Universidad de las Ciencias Informáticas

Facultad 1

Título: Análisis y Diseño del Subsistema Matrícula. Sistema Automatizado para la Gestión Académica – "Akademos 2.0".

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autora: Catherine Muñoz Velázquez

Tutora: Ing. Marianny Hernández Batista

Consultante: DrC. Fátima Addine Fernández

Junio 2008

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Facultad 1 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Catherine Muñoz Velázquez

_____

Ing. Marianny Hernández Batista

_____

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Análisis y Diseño del Subsistema Matrícula. Sistema Automatizado para la Gestión Académica - ¨Akademos 2.0¨.

Autora: Catherine Muñoz Velázquez.

La tutora del presente Trabajo de Diploma considera que durante su ejecución la estudiante mostró las cualidades que a continuación se detallan:

La estudiante desarrolló el análisis y diseño que posibilitará la posterior implementación de un sistema que gestione todos los procesos de matrícula que se llevan a cabo en los centros docentes. El trabajo fue realizado con mucha independencia y responsabilidad por parte de la diplomante, la cual demostró en todo momento estar preparada para asumir correctamente las tareas orientadas. La originalidad, dedicación, laboriosidad y sencillez fueron cualidades que se mantuvieron presente durante todo el desarrollo de la investigación. La comunicación con la tutora fue positiva y constante, demostrando interés en aspectos relacionados con el desarrollo del trabajo.

La alta calidad científico-técnica, se evidencia en la calidad de la investigación realizada, demostrando un alto dominio de los aspectos actuales, apoyándose en una amplia búsqueda bibliográfica; realizando profundos análisis y arribando a conclusiones que proveen el trabajo de un alto nivel científico.

El documento presentado tiene una estructura adecuada, hace un buen uso del lenguaje y refleja de manera clara y concisa todas las etapas desarrolladas durante la investigación. El trabajo contiene resultados que poseen valor para ser presentados en eventos y talleres científicos.

Por todo lo anteriormente expresado considero que la estudiante está apta para ejercer como Ingeniera en Ciencias Informáticas; y propongo que se le otorgue al Trabajo de Diploma la calificación de 5 puntos.

Ing. Marianny Hernández Batista.

_____ _____

Firma. Fecha.

AGRADECIMIENTOS

Agradezco a mis padres Ena Elsa y Osmel que sin ellos no habría sido posible estar aquí.

Agradezco a mi amor Joe por estar conmigo durante estos años en la universidad.

A toda mi familia que siempre estuvo atenta a mis estudios.

A Fátima por toda la ayudad incondicional que me ha dado.

A todos mis compañeros del proyecto, por la colaboración que ha habido entre nosotros.

A mi tutora Marianny por guiarme cada vez que tuvo tiempo.

Agradezco a la Revolución por darme la oportunidad de graduarme.

DEDICATORIA

Esta tesis está dedicada a la memoria de mi tía Elena, que siempre me apoyó en todo momento que lo necesité, a pesar de estar lejos.

RESUMEN

Actualmente en la Universidad de las Ciencias Informáticas existe el Sistema de Gestión Académica **Akademos,** formado por varios subsistemas. En esta investigación se centrará la atención en el Subsistema Matrícula, que hasta ahora, si bien ha resuelto muchas necesidades de la Universidad, resulta ineficaz, ya que sus funcionalidades excluyen muchos requerimientos de los usuarios finales y las cualidades que debe tener el software. Las herramientas usadas para el desarrollo del software incumplen con las políticas de migración a software libre establecidas por Cuba, en tanto son propietarias.

En el diagnóstico realizado se evidenció que las deficiencias fundamentales que presenta el subsistema son: funcionalidades redundantes, la manera en que está implementado hace muy difícil el mantenimiento del mismo y falta la documentación que avale el trabajo realizado.

Se propone el diseño para el desarrollo de una versión mejorada del Subsistema Matrícula de Akademos 2.0 que cumpla los requerimientos de las Universidades en sentido general, y con las cualidades que debe cumplir el subsistema, lo que garantizará que el presente resultado sirva de guía para la futura implementación del mismo.

Para el diseño del subsistema se utilizarán herramientas que respeten la política de migración a software libre de Cuba, tal es el caso de la herramienta de modelado Visual Paradigm, guiados por la metodología RUP y el lenguaje de modelado UML. También se tendrán en cuenta el lenguaje de programación PHP, y el Framework Symfony para elaborar un diseño superior al existente.

El diseño propuesto es superior al diseño actual del subsistema matrícula, porque tiene en cuenta todos los requerimientos funcionales y no funcionales, tan necesarios para el desarrollo de un software con calidad. Permitirá llevar a cabo la implementación del subsistema de manera organizada, lo que garantiza que se desarrolle de manera eficaz.

PALABRAS CLAVES: Matrícula, gestión académica, Akademos.

ÍNDICE

INTRODUCCIÓN 1

1 ANTECEDENTES. METODOLOGÍA, LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DEL SUBSISTEMA MATRÍCULA 7

1.1. Antecedentes y estado actual del Sistema de Gestión Académica –"AKADEMOS" 7
1.1.1. Ámbito Internacional 7
1.1.2. Ámbito Nacional 10
1.1.3. Estado actual del Sistema de Gestión Académica –"Akademos" 12
1.1.3.1. Módulos del Sistema de Gestión Académica Akademos 13
1.1.3.2. Subsistema de Matrícula del Sistema de Gestión Académica "Akademos" 14
1.2. Metodología, lenguajes y herramientas para el desarrollo de la nueva versión del subsistema matrícula. 15
1.2.1. Metodología, lenguaje y herramienta para realizar el diseño del subsistema matrícula .. 15
1.2.2. Lenguajes y herramienta que influyen en el diseño del subsistema matrícula. 20
Conclusiones del capítulo I 22
2 CARACTERÍSTICAS DEL SUBSISTEMA MATRÍCULA 24

2.1. Negocio 24
2.1.1. Procesos que engloba el subsistema matrícula. 24
2.1.2. Reglas del Negocio 25
2.1.3. Actores del negocio 25
2.1.4. Trabajadores del negocio 25
2.1.5. Diagrama de casos de uso del negocio 26
2.1.6. Descripción de los casos de uso del negocio 27
2.2. Requerimientos 30
2.2.1. Propuesta de funcionalidades para el subsistema matrícula 30
2.2.2. Requerimientos funcionales 31
2.2.3. Requerimientos no funcionales 38
2.2.4. Diagrama de casos de uso 45
2.2.5. Actores del sistema 46
2.2.6. Descripción de los casos de uso del sistema. 46

2.3. Planificación y estimación de proyecto 48
Conclusiones del capítulo II 51
3. ANÁLISIS Y DISEÑO DEL SUBSISTEMA MATRÍCULA 53
3.1. Análisis 53
3.1.1. Realización de los casos de uso 54
3.2. Diseño 57
3.2.1. Patrones de diseño 58
3.2.2. Realización de los casos de uso 60
3.2.3. Diagrama de clases persistentes 65
3.2.4. Modelo de datos 66
3.2.5. Diagrama de despliegue 66
Conclusiones del capítulo III 67
CONCLUSIONES 68
RECOMENDACIONES 69
BIBLIOGRAFÍA 70
GLOSARIO 72

ÍNDICE DE TABLAS

Tabla 2-1 Actores del Negocio 25
Tabla 2-2 Trabajadores del Negocio 25
Tabla 2-3 Descripción del caso de uso del negocio Formar Prematrícula 27
Tabla 2-4 Descripción del caso de uso del negocio Solicitar Matrícula 27
Tabla 2-5 Descripción del caso de uso del negocio Ratificar Matrícula 28
Tabla 2-6 Descripción del caso de uso del negocio Reingresar 28
Tabla 2-7 Descripción del caso de uso del negocio Causar Baja 29
Tabla 2-8 Descripción del caso de uso del negocio Solicitar Licencia de Matrícula 29
Tabla 2-9 Descripción del caso de uso del negocio Solicitar Traslado 30
Tabla 2-10 Descripción del caso de uso del negocio Graduar Estudiante 30
Tabla 2-11 Actores del sistema 46
Tabla 2-12 Descripción resumida del CUS Gestionar Estudiante 46
Tabla 2-13 Descripción resumida CUS Buscar Estudiantes 47
Tabla 2-14 Descripción resumida del CUS Gestionar Estados 47
Tabla 2-15 Descripción resumida del CUS Realizar Movimiento 48
Tabla 2-16 Descripción resumida del CUS Matricular Estudiante 48
Tabla 2-17 Factor de peso de los actores sin ajustar 49
Tabla 2-18 Factor de peso de los caso de uso sin ajustar 49
Tabla 2-19 Factor de complejidad técnica 50
Tabla 2-20 Factor ambiente 50
Tabla 2-21 Estimación del esfuerzo total del proyecto 51

ÍNDICE DE FIGURAS

Figura 2-1 Diagramas de Casos de uso del Negocio 26
Figura 2-2 Diagrama de casos de uso del Sistema 45
Figura 3-1 Diagrama de clases del análisis CUS Gestionar Estudiantes 54
Figura 3-2 Diagrama de clases del análisis CUS Buscar Estudiante 55
Figura 3-3 Diagrama de clases del análisis CUS Gestionar Estados 55
Figura 3-4 Diagrama de clases del análisis CUS Aplicar Movimiento 56
Figura 3-5 Diagrama de clases del análisis CUS Matricular Estudiante 57
Figura 3-6 Diagrama de clases del diseño CUS Gestionar Estudiantes 60
Figura 3-7 Diagrama de clases del diseño CUS Buscar Estudiante 61
Figura 3-8 Diagrama de clases del diseño CUS Gestionar Estados 62
Figura 3-9 Diagrama de clases del diseño CUS Aplicar Movimiento 63
Figura 3-10 Diagrama de clases del diseño CUS Matricular Estudiante 64
Figura 3-11 Diagrama de clases persistentes 65
Figura 3-12 Modelo de datos 66
Figura 3-13 Diagrama de despliegue 67

INTRODUCCIÓN

Desde la antigüedad ya existían universidades. Las civilizaciones más antiguas han tenido centros de altos estudios, por ejemplo, en China se encuentra registrada la existencia de la escuela superior imperial durante el periodo YU (2257 adC-2208 AC), en Pakistán la Universidad de Takshashila, fundada alrededor del siglo VII AC, en Grecia Platón fundó La Academia en el año 387 AC. Estas universidades y muchas otras son el precedente de la universidad moderna. Se conoce que algunas de estas universidades entregaban títulos a sus graduados, y aun hoy se encuentran registros de sus alumnos más ilustres. Muchas de ellas vinculadas con la medicina hacían exámenes rigurosos que solo los que los aprobaban podían ejercer la profesión.

La única manera de tener el control de los estudiantes que pertenecían a la Universidad, hacer los títulos para los graduados y tener el control de quienes aprobaban o no los exámenes tanto en ese tiempo como en el actual, es mediante el proceso de matrícula, que es el que permite obtener los datos de los estudiantes, de interés para la institución.

Debido al gran aumento que tienen las universidades en cantidad de estudiantes, así como de nuevas especialidades que van surgiendo con el desarrollo del hombre se les hace más difícil poder tener el control de tantos datos que se necesitan conocer, es por esta razón que han surgido sistemas informáticos dedicados a la gestión de la información académica, sobre todo en las Universidades.

Todo sistema de gestión académica debe tener la funcionalidad que permita la matrícula de los estudiantes, ya que es el primer paso que debe llevar a cabo cada estudiante que desee iniciar sus estudios en cualquier centro. También se deben tener en cuenta todos aquellos procesos que puedan cambiar el estado del estudiante, estos procesos se refieren al traslado de los estudiante, dar baja a un estudiante u otros que desee definir cada centro en particular.

La matrícula es lo que permite al sistema hacer otras operaciones tales como asignar planes de estudio a los estudiantes matriculados, generar los grupos que existirán en el centro de estudio a partir de la matrícula real, tener un control de las notas de las asignaturas que están cursando en ese momento los estudiantes.

Todos los procesos de matrícula a su vez deben tener un respaldo legal y estos documentos legales deben estar en el expediente académico de cada estudiante como constancia de su estancia en el centro.

Situación problemática

Entre los módulos del Sistema Automatizado para la Gestión Académica --“Akademos” está el Subsistema Matrícula que presenta las funcionalidades de pre matrícula y matrícula de los estudiantes, permite hacer traslados, bajas o reingresos, entre otros movimientos que se deseen definir.

Este subsistema ha resuelto las necesidades hasta el momento de la Universidad pero ineficazmente. Presenta funcionalidades redundantes como es el caso de Gestionar Movimiento, al cambiar de estado un estudiante, no permite llenar todos los documentos que avalan legalmente la operación. La base de datos del subsistema en cada matrícula que se realiza en la Universidad crece, ya que al modificar el formulario usado para este fin, se genera una nueva tabla en la base de datos. La documentación que existe, descarta funcionalidades que están implementadas en el subsistema, trayendo como consecuencia que los nuevos miembros del proyecto les falte la documentación por la cual puedan comprender el funcionamiento del mismo. El subsistema fue desarrollado con herramientas propietarias, por tanto viola las políticas de migración que se han trazado la Universidad y el país, es el caso de la herramienta de desarrollo Visual Studio.Net y el Gestor de Base de Datos SQL Server 2000.

Problema Profesional

¿Cómo perfeccionar las funcionalidades existentes del Módulo de Matrícula del Sistema Automatizado para la Gestión Académica --“Akademos”?

Objeto de estudio

Subsistema de Matrícula del Sistema Automatizado para la Gestión Académica --“Akademos”.

Campo de Acción

Las funcionalidades del Subsistema de Matrícula.

Objetivo General

Proponer un diseño mejorado para el Subsistema de Matrícula del Sistema de Gestión Académica Akademos 2.0 para satisfacer las necesidades de los usuarios.

Preguntas Científicas

¿Cuáles son los antecedentes y fundamentos del Subsistema Matrícula?

¿Cuál es la correspondencia entre las funcionalidades del subsistema y las necesidades del los usuarios finales?

¿Qué restricciones tendría el diseño del Subsistema Matrícula?

Tareas de Investigación

- Elección de los fundamentos teóricos del Subsistema Matrícula.
- Diagnóstico del estado actual de la correspondencia entre las funcionalidades del subsistema y las necesidades del usuario final.
- Elaboración de un diseño para el Subsistema Matrícula teniendo en cuenta las restricciones que influyen en el mismo.

Métodos científicos de investigación

Enfoque dialéctico materialista, posibilitó encontrar las contradicciones que existen entre las funcionalidades del sistema y las necesidades de los usuarios finales, ya que las funcionalidades que brinda actualmente el sistema no satisfacen al usuario final. Otra contradicción se establece en relación a las cualidades que debe cumplir el sistema para poder llegar a ser un sistema de calidad, que satisfaga a los usuarios.

Para llevar a cabo la investigación, se utilizaron diferentes métodos científicos, para, de esta manera darle cumplimiento a las tareas de investigación planteadas.

El análisis y la síntesis, permitieron el estudio y valoración del Sistema de Gestión Académica, en particular el subsistema matrícula y así determinar cómo mejorar sus funcionalidades.

Método histórico lógico, posibilitó identificar los antecedentes del Sistema de Gestión Académica Akademos, específicamente el subsistema matrícula, cuáles fueron las necesidades que provocaron el surgimiento, así como la existencia de otros Sistema de Gestión Académica, que brindan funcionalidades similares a las del subsistema matrícula, que llevan más tiempo de funcionamiento en el ambiente académico.

Método de la modelación, una de las técnicas usadas por los ingenieros para la construcción de sistemas, es el uso de modelos, con el objetivo de representar, los procesos que se desean automatizar, y también para representar la estructura interna del sistema, cómo va a ser la organización del mismo, y representar los componentes que lo conformarán, sus interrelaciones y características. Estos modelos son los que garantizan que nuevos miembros del equipo de desarrollo comprendan el alcance y el funcionamiento del sistema.

La metodología de desarrollo de software que se utilizó, divide el desarrollo de software en varias fases y cada una de ellas tiene varios entregables, siendo los diagramas modelados por los analistas los que más protagonismo alcanzan, tal es el caso, del diagrama de casos de uso del negocio, donde se modelan todos los procesos que realizan en la empresa o centro de estudio en este caso, los clientes del software. Está también el diagrama de casos de uso del sistema que modela las funcionalidades que va a tener el futuro sistema, además se modelan diagramas de clases del análisis que es la representación de las clases que podrían conformar el futuro sistema y se realiza teniendo en cuenta los requerimientos funcionales propuesto para el software y los diagramas de clases del diseño que viene siendo una abstracción de la implementación del software y se realiza el modelado tratando de cumplir los requerimientos no funcionales propuestos para el software. Existen otro tipos de diagramas que representan el comportamiento dinámico del software, tal es el caso de los diagramas de interacción, en el cual se representan los mensajes que se envían de un objeto, clase o componente a otro representando las interacciones entre los mismo.

Con la utilización de este método se logra darle cumplimiento a la tarea relacionada con la elaboración del diseño del subsistema.

Método Sistémico, para poder comprender el subsistema, fue necesario ver la manera en que está implementado, teniendo que interactuar con las clases que lo componen, que son las que brindan las funcionales actuales del subsistema. Es importante comprender cuáles son las funcionalidades que están presentes en cada clase, con qué otras clases se relacionan e interactúan, y de qué manera interactúa con los demás subsistemas que compone el Sistema de Gestión Académica Akademos.

Dentro de los **métodos empíricos** se utilizaron el **método de la observación** y **la entrevista**. La observación se realizó a algunos usuarios del sistema, fundamentalmente las secretarias del centro que son las que tienen acceso a todas las funcionalidades que brinda el subsistema matrícula y algunas personas que participaron en el desarrollo del proyecto. Esto permitió apreciar las características externas del sistema y cómo es el funcionamiento visto desde la interfaz.

La entrevista brindó un intercambio fructífero con los usuarios finales, conociendo sus necesidades, y cuáles son los inconvenientes que se les han presentado utilizando el sistema actual. Permitió valorar las propuestas de mejoras de los procesos que realizan y codecidir en las mismas.

El informe final está estructurado en tres capítulos, conclusiones, recomendaciones, 46 anexos y un glosario de términos.

El primer capítulo aborda los antecedentes de los Sistemas de Gestión Académica. Se fundamenta el uso de metodologías y herramientas que se tendrán en cuenta para la realización del diseño a proponer.

En el capítulo dos se describen los procesos que integran el subsistema matrícula, y se hace una propuesta de las funcionalidades que brindará la nueva versión. También se incluye el estudio de cálculo del costo por puntos de casos de uso.

El capítulo tres fundamenta las fases de análisis y diseño, donde se incluyen los modelos más importantes de ambas fases.

La estructura del informe de investigación se realizó teniendo en cuenta los flujos de trabajo que propone la metodología de desarrollo de software RUP.

LA INVESTIGACIÓN es de mucha importancia ya que ofrece una solución a las inconformidades que tienen los usuarios finales con respecto al subsistema, teniendo en cuenta los resultados obtenidos en

otros centros de estudios tanto en Cuba como a nivel internacional, estos últimos investigados a través de Internet. El diseño que se propone se hizo teniendo en cuenta tecnologías libres y de esta manera cumplir con la política de migración a software libre. Otra ventaja a tener en cuenta es que el informe del trabajo va a servir de documentación al subsistema matrícula, y los futuros desarrolladores que se incorporen tendrán una guía por donde aprender cómo funciona el mismo. Estas ventajas en el diseño lo hacen superior al existente por lo que contribuye a la necesaria mejora de AKADEMOS.

La idea principal que respalda esta investigación, es la de proponer un diseño para poder implementar un software de calidad. La calidad del software, hay que centrarla en las necesidades del cliente. La autora se adscribe al criterio del autor Watt Humprey "if the software does not provide the right function when the user need them nothing else matters"(HUMPHREY, 2005). A lo que se refiere el autor es que aunque el software funcione, si no muestra los resultados como el cliente los quiere y cuando los quiere entonces no es un software con calidad y nada más importa.

Los autores que más contribuyeron a perfeccionar las funcionalidades del subsistema fueron Roger S. Pressman, Jacobson , Booch y Rumbaugh.

1 ANTECEDENTES. METODOLOGÍA, LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DEL SUBSISTEMA MATRÍCULA

En el siguiente capítulo se hará un estudio del arte, donde se pondrá a consideración las características de otros sistemas de gestión académica, así como los inconvenientes que presentan, que imposibilitan hacer uso de ellos. El estudio del arte se centrará en la matrícula de los alumnos a los centros de estudio.

Se tendrá en cuenta también otros sistemas utilizados en el ámbito nacional, sobre todo en la Universidad de las Ciencias Informáticas, en particular el sistema de Gestión Académica AKADEMOS.

Por último se hace énfasis en las herramientas y metodologías que se utilizarán para el diseño de la nueva versión del subsistema matrícula del Sistema de Gestión Académica AKADEMOS, que debe contribuir a mejorar sus funcionalidades.

1.1. Antecedentes y estado actual del Sistema de Gestión Académica –"AKADEMOS"

En el ámbito internacional, los sistemas de gestión académica, que se han encontrado tienen un subsistema matrícula, existen funcionalidades comunes como es el caso de la auto matrícula de los estudiantes, lo mismo desde Internet o dando la posibilidad en las propias escuelas, además de permitir también la matrícula en las secretarías. Estos sistemas vinculan a la matrícula con nóminas de pago, lo que posibilita generar cartas de pagos para los estudiantes. La preinscripción por lo general es usada cuando la universidad tiene plazas limitadas en alguna carrera o asignatura a matricular y por tanto deben hacer una asignación de asignaturas y grupos. El cambio de estado del estudiante en la universidad, no se menciona.

1.1.1. Ámbito Internacional

1.1.1.1. **AGORA**: Gestión integral de academias y centros de formación. (Sistema de gestión de alumnado)

Ágora es un sistema de gestión docente desarrollado por la empresa de soluciones informáticas para el sector docente, Kherian Soft, que radica en España.

El sistema abarca todas aquellas actividades que están vinculadas con el ambiente académico, y se extienden desde las pruebas de accesos y preinscripción de los alumnos a la gestión de los títulos pasando por la matrícula, gestión de actas, expedientes, estadísticas y gestión de becas, incluyendo la gestión económica de la Universidad relacionada con la actividad académica.(Ágora. Presentación del Sistema de Gestión de Alumnado)

Entre diferentes subsistemas que lo cauterizan, están el subsistema de preinscripción y el de matrícula.

El módulo de preinscripción es para aquellas universidades que tienen un número de plazas limitadas, y deben gestionar la asignación de plazas por estudios.

El módulo de matrícula, da la posibilidad a los estudiantes de auto matricularse y también hacerla desde las secretarías. Durante este proceso los estudiantes pueden seleccionar las asignaturas que desean cursar y además en todo momento les permite ver las asignaturas que pueden matricular en dependencia del estado de su expediente. La matrícula en este sistema ofrece la opción de seleccionar las formas de pago a usar, permite la generación de cartas de pago. También brinda la opción de horarios personalizados en dependencia de las asignaturas que haya matriculado.

Ágora permite la anulación y borrado de la matrícula.

El sistema está desarrollado en modo gráfico, que le permite agilidad y rapidez en el manejo, aunque esto dificulta la distribución del software cuando deban utilizarlo muchos usuarios a la vez.

1.1.1.2. **GAUSS:** Gestión Académica Universitaria sobre Sistemas Informáticos

El sistema informático para la gestión académica, GAUSS, es una herramienta desarrollada por la Universidad de Cantabria en conjunto con la empresa Semicrol, utilizando tecnologías de última generación, gestiona todos los procesos de preinscripción, matrícula, pruebas de acceso y expedientes.

El sistema de Gestión Académica Gauss realiza el proceso de matrícula en autoservicio en todas las universidades en las que está implantado, este proceso está dotado de un sistema experto que propone en forma dinámica toda la formación accesible al estudiante por cada asignatura que

selecciona. Además se aplican todas las restricciones establecidas (ciclo, curso, asignatura) de manera automática una vez seleccionada cada asignatura.

Cuando el estudiante se matricula obtiene de manera impresa el resguardo de matrícula y los documentos normalizados de pago bancario si no domicilia el pago de tasas.

Mientras esté abierto el plazo de matrícula, el estudiante podrá acceder a la información de la matrícula previamente realizada.

El sistema de Gestión Académica Gauss incorpora también el proceso de matrícula en la modalidad de escritorio que ha sido optimizado para facilitar el trabajo del personal administrativo. Dentro del mismo proceso se permite matricular, adaptar y solicitar convalidaciones, visualizar las tasas generadas, obtener el resumen económico de liquidación, y comprobar en todo momento el estado económico de una matrícula de forma integrada con el subsistema de gestión de tasas: documentos generados, ingresos realizados y devoluciones.

Se permite la modificación de la matrícula en escritorio tantas veces como el estudiante lo desee, lo que genera las tasas correspondientes.

La funcionalidad pre matrícula del sistema es usada cuando se tienen problemas para asignar los grupos y las asignaturas que se presentan a los estudiantes para matricular. Gauss permite realizar preinscripciones completas como si se tratase de una matrícula. El estudiante debe declarar los créditos que desea matricular y la lista de asignaturas y grupos dentro de ellas con criterios de preferencia. A partir de los criterios asignados por el estudiante se resuelve de forma automática la adjudicación de asignaturas y grupos y se formaliza la matrícula. Permite el autoservicio a través de Internet.

El sistema de Gestión Académica Gauss está implementado en .NET, que es una tecnología propietaria y no cumple con los planes se migración a software libre que se han trazado la Universidad y el país.

1.1.1.3. **SIGA**: Sistema de Información de Gestión Académica.

SIGA es un sistema que fue diseñado y puesto en marcha, en la Universidad de Chile y en la Universidad Técnica Federico Santa María. Se ha estado trabajando en él alrededor del año 2003.

El sistema de Información de Gestión Académica SIGA permite al usuario organizar los estudios en varios planes de estudios, cada plan en varios cursos y cada curso con varias asignaturas. Ajuste horario semanal y total, presupuesto y precio de las asignaturas/cursos. Los estudiantes se matriculan en estudios organizados en grupos y turnos (sesión).

Permite al estudiante preinscribir/inscribir asignaturas durante el proceso de matrícula de cada periodo académico. Una vez realizado este proceso, podrá consultar sobre las asignaturas inscritas efectivamente, e informarse sobre las inscripciones denegadas.

Hasta el presente existen varios sistemas con el nombre SIGA pero implementados para universidades diferentes, y con las características de la Gestión Académica de la Universidad donde se encuentra.

Las universidades brindan poca información sobre el sistema, y se desconocen las tecnologías utilizadas en la implementación del mismo, lo que constituye una limitante para el análisis crítico de este antecedente

1.1.2. Ámbito Nacional

1.1.2.1. **GESTACAD**: **Sistema de Gestión Académica**

Este sistema surgió con la idea de desarrollar un software que permitiera automatizar la gestión académica de la Universidad de Matanzas; el mismo gestiona una parte de la información académica de los estudiantes universitarios y la información de los profesores que forman parte del proceso docente educativo.

El sistema está concebido por módulos, entre los que se diferencian, los módulos de actualización de datos y el sitio Web o módulo de información, a través del cual se muestran las diversas salidas de la aplicación.

El módulo de actualización permite realizar la matrícula a los estudiantes de nuevo ingreso, otorgar baja, rematricular a un estudiante que ha causado baja, sin permitir, ratificar la matrícula, realizar traslados, ni registrar datos necesarios para los graduados.

El módulo de información permite buscar un estudiante, mostrar una estadística general de cuantos hay por criterios (facultad, centro de procedencia, vía de ingreso, provincia).

Al interactuar con este sistema se puede notar que existen problemas con la navegabilidad, el ambiente de trabajo resulta en ocasiones restringido e inflexible aspectos que recrudecen la búsqueda de información, la interfaz requiere ser más agradable, interesante y atractiva a la vista del usuario.

1.1.2.2. UCIMAT

El sistema UCIMAT fue implementado en el curso 2002/2003 con vista a agilizar el censo de estudiantes. Permite búsquedas de estudiantes por determinados criterios, incluye la matrícula y modificación de los datos de los estudiantes, así como reportes generales de los datos de los estudiantes que están matriculados en el sistema, impidiendo que se puedan realizar otras operaciones importantes entre los que se puede destacar el reingreso, registro de traslado u otros datos requeridos.

Este sistema desarrollado por profesores de la UCI, permite automatizar parte de la gestión académica de una Universidad Cubana de forma general, gestiona la información académica de los estudiantes universitarios y la información de los profesores que dirigen el proceso docente educativo.

El sistema está concebido por módulos, entre los que se diferencian, los de actualización de datos y el sitio Web, a través del cual se muestran las diversas salidas de la aplicación.

1.1.2.3. Sistema de gestión de la nueva universidad (SIGENU)

El Sistema de Gestión de la Nueva Universidad (SIGENU) se ha desarrollado con el fin de ser una herramienta que permita la gestión de toda la información académica vinculada con la educación superior en Cuba. En correspondencia con su carácter nacional, este sistema ha sido diseñado de manera tal que sea capaz de ofrecer una mayor seguridad e integridad de la información, y a la vez, ser tan flexible que permita ser adaptado a todos los centros de educación superior del país con sus diversas particularidades y maneras de realizar determinados procedimientos.

SIGENU es un sistema diseñado a nivel de la secretaría del centro de estudio, esto quiere decir que otros usuarios que forman parte importante del ámbito académico como los profesores y los estudiantes no pueden interactuar con él.

El sistema SIGENU está formado por varias áreas o módulos, tales como Matrícula, Control de Estudiantes, Planes de Estudio, Evaluaciones, Reportes y Seguridad.

El módulo de matrícula permite realizar la matrícula a los estudiantes. El módulo de control de estudiantes ofrece las funcionalidades del cambio de estado del estudiante en el centro de estudio, pero solo se pueden usar los estados que ya tiene predefinidos. No da la oportunidad de que el usuario pueda definir cuáles son los estados que se usan en la universidad. Tampoco permite gestionar los documentos que avalen el cambio de estado, al final este es un papeleo que deben realizar las secretarias y que el sistema podría facilitar.

En la actualidad se está valorando una futura integración de SIGENU con el Sistema de Gestión Académica AKADEMOS

1.1.3. Estado actual del Sistema de Gestión Académica –"Akademos"

El Sistema Akademos se pone en funcionamiento en la UCI en el curso 2004/2005. Este sistema se desarrolló teniendo en cuenta las siguientes pautas:

- El dinamismo de gestión académica constituye la principal fuente de riesgo para un sistema que intente automatizarlo.
- Se debe lograr que los principales trabajadores del sistema (directivos, secretarias, profesores, estudiantes) tengan un papel activo en el proceso de desarrollo.
- El plan de estudio es la entidad fundamental de la gestión académica y rige los demás procesos (matrícula, control, planificación).(ROJAS *et al.*)

El sistema se desarrolló con el IDE de desarrollo Visual Studio.Net, y el gestor de base de datos SQL Server 2000.

1.1.3.1. Módulos del Sistema de Gestión Académica Akademos

El sistema está formado por los módulos de plan de estudio, matrícula, expediente académico, reporte, registro del profesor y seguridad.

1.1.3.1.1. Plan de estudio

El plan de estudios es la entidad fundamental del sistema. Este se define en Akademos como una sucesión de niveles, divididos en momentos (semestres) que los estudiantes deben vencer una vez concluidas las asignaturas, las mismas se ubican en un orden de precedencia dado en los diferentes momentos del plan de estudio. Este módulo también permite la definición de los diferentes perfiles y disciplinas en las que se agrupan las asignaturas.

1.1.3.1.2. Expediente

El módulo de expediente es un repositorio digital de los documentos de los estudiantes (Figura 2). Los expedientes pueden almacenar documentos basados en plantillas, así como otros de libre formato o generados por el propio sistema. Cada vez que se crea un documento este es almacenado junto con otros datos, como su fecha de creación y el autor, para facilitar su posterior localización.

1.1.3.1.3. Registro del profesor

El módulo de registro es el encargado de permitir el control del desarrollo de un período académico. Uno de los principios sobre los que se basó el diseño de Akademos es que la información sea introducida al sistema por los mismos que la originan. Son los profesores los que aplican las diferentes formas de control del avance de los estudiantes, de ahí que, Akademos les permita directamente introducir los resultados de todas las evaluaciones que apliquen

1.1.3.1.4. Reporte

A los usuarios que interactúan con este módulo se les brinda la posibilidad de desarrollar reportes dinámicos, pudiéndolos diseñar de acuerdo a sus necesidades. Cuenta también con reportes estáticos más sencillos de la información que por lo general, siempre se utiliza.

1.1.3.1.5. Seguridad

Este es el módulo encargado de gestionar todos los usuarios que pueden hacer uso del sistema, así como de gestionar los permisos de los que se pueden hacer uso. También permite registrar las incidencias que se van generando de acuerdo a las operaciones que se realizan en el sistema.

1.1.3.2. Subsistema de Matrícula del Sistema de Gestión Académica "Akademos"

Son los estudiantes la razón de existencia de cualquier centro universitario. El control de los datos de estos es el objetivo fundamental del módulo de matrícula, así como la gestión de los movimientos a que estos son sometidos en su estancia en la universidad. Se define como movimiento cualquier acción que se realiza sobre un estudiante que provoque un cambio de su estado dentro de la institución, y genere un conjunto de información que debe quedar registrada en su expediente académico.

Potencialmente cada institución educativa es libre de decidir cuales mecanismos regirá la dinámica de los movimientos de los estudiantes. Por esta razón, el sistema Akademos permite la definición de los estados de los estudiantes, las transiciones entre los estados y los documentos que deben llenarse por el personal de secretaría para asentar los datos del movimiento, los cuales van a conformar el expediente digital del estudiante. El actual sistema solamente permite asociar un documento al movimiento, y se debe tener en cuenta que en el mundo académico un trámite de este tipo (traslado, baja, reingreso u otros), por lo general genera varios documentos que lo avalan legalmente. La definición de los cambios de estados y la asociación de los documentos genera una funcionalidad más, que resulta redundante, ya que al estado se le pueden asociar los documentos necesarios para el cambio así como los estados predecesores. La definición de los estados incluye la identificación de estos con un color determinado, que es utilizado para marcar a los estudiantes en todos los listados en que aparezca. De esta forma se facilita la identificación de los estudiantes.

En las instituciones educacionales que aceptan cada año miles de estudiantes, la tarea de realizar la matrícula de estos puede convertirse en una labor titánica, siendo necesario involucrar a una gran cantidad de personas en este proceso. El módulo de matrícula permite la descentralización del proceso de ingreso en un conjunto de personas tan grande como se decida. El sistema da la posibilidad de asignarles a los estudiantes su fotografía, lo que mejora su identificación en cada una de las tareas que se llevan a cabo. Para la asignación de la fotografía el sistema Akademos permite al **Sistema de**

Identificación UCI encargarse de la gestión de las mismas, el cual se integró sin ningún problema al sistema Akademos.

1.2. Metodología, lenguajes y herramientas para el desarrollo de la nueva versión del subsistema matrícula.

1.2.1. Metodología, lenguaje y herramienta para realizar el diseño del subsistema matrícula

1.2.1.1. Metodología de desarrollo de software

El sistema que se desarrollará tendrá un enfoque Orientado a Objetos (OO). Existen varias metodologías OO que utilizan UML que será el lenguaje de modelado a utilizar, dentro de este grupo se pueden mencionar METRICA3, Programación Extrema (XP), Proceso Unificado de desarrollo de software (RUP), entre otras.

1.2.1.1.1. MÉTRICA Versión 3

La metodología MÉTRICA Versión 3 ofrece a las organizaciones un instrumento útil para la sistematización de las actividades que dan soporte al ciclo de vida del software. Esta metodología tiene varios años de vida.

La metodología descompone cada uno de los procesos en actividades, y éstas a su vez en tareas. Para cada tarea se describe su contenido haciendo referencia a sus principales acciones, productos, técnicas, prácticas y participantes.

El orden asignado a las actividades no debe interpretarse como secuencia en su realización, ya que éstas pueden realizarse en orden diferente a su numeración o bien en paralelo. Sin embargo, no se dará por acabado un proceso hasta no haber finalizado todas las actividades del mismo, determinadas al inicio del proceso de desarrollo.

Así los procesos de la estructura principal de MÉTRICA Versión 3 son los siguientes:

- Planificación de sistemas de información

- Desarrollo de sistemas de información
- Mantenimiento de sistemas de información

Para la utilización de esta metodología se ha desarrollado una herramienta nombrada Gestor Metodológico.

No se utilizará esta metodología ya que el grupo de desarrollo no cuenta con ningún tipo de experiencia, ni teórica ni práctica en la misma, y robaría tiempo valioso en el desarrollo del software. Además de que no presenta ningún aval de manera internacional que demuestre su eficiencia.

1.2.1.1.2. Extreme Programing (XP)

XP es una metodología de desarrollo ligera, utilizada para proyectos de corto plazo, un equipo de desarrollo pequeño y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad reside en tener al usuario final del producto como parte del equipo de desarrollo.

Las características mencionada discrepan con las necesidades del proyecto, ya que el sistema de gestión académica que se desea desarrollar tiene un amplio alcance, trayendo como consecuencia que sea un proyecto de larga duración (un año como mínimo), el grupo de desarrollo con el que se cuenta es muy grande en contradicción con XP que es para grupos de desarrollo pequeños y es difícil integrar al equipo de desarrollo al menos una representación de los usuarios finales que esté todo el tiempo que sea necesario, por las características que tiene el proceso académico de las universidades en general.

1.2.1.1.3. Proceso unificado de desarrollo de software (RUP)

El Proceso Unificado es una metodología de desarrollo de software. Es un conjunto de actividades necesarias para convertir los requisitos del usuario en un sistema software.

RUP utiliza UML como lenguaje de modelado, para la construcción de los esquemas del software. UML representa una parte esencial de RUP de hecho fueron desarrollados de forma paralela.

RUP tiene tres características fundamentales, dirigidos por casos de usos, centrado en la arquitectura e iterativo e incremental.

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Un caso de uso representa las necesidades de los usuarios o como diríamos un requisitos funcional. Los casos de uso no sólo representan las funcionalidades del sistema, sino que también guían los flujos de diseño, implementación y pruebas, ya que existe trazabilidad entre los casos de uso y los artefactos más importantes que se generan en los flujos mencionados

La arquitectura del software incluye los aspectos dinámicos y estáticos más significativos del software y se describe mediante diferentes vistas del software en construcción.

Aunque los casos de uso guían el proceso de desarrollo no se desarrollan aisladamente, lo hacen a la vez que se desarrolla la arquitectura del sistema. Los casos de uso guían el desarrollo de la arquitectura y la arquitectura influye en la selección de los casos de uso.

Que sea iterativo e incremental permite dividir el proceso de desarrollo en varios mini proyectos. Al culminar cada mini proyecto es una iteración en el proceso de desarrollo que resulta en un incremento en el trabajo realizado. Permitiendo al grupo de desarrollo ver el avance del mismo y mantener motivados a los desarrolladores.

Las características que presenta el proyecto exige el uso de la Metodología RUP, ya que tiene toda una gama de roles que permite hacer la distribución del trabajo de una manera organizada y eficiente.

Teniendo en cuenta las características plasmadas anteriormente, se usará la metodología RUP para el diseño del software. Además de que el equipo de desarrollo cuenta con experiencia en el uso de la misma.

1.2.1.2. Lenguaje Unificado de Modelado (UML)

UML es un lenguaje de modelado que surgió de manera paralela a RUP, siendo esta una de las causas de que se integren bien. Se toma como estándar en noviembre del 1997 por miembros de la OMG.

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas.

UML ayuda al usuario final a entender la realidad de la tecnología y la posibilidad de que reflexione antes de invertir y gastar grandes cantidades en proyectos que no estén seguros en su desarrollo, reduciendo el costo y el tiempo empleado en la construcción de las piezas que constituirán el modelo.

Tiene partes estáticas, dinámicas, de entorno y organizativas. Está diseñado para ser utilizado con herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos. El lenguaje de modelado UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre los objetos. El comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos.

UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo. Contiene construcciones para representar decisiones de implementación y para elementos de tiempo de ejecución en componentes.

1.2.1.3. Herramienta de Modelado

En la actualidad existen varias herramientas de modelado. Entre las herramientas que usan la metodología RUP se encuentras Rational Rose Enterprise Edition y Visual Paradigm.

Ambas tienen el inconveniente de ser herramientas propietarias, pero en el caso del Visual Paradigm la Universidad de las Ciencias Informáticas cuenta con una licencia casi por tiempo indefinido. Visual Paradigm permite la generación de código PHP que es el lenguaje que se utilizará en el desarrollo de la nueva versión del sistema de gestión académica Akademos.

1.2.1.3.1. Rational Rose Enterprise Edition

Rational Rose es una herramienta de desarrollo y de modelado de componentes gráficos que utiliza el lenguaje de modelado estándar UML.

Soporta los lenguajes de programación ANSI C++, Rose J y patrones de Visual C++, Enterprise JavaBeans y permite ingeniería inversa para la mayoría de los constructores de Java.

Ofrece calidad en el código generado, posibilita la sincronización de modelo y código generado.

Permite el modelado Web, con los estereotipos definidos para este fin.

Permite el diseño de base de datos, con la habilidad de representar la integración de los datos y los requerimientos de la aplicación a través del diseño lógico y físico.

Permite la creación de documentos XML para usarlo en la construcción del software.

El análisis crítico reveló la existencia de los aspectos positivos ante señalados pero Rational tiene el inconveniente que no es multiplataforma, ya que solo se puede usar en Windows, además no permite la generación del código en lenguajes como PHP, que es el lenguaje que se va a usar para el desarrollo del software.

1.2.1.3.2. Visual Paradigm

Visual Paradigm es una herramienta de modelado que utiliza el lenguaje de modelado estándar UML, permite la generación de códigos e ingeniería inversa. Con una clase de diseño bien especificada Visual Paradigm puede generar código hasta en 15 lenguajes de programación entre ellos PHP que es el que se utilizará en el desarrollo del sistema. Visual Paradigm es una herramienta que actualmente cumple con las políticas de migración a software libre en Cuba, ya que es una herramienta multiplataforma que se puede utilizar tanto en Linux como en Windows. Tiene una interfaz muy intuitiva y es de fácil aprendizaje para los desarrolladores. Permite la generación automática de diagrama a partir de descripciones de casos usos, por ejemplo diagramas de secuencia, permitiendo la agilidad en el trabajo del analista. Permite además hacer la descripción de los casos de usos dando una gran variedad de plantillas predeterminadas permitiendo personalizarlas. Con Visual Paradigm los analistas pueden generar la documentación necesaria de los artefactos obtenidos hasta el momento en el proyecto.

1.2.2. Lenguajes y herramienta que influyen en el diseño del subsistema matrícula.

1.2.2.1. Lenguaje de programación Hypertext Preprocessor (PHP)

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje de programación del lado del servidor especialmente creado para el desarrollo de páginas Web dinámicas. Puede ser incluido con facilidad dentro del código HTML, y permite una serie de funcionalidades extraordinarias, que facilitan la construcción de aplicaciones WEB, por ejemplo el trabajo con cookies, sesiones de usuarios, entre otros.

Entre sus características fundamentales están:

- **Gratuito**: Al tratarse de software libre puede descargarse y utilizarse en cualquier aplicación personal o profesional, de manera completamente libre.
- **Enorme eficiencia**: Con escaso mantenimiento y un servidor gratuito puede soportar sin problemas millones de visitas diarias.
- **Sencilla integración con múltiples bases de datos**: Esencial para una página Web dinámica es una correcta integración con bases de datos. Aunque MySQL es la base de datos que mejor trabaja con PHP puede conectarse también a PostgreSQL, Oracle o cualquier otra base de datos compatible con ODBC (Open Database Connectivity Standard).
- **Versatilidad**: PHP puede usarse en la mayoría de los sistemas operativos, ya sea basados en UNIX (Linux, Solaris etc.), como con Windows.
- **Gran número de funciones predefinidas**: A diferencia de otros lenguajes de programación, PHP fue diseñado especialmente para el desarrollo de páginas Web dinámicas. Por ello está dotado de un gran número de funciones que simplifican enormemente tareas habituales como, descargar documentos, enviar correos, trabajar con cookies, sesiones etc.

El análisis realizado demuestra que es un lenguaje muy versátil, donde podemos programar en cualquier lugar, desde un editor de texto, hasta las herramientas más complejas con completamiento de código y librerías.

1.2.2.2. Lenguaje extensible de marcado de hipertexto (XHTML)

HTML en las siglas en inglés quiere decir lenguaje de marcado de hipertexto, es un estándar conocido en el mundo y cuyas normas define un organismo sin ánimo de lucro llamado World Wide Web Consortium, más conocido como W3C. El propio W3C define el lenguaje HTML como un lenguaje reconocido universalmente y que permite publicar información de forma global.

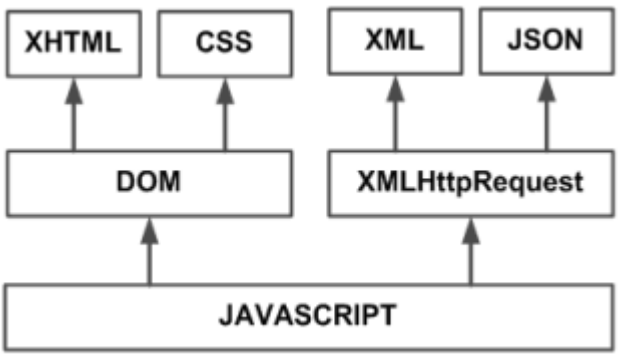
XML en sus siglas en inglés quiere decir lenguaje etiquetado extensible, y fue desarrollado por el mismo organismo que desarrollo HTML. En si XML no es un lenguaje en particular, sino que es una manera de definir lenguajes para diferentes necesidades uno de los lenguajes que usa XML para su definición es XHTML.

El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje de etiquetado extensible, XML. Las páginas creadas con XHTML son muy similares a las páginas creadas con HTML.

HTML es un lenguaje muy permisivo, que permite escribir sus etiquetas de maneras diferentes, esta flexibilidad podría parecer algo positivo, pero el resultado final es un documento muy desordenado, difícil de mantener y poco profesional. XHTM soluciona esto, estableciendo normas, para los programadores que usaran este lenguaje.

1.2.2.3. JavaScript Asíncrono+ XML (AJAX)

AJAX es la unión de varias tecnologías, está conformado por:



AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor. (PÉREZ)

Para la futura implementación del diseño que se va a proponer se hará uso de Ajax pero utilizando las bondades que permite el framework Symfony.

1.2.2.4. Framework Symfony

Un framework es una herramienta que moderniza el desarrollo de aplicaciones, automatizando muchos de los patrones utilizados para un propósito dado. Le añade estructura al código estimulando al desarrollador a escribir mejor, más legible y haciendo el código más sostenible. Se resume en que un framework simplifica la programación ya que empaqueta operaciones complicadas en declaraciones simples.

Symfony es un framework diseñado para optimizar el desarrollo de aplicaciones Web. Pone aparte las reglas del negocio de una aplicación Web, de la lógica del servidor, y puntos de vista de la presentación. Contiene numerosas herramientas y clases en las que se tuvieron en cuenta el desarrollo de aplicaciones web complejas. Automatiza tareas comunes con el objetivo de que el desarrollador pueda enfocarse en cuestiones específicas de la aplicación.

Fue escrito en PHP5. Ha sido probado a fondo en proyectos reales y está funcionando en sitios web de comercio electrónico de alta demanda. Es compatible con la mayoría de las bases de datos incluyendo MySQL, PostgreSQL, Oracle y Microsoft SQL. Funciona en plataformas UNIX y Windows.

Conclusiones del capítulo I

Los antecedentes del subsistema matrícula en el ámbito nacional tuvo en cuenta la existencia de sistemas que son antecesores directos de Akademos ya que fueron los primeros utilizados en la UCI, siendo ellos una de las razones por las que surgiera Akademos ya que ninguno lograba satisfacer las

necesidades reales de la Universidad y de los clientes. En el ámbito internacional se tuvo en cuenta sistemas que abarcaran los procesos fundamentales de la matrícula.

El uso de herramientas de modelado y de desarrollo que influyen en la propuesta del diseño constituye uno de los fundamentos así como el uso de metodologías de desarrollo, lenguajes de modelado, y de programación.

2 CARACTERÍSTICAS DEL SUBSISTEMA MATRÍCULA

El siguiente capítulo tiene tres momentos importantes, primero se van a describir los principales artefactos generados durante el flujo de trabajo de negocio, haciendo énfasis en la descripción de los principales procesos del negocio que se tendrán en cuenta para la propuesta de las funcionalidades que va a tener el subsistema matrícula.

En un segundo momento, se hará la descripción de los principales artefactos obtenidos durante el flujo de trabajo de requerimiento, se hará la propuesta de los requisitos funcionales que son las funcionalidades que debe cumplir el subsistema, y de los requisitos no funcionales, que serán las cualidades que debe cumplir el subsistema.

Por último se presenta la estimación obtenida del tiempo que demorará la terminación del subsistema, mediante el método de análisis de puntos de casos de usos.

Estos tres momentos representan los flujos de trabajo de Negocio y Requerimientos propuestos por la metodología de desarrollo de software RUP.

2.1. Negocio

2.1.1. Procesos que engloba el subsistema matrícula.

Cuando un estudiante desea entrar a una Universidad por lo general hacen una prueba de ingreso, si aprueba pasa a ser parte de la prematrícula de la Universidad, donde en esta primera etapa solo se saben los datos básicos del futuro estudiante. Luego se dirigen a la Universidad donde prematriculó a oficializar la matrícula, este el momento donde se hace un completamiento de los datos que necesita la institución sobre el estudiante. Cuando concluye el plazo de matrícula, se crean los grupos administrativos y se ubican los estudiantes en los mismos.

Existen otros procesos como son los traslados, bajas, licencia de matrícula, reingreso, en el caso de la universidad cubana que se realizan de la misma manera en todos los centros universitarios del país. Todos estos movimientos tienen algo similar, y es que para poder realizarlos se deben llenar una serie de documentos que luego se archivan en el expediente académico del estudiante constituyendo un respaldo legal del proceso que se realiza.

2.1.2. Reglas del Negocio

- Queda prohibido matricular asignaturas de un año académico superior al que le corresponda cursar.
- Para solicitar reingreso debe haber causado baja del centro.
- Un estudiante de primer año no puede pedir reingreso
- Para matricular un estudiante este debe estar prematriculado.

2.1.3. Actores del negocio

Actor	Descripción
Estudiante	Este actor representa al estudiante de la universidad
Posible Estudiante	Es el estudiante que está optando por ingresar a la Universidad.

Tabla 2-1 Actores del Negocio

2.1.4. Trabajadores del negocio

Trabajador	Descripción
Decano	Representa al decano de una facultad.
Secretaria General	Representa a la secretaria general de la facultad.
Secretaria Docente	Representa a las secretarias docentes de las facultades.
Rector	Representa al rector de la universidad.

Tabla 2-2 Trabajadores del Negocio

2.1.5. Diagrama de casos de uso del negocio

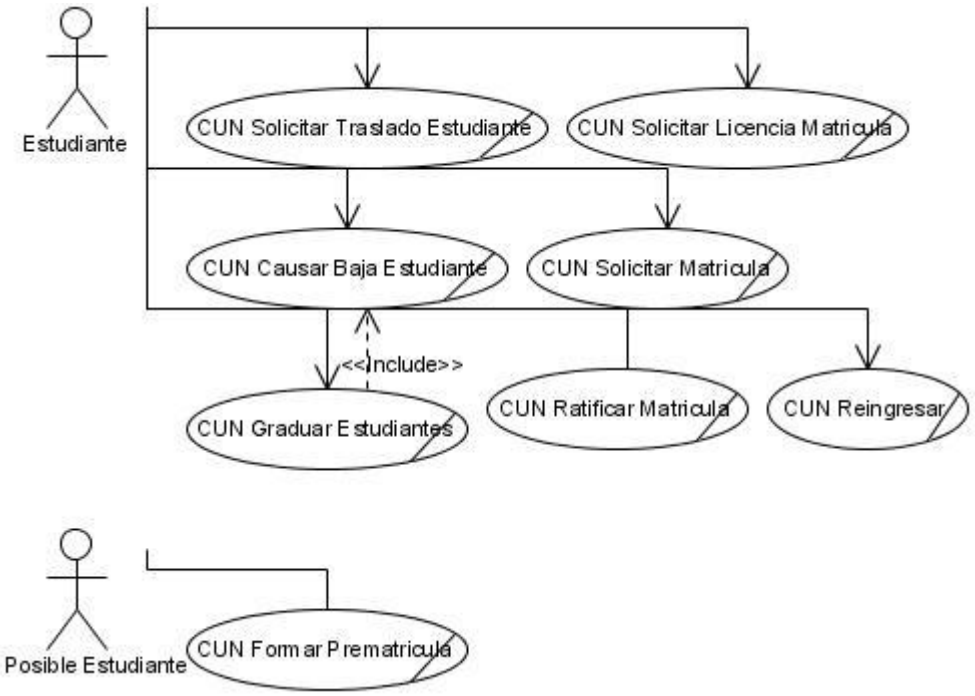


Figura 2-1 Diagramas de Casos de uso del Negocio

2.1.6. Descripción de los casos de uso del negocio

2.1.6.1. Formar prematrícula

Caso de uso del negocio	Formar prematrícula
Actores	Posible estudiante
Resumen	El caso de uso se inicia cuando se hace la convocatoria para el ingreso a la Universidad, los posibles candidatos hacen una test de ingreso, los seleccionados forman parte de la prematrícula de la Universidad. Con la prematrícula se crean los grupos académicos siguiendo diferentes criterios.
Casos de uso asociados	
Mejoras propuestas	

Tabla 2-3 Descripción del caso de uso del negocio Formar Prematrícula

2.1.6.2. Solicitar matrícula

Caso de uso del negocio	Solicitar Matrícula
Actores	Estudiante
Resumen	El caso de uso se inicia cuando el estudiante llega a la universidad y luego de tomarse la foto va a matricularse. Termina cuando concluye el tiempo para matricularse. Si es necesario se reorganizan los grupos.
Casos de uso asociados	
Mejoras propuestas	

Tabla 2-4 Descripción del caso de uso del negocio Solicitar Matrícula

2.1.6.3. Ratificar matrícula

Caso de uso del negocio	Ratificar Matrícula
Actores	Estudiante
Resumen	El caso de uso se inicia cuando el estudiante llega a secretaria general a ratificar la matrícula
Casos de uso asociados	
Mejoras propuestas	

Tabla 2-5 Descripción del caso de uso del negocio Ratificar Matrícula

2.1.6.4. Reingresar

Caso de uso del negocio	Reingresar
Actores	Estudiante
Resumen	El caso de uso se inicia cuando el estudiante después de haber causado baja se le permite reingresar a la universidad, terminando el caso de uso con una resolución emitida por el rector que certifica el reingreso del estudiante
Casos de uso asociados	
Mejoras propuestas	

Tabla 2-6 Descripción del caso de uso del negocio Reingresar

2.1.6.5. Causar baja

Caso de uso del negocio	Causar Baja
Actores	Estudiante
Resumen	El caso de uso se inicia cuando el estudiante se encuentra en alguno de estos casos, desaprueba más de 3 mundiales, incurre en alguna indisciplina grave, desea darse de baja voluntariamente o baja por deserción.
Casos de uso asociados	
Mejoras propuestas	

Tabla 2-7 Descripción del caso de uso del negocio Causar Baja

2.1.6.6. Solicitar licencia de matrícula

Caso de uso del negocio	Solicitar Licencia de Matrícula
Actores	Estudiante
Resumen	El caso de uso se inicia cuando el estudiante solicita una licencia de matrícula, da las razones de la misma, se hacen las verificaciones pertinentes y se le otorga o no la licencia
Casos de uso asociados	
Mejoras propuestas	

Tabla 2-8 Descripción del caso de uso del negocio Solicitar Licencia de Matrícula

2.1.6.7. Solicitar traslado

Caso de uso del negocio	Solicitar Traslado
Actores	Estudiante
Resumen	El caso de uso se inicia cuando el estudiante solicita el traslado de la universidad. Termina cuando le otorgan el traslado.
Casos de uso asociados	
Mejoras propuestas	

Tabla 2-9 Descripción del caso de uso del negocio Solicitar Traslado

2.1.6.8. Graduar estudiantes

Caso de uso del negocio	Graduar Estudiantes
Actores	Estudiante
Resumen	El caso de uso se inicia cuando el estudiante se gradúa.
Casos de uso asociados	Solicitar Baja
Mejoras propuestas	

Tabla 2-10 Descripción del caso de uso del negocio Graduar Estudiante

2.2. Requerimientos

2.2.1. Propuesta de funcionalidades para el subsistema matrícula

Después de las entrevistas realizadas con varias secretarías docentes de diferentes facultades de la Universidad de las Ciencias Informáticas, y las secretarías Generales de las Universidad de las Ciencias Informáticas y la Universidad de la Habana, así como la investigación realizada de otros

sistemas de gestión académica en el mundo, se llegó a la conclusión de lo importante que es el diseño del subsistema matrícula, y que el mismo cuente con las funcionalidades de prematrícula y matrícula de los estudiantes. Permitir modificar los datos de los estudiantes y hacer búsqueda de estudiante mediante varios criterios. Permitir la automatización de los procesos que comprenden los movimientos a los estudiantes (bajas, traslados, entre otros.). Para que el sistema sea genérico y no solo se asocie con estos tipos de estados, permitirá la gestión de los diferentes estados por los que puede pasar un estudiante durante su estancia en el centro de estudio.

A diferencia del Sistema de Gestiona Académica Akademos este nuevo sistema permitirá que cada movimiento tenga asociado más de un documento, y para poder aplicar un movimiento a un estudiante será necesario llenar cada uno de los documentos que este tenga asociado, de esta manera quedara asegurado que existan en el expediente virtual del estudiante cada uno de los documentos que avalen de manera legal la operación que se realice sobre él.

2.2.2. Requerimientos funcionales

2.2.2.1. R1 Gestionar datos del Estudiantes

2.2.2.1.1. Adicionar datos del estudiante

2.2.2.1.1.1. Verificar los permisos del usuario

2.2.2.1.1.2. Si tiene los permisos necesarios, debe mostrar la Interfaz Gestionar Estudiante que dará las opciones:

2.2.2.1.1.3. Adicionar datos de los estudiantes

2.2.2.1.1.4. Matricular un estudiante

2.2.2.1.1.5. Buscar estudiantes.

2.2.2.1.1.6. Prematrícula Masiva

2.2.2.1.1.7. Mostrar Interfaz con el formulario que se ha definido para pre matricular un estudiante.

2.2.2.1.1.8. Permitir llenar los campos del formulario con los datos del estudiante.

2.2.2.1.1.9. Validar los campos del formulario.

2.2.2.1.1.10. Si la validación de los datos es incorrecta muestra mensaje de error notificando el problema.

2.2.2.1.1.11. Permitir al usuario rectificar los datos del formulario.

2.2.2.1.1.12. Pre matricular el estudiante

2.2.2.1.1.13. Registrar Incidencia.

2.2.2.1.2. Modificar datos del Estudiante

2.2.2.1.2.1. Verificar los permisos del usuario

2.2.2.1.2.2. Si tienes los permisos necesarios mostrar Interfaz Gestionar Estudiante que dará las opciones:

2.2.2.1.2.3. Adicionar datos de los estudiantes

2.2.2.1.2.4. Matricular un estudiante

2.2.2.1.2.5. Buscar estudiantes.

2.2.2.1.2.6. Prematrícula Masiva

2.2.2.1.2.7. El buscador definido en la interfaz Gestionar datos del Estudiante permitirá realizar la búsqueda mediante los datos siguientes:

2.2.2.1.2.8. Nombre, Sexo, Código de Identificación (CI en nuestro país), Estructura (lugar del centro de estudio donde se encuentra ubicado el estudiante), País, Provincia, Municipio.

2.2.2.1.2.9. Los resultados mostrarán el nombre y la foto de los estudiantes.

2.2.2.1.2.10. Cada resultado debe permitir ver el expediente del estudiante.

2.2.2.1.2.11. El expediente del estudiante mostrará todos los documentos relacionados con el estudiante que se han generado hasta el momento.

2.2.2.1.2.12. Se irá al expediente del estudiante que se desee modificar.

2.2.2.1.2.13. Permitir al usuario abrir la hoja de matrícula para realizar las modificaciones.

2.2.2.1.2.14. Permitir modificar los campos del la hoja de matrícula

2.2.2.1.2.15. Validar todos los datos del formulario

2.2.2.1.2.16. Si la validación de los datos es incorrecta muestra mensaje de error notificando el problema.

2.2.2.1.2.17. Permitir rectificar los datos en el formulario.

2.2.2.1.2.18. Modificar los datos del estudiante.

2.2.2.1.2.19. Registrar Incidencia.

2.2.2.1.3. Prematrícula masiva

2.2.2.1.3.1. Verificar los permisos del usuario

2.2.2.1.3.2. Si tiene los permisos necesarios mostrar Interfaz Gestionar Estudiante que dará las opciones:

2.2.2.1.3.3. Adicionar datos de los estudiantes

2.2.2.1.3.4. Matricular un estudiante

2.2.2.1.3.5. Buscar estudiantes.

2.2.2.1.3.6. Prematrícula Masiva

2.2.2.1.3.7. Muestra interfaz que da la opción especificar fuente de donde se va a tomar los datos.

2.2.2.1.3.8. Si es un documento, permite especificar la dirección local del documento.

2.2.2.1.3.9. Verificar formato del documento

2.2.2.1.3.10. Si es un gestor de base de datos permite especificar la base de datos, la tabla y los atributos que se desea importar.

2.2.2.1.3.11. Se importan los datos de los estudiantes.

2.2.2.1.3.12. Se pone como estado prematriculado.

2.2.2.2. R2 Buscar estudiante

2.2.2.2.1. Verificar los permisos del usuario

2.2.2.2.2. Si tiene los permisos necesarios mostrar Interfaz Gestionar Estudiante que dará las opciones:

2.2.2.2.3. Adicionar datos de los estudiantes

2.2.2.2.4. Matricular un estudiante

2.2.2.2.5. Buscar estudiantes.

2.2.2.2.6. Prematrícula Masiva

2.2.2.2.7. El buscador definido en la interfaz Gestionar datos del Estudiante permitirá realizar la búsqueda mediante los datos siguientes:

2.2.2.2.8. Nombre, Sexo, Identificador (CI en nuestro país), Estructura (lugar del centro de estudio donde se encuentra ubicado el estudiante), País, Provincia, Municipio.

2.2.2.2.9. Los resultados mostrarán el nombre y la foto de los estudiantes.

2.2.2.2.10. Cada resultado debe permitir ver el expediente del estudiante.

2.2.2.2.11. Si el estudiante no se ha matriculado, dar la opción matricular.

2.2.2.2.12. El expediente del estudiante mostrará todos los documentos relacionados con el estudiante que se han generado hasta el momento.

2.2.2.2.13. Dar la opción de aplicar movimientos al estudiante.

2.2.2.2.14. Dar la opción de hacer cambios de estructura al estudiante.

2.2.2.3. R3 Gestionar datos de los estados

2.2.2.3.1. Adicionar datos de un Estado

2.2.2.3.1.1. Verificar los permisos del usuario

2.2.2.3.1.2. Si tiene los permisos necesarios mostrar una interfaz donde aparezcan los nombres de todos los estados definidos hasta el momento

2.2.2.3.1.3. Dar la opción de modificar los estados existentes.

2.2.2.3.1.4. Dar la opción de eliminar los estados existentes.

2.2.2.3.1.5. Dar la opción de crear un nuevo estado

2.2.2.3.1.6. El sistema mostrará un formulario donde permita poner nombre del estado y color del estado, estado padre, plantillas asociadas al estado.

2.2.2.3.1.7. Debe mostrar los colores disponibles para que no se repitan estados con el mismo color.

2.2.2.3.1.8. Validar los datos del formulario

2.2.2.3.1.9. Si los datos están incorrectos muestra mensaje de error, notifican el problema.

2.2.2.3.1.10. Permitir rectificar los datos del formulario.

2.2.2.3.1.11. Adicionar los datos del nuevo estado.

2.2.2.3.1.12. Registrar incidencia.

2.2.2.3.2. Modificar datos de un estado

2.2.2.3.2.1. Verificar los permisos del usuario

2.2.2.3.2.2. Si tiene los permisos necesarios mostrar una interfaz donde aparezcan los nombres de todos los estados definidos hasta el momento

2.2.2.3.2.3. Dar la opción de modificar los estados existentes.

2.2.2.3.2.4. Dar la opción de eliminar los estados existentes.

2.2.2.3.2.5. Dar la opción de crear un nuevo estado

2.2.2.3.2.6. Muestra un formulario donde permite cambiar nombre y color del estado, estados padre y plantillas asociadas.

2.2.2.3.2.7. Debe mostrar los colores disponibles para que no se repitan estados con el mismo color.

2.2.2.3.2.8. Validar los datos del formulario

2.2.2.3.2.9. Si los datos están incorrectos se muestra mensaje de error notificando el problema.

2.2.2.3.2.10. Permitir rectificar los datos del formulario.

2.2.2.3.2.11. Modificar los datos del estado.

2.2.2.3.2.12. Registrar incidencia.

2.2.2.3.3. Eliminar estado.

2.2.2.3.3.1. Verificar los permisos del usuario

2.2.2.3.3.2. Si tiene los permisos necesarios mostrar una interfaz donde aparezcan los nombres de todos los estados definidos hasta el momento

2.2.2.3.3.3. Dar la opción de modificar los estados existentes.

2.2.2.3.3.4. Dar la opción de eliminar los estados existentes.

2.2.2.3.3.5. Dar la opción de crear un nuevo estado

2.2.2.3.3.6. Verificar que no exista un estudiante que este en ese estado,

2.2.2.3.3.7. Si existe un estudiante en ese estado notifica al usuario, y no se elimina el estado.

2.2.2.3.3.8. Si no existe un estudiante que use el estado, muestra mensaje de confirmación, dando la posibilidad también de cancelar la operación.

2.2.2.3.3.9. Si cancela la operación vuelve a la página gestionar estados

2.2.2.3.3.10. Eliminar el estado

2.2.2.3.3.11. Registrar incidencia.

2.2.2.4. R4 Realizar movimiento

2.2.2.4.1. Realizar Movimiento

2.2.2.4.1.1. Verificar los permisos del usuario

2.2.2.4.1.2. Si tiene los permisos necesarios mostrar Interfaz Gestionar Estudiante que dará las opciones:

2.2.2.4.1.3. Adicionar datos de los estudiantes

2.2.2.4.1.4. Matricular un estudiante

2.2.2.4.1.5. Buscar estudiantes.

2.2.2.4.1.6. Prematrícula Masiva

2.2.2.4.1.7. El buscador definido en la interfaz Gestionar datos del Estudiante permitirá realizar la búsqueda mediante los datos siguientes:

2.2.2.4.1.8. Nombre, Sexo, Identificador (CI en nuestro país), Estructura (lugar del centro de estudio donde se encuentra ubicado el estudiante), País, Provincia, Municipio.

2.2.2.4.1.9. Los resultados mostrarán el nombre y la foto de los estudiantes.

2.2.2.4.1.10. Cada resultado debe permitir ver el expediente del estudiante.

2.2.2.4.1.11. El expediente del estudiante mostrará todos los documentos relacionados con el estudiante que se han generado hasta el momento.

2.2.2.4.1.12. Dar la opción de cambiar al estudiante de estructuras.

2.2.2.4.1.13. Cuando se está viendo el expediente de un estudiante la interfaz debe dar la opción de cambiar de estado al estudiante, mostrando los estados a los que puede cambiar el estudiante.

2.2.2.4.1.14. Mostrar el documento primario a llenar para llevar a cabo el cambio de estado.

2.2.2.4.1.15. Ir mostrando los demás documentos asociados (si es que los tiene) para ser llenados.

2.2.2.4.1.16. Cuando se termina de llenar un documento es que se debe mostrar el otro.

2.2.2.4.1.17. Cambiar de estado al estudiante cuando se llene el último documento asociado al estado al que se quiere cambiar.

2.2.2.4.1.18. Validar los datos de cada documento, asociado al movimiento.

2.2.2.4.1.19. Si los datos están correctos entonces se pasa al otro documento si es que hay.

2.2.2.4.1.20. Si es el último documento:

2.2.2.4.1.21. Cambiar de estado al estudiante.

2.2.2.4.1.22. Adjuntar los documentos al expediente

2.2.2.4.1.23. Registrar la incidencia.

2.2.2.4.1.24. Si los datos están incorrectos se muestra mensaje de error notificando el problema.

2.2.2.4.1.25. Permitir rectificar los datos del documento.

2.2.2.5. Cambiar de estructura interna

2.2.2.5.1. Verificar los permisos del usuario

2.2.2.5.2. Si tiene los permisos necesarios mostrar Interfaz Gestionar Estudiante que dará las opciones:

2.2.2.5.3. Adicionar datos de los estudiantes

2.2.2.5.4. Matricular un estudiante

2.2.2.5.5. Buscar estudiantes.

2.2.2.5.6. Prematrícula Masiva

2.2.2.5.7. El buscador definido en la interfaz Gestionar datos del Estudiante permitirá realizar la búsqueda mediante los datos siguientes:

2.2.2.5.8. Nombre, Sexo, Identificador (CI en nuestro país), Estructura (lugar del centro de estudio donde se encuentra ubicado el estudiante), País, Provincia, Municipio.

2.2.2.5.9. Los resultados mostrarán el nombre y la foto de los estudiantes.

2.2.2.5.10. Cada resultado debe permitir ver el expediente del estudiante.

2.2.2.5.11. El expediente del estudiante mostrará todos los documentos relacionados con el estudiante que se han generado hasta el momento.

2.2.2.5.12. Dar la opción de hacer cambios al estudiante de estructuras.

2.2.2.5.13. Mostrar interfaz que permita seleccionar la estructura destino hacia donde se cambiará el estudiante.

2.2.2.5.14. Mostrar las estructuras de manera jerárquica.

2.2.2.5.15. Efectuar el cambio deseado

2.2.2.5.16. Registrar incidencia de la operación que se ha hecho.

2.2.2.6. R7 Matricular estudiantes

2.2.2.6.1. Verificar los permisos del usuario

2.2.2.6.2. Si tiene los permisos necesarios mostrar Interfaz Gestionar Estudiante que dará las opciones:

2.2.2.6.3. Adicionar datos de los estudiantes

2.2.2.6.4. Matricular un estudiante

2.2.2.6.5. Buscar estudiantes.

2.2.2.6.6. Prematrícula Masiva

2.2.2.6.7. Si el estudiante está prematriculado el usuario podrá usar el buscador definido en la interfaz Gestionar datos del Estudiante que permitirá realizar la búsqueda mediante los datos siguientes:

2.2.2.6.8. Nombre, Sexo, Identificador (CI en nuestro país), Estructura (lugar del centro de estudio donde se encuentra ubicado el estudiante), País, Provincia, Municipio.

2.2.2.6.9. Los resultados mostrarán el nombre y la foto de los estudiantes.

2.2.2.6.10. Cada resultado debe permitir ver el expediente del estudiante.

2.2.2.6.11. Si el estudiante no se ha matriculado, dar la opción matricular.

2.2.2.6.12. Si el estudiante no se encuentra prematriculado llevar a cabo la matrícula a través de la opción dada por la interfaz Gestionar Estudiantes.

2.2.2.6.13. Mostrar el formulario definido para la matrícula del estudiante.

2.2.2.6.14. Permitir llenar los datos del formulario

2.2.2.6.15. Validar los datos del formulario

2.2.2.6.16. Si los datos no están correctos mostrar mensaje de error mostrando el problema.

2.2.2.6.17. Permitir rectificar los datos del formulario

2.2.2.6.18. Matricular el estudiante

2.2.2.6.19. Ubicar el estudiante en un grupo administrativo

2.2.2.6.20. Registrar la incidencia

2.2.3. Requerimientos no funcionales

2.2.3.1. Usabilidad

2.2.3.1.1. Los usuarios normales requerirán un tiempo de entrenamiento de 15 días.

2.2.3.1.2. Los usuarios avanzados requerirán un tiempo de entrenamiento de 7 días.

2.2.3.2. Estándares de usabilidad

2.2.3.2.1. Generales

2.2.3.2.1.1. Elegir imágenes claras y nítidas para los gráficos.

2.2.3.2.1.2. Limitar el número de enlaces en una página, como máximo 20 enlaces por página.

2.2.3.2.1.3. Evitar pequeños botones y enlaces con texto minúsculo

2.2.3.2.1.4. Dejar espacio entre los enlaces y botones

2.2.3.2.1.5. No utilizar menús en cascada (menús que se despliegan)

2.2.3.2.1.6. Proporcionar un título al formulario que exprese claramente su función.

2.2.3.2.2. Texto

2.2.3.2.2.1. Utilizar nomenclatura clara y familiar, sin tecnicismos ni extranjerismos

2.2.3.2.2.2. No utilizar preguntas complejas.

2.2.3.2.2.3. Redactar siempre las opciones de forma afirmativa

2.2.3.2.3. Organización

2.2.3.2.3.1. Organizar los campos en una sola columna de datos.

2.2.3.2.3.2. Agrupar, si es posible, los campos obligatorios al comienzo del formulario.

2.2.3.2.3.3. Evitar fragmentar la petición de información.

2.2.3.2.3.4. Proporcionar un diseño ordenado, alineando verticalmente todas las etiquetas y todos los campos entre sí.

2.2.3.2.3.5. Situar las respuestas de los campos *radio buttons* y *check box* después de los mismos.

2.2.3.2.4. Tipos de campos

2.2.3.2.4.1. El tamaño visible de los campos de texto debe corresponderse con la longitud del contenido que ha de introducir el usuario.

2.2.3.2.4.2. Evitar que las combos recarguen la página para rellenar otros campos, pero cuando así sea, asegúrese de que el formulario conserva el mismo estado que tenía antes de recargar la página: con los mismos campos visibles o activos, y con todos los campos rellenos con los mismos datos que antes de la recarga.

2.2.3.2.4.3. Seleccionar opción por defecto en los combos o *radio buttons*

2.2.3.2.5. Funcionamiento

2.2.3.2.5.1. Asegurar que la tecla "Intro" realiza la acción principal.

2.2.3.2.5.2. Evitar, mediante JavaScript, que el usuario pueda enviar dos veces el formulario.

2.2.3.2.6. Ayudas

2.2.3.2.6.1. Identificar claramente los campos obligatorios y los opcionales mediante el literal Obligatorio u Opcional.

2.2.3.2.6.2. Indicar los campos obligatorios cuando sean menos que los opcionales y viceversa.

2.2.3.2.7. Botones

2.2.3.2.7.1. En los formularios de un sólo paso evitar tener un botón **Cancelar** cuya función sea en realidad volver a la página anterior.

2.2.3.2.7.2. Dar nombre adecuado a los botones del formulario, relacionado con su acción y no de carácter general.

2.2.3.2.8. Errores

2.2.3.2.8.1. Cuando se produzca un error al rellenar el formulario proporcionar en la parte superior del mismo, y con suficiente contraste, un listado de los errores. Por cada error indique qué campo lo ha provocado, por qué motivo, cómo solucionarlo y un enlace al campo.

2.2.3.2.8.2. Cuando se produzca un error, el formulario no debe resetearse**.**

2.2.3.2.9. FeedBack

2.2.3.2.9.1. Cuando el usuario envíe el formulario, infórmele del resultado de su acción: indíquele si se ha realizado correctamente, qué datos se han enviado, cómo puede ponerse en contacto con los responsables del sitio si ha habido problemas o para hacer un seguimiento del mismo, o cómo puede modificar los datos enviados.

2.2.3.2.9.2. Incluir las cláusulas de protección de datos cuando sea pertinente.

2.2.3.2.9.3. Accesibilidad

2.2.3.2.9.3.1. Comprobar que el tabulador permite acceder a todos los campos en el mismo orden que el visual.

2.2.3.2.10. Formularios extensos

2.2.3.2.10.1. Si los formularios son muy extensos dividirlo en páginas bien rotuladas que indiquen al usuario en que paso está del proceso (por ejemplo Paso 3 de 4).

2.2.3.2.10.2. El usuario debe poder volver a los pasos anteriores.

2.2.3.2.10.3. Evitar la utilización de pestañas para crear formularios de varias páginas.

2.2.3.3. Fiabilidad

2.2.3.3.1. El sistema debe estar disponible las 24 horas del día y los 7 días de la semana.

2.2.3.3.2. El mantenimiento del sistema debe ser transparente para el usuario.

2.2.3.3.3. El tiempo medio permitido para que el sistema quede fuera de operación luego de haber fallado debe ser como máximo 7 días.

2.2.3.3.4. El tiempo medio entre fallos debe ser como máximo un año.

2.2.3.3.5. Las salidas del sistema deben ser de una alta precisión y exactitud de acuerdo a las necesidades del usuario.

2.2.3.3.6. La aplicación debe estar protegida de accesos no autorizados.

2.2.3.3.7. Los usuarios autenticados solo tendrán acceso a las áreas definidas para su rol.

2.2.3.3.8. Registrar todas las incidencias de los usuarios en el sistema.

2.2.3.3.9. La información generada por el sistema será objeto de un alto nivel de protección.

2.2.3.3.10. Los errores oscilaran entre 50 y 250 errores/MLC

2.2.3.3.11. Los errores se clasificaran en menores, significativos y críticos.

2.2.3.3.12. Menores: No resulta peligroso porque no causa daños significativos al sistema.

2.2.3.3.13. Significativos: Pueda causar daños mayores a la aplicación, pero si se actúa de manera correctiva inmediata puede lograrse la supervivencia del sistema.

2.2.3.3.14. Crítico: Puede causar daños fatales de información, o la pérdida del sistema completo.

2.2.3.3.15. Utilizar procedimientos almacenados para evitar los ataques de SQL Injections.

2.2.3.4. Eficiencia

2.2.3.4.1. El tiempo de respuesta por transacción debe ser entre 2 segundos y 5 segundos.

2.2.3.4.2. Se desea un sistema eficiente con gran nivel de precisión, con tiempo de procesamiento de información y tiempo de respuestas rápidos, y que mantenga siempre la consistencia de los datos.

2.2.3.4.3. El sistema debe tener alta capacidad de concurrencia del servidor de datos y del servidor web donde se encuentre alojada la aplicación.

2.2.3.4.4. Utilización de recursos.

2.2.3.4.4.1. Para Desarrollo:

2.2.3.4.4.1.1. Intel Pentium 4 o superior

2.2.3.4.4.1.2. CPU 3GHZ o superior

2.2.3.4.4.1.3. 512 MB RAM o superior

2.2.3.4.4.1.4. 20 GB HDD o superior

2.2.3.4.4.1.5. Tarjeta de red con velocidad mínima de 100 Mbs

2.2.3.4.4.2. Para Explotación:

2.2.3.4.4.2.1. Clientes:

2.2.3.4.4.2.1.1. Pentium 3 o superior

2.2.3.4.4.2.1.2. CPU 133 MHZ o superior

2.2.3.4.4.2.1.3. 128 RAM mínimo 512 RAM recomendada o superior.

2.2.3.4.4.2.1.4. Tarjeta de red con velocidad mínima de 100 Mbs

2.2.3.4.4.2.2. Servidor Web y de Base de Datos:

2.2.3.4.4.2.2.1. CPU: Intel Core 2 Duo 2.0 GHZ o superior

2.2.3.4.4.2.2.2. RAM: 6 GB

2.2.3.4.4.2.2.3. 250 GB HDD

2.2.3.4.5. La aplicación debe estar concebida para el consumo mínimo de recursos.

2.2.3.4.6. En caso de degradación debe mostrar al menos la información a los usuarios.

2.2.3.5. Soporte

2.2.3.5.1. Normas de codificación

2.2.3.5.1.1. La indentación debe ser a cuatro espacios sin caracteres de tabulación

2.2.3.5.1.2. Las estructuras de control deben tener un espacio entre la palabra clave de la estructura y el signo de apertura de paréntesis.

2.2.3.5.1.3. Las funciones deben ser llamadas sin espacios entre el nombre de la función, el signo de paréntesis y el primer parámetro;

2.2.3.5.1.4. Todas las funciones deben tener un comentario, antes de su declaración, explicando que hacen.

2.2.3.5.1.5. Cuando se incluya un archivo de dependencia incondicionalmente utilizar require_once y cuando sea condicionalmente, utilizar include_once.

2.2.3.5.1.6. Utilizar las etiquetas <?php ?> para abrir un bloque de código.

2.2.3.5.1.7. Los nombres de las clases deben de iniciar con letra mayúscula.

2.2.3.5.1.8. Los nombres de las variables y de las funciones pueden iniciar con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula (el nombre puede escribirse separado por signos de guión mayor).

2.2.3.5.1.9. Si una función, en una clase, es privada; deberá comenzar con el signo de guión mayor para una fácil identificación.

2.2.3.5.1.10. Las constantes deben de escribirse siempre en mayúsculas y tanto estas como las variables globales deben de tener como prefijo el nombre de la clase a la que pertenecen.

2.2.3.5.1.11. Los archivos con código PHP, deben de ser guardados en formato ASCII utilizando la codificación ISO-8859-1. (Actualizado).

2.2.3.5.2. Realizar distintas pruebas al software una vez concluido para comprobar su funcionalidad.

2.2.3.5.3. Prestar los servicios de instalación y configuración de la aplicación.

2.2.3.5.4. Prestar servicios de mantenimiento del software.

2.2.3.5.5. Para el servidor de aplicaciones instalar intérprete de ficheros PHP y con las últimas actualizaciones del lenguaje.

2.2.3.5.6. Para el servidor de base de datos instalar un gestor de base de datos que soporte grandes volúmenes de datos y velocidad de procesamiento.

2.2.3.6. Restricciones de diseño

2.2.3.6.1. Utilizar como lenguaje de programación PHP 5.

2.2.3.6.2. Utilizar como Framework de desarrollo Symfony.

2.2.3.6.3. Para el diseño de la Base de Datos utilizar la herramienta de modelado DB Designer.

2.2.3.6.4. Utilizar la arquitectura Modelo Vista Controlador (MVC).

2.2.3.6.5. Utilizar el Visual Paradigm integrado con el IDE de desarrollo Eclipse.

2.2.3.6.6. Requisitos para la documentación de usuarios en línea y ayuda del sistema.

2.2.3.6.6.1. Conformar ayuda en línea para los usuarios finales donde tendrán permiso a ver la ayuda de acuerdo a las operaciones que pueden hacer en el sistema.

2.2.3.7. Interfaz

2.2.3.7.1. Las páginas no deben exceder los 100 Kb en las imágenes.

2.2.3.7.2. La interfaz debe mantener el mismo formato en todas las páginas.

2.2.3.7.3. Utilizar javascript y CSS 2.0 o superior como hoja de estilo en cascada.

2.2.3.7.4. El sistema debe combinar correctamente los colores, tipo de letra y tamaño.

2.2.3.7.5. Los iconos deben estar en correspondencia con lo que representan.

2.2.3.7.6. Las páginas de la aplicación no se cargarán con mucha información y contendrán solo las imágenes necesarias.

2.2.3.7.7. Se requiere esté instalado un navegador que interprete javascript.

2.2.3.7.8. Utilizar para la publicación de la aplicación el puerto 8080.

2.2.3.7.9. Utilizar como protocolo de comunicación http y https, este último será utilizado para las transacciones que requieran mayor seguridad

2.2.3.7.10. Interfaces Hardware

2.2.3.7.10.1. Debe soportar interfaces para impresoras.

2.2.3.7.11. Interfaces de Comunicación

2.2.3.7.11.1. Debe brindar interfaces de comunicación para los sistemas que lo requieran, garantizando que los cambios que se hagan en el sistema se actualicen en los sistemas que lo necesiten.

2.2.3.7.12. Software

2.2.3.7.12.1. Cliente:

2.2.3.7.12.1.1. Sistema Operativo con interfaz gráfica y conexión a red.

2.2.3.7.12.1.2. Navegador Web (Mozilla FireFox).

2.2.3.7.12.2. Servidor:

2.2.3.7.12.2.1. Ubuntu Server 7.10

2.2.3.7.12.2.2. Symfony Framework 1.0.11

2.2.3.7.12.2.3. PostgreSQL 8.1

2.2.4. Diagrama de casos de uso

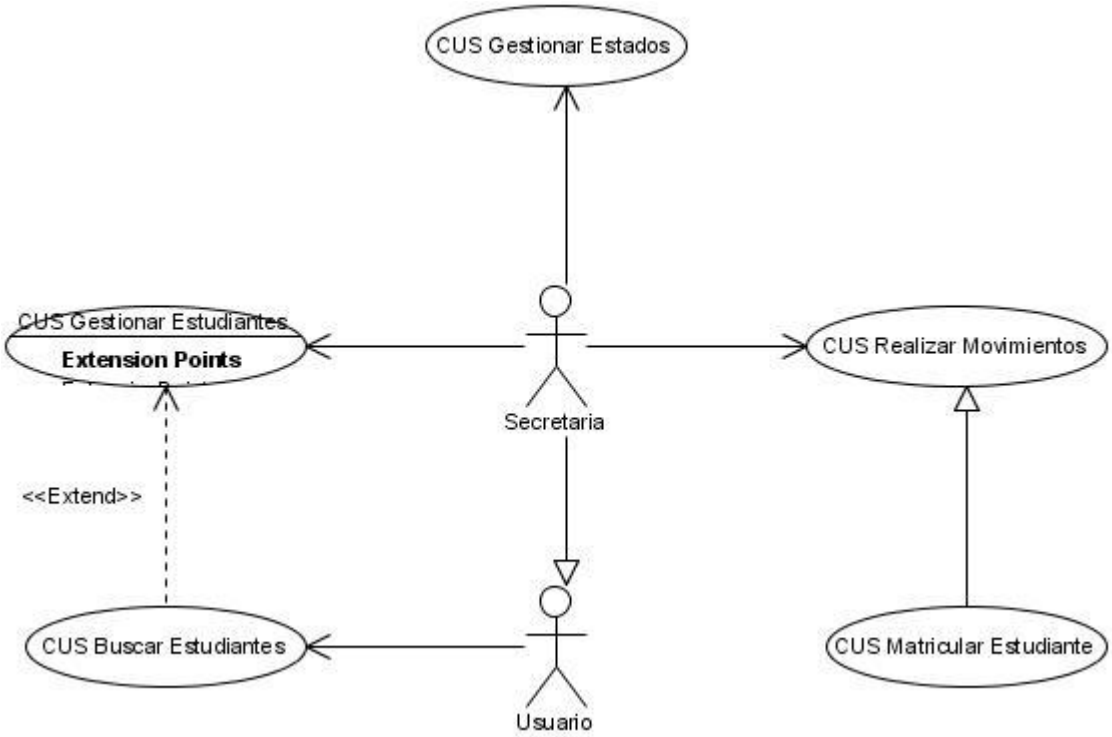


Figura 2-2 Diagrama de casos de uso del Sistema

2.2.5. Actores del sistema

Actor	Descripción
Secretaria	Es la encargada de la gestión de los datos de un estudiante. Así como de aplicarle los diferentes movimientos que se soliciten y sean aprobados. Será la encargada de definir los diferentes estados por los que pasa un estudiante y los movimientos que se pueden aplicar a un estudiante.
Usuario	Es el rol que pude acceder al buscador de estudiantes definido en el módulo de Matrícula.
Consultor	El consultor es el rol que puede acceder a los reportes definidos en el módulo matrícula.

Tabla 2-11 Actores del sistema

2.2.6. Descripción de los casos de uso del sistema.

2.2.6.1. Descripción del CUS Gestionar Estudiantes

Caso de Uso:	CUS Gestionar Estudiantes
Actores:	Secretaria
Resumen:	Este caso de uso le permite a la secretaria adicionar y/o modificar los datos de un estudiante en el sistema.
Precondiciones:	Cuando se desea modificar los datos de un estudiante el mismo debe existir en la base de datos. El usuario debe tener los permisos para poder realizar la operación.
Referencias	R1
Prioridad	Crítico
Pos condiciones	Cuando se crea un estudiante los datos del mismo quedan registrados en la base de datos, y se le pone estado de prematriculado. Cuando se modifica un estudiante los datos del mismo quedan modificados en la base de datos.

Tabla 2-12 Descripción resumida del CUS Gestionar Estudiante

2.2.6.2. Descripción del CUS Buscar Estudiantes

Caso de Uso:	CUS Buscar Estudiante
Actores:	Secretaria
Resumen:	Este caso de uso le permite a la secretaria buscar un estudiante
Precondiciones:	El usuario debe tener los permisos requeridos.
Referencias	R2
Prioridad	Crítico
Pos condiciones	

Tabla 2-13 Descripción resumida CUS Buscar Estudiantes

2.2.6.3. Descripción del CUS Gestionar Estados

Caso de Uso:	CUS Gestionar Estados
Actores:	Administrador
Resumen:	El caso de uso permite crear, modificar o eliminar un estado.
Precondiciones:	Cuando se desea modificar o eliminar un estado el mismo debe existir. El usuario debe tener los permisos para poder realizar la operación.
Referencias	RF3
Prioridad	Crítico
Pos condiciones	Cuando se crea un estado los datos del mismo quedan registrados en la base de datos. Cuando se modifica un estado los datos del mismo quedan modificados en la base de datos. Cuando se elimina un estado, los datos del mismo son borrados de la base de datos.

Tabla 2-14 Descripción resumida del CUS Gestionar Estados

2.2.6.4. Descripción del CUS Realizar Movimiento

Caso de Uso:	CUS Realizar Movimientos
Actores:	Secretaria
Resumen:	Este caso de uso permite a la secretaria aplicar cualquier movimiento que este definido en el sistema a un estudiante. También permitirá cambiar de estructura a un estudiante.
Precondiciones:	Cuando se desea aplicar un movimiento a un estudiante el mismo debe existir en la base de datos. El usuario debe tener los permisos para poder realizar la operación.
Referencias	RF4, Matricular Estudiantes.
Prioridad	Crítico
Pos condiciones	Se modifica el estado del estudiante poniendo como estado el estado final del movimiento.

Tabla 2-15 Descripción resumida del CUS Realizar Movimiento

2.2.6.5. Descripción del CUS Matricular Estudiante.

Caso de Uso:	CUS Matricular Estudiantes
Actores:	Secretaria
Resumen:	El caso de uso permite matricular un estudiante.
Precondiciones:	El usuario debe tener los permisos requeridos. El estudiante debe estar prematriculado
Referencias	RF7
Prioridad	crítico
Pos condiciones	El estudiante queda con estado matriculado.

Tabla 2-16 Descripción resumida del CUS Matricular Estudiante

2.3. Planificación y estimación de proyecto

El proceso de gestión de proyecto de software comienza con un conjunto de actividades que, globalmente, se denominan planificación del proyecto. La primera de estas actividades es la estimación, esta es una actividad importante que debe llevarse a cabo de forma cuidadosa. Existen técnicas útiles para la estimación de costes y de tiempos.

El método que se va a usar para la estimación es el Análisis de puntos de casos de usos. Este método permite predecir el tamaño de un sistema a partir de las características de los requisitos expresados en casos de uso.

La estimación mediante el análisis de puntos de casos de uso es un método propuesto originalmente por Gustav Karner de Objectory AB, y posteriormente refinado por muchos otros autores.

Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.

2.3.1. Cálculo de puntos de casos de uso sin ajustar.

UUCP=UAW+UUCW

UUCP: Puntos de casos de uso sin ajustar.

UAW: Factor de peso de las actores sin ajustar.

UUCW: Factor de peso de los casos de uso sin ajustar.

TIPO	PESO	VALOR	PESO*VALOR
Simple	1	0	0
Medio	2	0	0
Complejo	3	2	6
		Resultado :6	

Tabla 2-17 Factor de peso de los actores sin ajustar

TIPO	PESO	VALOR	PESO*VALOR
Simple	5	4	20
Medio	10	3	30
Complejo	15	0	0
		Resultado :50	

Tabla 2-18 Factor de peso de los caso de uso sin ajustar.

UUCP = 56

2.3.2. Cálculo de puntos de casos de uso ajustados

UCP=UUCP*TCF*EF

UCP: Puntos de caso de uso ajustados

UUCP: Puntos de casos de uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor ambiente

Factor	Peso	Valor	Peso*Valor
T1	2	0	0
T2	1	4	4
T3	1	2	2
T4	1	3	3
T5	1	5	5
T6	0.5	4	2
T7	0.5	3	1.5
T8	2	4	8
T9	1	3	3
T10	1	5	5
T11	1	5	5
T12	1	0	0
T13	1	3	3
TCF= 0.6+0.01*Σ(Peso i*Valor asignado i) : 1.015			

Tabla 2-19 Factor de complejidad técnica

Factor	Peso	Valor	Peso*Valor
E1	1.5	0	0
E2	0.5	3	1.5
E3	1	4	4
E4	0.5	5	2.5
E5	1	4	4
E6	2	2	4
E7	-1	0	0
E8	-1	2	-2
EF=1.4-0.03*Σ(Peso i*Valor asignado i): 0.98			

Tabla 2-20 Factor ambiente

UCP=55.7032

2.3.3. Estimación del esfuerzo

Teniendo en cuenta el factor ambiente, el factor de conversión es 20 horas/hombre por puntos de casos de uso.

El esfuerzo viene dado por la siguiente ecuación.

E=UCP*CF

CF: Factor de conversión.

E=55.7032*20 horas/hombres

E= 1114.064 horas/hombres

Actividad	%Esfuerzo	Valor Esfuerzo
Análisis	10	278.516
Diseño	20	557.032
Implementación	40	1114.064
Prueba	15	417.774
Sobrecarga	15	417.774
Total	100%	2785.16

Tabla 2-21 Estimación del esfuerzo total del proyecto

Conclusiones del capítulo II

Para la descripción de los procesos que ocurren en las universidades, se realizaron estudios, en la Universidad de las Ciencias Informáticas, la Universidad de la Habana y el análisis crítico de la documentación de los sistemas de gestión académica en el ámbito internacional, fundamentados en el primer capítulo, permitió determinar las contradicciones existentes entre las funcionalidades actuales del subsistema matrícula respecto a las necesidades de los usuarios finales.

Para cambiar a un estudiante de estado, se deben llenar varios documentos que avalen este proceso, y el sistema sólo permite llenar un documento.

La funcionalidad que define los cambios de estados es redundante, ya que esta definición se puede hacer en el momento que se gestiona el estado, evitando que el usuario final realice operaciones innecesarias.

3. ANÁLISIS Y DISEÑO DEL SUBSISTEMA MATRÍCULA

El siguiente capítulo está dividido en dos partes: en la primera se realiza el análisis del subsistema, donde se representan las realizaciones de los casos de uso, teniendo en cuenta los requerimientos funcionales que se plantearon en el capítulo anterior.

En la segunda parte se ponen de manifiesto las realizaciones de los casos de uso, en el diseño, incorporando los requisitos no funcionales establecidos para el subsistema matrícula.

3.1. Análisis

El análisis es una parte del flujo de trabajo **análisis y diseño** que propone el proceso unificado de desarrollo de software (RUP), que permite hacer la transición de los requerimientos, al diseño final del sistema, esta etapa no es obligatoria hacerla, pero facilita rehacer el diseño si en un futuro se cambian las restricciones definidas para el software (patrón de arquitectura, framework, lenguaje).

3.1.1. Realización de los casos de uso

3.1.1.1. CUS Gestionar Estudiantes

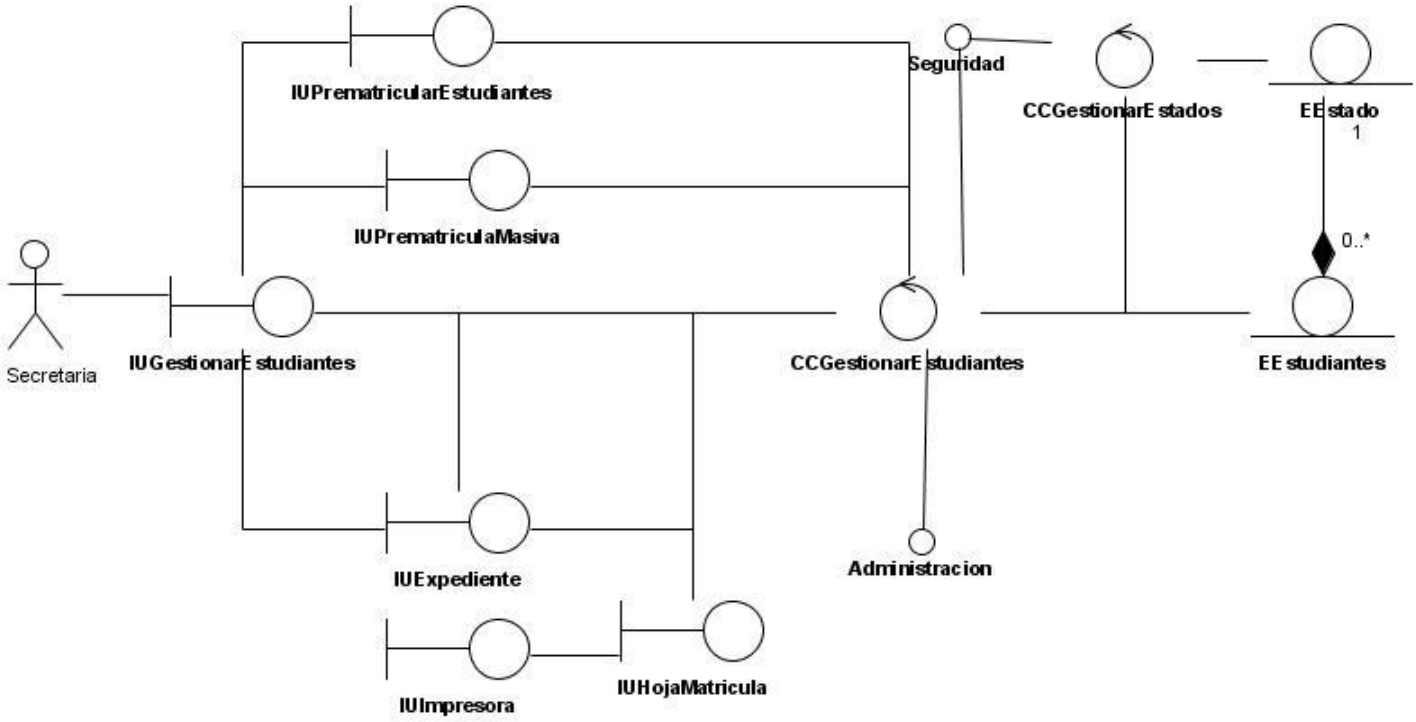


Figura 3-1 Diagrama de clases del análisis CUS Gestionar Estudiantes

3.1.1.2. CUS Buscar Estudiantes

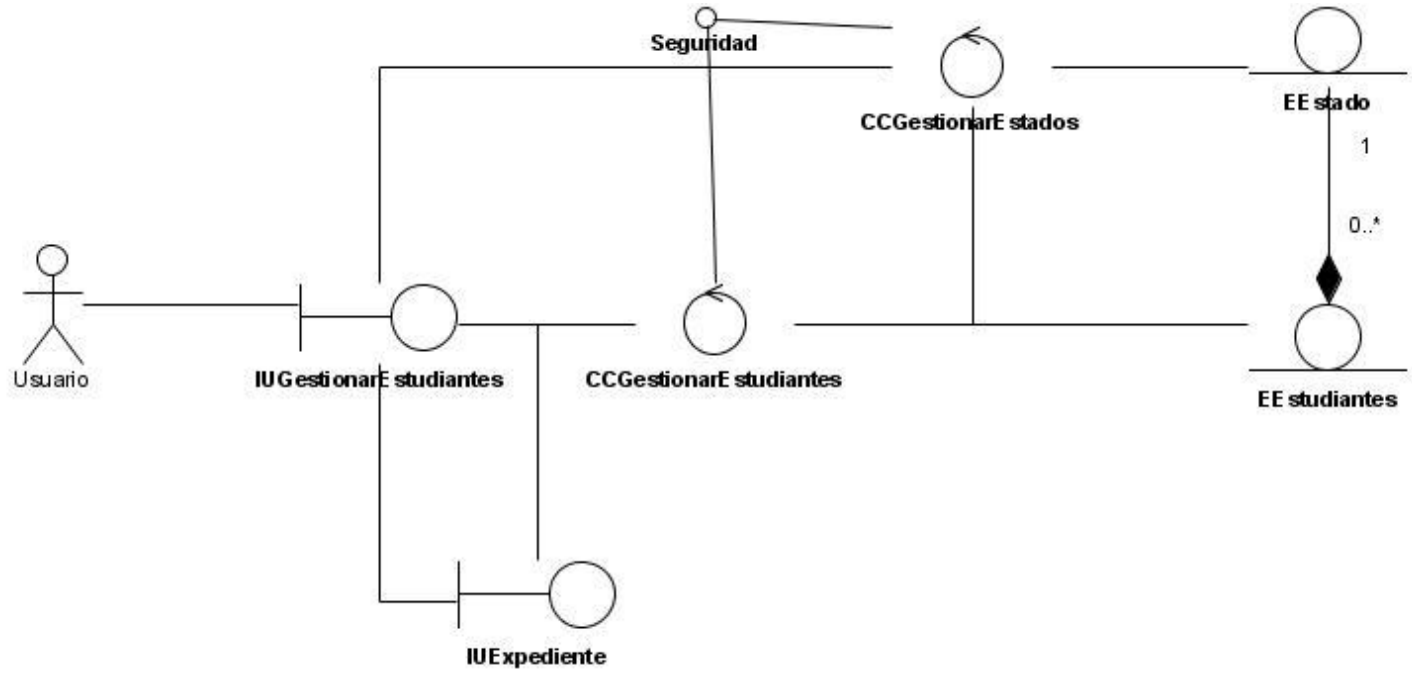


Figura 3-2 Diagrama de clases del análisis CUS Buscar Estudiante

3.1.1.3. CUS Gestionar Estados

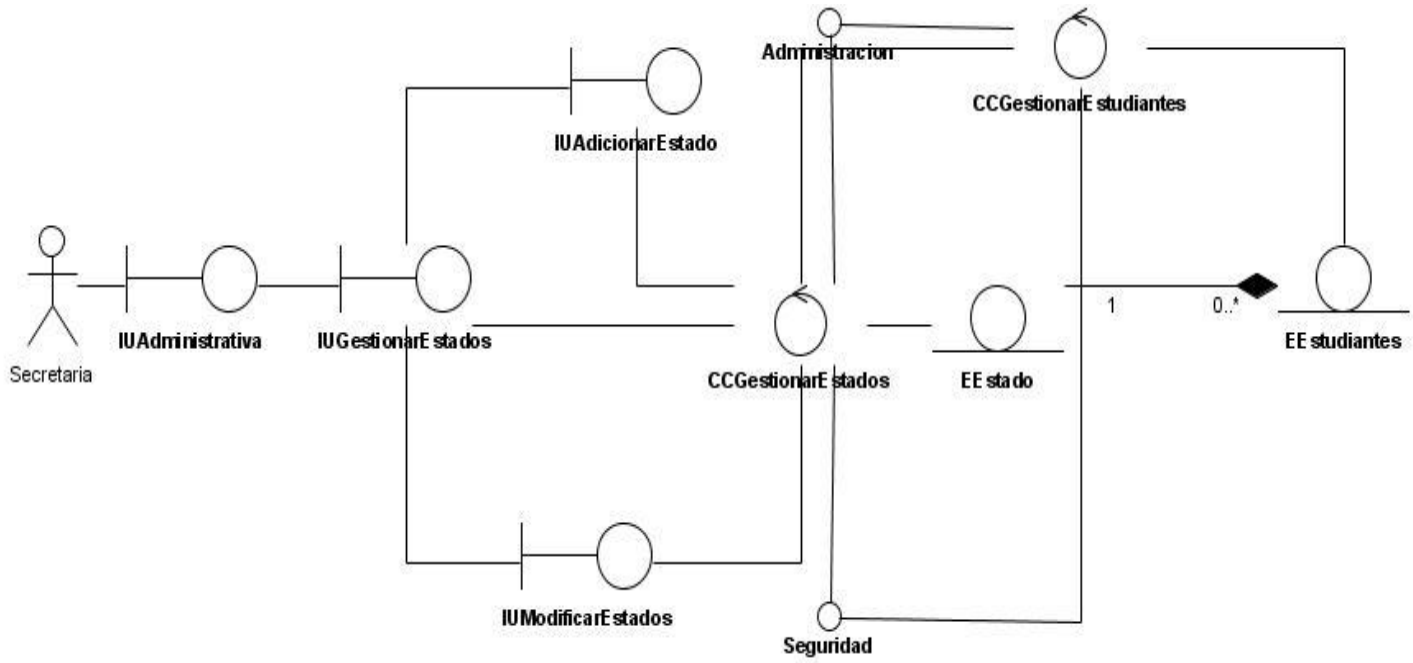


Figura 3-3 Diagrama de clases del análisis CUS Gestionar Estados

3.1.1.4. CUS Aplicar Movimiento

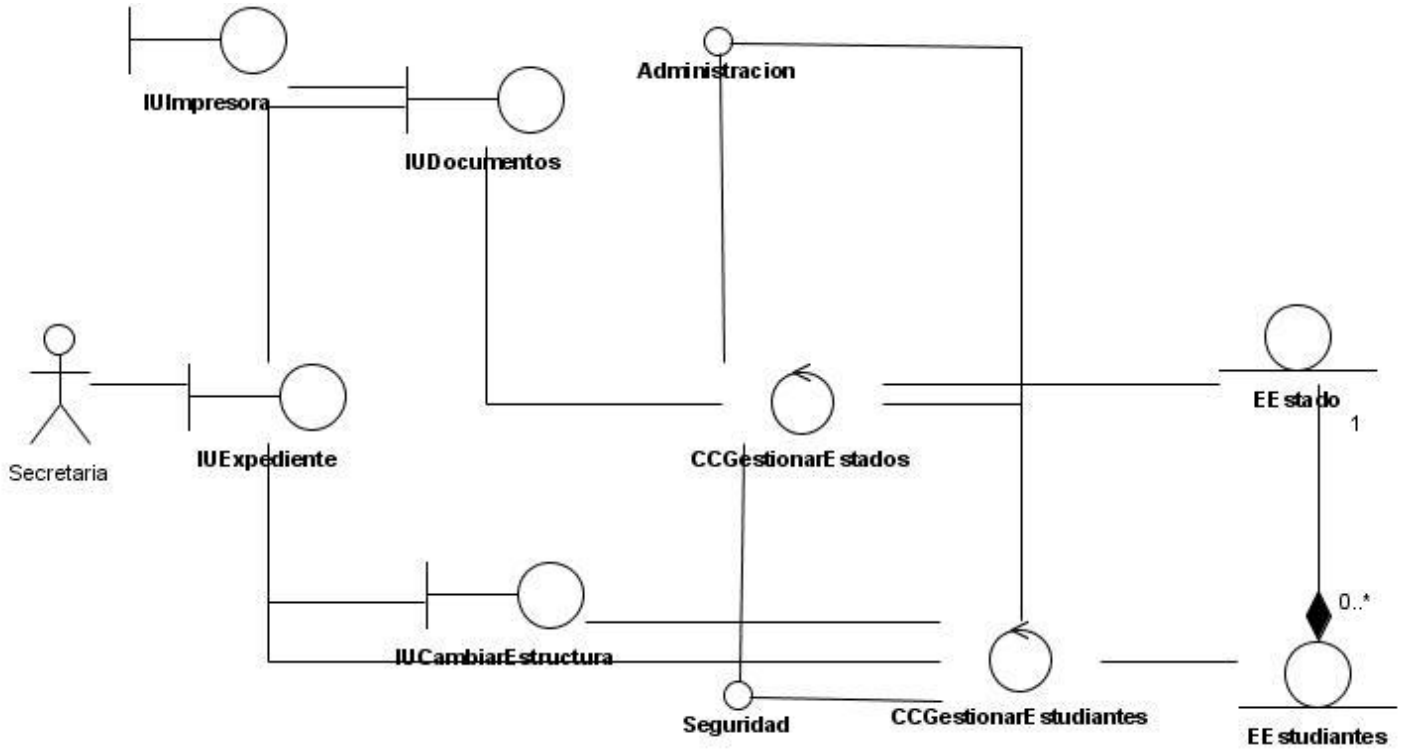


Figura 3-4 Diagrama de clases del análisis CUS Aplicar Movimiento

3.1.1.5. CUS Matricular Estudiantes

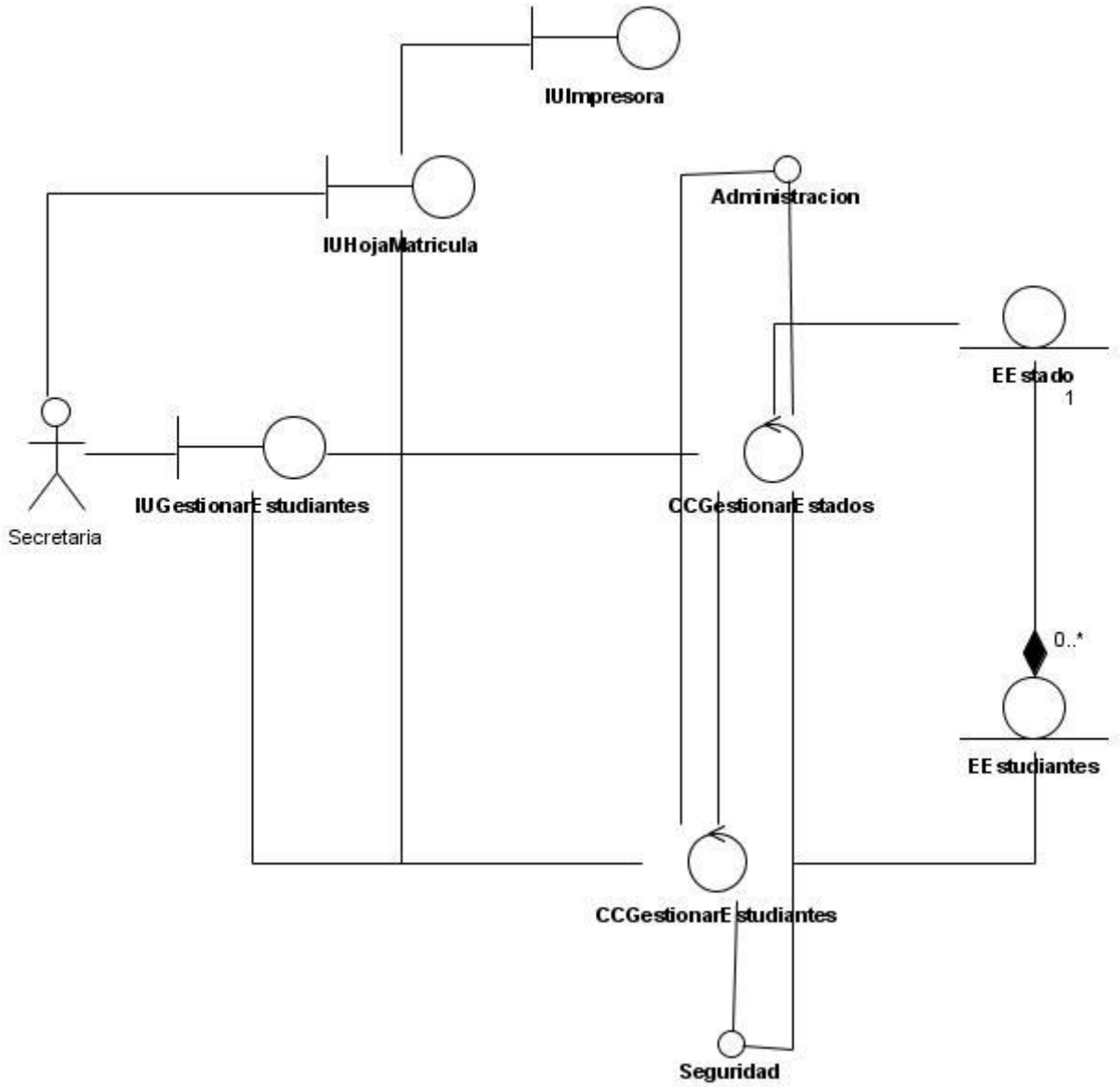


Figura 3-5 Diagrama de clases del análisis CUS Matricular Estudiante

3.2. Diseño

Para llevar a cabo el diseño del subsistema matrícula, se tuvieron en cuenta las restricciones del diseño planteadas en el capítulo anterior. El diseño se llevara a cabo teniendo en cuenta el patrón de arquitectura Modelo Vista Controlador (MVC), se hará uso del framework Symfony, que es un factor muy importante porque facilita poder hacer el diseño Orientado a Objetos, ya que tiene definidos

métodos que permite llevarlo a cabo de esta manera. Se tendrá en cuenta también el lenguaje de programación PHP5.

3.2.1. Patrones de diseño

Un patrón es una descripción de un problema y su solución, que recibe un nombre y puede emplearse en otros contextos, o sea, indica la manera de usarlo en circunstancias diversas.

3.2.1.1. Patrones de diseño de principios generales para asignar responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Existen varios patrones de asignación de responsabilidades, pero sólo se fundamentaran aquellos utilizados en el diseño del subsistema.

El patrón Experto es un patrón de asignación de responsabilidades, que permite darle solución al problema de cuál sería el principio fundamental para asignar responsabilidades en el diseño orientado a objetos. La solución factible a este problema es asignar la responsabilidad a la clase contenedora de la información necesaria para cumplir la responsabilidad. Mediante los diagramas de secuencia que se han realizado, se ha tratado de asignar las responsabilidades a las clases expertas en la información, para de esta manera garantizar que exista la menor dependencia posible entre las clases, donde aquí se pone de manifiesto el patrón de bajo acoplamiento el cual garantiza que exista una alta reutilización entre las funcionalidades de las clases y una escasa dependencia, facilitando el manteniendo de las clases. También haciendo uso del patrón experto se garantiza el cumplimiento de alta cohesión, el cual es un patrón que garantiza mantener la complejidad del diseño dentro de límites manejables.

La arquitectura usada para el diseño del subsistema es Modelo Vista Controlador, poniéndose de manifiesto el uso del patrón Controlador, ya que en el diseño va a existir una clase encargada de recibir todas las peticiones del usuario, mediante acciones con el mouse o el teclado. Esta clase va a ser la encargada de decodificar las peticiones del usuario y enviarlas hacia las clases que pueden darle solución a la petición.

3.2.1.2. Patrones de diseño Gang of Four (GOF)

Los patrones de diseño que pertenecen a la Pandilla de los 4, se clasifican en creacionales, estructurales y de comportamiento. El framework Symfony implementa varios patrones que están dentro de la clasificación GOF, en el caso de los creacionales se encuentra el patrón singular (singleton), el cual garantiza que se cree una sola instancia de una clase y crea un mecanismo de acceso global para esta. También hace uso del patrón Factoría Abstracta (Abstract Factory), ya que Symfony trabaja con clases de distintas familias, pero estas no se mezclan entre sí. De los patrones estructurales Symfony hace uso del patrón Decorador (Decorator), ya que la página que se le va a mostrar al usuario está compuesta por un layout y una plantilla, donde el primero decora a la segunda, permitiendo construir de esta manera diferentes vistas de acuerdo a los eventos del usuario. Debido al patrón de arquitectura que se va a usar uno de los patrones de diseño que están presentes es el patrón Fachada (Facade), en esta arquitectura, la clase o clases que realizan la función de controladoras reciben todas las peticiones de los usuarios, las decodifican y la envían hacia la acción que se debe ejecutar para darle respuesta, en eso cosiste el patrón Fachada, en proveer una interfaz unificada de manera que pueda interactuar con los demás componentes de la aplicación.

3.2.2. Realización de los casos de uso

3.2.2.1. CUS Gestionar Estudiantes

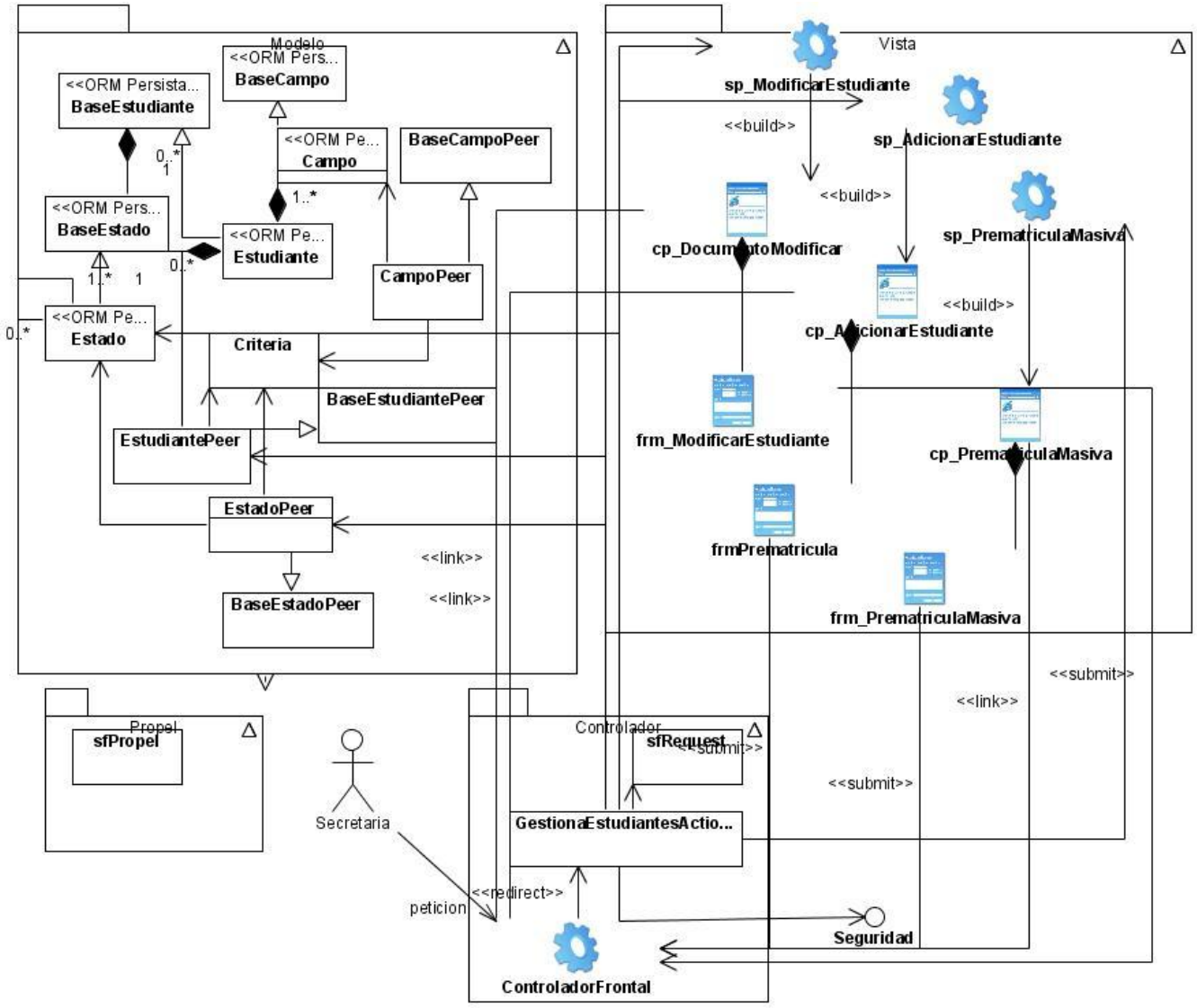


Figura 3-6 Diagrama de clases del diseño CUS Gestionar Estudiantes

3.2.2.2. CUS Buscar Estudiante

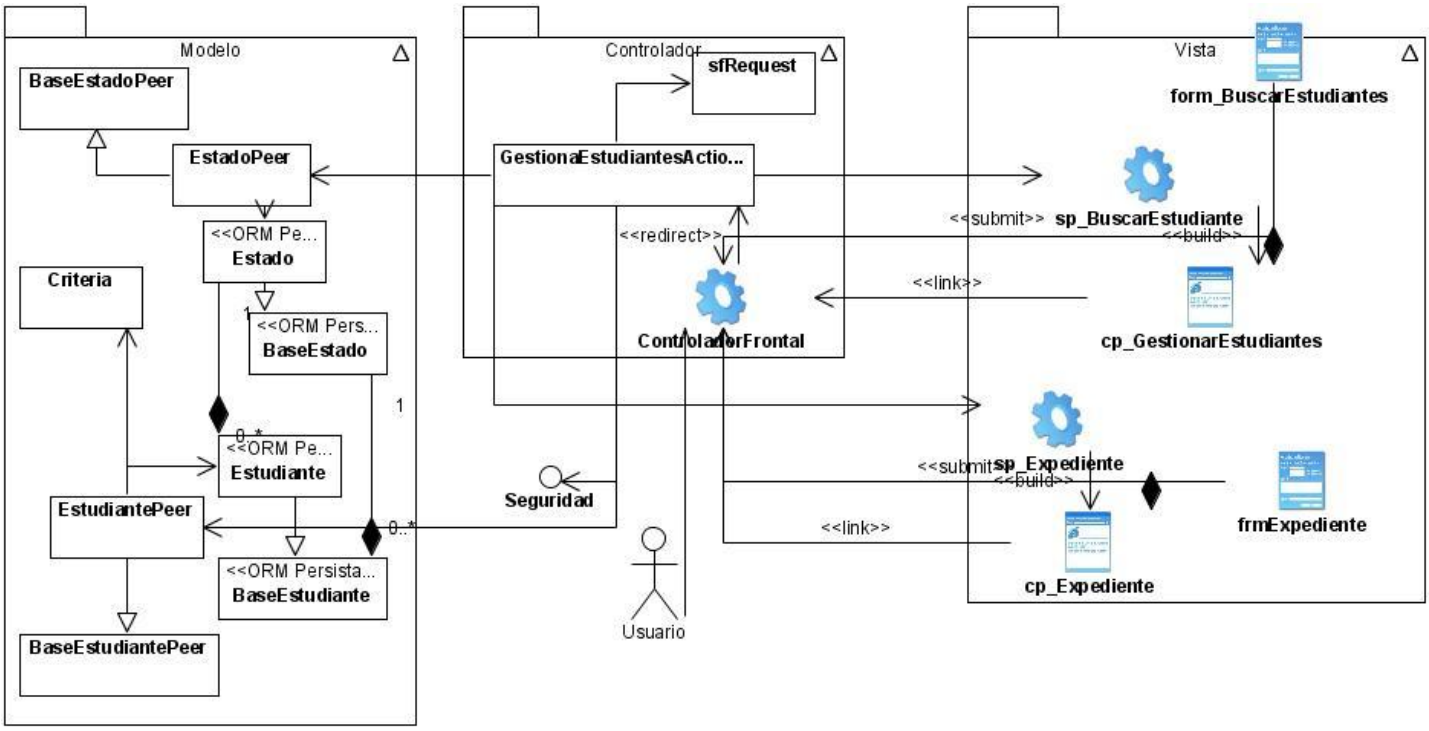


Figura 3-7 Diagrama de clases del diseño CUS Buscar Estudiante

3.2.2.3. CUS Gestionar Estados

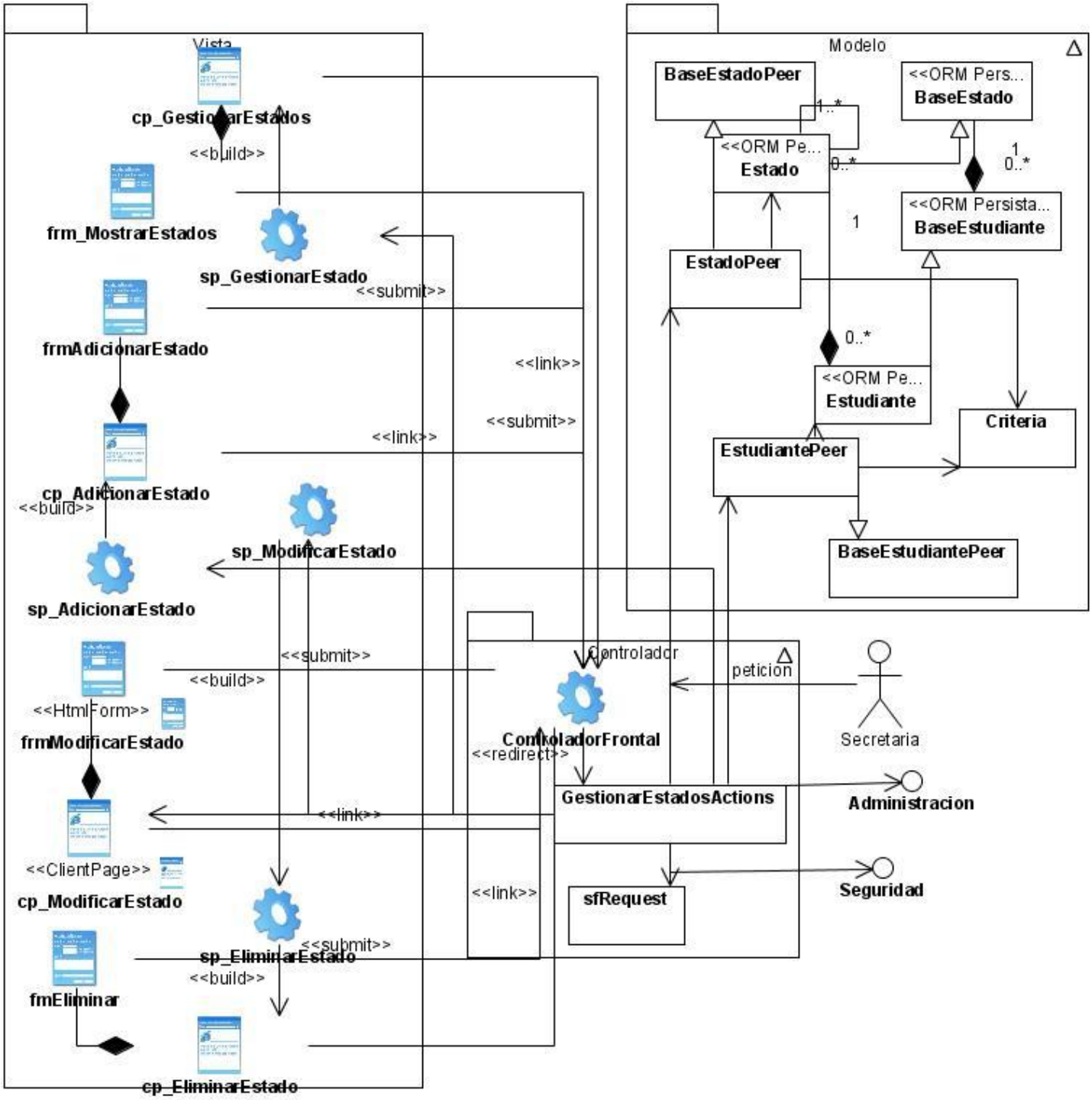


Figura 3-8 Diagrama de clases del diseño CUS Gestionar Estados

3.2.2.4. CUS Aplicar Movimiento

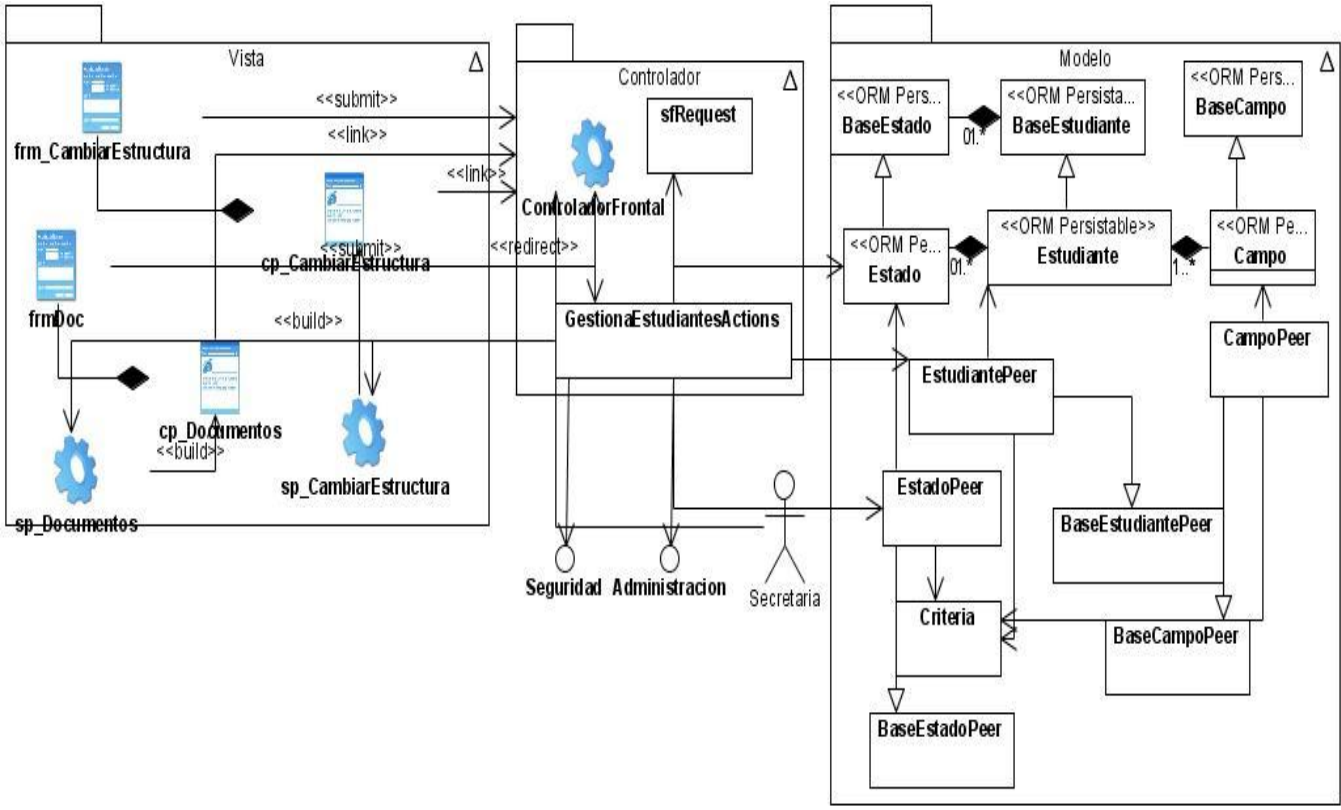


Figura 3-9 Diagrama de clases del diseño CUS Aplicar Movimiento

3.2.2.5. CUS Matricular Estudiante

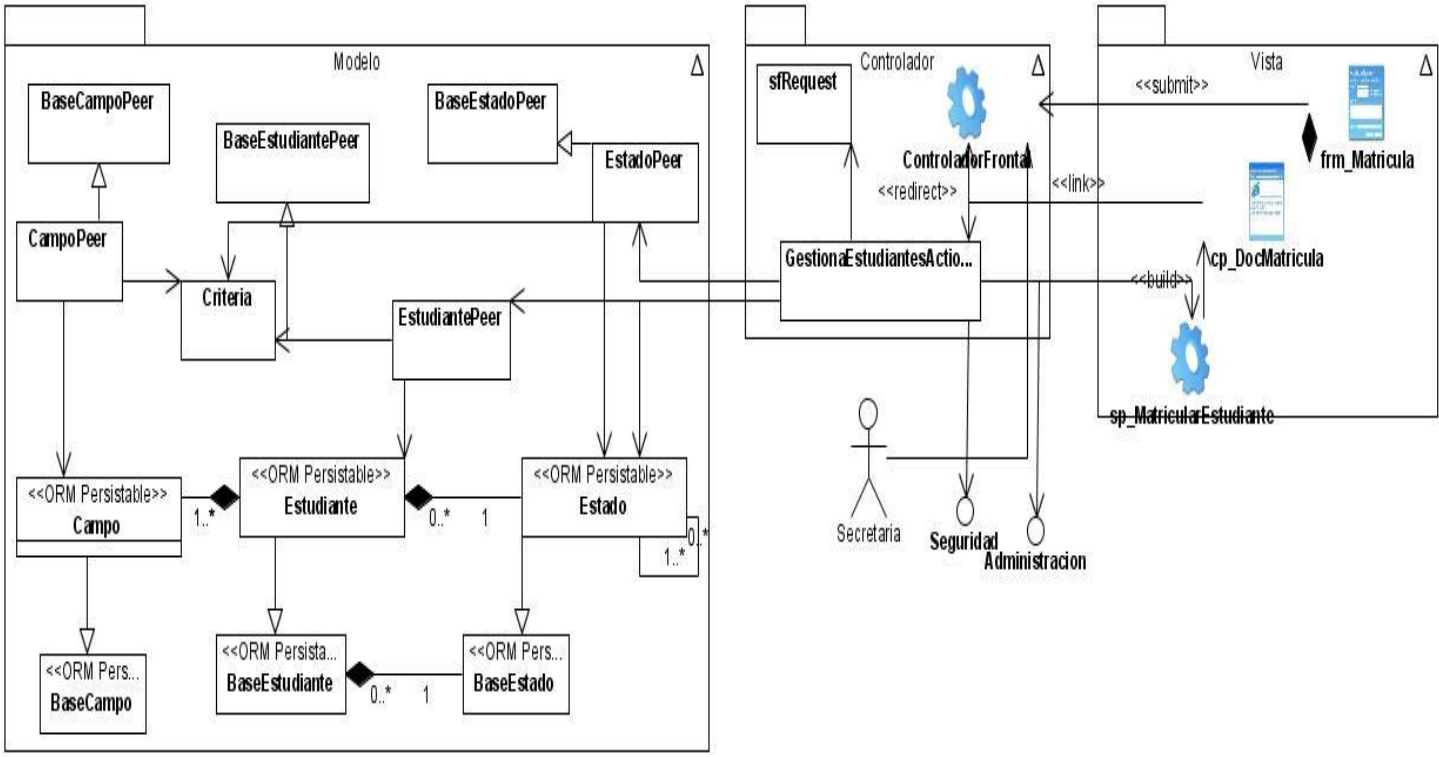


Figura 3-10 Diagrama de clases del diseño CUS Matricular Estudiante

3.2.3. Diagrama de clases persistentes

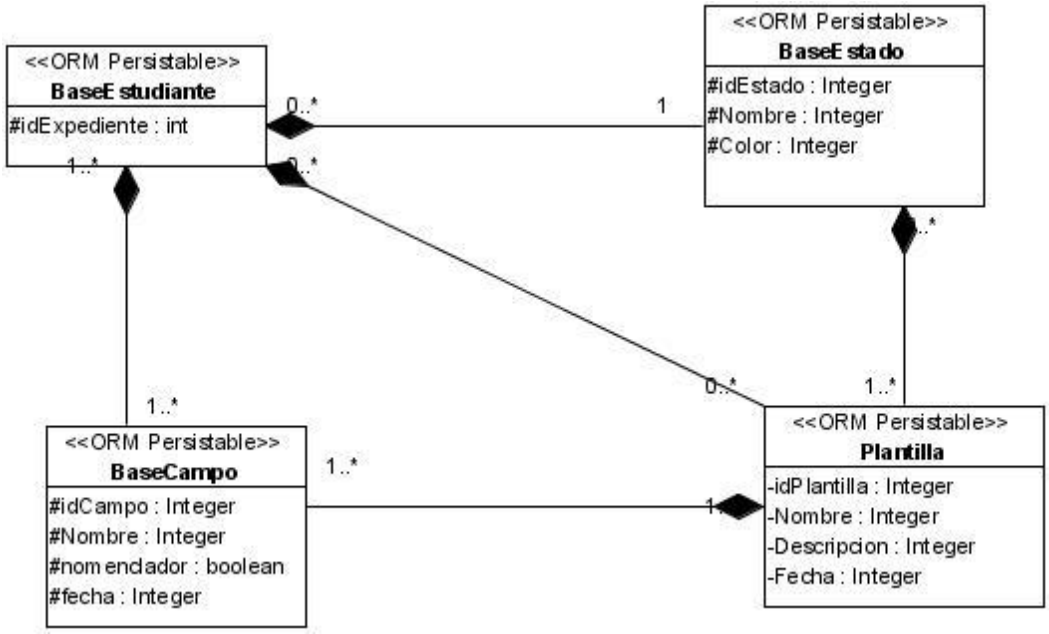


Figura 3-11 Diagrama de clases persistentes

3.2.4. Modelo de datos

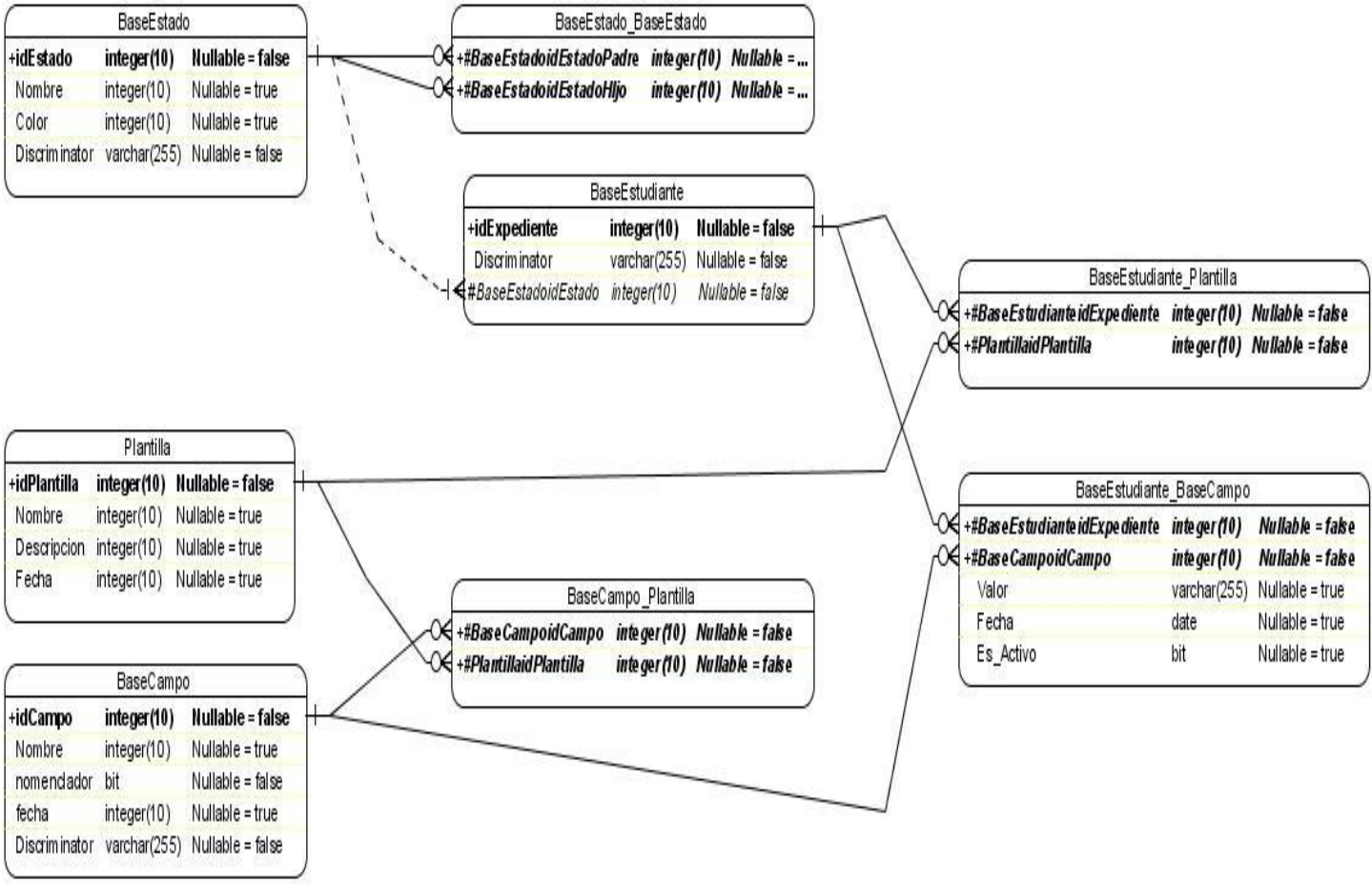


Figura 3-12 Modelo de datos

3.2.5. Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los diferentes nodos en tiempo de ejecución. Este diagrama de despliegue muestra la distribución del software una vez terminado. Como se puede observar, la aplicación va a estar distribuida en dos nodos diferentes, va existir un servidor de bases de datos y un servidor Web donde va a estar el subsistema. El nodo cliente representa las computadoras personales que se van a conectar al subsistema. Donde deben tener disponible una impresora para imprimir los documentos necesarios.

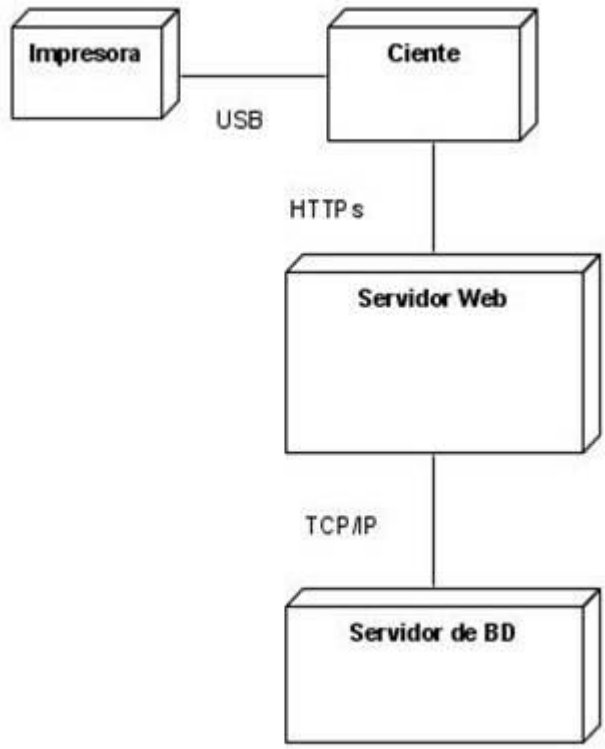


Figura 3-13 Diagrama de despliegue

Conclusiones del capítulo III

La propuesta de diseño por el cual se implementará el subsistema matrícula se realizó teniendo en cuenta las restricciones: lenguaje de programación PHP5, el framework Symfony y la arquitectura Modelo Vista Controlador (MVC). Con este diseño se podrá implementar un software de mayor calidad. Este informe servirá de documentación para los futuros desarrolladores que se integren al proyecto, y puedan conocer el funcionamiento del subsistema matrícula.

CONCLUSIONES

Akademos es un sistema de Gestión Académica, que mediante la propuesta de diseño que se hace aquí, se cumplirá con las expectativas de los usuarios finales. A través de esta investigación se han puesto de manifiesto diferentes temas que han ayudado al desarrollo de la propuesta final del diseño.

Los antecedentes del Subsistema han aportado mucho, en el ámbito internacional se tomaron en cuenta subsistemas que hasta el momento han cumplido con los requerimientos de las instituciones donde se utilizan, sirviendo de alguna manera como referencia para saber cuáles son los principales procesos en otras universidades del mundo. En Cuba, el desarrollo de sistemas de este tipo todavía es pobre, por esta razón se tomaron en cuenta ejemplos, que formaron parte de las razones por las que surge Akademos. De ellos se tiene en cuenta aquellos errores que no se deben cometer. La propuesta se basa en el análisis crítico de los errores cometidos anteriormente en el desarrollo de sistemas de este tipo, lo que garantiza la mejora del subsistema matrícula.

A partir de estas investigaciones se hace una propuesta de las funcionalidades para el subsistema matrícula, teniendo en cuenta también las opiniones de los usuarios finales que en este caso son las secretaras docentes y generales de las universidades. Se proponen un conjunto de requisitos no funcionales teniendo en cuenta estándares internacionales para aplicaciones Web, tanto en la codificación como en el diseño, para de esta manera hacer el trabajo de mantenimiento mucho más fácil, y la interfaz del usuario más amigable.

Se completa la propuesta de diseño, teniendo en cuenta algunas restricciones que contribuyen a mejorar el subsistema matrícula, ya que tienen en cuenta la política de migración a software libre establecida por Cuba.

RECOMENDACIONES

A lo largo de la investigación se realizan las siguientes recomendaciones:

- A los programadores del proyecto, implementar el subsistema teniendo en cuenta la propuesta de diseño hecha en la investigación.
- A los analistas y arquitecto del proyecto, que profundicen más su conocimiento de la herramienta Symfony, sobre todo en como mapea las relaciones mucho a mucho, ya que en el sistema existen muchas relaciones de este tipo, y los desarrolladores de más experiencia han tenido problemas en este sentido.
- Seguir incorporándoles nuevas mejoras al subsistema, y que se aprovechen las facilidades que ofrece Symfony para los formularios dinámicos que tanto se utilizan en el subsistema matrícula.

BIBLIOGRAFÍA

Ágora. Presentación del Sistema de Gestión de Alumnado. nº

GARCÍA, C. Curso de Hojas de Estilo. nº

HUMPHREY, W. S. A Selft-Improvement Process from Software Engineers 2005, nº

MUSCIANO, C. y KENNEDY, B. HTML La guía completa. 1999, nº

PÉREZ, J. E. Introducción a AJAX nº

---. Introducción a JavaScript nº

POTENCIER, F. y ZANINOTTO, F. Symfony la guía definitiva. 2008, nº

ROJAS, D. C.; DALMENDRAY, F. B.*, et al.* AKADEMOS, SISTEMA AUTOMATIZADO PARA LA GESTIÓN ACADÉMICA. nº

2008, nº [Consultado el: 04/04/2008]. Disponible en: www.phppatterns.com

RESOLUCION MINISTERIAL No. 86/98. nº

ASLESON, R. y SCHUTTA, N. T. Foundations of Ajax. Apress, 2006.

CUERVO, M. C. y MORENO, O. Y. B. Herramientas libres para modelar software Free tools to model software. Facultad de Ingeniería, 2005, vol. 14, nº p. 19.

DESCRIPCIÓN, M. y DE TAREAS, P. D. EMERGENCIA DE LOS PATRONES DE DISEÑO EN LAS ARQUITECTURAS DE SOFTWARE CON AYUDA DE RUP. nº

nº [Consultado el: 05/01/2008]. Disponible en: http://www.semicrol.es/gestion_universitaria2/gauss/pys_ciclo.html

JOHANSEN, E. Patrones de diseño. Revista ABB, 2006, nº p. 62-65.

RAMIREZ, A. Introducción a los Patrones de Diseño. Creative Commons. Agosto, 2004, nº

SIGENU, G. D. D. D. Manual de Usuario del SIGENU Sistema de Gestión de la Nueva Universidad. nº

RUMBAUGH, J.; JACOBSON, I., et al. El lenguaje unificado de modelado. Manual de referencia [material legible a máquina]. 2000, nº

PRESSMAN, R. Ingeniería del software. Un enfoque práctico. McGraw-Hill, 2001, nº p. 84-481.

LEÓN, R. A. H. y GONZÁLEZ, S. C. EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA Ciudad de la Habana: Editorial Universitaria(EDUNIV), ISBN 959-16-0343-6.

JACOBSON, I.; BOOCH, G., et al. El Proceso Unificado de Desarrollo de Software. Addison Wesley. Madrid, 2000, nº

GLOSARIO

Akademos: Sistemas de gestión académica desarrollado en la universidad de las Ciencias Informáticas.

Subsistema:

Estado: Situación en que se encuentra un estudiante en el centro de estudio.

Movimiento: Proceso para cambiar un estudiante de un estado a otro.

JavaScript: Lenguaje interpretado utilizado para en las páginas Web.

CUS: Casos de uso del sistema.

MySQL: Sistema de gestión de base de datos relacional.

PostgreSQL: Servidor de base de datos relacional orientada a objetos de software libre.

Oracle: Sistema de gestión de base de datos relacional fabricado por Oracle Corporation.

Open Database Connectivity (ODBC): Estándar de acceso a base de datos desarrollado por Microsoft Corporation.

HDD: Disco duro

RAM: Random access memory o memoria de acceso aleatorio.

CPU: Unidad central de procesamiento.