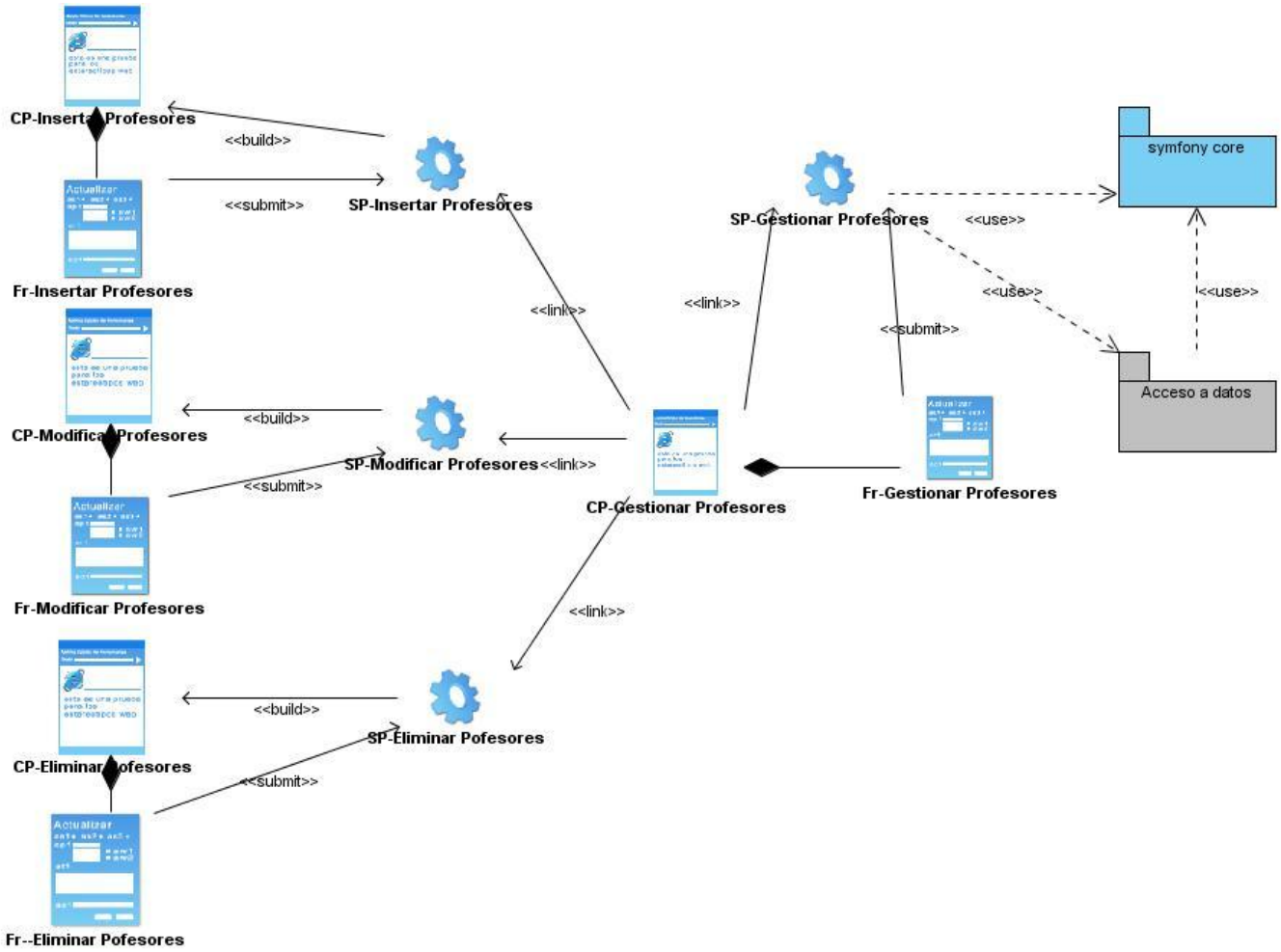

Figura 7 CU Gestionar Profesores

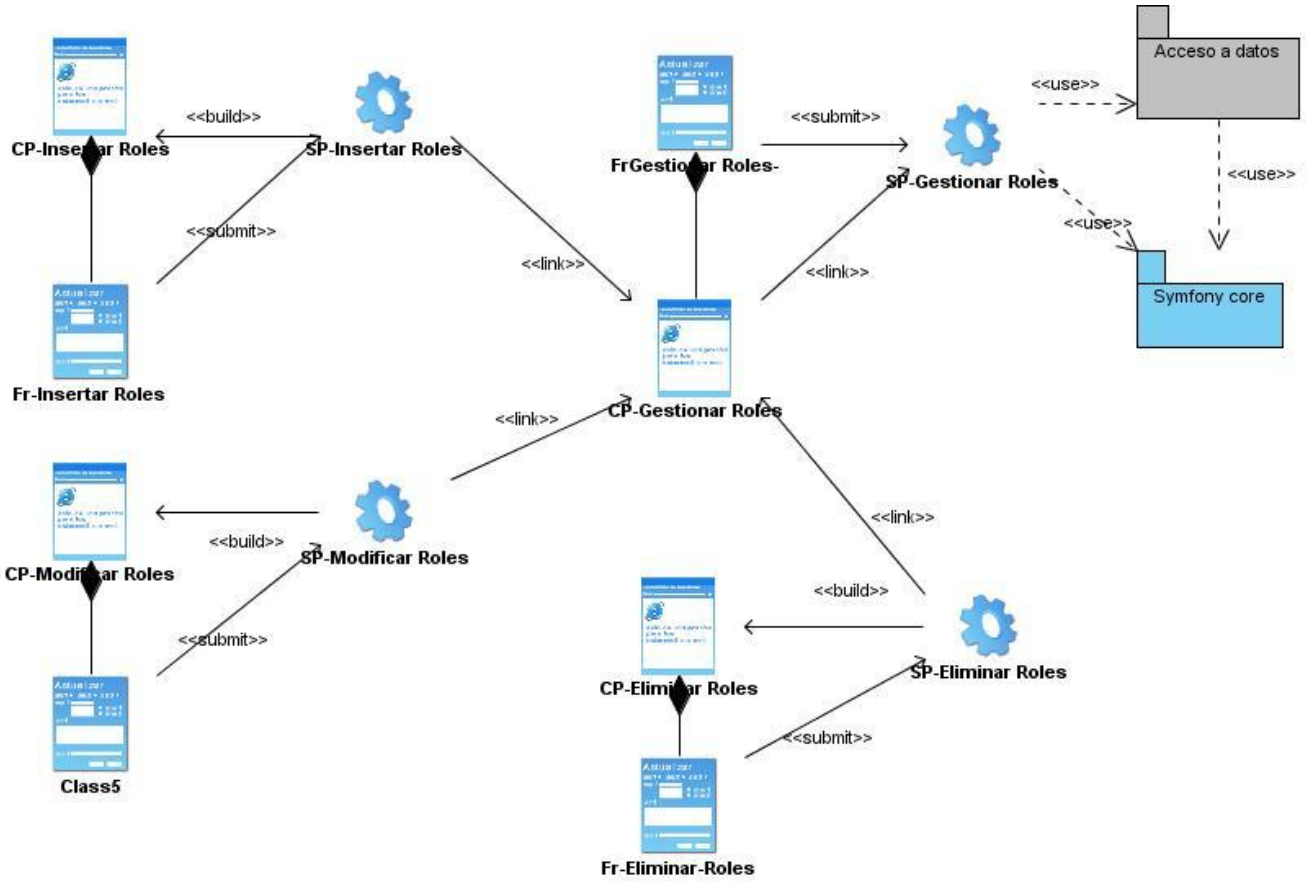

Figura 8 CU Gestionar Roles

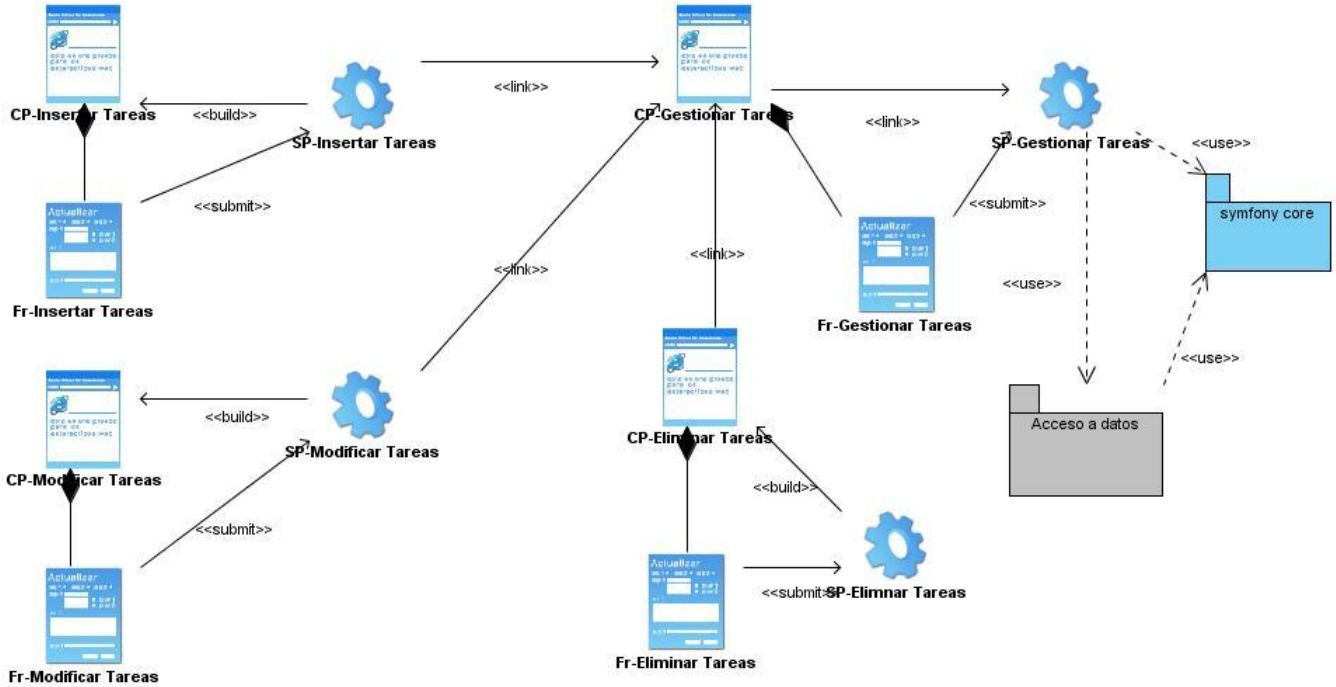

Figura 9 CU Gestionar Tareas

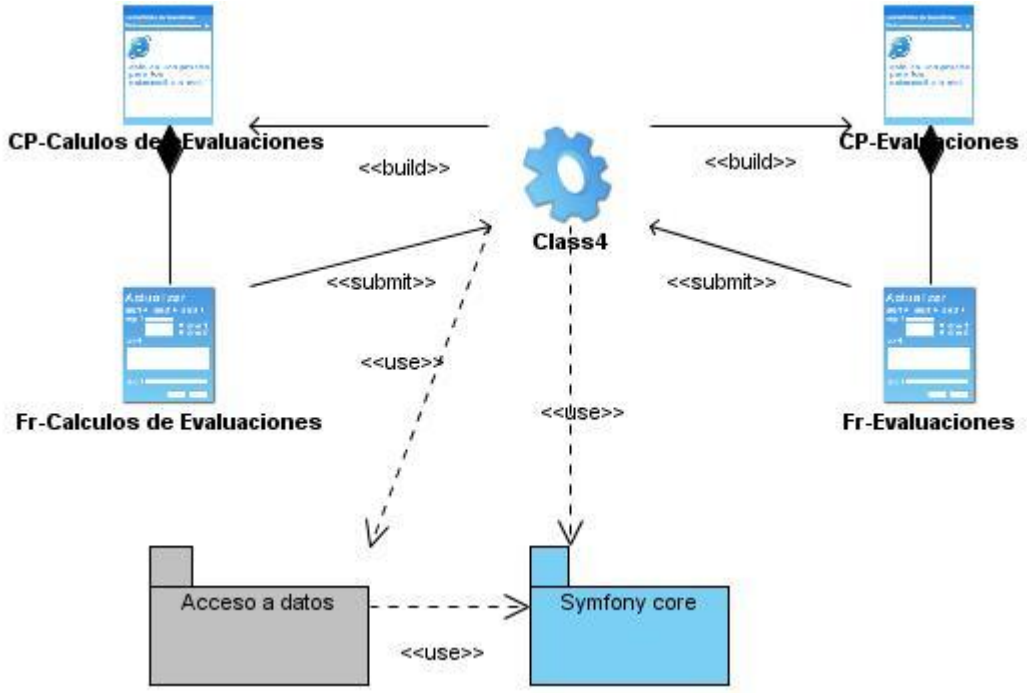

Figura 10 CU Calcular Evaluaciones

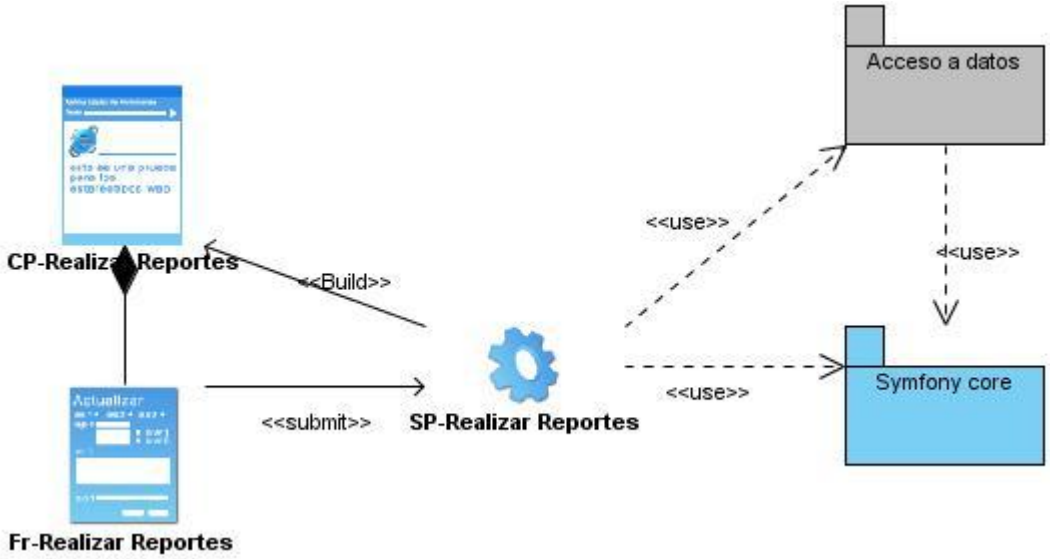

Figura 11 CU Gestionar Reportes

2. Paquete de Acceso a Datos

Son orientados a objeto, entonces para acceder a los datos necesitamos una capa que traduzca la lógica objeto a la lógica relacional, esta capa se llama ORM (object-relational mapping). Esta capa de abstracción está basada en el proyecto Doctrine tiene. Behaviors: Doctrine soporta la aplicación de varios "behaviors" a tus modelos, por ejemplo un modelo Timestampable

automáticamente creará 2 columnas: created_at y updated_at, las cuales guardarán la fecha cuando un registro es creado y cuando un registro es actualizado respectivamente. Searching: Doctrine tiene un motor de búsqueda fulltext. Además, Doctrine soporta Data fixtures y migraciones, caching, events, paginación, command line interface Doctrine. Básicamente, podemos acceder y modificar datos de una base de datos de forma sencilla utilizando la lógica objeto y no los clásicos Update y Select de SQL.

Figura 12 Paquete de acceso a datos

Su principal ventaja es el rendimiento en ejecución y la forma tan concisa en la que se pueden escribir consultas muy complejas.

3. Diagrama de Objetos

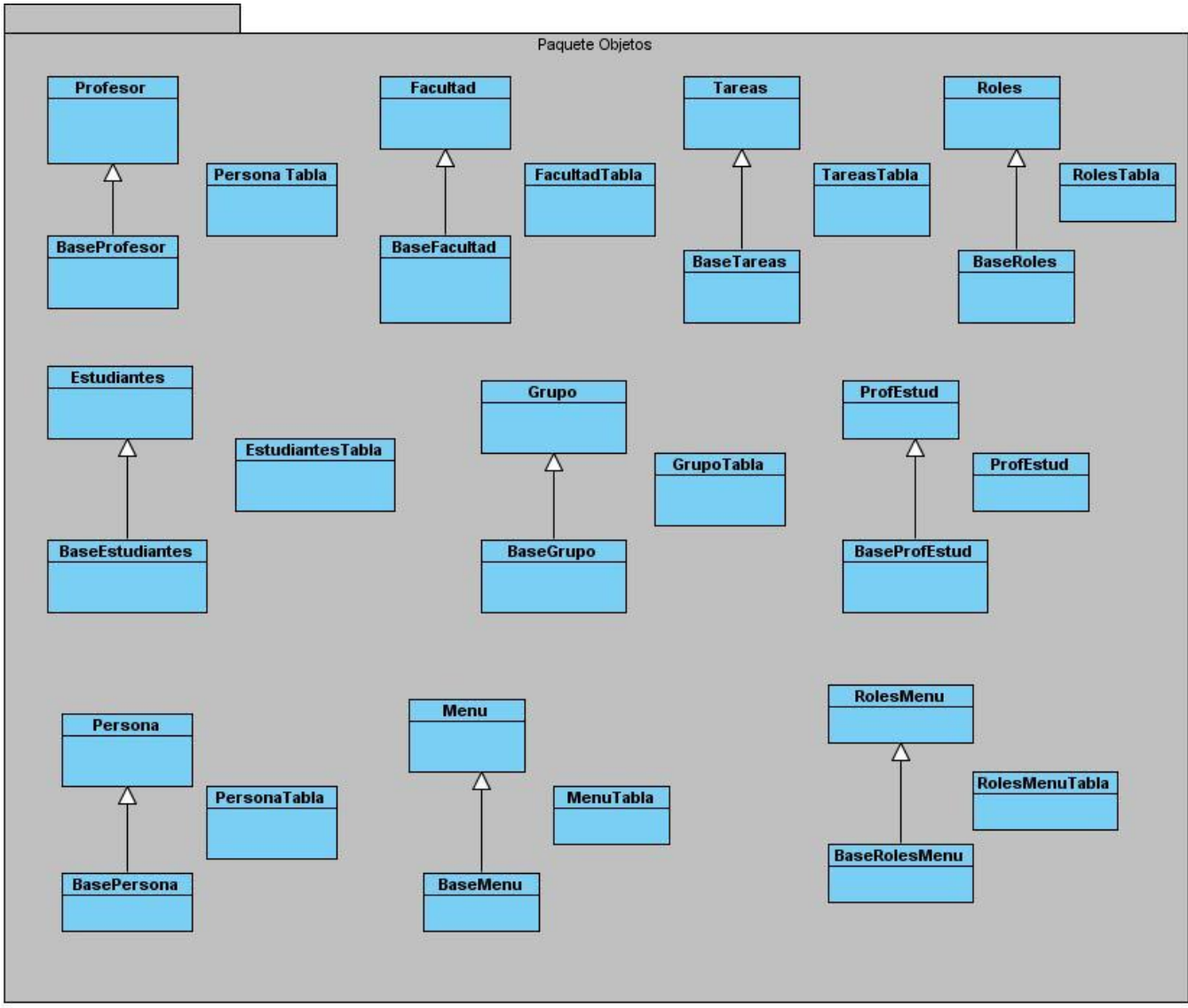

Figura 13. Diagrama de Objetos

4. Diagramas de Secuencias

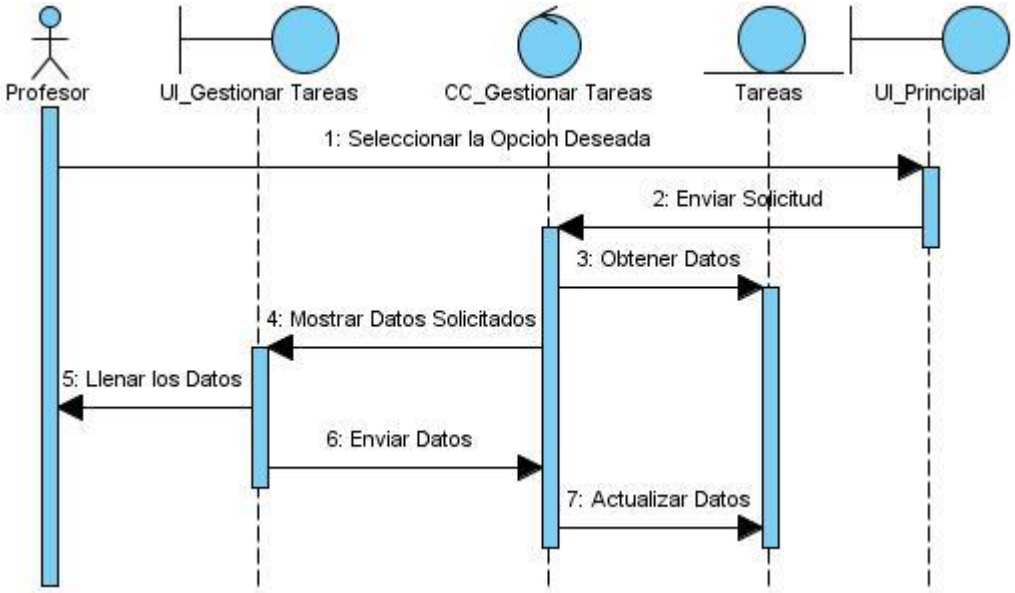

Figura 14. Diagrama de Secuencia CU Gestionar Tareas

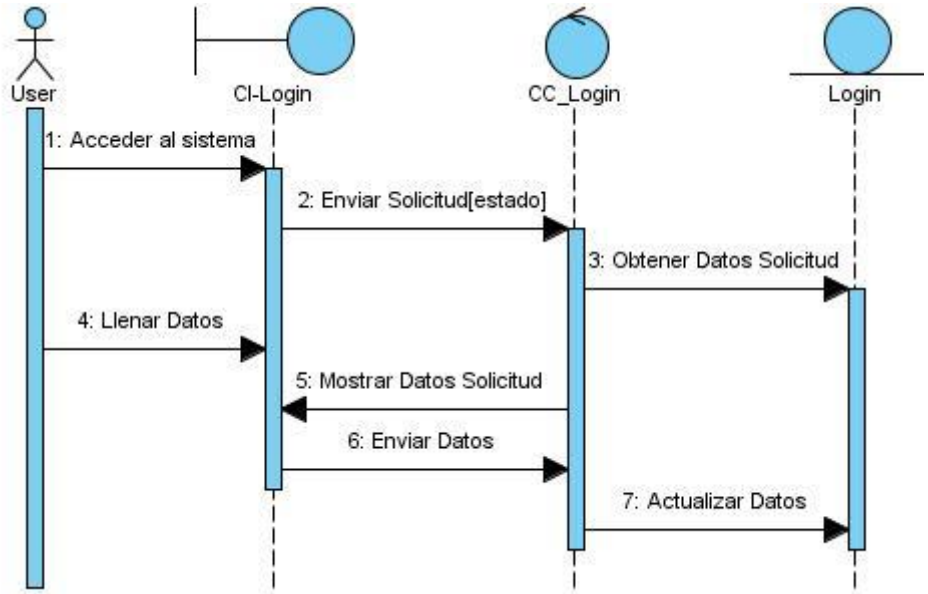

Figura 15. Diagrama de Secuencia CU Autenticarse

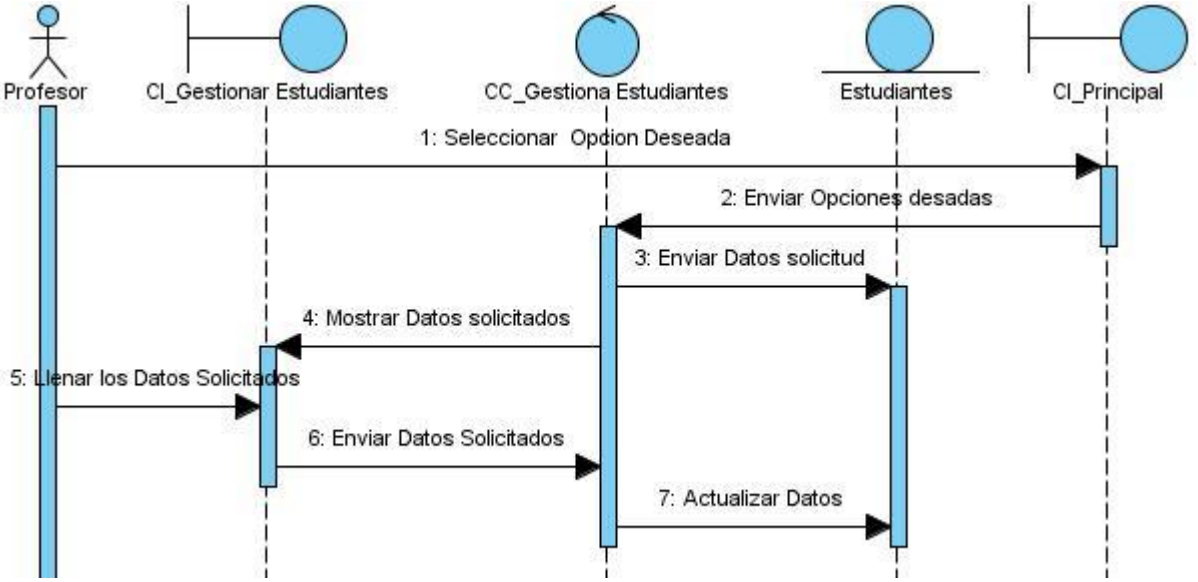

Figura 16. Diagrama de Secuencia CU Gestionar Estudiantes

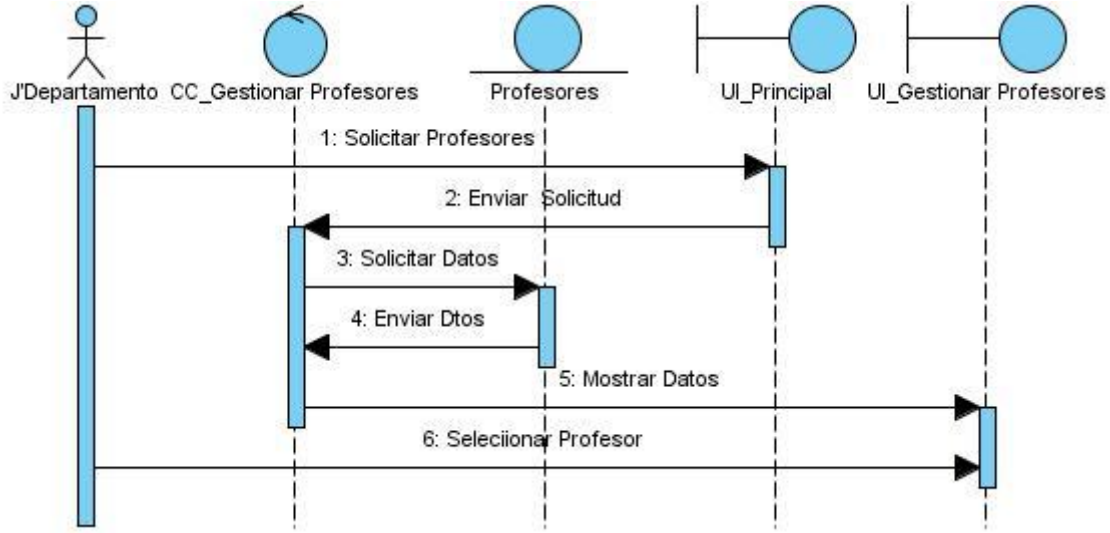

Figura 17. Diagrama de Secuencia CU Gestionar Profesores

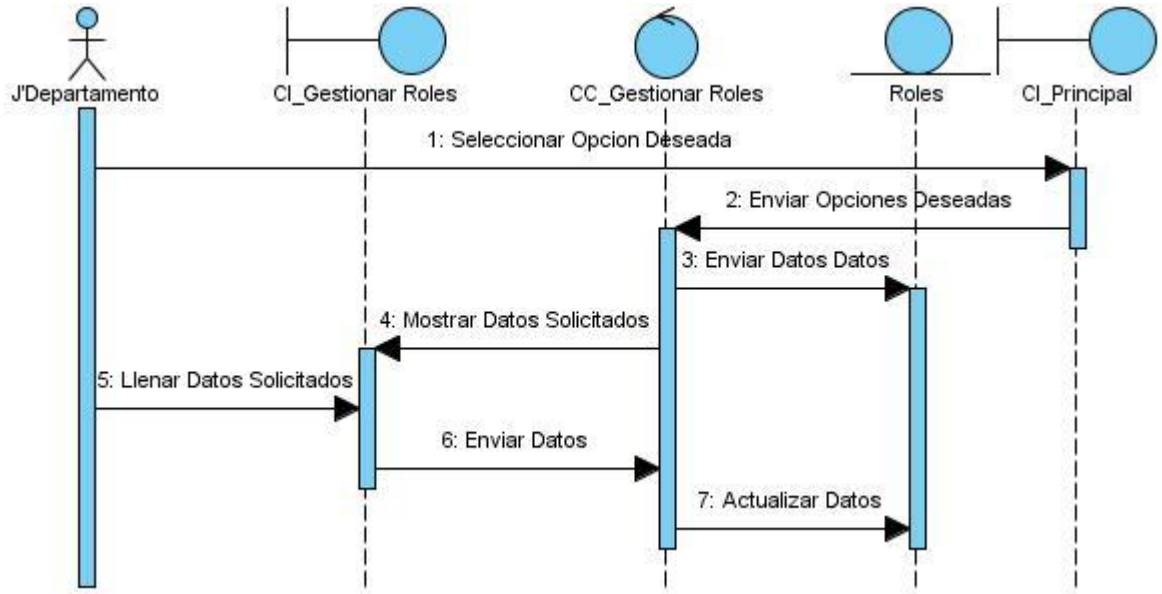

Figura 18. Diagrama de Secuencia CU Gestionar Roles

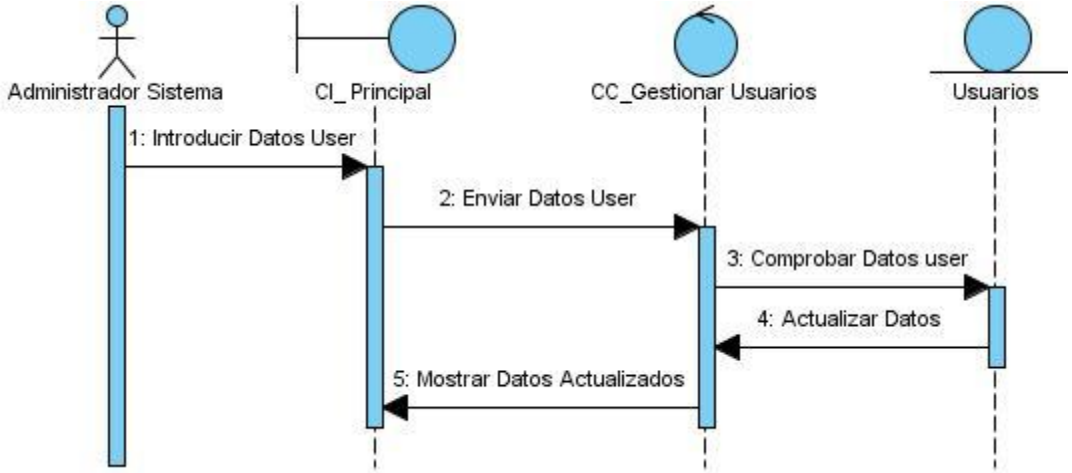

Figura 19. Diagrama de Secuencia CU Gestionar Usuarios

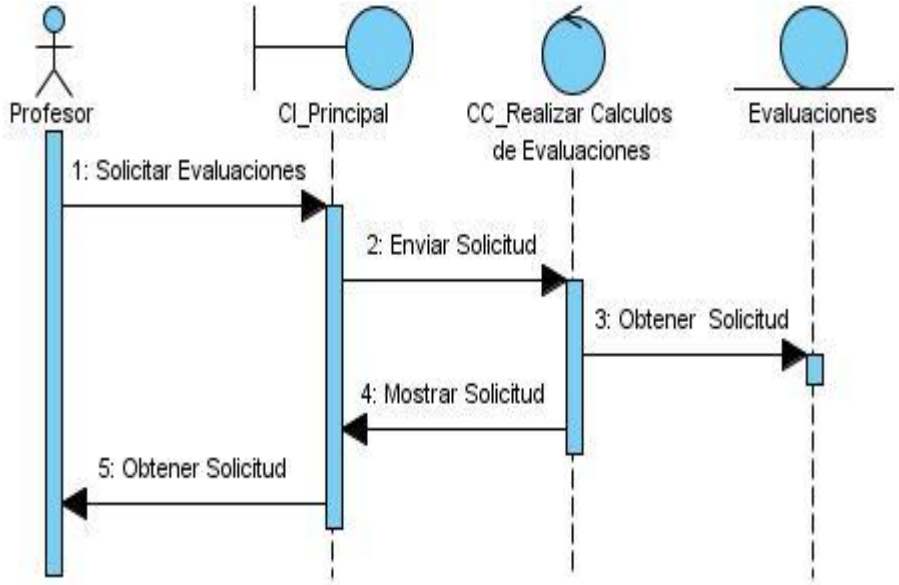

Figura 20. Diagrama de Secuencia CU Realizar Cálculos de Evaluaciones

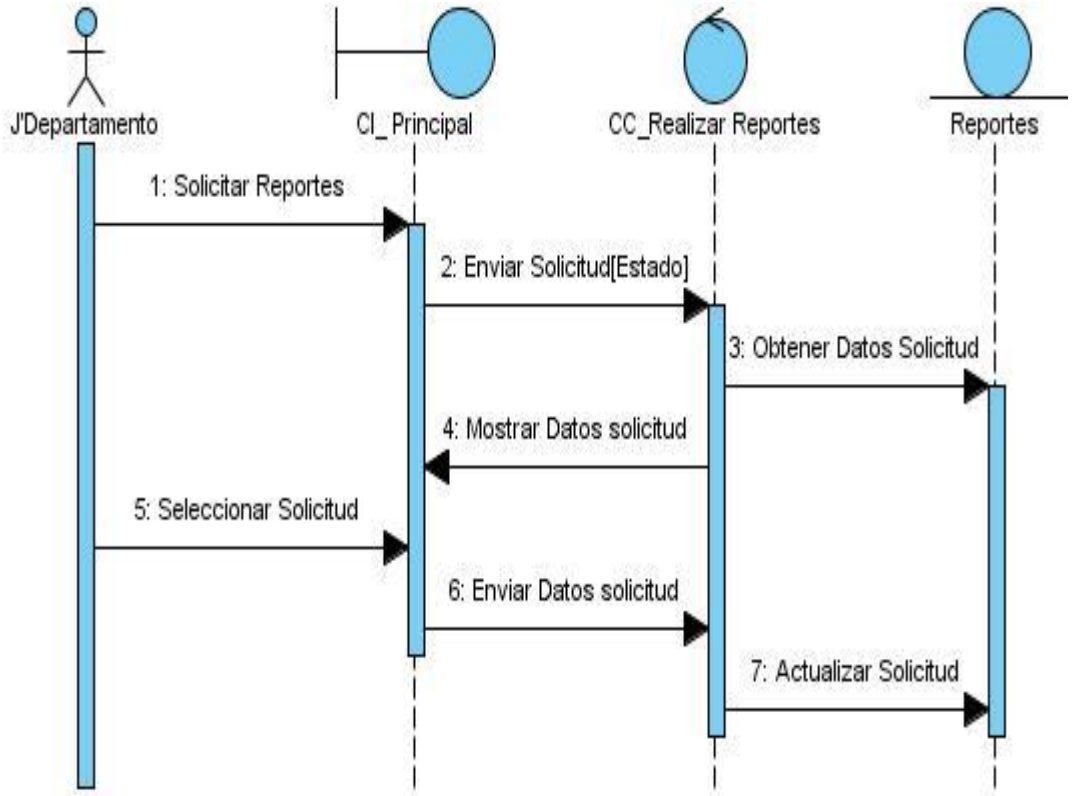

Figura 21. Diagrama de Secuencia CU Realizar Reportes

5. Diseño de la Base de Datos

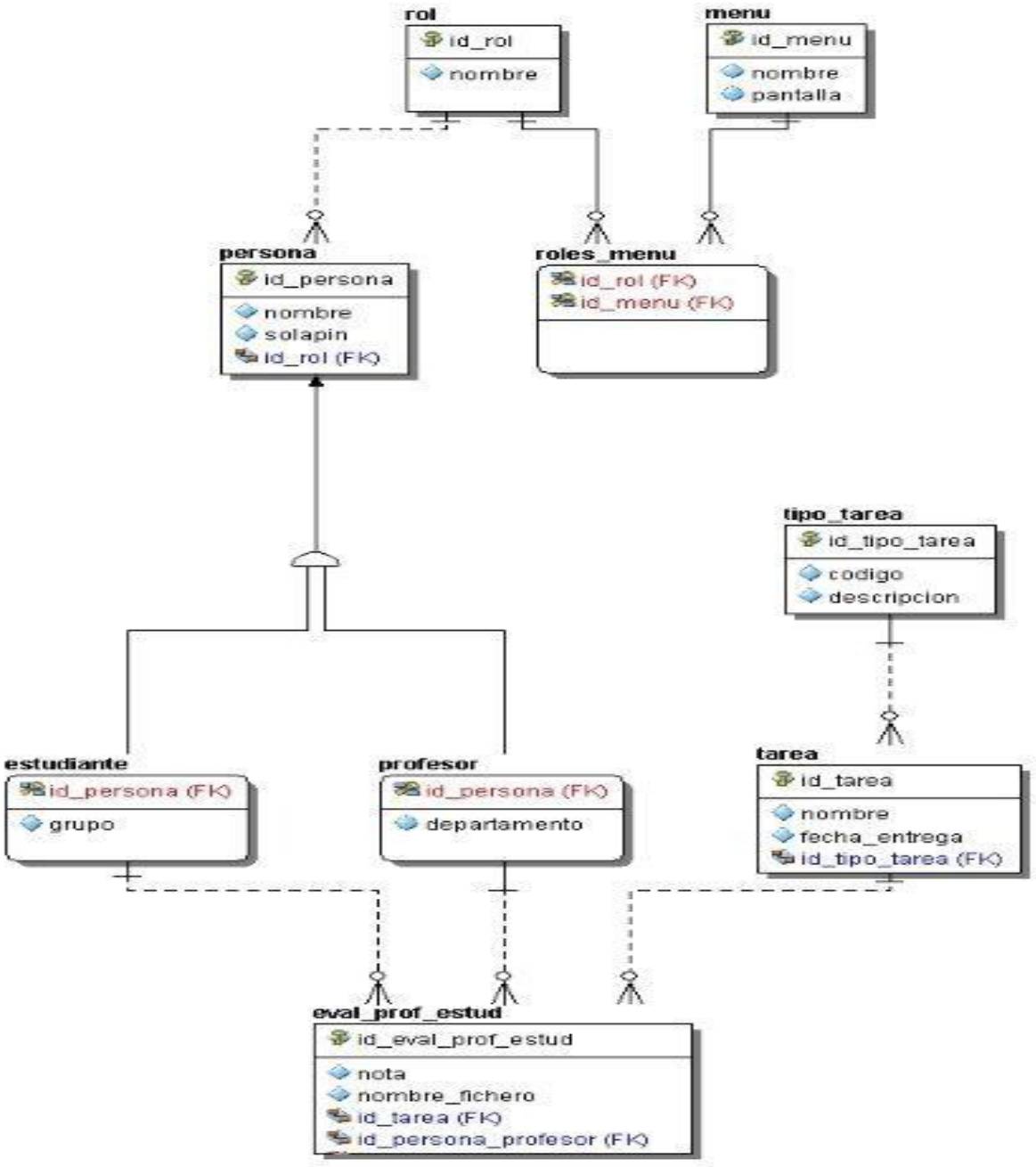

Figura 22. Modelo de la Base de Datos

6. Diagrama de Despliegue

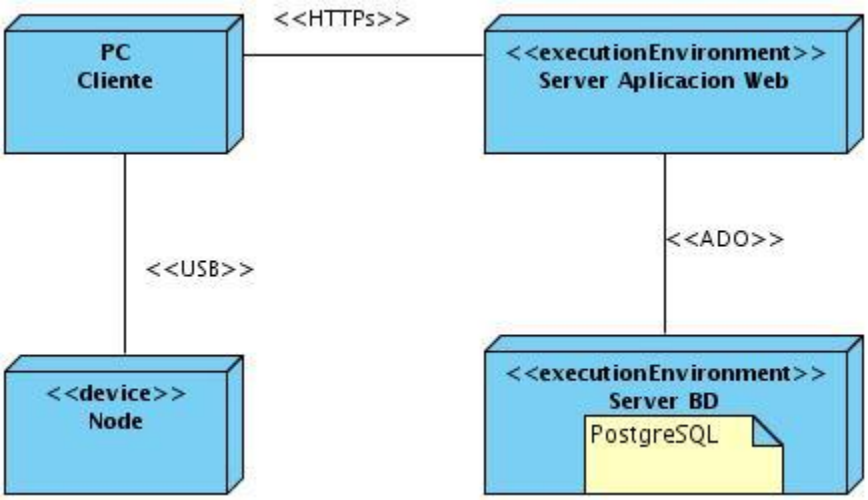

Figura 23. Diagrama de Despliegue

7. Conclusiones Parciales

El diseño de un sistema es tan fundamental como la implementación del mismo. Te garantiza la modelación del sistema con sus estereotipos Web definidos, los cuales permiten tener una visión objetiva de la aplicación a desarrollar en términos de componentes. Se realizó un diseño robusto de la aplicación la cual cuenta con una estructura que satisface las necesidades de los usuarios finales.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA

INTRODUCCIÓN

En el siguiente capítulo se tratará la implementación y prueba del sistema donde se obtendrán los diferentes artefactos generados en esta fase como los diagramas de secuencias y el diagrama de componentes. Además de realizar una validación de la aplicación en función de verificar que cumpla con todas las funcionalidades requeridas por los usuarios finales. Implementaremos el sistema en términos de componentes y subsistemas de implementación, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

1. Objetivos

El principal objetivo de la etapa de implementación es desarrollar la arquitectura y el sistema como un todo. De forma más específica, los propósitos de la Implementación son:

- Definir la organización del código.
- Planificar las integraciones de sistema necesarias en cada iteración.
- Implementar las clases y subsistemas encontrados durante el Diseño.

1.1. Implementación

El modelo de implementación describe cómo los elementos de diseño se implementan en componentes.

1.2. Diagrama de Componentes

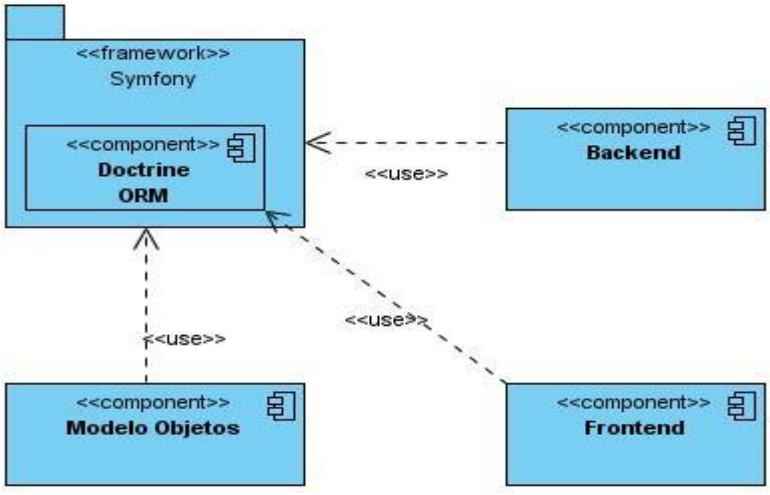

Figura 24. Diagrama de Componentes

2. Prueba

La prueba consiste en la verificación de conformidad de los servicios del trabajo con respecto a las especificaciones técnicas del cliente. Es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. Es un paso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

2.1. TIPOS DE PRUEBAS

2.1.1. Prueba de caja blanca

Permiten examinar la estructura interna del programa. Se diseñan casos de prueba para examinar la lógica del programa. Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar casos de prueba que garanticen que:

- Se ejercitan todos los caminos independientes de cada módulo.
- Se ejercitan todas las decisiones lógicas.
- Se ejecutan todos los bucles.
- Se ejecutan las estructuras de datos internas.

2.1.2. Prueba de caja negra

Las pruebas se llevan a cabo sobre la interfaz del software, y es completamente indiferente el comportamiento interno y la estructura del programa. Los casos de prueba de la caja negra pretende demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Se derivan conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales del programa. La prueba de la caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.

- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Los casos de prueba deben satisfacer los siguientes criterios:

- Reducir, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales.
- Que digan algo sobre la presencia o ausencia de clases de errores.

2.2. Diseño de Casos de Pruebas

Se mostrará el caso de prueba para el caso de uso Adicionar Tareas el resto será referenciado en los anexos del presente trabajo.

Diseño de casos de prueba

Adicionar Tareas

1.1

Fecha	Versión	Descripción	Autor
15/05/2010	1.0	Elaboración del DCP del requisito Adicionar Tareas	Aracelis Ruiz M

Tabla 10. Control de versiones

No.	Fecha	Aprobado por:	Cargo	Firma
<#>	<dd/mm/aaaa>	<nombre>	<cargo>	<firma>

Tabla 11. Tabla de aprobación

Reglas de seguridad

El que recibe el documento asume la custodia y control, comprometiéndose a no reproducir, divulgar, difundir o de cualquier manera hacer de conocimiento público su contenido, excepto para cumplir el propósito para el cual se ha generado.

1. Condiciones de ejecución

- Se debe seleccionar el subsistema de Gestión de Evaluaciones Docentes
- Se debe seleccionar la opción **/Gestionar Tareas**

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar Tarea	Se adiciona una nueva Tarea con los datos siguientes: nombre, tipo de Tarea , fecha de entrega y evaluación	EP 1.1: Adicionar una nueva Tarea introduciendo datos válidos.	• Se introducen los datos de la Tarea que • Se desea adicionar. • Se presiona el botón **Aceptar**. • Se muestra un mensaje: “Los datos han sido guardados satisfactoriamente.”.
		EP 1.2: Adicionar una nueva Tarea introduciendo datos inválidos.	• Se introducen datos inválidos. • Se presiona el botón **Aceptar**.
		EP 1.3: Adicionar una nueva evaluación dejando campos requeridos en blanco.	• Se introducen los datos de la evaluación dejando algún campo requerido en blanco. • Se presiona el botón **Aceptar**.
		EP 1.4: Cancelar.	• Se introducen o no los datos de la evaluación. • Se presiona el botón **Cancelar.**

Tabla 12. Requisitos a probar

No	Nombre de campo	Tipo	Válido	Inválido

1	Nombre	Cadena de caracteres	Sólo letras	Cualquier carácter extraño excepto letras.
2	Tipo de Tareas	Cadena de caracteres	Sólo letras.	Cualquier carácter extraño excepto letras.
3	Fecha de entrega	Date	<dd/mm/aaaa>	Sólo fecha.
4	Evaluación	Numérico	Sólo números	Cualquier carácter extraño excepto números.

Tabla 13. Descripción de variable

Id del escenario	Escenario	Nombre	Tipo de tarea	Fecha de entrega	Evaluación	Respuesta del sistema	Resultado de la prueba

EP 1.1	Se adiciona una nueva Tarea con los datos siguientes: nombre, tipo de tarea, fecha de	V(prueba2)	V(cadena de caracteres)	V(3/02/ 2010	V(sólo números)	El sistema muestra una ventana de información: "Los datos han sido guardados satisfactoria	
		V(prueba1)	V(caracteres)	V(date)	2	El sistema muestra una ventana de información: "Los datos han sido guardados satisfactoria	
EP 1.2	Se adiciona una nueva tarea introduciendo datos inválidos	I(erf)	V(cadena de caracteres)	V(7/5/09	kjd	El sistema muestra una ventana de información: "Los datos introducidos no son correctos.".	

		V(única)	I(kek4)	V(No)	myk	El sistema muestra una ventana de información: “Los datos introducidos no son correctos.”.	
EP 1.3	Se adiciona una nueva tarea introduciendo datos inválidos	I(Vacío)	V(321)	V(No)		El sistema muestra una ventana de información: “Ha dejado datos requeridos en blanco.”.	
		V(Lle6no)	I(Vacío)	V(No)		El sistema muestra una ventana de información: “Ha dejado datos requeridos en blanco.”.	

		V(totallidad 7)	V(este)	I(Vacío)		El sistema muestra una ventana de información: “Ha dejado datos requeridos en blanco.”.	
		V(sóloest3)	V(cuando)	V(Si)		El sistema muestra una ventana de información: “Ha dejado datos requeridos en blanco.”	
EP 1.4	Cancelar.	NA	NA	NA		Se cancela la operación y se cierra la ventana.	

Tabla 14. Juegos de datos a probar

[Las celdas de la tabla contienen V, I, o NA. V indica válido, I indica inválido, y NA se debe proporcionar un valor del dato en este caso].

3. Conclusiones Parciales

En el capítulo se abordó el tema referente a la implementación y prueba del sistema dónde se generaron los diferentes diagramas y se modeló la aplicación en términos de componentes y

subsistemas de implementación. Se realizaron diferentes tipos de pruebas para comprobar la factibilidad de sistema en función de los requerimientos definidos por los usuarios finales.

CONCLUSIONES

Luego de la investigación realizada para la confección de la presente tesis de diploma, se tuvo en cuenta una serie de actividades durante el proceso de desarrollo de la misma. Para ello se definieron varias tareas que dieron la posibilidad de obtener finalmente los resultados esperados. Se logró específicamente:

- Desarrollar un sistema para la gestión de evaluaciones docentes en la asignatura de Programación.
- La aplicación le permitirá a los profesores llevar un control estricto de las notas de sus estudiantes en el preciso momento que lo necesite.
- Registrar cada una de las evaluaciones realizadas durante todo un semestre.
- Obtener de manera automática los resultados finales de todos sus alumnos.
- Permite adquirir reportes de sus colegiales en la disciplina.
- Desarrollar un sistema que facilite la gestión de evaluaciones docentes en la Disciplina de Programación en la Universidad de las Ciencias Informáticas.

RECOMENDACIONES

Se recomienda que el presente trabajo sea utilizado por todos los profesores de la universidad en asignatura de programación para la gestión de sus evaluaciones. Añadirle nuevas funcionalidades en dependencia de las necesidades que vayan surgiendo.

REFERENCIAS BIBLIOGRÁFICAS

- (Blejmar, 2005) **Blejmar, Bernardo. 2005.** *es Gestionar hacer...* [ed.] Primera Edición Marzo de 2005. 1a. Colombia : Ediciones Novedades Educativas - NOVEDUC (Argentina - México), 2005. pág. 143. ISBN: 9788499240459
- **(MELGAR VELIZ, 2009)** Melgar Veliz, Manuel Alejandro. 2009. Lo Mejor De Melgar. *Lo Mejor De Melgar.* [En línea] 29 de Septiembre de 2009. [Citado el: 18 de Mayo de 2010.]http://oscar-lomejordemelgar.blogspot.com/2009/09/definicion-de-programacion.html.
- **(RETO, 2008) RETO,** Lic JUAN GARCÍA. 2008. La evaluación de la función docente debe ser constructiva. La evaluación de la función docente debe ser constructiva. [En línea] 17 de noviembre de 2008. [Citado el: 12 de abril de 2010.] http://www.blogextremo.com/licjuangarciareto.
- **(Maccario)** Maccario, Bernard. Evaluacion Educativa. *Evaluacion Educativa.* [En línea] [Citadoel:10 de Marzo de 2010.] http://www.chasque.apc.org/gamolnar/evaluacion%20educativa/evaluacion.01.html.
- **(Pila Teleña)** Pila Teleña, Augusto. Evaluación Educativa. *Evaluación Educativa.* s.l. : Editorial Verbum,S.L.
- (Stufflebeam, 2002) Stufflebeam, Daniel. 2002. Evaluación Sistemática: Guía Teórica Y Práctica. Evaluación Sistemática: Guía Teórica Y Práctica. s.l. Paidos Ibérica, 2002.
- (R. Tyler) R. Tyler, B. Bloom, G. De Landsheere, B. Maccario. Educacion Educativa. Educacion Educativa. [En línea] www.chasque.apc.org/.
- Wikilibros. Wikilibros. [En línea] http://es.wikipedia.org/wiki/PHP.

BIBLIOGRAFÍA

- Blejmar, Bernardo. 2005. *Gestionar es hacer...* [ed.] Primera Edición Marzo de 2005. 1a. Colombia : Ediciones Novedades Educativas - NOVEDUC (Argentina - México), 2005. pág. 143. ISBN: 9788499240459.
- —. 2005. Gestionar es hacer... *Gestionar es hacer...* Buenos Aires : Novedades Educativas, 2005.
- —. 2005. *Gestionar es hacer...* [ed.] Primera Edición Marzo de 2005. 1a. Colombia : Ediciones Novedades Educativas - NOVEDUC (Argentina - México), 2005. pág. 143. ISBN: 9788499240459.
- curso-aprende-programar. *curso-aprende-programar.* [En línea] [Citado el: 10 de febrero de 2010.] http://www.mailxmail.com/curso-aprende-programar/conceptos-basicos-programacion.
- doctrine-project. *doctrine-project.* [En línea] [Citado el: 10 de abril de 2010.]

  http://www.doctrine-project.org/.
- González Miranda, Luis Alejandro. 2002. *LGM: El Shell de Unix.* Chile : LGM Producciones, 2002.
- González Moreno, Juan Carlos. 2009. Análisis de Requisitos.IEEE. [En línea] 2009.
- groups.google.com. *groups.google.com.* [En línea] [Citado el: 16 de Mayo de 2010.] http://groups.google.com/group/symfony-users/msg/cd94d2ddb2057355.

  http://www.librosweb.es. *http://www.librosweb.es.* [En línea]
- implementacion_d.php. *implementacion_d.php.* [En línea] [Citado el: 20 de mayo de 2010.] http://lsi.ugr.es/~arroyo/inndoc/doc/implementacion/implementacion_d.php.
- Isaac, German. 2005. mastermagazine. *mastermagazine.* [En línea] 16 de Febrero de 2005. [Citado el: 12 de Marzo de 2010.]

  http://www.mastermagazine.info/termino/6400.php.

Bibliografía

- Maccario, Bernard. Evaluacion Educativa. *Evaluacion Educativa.* [En línea] [Citado el: 10 de Marzo de 2010.]

  http://www.chasque.apc.org/gamolnar/evaluacion%20educativa/evaluacion.01.html.

- MELGAR VELIZ, MANUEL ALEJANDRO. 2009. LO MEJOR DE MELGAR. *LO MEJOR DE MELGAR.* [En línea] 29 de Septiembre de 2009. [Citado el: 18 de Mayo de 2010.] http://oscar-lomejordemelgar.blogspot.com/2009/09/definicion-de-programacion.html.

- modelo+de+implementacion. *modelo+de+implementacion.* [En línea] [Citado el: 18 de mayo de 2010.]

  http://www.google.com.cu/#hl=es&source=hp&q=modelo+de+implementacion&aq=f&aqi=&aql=&oq=&gs_rfai=&fp=64acf8a8e7f19d1c.

- Pila Teleña, Augusto. Evaluación Educativa. *Evaluación Educativa.* s.l. : Editorial Verbum,S.L.

- postgresql. *postgresql.* [En línea] [Citado el: 20 de marzo de 2010.]

  http://www.sqlmanager.net/products/postgresql/manager/.

- Potencier, Fabien y Zaninotto, François. 2008. *Symfony la guía definitiva.* s.l. : Apress , 2008. ISBN-13: 978-1590597866.

- propel. *propel.* [En línea] [Citado el: 10 de abril de 2010.] http://propel.phpdb.org/.

- propel-y-el-modelo-symfony. *propel-y-el-modelo-symfony.* [En línea] [Citado el: 5 de abril de 2010.] http://palermi.wordpress.com/2006/07/01/propel-y-el-modelo-symfony/.

- pruebas_d.php. *pruebas_d.php.* [En línea] [Citado el: 25 de mayo de 2010.] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php.

- R. Tyler, B. Bloom, G. De Landsheere, B. Maccario. Educacion Educativa. *Educacion Educativa.* [En línea] www.chasque.apc.org/.

- RETO, Lic JUAN GARCÍA. 2008. LA EVALUACIÓN DE LA FUNCIÓN DOCENTE DEBE SER CONSTRUCTIVA. *LA EVALUACIÓN DE LA FUNCIÓN DOCENTE DEBE*

SER CONSTRUCTIVA. [En línea] 17 de noviembre de 2008. [Citado el: 12 de abril de 2010.]

http://www.blogextremo.com/licjuangarciareto.

- Stufflebeam, Daniel. 2002. Evaluacion Sistematica : Guia Teorica Y Practica. *Evaluacion Sistematica : Guia Teorica Y Practica.* s.l. : Paidos Iberica, 2002.
- Teleña, Augusto Pila. Evaluacion Educativa. *Evaluacion Educativa.* s.l. : Editorial Verbum,S.L.
- Viada, Emiliano. 2010. hasheado.com . *hasheado.com .* [En línea] 19 de enero de 2010. [Citado el: 25 de abril de 2010.] http://www.hasheado.com/doctrine-vs-propel.html.
- web2. *web2.* [En línea] [Citado el: 12 de febrero de 2010.]

  http://www.maestrosdelweb.com/editorial/web2/.
- webdocs.cs.ualberta.ca. *webdocs.cs.ualberta.ca.* [En línea]

  http://webdocs.cs.ualberta.ca/~pfiguero/soo/metod/requerimientos.html.
- Wikilibros. *Wikilibros.* [En línea] http://es.wikipedia.org/wiki/PHP.
- Wikilibros. *Wikilibros.* [En línea]

  http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_PHP.
- www.mastermagazine.info. *www.mastermagazine.info.* [En línea] [Citado el: 20 de enero de 2010.] http://www.mastermagazine.info/termino/6400.php.
- www.welie.com. *www.welie.com.* [En línea] [Citado el: 06 de Mayo de 2010.] http://www.welie.com/patterns.

GLOSARIO DE TÉRMINOS

Moodle: Entorno de Aprendizaje Dinámico Orientado a Objetos y Modular es un Ambiente Educativo Virtual, sistema de gestión de cursos, de distribución libre, que ayuda a los educadores a crear comunidades de aprendizaje en línea. Este tipo de plataformas tecnológicas también se conoce como LMS (Learning Management System).

Bash: es un programa informático cuya función consiste en interpretar órdenes. Está basado en la shell de Unix y es compatible con POSIX. Fue escrito para el proyecto GNU y es el intérprete de comandos por defecto en la mayoría de las distribuciones de Linux.

Shell: el Shell es un programa interpretador de comandos que lee cada comando que ingresa el usuario y dispone lo necesario para que se ejecuten.

EVA: Entorno Virtual del Aprendizaje

Akademos: Sistema de Gestión académica de la Universidad de las Ciencias Informáticas

Framework: es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Symfony: es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web.

MVC: Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

UML: Lenguaje Unificado de Modelado (LUM) o (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software

PostgreSQL: es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

UUID: Es un identificador único universal

Visual Paradigm: Es una herramienta CASE que utiliza “UML”: como lenguaje de modelaje.

Glosario de términos

CASE: (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

RUP: El Proceso Unificado de Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Doctrine: es un ORM para PHP que facilita la labor de desarrollo de aplicaciones web, gracias a la capa que transforma el tratamiento de la BD mediante objetos, con la que se puede recuperar, insertar y modificar datos.

ORM: (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

XML: es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

YAML: es un lenguaje de marcado "ligero" que permite especificar estructuras (tales como arrays) con menos caracteres que XML, de forma sencilla.

Casos de pruebas: son un producto de desarrollo de software que ayudan a validar y verificar las expectativas de los stakeholders.

Stakeholders: se puede definir como cualquier persona o entidad que es afectada por las actividades de una organización.

GiST: El Árbol de Búsqueda Generalizado (Generalized Search Tree, GiST) de PostgreSQL, mecanismo extensible de indexamiento, ha sido mejorado de manera que soporta concurrencia de alta velocidad, recuperabilidad y rendimiento de actualizaciones, que antes estaba disponible sólo para los índices B-Tree. GiST es el mecanismo central para el soporte de indexamiento total de texto, geoespacial (GIS) y de estructuras de árbol.

ANEXOS

Diseño de casos de prueba

Modificar Tareas

1.2

Control de Versiones

Fecha	Versión	Descripción	Autor
16/05/2010	1.0	Elaboración del DCP del requisito Modificar Tareas	Aracelis Ruiz M

Tabla de aprobación

No.	Fecha	Aprobado por:	Cargo	Firma
<#>	<dd/mm/aaaa>	<nombre>	<cargo>	<firma>

Reglas de seguridad

El que recibe el documento asume la custodia y control, comprometiéndose a no reproducir, divulgar, difundir o de cualquier manera hacer de conocimiento público su contenido, excepto para cumplir el propósito para el cual se ha generado.

1. Condiciones de ejecución

    - Se debe seleccionar el subsistema de Gestión de Evaluaciones Docentes
    - Se debe seleccionar la opción **/Gestionar Tareas**

2. Requisitos a probar

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar Tarea	Se Modifica una Tarea con los datos siguientes: nombre, tipo de Tarea , fecha de entrega y evaluación	EP 1.1: Modificar una Tarea introduciendo datos válidos.	• Se introducen los datos de la Tarea que se desea Modificar. • Se presiona el botón **Aceptar**. • Se muestra un mensaje: “Los datos han sido guardados satisfactoriamente.”.
		EP 1.2: Modificar una Tarea introduciendo datos inválidos.	• Se introducen datos inválidos. • Se presiona el botón **Aceptar**.
		EP 1.3: Modificar una tarea dejando campos requeridos en blanco.	• Se introducen los datos de la tarea dejando algún campo requerido en blanco. • Se presiona el botón **Aceptar**.
		EP 1.4: Cancelar.	• Se introducen o no los datos de la tarea • Se presiona el botón **Cancelar.**

3. Descripción de variable

No	Nombre de campo	Tipo	Válido	Inválido

1	Nombre	Cadena de caracteres	Sólo letras	Cualquier carácter extraño excepto letras.
2	Tipo de Tareas	Cadena de caracteres	Sólo letras.	Cualquier carácter extraño excepto letras.
3	Fecha de entrega	Date	<dd/mm/aaaa>	Sólo fecha.
4	Evaluación	Numérico	Sólo números	Cualquier carácter extraño excepto números.

4. Juegos de datos a probar

Id del escenario	**Escenario**	**Nombre**	**Tipo de tarea**	**Fecha de entrega**	**Evaluación**	**Respuesta del sistema**	**Resultado de la prueba**

EP 1.1	Se modifica una tarea con los datos siguientes: nombre, tipo de tarea, fecha de	V(prueba2)	V(cadena de caracteres)	V(3/02/ 2010	V(sólo números)	El sistema muestra una ventana de información: “Los datos han sido guardados satisfactoria	
		V(prueba1)	V(caracteres)	V(date)	2	El sistema muestra una ventana de información: “Los datos han sido guardados satisfactoria	
EP 1.2	Se modifica una tarea introduciendo datos inválidos	I(erf)	V(cadena de caracteres)	V(7/5/09	kjd	El sistema muestra una ventana de información: “Los datos introducidos no son correctos.”.	

		V(unica)	I(kek4)	V(No)	myk	El sistema muestra una ventana de información: “Los datos introducidos no son correctos.”.	
EP 1.3	Se modifica una tarea introduciendo datos inválidos	I(Vacío)	V(321)	V(No)		El sistema muestra una ventana de información: “Ha dejado datos requeridos en blanco.”.	
		V(Lle6no)	I(Vacío)	V(No)		El sistema muestra una ventana de información: “Ha dejado datos requeridos en blanco.”.	

		V(totallidad 7)	V(este)	I(Vacío)		El sistema muestra una ventana de información: “Ha dejado datos requeridos en blanco.”.	
		V(sóloest3)	V(cuando)	V(Si)		El sistema muestra una ventana de información: “Ha dejado datos requeridos en blanco.”	
EP 1.4	Cancelar.	NA	NA	NA		Se cancela la operación y se cierra la ventana.	

[Las celdas de la tabla contienen V, I, o NA. V indica válido, I indica inválido, y NA se debe proporcionar un valor del dato en este caso].

5. Registro de defectos y dificultades detectados

Requi sito	**No.**	**No confor**	**Aspecto correspor**	**Etapa de detección**	**Signifi cativa**	**No signif**	**Recomen ación**	**Estado NC**	**Respuesta del Equipo de desarrollo**

		midad	**diente**			**cativa**			
<Nombre del requisito>	<1>	<Descripción de la No conformidad> Se especifica de manera clara y precisa en que consiste la No conformidad utilizando un lenguaje técnico. Debe referirse a la palabra, el botón, al hiperví	<Descripción del Aspecto correspondiente> Se especifica de forma clara y precisa el lugar en que fue detectada la no conformidad de manera que sea fácil su hallazgo de ser necesario se puede apoyar la explicación con una	<Etapa de detección del error>	<X> Marcar con una x. Se consideran defectos significativos aquellos que afectan la funcionalidad de aplicación así como la nomenclatura de la mism	<X> Marcar con una x. Se consideran defectos no significativos aquellos que no afectan la funcionalidad del sistema: Diferenci	<X> Marcar con una x. Se consideran recomendaciones aquellas sugerencias que aporten una mejora a la aplicación.	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué]. NP: No procede <dd/mm/aaaa> Causas.

		nculo, comportamiento específico que se considera un defecto.	imagen anexada.		a: Incorrecta implementación o validación de los requerimientos. No implementación de un requerimiento funcional. Faltas de ortografía, Errores de concordancia y la mescla de	a de espacios entre botón. Diferencia de colores entre interface.		.] RA: Resuelta PD: Pendiente NP: No Procede <dd/mm/aaaa>	

					idiomas.				

Diseño de casos de prueba

Eliminar Tareas

1.3

Control de Versiones

Fecha	Versión	Descripción	Autor
18/05/2010	1.0	Elaboración del DCP del requisito Eliminar Tareas	Aracelis Ruiz M

Tabla de aprobación

No.	Fecha	Aprobado por:	Cargo	Firma
<#>	<dd/mm/aaaa>	<nombre>	<cargo>	<firma>

Reglas de seguridad

El que recibe el documento asume la custodia y control, comprometiéndose a no reproducir, divulgar, difundir o de cualquier manera hacer de conocimiento público su contenido, excepto para cumplir el propósito para el cual se ha generado.

1. Condiciones de ejecución

    - Se debe seleccionar el subsistema de Gestión de Evaluaciones Docentes

- Se debe seleccionar la opción **/Gestionar Tareas**

2. Requisitos a probar

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar Tarea	Se elimina una Tarea con los datos siguientes: nombre, tipo de Tarea , fecha de entrega y evaluación	EP 1.1: Eliminar una Tarea introduciendo datos válidos.	• Se introducen los datos de la Tarea que se desea Eliminar. • Se presiona el botón **Aceptar**. • Se muestra un mensaje: “Los datos han sido guardados satisfactoriamente.”.
		EP 1.2: Eliminar una Tarea introduciendo datos inválidos.	• Se introducen datos inválidos. • Se presiona el botón **Aceptar**.
		EP 1.3: Eliminar una tarea dejando campos requeridos en blanco.	• Se introducen los datos de la tarea dejando algún campo requerido en blanco. • Se presiona el botón **Aceptar**.
		EP 1.4: Cancelar.	• Se introducen o no los datos de la tarea • Se presiona el botón **Cancelar.**

3. Descripción de variable

No	Nombre de campo	Tipo	Válido	Inválido

1	Nombre	Cadena de caracteres	Sólo letras	Cualquier carácter extraño excepto letras.
2	Tipo de Tareas	Cadena de caracteres	Sólo letras.	Cualquier carácter extraño excepto letras.
3	Fecha de entrega	Date	<dd/mm/aaaa>	Sólo fecha.
4	Evaluación	Numérico	Sólo números	Cualquier carácter extraño excepto números.

4. Registro de defectos y dificultades detectados

Requisito	**No.**	**No conformidad**	**Aspecto correspondiente**	**Etapa de detección**	**Significativa**	**No significativa**	**Recomendación**	**Estado NC**	**Respuesta del Equipo de desarrollo**
<Nombr	<	<Descripción	<Descripción del	<Etapa de	<X>	<X>	<X>	[Se coloc	[Esta columna se comienza a llenar a

e del requisito>	1>	de la No conformidad> Se especifica de manera clara y precisa en que consiste la No conformidad utilizando un lenguaje técnico. Debe referirse a la palabra, el botón, al hipervínculo, comportamiento	Aspecto correspondiente> Se especifica de forma clara y precisa el lugar en que fue detectada la no conformidad de manera que sea fácil su hallazgo de ser necesario se puede apoyar la explicación con una imagen anexada.	detección del error>	Marcar con una x. Se consideran defectos significativos aquellos que afectan la funcionalidad de aplicación así como la nomenclatura de la misma: Incorrecta imple	Marcar con una x. Se consideran defectos no significativos aquellos que no afectan la funcionalidad del sistema: Diferencia de espacios	Marcar con una x. Se consideran recomendaciones aquellas sugerencias que aporten una mejora a la aplicación.	a el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta	partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué]. NP: No procede <dd/mm/aaaa> Causas.

		específico que se considera un defecto.			mentación o validación de los requerimientos. No implementación de un requerimiento funcional. Faltas de ortografía, Errores de concordancia y la mescla de idiomas.	entre botón. Diferencia de colores entre interface.		PD: Pendiente NP: No Procede <dd/mm/aaaa>	