Universidad de las Ciencias Informáticas
"Facultad 2"

Título:

“Sistema automatizado para la gestión de información de los cursos optativos.”

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

Autores: Doralis De La Cruz Comptis
Giorbis Miguel Lorié Montalvo

Tutor: David Porto Castellanos

“Año 49 de la Revolución”

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado “Sistema automatizado para la gestión de información de los cursos optativos”, fue realizado en la Universidad de las Ciencias Informáticas. Se considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

- ☐ Totalmente
- ☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

_____
_____
_____
_____
_____
_____

Como resultado de la implantación de este trabajo se reporta un efecto económico que asciende a

_____

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____

_____	_____
Nombre del representante de la entidad	Cargo
_____	_____
Firma	Cuño

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: “Sistema automatizado para la gestión de información de los cursos optativos

Autores: Doralis de la Cruz Comptis.

Giorbis Miguel Lorié Montalvo

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

_____
_____
_____
_____
_____
_____

Por todo lo anteriormente expresado considero que los estudiantes están aptos para ejercer como Ingenieros en Ciencias Informáticas; y propongo que se le otorgue al Trabajo de Diploma la calificación de _____.

_____ _____

David Porto Castellanos Fecha

DECLARACIÓN DE AUTORÍA

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

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

Autores

Doralis De La Cruz Comptis

_____

Giorbis Miguel Lorié Montalvo

_____

Tutor

David Porto Castellanos

_____

PENSAMIENTOS

La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica.
Aristóteles

Se alcanza el éxito convirtiendo cada paso en una meta y cada meta en un paso.

C.C. Cortéz

AGRADECIMIENTOS.

De Doralis

Primeramente quiero agradecer a Dios por haberme hecho todo cuanto soy. A mi madre por haberme dado todo lo que tengo hasta hoy. También debo agradecer a mi familia, incluyendo a mi papa que en el lugar que se encuentre sabe que este triunfo es suyo, gracias papá. A mi pareja (Alex) por la comprensión y apoyo incondicional que me ha brindado. Y a mis amistades por tenderme la mano cuando lo he necesitado.

A mi tutor David por habernos dedicado parte de su tiempo.

Gracias.

De Giorbis

Agradezco a todos los que nos han acompañado en este largo camino de la vida y la educación dándonos fuerzas para salir adelante y realizar nuestros sueños.

Gracias a todos

DEDICATORIA

De Doralis

A quienes han soñado con este momento mi mamá, familia y amistades.

Doralis de la Cruz Comptis.

De Giorbis

A quienes con su sufrimiento me enseñaron las lecciones de la vida, me enseñaron a caminar, a levantarme si me caigo, a aprender de mis errores, fuente de mi vida y de mis ganas de vivir.

A mi madre y mi padre.

Giorbis Miguel Lorié Montalvo

A todas las personas que creen que un mundo sin Windows es posible.

RESUMEN

La gestión de la información se ha convertido en un elemento importante para el desarrollo de la sociedad, capaz de determinar el buen funcionamiento de cualquier empresa o proceso. En la Universidad de las Ciencias Informáticas es de vital importancia para ganar en eficiencia pues es mucha la información que se genera para llevar el control de los estudiantes.

El Sistema Automatizado para la Gestión de Información de los Cursos Optativos (CO) que se desarrollará en este trabajo mejorará el control y manipulación de la información que se genera para el control de la participación de los estudiantes en estos cursos que se imparten en la facultad y que en cierto sentido son de obligatorio cumplimiento.

En la Facultad II todo el trabajo con los CO es muy tedioso pues se gestiona de forma manual gran parte de la información. También el Vice-decano de docencia envía por correo electrónico a los estudiantes y profesores casi toda la información sobre los CO. Si por alguna razón cambia el aula o laboratorio, la hora de comienzo o el profesor del CO, se incurre en pérdida de tiempo pues el estudiante debe esperar por el correo con esta información actualizada.

Este proyecto pretende entender toda la lógica de los procesos que están vigentes en el funcionamiento de gestión de los Cursos Optativos actualmente modelando los procesos que en esta actividad intervienen a través del Proceso Unificado de Rational y confeccionar una aplicación Web que permita gestionar los cursos agilizando el proceso de matrícula de los estudiantes y brinde otras ventajas tales como tablas resúmenes, información actualizada y al alcance de todos los usuarios a través de la Intranet de la institución.

Los resultados alcanzados en este trabajo son: 1) Mayor conocimiento sobre la gestión de la información en los CO. 2) Agilización y eficiencia en la confección de los documentos necesarios para la gestión de los CO. 3) Centralización de la información, con más eficacia y rapidez.

INDICE

INTRODUCCION

La mayoría de las entidades cubanas están haciendo uso de las nuevas tecnologías informáticas para la gestión de información pues son más que probadas las ventajas que estas brindan. En la Universidad de las Ciencias Informáticas (UCI), específicamente en la Facultad II, se precisa la gestión de la información de los CO. Un CO se imparte en un período de tiempo relativamente corto y tiene como objetivos: 1) Desarrollar en los estudiantes los hábitos de estudio individual que se requieren para poder enfrentarse a la solución de problemas en el mundo de la computación; 2) Contribuir a la formación de profesionales con una alta calificación en el uso de las nuevas tecnologías; 3) Fomentar en los estudiantes un estilo de trabajo en el que siempre este presente la atención a la calidad del Software y 4) Posibilitar que los estudiantes conozcan diversas técnicas, herramientas y tecnologías.

En la UCI cada estudiante debe cursar no menos de ocho CO antes de graduarse. Esto implica llevar un control de la información referente a los CO. Actualmente en la Facultad II todo este trabajo resulta bastante tedioso, pues se gestiona manualmente gran parte de la información.

Los CO pueden ser a nivel de Universidad o Facultad. Los de nivel de Universidad son planificados e impartidos por los departamentos generales de las asignaturas correspondientes. En el caso de los CO de la Facultad son impartidos por el Departamento de la Especialidad de cada facultad. Estos CO se imparten en horario extra-clase y por lo general ayudan al estudiante en su desempeño en las asignaturas que recibirá en años siguientes, ya que les aporta conocimientos relacionados con las mismas. Los CO tienen un perfil: *obligatorio* u *opcional*. Dentro del primero se agrupan aquellos cursos relacionados con el perfil de estudio de la facultad y el segundo no, lo que le da la oportunidad al estudiante de aprender sobre algún tema en específico que sea de su interés y que no esté contemplado dentro del plan de estudio de la facultad a la que pertenece.

Para conformar un CO el vice-decano de docencia hace un pedido al jefe del departamento de la especialidad sobre la disponibilidad de los CO. El jefe de departamento de la especialidad consulta con los profesores, para que ellos le informen si van impartir algún curso, y de ser así, les exige la confección del Programa Analítico (P1) correspondiente al curso.

El vice-decano planifica el horario que se le debe asignar a cada CO. Cabe destacar que el proceso de planificación del horario incluye no solo lo referente a la hora de inicio y fin, sino además lo relativo a las

aulas o laboratorios, es decir, hora y lugar. El local puede ser laboratorio, aula o salón de conferencia y la planificación del tiempo comprende fecha de inicio y fin de los turnos, que pueden ser en la sección de la mañana o la tarde.

Una vez concluida la planificación del horario de los CO, el vice-decano de Docencia envía a todos los estudiantes y profesores un documento Excel con toda la información. Los profesores consultan este documento con el fin de saber la planificación del horario y los estudiantes lo consultan para conocer la disponibilidad de cursos y decidir cual de ellos desea matricular. Para hacer efectiva la matrícula en un curso, el estudiante debe presentarse en el Departamento de la Especialidad para anotarse en dicho CO.

Cuando se comienzan a impartir los CO, los profesores tienen que recoger el listado de los estudiantes que matricularon. Por último, el profesor debe insertar en la plataforma “Akademos”, en un tiempo prudencial, las notas obtenidas por los estudiantes en el CO, y en caso de vencido este tiempo, deberá entregar ese listado a la secretaria de Docencia para que ella efectúe esa operación y publique los resultados en la mencionada plataforma.

Luego, la situación problemática queda planteada por los siguientes puntos:

- Ya que la cantidad de CO mínima a cursar por un estudiante es ocho, esto suele generar grandes volúmenes de información, determinado además por el número de estudiantes de cada facultad y por la obligatoriedad de los CO (es un requisito indispensable para graduarse haber cursado los ocho CO).

- Casi toda la información se almacena utilizando documentos Excel, lo que dificultad la gestión de la información referente a los CO.

- La información sobre los CO se envía a los estudiantes utilizando la Intranet de la UCI mediante el correo electrónico.

- La matricula se realiza en el Departamento de la Especialidad, provocando que muchos estudiantes se acumulen y se hagan colas en el Departamento de la Especialidad pues solo existe una persona encargada de llevar el control de la matrícula.

- También existen problemas con los requisitos para ingresar a los cursos, los cuales debe cumplir el estudiante que desea matricular. Pero esto no funciona de forma eficiente pues muchos estudiantes se matriculan en cursos cuando no cumplen con esas precondiciones.

- Por otra parte, los estudiantes no tienen acceso al P1 de los CO, lo cual les impide valorar si la temática que se impartirá es de su interés y si realmente desean recibirlo.

- Los listados de los estudiantes que matricularon en un curso por lo general son hojas que si se extravían el profesor pierde los estudiantes que matricularon en su CO.

- Si los profesores tienen que publicar alguna documentación referente al curso deben hacerlo en sus máquinas personales cuando toda esta información debería encontrarse en un servidor dedicado a este propósito.

- Los locales donde se van a impartir los CO pueden cambiar. En estos casos los estudiantes deben buscar en nuevo local por todo el edificio de la facultad incurriendo en tardanzas, incluso ausencias.

- En ocasiones la entrega de los listados por parte de los profesores no se efectúa en la fecha planificada, lo que produce atrasos en la publicación de las notas.

Consecuentemente con dicha *situación problemática* se identifican los siguientes *problemas*:

- ¿Cómo mejorar el proceso de matrícula de los estudiantes en los CO?

- ¿Cómo lograr una gestión de la información correspondiente a estos cursos de forma ágil, dinámica y eficiente?

El objeto de estudio de este trabajo lo constituyen el conjunto de fases que tienen lugar durante el proceso de matricula de los alumnos en un curso dado así como la gestión de la información resultante de este proceso.

El campo de acción se centra en el estudio de los procesos que intervienen en la gestión de información de los cursos optativos en la UCI, específicamente en la Facultad II.

A partir de estos puntos, se elabora la siguiente **hipótesis**:

"La solución para llevar a cabo una gestión dinámica, rápida y eficiente de los CO, tanto por parte de los profesores involucrados en esta actividad como por los estudiantes que los reciben, puede ser el diseño de una aplicación Web accesible por los usuarios a través de la Intranet de la Universidad."

Este proyecto tiene como objetivo general **desarrollar un sistema para la gestión de Cursos Optativos**. A partir de este objetivo se definieron los siguientes objetivos específicos:

- Hacer uso de herramientas no propietarias acorde a la política que están siguiendo muchas empresas de software del país, incluso la propia UCI.
- Analizar y Diseñar el sistema en cuestión.
- Obtener, al menos, una primera versión del software.
- Poner a disposición, tanto de profesores como alumnos, toda la información correspondiente a los CO de la Facultad II.

Para dar cumplimiento a los objetivos trazados, se llevaron a cabo diversas tareas tales como:

- Investigar acerca de las herramientas no propietarias que ayudan a desarrollar la aplicación.
- Indagar sobre el trabajo con la información de los cursos optativos.
- Profundizar en el estudio de herramientas de desarrollo en ambiente Web, principalmente aquellas de distribución gratuita con código libre.
- Estudiar las características del Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), la metodología del Proceso Unificado de Desarrollo de Software (RUP, por sus siglas en inglés), y de la Programación Orientada a Objetos (POO).

La introducción de esta herramienta tendrá varios aportes prácticos para la Facultad II, entre los que se destacan.

- Centralización de la información, con más eficiencia y rapidez.
- Calidad y confiabilidad de la información que se maneja sobre los CO y las notas de los estudiantes.

- Mayor conocimiento de la gestión de los CO en la Facultad II.
- Agilización y eficiencia en la confección de los documentos necesarios para la gestión de los CO.

Por otra parte, esta herramienta contribuirá a materializar el deseo de muchas empresas desarrolladoras de software del país, en su empeño por eliminar las herramientas propietarias y fomentar el uso de tecnologías de desarrollo de software libre.

Capítulo Primero: **Fundamentación Teórica.** Se exponen las bases que sustentan la investigación, las características de la metodología de desarrollo, las tendencias técnicas, software usado en la actualidad y las herramientas utilizadas para la solución del problema.

Capítulo Segundo: **Características del sistema.** Se puntualiza el flujo de los procesos involucrados en el problema, haciendo una valoración critica de cómo se desarrollan, y analizando la información que el sistema debe gestionar. También se destacan los aspectos más relevantes del negocio que se analiza a través de diagramas de casos de uso del negocio y la descripción de estos mediante diagramas de actividad que propone la metodología de desarrollo empleada, con el objetivo de entender el contexto del sistema a desarrollar.

Capítulo Tercero: **Análisis y diseño del sistema.** Se representa una vista interna del sistema en la que, usando el lenguaje propuesto se refinan los requisitos. Este proceso continúa en el diseño hasta obtener los objetos que interactúan para cumplir los requisitos funcionales y no funcionales obtenidos. Además, aparecen en este capítulo la representación grafica del diagrama entidad relación y las definiciones de diseño que se aplicaron.

Capítulo Cuarto: **Implementación.** Aborda los flujos de trabajo de implementación, exponiendo a través de un grupo de diagramas cómo ha sido llevada la solución desde el diseño hasta la implementación.

Capítulo Quinto y Final: **Estudio de Factibilidad.** Realiza la estimación del esfuerzo y tiempo de desarrollo para alcanzar un producto final. También se incluye el análisis de costo y beneficios involucrados en la realización del mismo.

CAPÍTULO I: FUNDAMENTACION TEORICA

1.1 Introducción

En este capitulo se hace referencia a los temas con los que se le da solución al sistema. El lector encontrará todo lo referente al estado del arte a nivel de la Universidad, nacional e internacional. Además, aparecen en este capítulo las tendencias técnicas, tecnologías, metodologías y software usados en la actualidad para dar solución al problema.

1.2 Estado del arte

En la actualidad existen antecedentes de este tipo de aplicación ya que numerosas entidades educacionales permiten la matrícula de estudiantes a través de aplicaciones Web, ejemplo de eso tenemos [1], [2]. En estos sitios Web los navegantes interesados reciben la información sobre los cursos. Entre ellos tenemos el día de comienzo, tiempo de duración, tema y otros detalles dependientemente del curso. Además, se les solicita información personal necesaria para matricularse.

Se deduce que detrás de todas estas aplicaciones funcionen módulos de administración a los cuales no se puede tener acceso. Estos módulos de administración deben permitir gestionar y administrar la información de los matriculados.

En nuestro país existen varias formas de realizar estas matriculas; algunas simplemente indican el lugar donde deben ir los interesados y un correo electrónico para escribirle si desea matricular, ejemplo [3].

En la Universidad, hasta el momento, no existe ninguna aplicación que se encargue de la gestión de los CO. Toda la información referente a los CO se encuentra almacenada en documentos Excel, salvo la plataforma Web “Akademos”, que almacena los cursos que los estudiantes han superado con las respectivas notas.

1.3 Descripción del tipo de aplicación:

Existen dos tipos de aplicaciones informáticas: aplicaciones de Escritorio (o Desktop como se les conoce en el lenguaje informático) y aplicaciones Web.

La aplicación Desktop no resulta la solución más eficiente para el problema que se presenta pues centraliza la gestión de la información a una o solo algunas estaciones de trabajo donde se hace

necesario instalar todas las herramientas que requiere para su funcionamiento, limitando el uso del software a aquellos usuarios que tengan acceso a dichas computadoras.

Por otra parte, una aplicación Web ofrece facilidades que pudieran ser la solución a los problemas planteados al inicio del trabajo. Primeramente, permite consultar la información contenida desde cualquier estación de trabajo conectado a la Intranet de la Institución. También, las actualizaciones o mejoras que se le realicen a la aplicación resultan transparentes al usuario final pues se realizan solamente en la estación que actúa como servidor de aplicaciones Web independiente completamente de la estación cliente. Para acceder a la aplicación solo se necesita de un navegador Web como cliente ligero y la conexión puede ser desde cualquier máquina que esté conectada a la red. El usuario sólo debe preocuparse por la seguridad de su navegador pues la seguridad de la aplicación estará en manos de un grupo de especialistas encargados de dar mantenimiento al software.

Las aplicaciones Web al alcanzado gran auge en el mercado mundo actual gracias a la evolución de las tecnologías de la información y las comunicaciones, y en especial, de Internet. Una aplicación Web no puede ser tomada como una simple página Web, sino como un programa complejo de naturaleza concurrente, multiusuario y capaz de ser utilizado en múltiples entornos. Tal es así que las viejas páginas Web con HTML y JavaScript han dado paso a aplicaciones Web con conexiones a bases de datos, servidores de aplicaciones, así como sistemas de comunicación instantánea.

1.4 Descripción de las metodologías

1.4.1 Lenguaje Unificado de Modelación.

"El Lenguaje Unificado de Modelación (UML, por sus siglas en ingles) constituye una notación para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientados a objetos. Un artefacto es una información que es utilizada o producida mediante un proceso de software y puede ser un modelo, una descripción o un software; los artefactos de UML se especifican en forma de diagramas, los que pueden ser definidos como una representación gráfica de una colección de elementos del modelo". [4].

UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Aún cuando todavía no es un estándar oficial, está apoyado en gran manera por el Grupo de Administración de

Objetos (OMG, por sus siglas en inglés Object Management Group). UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

El punto importante es que UML es un "lenguaje" para especificar y no un método o un proceso. UML se usa para definir un sistema de software; para detallar los artefactos en el sistema; para documentar y construir es el lenguaje en el que está descrito el modelo. UML se puede usar en una gran variedad de formas para soportar una metodología de desarrollo de software (tal como el Proceso Unificado de Rational) pero no especifica en sí mismo qué metodología o proceso usar.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

Los diagramas fundamentales que emplea UML son los siguientes:

• **Diagrama de casos de uso:** Especifica las funcionalidades y el comportamiento de un sistema, su interacción con los usuarios del mismo y otros sistemas.

• **Diagrama de Comportamiento o Interacción:** Muestra las interacciones entre objetos ocurridas en un escenario o parte del sistema.

• **Diagrama de Clases:** Representa un conjunto de elementos del modelo que son estáticos. Estos aspectos estáticos modelan características del software como pueden ser su estructura interna y la representación que se hará de la información en la aplicación.

• **Diagrama de Implementación***:* Muestra los aspectos físicos del sistema e incluye la estructura del código fuente y la implementación.

1.4.2 Proceso Unificado del Racional.

Para un desarrollo eficiente de la aplicación se seleccionó la metodología del Proceso Unificado de Racional (RUP, por sus siglas en inglés). La misma orienta a los diseñadores en las acciones que deben realizar para lograr un entendimiento entre todos los desarrolladores de nuestra aplicación.

RUP es un proceso de desarrollo de software y junto con UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

Aunque RUP es un proceso de desarrollo de software genérico, se concibió en gran medida para el desarrollo de sistemas basados en programación orientada a objetos.

Entre sus características principales están:

- La forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Tiene un desarrollo iterativo.
- Administra requisitos.
- Usa una arquitectura basada en componentes.
- Tiene control de cambios, un modelado visual del software.
- Tiene una verificación de la calidad del software.

1.4.3 POO como técnica de programación

La Programación Orientada a Objetos (POO) es soportada por las nuevas versiones de muchos de los lenguajes de programación de alto nivel. Es una técnica de estructuración, en la que los *objetos* son los principales elementos de construcción que se conectan entre sí por medio de mensajes, y pueden ser entidades reales e incluso entes conceptuales que con frecuencia encontramos en el mundo que nos rodea. Las principales ventajas que brinda esta técnica vienen determinadas por el encapsulamiento, la herencia y el polimorfismo: 1) El encapsulamiento es el proceso de mantener agrupado bajo una misma cápsula o unidad la estructura y el comportamiento de los *objetos*; 2) la herencia es el mecanismo que permite construir *clases* nuevas a partir de clases existentes; 3) y el polimorfismo constituye la capacidad que poseen *objetos* pertenecientes a *clases* diferentes de comportarse en forma (*morfismo*) diferente (*poli*) ante la recepción de un mismo mensaje.

Rational Rose es la herramienta de modelación visual que provee el modelado UML. Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y entender el entorno de los sistemas. Una de las grandes ventajas de Rose es que utiliza la notación estándar en la arquitectura de software (UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común. Además, los diseñadores pueden modelar sus componentes e interfaces de forma individual y luego, unirlos con otros componentes del proyecto.

El sistema que se propone será desarrollado con las técnicas de POO, apoyándose en RUP y Rational Rose y usando como notación el lenguaje UML, que es la base del funcionamiento visual de RUP.

1. 5 Descripción de los lenguajes de programación.

1.5.1 PHP.

“PHP (acrónimo recursivo de "PHP: Hypertext Preprocessor", originado inicialmente del nombre PHP Tools, o Personal Home Page Tools)” es el lenguaje de programación que se utilizará para el desarrollo de esta aplicación. Ya que PHP es de fácil uso y tiene una gran similitud con lenguajes muy comunes de programación estructurada, tales como C++ y Perl, lo que facilita que sea muy fácil de aprender. Aunque para personas que no tengan experiencia es muy fácil de aprender también. También permite involucrarse

con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones y prácticas.

La interpretación y ejecución se realiza en el servidor, en el cual se encuentra almacenada la página y el cliente solo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página Web, enriquecida con código PHP, el servidor interpretará las instrucciones mezcladas en el cuerpo de la página y las sustituirá con el resultado de la ejecución antes de enviar el resultado a la computadora del cliente. Además, es posible utilizarlo para generar archivos PDF, Flash o JPG, entre otros.

Los principales usos de PHP son los siguientes:

- Programación de páginas Web dinámicas, habitualmente en combinación con el motor de base datos MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión.
- Programación en consola, al estilo de Perl, en Linux, Windows y Macintosh.
- Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y GTK (GIMP Tool Kit), que permite desarrollar aplicaciones de escritorio tanto para los sistemas operativos basados en Unix, como para Windows y Mac OS X.

Entre las numerosas ventajas que ofrece PHP pueden destacarse:

- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad tales como MySQL, Postgres, Oracle, ODBC, IBM DB2, Microsoft SQL Server y SQLite, lo cual permite la creación de Aplicaciones Web muy robustas.
- Capacidad de leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- PHP permite ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux, Windows y Mac OS X, y puede interactuar con los servidores de Web más populares.
- Posibilidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee muy buena documentación en su página oficial.

- Es Libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos (POO).

Además de las ventajas mencionadas PHP cuenta con *frameworks* que facilitan el trabajo con este lenguaje. A continuación se explica el porque.

Hoy en día existe una gran cantidad de código que se pudiera reutilizar para aprovechar bien las potencialidades de este lenguaje, pero esto requiere de un trabajo de búsqueda muy extenso cuando se cuenta con un framework de la talla de **CodeIgniter**, el cual nos permite no solo reutilizar el código sino hacerlo de la manera más organizada y eficiente que se puede hacer hoy en día.

CodeIgniter es un Framework para desarrollo de Aplicaciones Web. Su objetivo es poder hacer los proyectos más rápidos de lo que usted puede hacerlos escribiendo código desde el principio, suministrando un conjunto de librerías para tareas comunes, con una simple interface y estructura lógica para acceder a estas librerías. CodeIgniter deja que su creatividad se centre en el proyecto minimizando la cantidad de código necesitado para ejecutar una tarea.

Entre sus particularidades tenemos:

- Ser libre.
- Se ejecuta sobre PHP4 y en PHP5.
- Es rápido.
- Usa el patrón *Modelo Vista Controlador* (MVC).
- Genera URL's limpias.
- Paquetes de librerías.
- Extensible.
- No requiere un motor de plantillas adicional.
- Ampliamente documentado.

1.5.2 Javascript.

Actualmente JavaScript resulta de mucha utilidad pues proporciona una forma eficaz de interactuar con el usuario sin necesidad de ir al servidor y comprobar datos que el usuario ha entrado incorrectamente.

Entre sus características más importantes se destacan:

- Ser un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis similar a la del lenguaje Java y el C++.
- No es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia. Es más bien un lenguaje basado en prototipos, pues las nuevas clases se generan "clonando" las clases base (prototipos) y extendiendo su funcionalidad.
- Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del DOM.

1.5.3 AJAX

"**AJAX**, acrónimo de *Asynchronous JavaScript And XML* (JavaScript y XML asíncronos, donde XML es un acrónimo de Extensible Markup Language), es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma". [5].

El uso de AJAX en este proyecto tiene como objetivo lograr un ambiente "escritorio" en nuestra aplicación Web, permitiendo que el usuario pueda hacer varias peticiones al servidor sin necesidad de estar esperando por las que ha hecho anteriormente.

Entre sus características se encuentran:

- Contiene presentación basada en estándares usando XHTML y CSS.
- Interacción y visualización dinámicas utilizando el Document Object Model (DOM).
- Manipulación e intercambio de datos usando XML and XSLT.

- Transferencia de datos asíncrona usando XMLHttpRequest, y Javascript poniendo todo junto.
- AJAX no es un lenguaje, no tiene estándares, es sólo la integración de varios lenguajes.

Actualmente existen muchas librerías que implementan un conjunto de funcionalidades de AJAX. Entre ellas tenemos *Script.aculo.us*, la cual se utiliza en el desarrollo de esta aplicación Web pues posee un conjunto de funciones predefinidas que facilitan el uso de las grandes prestaciones de AJAX.

Entre sus principales funcionalidades se encuentran:

- Permite el uso de controles Ajax, "aparecer", "desaparecer", "slide", "agitación" y otros efectos visuales en una página Web como arrastrar y soltar que incluye la ordenación de listas y el movimiento de divs.
- Se distribuye mediante descargas en varios formatos de archivo, y también está incluido en Ruby on Rails y otros frameworks de desarrollo Web.
- Muchos sitios en Internet hacen uso de está excelente herramienta, tales como: Gucci, Digg, Apple, fluxiom y muchas mas.

1.6 Descripción de las tecnologías y herramientas.

1.6.1 Servidor Web Apache.

"Apache es la prueba, al igual que el sistema operativo Linux, de que el trabajo voluntario y cooperativo dentro de Internet es capaz de producir aplicaciones de calidad profesional difíciles de igualar. La licencia Apache permite hacer cualquier aplicación con el código fuente (incluso productos propietarios) siempre que se reconozca el trabajo de los programadores de dicho código". [6].

El servidor Apache es un software estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Los módulos del Apache se pueden clasificar en tres categorías:

• *Módulos Base:* Módulo con las funciones básicas del Apache.

• *Módulos Multiproceso:* Son los responsables de la unión con los puertos de la máquina.

• *Módulos Adicionales:* Cualquier otro módulo que le añada una funcionalidad al servidor.

“Las funcionalidades más elementales se encuentran en el módulo base, lo cual hace necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta Apache, optimizando el rendimiento y rapidez del código. El resto de las funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para introducir un conjunto de utilidades al servidor, hay que añadirle simplemente un módulo, de forma tal que no sea necesario reinstalar el software”. [6]

Según la referencia [7], las principales razones de la gran popularidad de Apache, reconocido en muchos ámbitos empresariales y tecnológicos, y por las cuales se selecciona como servidor Web de la aplicación a desarrollar, son:

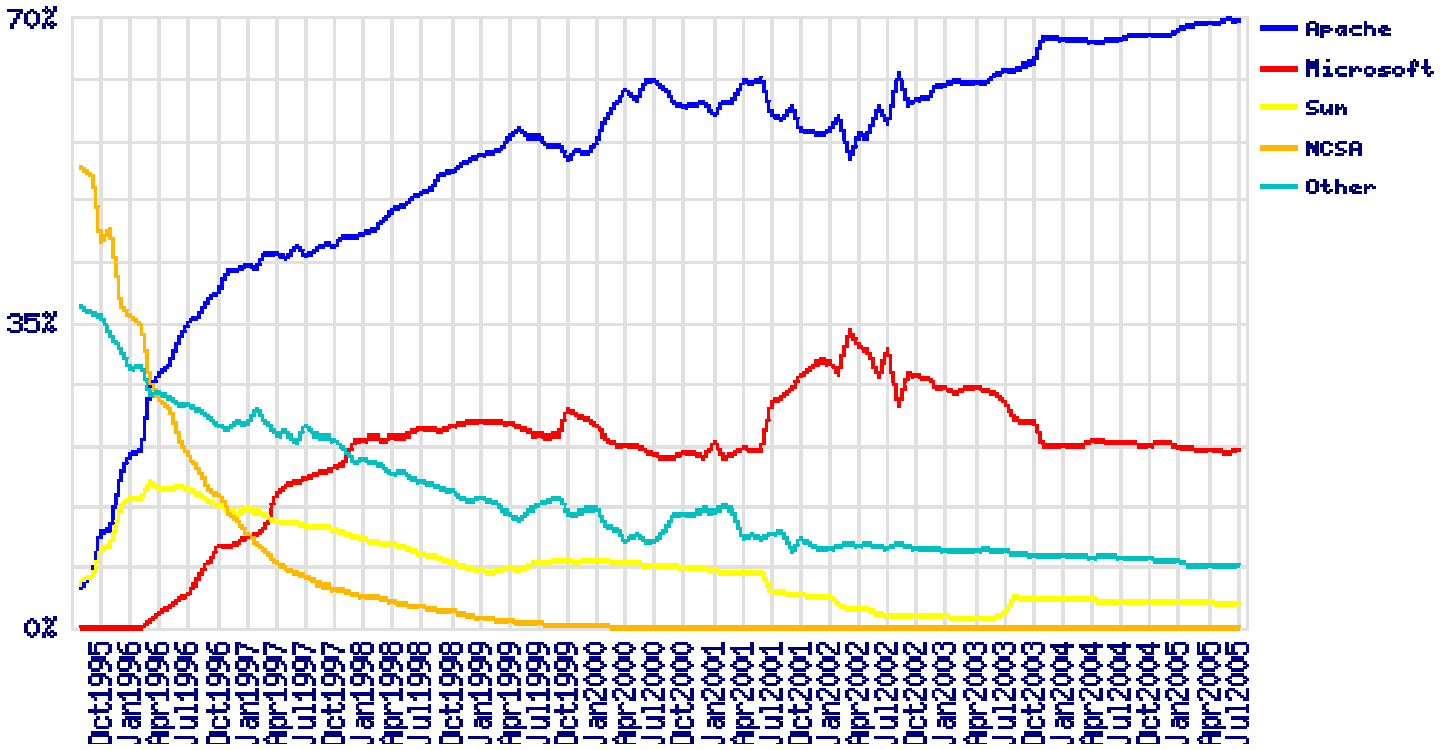
• Compatibilidad con Windows NT/9x, Netware 5.x, OS/2, y en la mayoría de las versiones de Unix, así como en otros sistemas operativos, lo cual lo hace prácticamente universal.

• Tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto proporciona una transparencia extraordinaria a este servidor Web.

• Es un servidor de diseño modular altamente configurable y resulta muy sencillo ampliar sus capacidades. Otro aspecto importante es que cualquiera que posea suficiente experiencia en la programación de C o Perl, puede escribir un modulo para realizar una función determinada.

• Permite personalizar la respuesta ante posibles errores que puedan ocurrir en el servidor. Es posible configurar Apache para que ejecute un determinado script, cuando ocurra un error específico.

Comparación del servidor apache con otros servidores Web. [8].



1.6.2 Gestor de Base de Datos PostgreSQL.

PostgreSQL es un servidor de base de datos relacional libre, liberado bajo la licencia BSD. Es una alternativa a otros sistemas de bases de datos de código abierto como MySQL, así como sistemas propietarios como Oracle o DB2.

Algunas de sus características más relevantes son:

- Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas (*foreign keys*) Disparadores (*triggers*).

- Vistas Integridad transaccional Acceso concurrente multiversión (no se bloquean las tablas, ni siquiera las filas, cuando un proceso escribe).

- Capacidad de albergar programas en el servidor en varios lenguajes. Herencia de tablas Tipos de datos y operaciones geométricas.

- Su gran escalabilidad. Es capaz de ajustarse al número de CPU y a la cantidad de memoria que posee el sistema de forma óptima, permitiéndole soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos casos, se dice que ha llegado a soportar el triple de carga de lo que soporta MySQL).

- Soporta distintos tipos de datos: Además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP,...), cadenas de bits, etc. También permite la creación de tipos propios.

- Incorpora funciones de diversa índole: Manejo de fechas, geométricas, orientadas a operaciones con redes, etc.

- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.

- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

- Introduce el uso de *rollbacks*, subconsultas y transacciones, lo cual redunda en un funcionamiento mucho más eficaz y ofrece soluciones en campos en las que MySQL no incursiona.

- Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, tales como Oracle.

- Por otra parte, los mayores inconvenientes que pudieran aparecer durante el trabajo con este gestor son:

- Consume gran cantidad de recursos.

- Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.

- Es 2 a 3 veces más lento que MySQL.

Por lo tanto, aunque MySQL desarrolla mayor velocidad de análisis y consume menos recursos del sistema. Debido a las mencionadas ventajas que ofrece PostgreSQL y que la aplicación es un modulo de lo que será el sistema general de la gestión de información sobre la docencia en la facultad II se selecciona PostgreSQL como el sistema gestor de base de datos a emplear.

1.6.3 Editor de código PHP Zend Studio.

Zend Studio es una poderosa herramienta para el trabajo con aplicaciones Web usando el lenguaje de programación PHP, y que se utilizará para facilitar el trabajo con este lenguaje de programación. Cabe destacar que actualmente existen muchas herramientas que pueden realizar esta misma operación, aunque las bondades de Zend Studio superan las de cualquier otra.

Zend Studio, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado; la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. La parte del programa que permite escribir los scripts es bastante útil para la programación en PHP. La interfaz está compuesta por varias partes, en las que encontramos un explorador de archivos, una ventana de depuración, los menús y otra para mostrar el código de las páginas.

El programa en conjunto está implementado en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más.

Lo más notable es que contiene una ayuda contextual con todas las librerías de funciones del lenguaje que asiste en todo momento, ofreciendo nombres de las funciones y parámetros que deben recibir. Aunque esta ayuda contextual no solo se queda en las funciones definidas en el lenguaje, sino que también reporta ayudas con las funciones que vayamos creando nosotros, incluso en páginas que

tengamos incluidas con la función *include().* Otras ayudas que ofrece a la hora de escribir son las típicas en editores avanzados, como permitir editar varios archivos, y moverse fácilmente entre ellos, marcar a qué elementos corresponden los inicios y cierres de las etiquetas, paréntesis o llaves, moverse al principio o al final de una función, identificación automática del código.

Las mejores cualidades de Zend Studio no se encuentran en la parte de edición sino en las posibilidades de gestión de proyectos y depuración. La barra de la izquierda, que permite navegar los archivos de nuestro ordenador, también dispone de herramientas para gestionar los proyectos, muy útiles para mejorar la productividad en la programación. Los proyectos permiten guardar mucha más información al programa sobre los archivos, discos, servidores, etc. que se gestionen en nuestras aplicaciones PHP.

Una vez que los archivos se han añadido al proyecto se pueden guardar señales como puntos de ruptura en las depuraciones, asimismo, cuando se ejecuta Zend Studio, se vuelven a abrir los archivos que estuvieran abiertos la última vez que el programa se cerró y las herramientas de completar código mejoran sus comportamientos, asumiendo toda la información de los archivos relacionados con el proyecto.

Zend Studio implementa además unas interesantes opciones para trabajar en grupo, al integrar el sistema de trabajo conocido como Control Version System (*CVS*). Sin duda, más de una vez los programadores de PHP se han visto en un duro problema por no encontrar un error en algún script que está dando resultados inesperados. Para facilitar el trabajo, Zend Studio dispone de una herramienta muy interesante de debug o depuración. Gracias a ella podemos ejecutar páginas y conocer en todo momento el contenido de las variables de la aplicación y las variables del entorno como las cookies, las recibidas por formulario o en la sesión. Podemos colocar puntos de parada de los scripts y realizar las acciones típicas de depuración.

Además de la ventana para visualizar el contenido de las variables, dispone de otras donde muestra la salida del script según se va generando, y otra donde se pueden ver las alertas y errores. Las posibilidades se completan con distintos tipos de depuración, en local, en remoto o a partir de una URL.

1.6.4 Rational Rose.

Esta es herramienta es de gran importancia ya que permite hacer uso correcto de la metodología que seleccionamos. También es la herramienta que más se utiliza durante la enseñanza de la asignatura

Ingeniería de Software que reciben los autores del proyecto, por lo que poseen mayor experiencia que con cualquier otra.

Entre sus principales características se cuentan:

- Es una herramienta para el Modelado Visual mediante el Lenguaje Unificado de Modelación (UML por sus siglas en inglés) de sistemas software.
- Permite especificar, analizar, diseñar el sistema antes de codificarlo.
- Mantiene la consistencia de los modelos del sistema software.
- Chequeo de la sintaxis UML.
- Generación de documento automáticamente.
- Generación de código a partir de los modelos.
- Ingeniería inversa (crea modelo a partir del código).

1.7 Conclusiones

En este capitulo se definió el estado del arte, todas las herramientas, metodologías y lenguajes de programación seleccionados para implementar la aplicación luego de analizar sus ventajas y desventajas con respecto a otros del mercado. Además, se describen las plataformas necesarias para el desarrollo del software, así como sus principales funcionalidades.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capitulo aparece como objetivo principal analizar el flujo de los procesos involucrados en el problema planteado, para definir lo que se desea automatizar, la información que el sistema debe gestionar, así como presentar una propuesta del sistema, haciendo una valoración critica de cómo se desarrollan estos procesos. Utilizando la metodología propuesta, RUP, se pretende usar el Modelo de Negocio, pues es una técnica de probada eficiencia, utilizada con el propósito que sea el propio *negocio* quién determine los requisitos del software a desarrollar. Comprender la estructura y dinámica de la organización en la cual se va a implantar el sistema, garantizar un lenguaje común entre desarrolladores y clientes, y entender los problemas actuales de la organización son los otros objetivos de este capítulo, en el que se pretende brindar una panorámica de los puntos que caracterizan el negocio en cuestión. También se analizarán y definirán los requisitos, que pueden ser funcionales y no funcionales. Además, se exponen los principales artefactos propuestos por RUP para modelar el negocio: *Diagrama de Casos de Uso del Negocio, Diagrama de Actividad de cada Caso de Uso* y *Modelo de Objetos del Negocio*.

2.2 Objeto de estudio.

2.2.1 Problema y situación problémica.

2.2.1.1 Flujo actual de los procesos involucrados en el campo de acción.

El vice-decano de Docencia que es el mas interesado en que los estudiantes se superen hace un pedido al jefe de Departamento de la Especialidad sobre la disponibilidad de CO; este último consulta con los profesores para que ellos le informen si van impartir algún curso. De ser así, les orienta la confección del P1 del mismo. Con esta información el jefe del Departamento de la Especialidad crea los cursos en un documento Excel.

Posteriormente, debe enviar un correo electrónico al vice-decano de Docencia, o ir personalmente a entregarle el documento Excel. El vice decano de Docencia debe planificar en el mismo Excel el horario que se le asignará a cada curso. Una vez concluida la planificación del horario de los cursos, el vice-decano de Docencia envía a todos los estudiantes y profesores el documento Excel.

Para hacer efectiva la matrícula en algún CO los estudiantes deben presentarse en el Departamento de la Especialidad e inscribirse en el curso que les interesa. Además de la disponibilidad de cursos, es decir, capacidad de alumnos, se debe verificar que el estudiante cumple los requisitos del CO pues está claro que los estudiantes del primer año no pueden participar en cursos planificados para los estudiantes de quinto.

El listado para controlar la asistencia de los estudiantes al curso deben pasar a recogerlo los profesores de cada CO. Por último, el profesor debe entregar a la secretaria de Docencia el listado oficial de los estudiantes que asistieron al curso cuando este finaliza, con las respectivas notas para que ella consolide la información correspondiente a todos los CO y publique los resultados en la plataforma “Akademos”.

2.2.1.2 Análisis crítico de cómo se ejecutan actualmente esos procesos, las causas que originan la situación problémica y las consecuencias.

Como se ha explicado anteriormente los cursos optativos son creados en un documento Excel. El trabajo en estos documentos es muy tedioso, consume mucho tiempo porque un documento Excel no le ofrece ninguna funcionalidad en este tema y el trabajo con esos documentos suele ser muy engorroso cuando se trabaja con mucha información.

Este documento Excel se envía posteriormente al vice-decano de docencia utilizando la intranet de la UCI para planificar el horario de los CO otro proceso que requiere de tiempo y que contribuye al envió innecesario de paquetes por la red.

Posteriormente los estudiantes y profesores reciben este documento Excel utilizando de nuevo la red corporativa para ver los CO en que pueden matricular y mirar el horario a impartir los CO respectivamente. Documento que tienen que revisar completo para encontrar la información que les interesa.

Al estudiante ir al departamento de la especialidad tiene que anotarse su nombre y datos en una hoja para matricular en el CO, muchas veces lo hacen sin tener en cuenta los requisitos del CO y si esta llena la matricula, provocando un trabajo innecesario al jefe del departamento pues debe revisar si existe algún estudiante que no cumpla con estos requisitos. Además los estudiantes no pueden ver el P1 de los cursos

optativos lo cual les permite valorar si los temas que se impartirán ya los conocen y si desean recibirlo o no. Todo esto sin mencionar las colas innecesarias que se crean en el departamento de la especialidad.

El listado donde se matricularon los estudiantes es una simple hoja que el profesor la utiliza para controlar la asistencia de los estudiantes al curso, destacar que si se extravía, el profesor ya no contará con los estudiantes que matricularon.

La documentación referente al curso tienen que ser publicarla en maquinas personales de los profesores para que los estudiantes pueden consultarla lo cual no es lo mas adecuado por que esta información puede estar ubicada en un servidor con esa tarea.

Los locales de los cursos pueden cambiar, cuando esto ocurre los estudiantes tienen que buscar en el docente para ver si encuentran el laboratorio donde se esta impartiendo el curso, lo cual provoca

tardanzas y inasistencias al los CO que si se analiza se ha realizado todo un proceso para que después se pierda el CO por nada.

En ocasiones la entrega de los listados por parte de los profesores no se efectúa en la fecha planificada, lo que provoca atrasos en la publicación de las notas porque la secretaria de docencia debe esperar a que ellos le proporcionen el listado de los estudiantes con las notas que alcanzaron en el CO.

2.2.2 Objeto de automatización.

En el proceso de publicar un curso será automatizado la creación de los cursos, que también incluye adicionarle, planificar los horarios y asignarle los locales donde serán impartidos. Para este proceso el jefe del Departamento de la Especialidad creará a través de la aplicación el CO, lo cual será notificado por correo a la secretaria de Docencia. Al acceder a la aplicación el sistema le informará que cursos le faltan por asignarle los locales y el horario correspondientes.

El proceso de activar y desactivar los cursos optativos para que los estudiantes conozcan los cursos optativos que están disponibles.

El proceso de obtener el listado de los estudiantes matriculados en el curso por el profesor que lo impartirá se hará con ayuda del sistema. En ese sentido la aplicación le mostrará al profesor los cursos que el imparte, seleccionará un curso y se generará un reporte con los estudiantes matriculados.

El proceso de matricula de los estudiantes en el curso será igualmente automatizado. Para esto el estudiantes deberá acceder al sistema y le mostrará los cursos en los que él puede matricular. Seguidamente, él seleccionará el curso y se matriculará dependiendo de la capacidad de matrícula del CO. El sistema validará que los estudiantes cumplan los requisitos para poder matricular en el curso.

Además, se automatizará el proceso de búsqueda de información para un curso. El profesor, mediante el sistema, publicará cualquier documento con información complementaria para que sea consultado por los alumnos.

2.2.3 Información que se maneja.

El sistema trabajará con las siguientes entidades: 1) *Estudiante:* almacenará toda la información necesaria del estudiante, tales como el nombre, apellidos, grupo, cantidad de cursos matriculados y las notas en esos cursos. 2) *Profesor*: tendrá el nombre del profesor, apellidos y cursos que imparte. 3) *Curso*: tendrá el título del curso, perfil, asignatura a la que pertenece, matricula, lugar donde se impartirá, año de estudio al que está dirigido, fecha de comienzo, fecha de conclusión, tiempo de duración, modalidad, nivel de complejidad, año, semestre, curso escolar, recursos, contenido por horas, profesor y estudiantes. 4) *Recursos*: tendrá un nombre del recurso y una descripción.

2.3 Propuesta de sistema.

El sistema gestionará toda la información referente a los CO. El sistema debe ser capaz de crear, eliminar y modificar los CO, así como asignarle laboratorios y los horarios a un curso, los cuales se podrán actualizar en caso de cambios. En el sistema los estudiantes podrán matricular “online”, esta matricula controlará los requisitos que debe cumplir el estudiante para ingresar al curso optativo. Generará reportes con los estudiantes y sus notas en los CO. También debe dar la posibilidad de obtener listados de los estudiantes en un curso, modificar los datos de esos estudiantes, eliminarlos, agregarlos y adicionar documentación relacionada con el curso. La solución que existe hasta el momento es la que está vigente en la Facultad II que no es eficiente pues casi todo el control de la información se hace en documentos Excel de forma manual. Además, existe la aplicación “Akademos”, pero este sistema sólo almacena las notas de los estudiantes en los CO. La propuesta se diferencia totalmente pues es un sistema automatizado y supera la solución existente en los siguientes aspectos: 1) Rapidez para acceder a la

información de la aplicación, 2) Más eficiencia al manejar los datos, 3) Los usuarios contarán con una aplicación sencilla y de fácil uso que estará disponible en todo momento, 4) La información estará más centralizada.

2.4 Modelo de negocio actual.

2.4.1 Reglas del negocio objeto de automatización.

- Los cursos optativos solo pueden ser creados por el jefe del Departamento de la Especialidad.
- Los estudiantes solo pueden matricular en dos cursos.
- Los estudiantes tienen que pertenecer a la Facultad II.
- Los estudiantes tienen que cumplir con los requisitos del curso.
- Los profesores son los encargados de publicar la información referente a los cursos.
- Los profesores tienen que entregarle las notas a la secretaria de Docencia, como también el listado oficial de los estudiantes que participan en el curso.
- El vice-decano de docencia es el encargado de asignar los locales y planificar el horario de los cursos optativos.

2.4.2 Actores y Trabajadores del negocio.

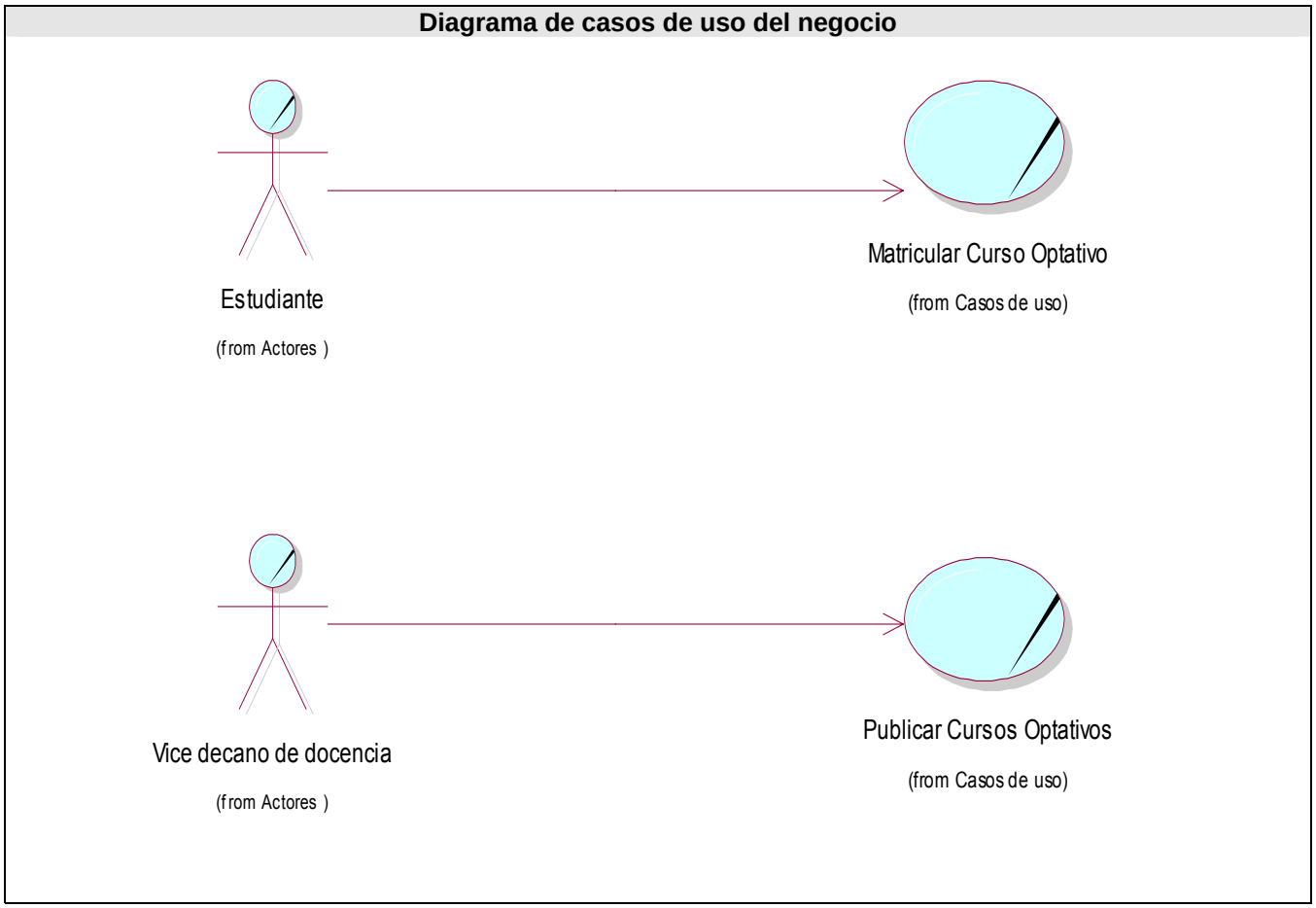
Tabla 2.1 Descripción de los actores del negocio.

Actores del negocio	Justificación
Estudiante	El estudiante es el que matricula en los cursos optativos es de su interés obtener nuevos conocimientos y tener más de 8 cursos aprobados para poder graduarse.
Vice-decano de docencia	Es el encargado en que los estudiantes reciban más de ocho CO antes de graduarse.

Tabla 2.2 Descripción de los trabajadores del negocio.

Trabajadores del negocio	Justificación
Jefe del Departamento de la Especialidad	El jefe del Departamento de la Especialidad es el encargado de gestionar los CO.
Profesor	El profesor es el encargado de crear el programa analítico del CO que va a impartir.

2.4.3 Diagrama de casos de uso del negocio



2.4.4 Descripción de los Casos de Uso del Negocio.

Ver Anexo No. 1

2.4.5 Diagramas de actividades de los casos de uso del negocio.

Un diagrama de actividad demuestra la serie de actividades que deben ser realizadas en un proceso del negocio, así como las distintas rutas que se pueden ir desencadenando. Este es dividido en canales, donde cada canal representa el actor que está llevando a cabo la actividad y muestra cómo se utilizan las entidades del negocio. Para observar los Diagramas de Actividad **Ver Anexo No. 2**.

2.4.6 Diagrama de clases del modelo de objetos

Un modelo de objetos del negocio es un modelo interno a un negocio. Describe como cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo. Para ver el Diagrama de clases del modelo de objeto.

Ver Anexo No. 3.

2.5 Especificación de los requisitos de software

Los requisitos son capacidades y condiciones con las cuales debe cumplir el sistema ya sean funcionales o no funcionales.

2.5.1 Relación de los requisitos funcionales.

Los requisitos funcionales indican el comportamiento del sistema. Posteriormente estos requisitos son modelados a través del diagrama de casos de uso del sistema.

El sistema debe:

R.1 Gestionar los CO.

R.1.1 Crear CO.

R.1.2 Eliminar CO.

R.1.3 Modificar CO.

R.2 Matricular estudiante en un CO.

R.3 Gestionar materiales de un CO.

R.3.1 Adicionar materiales.

R.3.2 Eliminar materiales.

R.4 Insertar la nota a un estudiante en el CO.

R.5 Obtener listado de los estudiantes matriculados en el CO.

R.6 Gestionar local y horario a un CO.

R.6.1 Asignar local y horario.

R.6.2 Modificar local y horario.

R.7 Autenticar usuario.

R.8 Generar reportes.

R.9 Gestionar usuarios.

R.9.1 Crear usuario.

R.9.2 Eliminar usuario.

R.9.3 Modificar usuario.

R.10 Realizar Búsquedas.

R.11 Ver materiales de un CO.

R.12 Cancelar matricula en un CO.

2.5.2 Relación de los requisitos no funciones.

Los requisitos no funcionales especifican las propiedades que debe poseer el sistema y aunque no alteran las funcionalidades del mismo, sí añaden características que implican un mejoramiento del producto final.

1- **Apariencia o interfaz externa:** El sistema debe tener una apariencia agradable, con colores y letras suaves, coherente con la imagen de las aplicaciones Web que existen en la universidad. Los mensajes de error deben ser mostrados de una manera sencilla, indicando las posibles causas que le dieron origen; el color de estos mensajes debe ser rojo. Predominará el azul, con algunas tonalidades amarillas. correspondiéndose con los colores que caracterizan nuestra entidad.

2- **Requerimientos de Usabilidad:** El sistema debe ofrecer facilidad de uso para aquellos usuarios sin experiencia en el trabajo con aplicaciones Web, pues será consultado por los estudiantes y trabajadores relacionados con la gestión de los cursos optativos. Además, debe ser accesible a través del sitio de la facultad.

3- **Requerimientos de Rendimiento:** Producto del tamaño que pueden alcanzar la base de datos y la complejidad de las consultas es poca, resulta viable perfeccionar el rendimiento. Lo que garantiza que la respuesta a solicitudes de los usuarios se produzca en un período de tiempo breve.

4- **Requerimientos de Soporte:** El sistema puede estar en un servidor o distribuido en dos servidores, el servidor de base de datos y el servidor de aplicaciones Web. En caso de los dos servidores se deberán realizar pruebas en cada uno de los servidores finales en los que quedará instalado el sistema con el propósito de identificar los posibles errores de configuración.

5- **Requerimientos de Portabilidad:** El simple hecho de utilizar tecnologías de código abierto para la implementación del sistema garantiza, que la aplicación pueda instalarse en diversas plataformas, fundamentalmente en dos de los sistemas operativos más populares, Linux y Windows 2003.

6- **Requerimientos de Seguridad:** La información que el sistema gestiona no es crítica. Aunque la administración de usuarios y las notas de los estudiantes deben ser protegidas. Para eso el gestionara el acceso a los diferentes módulos de tal forma que solo los autorizados puedan trabajar con este tipo de información. Los servidores de bases de datos y de aplicaciones Web, deben ubicarse en un local donde se garantice la seguridad física de los mismos. La facultad debe encargarse de mantener los servidores actualizados de antivirus, de parches de seguridad y actualizaciones del sistema operativo.

7- **Requerimientos de Confiabilidad:** El sistema debe validar todas las posibles acciones del usuario en el proceso, teniendo en cuenta los privilegios de cada uno.

8- **Requerimientos de Software:** Para el desarrollo del sistema se seleccionó, por las características ya mencionadas, PHP como lenguaje de programación en el servidor Web, PostgreSQL como gestor de base de datos y Apache como servidor de aplicaciones Web.

Por lo tanto, los requisitos de software para el servidor de la aplicación son:

- Windows Server 2003, RedHat 8, Debian Sarge 3.1.
- PHP versión 5.1.4.
- Servidor Web Apache Versión 2.2.2.
- Gestor de base de datos PostgreSQL 8.2.

Para la unidad del cliente solo se requiere Mozilla Firefox 1.5 o versión superior e Internet Explorer 6.0 o versión superior.

9- **Requerimientos de Hardware:** La microcomputadora donde se ubique el servidor de aplicaciones Web requiere, como mínimo, un procesador Pentium IV, ~1.4 GHz y 512 MB de memoria RAM, El servidor de bases de datos, no tiene que tener mucha capacidad de almacenamiento ya que el crecimiento de la información no suele ser rápido y puede tener las mismas características que el servidor Web. No obstante, ambos servidores, el de base de datos y el de aplicaciones Web, pueden ubicarse en la misma unidad, si las características de hardware de esta lo permite.

En el caso del cliente, deberá disponer de una microcomputadora Pentium 2 o superior, con 64 MB de memoria RAM. Ambas unidades, servidor y cliente, deben poseer adaptadores de red para conectarse a la Intranet.

10- **Restricciones en el diseño y la implementación:** Debe ser una aplicación Web desarrollada con técnicas de programación Orientada a Objetos y herramientas de desarrollo de distribución gratuita. El diseño del sistema sea hará utilizando el patrón de arquitectura de software *Modelo Vista Controlador (MVC)* ya que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos.

11- Requerimiento de Ayuda y documentación en línea: El sistema contará con documentación en línea para que los usuarios puedan conocer como se pueden llevar a cabo las acciones que quieren realizar.

12- Requerimiento de Legalidad: La plataforma escogida para el desarrollo de la aplicación, debe estar basada en la licencia GNU/GPL.

2.6 Modelo de los Casos de Uso del Sistema.

2.6.1 Definición de los actores del sistema.

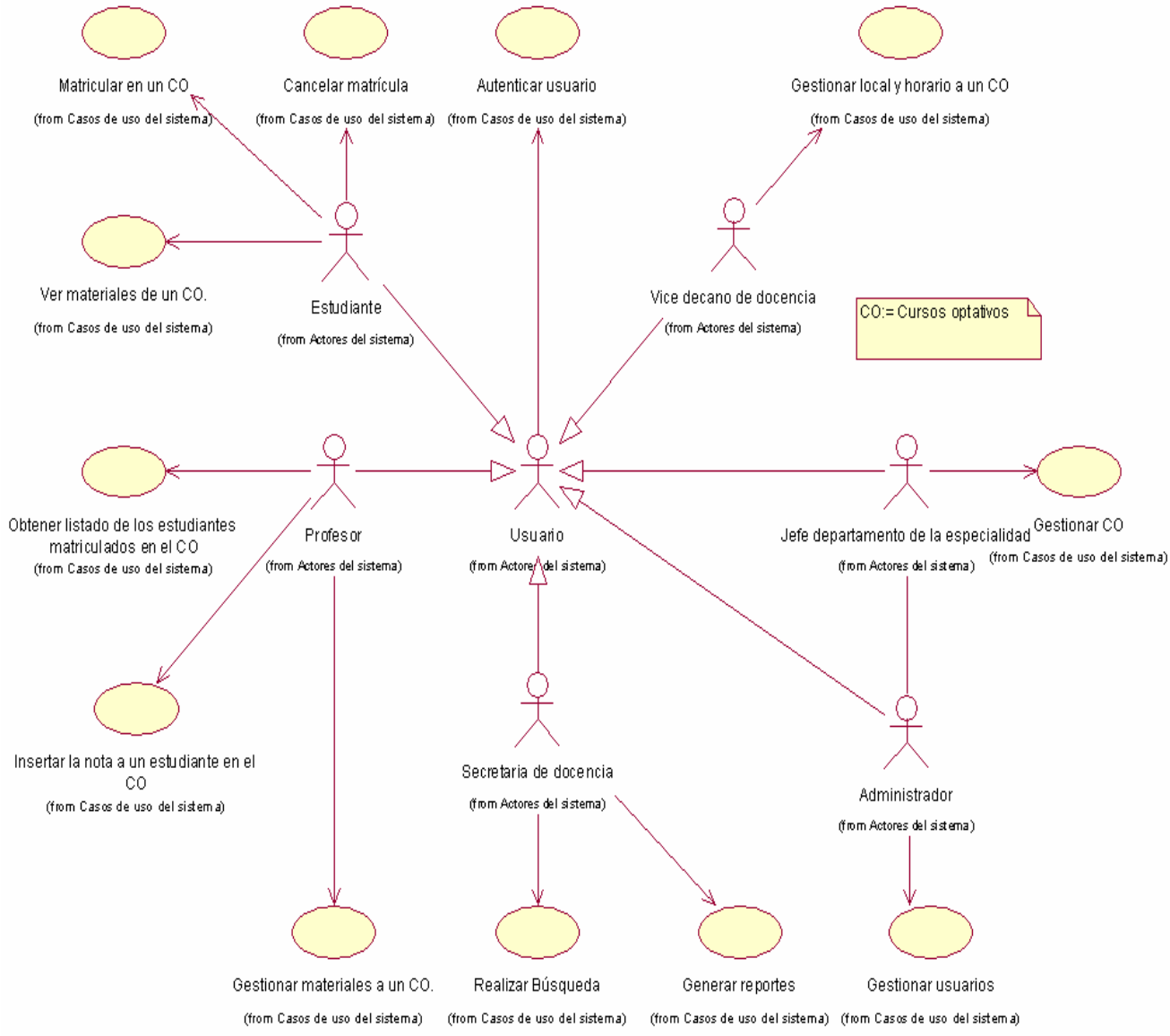
Tabla 2.5 Descripción de los actores del sistema.

Actores	Justificación
Jefe del departamento de la especialidad	Es el encargado de gestionar los CO esto incluye actividades como: • Eliminar un curso optativo. • Crear un curso optativo. • Modificar un curso optativo.
Vice-decano de docencia	Es el encargado de asignar los locales y planificar los horarios de los CO o modificarlos para en el caso que acontezca algún cambio.
Estudiante	Es de su interés matricular en los CO y ver los materiales asociados al curso para estudiar.
Secretaria de docencia	Es de su interés generar los reportes sobre los listados oficiales de los CO con las notas que alcanzaron los estudiantes en el mismo y realizar búsquedas de un estudiante en específico.

Profesor	Es el encargado de obtener el listado de los estudiantes que matricularon, añadir materiales a un CO, eliminar y adicionar la nota de los estudiantes en el CO que imparte.
Usuario	Todos los usuarios que se autentican en el sistema pues para poder acceder al mismo hay que ser estudiante de la facultad II o ser usuario del sistema. Después que se accede hacen las acciones asignadas según su rol.
Administrador	Es el encargado de gestionar la información de los usuarios del sistema esto incluye crear, modificar, eliminar y asignar permisos a los mismos.

2.6.2 Diagrama de casos de usos del sistema

Diagrama de casos de usos



2.6.3 Descripción de los casos de uso del sistema. **Ver Anexo No. 4**

2.7 Conclusiones

En este capitulo se cumplieron todos los objetivos que se plantearon. Se definió lo que la aplicación automatizara, la información que gestionara y se concretó una propuesta del sistema. Además se hizo una valoración critica de cómo funcionan los procesos comprendiendo así donde están los principales problemas.

Con el uso de la metodología RUP se hizo una modelación de estos procesos y se garantizo un lenguaje común entre los desarrolladores y los clientes ayudando a que se definieran los más exacto posible los requisitos que debe cumplir el sistema.

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA.

3.1 Introducción

En este capitulo se hace referencia a los temas con los que se le da solución a algunas de las necesidades de nuestra aplicación.

Las actividades contempladas en el análisis comenzaron a adentrarnos en el problema a resolver, por lo que a través de ellas representamos una vista interna del sistema en la que, usando el lenguaje de los desarrolladores se refinan los requisitos y se estructuran en base a clases y paquetes. Este proceso continúa en el diseño hasta obtener los objetos que interactúan para cumplir los requisitos funcionales y no funcionales obtenidos.

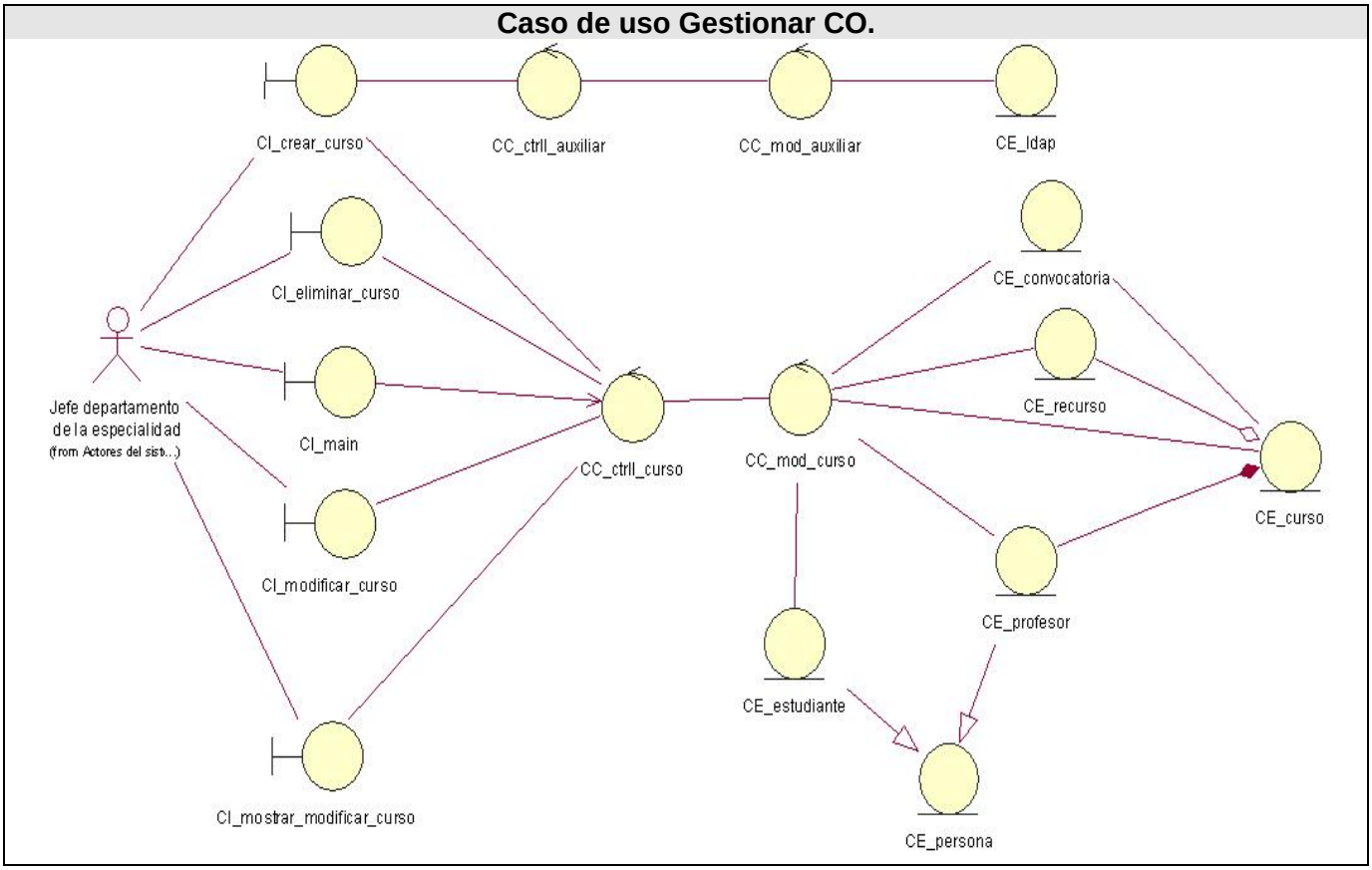
El Modelo de Análisis puede considerarse como una primera aproximación al Modelo de Diseño. Es una entrada fundamental cuando se da forma al sistema en el Diseño y en la Implementación. Además, aparecen en este capítulo la representación grafica de cada uno de los diagramas y la descripción de cada una de las clases, el diagrama entidad relación con sus respectivas tablas y las definiciones de diseño que se apliquen, tratamiento de errores, seguridad, e interfaz.

3.2 Análisis.

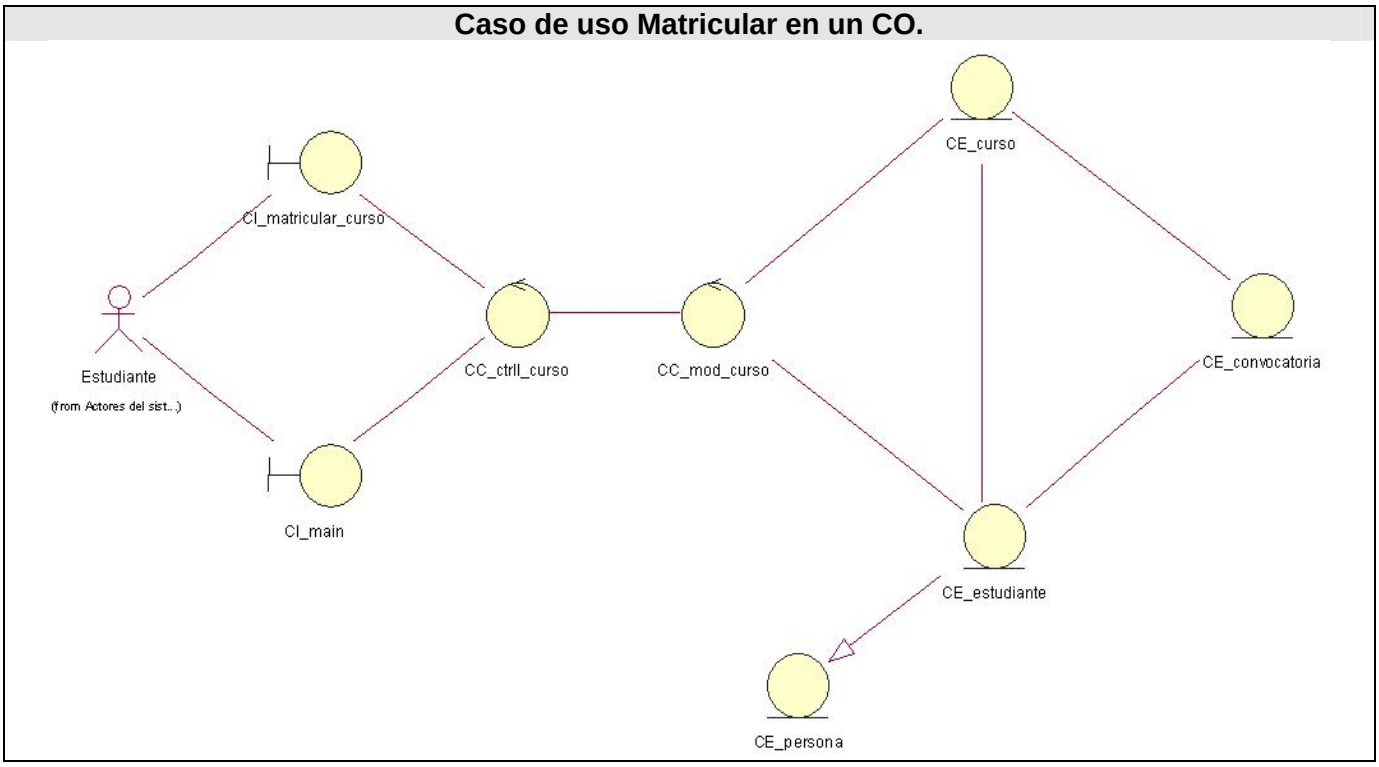
3.2.1 Diagramas de clases de análisis.

3.2.1.1 Gestionar CO.

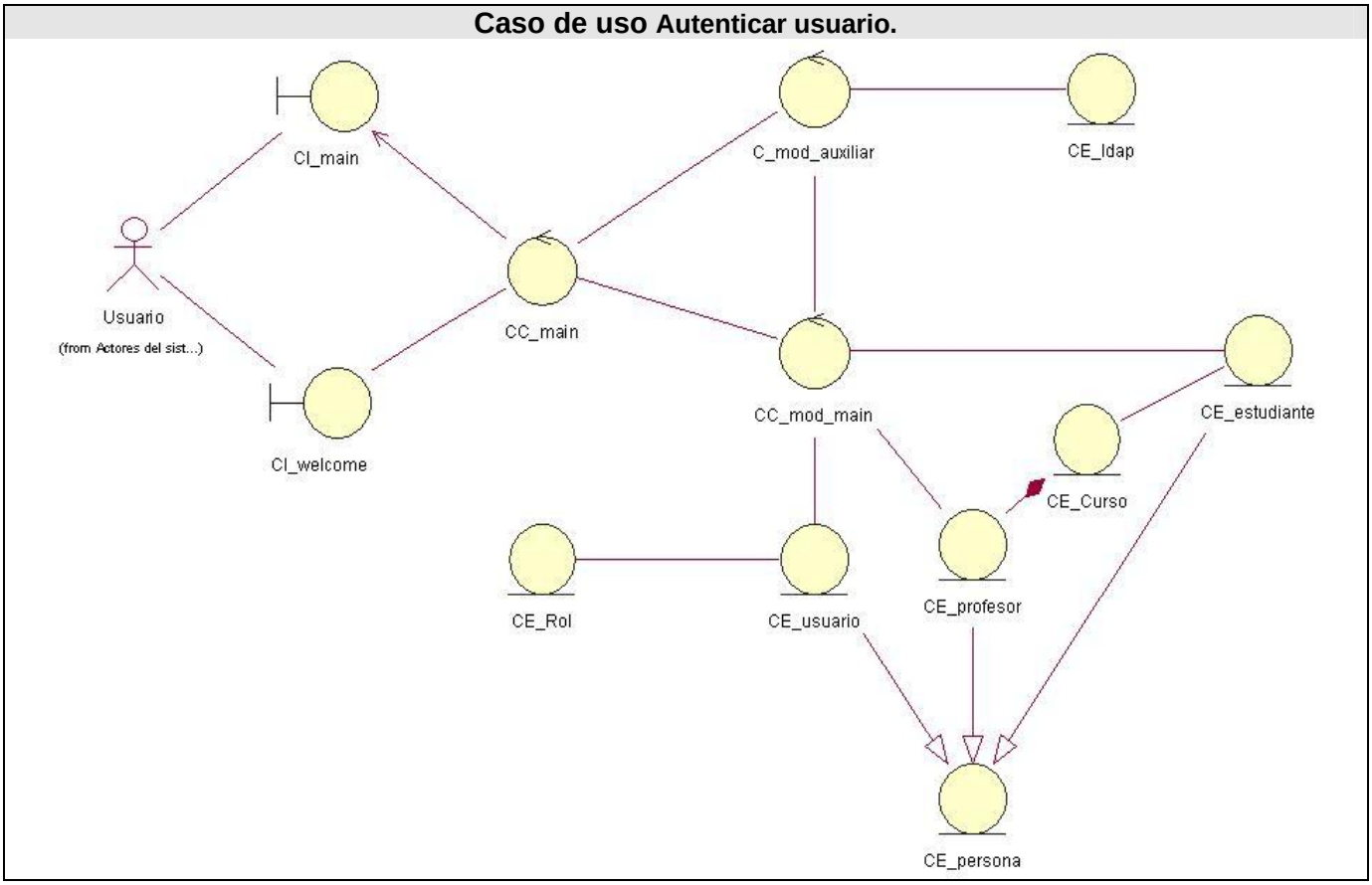
Caso de uso Gestionar CO.



3.2.1.4. Matricular en un CO.



3.2.1.5. Autenticar usuario.



3.3 Diseño.

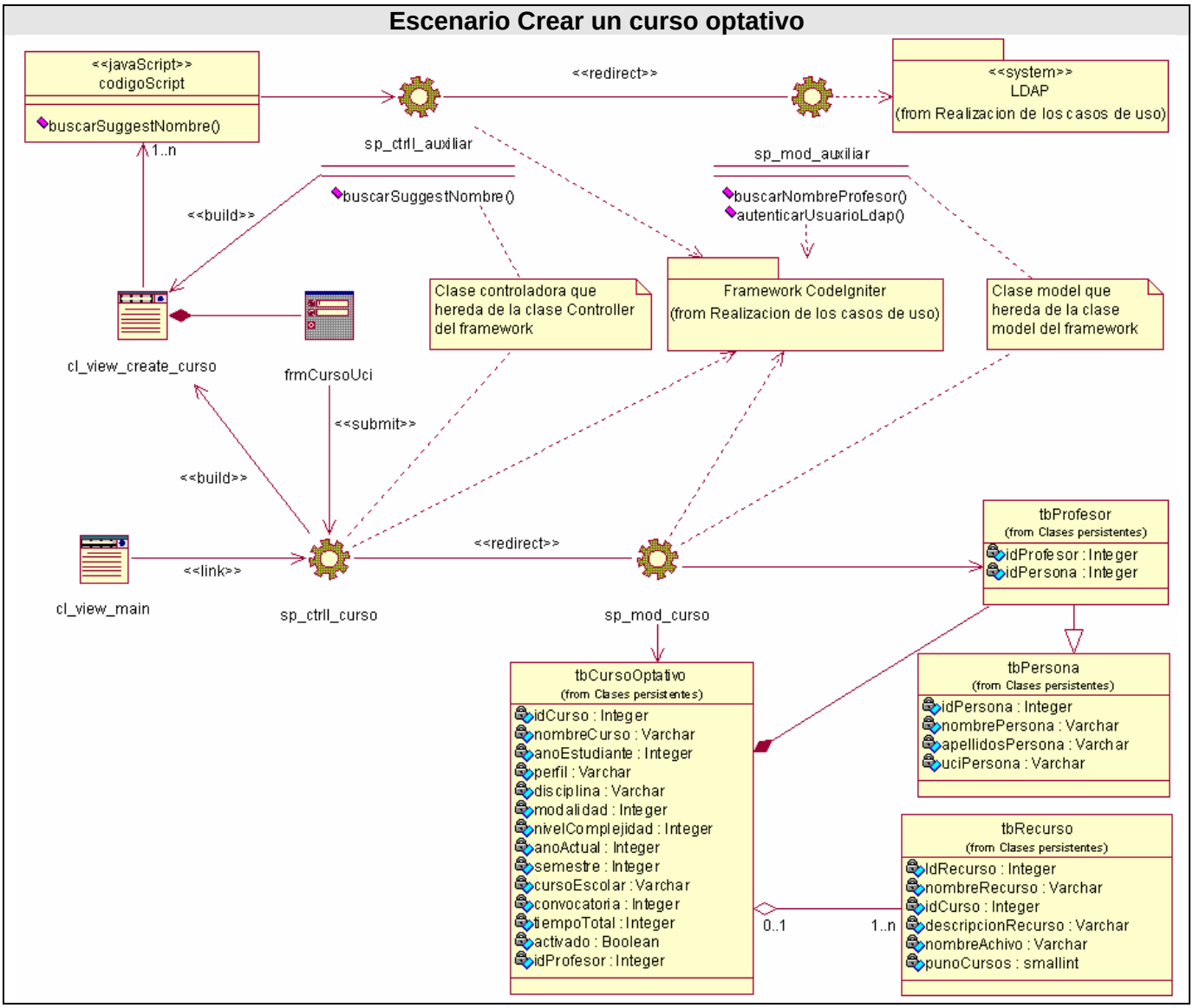
"En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al sistema". [9]

3.3.1 Diagramas de interacción (Colaboración). **Ver Anexo No. 5**

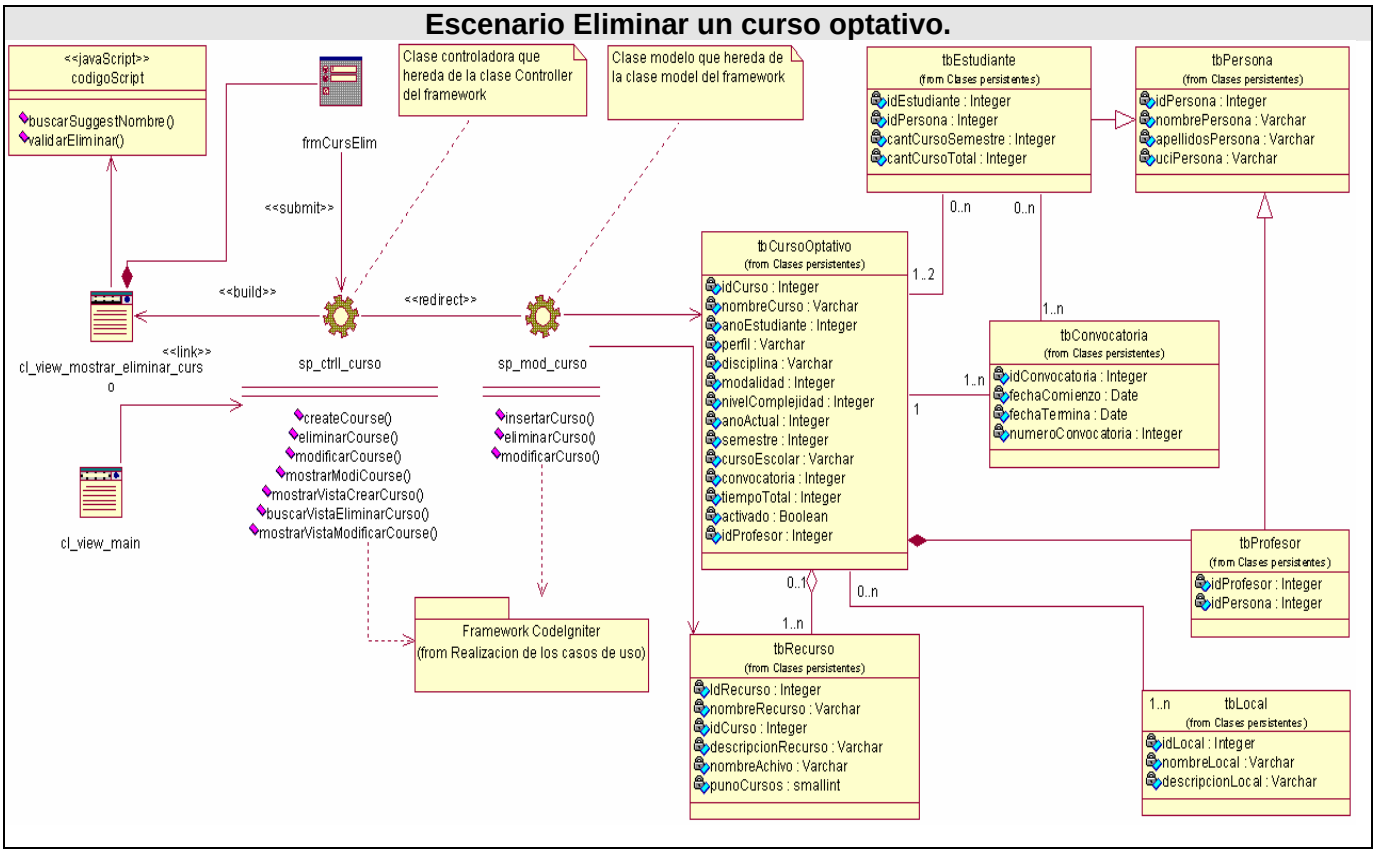
3.3.2 Diagramas de extensiones Web.

3.3.2.1 Caso de uso Gestionar cursos optativos.

1) 3.3.2.1.1 Escenario Crear un curso optativo.

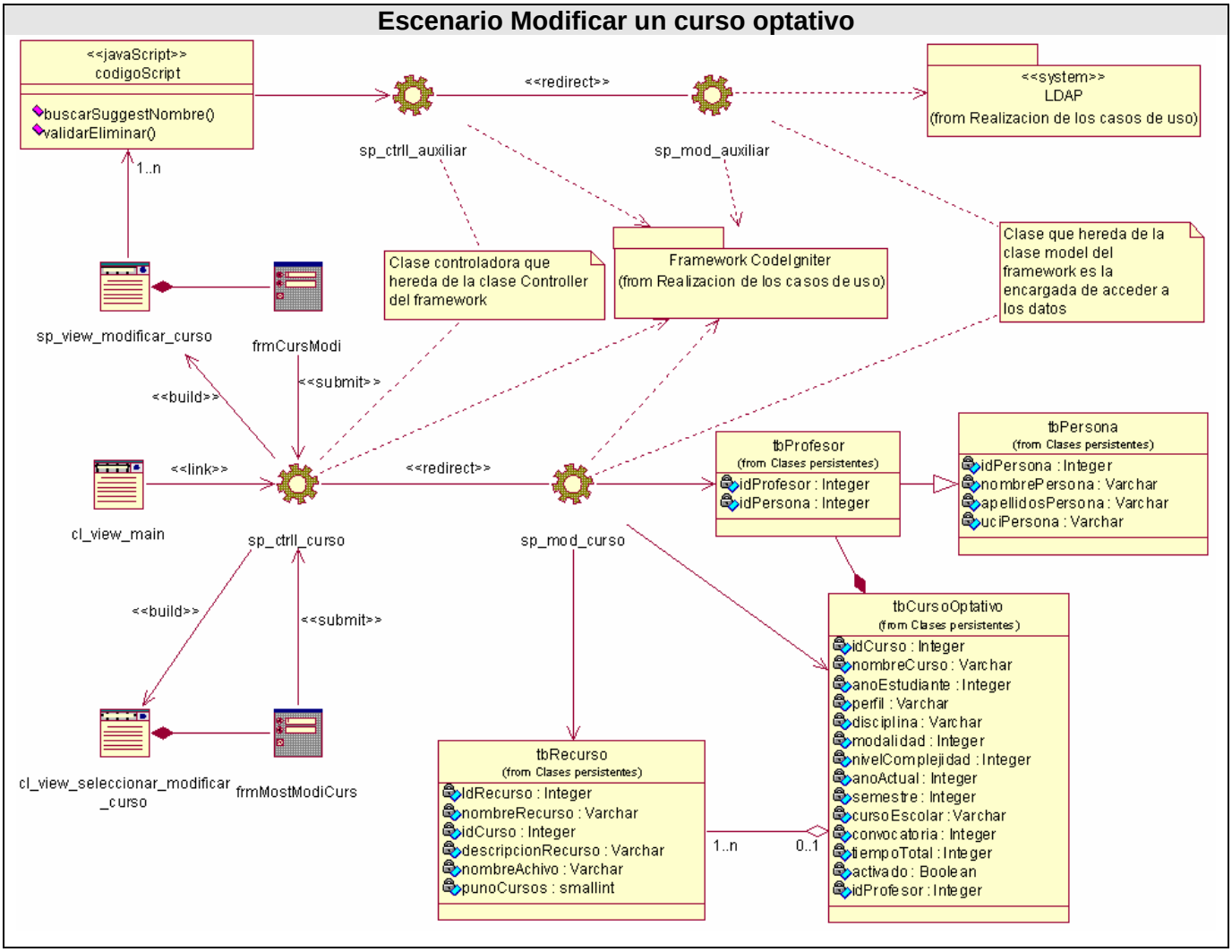


3.3.2.1.2 Escenario Eliminar un curso optativo.

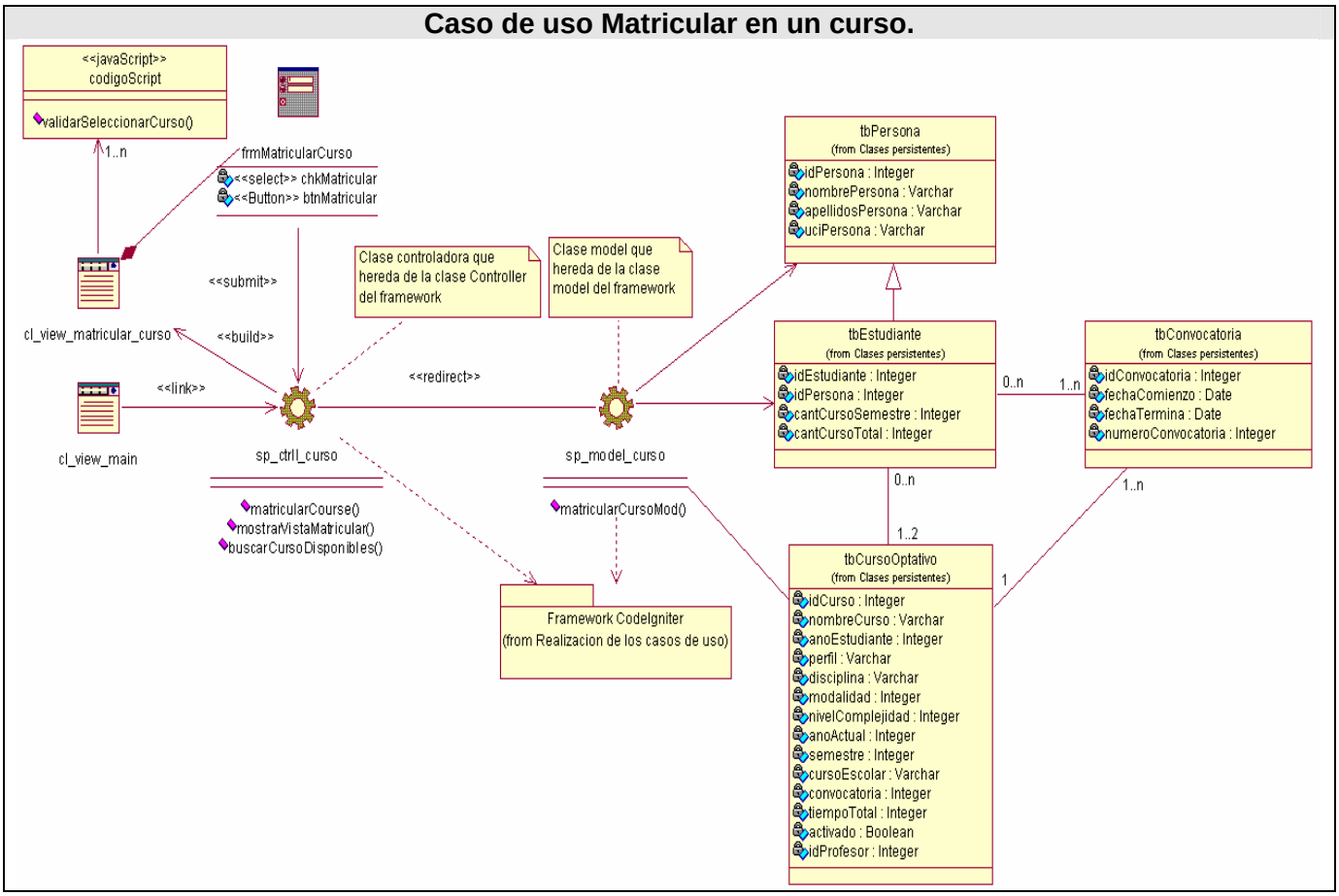


3.3.2.1.3 Escenario Modificar un curso optativo.

Escenario Modificar un curso optativo

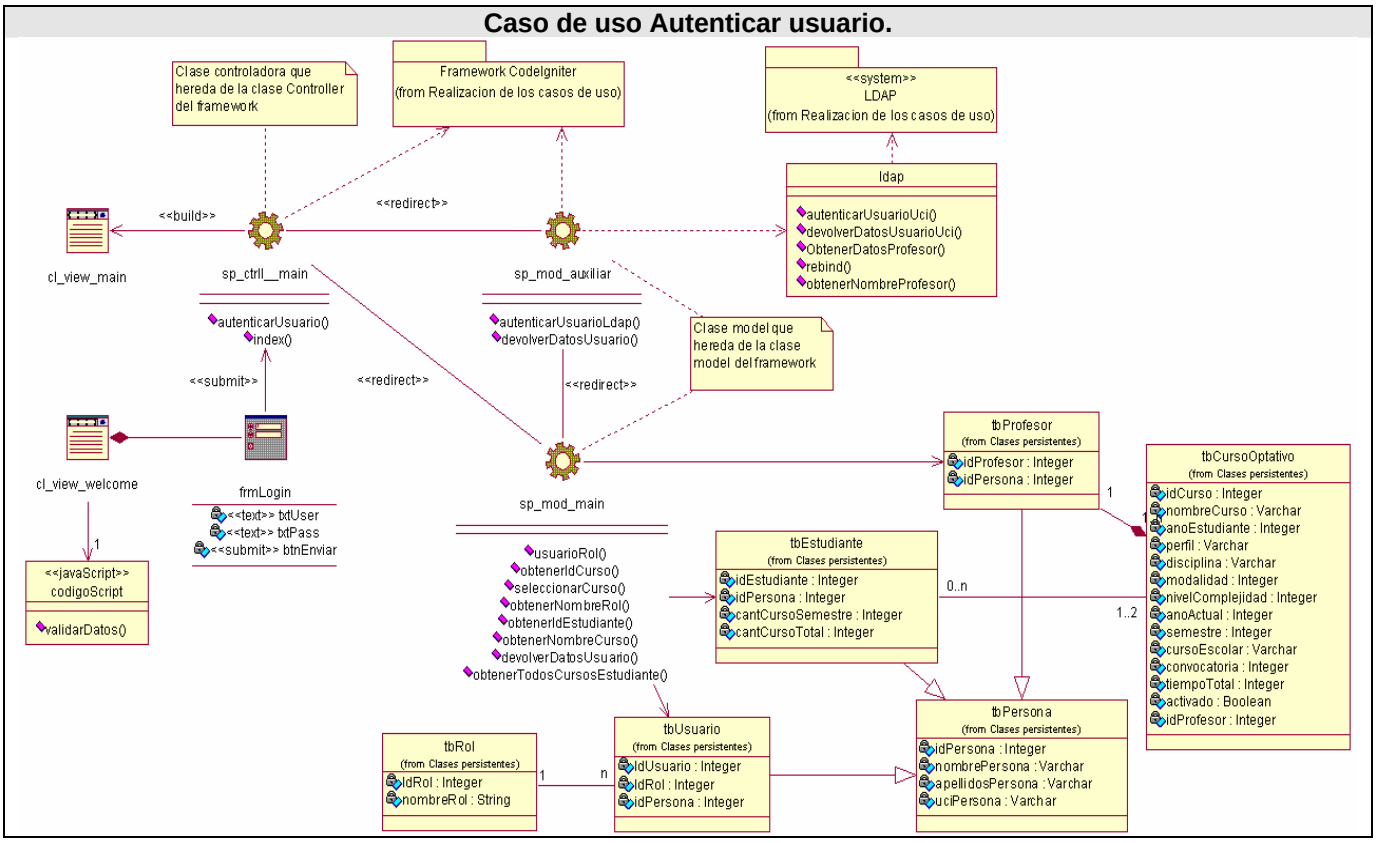


3.3.2.2. Matricular en un curso



3.3.2.3. Caso de uso Autenticar usuario.

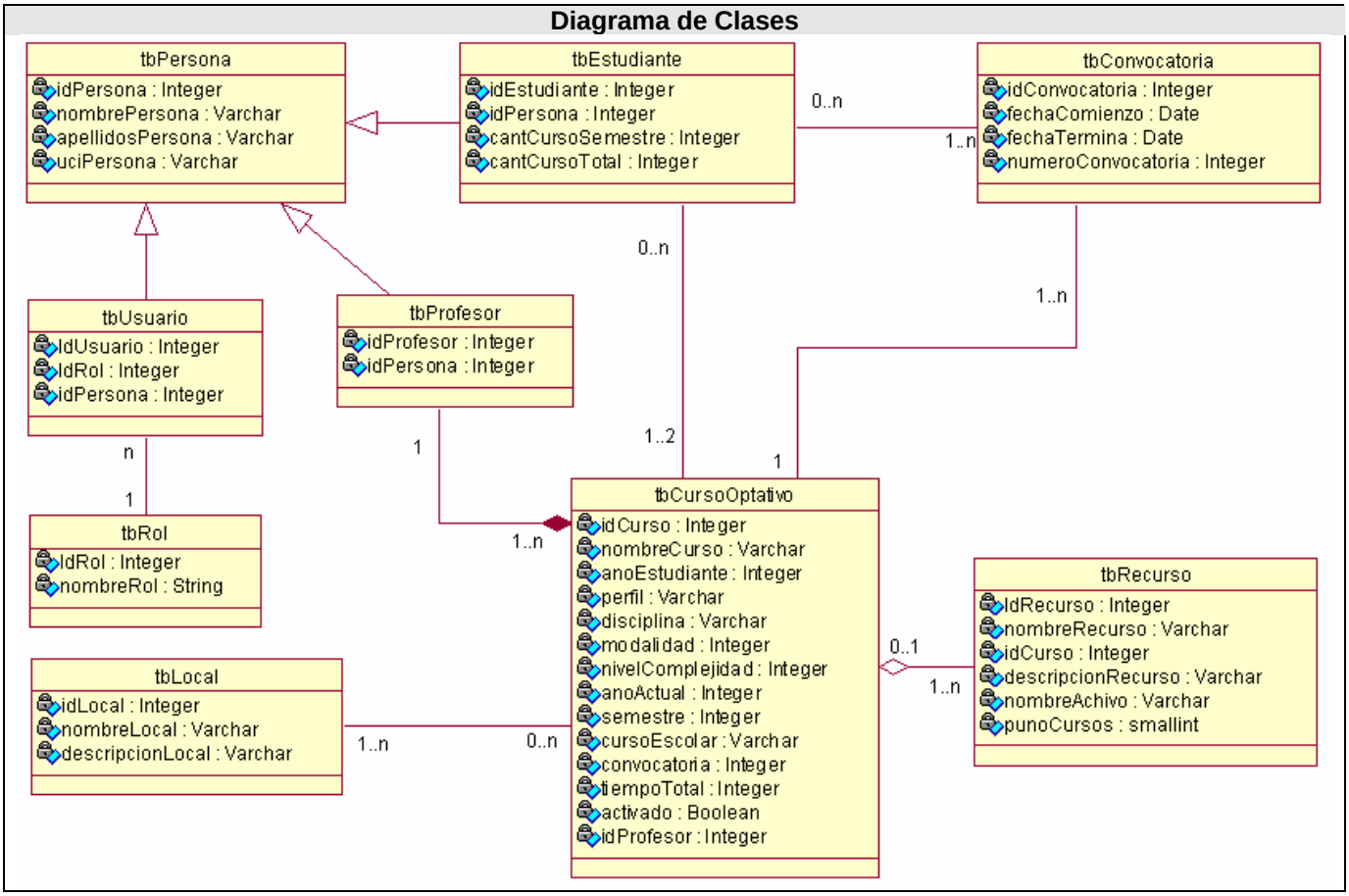
Caso de uso Autenticar usuario.



3.4 Diseño de la BD.

3.4.1 Diagrama de clases Base datos.

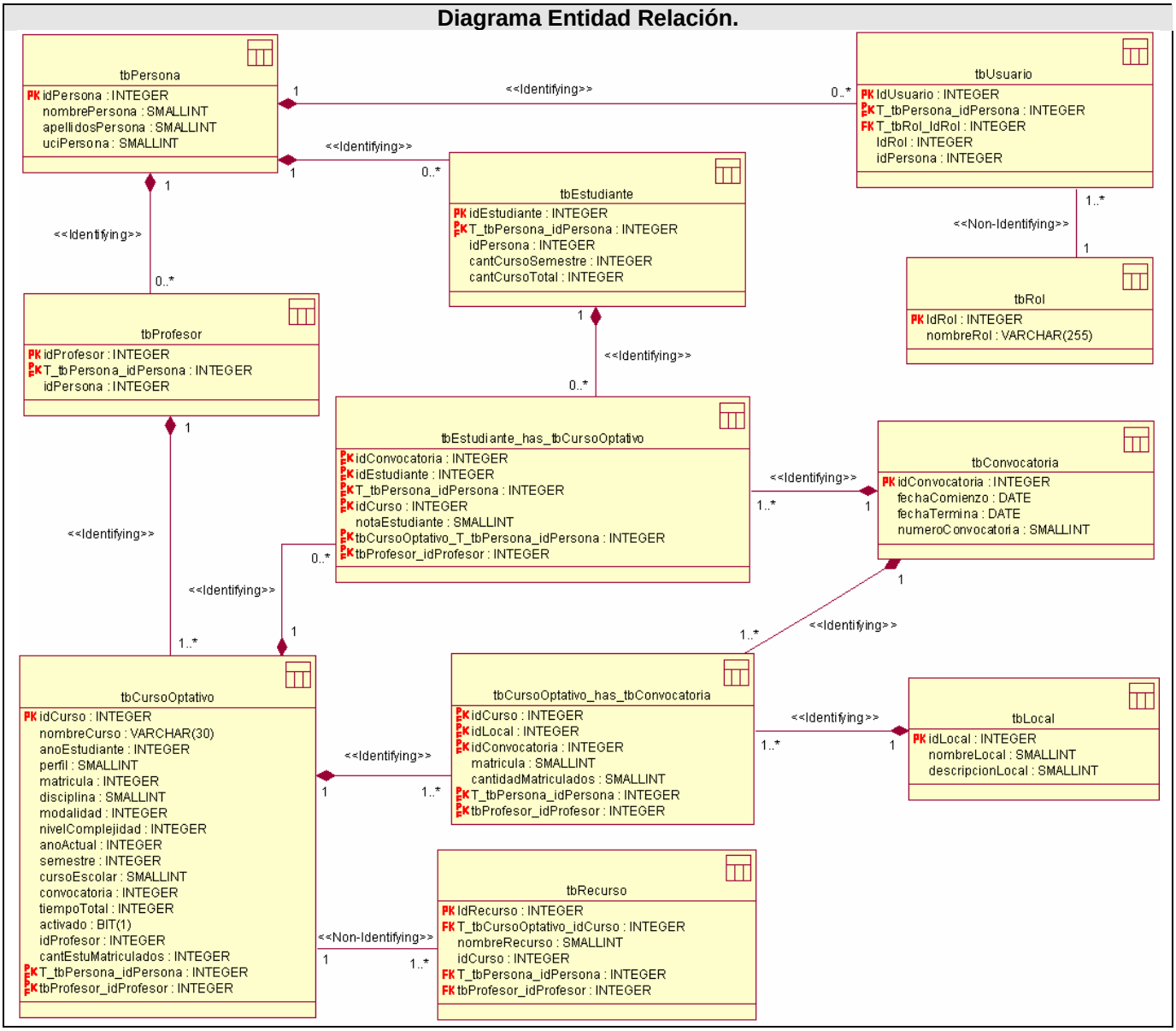
Las clases persistentes son las clases que necesitan ser capaz de guardar su estado en un medio permanente, la necesidad de guardar su estado esta dado por al almacenamiento físico permanente de la información de la clase, para la copia de seguridad en caso del fracaso del sistema, o para el intercambio de información.



3.4.2 Diagrama Entidad Relación de la Base datos.

El modelo de los datos describe la representación lógica y física de datos persistentes en el sistema.

Diagrama Entidad Relación.



3.5 Definiciones de diseño que se apliquen.

En la actualidad existen muchos estándares para la confesión de los sistemas cuando vamos a programar. Los cuales ayudan a que el mantenimiento del software sea más fácil y comprensible. Para la implementación del código se siguió el patrón de arquitectura MVC.

3.6 Tratamiento de errores

En el sistema propuesto se evitan, minimizan y tratan los posibles errores, con el fin de garantizar la integridad y confiabilidad de la información que en este se registra y muestra.

Los mensajes de error que emite el sistema se muestran en un lenguaje de fácil comprensión para los usuarios. Cuando se introduce información en un formulario y faltan datos, sale un cuadro de alerta indicando el campo o dato que falta. Similar ocurre cuando se introduce información errónea en un campo numérico, un campo de letras, un campo que requiere una cantidad específica de caracteres, una cantidad de caracteres por debajo de un valor especifico y cuando el campo es requerido, Todas estas validaciones ocurren de lado del cliente y el servidor, pues se tiene en cuenta la posibilidad de que el usuario deshabilité el JavaScript en su navegador Web.

3.7 Seguridad

La mayoría de los sistemas que se desarrollan en la actualidad gestionan información de mucha importancia. En muchos sistemas esta información necesita ser protegida para que personas ajenas no causen daños que podría reportar grandes pérdidas. Tener en cuenta este aspecto en la implementación y configuración de un sistema contribuye a su eficiencia.

Este importante aspecto se tiene en cuenta desde la misma entrada de datos del usuario ya que son diversas la cantidad de formas que una persona mal intencionada puede atacar el sistemas.

El sistema es capaz neutralizar cualquier intento de ataque con javascript que se quiera realizar. Para ello validad la entrada de datos según el campo que el usuario quiere llenar. Después de pasar este filtro todos estos datos son analizados del lado del servidor utilizando funciones que eliminan los espacios innecesarios, adicionan slash en caso de encontrar alguna comilla y elimina cualquier TAB script que se

introduzca en un campo, deshabilitando la ejecución de estos códigos malignos. También se manejan los errores de conexiones a la base de datos evitando que al ocurrir alguno el sistema le muestre información que puede comprometer la seguridad mostrando alguna información de la base datos o la ubicación del los archivos en el servidor.

3.8 Interfaz

El diseño de interfaces de usuario es una tarea que ha adquirido relevancia en el desarrollo de un sistema, se puede definir como: “el conjunto de trabajos y pasos que seguirá el usuario, durante todo el tiempo que se relacione con el programa, detallando lo que verá y escuchará en cada momento, y las acciones que realizará, así como las respuestas que el sistema dará”.

La calidad de la interfaz de usuario puede ser uno de los motivos que conduzca a un sistema al éxito o al fracaso, es por eso que uno de los aspectos más relevantes de la usabilidad de un sistema es la consistencia de su interfaz de usuario.

Para el desarrollo de la interfaz se tuvo en cuenta los siguientes aspectos:

1). Reducir la carga a la memoria.

2). Atajos a Usuarios expertos.

3). Diseño de diálogos que conducen a una conclusión.

4). Lograr deshacer acciones fácilmente.

5). Que el usuario sintiera la sensación de control.

Una de las premisas fundamentales de la aplicación es la ventaja que proporcionan las interfaces Web sobre las interfaces de comando. Ya que las interfaces Web:

- Proporcionan un ambiente amigable.
- Conducen a un aprendizaje más natural.
- Establecen un “sentimiento” (sobre todo en la uniformidad del ambiente) al usuario que enriquece su experiencia en el uso de la aplicación.

- Además de estos principios, se tuvieron en cuenta las siguientes características:
- Utilizar una misma tipografía, forma y estilo en todas las páginas.
- La facilidad del usuario de poder navegar desde cualquier punto a otro dentro de la aplicación.
- Se tuvo presente siempre el ancho de banda y por ello se utilizaron formato de imágenes de compresión favorables.
- La simplicidad y consistencia, favoreciendo la usabilidad de la aplicación.
- Navegación simple en todas las páginas de la aplicación, de forma tal que siempre sea accesible por el usuario.
- Estabilidad y uniformidad del diseño, para así poder ubicar al usuario dentro del mismo y hacerlo sentir parte de él.

Se utilizó una hoja de estilos para guardar la configuración del diseño para todas las páginas, para los botones y las líneas se utilizaron estos estilos, eliminando así el número de imágenes que demoren la presentación de la página.

Los formularios de entradas ocupan el centro superior y las entradas organizadas por importancia. Se incluye una breve explicación del objetivo del formulario, y alguna especificación con respecto a las entradas.

Se realizan múltiples operaciones en cada página, de forma que el usuario no tenga que moverse tanto dentro de la aplicación, para completar una operación.

3.9 Concepción de la ayuda.

La confección de la ayuda en cualquier sistema o aplicación de software es de gran importancia, pues esta guía y aclara dudas a los usuarios cuando están llevando a cabo una tarea específica. Debido a que el sistema no es sumamente complejo y la interfaz de usuario es capaz de orientar al usuario en las tareas que puede realizar las personas que lo utilizaran no se contara con una ayuda detallada.

3.10 Conclusiones.

En este capitulo se cumplieron todos los objetivos que se plantearon. Nos percatamos que luego de haber incorporado nuevos conceptos y diagramas de la fase de elaboración, es recomendable que no se deje de realizar el modelo Análisis antes de comenzar a definir el modelo del Diseño, garantizando así, en caso de ser necesario, que los cambios que haya que realizar no sean tan difíciles de llevar a cabo. También modelamos los diagramas de Clases y Entidad Relación así como la descripción de los datos que se almacenan en sus tablas. Vimos el tratamiento de errores, la seguridad de nuestro software, así como su Interfaz.

Gracias a estos diagramas se obtuvo la arquitectura candidata del sistema, la cual se refinó a través de las iteraciones durante la fase de elaboración, quedando definida la arquitectura base del sistema a implementar.

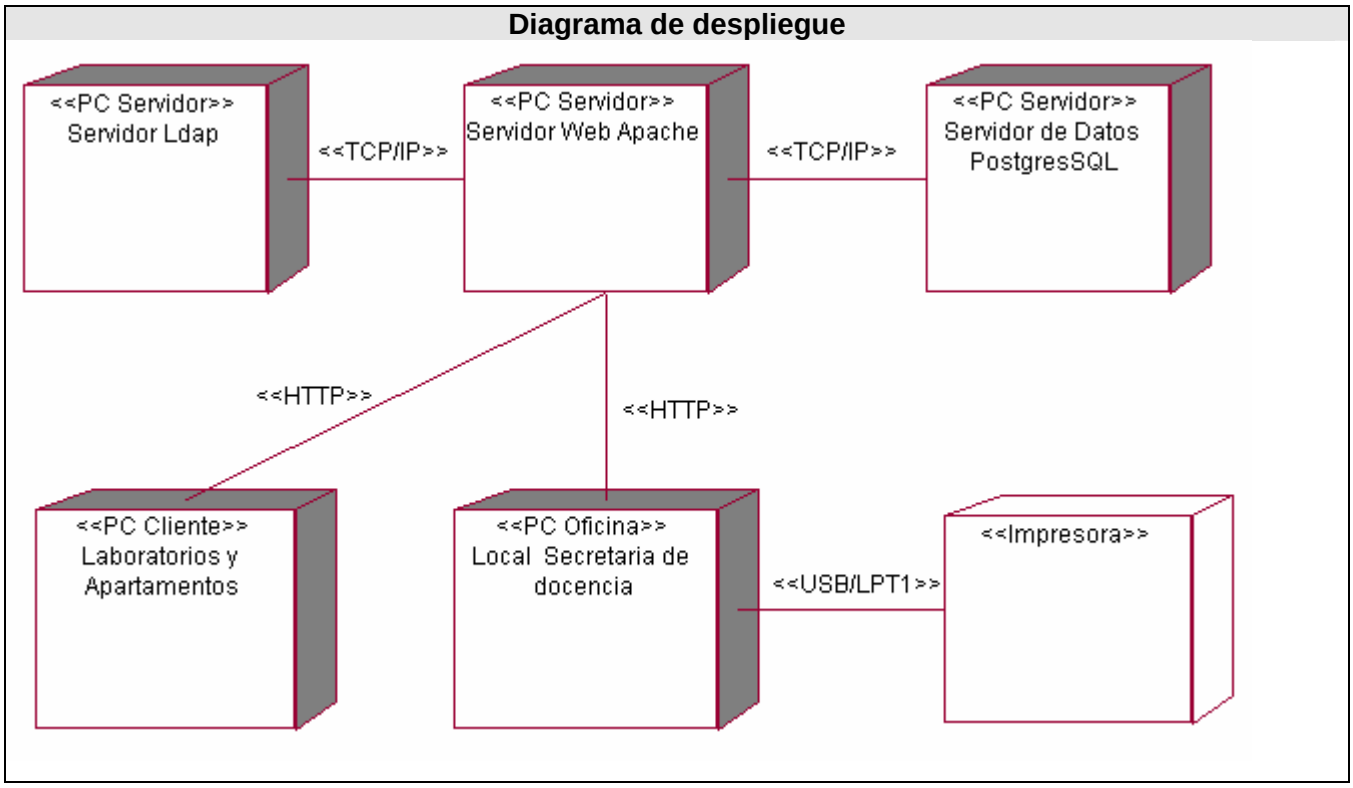
CAPÍTULO IV: IMPLEMENTACIÓN

4.1 Introducción

En este capítulo se demuestra como se implementan en términos de componentes y como se organizan y relacionan los nodos específicos de la aplicación a través de diagrama de componentes y despliegue respectivamente. También hace una descripción del estilo arquitectónico a utilizar.

4.2 Diagrama de despliegue

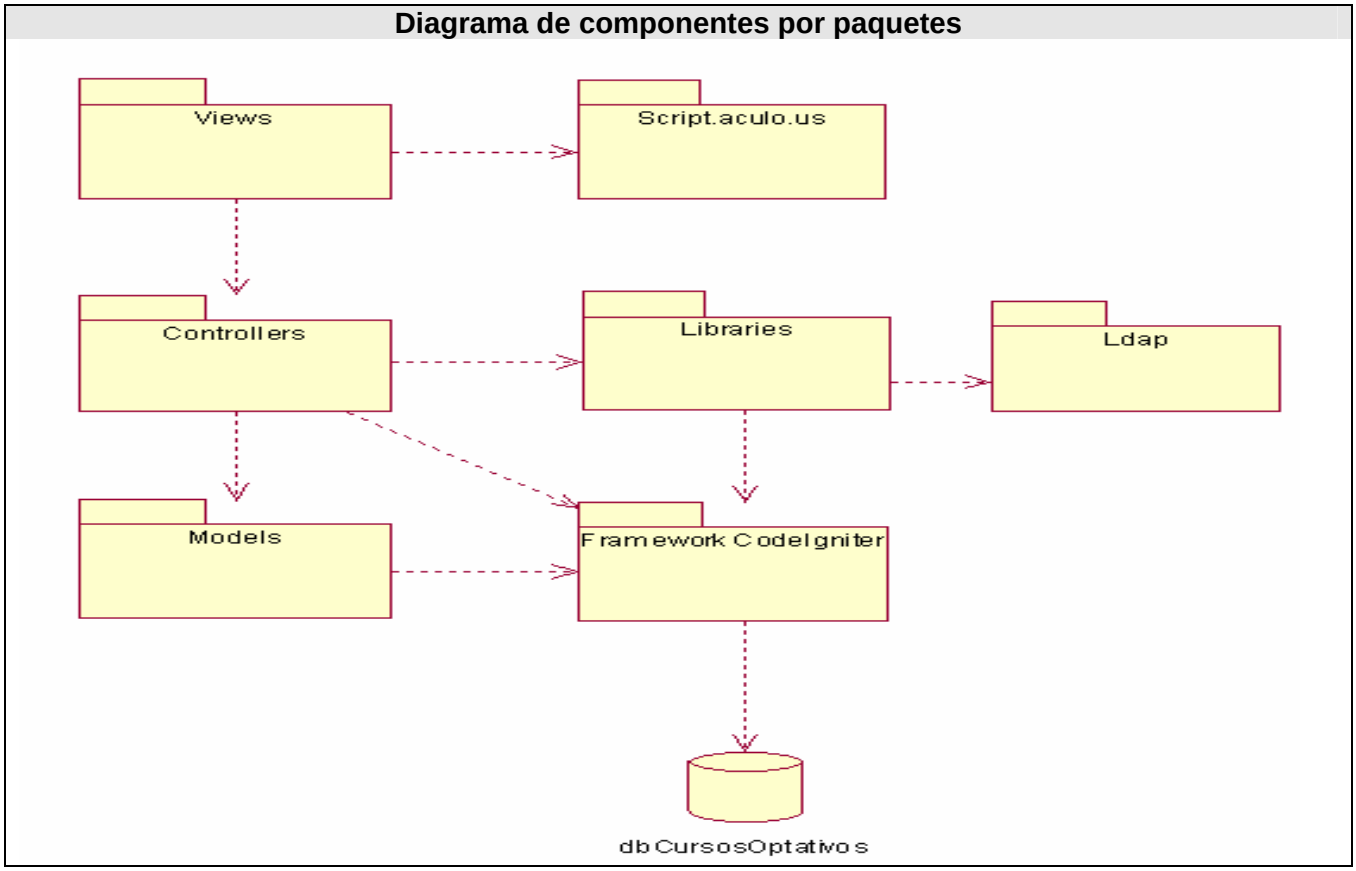
Mediante el diagrama de despliegue podemos ver cómo se encuentran relacionados físicamente los componentes de la aplicación.



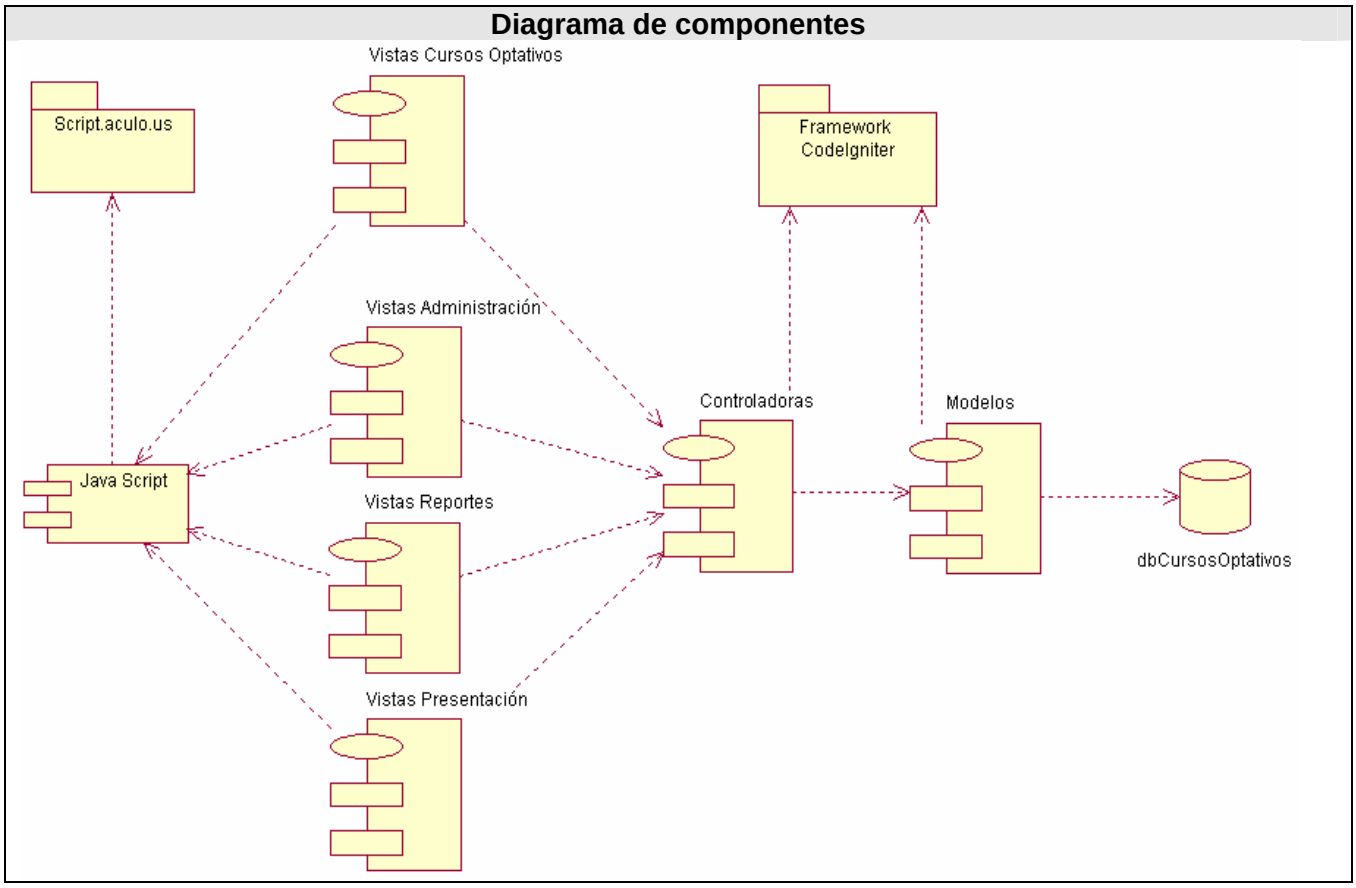
4.3 Diagramas de componentes

4.3.1 Diagrama de componentes por paquetes

Diagrama de componentes por paquetes



4.3.2 Diagrama de componentes



4.4 Conclusiones

Con la culminación de este capitulo se logró representar la aplicación en términos de componentes. Como se organizan los nodos de la aplicación y sus relaciones. Para ello se realizaron los diagramas de componentes y despliegue.

CAPÍTULO V: ESTUDIO DE FACTIBILIDAD.

5.1 Introducción

El estudio de factibilidad es un paso importante en la realización de un proyecto, pues brinda al equipo de trabajo información inicial relacionada con el costo total del producto, tiempo estimado de desarrollo, cantidad de personas que intervienen y los gastos económicos que esto implicaría. En este capítulo se abordarán aspectos relacionados con la estimación de esfuerzos (costes) de desarrollo del sistema.

5.2 Estimación de esfuerzo y tiempo de desarrollo por Puntos de Casos de Uso

5.2.1 Identificar los Puntos de casos de uso Desajustados

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

UUCW: Factor de Peso de los Casos de Uso sin ajustar

- Para calcular UAW

Tipo	Descripción	Peso	Cant * peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface)	1	0*1=0
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2	0*2=0
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3	3*3=3
		Total	**9**

En nuestro caso es complejo

- Para calcular UUCW

Tipo	Descripción	Peso	Cant * peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5	9*5
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10	1*10
Complejo	El Caso de Uso contiene más de 8 transacciones	15	1*15
		Total	**79**

Luego **UUCP = 9+70**

UUCP = 79

5.2.2 Ajustar los Puntos de casos de uso

UCP = UUCP * TCF * EF

Donde:

UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

- Para Calcular TCF

TCF = 0.6 + 0.01 * Σ **(Peso** * **Valor$_i$)** (Donde Valor es un número del 0 al 5)

Significado de los valores

0: No presente o sin influencia,

1: Influencia incidental o presencia incidental

2: Influencia moderada o presencia moderada

3: Influencia media o presencia media

4: Influencia significativa o presencia significativa

5: Fuerte influencia o fuerte presencia

Factor	Descripción	Peso	Valor	Comentario	Σ (Peso$_i$ * Valor$_i$)
T1	Sistema distribuido	2	0	El sistema es centralizado	0
T2	Objetivos de performance o tiempo de respuesta	1	4	El tiempo de respuesta es bastante rápido	4
T3	Eficiencia del usuario final	1	4	Es necesario que el usuario final conozca el sistema	4
T4	Procesamiento interno complejo	1	4	Presenta cálculos complejos	4
T5	El código debe ser reutilizable	1	5	Se requiere que el código sea reutilizable	5
T6	Facilidad de instalación	0.5	3	Posee algunos requerimientos de instalación	1.5
T7	Facilidad de uso	0.5	5	Fácil de usar	2.5
T8	Portabilidad	2	5		10
T9	Facilidad de cambio	1	4		4
T10	Concurrencia	1	0	No hay	3

				concurrencia	
T11	Incluye objetivos especiales de seguridad	1	1		1
T12	Provee acceso directo a terceras partes	1	0	Los usuarios Web no tienen acceso	0
T13	Se requieren facilidades especiales de entrenamiento a los usuarios	1	2	Pocos usuarios internos, sistema fácil de usar.	2
			Total		

TCF = 0.6 + 0.01 * 41
TCF = 1.01

- Para Calcular EF

EF = 1.4 - 0.03 * Σ **(Peso$_i$ * Valor$_i$)** (Donde Valor es un número del 0 al 5)

Factor	**Descripción**	**Peso**	**Valor**	**Comentario**	Σ **(Peso$_i$ * Valor$_i$)**
E1	Familiaridad con el modelo de proyecto utilizado	1.5	3	El grupo está un poco familiarizado con el modelo	4.5
E2	Experiencia en la aplicación	0.5	1	La mayoría del grupo no tiene mucha experiencia en la aplicación	0.5
E3	Experiencia en orientación a objetos	1	4	Grupo programa en Objetos	4

E4	Capacidad del analista líder	0.5	0	No se contrató a un Especialista	0
E5	Motivación	1	4	El grupo está motivado	4
E6	Estabilidad de los requerimientos	2	2	Se esperan cambios	4
E7	Personal part-time	-1	4	el grupo todo el tiempo no es full-time	-4
E8	Dificultad del lenguaje de programación	-1	2	Se usará lenguaje php	-2
			Total		

EF = 1.4 - 0.03 * 11

EF = 1.07

Luego **UCP = 79*1.01*1.07**

UCP = 85.38

5.2.3 Calcular esfuerzo de FT Implementación

E = UCP * CF

- Para calcular CF

CF = 20 horas-hombre (si $Total_{EF} \leq 2$)

CF = 28 horas-hombre (si $Total_{EF} = 3$ ó $Total_{EF} = 4$)

CF = abandonar o cambiar proyecto (si $Total_{EF} \geq 5$)

$Total_{EF}$ = Cant EF < 3 (entre E1 –E6) + Cant EF > 3 (entre E7, E8)

Total = 2+0

CF = 20 horas-hombre (porque $Total_{EF} \leq 2$)

Luego **E = 85.38 * 20 horas-hombre**

E = 1707.6 horas-hombre

Paso 4. Calcular esfuerzo de todo el proyecto

Actividad	**% esfuerzo**	**Valor esfuerzo**
Análisis	10%	
Diseño	20%	
Implementación	40%	1707 horas-hombre
Prueba	15%	
Sobrecarga	15%	
Total	100%	4269

Si **E_T = 1707 horas-hombre** y por cada 240 horas yo tengo 1 mes eso daría un **E_T = 7.11 mes-hombre**

5.3 Conclusiones

En este capítulo se analizó la factibilidad de realización del sistema, se calculó el costo de producción del mismo, el tiempo que se estimó en que debía estar listo, el esfuerzo que debía realizar el equipo de desarrollo y la cantidad de personas necesarias para la realización del sistema, permitiendo confirmar la factibilidad de la construcción del sistema propuesto.

CONCLUSIONES GENERALES

El uso de herramientas o software informáticos en el trabajo cotidiano de cualquier empresa o institución sin duda trae consigo una mejora en el desempeño de la misma. Estas herramientas acompañadas de una documentación que explica como se llevo a cabo todo el proceso de confección influyen en la solución eficiente de muchos problemas que surgen en muchos organismos.

Este trabajo juega un papel beneficioso en el estudio y comprensión de la gestión de la información de los CO en la universidad, específicamente en la facultad II. Aunque es un pionero en este tema el aporte sin duda es muy útil ya que todas las personas involucradas en este tema han madurado su conocimiento y han comprendido como esta herramienta traería consigo mejoras para el trabajo de la facultad.

En este trabajo se considera haber cumplido con el objetivo fundamental, desarrollar una aplicación Web capaz de gestionar la información sobre los CO en la facultad II. También se cumplieron con todos los objetivos específicos. El sistema se desarrollo utilizando en su mayoría herramientas no propietarias lo cual contribuyo a aumentar el conocimiento de estas herramientas adicionando el apoyo a las políticas de la UCI y el país en el uso de este tipo de herramientas.

La aplicación se analizo y diseño con la metodología RUP la cual se considera muy eficiente para resolver el problema con medios informáticos que en un futuro ayudaran a personas interesadas en realizar versiones de la aplicación.

RECOMENDACIONES

- Implementar el sistema usando Web Service.
- Seguir con el estudio del Framework CodeIgniter.
- Seguir con el estudio del Framework Script.aculos.us.
- Continuar el estudio del flujo de la información en la gestión de los cursos optativos para aumentar las funcionalidades del sistema.
- Mejorar el diseño del las páginas de presentación al usuario con persona especializadas en el tema para que cumpla con los estándares de la mayoría de los navegadores Web mas utilizados.

BIBLIOGRAFIA

http://www.maestrosdelweb.com/actualidad/2334/ [Fecha de consulta 12-2-07]

http://www.desarrolloweb.com/articulos/935.php [Fecha de consulta 20-2-07]

http://www.intellia.com.mx/esp/servicios/aplicaciones_web_a_la_medida.php [Fecha de consulta 20-2-07]

http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web [Fecha de consulta 5-3-07]

http://www.avidos.net/blogold/aplicaciones-web/ [Fecha de consulta 25-4-07]

http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos [Fecha de consulta 5-3-07]

http://www.netpecos.org/docs/mysql_postgres/index.html [Fecha de consulta 15-3-07]

http://www.linalco.com/apache.html [Fecha de consulta 12-2-07]

http://es.wikipedia.org/wiki/Modelo_Vista_Controlador [Fecha de consulta 20-3-07]

http://es.wikipedia.org/wiki/Rational_Unified_Process [Fecha de consulta 20-3-07]

http://www.maestrosdelweb.com/editorial/zendstudio/ [Fecha de consulta 20-4-07]

http://wiki.prod.uci.cu [Fecha de consulta 12-4-07]

http://www.vivaphp.com.ar/articulos/168-css-layouts.html [Fecha de consulta 18-3-07]

http://es.tldp.org/Postgresql-es/web/navegable/faq/faq-es.htm [Fecha de consulta 30-3-07]

http://www.php.net/manual/en [Fecha de consulta 20-5-07]

http://www.efectosjavascript.com/botonatras.html [Fecha de consulta 14-4-07]

http://www.forosdelweb.com/showthread.php?t=289269 [Fecha de consulta 24-3-07]

http://alexsancho.name/archives/2006/09/validacion-de-formularios-con-prototype/ [Fecha de consulta 18-2-07]

http://www.htmlquick.com/es/reference/tags/input.html [Fecha de consulta 14-4-07]

http://codeigniter.com/forums/viewthread/51587/ [Fecha de consulta 21-5-07]

http://www.clikear.com/manuales/uml/index.asp [Fecha de consulta 16-4-07]

http://codeigniter.com/forums/viewthread/51313/ [Fecha de consulta 25-04-07]

http://www.postgresql.org/docs/8.1/interactive/triggers.html [Fecha de consulta 17-3-07]

http://www.zend.com/products/zend_studio/professional_edition [Fecha de consulta 5-2-07]

http://www.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/index.html [Fecha de consulta 23-3-07]

Ivar Jacobson, Grady Booch, James Rum Baugh. El proceso unificado de desarrollo de software , Ciudad de la Habana, Cuba, Félix Varela, 2004. 438 páginas.

REFERENCIAS BIBLIOGRAFICAS.

1. **http://www.inhispania.com/Espanol/General/Online_registration.html**
2. **http://www.aularagon.org/matricula_OnLine/matricular.asp?idasignatura=1031**
3. **http://www.ehtpe.co.cu/ofertas/quehacer.htm**
4. Moreno Martínez, Gerardo,. *Ingeniería de Software UML*, **http://www.monografia.com/trabajos5/insof/insof.html** (2007-04-03).
5. Garret, Jesse James. *Ajax: Un Nuevo acercamiento a las Aplicaciones Web [en línea]* **http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php** (2007-04-18).
6. Artículo, *Una Introducción a APACHE* **http://linux.ciberaula.com/articulo/linux_apache_intro** (2006-04-03).
7. Carlos Luís, Cuenca. *Arquitectura del servidor Apache [en línea]* **http://www.helloworldsolutions.com** (2006-04-03).
8. Manuel, Dávila Guerra. *Linux y el software libre [en línea] Blog* **http://tr.eltiempo.terra.com.co/blogs_enter/home/contenidoblog.php?blog=12094979256l** (2007-04-18)
9. Ivar Jacobson ,G. B, James Rumbaugh, , El proceso unificado de desarrollo del software 1999: p. 168.

ANEXOS

Anexo No. 1

Anexo 1.1 Tabla 2.1

Caso de uso del negocio:	Matricular Curso Optativo
Actores del negocio:	Estudiante
Propósito:	Que los estudiantes matriculen en los CO que la Facultad o la Universidad proponen.
Resumen: El caso de uso se inicia cuando el estudiante se personaliza en el departamento de la especialidad y manifiesta su intención de matricular en un CO. Acto seguido es atendido por el jefe de Departamento de la Especialidad, quien le enseña los CO disponibles, El estudiante selecciona un CO y si cumple con los requisitos del mismo es matriculado.	
Curso normal de los eventos:	
Acción del actor	**Respuesta del proceso de negocio**
1- Se Personaliza en el departamento de la especialidad. 1.2 Manifiesta su intención de matricular en un CO.	2- El jefe del departamento de la especialidad recibe la solicitud hecha. 2.1 El jefe del departamento de la especialidad Muestra los CO disponibles.
3- Selecciona un CO.	4- El jefe del departamento de la especialidad verifica que el estudiante cumpla con los requisitos del CO y que no este completa la matricula. 4.1 Si el estudiante cumple con los requisitos y la matricula no esta llena el jefe del departamento de la especialidad informa al estudiante que puede matricular en el CO.
5- Matricula en el CO.	
Flujo alterno de los eventos:	
Acción del actor:	**Respuesta del proceso de negocio:**
	Si el estudiante no cumple con los requisitos del CO o la matricula esta completa el jefe del departamento de la especialidad informa al estudiante que no puede matricular en el CO.
Otras secciones:	

Anexo 1.2 Tabla 2.2

Caso de uso del negocio:	Publicar Cursos Optativos
Actores del negocio:	Vice-decano de docencia
Propósito:	Que los estudiantes puedan consultar los CO disponibles y el horario, Y que los profesores consulten el horario del los CO.

Resumen:

El caso de uso se inicia cuando el vice-decano de docencia hace un pedido al jefe del departamento de la especialidad sobre la disponibilidad de CO, el jefe del departamento de la especialidad consulta con los profesores para que ellos le informen si van impartir algún curso y si lo van a impartir ahí mismo les pide que realice el P1 del curso. Con los datos del CO y el P1 el jefe del departamento de la especialidad crea los CO y se los envía al vice-decano de docencia. El vice-decano planifica el horario y se lo envía a todos los estudiantes y profesores de la facultad. Así finaliza el caso de uso.

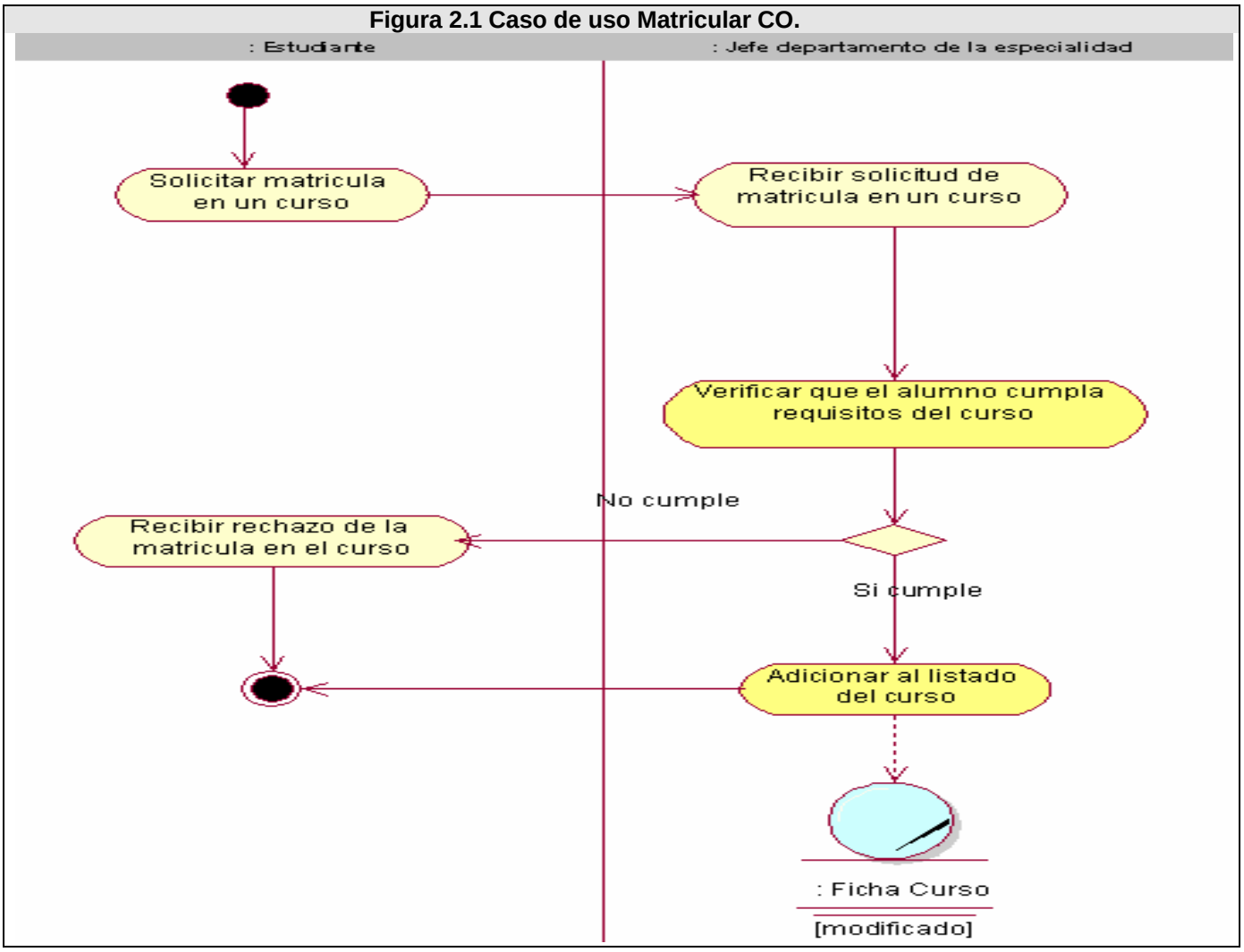
Curso normal de los eventos:	
Acción del actor	**Respuesta del proceso de negocio**
1- Pide al jefe del departamento de la especialidad la disponibilidad de CO.	2- El jefe del departamento de la Especialidad recibe la solicitud hecha. 2.1 Pide el P1 de los CO a los profesores. 2.2 Los profesores le entregan el P1. 2.3 El jefe del departamento crea los CO en un Excel y se lo envía al vice decano de docencia.
3- Recibe el Excel. 3.1 Envía Excel por correo a los estudiantes y profesores de la facultad.	4- Los estudiantes y profesores consultan el Excel.

Flujo alterno de los eventos:	
Acción del actor:	**Respuesta del proceso de negocio:**
Prioridad: Alta	
Mejoras:	
Otras secciones:	

Anexo No. 2 Diagramas de actividades

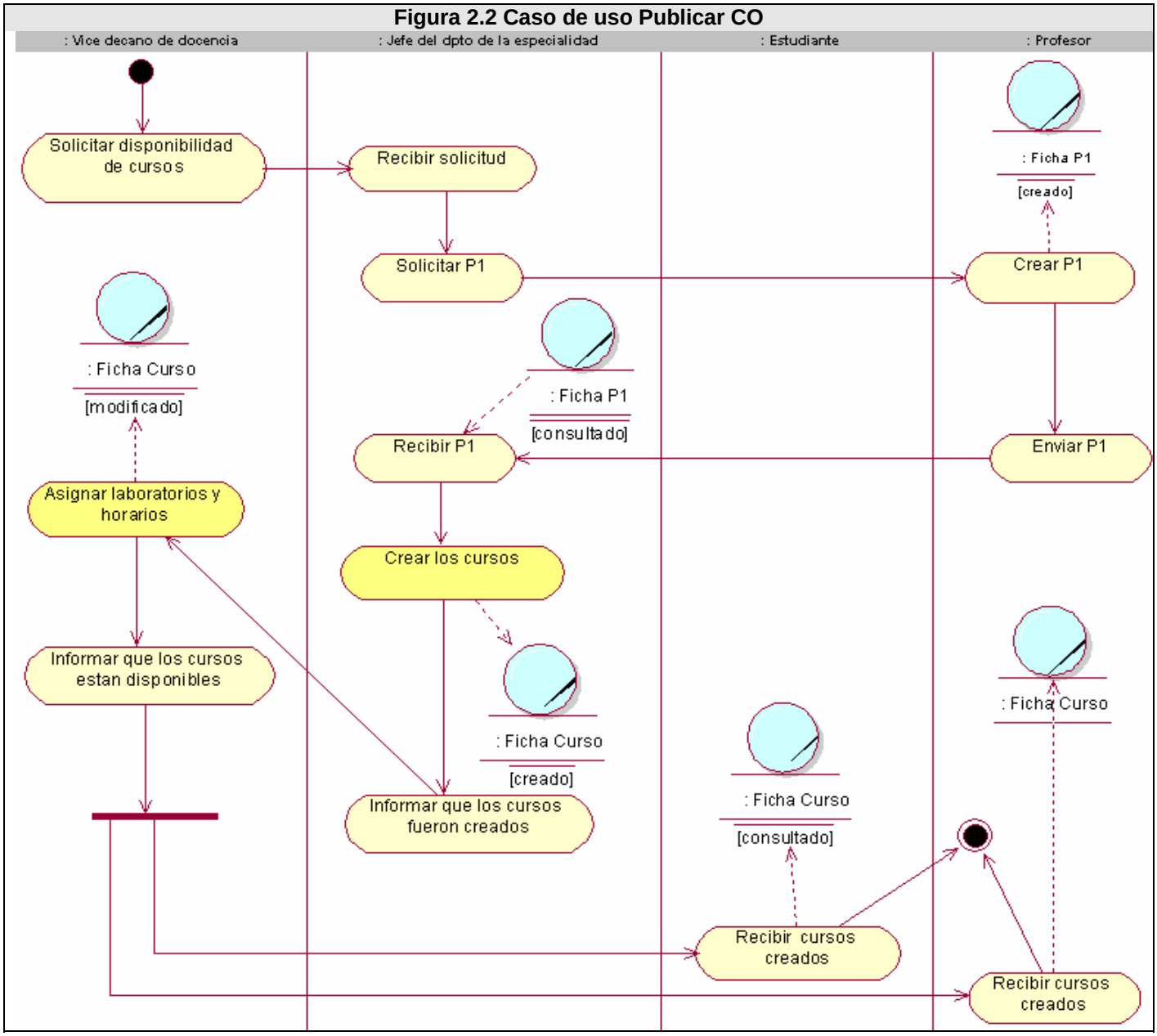
Anexo 2.1

Figura 2.1 Caso de uso Matricular CO.



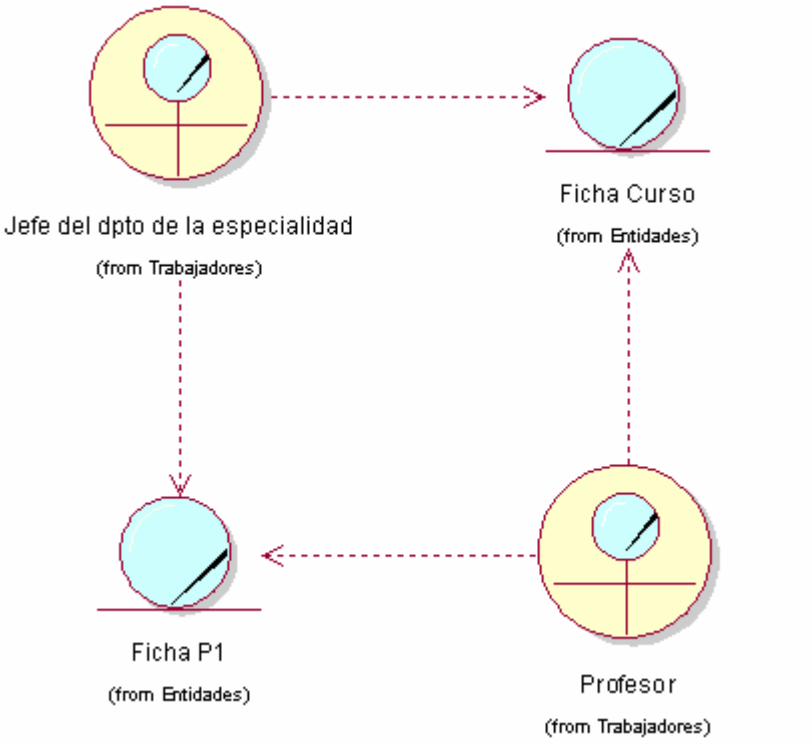
Anexo 2.2

Figura 2.2 Caso de uso Publicar CO



Anexo No. 3

Figura 3.1 Diagrama de clases del modelo de objetos



Anexo No. 4

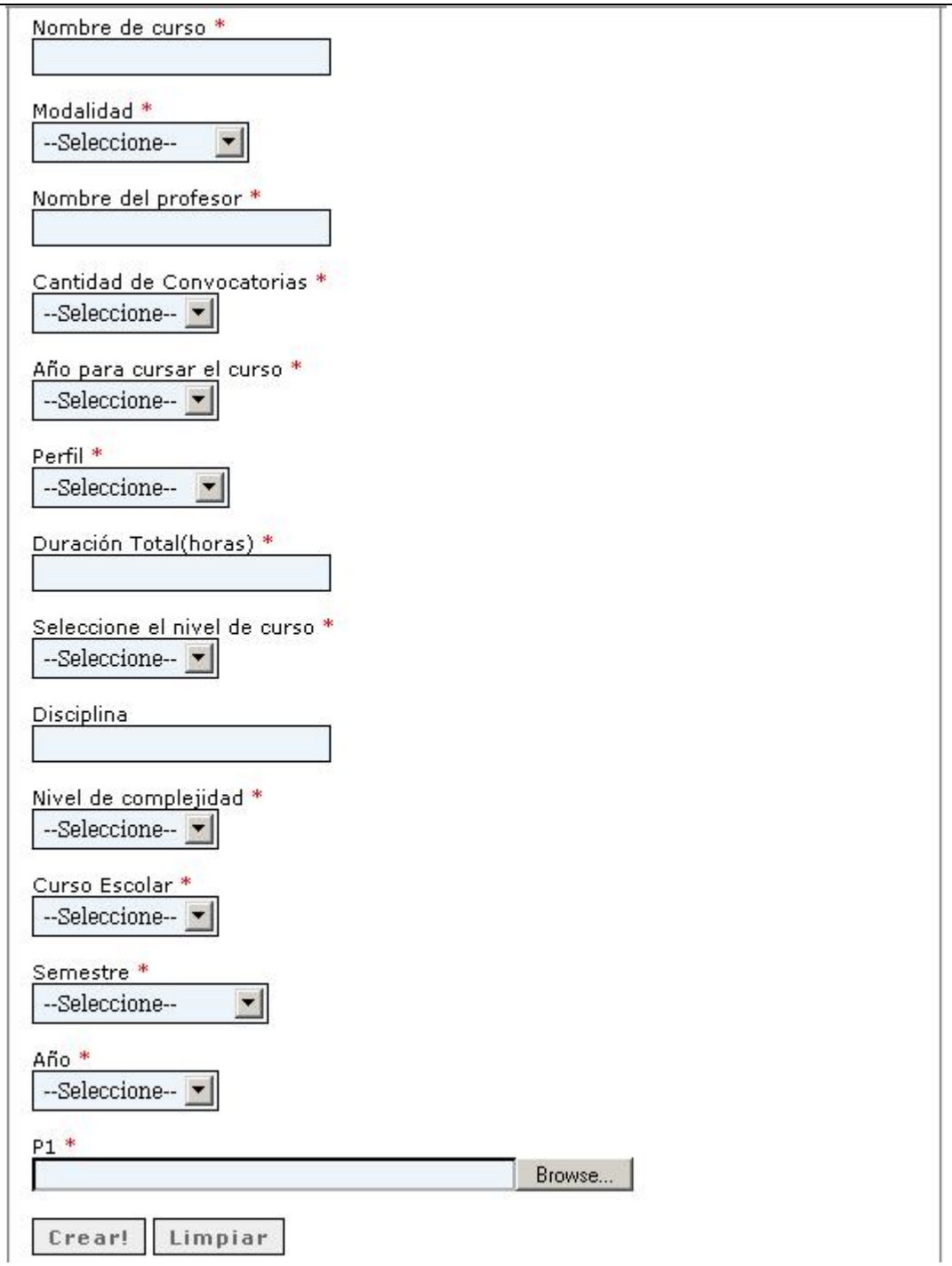
Anexo 4.1 Tabla 4.1

Caso de uso	
CU1	Gestionar CO.
Propósito:	Gestionar los CO lo cual incluye las siguientes acciones, crear, modificar o eliminar.
Actor:	Jefe del departamento de la especialidad

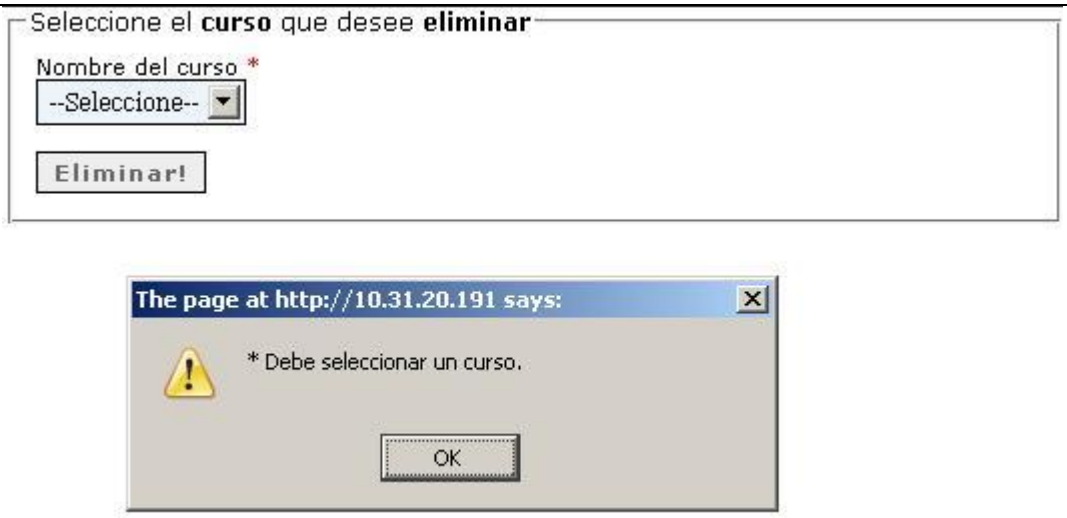
<table>
<tr><td colspan="3">Resumen:
El caso de uso comienza cuando un usuario que tiene el rol de jefe del departamento de la especialidad se autentica en el sistema, entonces el mismo le visualiza las acciones que puede hacer sobre el CO y este selecciona una de ellas, acto seguido el sistema guía al usuario a llevar a efectuar la opción seleccionada y así termina el caso de uso.</td></tr>
<tr><td>Referencias:</td><td colspan="2">R.1, R.1.1, R.1.2, R.1.3</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">1- El usuario debe tener permisos del rol jefe del departamento de la especialidad para acceder a esta parte del sistema</td></tr>
<tr><td>Poscondiciones:</td><td colspan="2"></td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1- El jefe del departamento de la especialidad selecciona la opción gestionar curso optativo.</td><td>2- El sistema le visualiza una interfaz para que seleccione una de las acciones que desea realizar: crear, eliminar o modificar.</td></tr>
<tr><td colspan="3">Escenario 1: Crear un CO.</td></tr>
<tr><td colspan="2">1- El jefe del departamento selecciona la opción de crear un curso.</td><td>2- El sistema le muestra una interfaz que le ayuda a entrar todos los datos necesario para crear un CO.</td></tr>
<tr><td colspan="2">3- El jefe del departamento de la especialidad entra los datos y completa la petición.</td><td>4- El sistema valida los datos entrados por el jefe del departamento.
4.1 Si los datos son correctos el sistema crea el CO y le informa al administrador que ha sido creado el CO.</td></tr>
<tr><td colspan="2">Escenario 2: Eliminar un CO.</td><td></td></tr>
<tr><td colspan="2">1- El jefe del departamento de la especialidad selecciona la opción de eliminar un curso.</td><td>2- El sistema le muestra los CO disponibles.</td></tr>
</table>

3- El jefe del departamento selecciona el CO que desea eliminar.	3- El sistema elimina el CO y le muestra un mensaje al administrador que ha sido eliminado el CO.
Escenario 3: Modificar un curso optativo	
1- El jefe del departamento de la especialidad selecciona la opción modificar curso.	2- El sistema le muestra los CO disponibles.
3- El jefe del departamento selecciona el CO que desea modificar.	4- El sistema le muestra los campos del CO para que modifique los que desee.
5- El jefe del departamento de la especialidad entra los datos y los envía.	6- el sistema actualiza los datos del CO.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Escenario 1: Crear curso optativo	
	4.1 Si los datos son incorrectos el sistema le muestra un mensaje de error, y lo envía a la acción 2 del sistema.
Escenario 2: Eliminar un curso optativo.	
	3- Si el jefe del departamento de la especialidad no selecciona ningún CO el sistema le muestra que debe seleccionar un CO y lo envía a la acción 2.
Escenario 3: Modificar un curso optativo	
	4- Si el jefe del departamento de la especialidad no selecciona ningún CO el sistema le muestra un mensaje que debe seleccionar un CO y lo envía a la acción 2.
Prioridad:	Critico
Puntos de extensión	

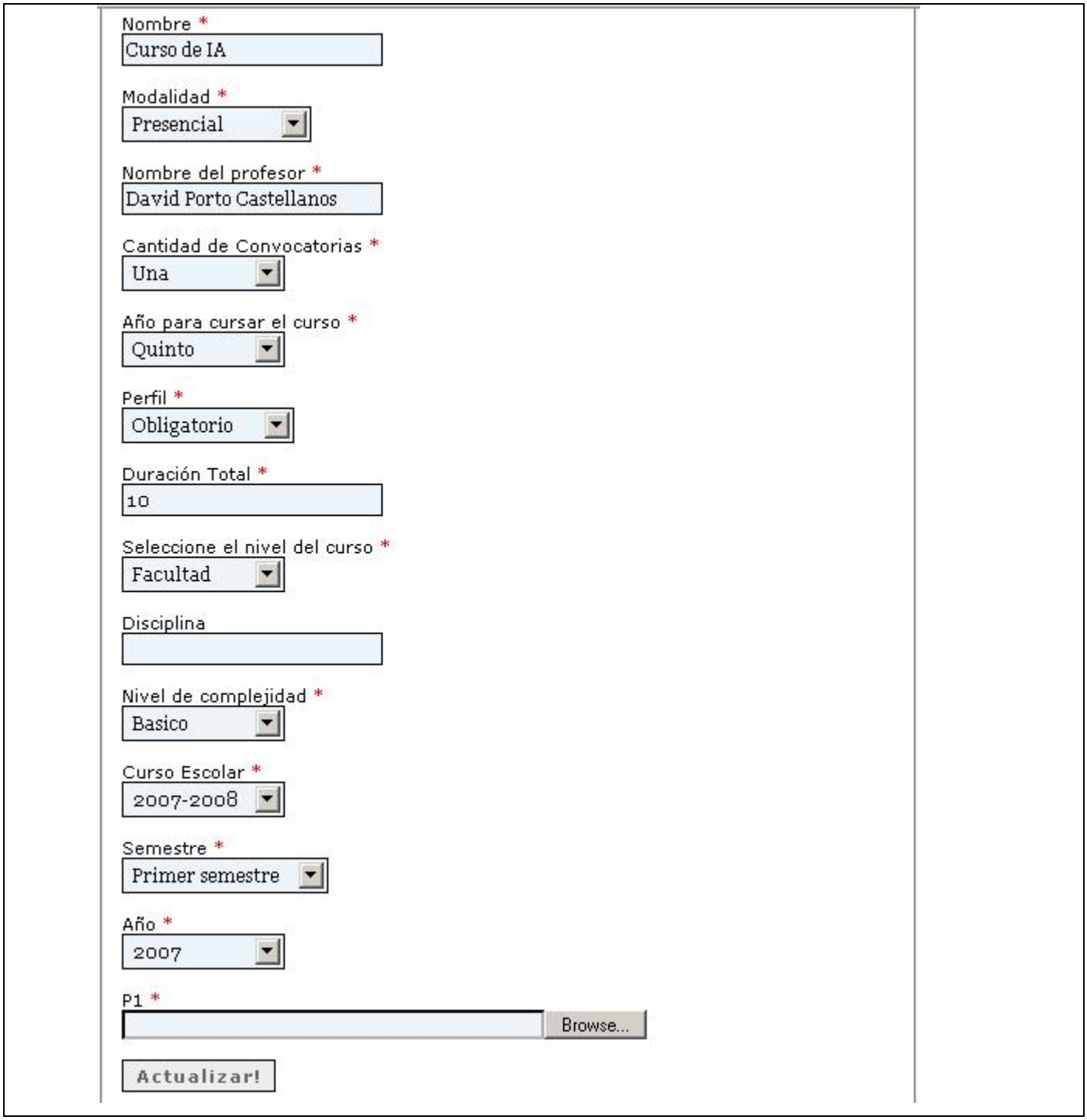
Autentificarse
Prototipo Escenario 1: Crear curso optativo



Prototipo Escenario 2: Eliminar un curso optativo.



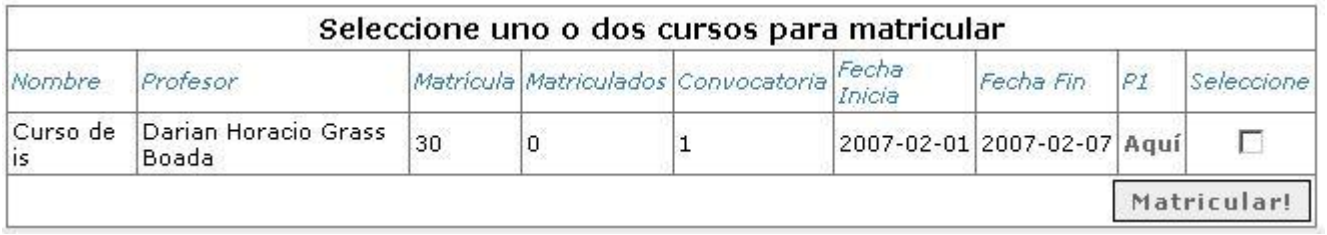
Prototipo Escenario 3: Modificar un curso optativo


Nombre *
Curso de IA
Modalidad *
Presencial
Nombre del profesor *
David Porto Castellanos
Cantidad de Convocatorias *
Una
Año para cursar el curso *
Quinto
Perfil *
Obligatorio
Duración Total *
10
Seleccione el nivel del curso *
Facultad
Disciplina
Nivel de complejidad *
Basico
Curso Escolar *
2007-2008
Semestre *
Primer semestre
Año *
2007
P1 *
Browse...
Actualizar!

Anexo 4.2 Tabla 4.2

Caso de uso		
CU2	Matricular en un CO.	
Propósito:	Que los estudiantes matriculen en al menos dos CO por semestre siempre y cuando cumplan con los requisitos del curso y el mismo no tenga la matricula llena.	
Actor:	Estudiante	
Resumen: El caso de uso comienza cuando el estudiante desea matricular en un CO. Para eso busca los CO que están disponibles, selecciona unos de estos cursos y acto seguido el sistema visualiza si puede matricular o no finalizando así el caso de uso.		
Referencias:	R.2	
Precondiciones:	1- El usuario debe tener permisos del rol estudiante para acceder a esta parte del sistema.	
Poscondiciones:	2-	
Acción del actor		**Respuesta del sistema**
1- El estudiante selecciona la opción matricular en un curso.		2- El sistema visualiza una interfaz donde el estudiante puede observar los CO que puede matricular.
3- El estudiante selecciona el curso que desea matricular y hace la petición.		4- El sistema le muestra todas las características del CO.
5- Si al estudiantes le interesan los contenidos del CO realiza la matricula en el mismo.		
Flujo alternativo		
Acción del actor		**Respuesta del sistema**

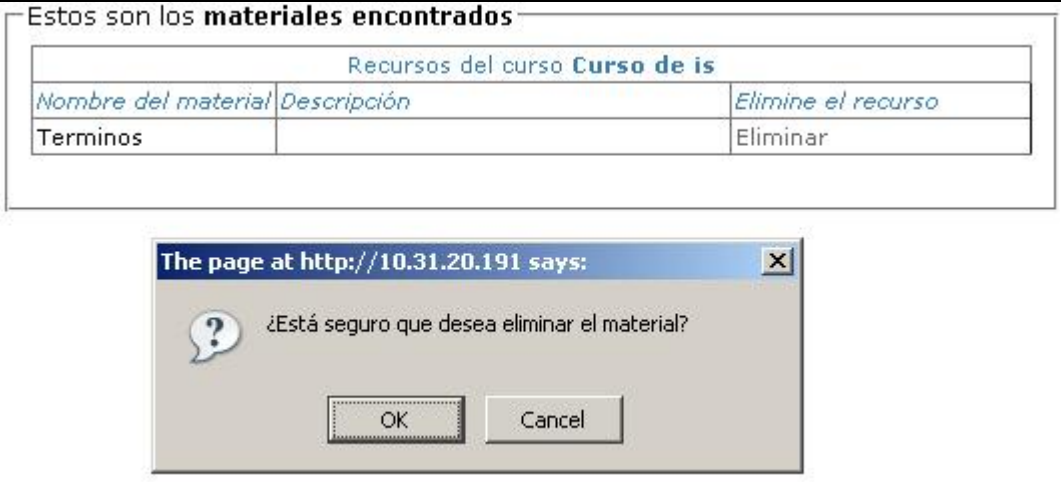
5- Si el estudiante no esta interesado en los contenidos del CO, cancela la matricula y el sistema lo envía a 2.	4- Si el estudiante no selecciona ningún CO el sistema le muestra un mensaje y lo envía a realizar la acción
Prioridad:	Critico
Puntos de extensión	
Autentificarse	
Prototipo	



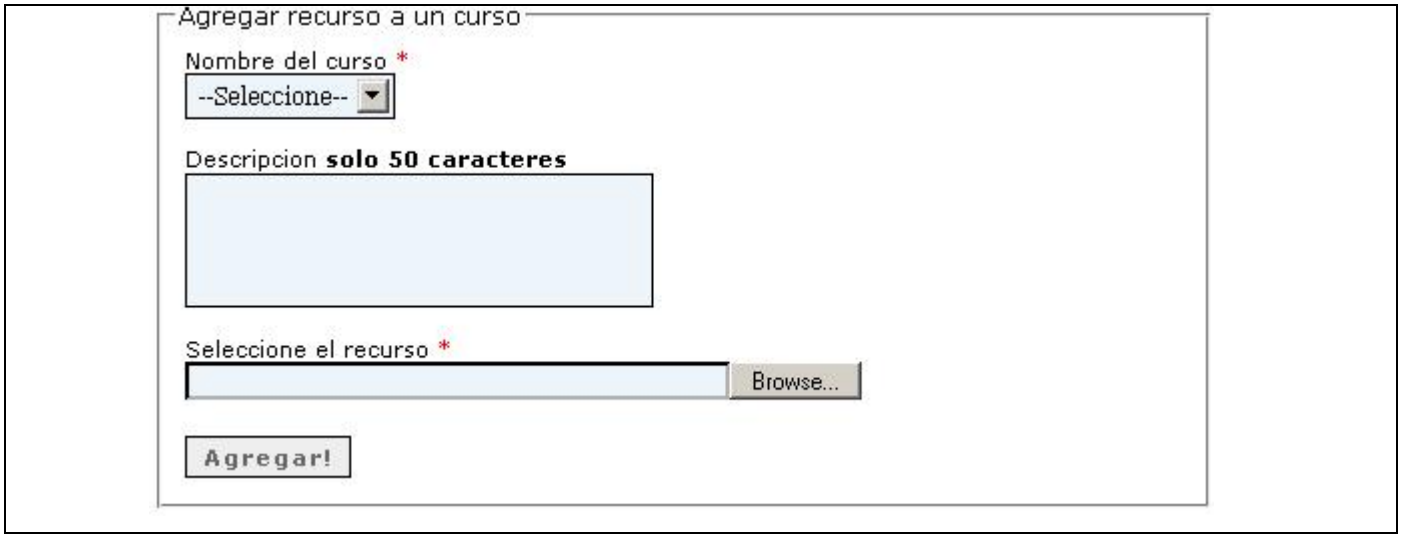
Anexo 4.3 Tabla 4.3

Caso de uso	
CU3	Gestionar materiales a un CO
Propósito	Que los profesor pueda adicionarle documentos con información sobre los CO y los estudiantes consultar dicha información.
Actor:	Profesor
Resumen: El caso de uso comienza cuando el profesor desea publicar algún documento referente al CO. Para ello busca los cursos que el imparte, selecciona el curso al que el desea subir la información selecciona el documento y lo sube al servidor terminando así el caso de uso.	
Referencias	R.3, R.3.1, R.3.2
Precondiciones:	1- El usuario debe tener permisos del rol profesor para acceder a esta parte del sistema.
Poscondiciones:	

Acción del actor	Respuesta del sistema
1- El profesor selecciona la opción eliminar o agregar materiales a un CO.	2- El sistema visualiza una interfaz donde el profesor puede realizar la acción solicitada.
3- El profesor selecciona el curso para agregar o eliminar el documento asociado al mismo.	4- El sistema le visualiza un mensaje informando al profesor que el documento ha sido subido o eliminado con éxito.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	4- El sistema le muestra un mensaje de error y lo envía a realizar la acción 2 si, el profesor no selecciona ningún curso.
Prioridad:	Secundario
Puntos de extensión	
Autentificarse	
Prototipo Eliminar material	



Prototipo Agregar material



Anexo 4.4 Tabla 4.4

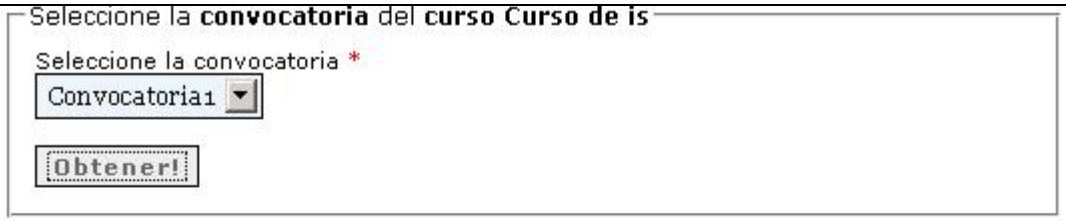
Caso de uso	
CU4	Insertar la nota a un estudiante en el CO.
Propósito:	Que el profesor pueda adicionar la nota que los estudiantes obtuvieron al terminar el CO y eliminarlos en caso de no asistir al mismo.
Actor:	Profesor
Resumen: El caso de uso comienza cuando el profesor desea guardar las notas obtenidas por los estudiantes en el CO. Con este propósito busca los estudiantes que matricularon en el CO. Si el estudiante no se evaluó o no asistió al CO lo elimina y en caso contrario le adiciona nota. Al concluir estas acciones el profesor guardar los datos en el sistema finalizando así el caso de uso.	
Referencias:	R.4
Precondiciones:	1- El usuario debe tener permisos del rol profesor para acceder a esta parte del sistema.
Poscondiciones:	
Acción del actor	**Respuesta del sistema**

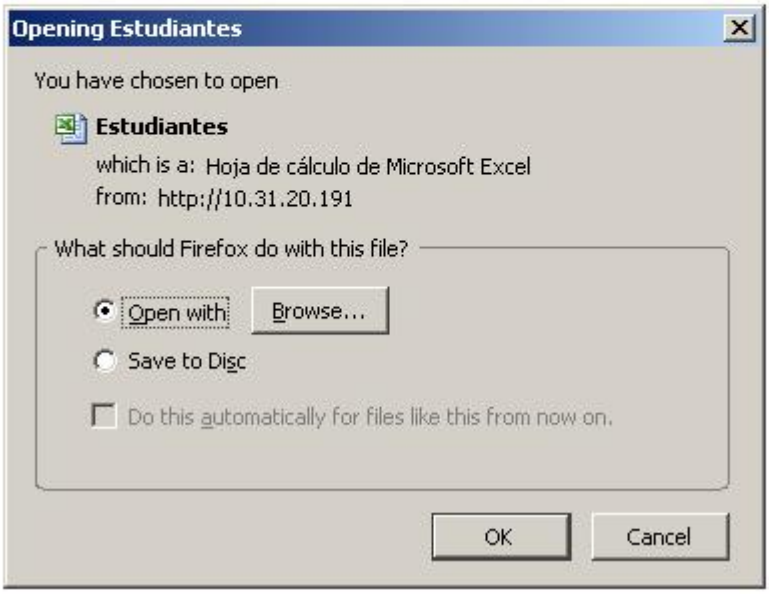
1- El profesor selecciona la opción Insertar la nota a un estudiante en el CO.	2- El sistema visualiza una interfaz donde el profesor puede observar los CO que el imparte.
3- El profesor selecciona el CO al que pertenece el estudiante y hace la petición.	4- El sistema le visualiza todos los estudiantes que pertenecen al curso.
5- El profesor introduce las notas de los estudiantes, marca a los que desea eliminar de listado y hace la petición.	6- El sistema visualiza un mensaje destacando que las notas de los estudiantes fueron adicionadas.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	4- El sistema le muestra un mensaje de error y lo envía a realizar la acción 2 si el profesor no selecciona ningún CO.
Prioridad:	Secundario
Puntos de extensión	
Autentificarse	
Prototipo	

Anexo 4.5 Tabla 4.5

Caso de uso		
CU5	Obtener listado de los estudiantes matriculados en el CO.	
Propósito:	Que los profesor pueda obtener el listado de los estudiantes que matricularon en los CO que el imparte.	
Actor:	Profesor	
Resumen: El caso de uso comienza cuando el profesor necesita empezar a impartir el CO el cual para controlar los estudiantes necesita el listado de los que matricularon. Para ello busca el CO que el imparte para generar un reporte con todos los estudiantes del mismo terminado así el caso de uso.		
Referencias:	R.5	
Precondiciones:	1- El usuario debe tener permisos del rol profesor para acceder a esta parte del sistema.	
Poscondiciones:		
Acción del actor		**Respuesta del sistema**
1- El profesor selecciona la opción Obtener listado de los estudiantes en un curso.		2- El sistema visualiza una interfaz donde el profesor puede selecciona el CO del que quiere generar el listado de los estudiantes.
3- El profesor selecciona el CO y realiza la petición.		4- El sistema le visualiza todos los estudiantes que pertenecen al CO.
5- El profesor selecciona la opción imprimir listado.		6- El sistema envía esta petición a la impresora.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**

	4- El sistema le muestra un mensaje de error y lo envía a realizar la acción 2 si el profesor no selecciona ningún CO.
Prioridad:	Secundario
Puntos de extensión	
Autentificarse	
Prototipo	

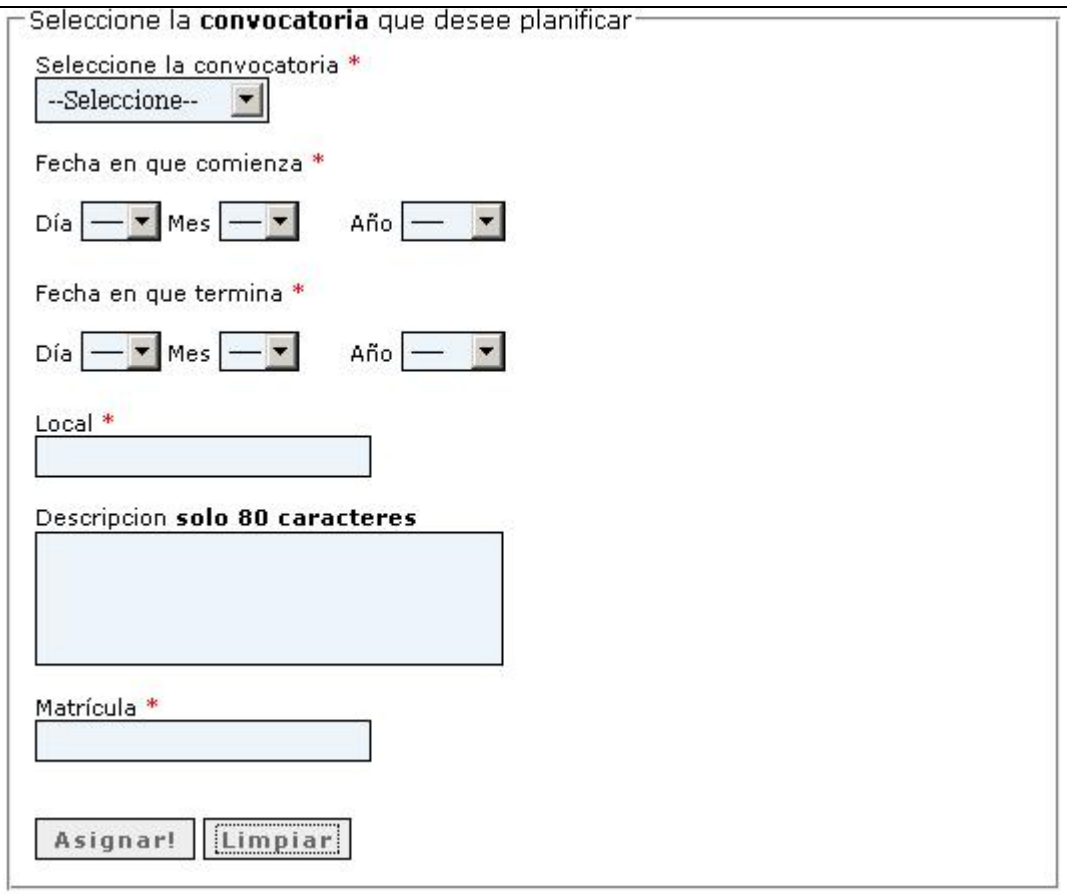




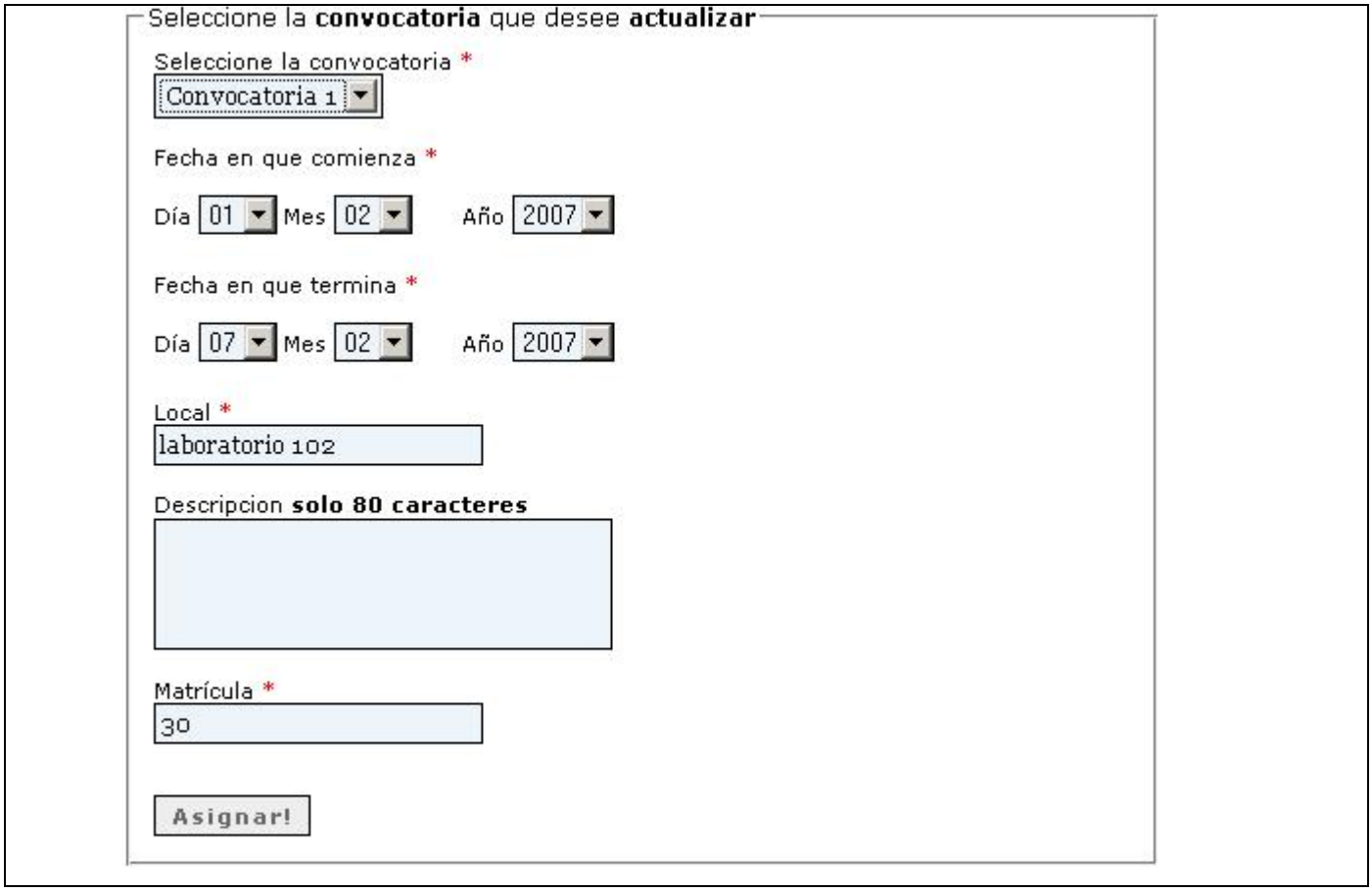
Anexo 4.6 Tabla 4.6

Caso de uso		
CU6	Gestionar local y horario a un CO.	
Propósito:	Que la vice decana de formación le asigne o modifique a los CO los locales donde se impartirán y el horario. Con estos datos el CO ya queda listo para ser activado posteriormente lo cual implica que los estudiantes puedan matricular.	
Actor:	Vice-decano de docencia	
Resumen: El caso de uso comienza cuando la vice-decano de formación necesita que los CO tengan un horario y local para ser impartidos o modificar para el caso que exista algún cambio de estos datos. Para ello busca el CO que desea asignarle el local y el horario terminado.		
Referencias:	R.6, R.6.1, R.6.2	
Precondiciones:	1- El usuario debe tener permisos del rol vice-decano de docencia para acceder a esta parte del sistema.	
Poscondiciones:		
Acción del actor		**Respuesta del sistema**
1- El vice-decano de docencia selecciona la opción Asignar local y horario a un curso.		2- El sistema visualiza una interfaz donde el vice-decano de docencia puede seleccionar los CO que no se le ha asignado el local y los horarios.
3- El vice-decano de docencia selecciona el CO y hace la petición		4- El sistema muestra interfaz donde el vice-decano de docencia puede introducir los datos.
5- El vice-decano de docencia entra los datos.		6- El sistema le muestra un mensaje para que el usuario sepa que los datos fueron guardados con éxito.

Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	4 El sistema le muestra un mensaje de error y lo envía a realizar la acción 2 si el profesor no selecciona ningún CO.
	5 El sistema le muestra un mensaje de error y lo envía a la acción 2 si el usuario no entra los datos correctamente.
Prioridad:	Secundario
Puntos de extensión	
Autentificarse	
Prototipo Asignar local y horario	



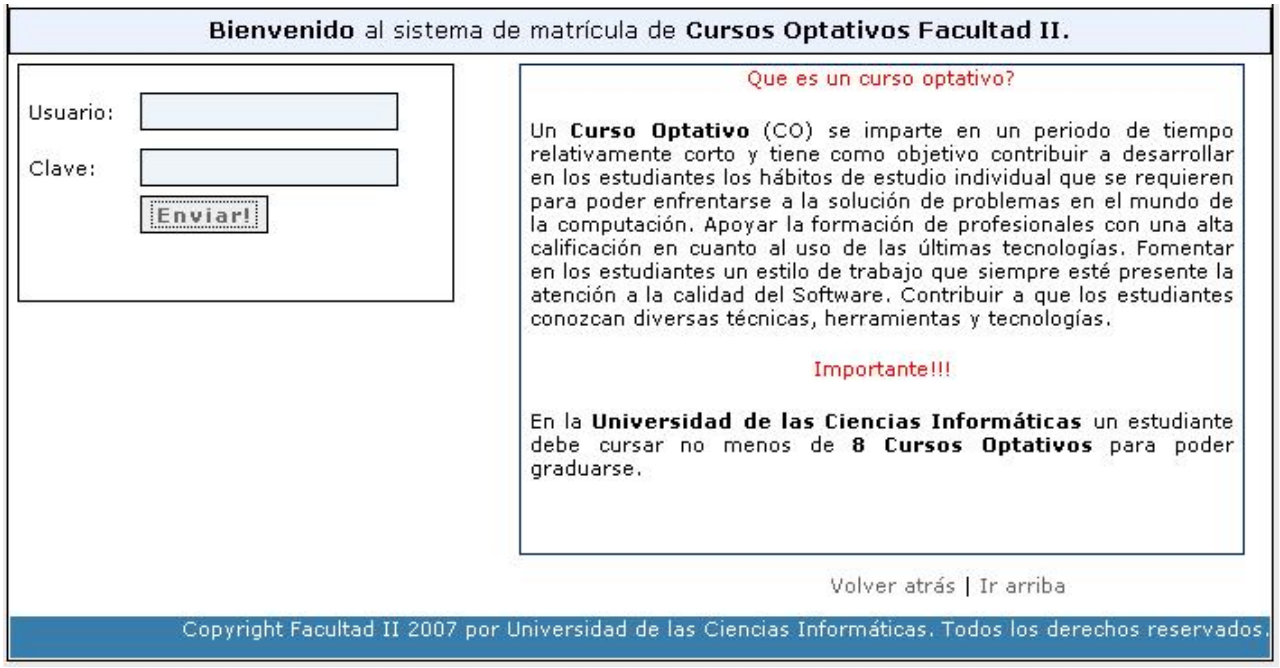
Prototipo Modificar local y horario


Seleccione la convocatoria que desee actualizar
Seleccione la convocatoria *
Convocatoria 1
Fecha en que comienza *
Día 01 Mes 02 Año 2007
Fecha en que termina *
Día 07 Mes 02 Año 2007
Local *
laboratorio 102
Descripcion solo 80 caracteres
Matrícula *
30
Asignar!

Anexo 4.7 Tabla 4.7 Caso

Caso de uso		
CU7	Autenticar usuario.	
Propósito	Que todos los usuarios que trabajen en el sistema estén identificados para así tener un control de sus datos en la aplicación	
Actor:	Usuario	
Resumen: El caso de uso comienza cuando alguna persona de las interesadas en realizar alguna petición a la aplicación desea autenticarse. Para ello simplemente accede a la interfaz de bienvenida y se autentica entrando o no al sistema dependiendo de la autenticidad de sus datos. Terminado así el caso de uso		
Referencias:	R.7	
Precondiciones:		
Poscondiciones:	1- El sistema lleva un control del rol que tiene el usuario que se ha autenticado.	
Acción del actor		**Respuesta del sistema**
1- El usuario acceden a la aplicación.		2- El sistema visualiza la interfaz de bienvenida teniendo en cuenta el tipo de usuario.
3- El usuario se autentica en el sistema.		4- El sistema visualiza la interfaz teniendo en cuenta el tipo de usuario. Y lleva un control de los datos y las acciones del usuario.
5- El usuario realiza las acciones que necesita del sistema.		6- El sistema responde a todas las acciones del usuario.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
		4- El sistema le muestra un mensaje de error y lo envía a realizar la acción 2 si el usuario entra datos erróneos al autenticarse.

Prioridad:	Critico
Puntos de extensión	
Autenticarse	
Prototipo	



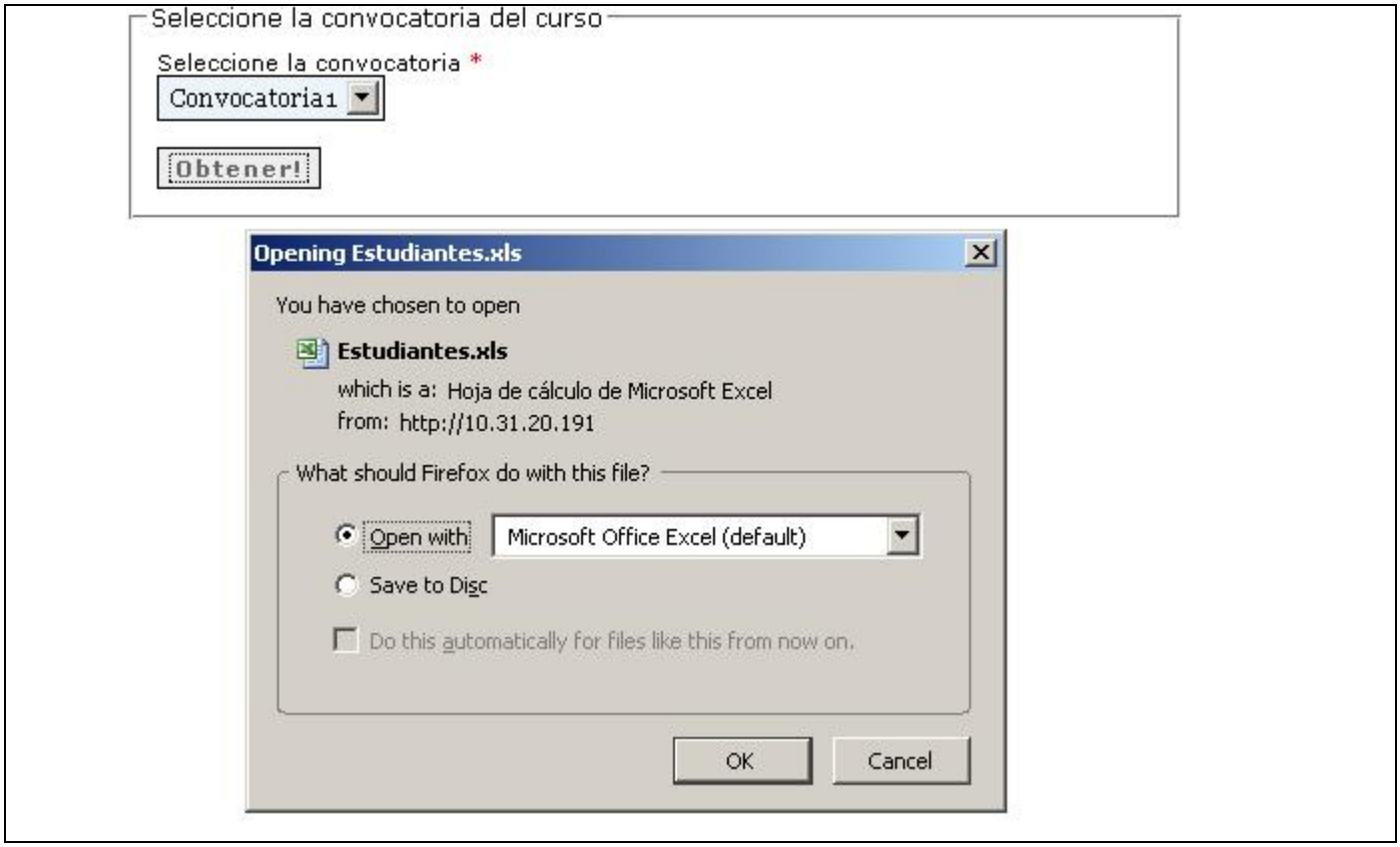
Anexo 4.8 Tabla 4.8

Caso de uso	
CU8	Generar reportes
Propósito:	Que la secretaria de docencia pueda generar los reportes que necesite sobre los CO.
Actor:	Secretaria de docencia

Resumen: El caso de uso comienza cuando la secretaria de docencia desea obtener el listado los estudiantes con las notas que obtuvieron en el CO. Para ello busca el curso del que desea generar el reporte y obtiene la información finalizando así el caso de uso.	
Referencias:	R.8
Precondiciones:	1- El usuario debe tener permisos del rol secretaria de docencia para acceder a esta parte del sistema.
Poscondiciones:	

Acción del actor	**Respuesta del sistema**
1- El secretaria de docencia selecciona la opción Generar reportes.	2- El sistema visualiza una interfaz donde la secretaria de docencia pueda seleccionar el CO del que desea generar el reporte.
3- La secretaria de docencia selecciona el CO de que desea obtener las notas de los estudiantes.	4- El sistema muestra interfaz donde la secretaria de docencia puede observar el reporte desea.
5**-** La secretaria de docencia hace una petición al sistema para que imprima el reporte.	6- El sistema le envía la petición a la impresora.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	4- El sistema le muestra un mensaje de error y lo envía a realizar la acción 2 si la secretaria de docencia no selecciona ningún CO.

Prioridad:	Secundario
Puntos de extensión	
Autentificarse	
Prototipo	



Anexo 4.9 Tabla 4.9

Caso de uso	
CU9	Gestionar usuarios.
Propósito:	Que el administrador gestione los usuarios del sistema esto comprende asignarle permisos del roles vice-decano de docencia, secretaria de docencia y jefe del departamento de la especialidad. También crear, modificar y eliminar estos usuarios.
Actor:	Administrador

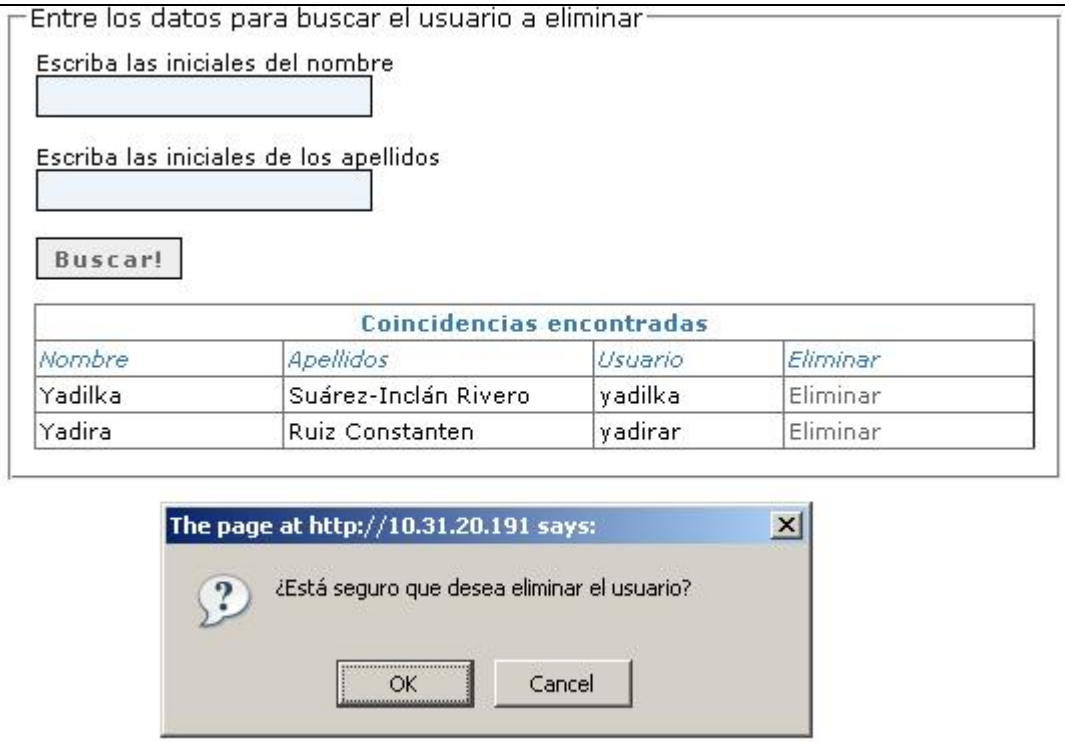
Resumen: El caso de uso comienza cuando un usuario que tiene el rol de administrador se autentica en el sistema, entonces el mismo le visualiza las acciones que puede hacer sobre los usuarios y este selecciona una de ellas, acto seguido el sistema guía al usuario a llevar a cabo la opción seleccionada y así termina el caso de uso.		
Referencias:	R.9, R.9.1, R.9.2, R.9.3	
Precondiciones:	1- El usuario debe tener permisos de los que tiene asignado el rol administrador para acceder a esta parte del sistema	
Poscondiciones:		
Acción del actor		**Respuesta del sistema**
1- El administrador selecciona la opción gestionar usuarios.		2- El sistema le visualiza una interfaz para que seleccione una de las acciones que desea realizar: crear, eliminar o modificar usuario.
Escenario 1: Crear usuario.		
1- El administrador selecciona la opción de crear un usuario.		2- El sistema le muestra una interfaz que le ayuda a entrar todos los datos necesario para crear un usuario.
3- El administrador entra los datos del usuario, le asigna los permisos y envía los datos del usuario.		4- El sistema valida los datos entrados por el administrador. 4.1 Si los datos son correctos el sistema crea el usuario y le muestra un mensaje al administrador para que sepa que la acción ha sido realizada con existo.
Escenario 2: Eliminar usuario.		
1- El administrador selecciona la opción de eliminar un usuario.		2- El sistema le muestra los usuarios del sistema.

3- El administrador selecciona el usuario a eliminar y hace la petición de que sea eliminado.	3- El sistema eliminar el curso optativo y le muestra un mensaje al administrador para que sepa que la acción ha sido realizada con éxito.
Escenario 3: Modificar usuario.	
1- El administrador selecciona la opción modificar usuario.	2- El sistema le muestra los usuarios del sistema.
3- El administrador selecciona el usuario que desea modificar.	4- El sistema le muestra los campos del usuario para que los modifique.
5- El administrador entra los datos y los envía.	6- El sistema actualiza los datos del usuario y le muestra un mensaje al administrador para que sepa que la acción ha sido realizada con éxito.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Escenario 1: Crear usuario.	
	4.1 Si los datos son incorrectos el sistema le muestra un mensaje de error, donde esta el error y lo envía a la acción 2 del sistema
Escenario 2: Eliminar usuario.	
	3- Si el administrador no selecciona ningún usuario el sistema le muestra que debe seleccionar un usuario y lo envía a la acción 2
Escenario 3: Modificar usuario.	
	4- Si el administrador no selecciona ningún usuario el sistema le muestra un mensaje que debe seleccionar un usuario y lo envía a la acción 2
Prioridad:	Secundario
Puntos de extensión	

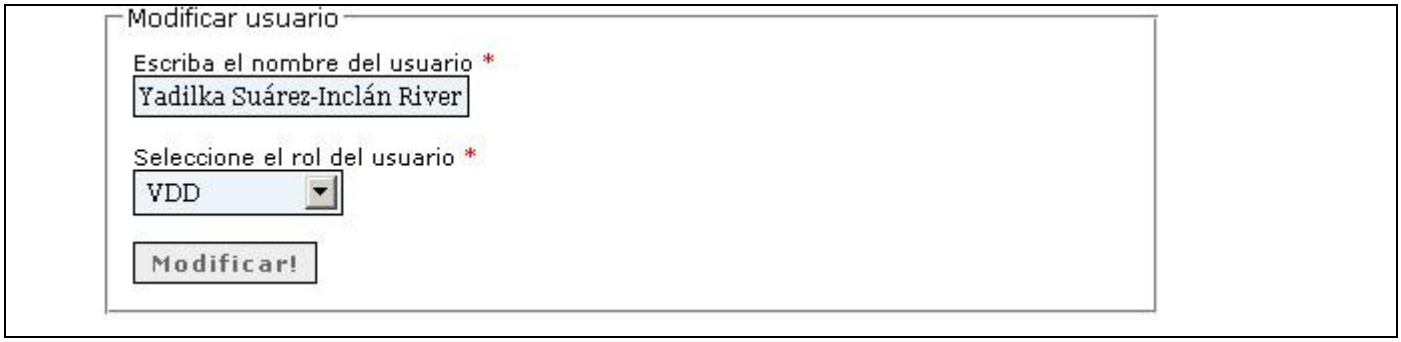
Autentificarse

Prototipo Escenario 1: Crear usuario

Prototipo Escenario 2: Eliminar usuario



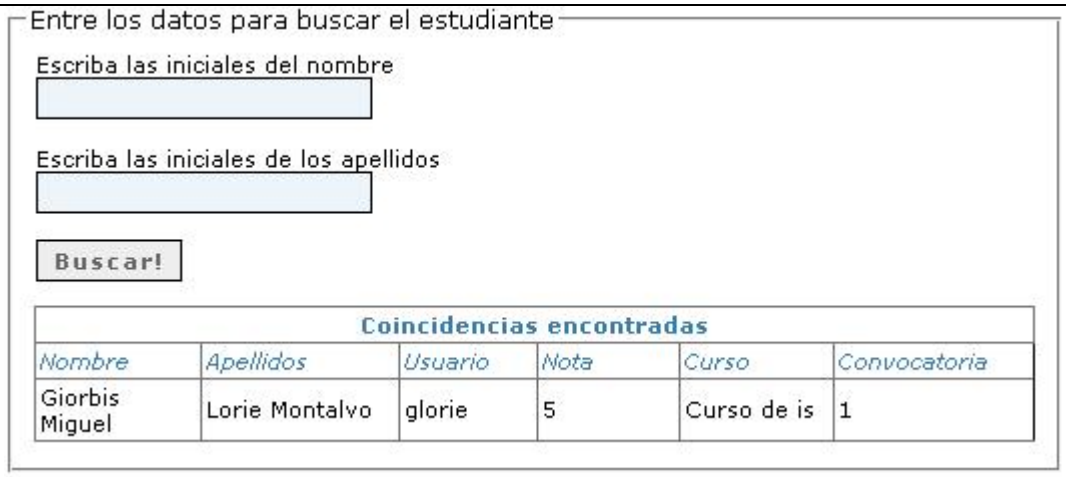
Prototipo Escenario 3: Modificar usuario.



Anexo 4.10 Tablar 4.10

Caso de uso		
CU10	Realizar Búsquedas	
Propósito:	Que la secretaria de docencia pueda obtener información sobre algún estudiante en específico.	
Actor:	Secretaria de docencia	
Resumen: El caso de uso comienza cuando la secretaria de docencia desea obtener alguna información sobre algún estudiante en los CO que recibió. Para ello realiza una búsqueda del mismo, obteniendo los datos deseados. Terminado así el caso de uso.		
Referencias:	R.10	
Precondiciones:	1- El usuario debe tener permisos del rol secretaria de docencia para acceder a esta parte del sistema.	
Poscondiciones:		
Acción del actor		**Respuesta del sistema**
1- El secretaria de docencia selecciona la opción Realizar Búsqueda.		2- El sistema visualiza una interfaz donde la secretaria de docencia pueda entrar los datos por lo que desea realizar la búsqueda.

3- La secretaria de docencia entra los datos que posee del estudiante y hace la petición de búsqueda.	4- El sistema muestra interfaz donde la secretaria de docencia puede observar si existe algún estudiante con esos datos entrados.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	4- El sistema le muestra un mensaje de error y lo envía a realizar la acción 2 si la secretaria de docencia no entra ningún dato para realizar la búsqueda.
Prioridad:	Secundario
Puntos de extensión	
Autentificarse	
Prototipo	



Anexo 4.11 Tabla 4.11

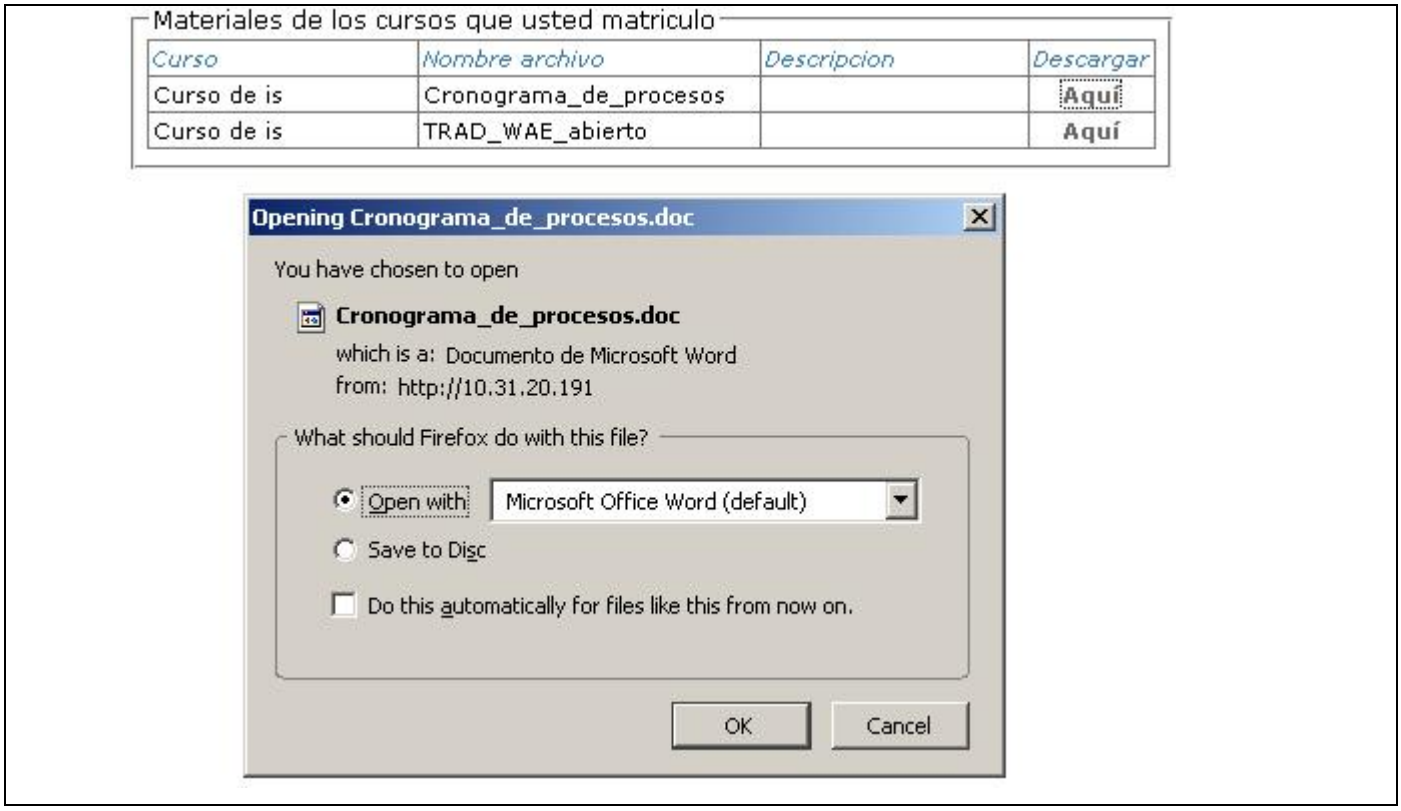
Caso de uso	
CU11	Ver materiales de un CO.
Propósito:	Que los estudiantes puedan consultar la información referente a los CO en que están matriculados.

Actor:	Estudiante
Resumen: El caso de uso comienza cuando el estudiante desea observar algún material del CO. Para realizar esta acción selecciona la opción ver materiales de un CO y el sistema le muestra vínculos con accesos a estos documentos.	
Referencias:	R.11
Precondiciones:	1- El usuario debe tener permisos del rol estudiante y ser miembro de los estudiantes de la facultad II para acceder a esta parte del sistema.
Poscondiciones:	

Acción del actor	**Respuesta del sistema**
1- El estudiante selecciona la opción Ver materiales de un curso.	2- El sistema visualiza una interfaz donde la estudiante puede ver la documentación referente a los cursos que el esta matriculado.
3- El estudiante descarga o puede ver los documentos en el navegador.	

Flujo alternativo	
Acción del actor	**Respuesta del sistema**

Prioridad:	Secundario
Puntos de extensión	
Autentificarse	
Prototipo	



Anexo 4.12 Tabla 4.12

Caso de uso	
CU12	Cancelar matrícula en un CO.
Propósito:	Que los estudiantes puedan cancelar la matricula de en un CO.
Actor:	Estudiante
Resumen: El caso de uso comienza cuando el estudiante desea cancelar la matricula para ello selecciona la opción de cancelar matrícula. Posteriormente el sistema lo guía para que se realicen correctamente los pasos. El estudiante cancela la matricula en el curso y así termina el caso de uso.	
Referencias:	R.12

Precondiciones:	1- El usuario debe tener permisos del rol estudiante y ser miembro de los estudiantes de la facultad II para acceder a esta parte del sistema.
Poscondiciones:	

Acción del actor	**Respuesta del sistema**
1- El estudiante selecciona la opción Cancelar matricula.	2- El sistema visualiza una interfaz donde la estudiante puede ver los cursos en que el esta matriculado para que cancele la matrícula.
3- El estudiante selecciona el CO del cual desea cancelar la matricula.	4- El sistema le muestra un mensaje informándole al estudiante que la matrícula en el curso seleccionado ha sido cancelado.

Flujo alternativo

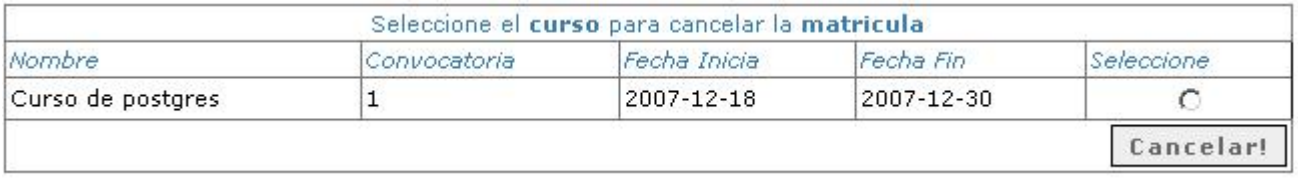
Acción del actor	**Respuesta del sistema**
	2- Si el estudiante no selecciona un CO el sistema le muestra un mensaje informándole que debe seleccionar un CO para cancelar matrícula.

Prioridad:	Secundario

Puntos de extensión

Autentificarse

Prototipo



Anexo No. 5 Diagramas de colaboración.

Anexo 5.1 Caso de uso Gestionar cursos optativos.

Figura 5.1 Escenario Crear un curso optativo

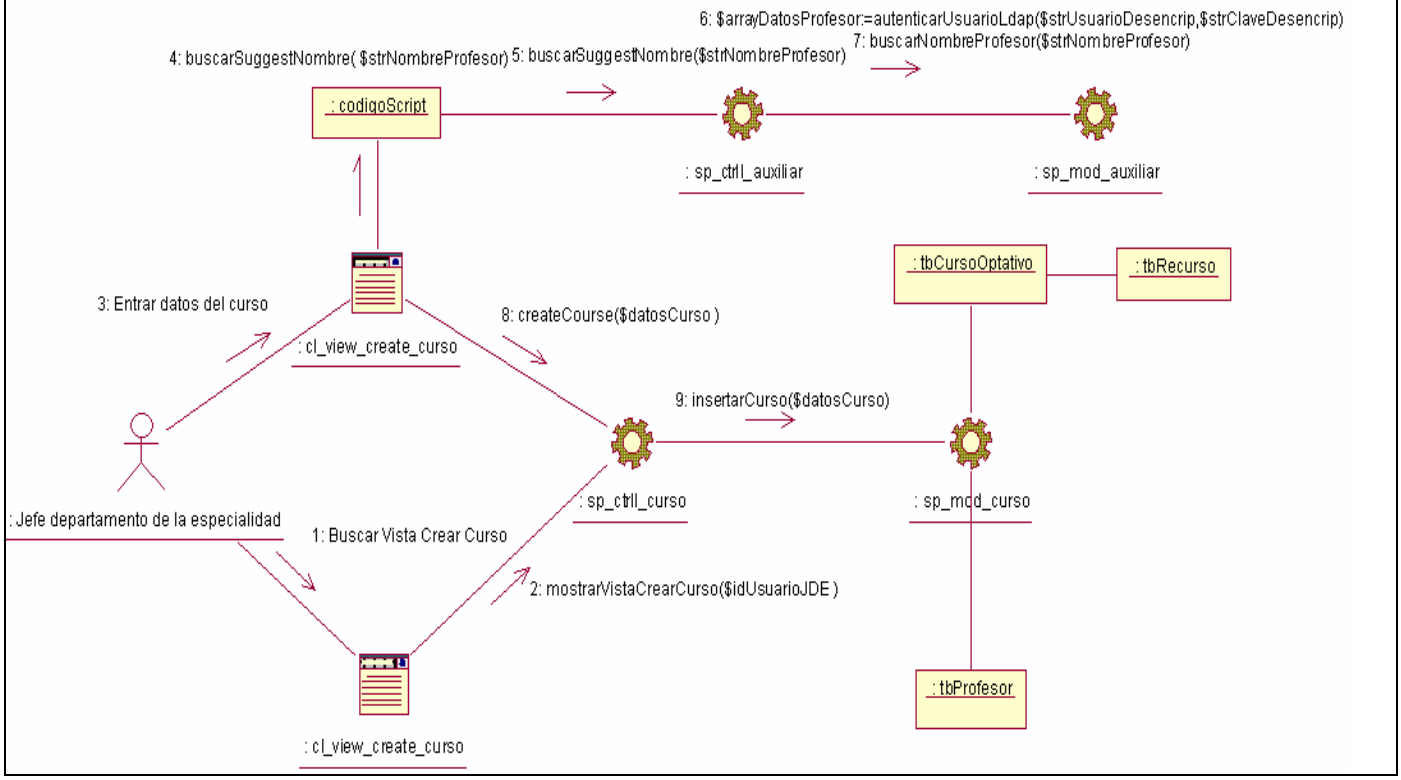


Figura 5.2 Escenario Eliminar un curso optativo

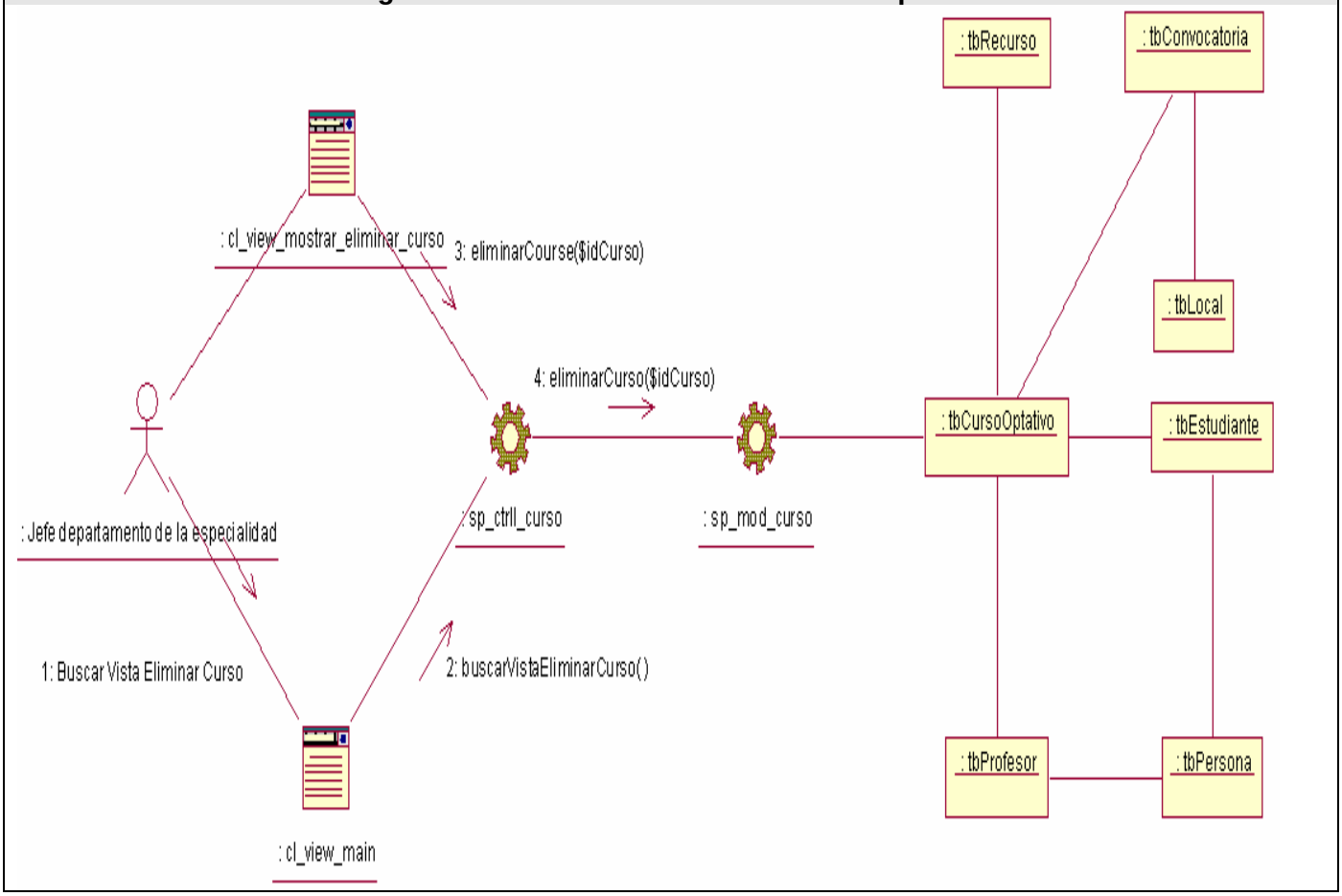
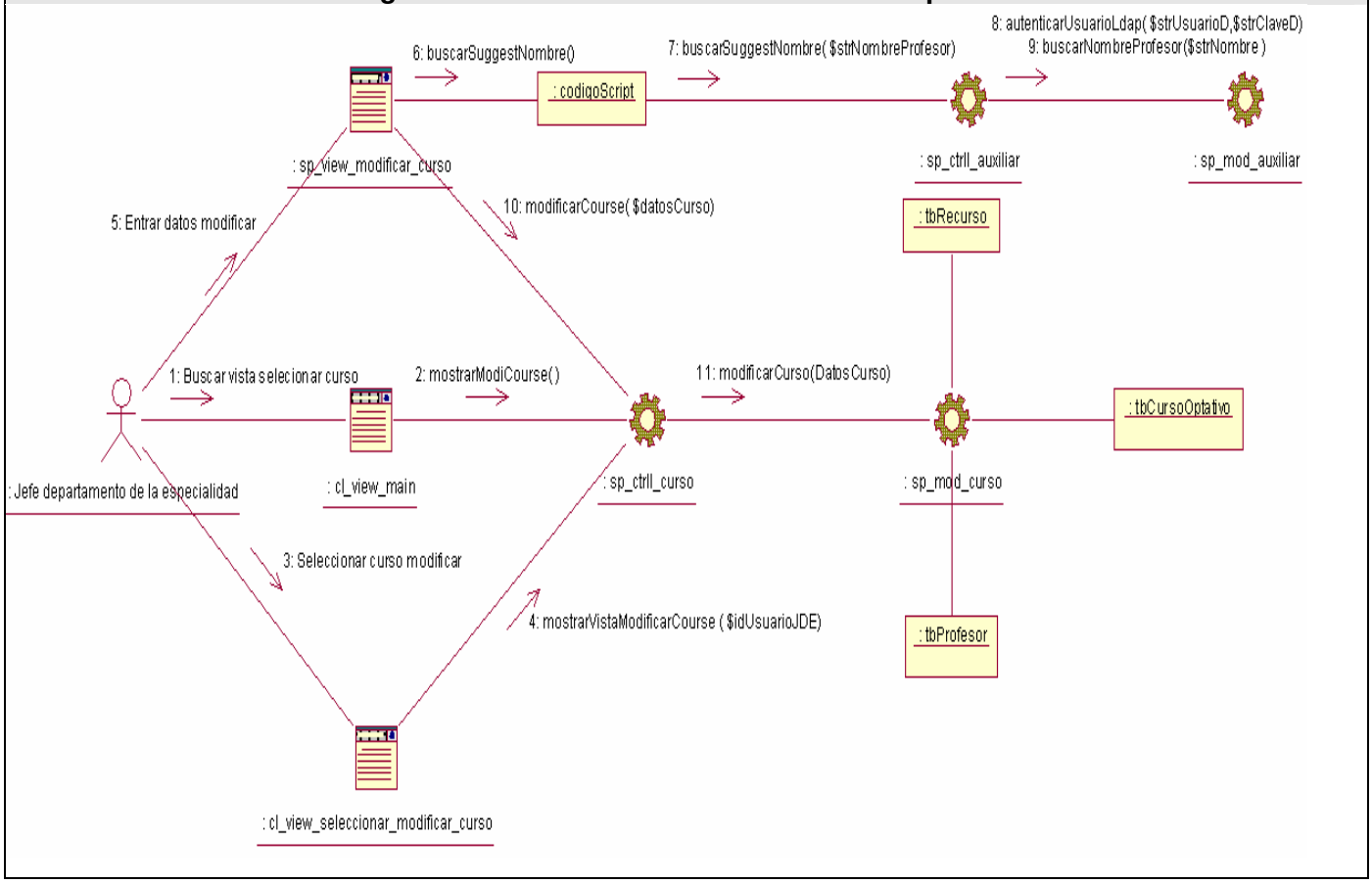
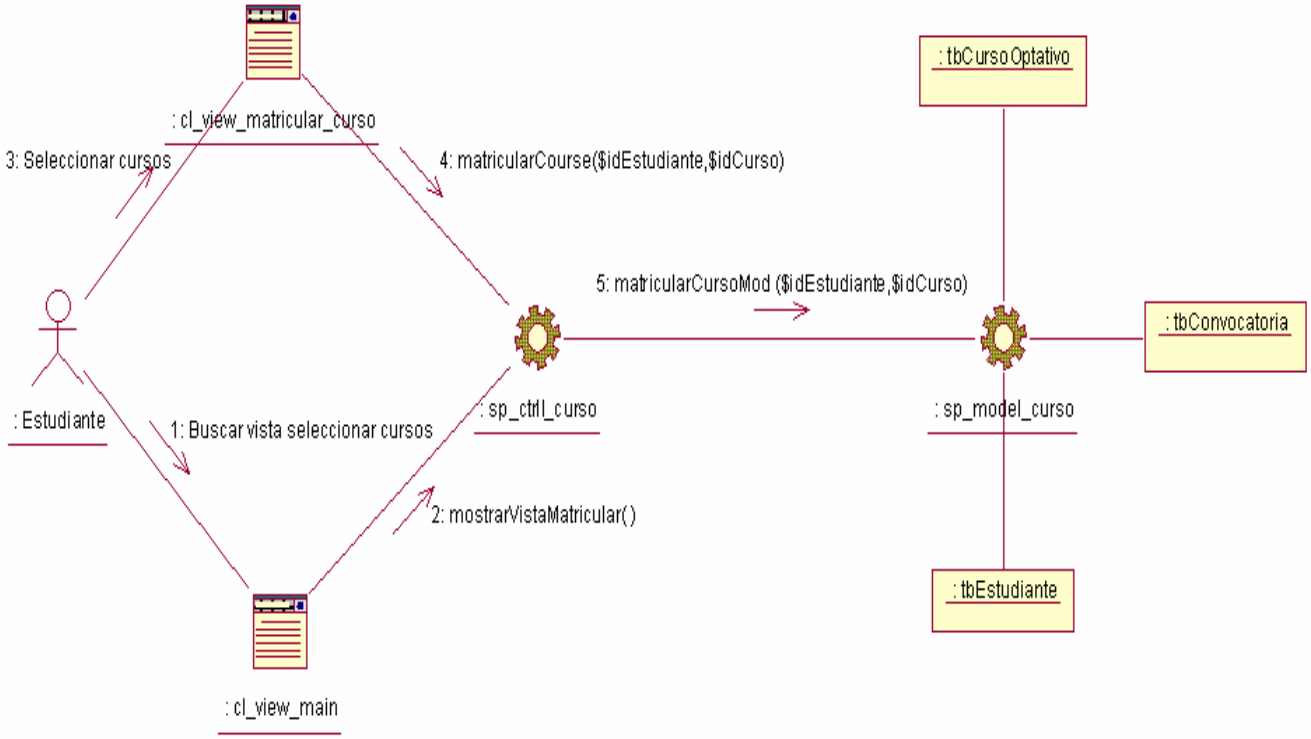


Figura 5.3 Escenario Modificar un curso optativo



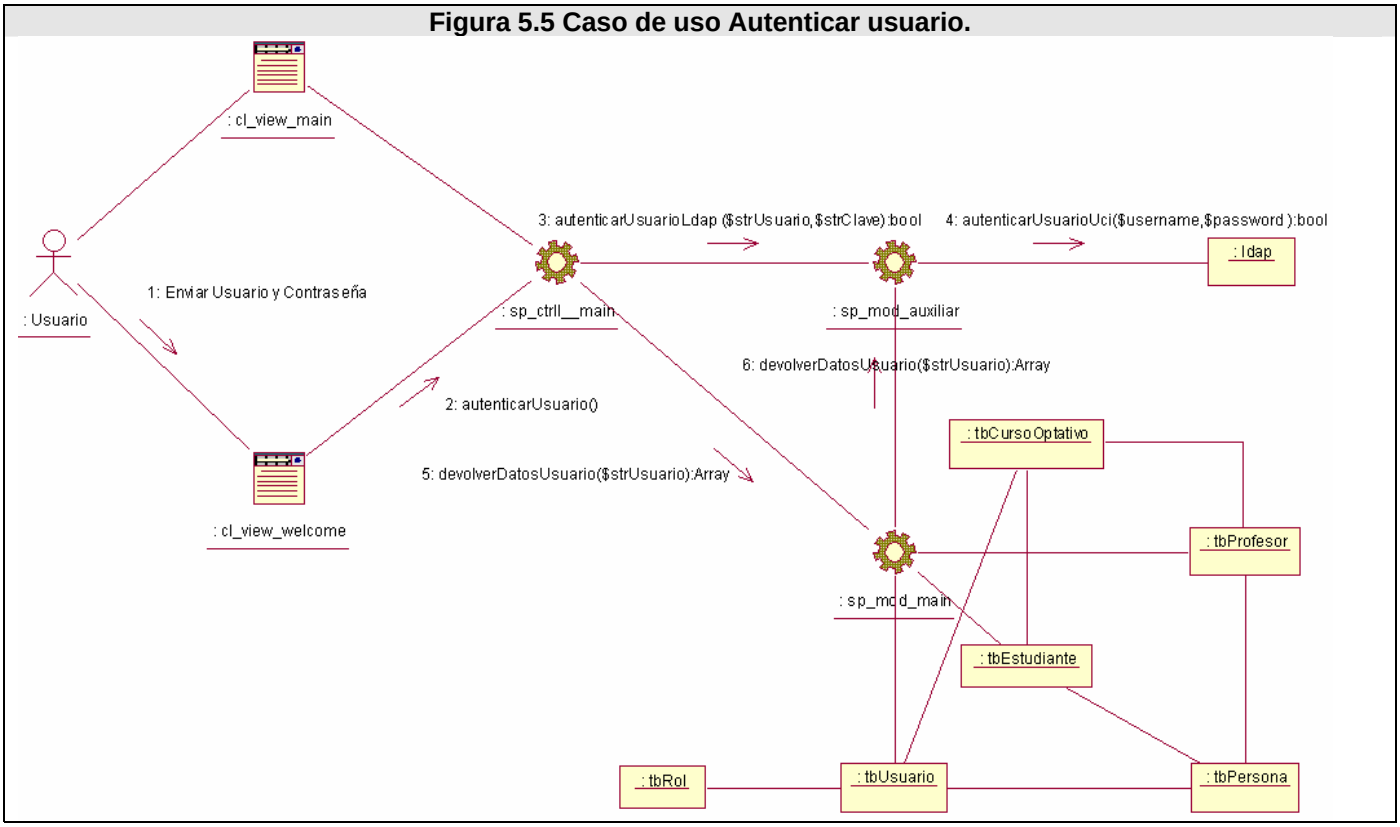
Anexo 5.2

Figura 5.4 Caso de uso Matricular en un curso.



Anexo 5.3

Figura 5.5 Caso de uso Autenticar usuario.



GLOSARIO DE TERMINOS

• **Clase:** Conjunto de objetos que comparten atributos, operaciones, relaciones y semántica.

• **Colaboración:** Define la interacción entre los elementos que proporcionan un comportamiento cooperativo mayor que la suma de los comportamientos de sus elementos.

• **Caso de uso:** Conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor.

• **Componente:** Es una parte física y reemplazable de un sistema que conforman un conjunto de interfaces y proporciona la implementación de dicho conjunto.

• **Interacción:** Conjunto de mensajes intercambiados entre un conjunto de objetos para alcanzar un propósito específico.

• **Dependencia:** relación semántica que indica que un cambio en un elemento afecta a la semántica de otro elemento.

• **Asociación:** Relación estructural que describe las conexiones entre objetos.

• **Generalización/Especialización:** Relación en la que el hijo comparte la estructura y el comportamiento del padre.

• **Diagramas:** Es la representación gráfica de un conjunto de elementos. Visualizan un sistema desde diferentes perspectivas.

• **Diagrama de clases:** Conjunto de clases, interfaces y colaboraciones; así como sus colaboraciones.

• **Diagrama de objetos**: Conjunto de objetos y sus relaciones.

• **Diagrama de casos de uso:** Conjunto de casos de uso y actores y sus relaciones.

• **Diagramas de interacción (secuencia y colaboración):** Objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.

• **Diagrama de actividad:** Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

• **Diagrama de componentes:** Organización y las dependencias entre un conjunto de componentes.

• **Diagrama de despliegue:** Configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

• **Framework:** Un el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

• **Lenguajes de Programación:** Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático.

•**Apache:** es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1.

•**Ajax**: Asynchronous JavaScript And XML.

•**Html:** *HyperText Markup Language.* Lenguaje usado para escribir documentos para servidores World Wide Web. Es una aplicación de la ISO Standard 8879:1986. Es un lenguaje de marcas. Los lenguajes de marcas no son equivalentes a los lenguajes de programación aunque se definan igualmente como "lenguajes". Son sistemas complejos de descripción de información, normalmente documentos, que se pueden controlar desde cualquier editor ASCII.

•**Linux:** Es el nombre de un núcleo, pero se suele denominar con este nombre a un sistema operativo de libre distribución software libre (y de código abierto), donde el código fuente está disponible públicamente y cualquier persona, con los conocimientos informáticos adecuados, puede libremente estudiarlo, usarlo, modificarlo y redistribuirlo.

•**MySQL:** Es un sistema de gestión de bases de datos relacional que cuentan con todas las características de un motor de BD comercial: transacciones atómicas, triggers, replicación, llaves foráneas entre otras. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar.

•**PHP:** *Hypertext Preprocessor*. Es un ambiente script del lado del servidor que permite crear y ejecutar aplicaciones Web dinámicas e interactivas. Con PHP se pueden combinar páginas HTML y script*s*. Con el objetivo de crear aplicaciones potentes.

•**RUP:** *Rational Unified Process* (Proceso Unificado de desarrollo). Metodología para el desarrollo de Software.

•**Software:** Programas de sistema, utilerías o aplicaciones expresados en un lenguaje de máquina.

•**UML:** *Unified Modeling Language*. Es una notación estándar para modelar objetos del mundo real como primer paso en el desarrollo de programas orientados a objetos. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software.

•**XML:** *Extensible Markup Language*. Es un lenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium. Orientado principalmente al almacenamiento, procesamiento y transmisión de mensajes.

Navegador Web, ojeador o browser es una aplicación software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores Web de todo el mundo a través de Internet. Esta red de documentos es denominada World Wide Web (WWW) o Telaraña Mundial. Los navegadores actuales permiten mostrar o ejecutar: gráficos, secuencias de vídeo, sonido, animaciones y programas diversos además del texto y los hipervínculos o enlaces.

Servidor Web: Un servidor Web se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente HTTP que solemos conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita. A modo de ejemplo, al teclear www.wikipedia.org en nuestro navegador, éste realiza una petición HTTP al servidor de dicha dirección. El servidor responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo muestra en pantalla. Como vemos con este ejemplo, el cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Web Service: Es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

MVC: El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Maneja la visualización de la información.

Controlador: Controla el flujo entre la vista y el modelo (los datos). Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo, independientemente de la representación visual.

Patrones: Son una disciplina de resolución de problemas reciente en la ingeniería del software que ha emergido en mayor medida de la comunidad de orientación a objetos, aunque pueden ser aplicados en cualquier ámbito de la informática y las ciencias en general.