Universidad de las Ciencias Informáticas
Faculta V

Título: Sistema de Gestión de Información del DDC-Ingeniería y Gestión de Software UCI

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

Autor(es): Yailín Estrada Herrera
Lenna Carballo Muñoz

Tutor(es): MSc. Yamilis Fernádez Pérez

Ciudad de la Habana, 23 de Junio 2007

DECLARACIÓN DE AUTORÍA

Declaro que somos las únicas autoras de este trabajo y autorizo al Departamento Docente Central de Ingeniería y Gestión de Software de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autor:
Yailín Estrada Herrera
Lenna Carballo Muñoz

Tutor:
MSc.Yamilis Fernández Pérez

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado Sistema de Gestión de Información del DDC-Ingeniería y Gestión de Software UCI, fue realizado en nuestra entidad, Departamento Docente Central de Ingeniería y Gestión de Software. 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:

_____

_____

_____

_____

_____

_____

Como resultado de la implantación de este trabajo se reporta un efecto económico que asciende a MN y/o CUC.

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año 2006.

Representante de la entidad	Cargo
Firma	Cuño

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Sistema de Gestión de Información del DDC-Ingeniería y Gestión de Software UCI

Autores: Yailín Estrada Herrera y Lenna Carballo Muñoz

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

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

....

Tutor Fecha

AGRADECIMIENTOS

A la Revolución a Fidel y a la Universidad de las Ciencias Informáticas, por habernos acogido en su seno y formar parte de este proyecto futuro, surgido a la raíz de la batalla de ideas.
A nuestra tutora Yamilis por poner todo su esfuerzo en este trabajo, atendernos en cualquier momento, estando siempre atenta por lo que nos haga falta y revisándonos constantemente esta tesis.
A Rafael por ayudarnos muchísimo, en todo momento, y ser tan preocupado.
A Arniel por habernos ayudado en todo momento sin reparos y sin condiciones.
A Febe por estar presente siempre que necesitábamos ayuda.
A todas nuestras amistades y conocidos que siempre nos brindaron su apoyo en los momentos difíciles, y nos tendieron la mano cuando necesitamos ayuda, a Yirka, Yoander, Irina, Lisbe, Diana, Joel, Hissel en fin a todos.

A mis abuelos Pascual y Aurelia y a mamá, por quererme tanto y haberme aconsejado y guiado en este momento, por darme todo su cariño y estar presente cada vez que los necesitaba, gracias a ellos me graduaré hoy. A mi novio Rafael por darme todo su amor y por estar siempre a mi lado apoyándome y ayudándome en todo momento. A mi hermanito por quererme tanto, a Ricardo por estar siempre en disposición para cualquier cosa que necesitase. A mis primos y mi tía Iliana, a mi prima Leity por darme consejos y estar presente en todo momento. En fin a todos aquellos que de una forma u otra colaboraron con la realización de mi tesis y que me apoyaron para llegar ahora donde estoy.

Lenna

A mis padres Rafael y Teresa, especialmente a mi mamá que fue la que estuvo a mi lado en todo momento apoyándome, aconsejándome y esforzándose porque yo saliera adelante siempre muy preocupada por mi. A mi abuelita Lina que me aconsejó muchísimo y que me ayudó también para que yo estuviera aquí. A mi hermana Yani que tanto me apoya y me quiere. A Ignacio que de una forma u otra me ayuda muchísimo. A mi novio Santiago que me da muchas fuerzas para seguir adelante con su infinito amor y cariño, al igual que su mamá Sandra que también me quiere mucho. A mi prima Yadirka que me pidió que estudiara mucho. En fin a todos mis tíos, parientes y conocidos que siempre estuvieron al tanto de mi, muchísimas gracias por todo y por lograr que yo esté aquí en estos momentos.

Yailin

DEDICATORIA

Especialmente a mi abuelo, que aunque ya no se encuentra entre nosotros pero siempre deseo que yo me graduara y tuviera mi titulo.
A mi abuela, a mi mamá, a mi padrastro y a mi hermanito para darles un motivo más para que se sientan orgullosos de mí.
A mi Rafa por traerme la felicidad y el amor a mí vida.

Lenna

Se lo dedico especialmente a mi mamá y a mi abuela que sus mayores deseos siempre han sido que me gradúe y tenga mi título.
A mi hermana, mi papá, mi padrastro para que mantengan siempre el orgullo que sienten por mi y a mi suegra y mi cuñado también.
A mi novio que tanto amo, que siempre me apoya en todo, con gran amor y cariño.

Yailín

RESUMEN

La Universidad de las Ciencias Informáticas (UCI) se plantea elevar la calidad del proceso de formación del profesional. El principio fundamental en este proceso es la formación desde la producción, para ello es decisivo el trabajo metodológico del claustro, de allí la vital importancia que representan la Dirección de Formación y los Departamentos Docentes Centrales. En estos Departamentos se realizan una serie de tareas, que conllevan a que exista un gran flujo de información que debe ser de un seguro y rápido acceso. Actualmente muchas de estas tareas se llevan a cabo de forma manual con ayuda de máquinas computadoras, pero que no cumplen ciertos requisitos de seguridad y control, produciendo pérdidas de tiempo y recursos.

En este trabajo se desarrolla todo el ciclo de vida del software, comenzando con la identificación de los principales procesos del Departamento Docente Central (DDC) de Ingeniería y Gestión de Software, que son objeto a automatizar, y finalizando con la implementación de un sistema informático que gestione dichos procesos. Para esto se utiliza como metodología de desarrollo de software a RUP, y una serie de tecnologías libres que propician una buena calidad y eficiencia al sistema.

ÍNDICE

INTRODUCCIÓN **1**

1 CAPÍTULO1: FUNDAMENTACIÓN TEÓRICA **4**

1.1 INTRODUCCIÓN 4
1.2 OBJETO DE ESTUDIO 4
1.3 SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS AL CAMPO DE ACCIÓN 4
1.4 PROCESOS ELEMENTALES DEL NEGOCIO 5
1.5 TENDENCIAS Y TECNOLOGÍAS ACTUALES 6
1.5.1 Política de migración a software libre *6*
1.5.2 Hipertexto Preprocesor (PHP) *7*
1.5.3 Plataforma de desarrollo Web *8*
1.5.4 NuSphere PHpED *8*
1.5.5 Sistema gestor de bases de datos PostgreSQL *9*
1.5.5.1 EMS PostgreSQL Manager 10
1.5.6 ERwin *11*
1.5.7 Extensible Markup Language (XML) *11*
1.5.8 Hojas de Estilo en Cascada (Cascade Style Sheets, CSS). *12*
1.5.9 UML (Lenguaje Unificado de Modelado) *12*
1.5.10 Otras herramientas utilizadas *13*
1.5.11 El Proceso Unificado de Desarrollo de Software (RUP) *13*
1.6 SELECCIÓN DE LAS TECNOLOGÍAS A UTILIZAR EN LA PROPUESTA SOLUCIÓN 15
1.7 CONCLUSIONES 15

2 CAPÍTULO2: REQUERIMIENTOS **16**

2.1 INTRODUCCIÓN 16
2.2 ACTORES DEL SISTEMA 16
2.2.1 Vista Global de Actores *18*
2.3 REQUERIMIENTOS FUNCIONALES REFINADOS 18
2.4 DEFINICIÓN DE LOS REQUERIMIENTOS NO FUNCIONALES 23
2.5 DIAGRAMA DE CASO DE USO 25
2.5.1 Diagrama de Paquetes *25*
2.5.1.1 Paquete: Módulo Docente-Metodológico 27
2.5.1.2 Paquete: Módulo Administrativo 28

2.6 DESCRIPCIÓN DE LOS CASOS DE USO 29
2.7 CONCLUSIONES 37

3 CAPÍTULO3: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA 38

3.1 INTRODUCCIÓN 38
3.2 DESCRIPCIÓN DE LA ARQUITECTURA 38
3.2.1 Arquitectura de 4 Capas 38
3.3 PATRONES DE DISEÑO 40
3.3.1 Patrón Fábrica Pura 40
3.4 MODELO DE DISEÑO 40
3.4.1 Diagrama de clases del diseño 41
3.4.2 Módulo Docente-Metodológico 42
3.4.3 Módulo Administrativo 46
3.4.4 Paquete Servicios 49
3.4.4.1 Módulo Docente-Metodológico 49
3.4.4.2 Módulo Administrativo 53
3.4.5 Diagramas de interacción 56
3.5 DISEÑO DE LA BASE DE DATOS 58
3.6 DIAGRAMA DE DESPLIEGUE 60
3.7 CONCLUSIONES 61

4 CAPÍTULO4: ANÁLISIS DE ESTIMACIÓN DE ESFUERZO Y TIEMPO DE DESARROLLO 62

4.1 INTRODUCCIÓN 62
4.2 ESTIMACIÓN DE ESFUERZO 62
4.3 BENEFICIOS TANGIBLES E INTANGIBLES 67
4.4 ANÁLISIS DE LAS ESTIMACIONES Y BENEFICIOS 68
4.5 CONCLUSIONES 68

CONCLUSIONES GENERALES 69

RECOMENDACIONES 70

REFERENCIAS BIBLIOGRÁFICAS 71

ANEXOS 73

GLOSARIO DE TÉRMINOS 93

ÍNDICE DE FIGURAS

Figura 1. Vista Global de Actores 18
Figura 2. Diagrama de Paquetes 26
Figura 3. Diagrama de caso de Uso. Módulo Docente-Metodológico. 27
Figura 4. Diagrama de caso de Uso. Módulo Administrativo. 28
Figura 5. Arquitectura de 4 Capas 39
Figura 6. Diagrama de Clases: Gestionar Plantilla del Colectivo 42
Figura 7. Diagrama de Clases: Gestionar Plan de Trabajo Metodológico 43
Figura 8. Diagrama de Clases: Gestionar Plan de Trabajo Individual 44
Figura 9. Diagrama de Clases: Gestionar Controles a Clases 45
Figura 10. Diagrama de Clases: Gestionar Usuarios 46
Figura 11. Diagrama de Clases: Autenticar 47
Figura 12. Diagrama de Clases: Gestionar Noticias 48
Figura 13. Diagrama de Clases Paquete de servicios: Gestionar Plantilla del Colectivo 49
Figura 14. Diagrama de Clases Paquete de servicios: Gestionar Plan de Trabajo Metodológico 50
Figura 15. Diagrama de Clases Paquete de servicios: Gestionar Plan de trabajo Individual del Asesor 51
Figura16. Diagrama de Clases Paquete de servicios: Gestionar Controles a Clases 52
Figura 17. Diagrama de Clases Paquete de servicios: Gestionar Usuarios 53
Figura 18. Diagrama de Clases Paquete de servicios: Autenticar 54
Figura 19. Diagrama de Clases Paquete de servicios: Gestionar Noticias 55
Figura 20. Modelo de Datos 59
Figura 21. Diagrama de Despliegue 60

ÍNDICE DE TABLAS

Tabla 1. Descripción de Actores 16
Tabla 2. Gestionar Plan de trabajo Metodológico (PTM) 29
Tabla 3. Evaluar PTM 29
Tabla 4. Gestionar Plan de Trabajo Individual (PTI) 30
Tabla 5. Evaluar Asesor 30
Tabla 6. Gestionar Controles a Clases 30
Tabla 7. Evaluar Controles a Clases 31
Tabla 8. Gestionar Actas de Reunión 31
Tabla 9. Gestionar plantilla de colectivo. 31
Tabla 10. Gestionar plantilla de Alumnos Ayudantes. 32
Tabla 11. Mostrar Informe del Claustro 32
Tabla 12. Mostrar Control a Clases 33
Tabla 13. Mostrar Plantilla de la Disciplina 33
Tabla 14. Mostrar Plantilla de Alumnos Ayudantes 33
Tabla 15. Gestionar Usuarios 34
Tabla 16. Gestionar Prenómina de Pago y Aseo. 34
Tabla 17. Gestionar Solicitud de Materiales. 35
Tabla 18. Gestionar noticias 35
Tabla 19. Gestionar Rol 35
Tabla 20. Autenticar usuario 36
Tabla 21. Mostrar Reportes de Materiales solicitados. 36
Tabla 22. Mostrar Noticias. 36
Tabla 23. Diagramas de Secuencia 56
Tabla 24. Clasificación de los Actores 63
Tabla 25. Clasificación de los Casos de Uso 63
Tabla 26. Factor de Complejidad 64
Tabla 27. Factor Ambiente 65
Tabla 28. Esfuerzo 67

INTRODUCCIÓN

La Universidad de las Ciencias Informáticas (UCI) es un centro creado a raíz de la batalla de ideas, cuya misión es formar de manera continua profesionales integrales comprometidos con la patria, ser el soporte de la informatización del país y la competitividad internacional de la industria cubana del software. Por ello es necesario lograr una buena organización y control del proceso metodológico, que constituye el corazón de los centros de educación superior. En la actualidad en la UCI se realiza un arduo trabajo en este sentido, con la finalidad de lograr una mayor eficiencia en la planificación, y elevar la calidad de los resultados.

En la UCI la responsabilidad del proceso docente metodológico recae en la Vicerrectoría de Formación. Es ella la encargada de organizar y dirigir el proceso metodológico, que incluye la planificación y organización del proceso docente educativo. Esta a su vez es parte del trabajo metodológico, lo que posibilita que las actividades docentes se integren y sistematicen eficientemente para el logro de los objetivos generales de la formación de los profesionales. Además estas tareas contribuyen a garantizar un adecuado balance de la carga docente del estudiante y un uso Rational de los recursos humanos y materiales de los que dispone el país, y en este caso la Universidad. La Dirección de Formación del Profesional y los Departamentos Docentes Centrales (DDC) tienen una gran responsabilidad en el proceso docente metodológico. Ambos tienen como objetivo superar y capacitar profesionales de esta universidad, orientando así este proceso metodológico en los Departamentos Docentes de las Facultades, los que desarrollan una serie de tareas para llevar a cabo una buena preparación metodológica y profesional, además de tener un control estricto de la docencia por asignatura.

El Departamento Docente Central (DDC) de Ingeniería y Gestión de Software de la UCI no realiza un control eficiente de la información en el proceso docente, metodológico, y laboral. Esto trae como consecuencia que se dificulte y demore el acceso a la información que se necesite para la toma de decisiones, y que se haga difícil el trabajo de todos los miembros del departamento en el chequeo del cumplimiento de las actividades docente, y planeación metodológica.

Para esto se propone como problema científico lo siguiente:

¿Cómo erradicar los problemas del funcionamiento y control de los procesos metodológicos y profesionales en el Departamento Docente Central de Ingeniería y Gestión de Software mediante la automatización de un Sistema de Gestión?

El objeto de estudio radica en la gestión de información asociada a los procesos existentes en el DDC - Ingeniería y Gestión de Software -UCI.

El campo de acción de este trabajo se centra en la gestión de información asociada a los procesos de dirección y control del DDC - Ingeniería y Gestión de Software de la UCI.

El objetivo general de este trabajo es implementar un sistema informático para la automatización de los procesos de dirección y control del DDC - Ingeniería y Gestión de Software.

Para dar cumplimiento a este objetivo se desarrollarán diferentes tareas:

- Identificar y describir los procesos elementales del negocio.
- Realizar un estudio del estado del arte de la tecnología a utilizar, analizando la posibilidad de su uso para implementar el software.
- Definir la arquitectura a utilizar en la aplicación a desarrollar.
- Implementar la Aplicación Web haciendo uso de RUP como Metodología de desarrollo de Software.

Como idea a defender se plantea que si se desarrolla una aplicación Web que automatice la gestión de información en el DDC - Ingeniería y Gestión de Software -UCI, entonces se facilitará la planeación y toma de decisiones en el proceso docente metodológico del departamento.

A su vez, para guiar la investigación científica se utilizarán los métodos:

- Teóricos
    - Analítico _ Sintético: Se analizan las teorías, documentos entre otros, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.
    - Inductivo _ deductivo: Son formas de razonamiento que permiten llegar a un grupo de conocimientos generalizadores, tanto desde el análisis de lo particular a lo general, como desde el análisis de los elementos generalizadores a uno de menor nivel de generalización.

- Empíricos

    - Observación: Registro visual de lo que ocurre en una situación real, en un fenómeno determinado, clasificando y consignando los hechos y acontecimientos pertinentes de acuerdo con algún esquema previsto.

El presente trabajo ha sido organizado de la siguiente manera:

Capítulo 1: En este Capítulo se detallan los procesos elementales de negocio, se realiza un estudio y análisis de las tecnologías apropiadas para desarrollos de aplicaciones Web, escogiendo la mejor opción para la construcción del sistema propuesto.

Capítulo 2: Después de haber analizado y refinado cada uno de los requerimientos imprescindibles para el funcionamiento del nuevo sistema, este capítulo se propone mostrar a partir de estos requisitos un mejoramiento del Diagrama de Caso de Uso, el cual representa las relaciones de los actores que interactúan con el sistema y el flujo de actividades con las que interactúan.

Capítulo 3: En el presente capítulo se realiza el diseño de la propuesta de solución, seleccionando una arquitectura y modelándose los artefactos que contribuyen al desarrollo de aplicaciones Web, con el uso de Frameworks Manipuladores de Contenidos flexibles. Se elabora el modelo de datos adecuado. Se analiza como va a estar distribuido el sistema.

Capítulo 4: En este capítulo se evalúa la factibilidad y beneficios del sistema propuesto, utilizando el método de estimación por Puntos de Casos de Uso. Se obtendrán valores de importantes indicadores como son: esfuerzo y tiempo de desarrollo.

Capítulo 1

1 CAPÍTULO1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este Capítulo se detallan los procesos elementales de negocio, se realiza un estudio y análisis de las tecnologías apropiadas para desarrollos de aplicaciones Web, escogiendo la mejor opción para la construcción del sistema propuesto.

1.2 Objeto de estudio

La dirección y control de los procesos metodológicos en la UCI es responsabilidad de la Vicerrectoría de Formación, conformada por la vicerrectora y diferentes direcciones como: la Dirección de Planificación y Control, Dirección de Teleformación, Dirección de Formación del Posgraduado, y Dirección de Formación del Profesional. Ellas en su conjunto dirigen los Departamentos Docentes Centrales, entre los que se encuentra el Departamento Docente Central de Ingeniería y Gestión de Software. Este Departamento necesita gestionar la información para un mejor control del flujo de datos y un rápido acceso a los mismos para así garantizar la organización y dirección de los procesos en el trabajo docente metodológico.

1.3 Sistemas automatizados existentes vinculados al campo de acción

En el estudio realizado sobre los sistemas de gestión existentes en el mundo no se encontró ninguno que se adaptara a los requisitos que se necesitan en el Sistema de Gestión Metodológica Central de la UCI. En la actualidad en la Universidad no se ha llegado a realizar ningún software relacionado con este campo de acción, pero si se ha realizado un estudio y análisis de los principales procesos y actividades elementales que se desarrollan en el Departamento Central de Ingeniería y Gestión de Software(BATISTA VELAZQUEZ 2006), así como sus dificultades al gestionar los mismos, por lo que se detallaron los procesos elementales del negocio como: Confeccionar la Plantilla de la Disciplina, Elaboración y control

del Plan de Trabajo Metodológico, Elaboración y control del Plan de Trabajo Individual, Planificación y control de los Controles a Clases, Confección y archivo de la Solicitud de Materiales, Confección de la Prenómina de Pago y Aseo y la Gestión de Noticias. Además se realizó un estudio de las posibles tecnologías a utilizar, de ellas se seleccionó como Lenguaje de Programación PHP, Zend Studio como un editor de texto para paginas Web programadas en PHP, como sistema gestor de base de datos se estudió MySQL, como sistema Gestor de Contenidos Drupal y la metodología de desarrollo del software RUP y como diseño arquitectónico se analizó al Modelo Vista Controlador.

A partir del estudio realizado, retomando los procesos elementales del negocio se ha refinado el análisis de cada uno de ellos precisando las funciones que se realizarán en cada uno y el responsable, o sea, el encargado de realizar dicha actividad o proceso. También partiendo de las tecnologías anteriores se han analizado con mayor profundidad algunas de ellas y se han estudiado otras de gran importancia que serán utilizadas en el desarrollo del sistema informático. Además se describe la arquitectura que se utilizará para implementar el sistema.

1.4 Procesos elementales del Negocio

Para la creación del Sistema de Gestión del Departamento Docente Central de Ingeniería y Gestión de Software se detallaron diferentes procesos a automatizar que cubren las necesidades fundamentales de este departamento. Este trabajo se propone partir de esos procesos, a los cuales se le realizarían algunas modificaciones pertinentes para la construcción del sistema que cubra la totalidad de las necesidades y preocupaciones de este departamento para con esto lograr un excelente trabajo docente metodológico.

A continuación se muestran los procesos a automatizar:

1. **Gestión de la Plantilla del Colectivo:** Para una buena organización del colectivo el Jefe de Colectivo registra todos los datos de profesores de cada colectivo por facultad. Si sucedieran modificaciones del colectivo el departamento debe tener control de estas, lo cual se realiza a través de la plantilla del colectivo.
2. **Gestión y control del Plan de Trabajo Metodológico:** Las actividades que se desarrollan en el semestre son el resultado del análisis de etapas anteriores y nuevas proyecciones para mejorar el proceso docente metodológico. Una buena planificación, ejecución y control garantizan el éxito del resultado docente metodológico. Esta planificación la efectúa el Metodólogo (Jefe de Departamento, Asesor) en el Plan de Trabajo Metodológico, El Jefe de Departamento es el

encargado de verificar el estado de cumplimiento de las actividades planificadas, chequeando resultados de las actividades hasta el momento y emitiendo una observación que justifica el estado asignado.

3. **Gestión y control del Plan de Trabajo Individual (PTI):** El Metodólogo es el encargado de elaborar el Plan de Trabajo Individual de los asesores, el cual se deriva del Plan de Trabajo Metodológico, donde los asesores se trazan objetivos, y actividades a cumplir. Es de suma importancia el papel de los asesores, pues ellos complementan los objetivos del departamento. Periódicamente el Jefe de Departamento evalúa el cumplimiento de los objetivos del PTI.
4. **Gestión y control de los controles a clases:** Los controles a clases se planifican seleccionando responsables, facultades y profesores a controlar. Analizan las deficiencias detectadas y permite introducir actividades en el Plan de Trabajo Metodológico en favor de erradicarlas, registrando los resultados de los controles. Esto da un indicador del nivel de preparación de los profesores y dónde están los principales problemas. La planificación de estos controles es realizada por el Metodólogo.
5. **Gestión de solicitud de materiales:** Para la obtención de materiales docentes, el Jefe de Departamento realiza una solicitud a la Vicerrectoría de Formación. Es de suma importancia un control eficaz de estas solicitudes, con el fin de garantizar peticiones acorde a las solicitudes anteriores, la secretaria es la que confecciona esta solicitud.
6. **Gestión de la prenómina de pago y aseo:** Para la obtención del pago y aseo que reciben los trabajadores del departamento cada mes, es responsabilidad de la secretaria confeccionar la prenómina y enviarla a la dirección de recursos humanos y el Jefe de Departamento es el encargado de aprobarla. (BATISTA VELAZQUEZ 2006)

1.5 Tendencias y tecnologías actuales

1.5.1 Política de migración a software libre

En al actualidad Cuba avanza hacia el 'software' libre. El plan es utilizar el sistema operativo GNU/Linux como sistema de base, para reemplazar al sistema actualmente en uso, Microsoft Windows.

La base de usuarios de Linux en Cuba es de cerca de 1.500 usuarios y se dispone de una distribución Linux propia. En Cuba también hay algunos desarrolladores de software libre y la Universidad de las Ciencias Informáticas, con una gran cantidad de estudiantes, ha asegurado la participación de una de sus facultades para el desarrollo de programas para Linux. (SÁNCHEZ MATÍAS 2004)

Como institución de avanzada en el campo de la informática, la UCI está prácticamente obligada a llevar a cabo esta política, y cuanto antes mejor. El presente trabajo parte de esa premisa y se propone la construcción de un sistema que satisfaga las necesidades y preocupaciones que dieron lugar a su origen, haciendo uso de herramientas y tecnologías libres.

1.5.2 Hipertexto Preprocesor (PHP)

"PHP, acrónimo de Hypertext Preprocessor", es un lenguaje "Open Source"(código abierto) interpretado y de alto nivel, especialmente pensado para desarrollos Web. La mayoría de su sintaxis es similar a C, Java y Perl y es extremadamente fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas Web, páginas dinámicas de una manera rápida y fácil.

PHP es uno de los lenguajes de lado servidor más extendidos en la Web. Nacido en 1994, se trata de un lenguaje de creación relativamente creciente que ha tenido una gran aceptación en la comunidad de webmasters debido sobre todo a la potencia y simplicidad que lo caracterizan. (VIAL S 2001-2006)

El lenguaje PHP esta siendo más utilizado en la actualidad, por una serie de ventajas con respecto a los demás lenguajes de scripting en el lado del servidor como son:

- PHP es libre y abierto.
- Presenta el código fuente disponible, diseñado para la Web, es multiplataforma hardware, multisistema operativo.
- Soporte para varios servidores Web.
- Soporte para cualquier Base de Datos.
- Tiene una buena documentación, miles de ejemplos, es bastante sencillo de aprender y utilizar.
- Presenta amplia base de usuarios, no depende de un único proveedor de servicios.
- Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envió de correo, gestión de un e-commerce, xml, creación de PDF...).

La quinta versión cuenta con innumerables mejoras que consolidan su éxito. Se ofrece la posibilidad de hacer programas orientados a objetos, lectura de archivos XML de forma sencilla, utilización de la base de datos ligera SQLite o la implementación de servicios Web. (CABEZAS GRANADO 2004)

Actualmente se encuentra en su versión 6. Es evidente que se ha convertido en la gran tendencia en el mundo de Internet. Las estadísticas arrojan que cada mes su uso crece en un 15% y cuenta con una de

las comunidades más grandes de Internet, lo cual facilita encontrar ayuda, documentación, y otros recursos relacionados.

1.5.3 Plataforma de desarrollo Web

Un framework es una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación, este puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

También un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Un framework Web, por tanto, podemos definirlo como un conjunto de componentes (por ejemplo clases en java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web.

Existen varios tipos de frameworks Web: orientados a la interfaz de usuario, como Java Server Faces, orientados a aplicaciones de publicación de documentos, como Coocon, orientados a la parte de control de eventos, como Struts y algunos que incluyen varios elementos como Tapestry. La mayoría de frameworks Web se encargan de ofrecer una capa de controladores de acuerdo con el patrón Modelo Vista Controlador (MVC) ofreciendo mecanismos para facilitar la integración con otras herramientas para la implementación de las capas de negocio y presentación. (GUTIÉRREZ. 2007)

1.5.4 NuSphere PHpED

Es un entorno de desarrollo profesional para PHP, con depurador, análisis de errores, ayudas para la localización de cuellos de botella en el código, publicación segura de código en servidores e integración con herramientas de terceros. NuSphere PhpED es hoy en día un ambiente de desarrollo integrado superior para el php. Conveniente ambos para los trabajos individuales pequeños y los proyectos grandes del multi-revelador. PhpED es una herramienta robusta que ofrece la funcionalidad del full-cycle para el desarrollo de Sitios y Aplicaciones Web. Como en cualquier proceso complicado, la eficacia del desarrollo del php depende de la opción de las herramientas de la producción.

NuSphere ha desarrollado a Nu-Codificador - un codificador de gran alcance de PHP que permite a los reveladores de PHP proteger su código de PHP del copiado desautorizado. El Nu-Codificador convierte el código fuente de la escritura de PHP en los bytecodes compilados de PHP para el funcionamiento acelerado y su máxima seguridad. Este codificador tiene grandes ventajas como:

- Facilidad de Empleo
- Alto rendimiento
- Integración apretada
- Ayuda técnica (CORPORATION 2007-03-29)

1.5.5 Sistema gestor de bases de datos PostgreSQL

Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad.

El Sistema Gestor de Base de Datos PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre.

PostgreSQL ofrece muchas ventajas respecto a otros sistemas de bases de datos:

Instalación ilimitada: Es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia. Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.

Esto tiene varias ventajas adicionales:

- Modelos de negocios más rentables con instalaciones a gran escala.
- No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.

Ahorros considerables en costos de operación: El software ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que los productos de los proveedores comerciales, conservando todas las características, estabilidad y rendimiento.

Estabilidad y confiabilidad legendarias: En contraste a muchos sistemas de bases de datos comerciales, es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.

Extensible: El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo que también extienden PostgreSQL todos los días.

Multiplataforma: PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows está actualmente en estado beta de pruebas.

Diseñado para ambientes de alto volumen: PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC (Acceso concurrente multiversión) para conseguir una mejor respuesta en ambientes de grandes volúmenes. Los principales proveedores de sistemas de bases de datos comerciales usan también esta tecnología, por las mismas razones.

Herramientas gráficas de diseño y administración de bases de datos: Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Tora, Data Architect). (DÍAZ 2003)

1.5.5.1 EMS PostgreSQL Manager

EMS PostgreSQL Manager es una herramienta de alto rendimiento para la administración de servidores de base de datos en PostgreSQL. Este trabaja con cualquier versión de PostgreSQL hasta la 8.0 y soporta todas las características últimas de este incluyendo los tablespaces, los argumentos de los nombres en funciones. Ofrece herramientas de gran alcance para que los usuarios experimentados satisfagan todas sus necesidades, tales como, el diseñador visual de la base de datos, el constructor visual entre otros. (SOLUTIONS 2005-07-05)

1.5.6 ERwin

ALLFusion Erwin Data Modeler es una herramienta de diseño de bases de datos que ayuda a generar, mantener alta calidad y permite gran rendimiento en las aplicaciones de bases de datos. Desde un modelo lógico de los requerimientos de información y las reglas de negocio que definen las bases de datos al modelo físico optimizado por las características específicas de las bases de datos. ALLFusion Erwin Data Modeler permite visualizar la estructura y elementos clave, además optimizar el diseño de las bases de datos. Esta automáticamente genera tablas y cientos de líneas de procedimientos almacenados y código trigger para las bases de datos. La tecnología "complete-compare" permite el desarrollo iterativo para que los modelos estén siempre sincronizados con la base de datos. Esta presenta como ventajas:

- Fácil acceso a cualquier base de datos relacional.
- Comparación comprensiva entre el modelo de datos y la base de datos

Soporta la separación del modelo lógico y del físico. (MANAGEMENT 2007)

1.5.7 Extensible Markup Language (XML)

XML es Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas. Es una simplificación y adaptación del SGML(Lenguaje de Marcación Generalizado) y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML(lenguaje extensible de marcado de hipertexto), SVG(Scalable Vector Graphics), MathML(lenguaje de marcado basado en XML).

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. Se pueden resumir las siguientes ventajas:

- Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.

- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan bugs y se acelera el desarrollo de la aplicación.

- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones. (WIKIMEDIA 20 de marzo 2007)

1.5.8 Hojas de Estilo en Cascada (Cascade Style Sheets, CSS).

Las hojas de estilo en cascada son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML. La finalidad de ellas es crear unos estilos físicos, separados de las etiquetas HTML, y aplicarlos en los bloques de texto en los que se desean. Estos estilos podrán ser modificados en algunas ocasiones desde JavaScript, lo que brinda mayor interactividad. Además la sintaxis CSS permite aplicar al documento formato de modo mucho más exacto, y puede definir atributos en las páginas utilizando unidades como: pixels, porcentaje, pulgadas, puntos y centímetros. (NIELSEN 2007)

En resumen, las hojas de estilo permiten separar el formato visual de las páginas de contenido. Después de analizar la ventaja que representan, se propone su utilización en la propuesta de solución.

1.5.9 UML (Lenguaje Unificado de Modelado)

El Lenguaje Unificado de Modelado (UML) es un lenguaje estándar para escribir planos de software. Este puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema. (BARRIENTOS ENRÍQUEZ 2005)

Es un lenguaje de propósito general para el modelado orientado a objetos. Este es también un lenguaje de modelamiento visual que permite una abstracción del sistema y sus componentes. (RUMBAUGH *et al.* 2000)

El mismo pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguaje de programación, así como construir modelos por ingeniería inversa a partir de programas existentes.

UML ha ejercido un gran impacto en la comunidad software, su utilización se ha extendido en todo el mundo para construir aplicaciones en todos los dominios y de todos los tamaños. Los entornos de desarrollos más utilizados como son los de Borland, Microsoft e IBM intengran herramientas para modelado con UML. (ROSSI *et al.* Marzo-Abril)

1.5.10 Otras herramientas utilizadas

Para confeccionar la propuesta de este trabajo, se utilizó como editor de páginas Web el Macromedia Dreamweaver MX 2004, por ser la herramienta de creación de sitios Web más utilizada en la actualidad y poseer un amplio soporte para la creación y utilización de CSS, logrando un diseño sencillo y óptimo.

Para modelar la aplicación se utilizó Rational Rose Enterprise Edition 2003, herramienta líder para este propósito, la cual facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue).

1.5.11 El Proceso Unificado de Desarrollo de Software (RUP)

El software es un componente esencial de toda actividad basado en el uso de la informática, es por ello que la calidad en su desarrollo y mantenimiento resulta hoy en día uno de los principales objetivos estratégicos de las organizaciones. En los últimos años se han publicado diversos estudios en los que se presentan los principios que se deben seguir para mejorar los procesos de software, y de esta forma evitar las grandes catástrofes que conllevan al fracaso de un gran número de proyectos.

El Proceso Unificado de Desarrollo de Software es una de las metodologías más generales y más usadas de las que existen en la actualidad, pues está pensada para adaptarse a cualquier proyecto. Constituye además, una propuesta de proceso para el desarrollo de software orientado a objeto.

El RUP provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible.

El Proceso Unificado se basa en componentes, lo que significa que el sistema en construcción está hecho de componentes de software interconectados por medio de interfaces bien definidas. Este además usa el Lenguaje Unificado de Modelado (UML) en la preparación de todos los planos del sistema. De hecho, UML es una parte integral del Proceso Unificado, fueron desarrollados a la par. Los aspectos distintivos del Proceso Unificado están capturados en tres conceptos clave:

- Dirigido por casos de uso (use-case driven): Un caso de uso es una pieza en la funcionalidad del sistema que le da al usuario un resultado de valor. Los casos de uso capturan los requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad completa del sistema. Este modelo reemplaza la tradicional especificación funcional del sistema. Una especificación funcional tradicional se concentra en responder la pregunta: ¿Qué se supone que el sistema debe hacer? La estrategia de casos de uso puede ser definida agregando tres palabras al final de la pregunta… ¿por cada usuario? Estas tres palabras tienen una implicación importante, nos fuerzan a pensar en términos del valor a los usuarios y no solamente en términos de las funciones que sería bueno que tuviera. Sin embargo, los casos de uso no son solamente una herramienta para especificar los requerimientos del sistema, también dirigen su diseño, implementación y pruebas, esto es, dirigen el proceso de desarrollo.

- Centrado en la arquitectura (architecture-centric): El concepto de arquitectura de software involucra los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, tal y como las interpretan los usuarios y otros stakeholders, y tal y como están reflejadas en los casos de uso. Sin embargo, también está influenciada por muchos otros factores, tales como la plataforma de software en la que se ejecutará, la disponibilidad de componentes reutilizables, consideraciones de instalación, sistemas legados, requerimientos no funcionales (ej. desempeño, confiabilidad). La arquitectura es la vista del diseño completo con las características más importantes hechas más visibles y dejando los detalles de lado. Ya que lo importante depende en parte del criterio, el cual a su vez viene con la experiencia, el valor de la arquitectura depende del personal asignado a esta tarea.

Sin embargo, el proceso ayuda al arquitecto a enfocarse en las metas correctas, tales como claridad flexibilidad en los cambios futuros y reuso.

- Iterativo e incremental: Desarrollar un producto de software comercial es una tarea enorme que puede continuar por varios meses o años. Es práctico dividir el trabajo en pequeños pedazos o mini-proyectos. Cada mini-proyecto es una iteración que finaliza en un incremento. Las iteraciones se refieren a pasos en el flujo de trabajo, los incrementos se refieren a crecimiento en el producto. Para ser más efectivo, las iteraciones deben estar controladas, esto es, deben ser seleccionadas y llevadas a cabo de una manera planeada. (JACOBSON *et al.* 2000)

Todo esto en su conjunto es lo que hace único al Proceso Unificado.

1.6 Selección de las Tecnologías a utilizar en la propuesta solución

Luego de haber realizado un profundo análisis de las principales características de las tecnologías más utilizadas en la actualidad, se acuerda utilizar:

- PHP como lenguaje de programación del lado del servidor, por todas las ventajas que presenta. Es multiplataforma, multisistema operativo, con una sintaxis familiar y bien definida a los programadores, bastante sencillo y fácil de aprender, dispone de gran cantidad de documentos y ejemplos en Internet y además es libre y de código abierto. Como ambiente de desarrollo se escoge NuSphere pues trae varias ventajas durante la programación en PHP.
- El sistema gestor de bases de datos PostGreSQL porque con respecto a MySQL presenta una serie de ventajas como gran escalabilidad, por ser capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta, por implementar el uso de subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y por ofrecer soluciones en campos en las que MySQL no podría, además PostGreSQL presenta mayor cantidad de tipos de datos, lo que permite optimizar espacio en la Base de Datos.
- Como modelador de datos el ERwin, ya que permite visualizar todos los datos de una base de datos.
- XML como lenguaje de marcas extensible.
- Como metodología de desarrollo RUP por todas las ventajas de organización que brinda y por venir acompañada de una potente herramienta que soporta todos los procesos básicos de RUP: Suite del Rational. Además se utilizará a UML como lenguaje Unificado de Modelado.

1.7 Conclusiones

En este capítulo se han descrito los procesos elementales del negocio, los cuales son de gran importancia para la automatización del sistema. Además se realizó un análisis de las tecnologías y herramientas actuales, describiéndose las características fundamentales de las seleccionadas para el desarrollo del sistema.

Capítulo 2

2 CAPÍTULO2: REQUERIMIENTOS

2.1 Introducción

Después de haber analizado y refinado cada uno de los requerimientos imprescindibles para el funcionamiento del nuevo sistema, este capítulo se propone mostrar a partir de estos requisitos un mejoramiento del Diagrama de Caso de Uso, el cual representa las relaciones de los actores que interactúan con el sistema y el flujo de actividades con las que interactúan.

2.2 Actores del Sistema

Un actor es una idealización de una persona externa, de un proceso, o de una cosa que interactúa con un sistema, un subsistema, o una clase. Un actor caracteriza las interacciones que los usuarios exteriores pueden tener con el sistema, pueden ser definidos en jerarquías de generalización, en las cuales una descripción abstracta del actor es compartida y aumentada por una o más descripciones específicas del actor. Un actor puede ser un ser humano, otro sistema informático, o un cierto proceso ejecutable. Se dibuja a un actor como una persona pequeña con trazos lineales y el nombre debajo de él. En la Tabla 1 se hace una descripción de los actores del sistema que se desarrolla.

Tabla 1. Descripción de Actores

Nombre del actor	Descripción
Metodólogo	Es el encargado de Gestionar el Plan de Trabajo Metodológico, los Controles a Clases, así como mostrar los mismos, Gestionar Plan de trabajo individual del asesor, Gestionar noticias, Mostrar

	informe del claustro, Mostrar reportes de materiales solicitados. Las acciones de este actor las puede ejecutar el Jefe de Departamento y el Asesor.
Jefe de Departamento	Es responsable de Mostrar la plantilla de la disciplina, así como de chequear y evaluar el cumplimiento de las actividades metodológicas y del asesor, y evaluar los controles a clases.
Jefe de Colectivo	Es el encargado de la gestión de la plantilla del colectivo y de la de alumnos ayudantes.
Usuario	Es aquel que en determinado momento solicita entrar al sistema y posterior a eso puede tener acceso a determinado contenido, en correspondencia con el rol que este juegue y es encargado de mostrar las noticias.
Administrador	Es el responsable de instalación y soporte del sistema, gestionar usuarios y gestionar roles.
Secretaria	Es la responsable de Gestionar la Prenómina de Pago y Aseo y la Solicitud de Materiales.
Responsable de Reunión	Es el encargado después de terminada las reuniones gestionar las actas.
Asesor	Este realiza las mismas actividades que el Metodólogo.

2.2.1 Vista Global de Actores

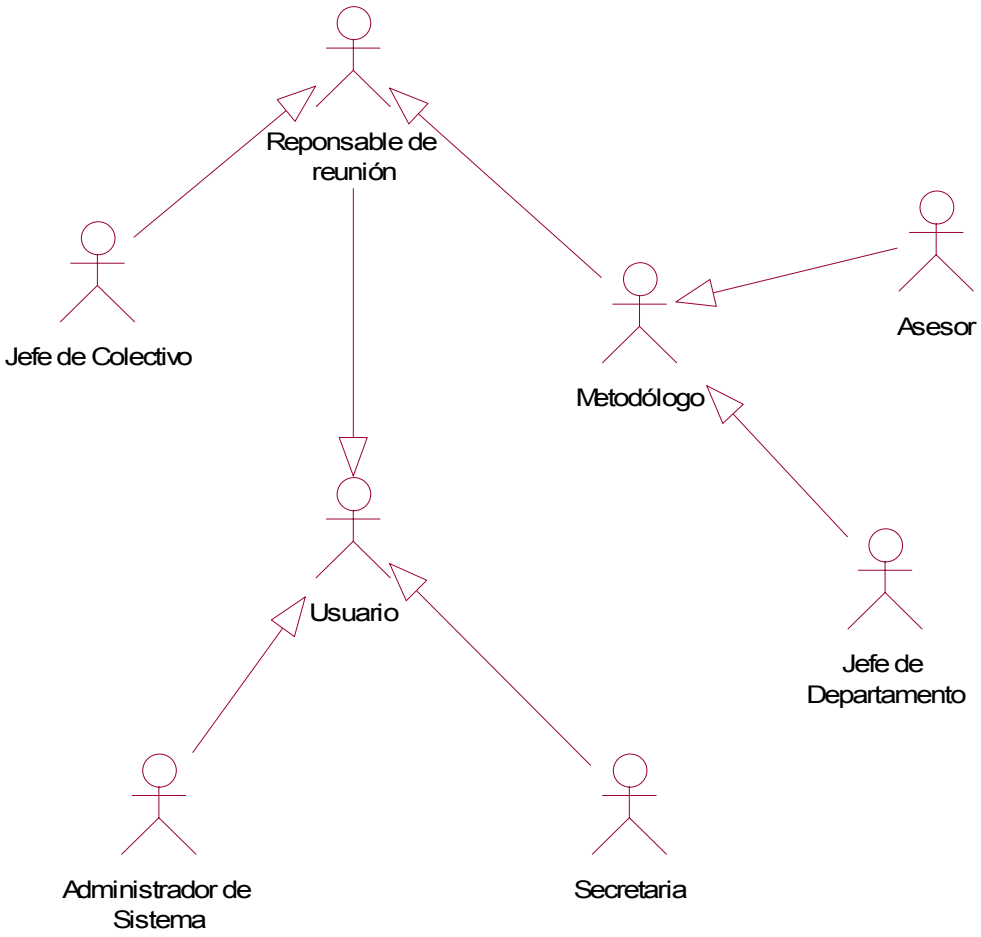


Figura 1. Vista Global de Actores

2.3 Requerimientos funcionales Refinados

RF1- Gestionar Plan de Trabajo Metodológico (PTM).

RF1.1- Registrar introducción al PTM.

RF1.2- Registrar Diagnóstico del Claustro al PTM.

RF1.3- Registrar objetivos al PTM.

RF1.4- Modificar Objetivos**.**

RF1.5- Eliminar Objetivos.

RF1.6- Registrar Acciones al PTM (nombre, tipo de acción, responsable).

RF1.7- Modificar Acciones.

RF1.8- Eliminar Acciones.

RF1.9- Registrar actividades al PTM (nombre, fecha, tipo de actividad, responsable).

RF1.10- Modificar Actividades del PTM.

RF1.11- Eliminar Actividades del PTM.

RF1.12- Mostrar PTM (dada una fecha o curso).

RF1.13- Imprimir el PTM por impresora.

RF1.14- Enviar por e-mail el PTM.

RF2- Evaluar PTM.

RF2.1- Evaluar estado de cumplimiento de las actividades (cumplida, no cumplida, parcialmente cumplida).

RF2.2- Mostrar resumen del cumplimiento de las actividades por tipo de actividad, mostrando total y estado actual.

RF2.3- Registrar observación general de la evaluación.

RF2.4- Evaluar el Plan de Trabajo (cumplido, no cumplido, parcialmente cumplido).

RF3- Gestionar Plan de Trabajo del Asesor (PTI).

RF3.1- Seleccionar Curso.

RF3.2- Agregar actividades (nombre, tipo de actividad).

RF3.3- Modificar y eliminar actividades.

RF3.4- Mostrar PTI de un asesor dado.

RF3.5- Imprimir PTI por impresora.

RF3.6- Enviar resumen del PTI por e-mail.

RF4- Evaluar asesor.

RF4.1- Evaluar estado de cumplimiento de las actividades (cumplida, no cumplida, parcialmente cumplida).

RF4.2- Mostrar resumen del cumplimiento de las actividades por tipo de actividad, mostrando total y estado actual.

RF4.3- Registrar observación general de la evaluación.

RF4.4- Evaluar el Plan de Trabajo Individual.

RF5- Gestionar plantilla del colectivo.

RF5.1- Introducir profesores al colectivo (facultad, nombre, correo, asignatura, categoría docente, categoría científica, años de experiencia, jefe de colectivo, jefe de asignatura).

RF5.2- Modificar profesores del colectivo.

RF5.3- Eliminar profesores del colectivo.

RF5.4- Mostrar Plantilla del Colectivo.

RF6- Gestionar Plantilla de Alumnos Ayudantes.

RF6.1- Adicionar Alumnos Ayudantes (facultad, nombre, correo, años, centro, tutor, asignatura).

RF6.2- Mostrar plantilla de Alumnos Ayudantes.

RF6.3- Modificar datos del Alumno Ayudante.

RF6.4- Eliminar alumno Ayudante de la plantilla.

RF7- Gestionar Actas de Reunión.

RF7.1- Almacenar Acta.

RF7.2- Modificar Acta.

RF7.3- Eliminar Acta.

RF7.4- Mostrar Acta.

RF8- Gestionar Controles a Clases.

RF8.1- Adicionar control a clase (nombre de visitado, nombre del visitante, fecha, turno, lugar).

RF8.2- Modificar control a clases.

RF8.3- Eliminar Control a clases.

RF8.4- Evaluar control a clase (nombre del visitado, fecha, evaluación, documento).

RF8.5- Mostrar Evaluación.

RF8.6- Modificar Evaluación.

RF8.7- Eliminar Evaluación.

RF8.8- Mostrar reporte de la cantidad de controles a clases realizados, cantidad de Bien, Regular y Mal.

RF9- Mostrar Informe del Claustro.

RF9.1- Mostrar características del claustro como diagnóstico inicial del PTM (total de profesores, total de profesores por categoría docente, Año promedio de experiencia en la disciplina, cantidad de alumnos ayudantes).

RF10- Mostrar Plantilla de la Disciplina

RF10.1- Mostrar Datos del Profesor (nombre, correo, asignatura, categoría docente, categoría científica, años de experiencia, jefe de colectivo, jefe de asignatura).

RF11- Mostrar Plantilla de Alumnos Ayudantes

RF11.1- Mostrar Datos de los Alumnos Ayudantes (nombre, correo, años, centro, tutor, asignatura).

RF12- Autentificar Usuario.

RF12.1- Verificar que el usuario este registrado en el sistema.

RF12.2- Verificar usuario y contraseña en el dominio UCI.

RF12.3- Verificar usuario y contraseña en el sistema.

RF12.4- Chequear y almacenar privilegios de acceso a una página abierta.

RF13- Gestionar Usuarios.

RF13.1- Adicionar un nuevo usuario al sistema (usuario, nombre, email, rol).

RF13.2- Modificar usuario existente.

RF13.3- Eliminar usuario.

RF13.4- Asignar privilegios.

RF13.5- Mostrar Usuarios.

RF14- Gestionar Solicitud de Materiales.

RF14.1- Crear planilla de solicitud de materiales (dirección, código1, almacén, código 2, orden de producción No, Lote No, centro de costo, código 3, orden de trabajo No, producto código, otros, materiales, solicita, recibe, autoriza, No).

RF14.1.1- Agregar materiales.

RF14.1.2- Eliminar materiales.

RF14.1.3- Modificar Materiales.

RF14.2- Modificar Plantilla de Solicitud de Materiales.

RF14.3- Mostrar planilla de solicitud además de permitir imprimir y mandar por e-mail.

RF14.4- Guardar la cantidad de materiales aprobados de la solicitud.

RF14.5- Mostrar reporte de materiales solicitados contra aceptados, por solicitud y periodo.

RF14.6- Eliminar Plantilla de solicitud de Materiales.

RF15- Gestionar Prenómina de Pago y Aseo.

RF15.1- Crear la planilla de prenómina de pago (dependencia, elaborado por, aprobado por, trabajadores).

RF15.2- Adicionar, Modificar y Eliminar Trabajadores a la Prenómina.

RF15.3- Mostrar planilla de prenómina.

RF15.4- Modificar Plantilla.

RF15.5- Imprimir la planilla de prenómina.

RF15.6- Eliminar Prenómina.

RF16- Gestionar noticias.

RF16.1- Publicar noticias (título, descripción).

RF16.2- Modificar noticias.

RF16.3- Eliminar noticias publicadas.

RF16.4- Mostrar Noticias

RF17- Gestionar Roles

RF17.1- Adicionar rol (nombre, descripción).

RF17.2- Modificar rol.

RF17.3- Eliminar rol.

RF17.4- Mostrar Rol.

2.4 Definición de los requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con la toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. (JACOBSON *et al.* 2000)

A continuación se detallan los requerimientos no funcionales del sistema:

1. **Apariencia o interfaz externa:**
   RNF1.1- Diseño orientado a llamar la atención del usuario y con una navegación sencilla.
   RNF1.2- Construcción de enlaces rápidos o anclas para los documentos muy largos.
2. **Usabilidad:**
   RNF2.1- El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.
3. **Rendimiento:**
   RNF3.1- Tiempos de respuestas no mayor de 2 segundos, al igual que la velocidad de procesamiento de la información.
4. **Soporte:**
   RNF 4.1- Se requiere un servidor de bases de datos con las siguientes características:
   RNF 4.1.1- Soporte para medianos volúmenes de datos y velocidad de procesamiento.
   RNF 4.1.2-Tiempo de respuesta de no más de 3 segundos en accesos concurrentes.

5. **Portabilidad:**

   RNF 5.1- Necesidad de que el sistema sea multiplataformas.

6. **Seguridad:**

   RNF 6.1- Identificar al usuario antes de que pueda realizar cualquier acción sobre el contenido del sistema.

   RNF 6.2- Garantizar que la información sea editada únicamente por quien tiene derecho a editarla.

   RNF 6.3- Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de acceso del usuario activo.

   RNF 6.4- Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

   RNF 6.5- Verificación sobre acciones irreversibles (eliminaciones).

7. **Legales:**

   RNF 7.1- La plataforma escogida para el desarrollo de la aplicación, está basada en la licencia GNU/GPL.

8. **Confiabilidad:**

   RNF 8.1- La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.

9. **Funcionalidad:**

   RNF 9.1- Reducir al mínimo el tiempo en que carga el sistema.

   RNF 9.2- Guardar en caché páginas de contenido para agilizar la navegación del portal.

10. **Software:**

    RNF 10.1- Navegador compatible o superior con Internet Explorer 4, o NetsCape Navegator.

    RNF 10.2- Macromedia Dreamweaver MX 2004.

    RNF 10.3- PostGreSQL 8.1.x

    RNF 10.4- Apache 2.0.x

    RNF 10.5- PHP 5.0.x

    RNF 10.6- NuSphere PHPED

    RNF 10.7- Sistema Operativo Linux.

2.5 Diagrama de Caso de Uso

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar como reacciona una respuesta a eventos que se producen en el mismo. (WIKIMEDIA 2007)

2.5.1 Diagrama de Paquetes

Un paquete es una parte de un modelo. Cada parte del modelo debe pertenecer a un paquete. Pero para ser funcional, la asignación debe seguir un cierto principio Rational, tal como funcionalidad común, implementación relacionada y punto de vista común. Los paquetes ofrecen un mecanismo general para la organización de los modelos/subsistemas agrupando elementos de modelado. Cada paquete corresponde a un submodelo (subsistema) del modelo (sistema). Los paquetes son unidades de organización jerárquica de uso general de los modelos de UML. Por lo que un diagrama de paquetes permite dividir al sistema orientado a objetos organizándolo en subsistemas y detallando sus relaciones.

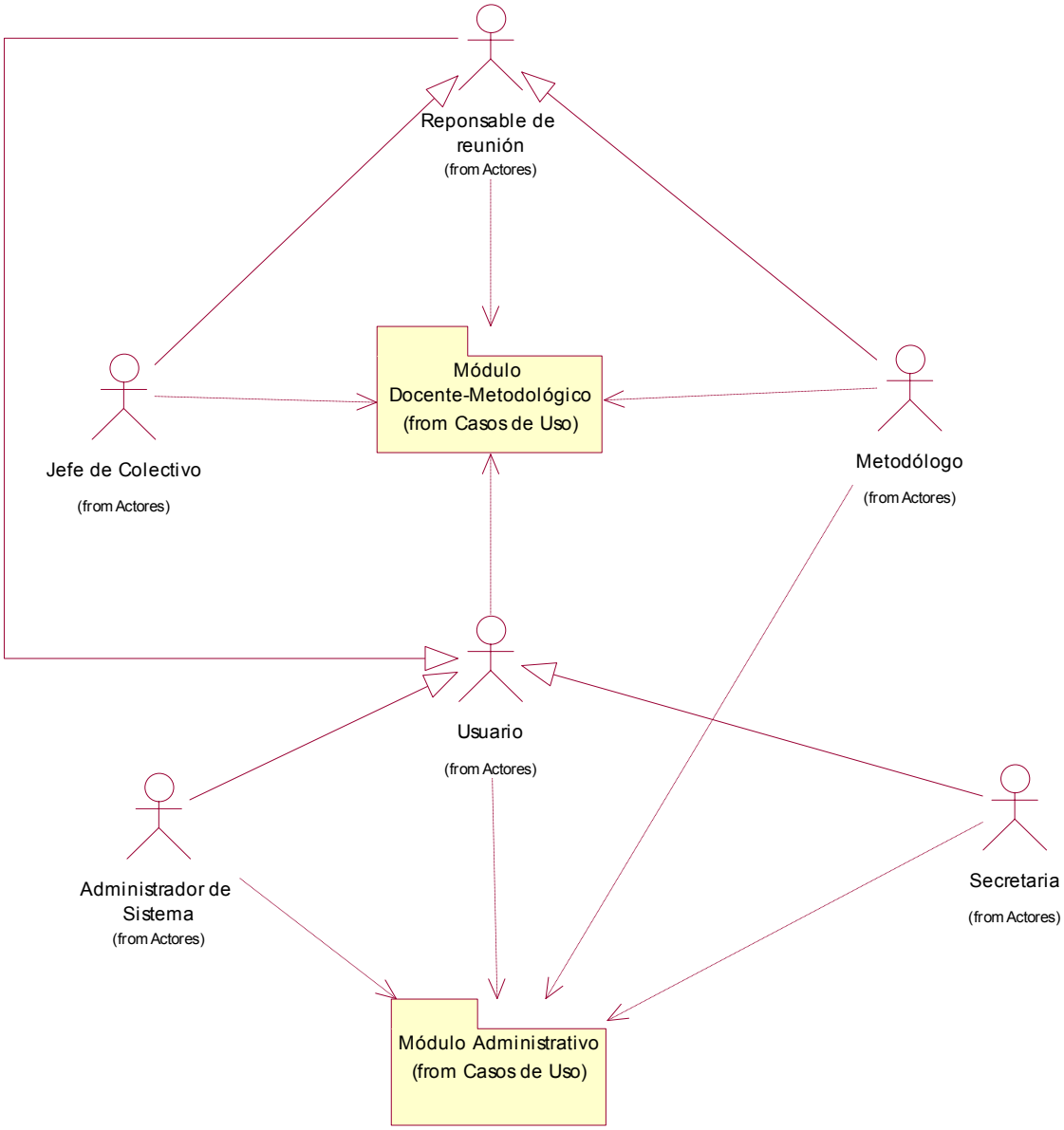


Figura 2. Diagrama de Paquetes

2.5.1.1 Paquete: Módulo Docente-Metodológico

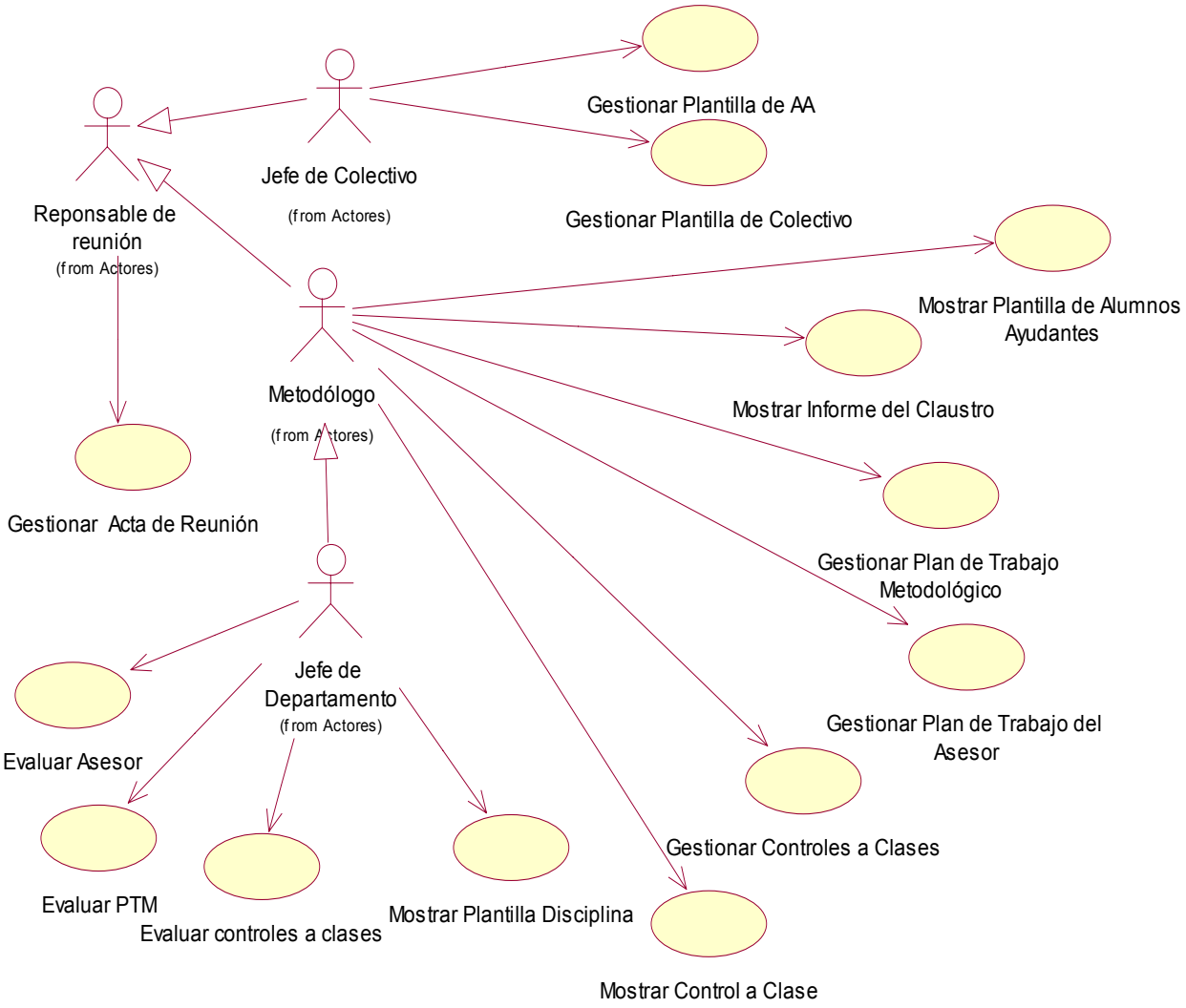


Figura 3. Diagrama de caso de Uso. Módulo Docente-Metodológico.

2.5.1.2 Paquete: Módulo Administrativo

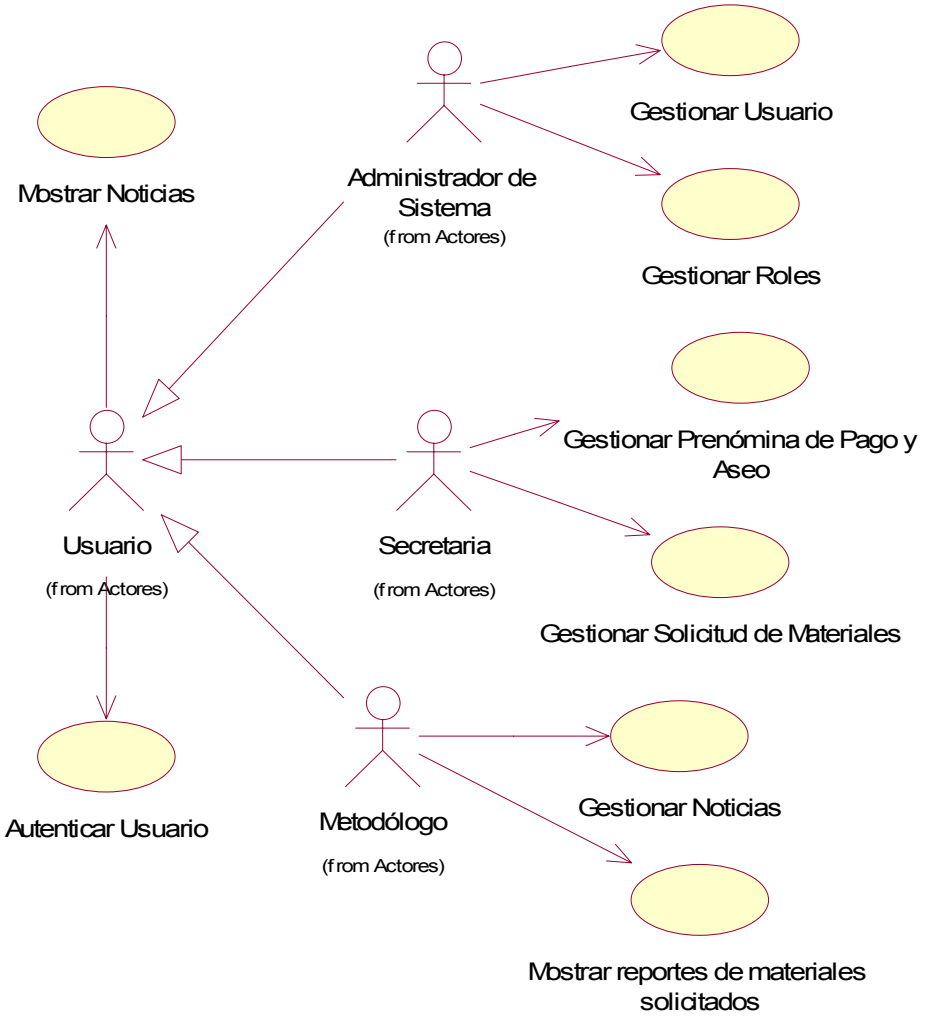


Figura 4. Diagrama de caso de Uso. Módulo Administrativo.

2.6 Descripción de los Casos de Uso

Tabla 2. Gestionar Plan de trabajo Metodológico (PTM)

CU-1	**Gestionar Plan de trabajo Metodológico(PTM)**
Actor	Metodólogo
Resumen	El caso de uso se inicia cuando el Metodólogo, decide crear el PTM seleccionando el curso y luego introduciendo los datos de mayor importancia para la creación de este como: El Diagnóstico del claustro, los objetivos, las acciones y los tipos de esta, Actividades Metodológicas y el tipo de estas. Este caso de uso culmina cuando se tiene creado el PTM se imprime y se envía por correo.
Referencia	RF 1, RF 1.1, RF 1.2, RF1.3, RF1.4, RF1.5, RF1.6, RF1.7, RF1.8, RF1.9, RF1.10, RF1.11, RF1.12, RF1.13, RF1.14.
Precondiciones	Se debe constar con la plantilla de la disciplina confeccionada.
Poscondiciones	El PTM queda confeccionado.

Tabla 3. Evaluar PTM

CU-2	**Evaluar Plan de Trabajo Metodológico**
Actor	Jefe de Departamento
Resumen	El caso de uso se inicia cuando el Jefe de Departamento decide verificar el estado de cumplimiento de las actividades del Plan de Trabajo Metodológico, chequeando resultados de las actividades hasta el momento y emitiendo una observación que justifica el estado asignado.
Referencia	RF2, RF2.1, RF2.2, RF2.3, RF2.4.
Precondiciones	Debe haber pasado la fecha de ejecución de alguna actividad.
Poscondiciones	Cambia el estado de las actividades después de pasado el tiempo de ejecución.

Tabla 4. Gestionar Plan de Trabajo Individual (PTI)

CU-3	**Gestionar Plan de Trabajo Individual (PTI)**
Actor	Metodólogo
Resumen	El caso de uso se inicia cuando el Metodólogo, que puede ser un Asesor o el Jefe de Departamento, consulta el Plan de Trabajo Individual de algún asesor y decide modificarle los objetivos y observaciones. Después de período de confección este no puede ser modificado.
Referencia	RF3, RF3.1, RF3.2, RF3.3, RF3.4, RF3.5, RF3.6.
Precondiciones	Se debe constar con la plantilla de la disciplina confeccionada.
Poscondiciones	El PTI queda confeccionado.

Tabla 5. Evaluar Asesor

CU-4	**Evaluar Asesor**
Actor	Jefe de Departamento
Resumen	El caso de uso se inicia cuando el Jefe de Departamento consulta el cumplimiento de los objetivos del PTI de algún asesor, evaluando el cumplimiento y emitiendo una observación.
Referencia	RF4, RF4.1, RF4.2, RF4.3, RF4.4.
Precondiciones	Debe haber pasado la fecha de ejecución de alguna actividad.
Poscondiciones	Esto puede cambiar el estado de cumplimiento de los objetivos plasmados en la planificación del PTI.

Tabla 6. Gestionar Controles a Clases

CU-5	**Gestionar Controles a Clases.**
Actor	Metodólogo
Resumen	El caso de uso se inicia cuando el Metodólogo agrega una nueva planificación de control a clases, también puede iniciarse al actualizar

	planificaciones después del cumplimiento.
Referencia	RF8, RF8.1, RF8.2, RF8.3, RF8.4, RF8.5, RF8.6, RF8.7, RF8.8.
Precondiciones	Necesita que la plantilla del la disciplina esté confeccionada
Poscondiciones	Quedan planificados los controles a clases

Tabla 7. Evaluar Controles a Clases

CU-6	**Evaluar Control a Clases.**
Actor	Jefe de Departamento
Resumen	El caso de uso se inicia cuando el Jefe de Departamento consulta los controles realizados a clases y evalúa los mismos.
Referencia	RF8, RF8.1, RF8.2, RF8.3, RF8.4, RF8.5, RF8.6, RF8.7, RF8.8.
Precondiciones	Se debe haber planificado un control a clases.
Poscondiciones	El control queda evaluado

Tabla 8. Gestionar Actas de Reunión

CU-7	**Gestionar Actas de Reunión**
Actor	Responsable de Reunión.
Resumen	El caso de uso se inicia cuando el Responsable de Reunión decide subir el Acta para almacenarla, para luego poder realizarle si desea modificaciones.
Referencia	RF7, RF7.1, RF7.2, RF7.3, RF7.4.
Precondiciones	Debe estar confeccionada la plantilla por colectivo.
Poscondiciones	Acta de reunión subida.

Tabla 9. Gestionar plantilla de colectivo.

CU-8	**Gestionar plantilla de colectivo.**

Actor	Jefe de Colectivo.
Resumen	El caso de uso se inicia cuando el Jefe de Colectivo registra los profesores del colectivo, definiendo sus datos personales si no está registrado, profesionales y docentes.
Referencia	RF5, RF5.1, RF5.2, RF5.3, RF5.4
Precondiciones	El Jefe de Colectivo debe haber sido previamente notificado de su rol en el sistema.
Poscondiciones	Plantilla del colectivo confeccionada.

Tabla 10. Gestionar plantilla de Alumnos Ayudantes.

CU-9	**Gestionar plantilla de Alumnos Ayudantes.**
Actor	Jefe de Colectivo.
Resumen	El caso de uso se inicia cuando el Jefe de Colectivo registra los Alumnos Ayudantes, definiendo todos sus datos.
Referencia	RF6, RF6.1, RF6.2, RF6.3, RF6.4
Precondiciones	El Jefe de Colectivo debe haber sido previamente notificado de su rol en el sistema.
Poscondiciones	Plantilla de alumnos ayudantes del colectivo confeccionada.

Tabla 11. Mostrar Informe del Claustro

CU-10	**Mostrar Informe del Claustro**
Actor	Metodólogo
Resumen	El caso de uso se inicia cuando el Metodólogo decide conocer el estado del claustro, cuantos con categoría docente, cuantos con categoría científica, el año promedio de experiencia en la asignatura, así como la cantidad de Alumnos Ayudantes.

Referencia	RF9, RF9.1.
Precondiciones	Tiene que estar creada la plantilla del colectivo.
Poscondiciones	-

Tabla 12. Mostrar Control a Clases

CU-11	**Mostrar Control a Clases**
Actor	Metodólogo
Resumen	El caso de uso se inicia cuando el Metodólogo decide mostrar la cantidad de controles a clases realizados por mes y año y además la cantidad de ellos bien, regular y mal.
Referencia	RF8, RF8.1, RF8.2, RF8.3, RF8.4, RF8.5, RF8.6, RF8.7, RF8.8.
Precondiciones	Tiene que haberse realizado el control a clases.
Poscondiciones	-

Tabla 13. Mostrar Plantilla de la Disciplina

CU-12	**Mostrar Plantilla de la Disciplina**
Actor	Jefe de Departamento
Resumen	El caso de uso se inicia cuando el Jefe de Departamento desea mostrar seleccionando la asignatura todos los profesores de cada facultad con sus datos correspondientes.
Referencia	RF10, RF10.1.
Precondiciones	Tiene que estar creada la plantilla del colectivo.
Poscondiciones	-

Tabla 14. Mostrar Plantilla de Alumnos Ayudantes

CU-12	**Mostrar Plantilla de Alumnos Ayudantes**
Actor	Metodólogo
Resumen	El caso de uso se inicia cuando el Metodólogo desea mostrar

	seleccionando la asignatura todos los alumnos ayudantes de cada facultad con sus datos correspondientes.
Referencia	RF11, RF11.1.
Precondiciones	Tiene que estar creada la plantilla de AA.
Poscondiciones	-

Tabla 15. Gestionar Usuarios

CU-13	**Gestionar Usuarios**
Actor	Administrador
Resumen	El caso de uso inicia cuando el Administrador decide Crear Nuevo Usuario, Modificar un Usuario, Eliminar Usuario, Asignar Privilegios o Mostrar Usuarios para controlar los usuarios que pueden acceder al sistema y a qué parte del sistema lo hacen.
Referencia	RF13, RF13.1, RF13.2, RF13.3, RF13.4, RF13.5.
Precondiciones	El administrador debe estar autenticado en el sistema
Poscondiciones	Quedan creados los usuarios con sus roles correspondientes.

Tabla 16. Gestionar Prenómina de Pago y Aseo.

CU-14	**Gestionar Prenómina de Pago y Aseo.**
Actor	Secretaria
Resumen	El caso de uso se inicia cuando la secretaria confecciona la prenómina de pago y aseo, a la que puede modificar los trabajadores que la componen así como imprimir el modelo, enviarlo por correo, y guardarlo en el sistema.
Referencia	RF15, RF15.1, RF15.2, RF15.3, RF15.4, RF15.5, RF15.6.
Precondiciones	Debe estar confeccionada la plantilla del departamento.
Poscondiciones	Queda archivada la prenómina de pago y aseo.

Tabla 17. Gestionar Solicitud de Materiales.

CU-15	**Gestionar Solicitud de Materiales.**
Actor	Secretaria
Resumen	El caso de uso se inicia cuando la secretaria comienza a crear la plantilla de solicitud de materiales llevando dicha plantilla a ATM. Se elabora una justificación por cada material que se solicite.
Referencia	RF14, RF14.1, RF14.1.1, RF14.1.2, RF14.1.3, RF14.2, RF14.3, RF14.4, RF14.5, RF14.6.
Precondiciones	**-**
Poscondiciones	Solicitud creada

Tabla 18. Gestionar noticias

CU-16	**Gestionar noticias**
Actor	Metodólogo
Resumen	Este caso de uso se inicia cuando uno de los actores concebido como Metodólogo, edita o publica noticias en el sistema, esto incluye actualización y eliminación. Permite además mostrar las noticias publicadas al mes correspondiente.
Referencia	RF16, RF16.1, RF16.2, RF16.3, RF16.4.
Precondiciones	**-**
Poscondiciones	-

Tabla 19. Gestionar Rol

CU-17	**Gestionar Roles**
Actor	Administrador
Resumen	Este caso de uso se inicia cuando el administrador decide crear los diferentes roles para asignarlo a cada usuario.
Referencia	RF17, RF17.1, RF17.2, RF17.3, RF17.4

Precondiciones	-
Poscondiciones	-

Tabla 20. Autenticar usuario

CU-18	**Autenticar usuario**
Actor	Usuario
Resumen	El caso de uso se inicia cuando el usuario solicita entrar al sistema, introduciendo nombre y contraseña, verificando sus credenciales en el dominio uci.cu para permitir el acceso. En caso de que no pueda autentificar en el dominio entonces los intenta en el sistema, consultando su perfil de usuario.
Referencia	RF12, RF12.1, RF12.2, RF12.3, RF12.4.
Precondiciones	El usuario debe haber sido adicionado previamente al sistema.
Poscondiciones	

Tabla 21. Mostrar Reportes de Materiales solicitados.

CU-20	**Mostrar Reportes de Materiales solicitados.**
Actor	Metodólogo
Resumen	Este caso de uso inicia cuando el metodólogo quiere saber cuantos materiales y de que tipo fueron enviados según la solicitud realizada y si dicha solicitud fue rechazada o aceptada por completa.
Referencia	RF14, RF14.1, RF14.1.1, RF14.1.2, RF14.1.3, RF14.2, RF14.3, RF14.4, RF14.5, RF14.6.
Precondiciones	Tiene que estar la solicitud de materiales hecha.
Poscondiciones	-

Tabla 22. Mostrar Noticias.

CU-21	**Mostrar Noticias.**

Actor	Usuario
Resumen	Este caso de uso inicia cuando el usuario quiere publicar la noticia, esta cuando es publicada aparece en la página principal.
Referencia	RF16, RF16.1, RF16.2, RF16.3, RF16.4.
Precondiciones	-
Poscondiciones	-

2.7 Conclusiones

En este Capítulo se ha realizado una breve descripción de los actores del sistema y sus responsabilidades, refinado los requerimientos funcionales, los cuales son de gran importancia para el desarrollo del sistema, ya que constituyen la funcionalidad del mismo, así como presentando el diagrama de caso de uso por módulo. También se realizó un refinamiento de los requerimientos no funcionales, los que son de gran importancia ya que son propiedades o cualidades que el sistema debe cumplir y además se detallaron los casos de usos arquitectónicamente significativos para la implementación del sistema propuesto.

Capítulo 3

3 CAPÍTULO3: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En el presente capítulo se realiza el diseño de la propuesta de solución, seleccionando una arquitectura y modelándose los artefactos que contribuyen al desarrollo de aplicaciones Web, con el uso de Frameworks Manipuladores de Contenidos flexibles. Se elabora el modelo de datos adecuado. Se analiza como va a estar distribuido el sistema.

3.2 Descripción de la arquitectura

Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software de un sistema de información. La Arquitectura de Software establece los fundamentos para trabajar en una línea común que permite alcanzar los objetivos y necesidades del sistema de información. (JACOBSON *et al.* 2000)

3.2.1 Arquitectura de 4 Capas

El Sistema de Gestión Metodológica Central ha sido desarrollado sobre la arquitectura de 4 capas o niveles. Este Patrón define cómo organizar el modelo de diseño en capas. Los componentes de una capa solo pueden hacer referencia a componentes en capas inmediatamente inferiores, esto simplifica la comprensión y la organización de sistemas complejos.

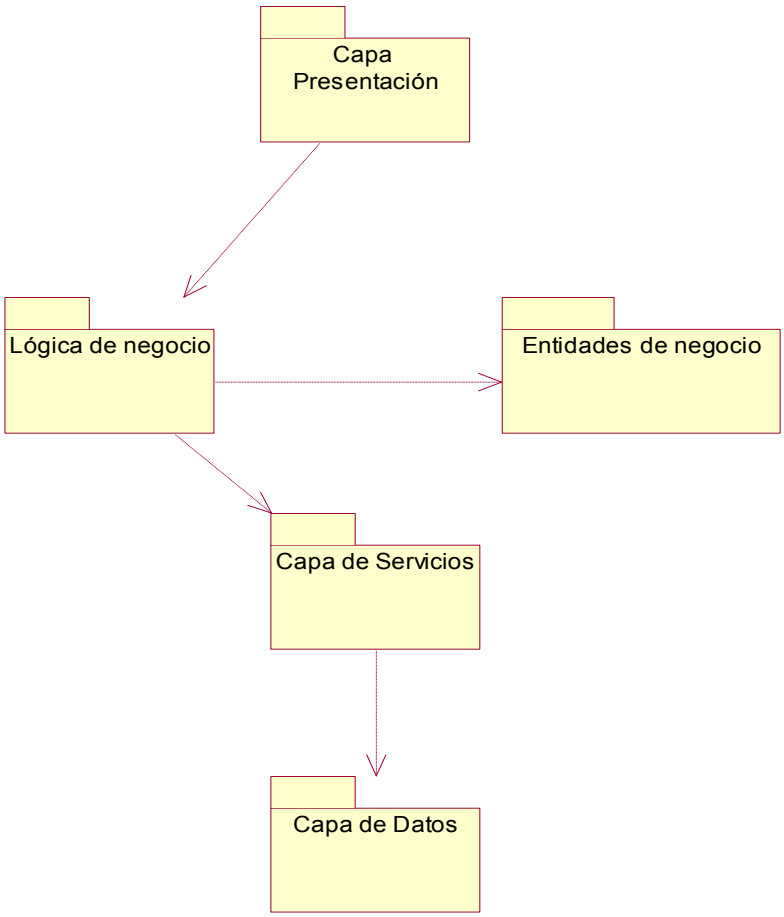


Figura 5. Arquitectura de 4 Capas

Esta arquitectura se descompone en las siguientes partes:

Capa de presentación: En esta capa se almacenan todas las clases interfaces de la aplicación, las cuales son las que manejan la interacción del usuario con la misma y cómo son gestionadas las demandas de dicho usuario.

Capa Lógica de negocio: Esta capa se divide en dos partes está la lógica de negocio y las entidades de negocio, la primera contiene todas las clases controladoras de negocio que son la que modelan las reglas de negocio de la aplicación, y la última contiene las clases entidades de negocio.

Capa de Servicios: En esta capa se encuentran todas las clases de acceso a datos, de configuración y de seguridad.

Capa de datos: En esta capa se encuentra la Base de Datos Física.

3.3 Patrones de Diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, no es más que una solución estándar para un problema común de programación, es una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.(PRESSMAN 2005)

3.3.1 Patrón Fábrica Pura

La fabricación pura se da en las clases que no representan un ente u objeto real del dominio del problema, si no que se ha creado intencionadamente para disminuir el acoplamiento, aumentar la cohesión y/o potenciar la reutilización del código. Es decir que es una clase "inventada" o que no existe en el problema como tal, pero que añadiéndola se logra mejorar estructuralmente el sistema. Generalmente se considera que la fabricación es parte de la capa de servicios orientada a objetos de alto nivel en una arquitectura.

Este patrón brinda soporte a una Alta Cohesión porque las responsabilidades se dividen en una clase de granularidad fina que se centra exclusivamente en un conjunto muy específico de tareas a fines. Además puede aumentar el potencial de reutilización debido a la presencia de clases de Fabricación Pura de granularidad fina, cuyas responsabilidades pueden utilizarse en otras aplicaciones. (LARMAN 2004)

En el desarrollo de este sistema se utiliza dicho patrón para proporcionar una mejor estructura en la programación, producto a esto se crean clases fábricas por cada una de las entidades, las cuales permiten diferentes operaciones y la conexión con la base de datos.

3.4 Modelo de diseño

En la fase de diseño se modela el sistema de manera que soporte todos los requerimientos, tanto funcionales como no funcionales. Uno de sus propósitos fundamentales es: “Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes capturando los requerimientos o subsistemas individuales, interfaces y clases”, se dice que el diseño es un esquema a la implementación. (BARRIENTOS ENRÍQUEZ 2005)

La esencia de esta fase es la elaboración de diagramas de interacción, que muestran gráficamente como los objetos se comunican entre ellos a fin de cumplir con los requerimientos. Estos diagramas permiten la realización de los diagramas de clases del diseño, los cuales resumen la definición de las clases que se pueden implementar en el software.

3.4.1 Diagrama de clases del diseño

El diagrama de clases para las Aplicaciones Web, difiere un poco del resto de las aplicaciones que estamos acostumbrados a construir, puesto que en ellas son más importantes la modelación de la lógica y estado del negocio que los detalles de presentación, para esto se utilizarán los estereotipos Web. Para obtener un nivel correcto de abstracción y detalle que nos permita obtener un resultado final, es mejor modelar los artefactos del sistema, es decir, modelar las páginas, los enlaces entre estas, así como el contenido dinámico de las mismas, una vez que estén en el navegador del cliente; estos son los artefactos que se necesita modelar para la implementación del producto final.

De acuerdo a la forma en que se ha organizado el contenido del trabajo, se considera que se debe presentar solamente aquellos Casos de Uso más importantes para el Módulo Docente-Metodológico y para el Administrativo. El resto de las realizaciones de los casos de usos se encuentran en la documentación del expediente de proyecto y en el fichero del Rational Rose (DDCactual.mdl).

Para el desarrollo de la aplicación se utilizaron diferentes clases como:

- La clase Collections: esta se utiliza para tratar colecciones de objetos, principalmente para cuando se hace una selección de datos de la Base de Datos, estos se cargan en una lista de objetos para trabajar con ellos.
- Las clases Factory: son las que se encargan de hacer todas las operaciones en las base de datos, para separar la lógica del negocio de los datos. Por cada entidad se ha creado una factory.
- La clase de seguridad: se utiliza para la seguridad del sistema.
- La clase Configuración: se encarga de recoger todos los parámetros de configuración de la aplicación y los centraliza en un único objeto que luego es instanciado como objeto global. De este modo cualquier objeto de la aplicación puede consultar sin demasiados problemas cual es

la configuración que le corresponde. Utilizando esta clase se pueden modificar algunos parámetros en tiempo de ejecución.

- La clase PHP Data Objects (PDO): es una clase predefinida de PHP, que define una interfaz ligera y consistente para acceder a bases de datos en PHP. Cada driver de bases de datos que implementa la interfaz PDO puede exponer características específicas de la base de datos como funciones de extensión regulares. Esta presenta una capa de abstracción de datos.

3.4.2 Módulo Docente-Metodológico

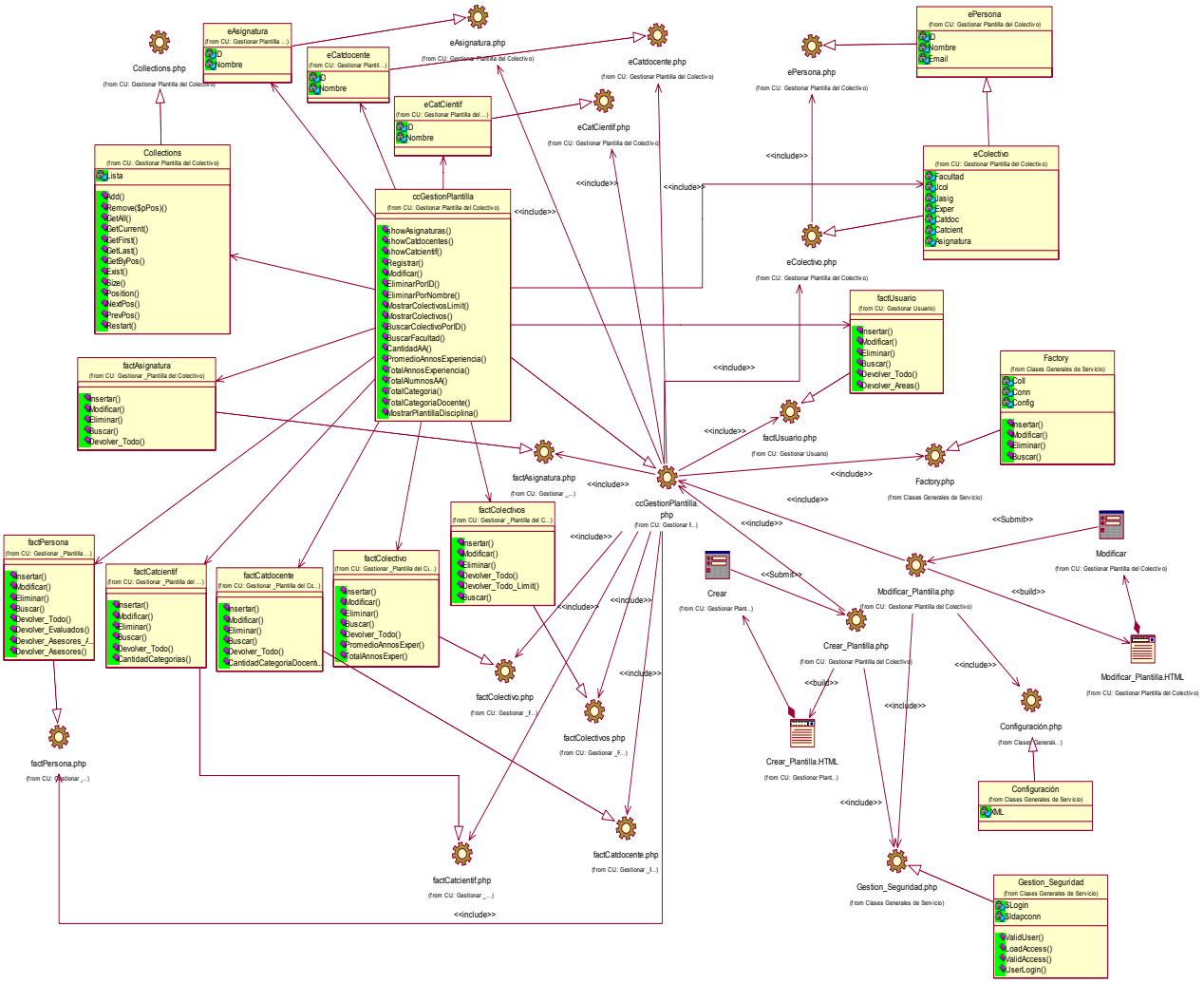


Figura 6. Diagrama de Clases: Gestionar Plantilla del Colectivo

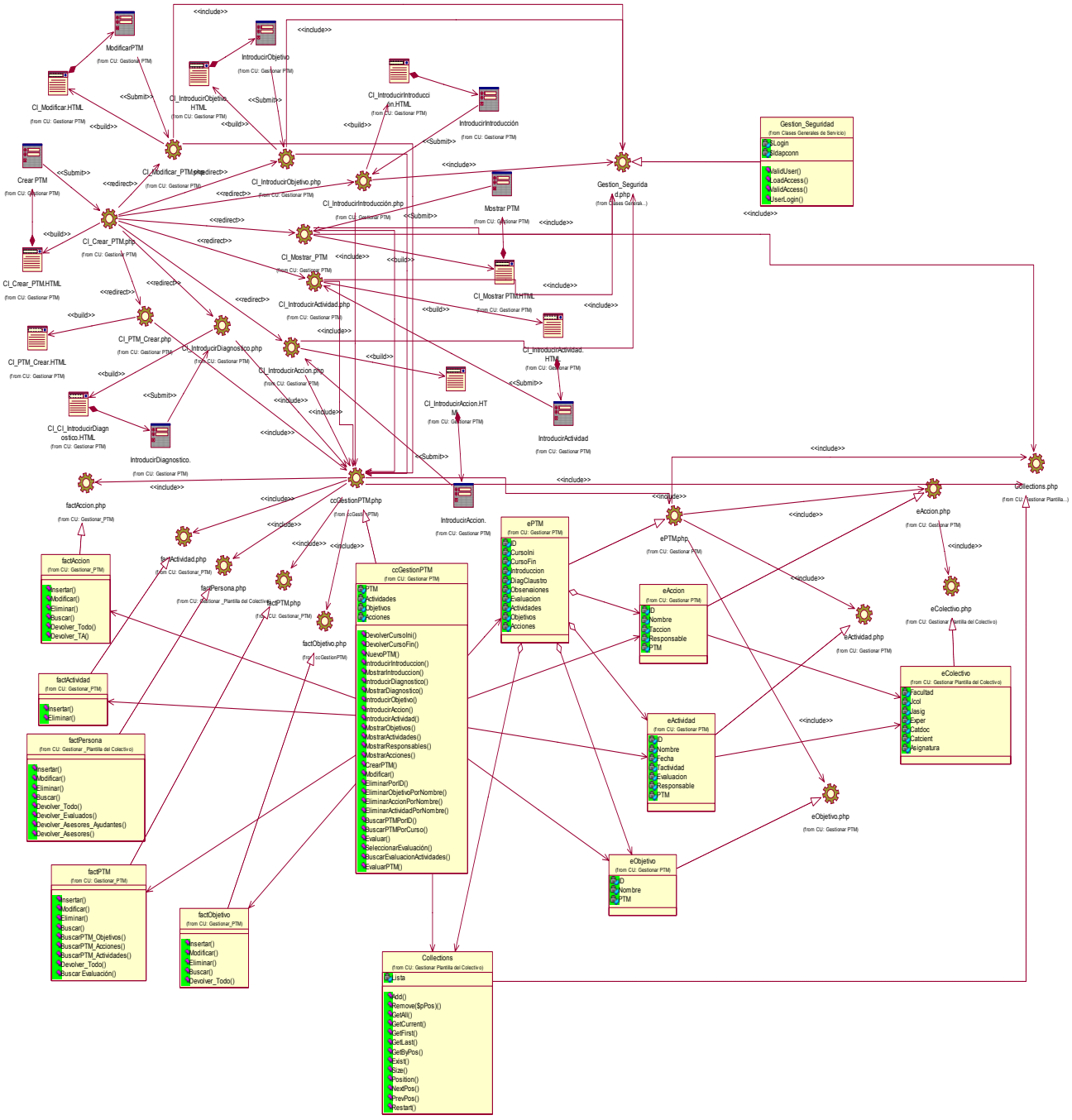


Figura 7. Diagrama de Clases: Gestionar Plan de Trabajo Metodológico

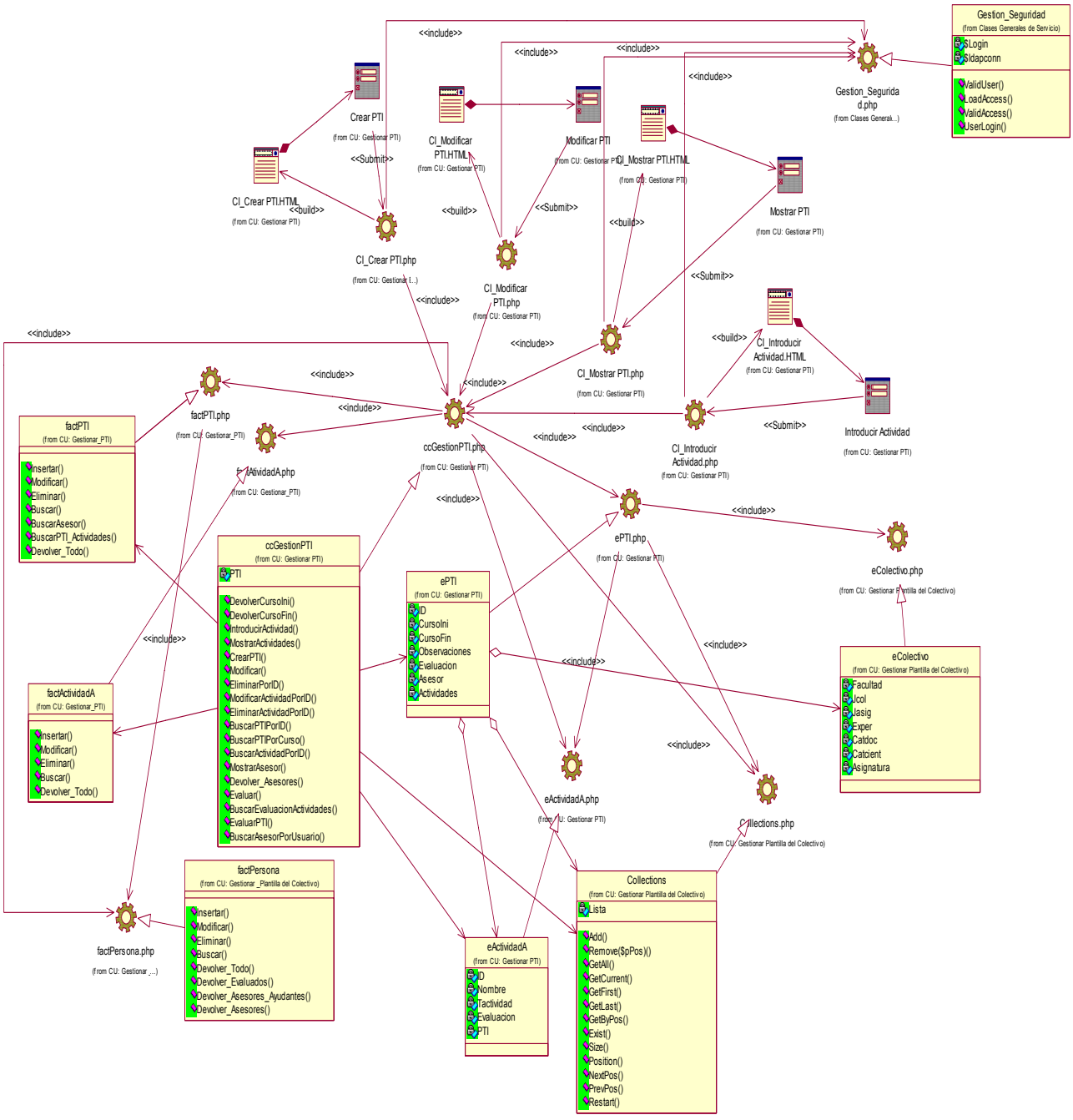


Figura 8. Diagrama de Clases: Gestionar Plan de Trabajo Individual

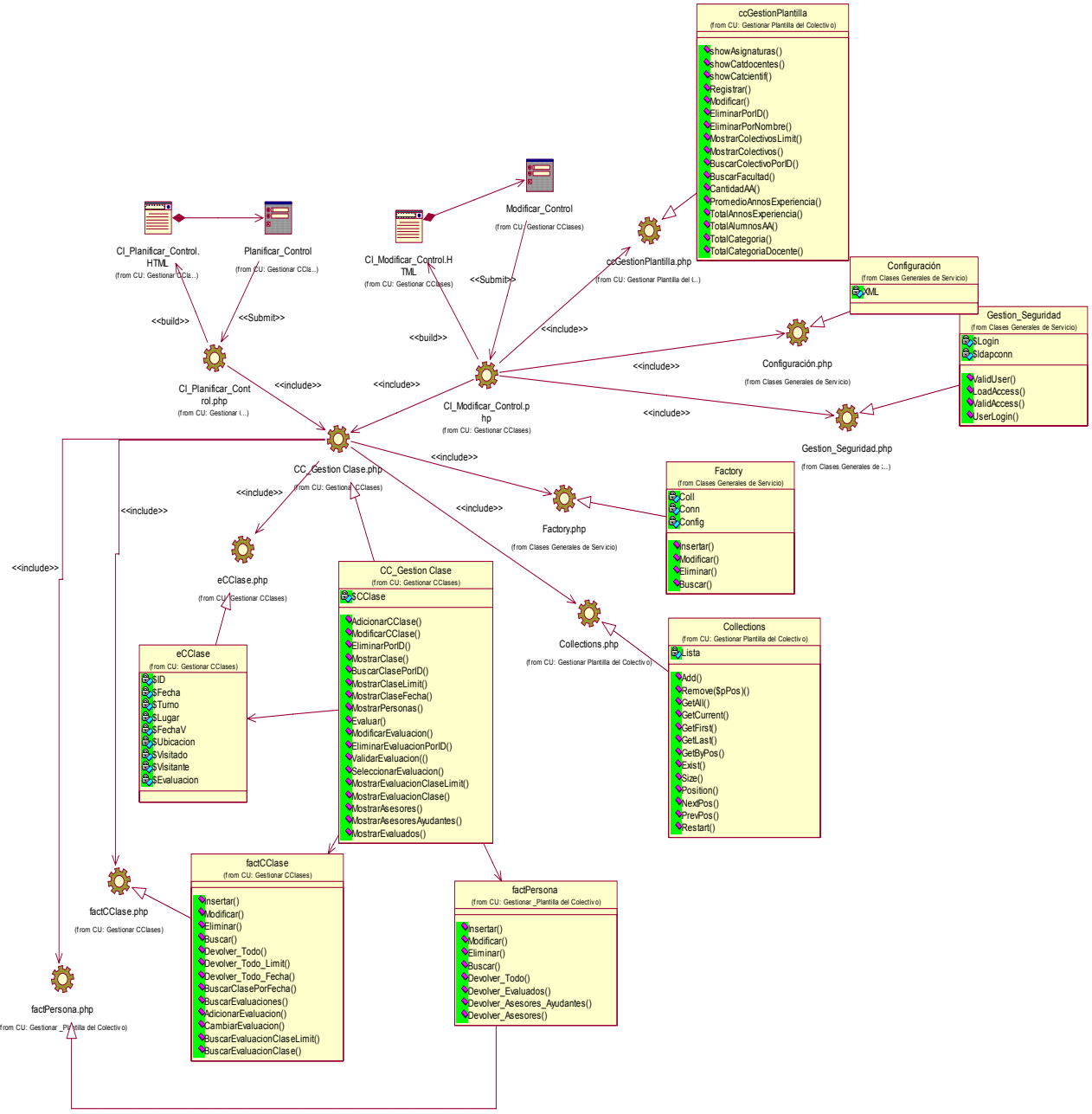


Figura 9. Diagrama de Clases: Gestionar Controles a Clases

3.4.3 Módulo Administrativo

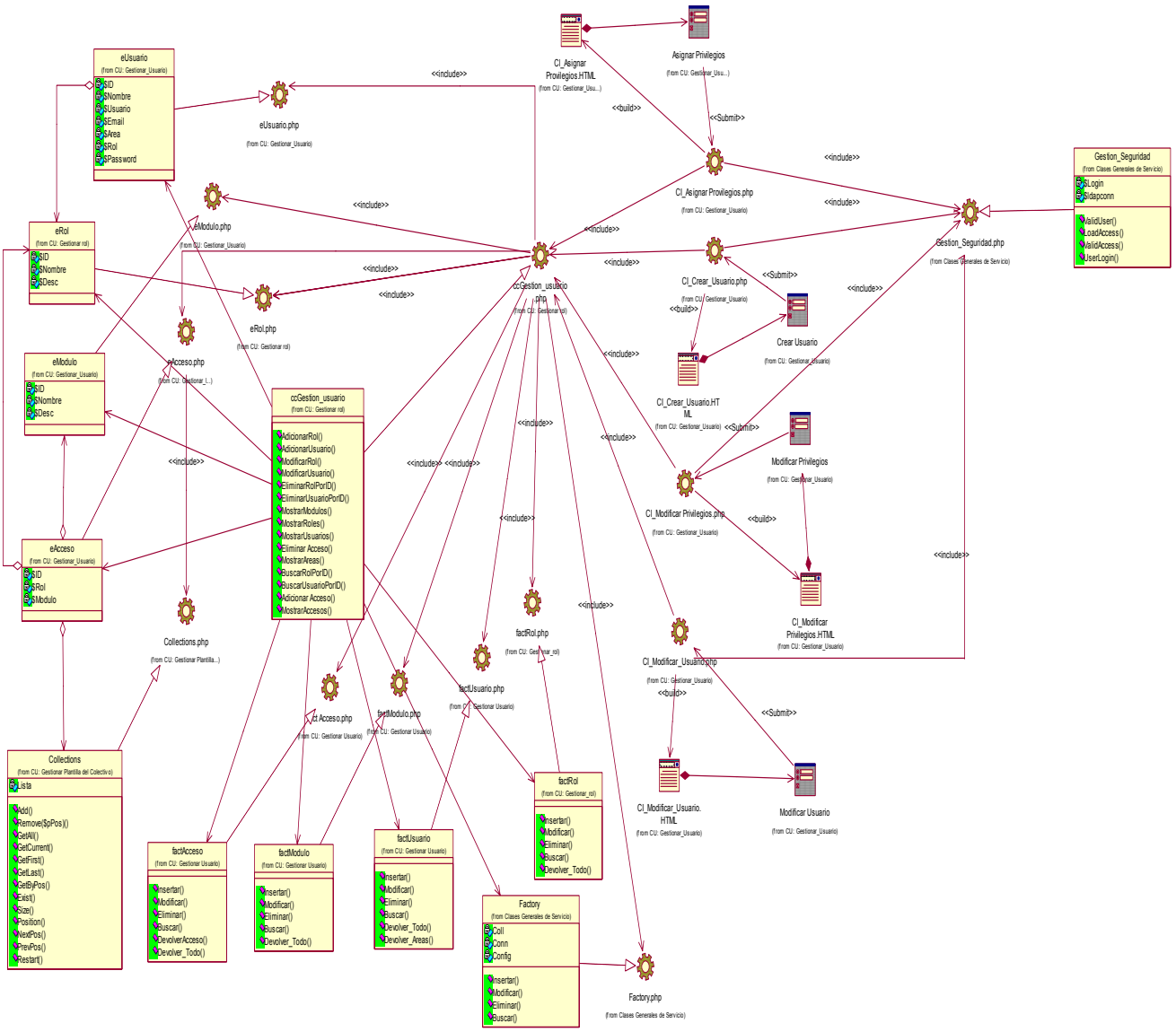


Figura 10. Diagrama de Clases: Gestionar Usuarios

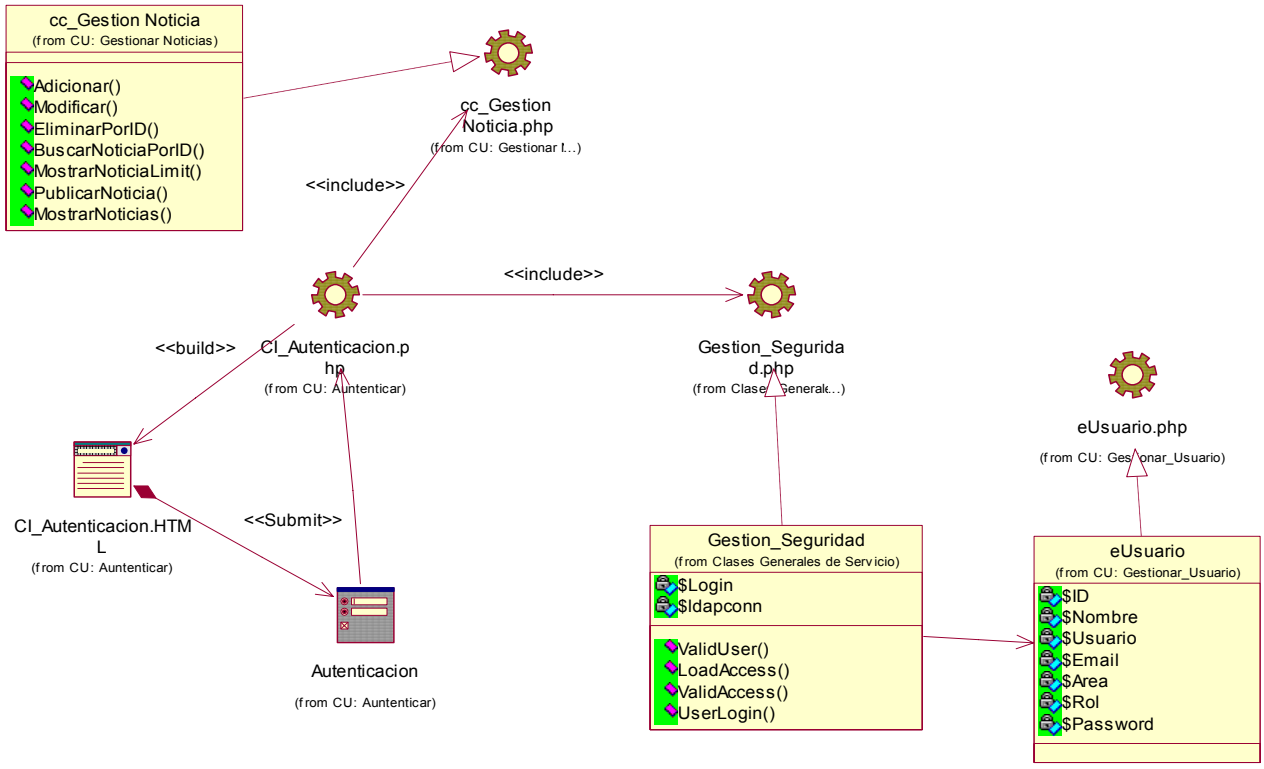


Figura 11. Diagrama de Clases: Autenticar

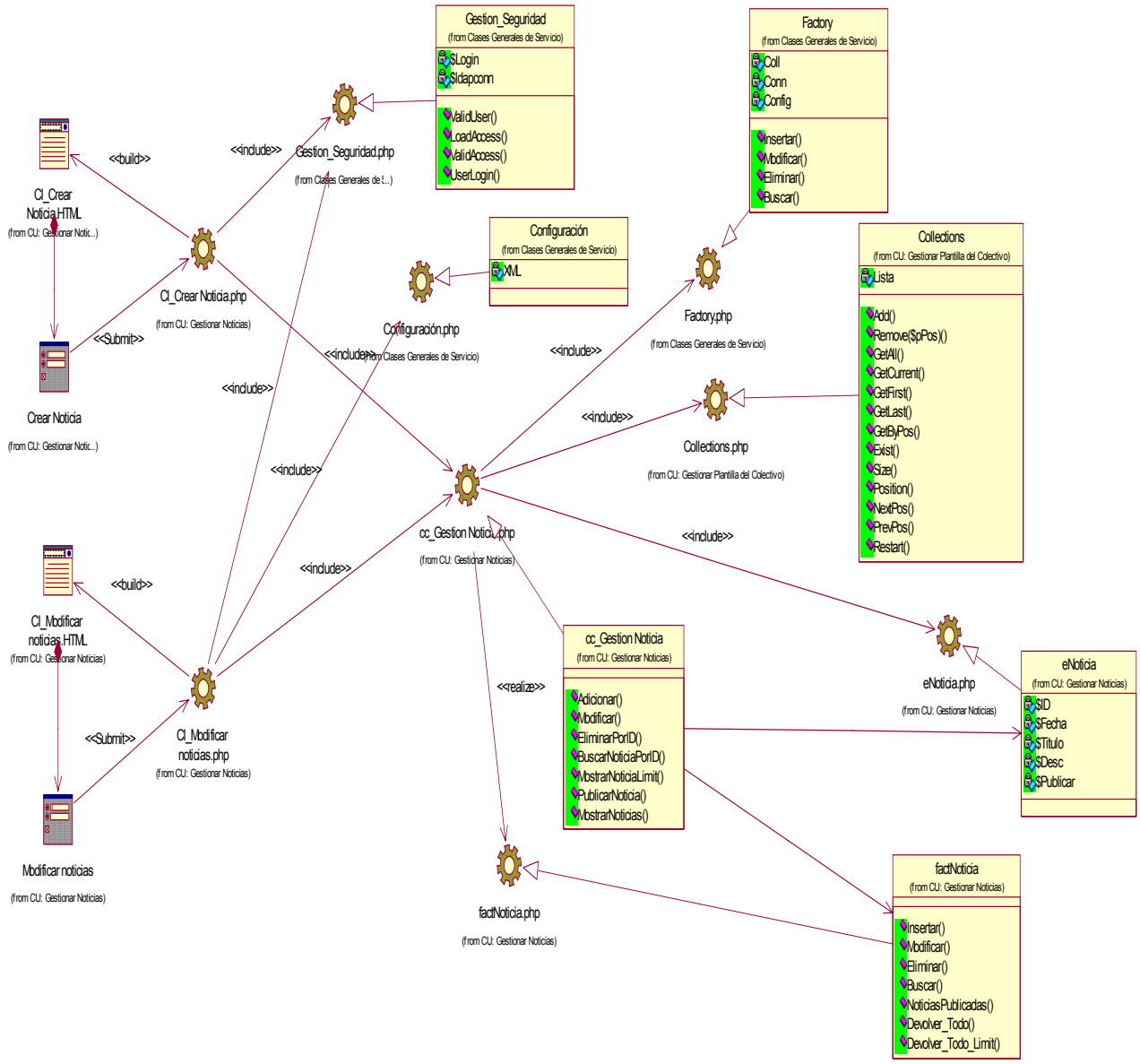


Figura 12. Diagrama de Clases: Gestionar Noticias

3.4.4 Paquete Servicios

3.4.4.1 Módulo Docente-Metodológico

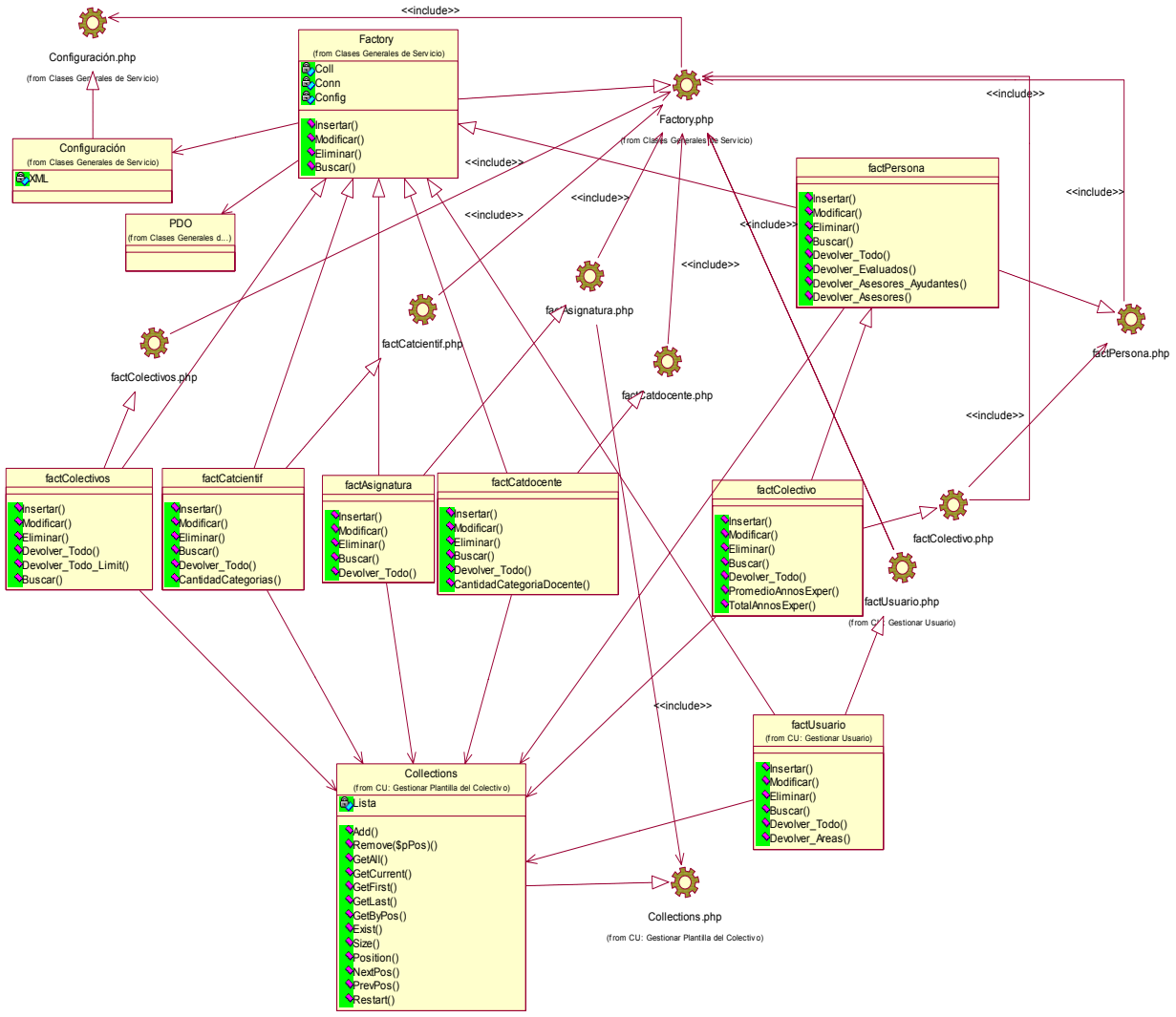


Figura 13. Diagrama de Clases Paquete de servicios: Gestionar Plantilla del Colectivo

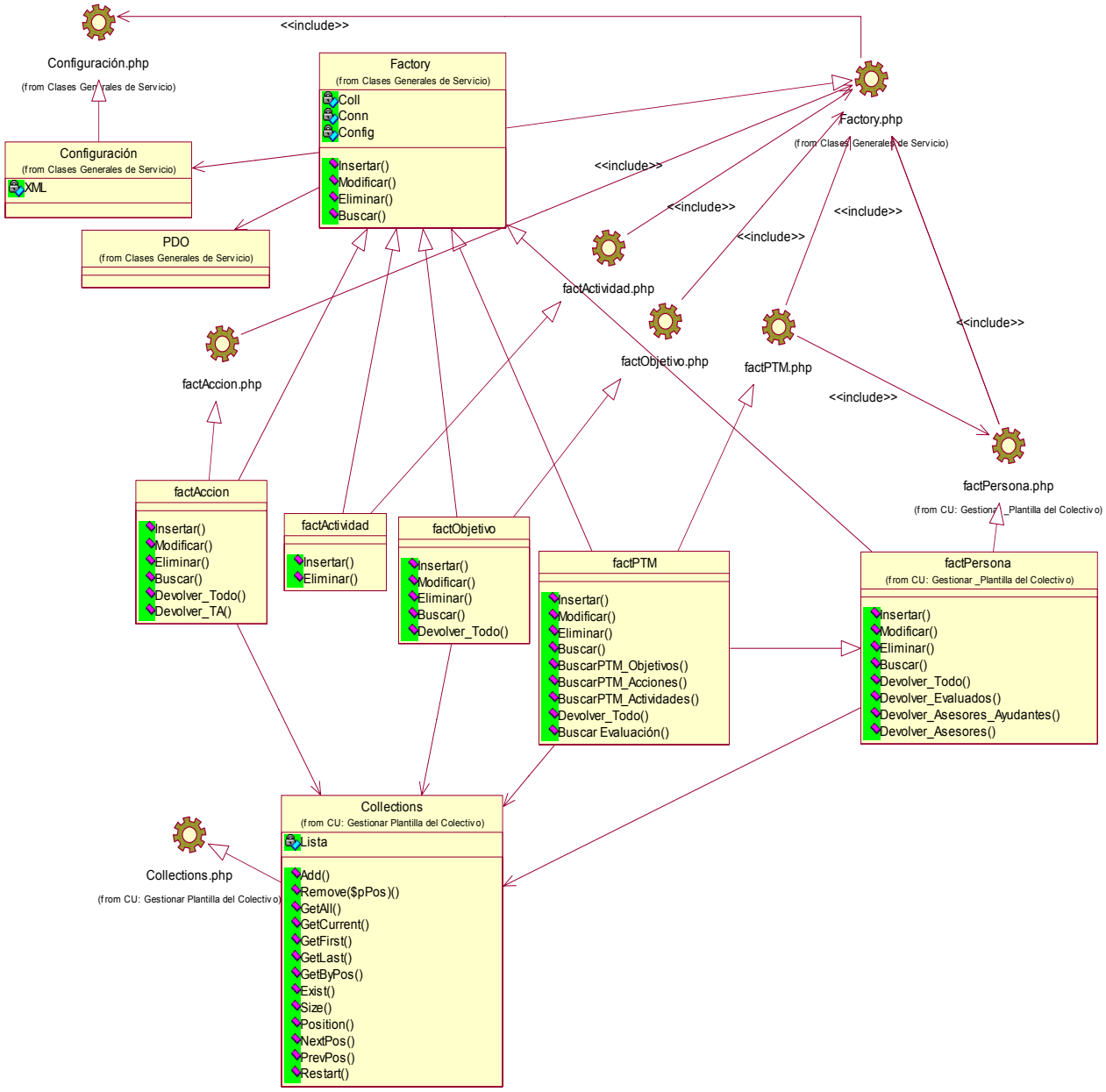


Figura 14. Diagrama de Clases Paquete de servicios: Gestionar Plan de Trabajo Metodológico

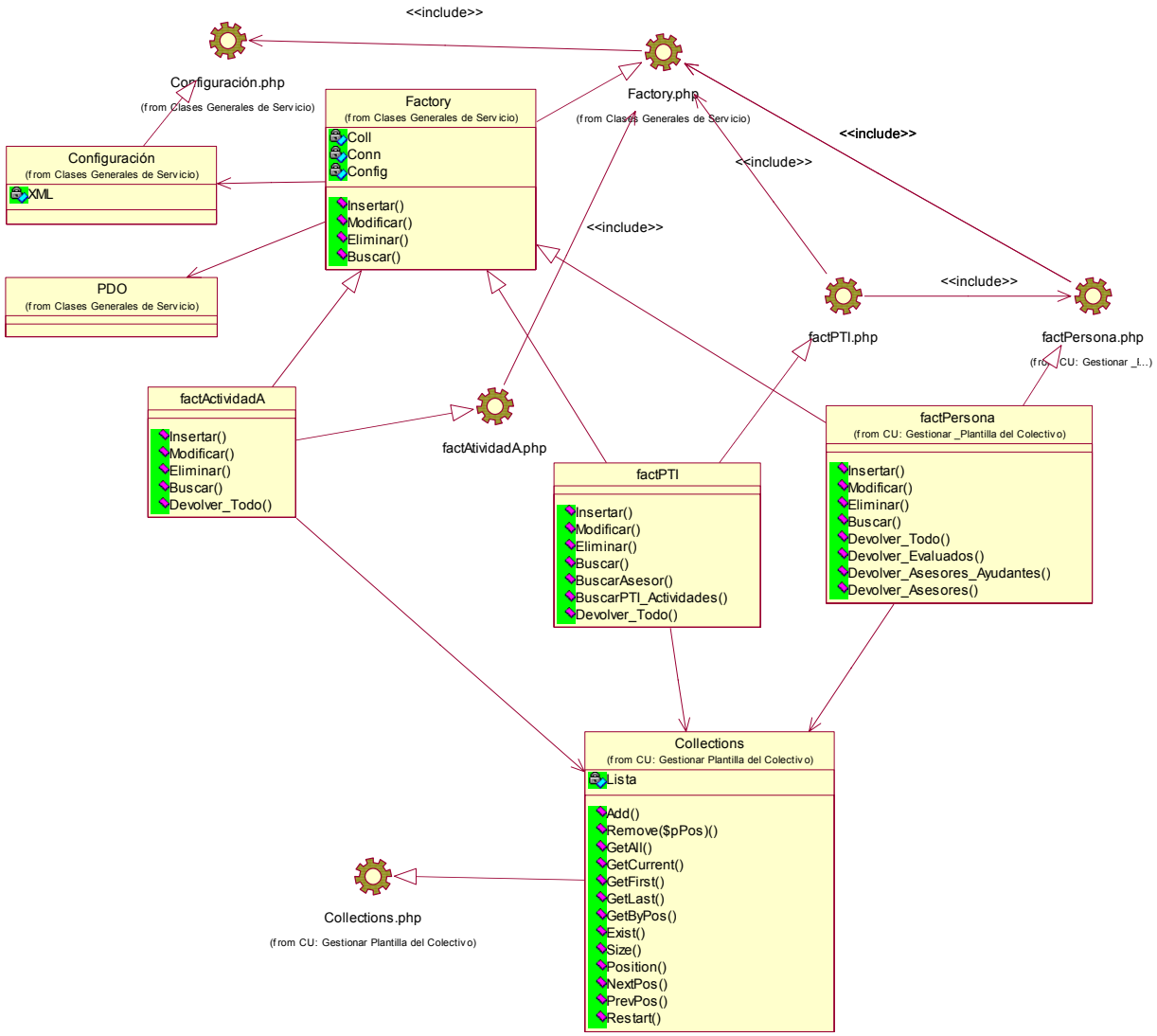


Figura 15. Diagrama de Clases Paquete de servicios: Gestionar Plan de trabajo Individual del Asesor

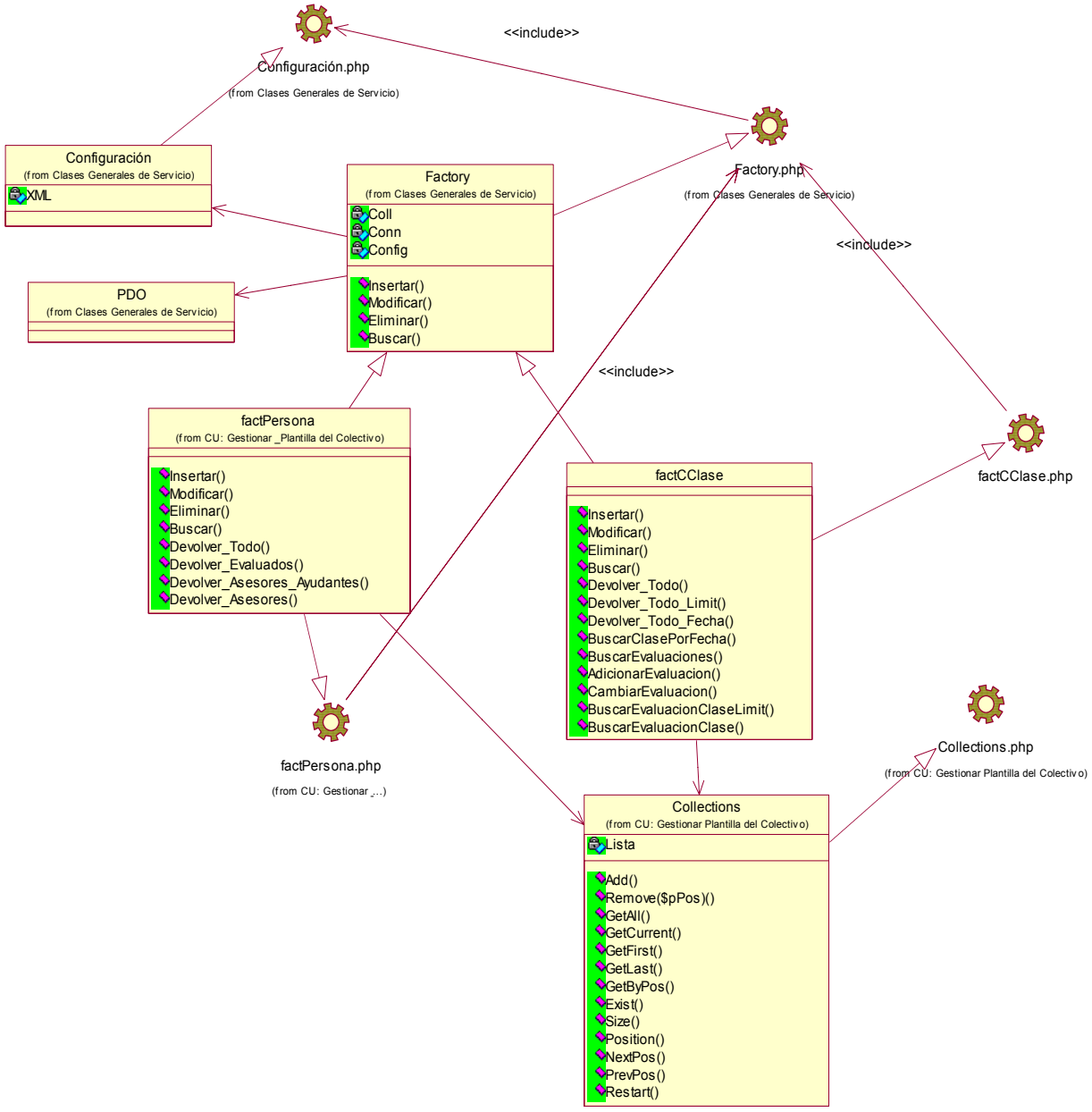


Figura16. Diagrama de Clases Paquete de servicios: Gestionar Controles a Clases

3.4.4.2 Módulo Administrativo

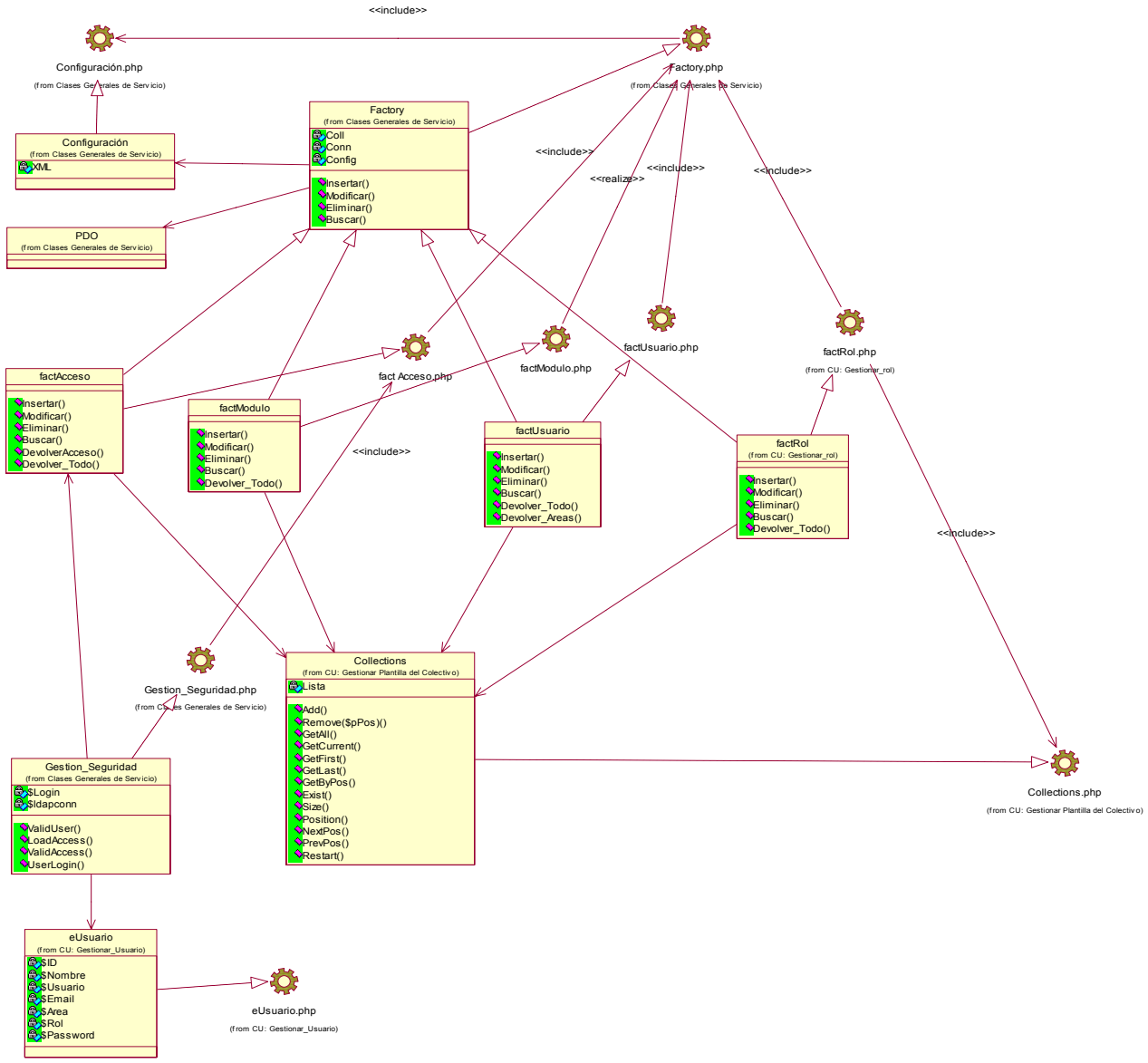


Figura 17. Diagrama de Clases Paquete de servicios: Gestionar Usuarios

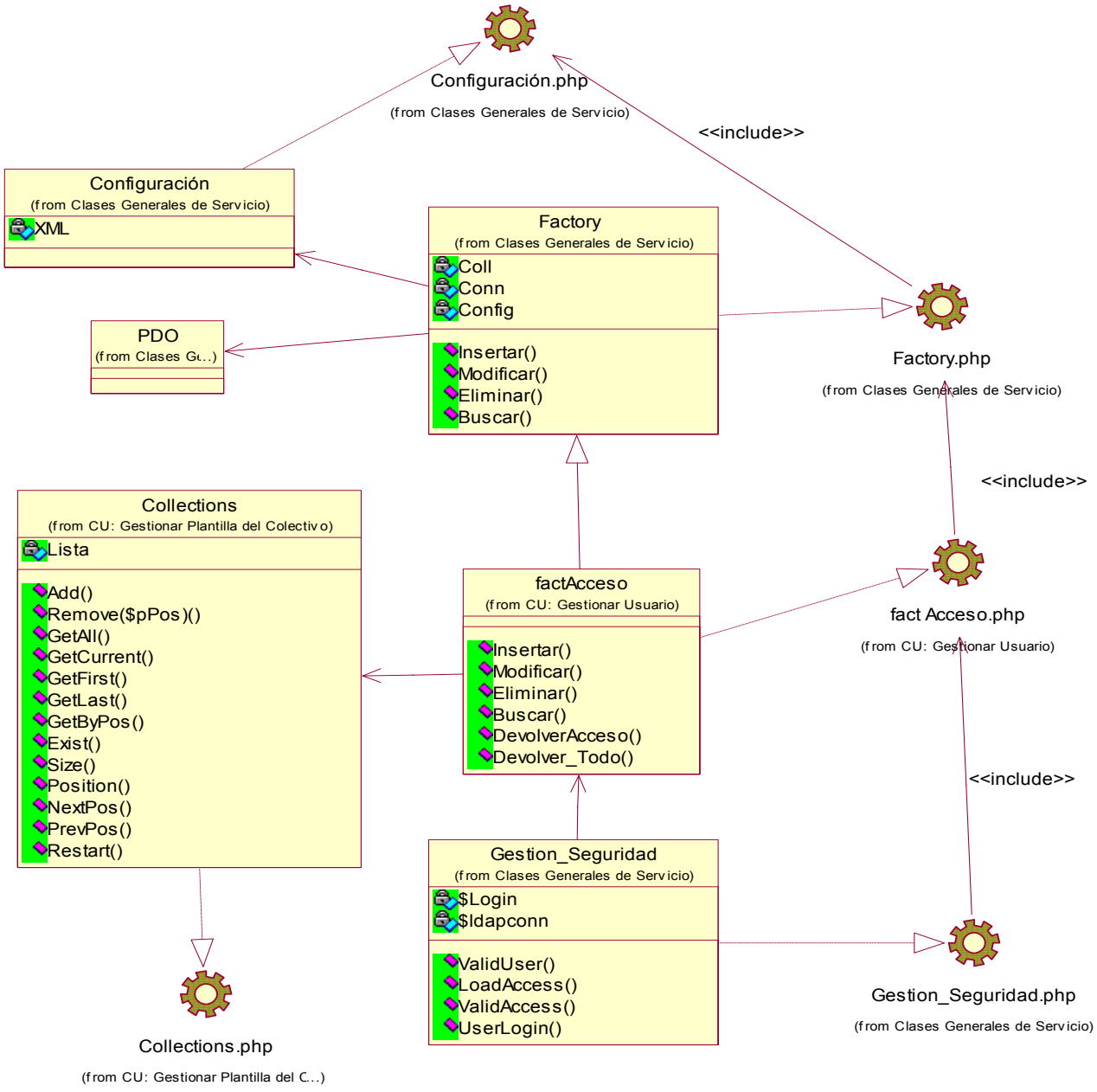


Figura 18. Diagrama de Clases Paquete de servicios: Autenticar

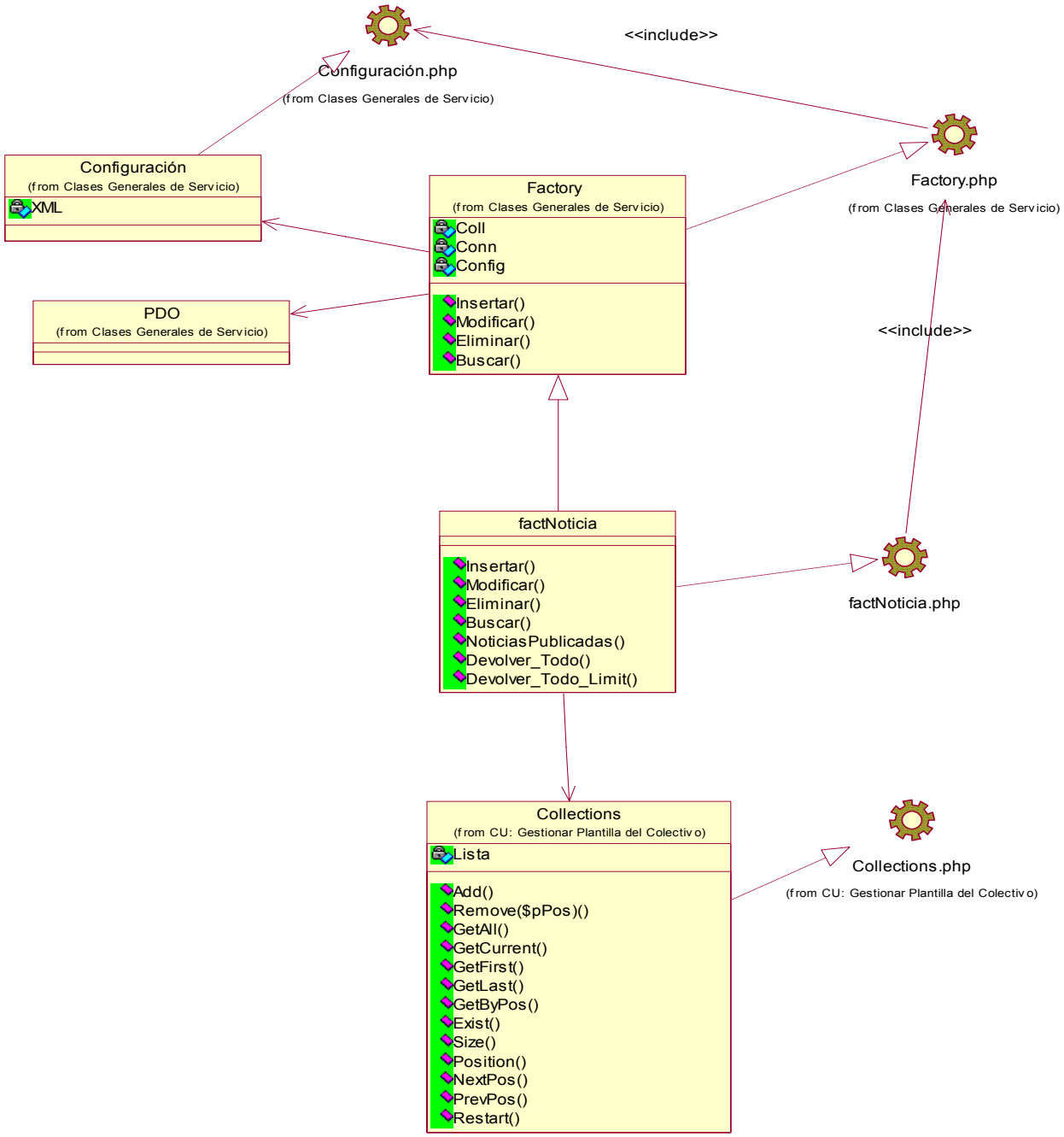


Figura 19. Diagrama de Clases Paquete de servicios: Gestionar Noticias

3.4.5 Diagramas de interacción

Anteriormente se reflejó que se había dividido el sistema en dos módulos (Ver Figura2, Capitulo 2), partiendo de esto, en este epígrafe se representarán los diagramas de secuencia del Módulo con mayor peso para el sistema como lo es el Docente-Metodológico. Los diagramas de interacción se dividen en dos tipos de diagramas de UML, los diagramas de secuencia y los diagramas de colaboración. Para modelar los aspectos dinámicos de este sistema se utilizará diagramas de secuencia por cada uno de los escenarios de cada caso de uso, destacando la secuencia temporal de los mensajes.

Tabla 23. Diagramas de Secuencia

	Escenario	**Anexo**
Gestionar Plantilla del Colectivo	Crear Plantilla del colectivo	Anexo 1
	Modificar Plantilla del Colectivo	Anexo 2
	Eliminar Plantilla del Colectivo	Anexo 3
Gestionar PTM	Crear PTM	Anexo 4
	Eliminar PTM	Anexo 5
	Modificar PTM	Anexo 6
	Mostrar PTM	Anexo 7
	Introducir Introducción	Anexo 8
	Introducir Diagnóstico	Anexo 9
	Introducir Objetivos	Anexo 10
	Introducir Actividad	Anexo 11
	Introducir Acciones	Anexo 12
Gestionar PTI	Crear PTI	Anexo 13
	Eliminar PTI	Anexo 14
	Introducir Actividad	Anexo 15
	Modificar Actividades	Anexo 16
	Mostrar PTI	Anexo 17

Gestionar Controles a Clases	Adicionar control	Anexo18
	Modificar Control	Anexo19
	Eliminar Control	Anexo20
Evaluar PTI	Evaluar ActividadA	(Ver Rational DDCactual.mdl)
	Evaluar PTI	(Ver Rational DDCactual.mdl)
	Eliminar EvaluaciónA	(Ver Rational DDCactual.mdl)
Evaluar PTM	Eliminar Evaluación	(Ver Rational DDCactual.mdl)
	Evaluar Actividad	(Ver Rational DDCactual.mdl)
	Evaluar PTM	(Ver Rational DDCactual.mdl)
Gestionar Acta	Crear Acta	(Ver Rational DDCactual.mdl)
	Eliminar Acta	(Ver Rational DDCactual.mdl)
Evaluar Controles a Clases	Eliminar E CClase	(Ver Rational DDCactual.mdl)
	Evaluar CClase	(Ver Rational DDCactual.mdl)
	Modificar ECCalse	(Ver Rational DDCactual.mdl)
Gestionar Plantilla de Alumnos Ayudantes	AdicionarAA	(Ver Rational DDCactual.mdl)
	ModificarAA	(Ver Rational DDCactual.mdl)
	EliminarAA	(Ver Rational DDCactual.mdl)
Mostrar informe del claustro	Mostrar Informe	(Ver Rational DDCactual.mdl)
Mostrar Plantilla de la Disciplina	Mostrar Plantilla	(Ver Rational DDCactual.mdl)
Mostrar Plantilla de AA	Mostrar Plantilla AA	(Ver Rational DDCactual.mdl)

Mostrar Controles a clases	Mostrar Controles a Clases	(Ver Rational DDCactual.mdl)

3.5 Diseño de la base de datos

Para diseñar la base de datos del sistema, se utilizó el diagrama el modelo de datos, el cual está basado en la modelación de las clases persistentes que son utilizadas en el modelo del diseño.

El modelo de datos describe de una forma abstracta cómo se representan los datos, en un sistema de gestión de base de datos. Es una herramienta para especificar los tipos de datos y la organización de los mismos que son permisibles en una base de datos específica. Este modelo es una guía para el diseño de la base de datos y es el elemento clave en el diseño de la arquitectura de un manejador de bases de datos. (Figura 20)

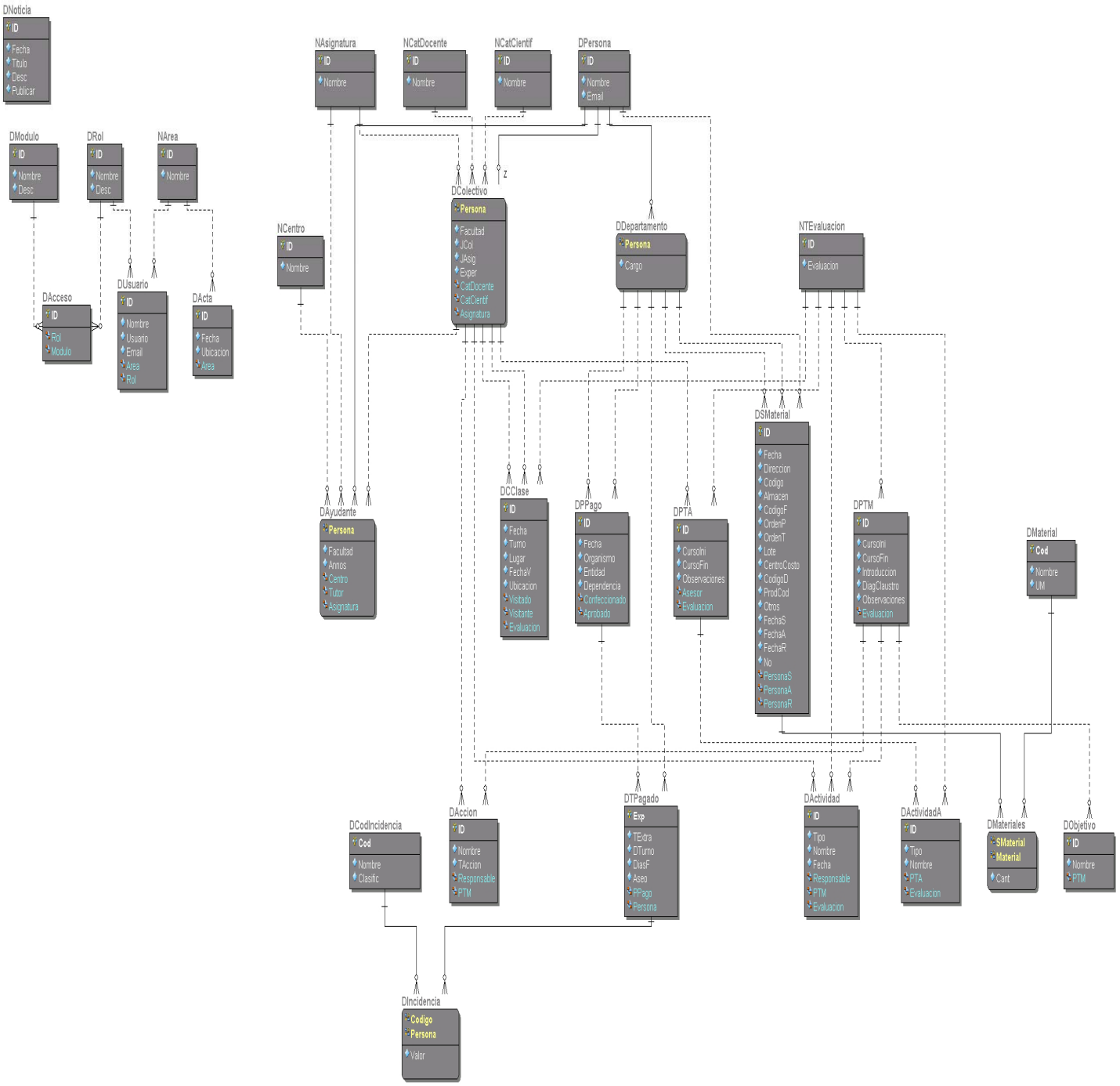


Figura 20. Modelo de Datos

3.6 Diagrama de despliegue

El diagrama de despliegue permite apreciar de forma visual como se encuentran relacionados físicamente los componentes de la aplicación. En este caso el usuario accede al sistema desde un navegador Web por medio del protocolo HTTP. La aplicación se encuentra hospedada en un servidor Web, el cual se conecta al servidor de base de datos (PostgreSQL) mediante el protocolo TCP/IP. (Ver figura 21)

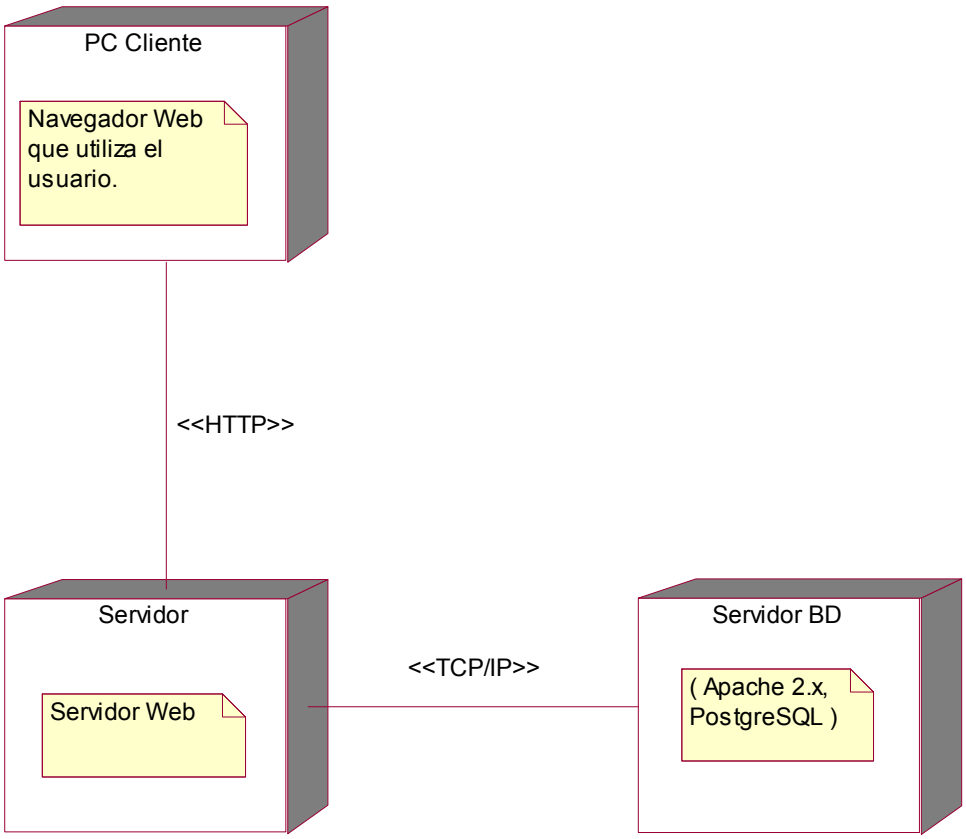


Figura 21. Diagrama de Despliegue

3.7 Conclusiones

En el presente capítulo se obtuvo la solución al sistema propuesto, describiendo la arquitectura y patrones utilizados para el desarrollo de la aplicación. Se mostraron los resultados de la etapa de diseño, para ello se mostró los diagramas de clases por módulos lo cual brindó una mejor comprensión del sistema. Se modelaron los componentes como clases utilizando las extensiones Web del UML. Además se realizó el modelo de datos, que permite una mejor vista para realización de la base de datos. Se realizó el diseño del diagrama de despliegue donde se describen los nodos de procesamiento en tiempo real donde se ejecutará la aplicación y los vínculos entre ellos. Todos estos elementos obtenidos son claves para la correcta implementación del sistema propuesto.

Capítulo 4

4 CAPÍTULO4: ANÁLISIS DE ESTIMACIÓN DE ESFUERZO Y TIEMPO DE DESARROLLO

4.1 Introducción

En este capítulo se estima el esfuerzo y se analizan los beneficios del sistema propuesto, utilizando el método de estimación por Puntos de Casos de Uso. Se obtendrán valores de importantes indicadores como son: esfuerzo y tiempo de desarrollo.

4.2 Estimación de esfuerzo

La estimación mediante el análisis de Puntos de Casos de Uso es un método propuesto originalmente por Gustav Karner de Objectory AB, y posteriormente refinado por muchos otros autores. Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.

A continuación, se realizará la estimación al Sistema de Gestión de Información del DDC-Ingeniería y Gestión de Software UCI.

Cálculo de Puntos de Casos de Uso sin ajustar

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.

Factor de Peso de los Actores sin Ajustar (UAW)

- Para calcular UAW

Tabla 24. Clasificación de los Actores

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
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2	0*2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3	8*3
		Total	**24**

Factor de Peso de los Casos de Uso sin ajustar (UUCW)

- Para calcular UUCW

Tabla 25. Clasificación de los Casos de Uso

Tipo	Descripción	Peso	Cant * peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5	5*5
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10	8*10
Complejo	El Caso de Uso contiene más de 8 transacciones	15	7*15
		Total	**210**

Luego, **UUCP= 24+ 210= 234**

Cálculo de Puntos de Casos de Uso Ajustados

UCP = UUCP x TCF x 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

Factor de complejidad técnica (TCF)

TCF = 0.6 + 0.01 * Σ **(Peso$_i$** * **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

Tabla 26. Factor de Complejidad

Factor	Descripción	Peso	Valor	Σ (Peso$_i$ * Valor$_i$)
T1	Sistema distribuído	2	0	0
T2	Objetivos de performance o tiempo de respuesta	1	2	2
T3	Eficiencia del usuario final	1	1	1
T4	Procesamiento interno complejo	1	0	0
T5	El código debe ser reutilizable	1	2	2
T6	Facilidad de instalación	0.5	2	1

T7	Facilidad de uso	0.5	4	2
T8	Portabilidad	2	0	0
T9	Facilidad de cambio	1	3	3
T10	Concurrencia	1	0	0
T11	Incluye objetivos especiales de seguridad	1	3	3
T12	Provee acceso directo a terceras partes	1	3	3
T13	Se requieren facilidades especiales de entrenamiento a los usuarios	1	0	0
			Total	**17**

Luego: **TCF = 0.6+0.01* 17= 0.77**

Factor de ambiente (EF)

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

Tabla 27. Factor Ambiente

Factor	**Descripción**	**Peso**	**Valor**	Σ **(Peso$_i$** * **Valor$_i$)**
E1	Familiaridad con el modelo de proyecto utilizado	1.5	2	3
E2	Experiencia en la aplicación	0.5	3	1.5
E3	Experiencia en orientación a objetos	1	2	2
E4	Capacidad del analista líder	0.5	5	2.5

E5	Motivación	1	5	5
E6	Estabilidad de los requerimientos	2	3	6
E7	Personal part-time	-1	0	0
E8	Dificultad del lenguaje de programación	-1	3	-3
			Total	**17**

Luego: **EF = 1.4 - 0.03 *17= 0.89**

Después de haber calculado los factores anteriores se calculará:

UCP= 234* 0.77* 0.89= 160.36

De los Puntos de Casos de Uso a la estimación del esfuerzo

E = UCP x CF

Donde:

E: esfuerzo estimado en horas-hombre

UCP: Puntos de Casos de Uso ajustados

CF: factor de conversión

- 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)

= 3+ 0=0

Como **$Total_{EF}$** = 3

CF = 28 Horas-Hombre

Luego, **E =160.36 * 28 Horas-Hombre= 4490.08 Horas-Hombre**

Calcular esfuerzo de todo el proyecto

Tabla 28. Esfuerzo

Actividad	**% esfuerzo**	**Valor esfuerzo**
Análisis	10%	449.01 Horas-Hombre
Diseño	20%	898.02 Horas-Hombre
Implementación	40%	1796.03 Horas-Hombre
Prueba	15%	673.51 Horas-Hombre
Sobrecarga	15%	673.51 Horas-Hombre
Total	100%	4490.08 Horas-Hombre

Un mes tiene como promedio 24 días, por lo que la cantidad de horas que puede trabajar una persona en ese espacio de tiempo es igual a 192.

E = 4490.08 Horas-Hombre.

Por cada hombre, trabajando 8 horas diarias en 24 días, que significaría un total de 192 horas, tendríamos que **E** = 23.4 mes-hombre.

Como el sistema es desarrollado por dos personas, el problema analizado puede desarrollarse en alrededor de 11.7 meses

4.3 Beneficios tangibles e intangibles

El Sistema de Gestión de Información del DDC-Ingeniería y Gestión de Software UCI es un software que puede adaptarse fácilmente a otro entorno, por lo que puede utilizarse no sólo para el departamento de Ingeniería y Gestión de Software, sino para cualquier departamento de la Universidad.

Su principal objetivo es controlar y organizar procesos metodológicos y de soporte de este departamento por tanto, los beneficios inmediatos son mayormente intangibles:

- Ahorro de tiempo de trabajo para los miembros del departamento.
- Hacer más fácil el trabajo de todos los miembros del departamento en el chequeo del cumplimiento de las actividades docente, y planeación metodológica.
- Posibilidad de ver reportes importantes que realizan los diferentes miembros del departamento.

- Hacer más eficiente el control de la información en el proceso metodológico y laboral.
- Facilitar el acceso a la información que se necesite para la toma de decisiones.
- Posibilidad de mantener informado a los usuarios sobres las tareas que debe realizar.

4.4 Análisis de las estimaciones y beneficios

El desarrollo de este sistema no requiere grandes gastos de recursos, ni de tiempo; los servidores que existen en la Universidad son capaces de soportar la base de datos que contiene la información, así como el software en su totalidad.

El sistema está orientado al usuario, es de fácil manejo, por lo que no reporta gastos por concepto de entrenamiento a los usuarios del mismo. Además, puede ser extendido para uso general (En todos los departamentos).

Por todo ello se considera que es factible el desarrollo de la aplicación y que el esfuerzo de desarrollo es 23.4 mes-hombre.

4.5 Conclusiones

En este capítulo se ha realizado un estudio de factibilidad de todo el sistema, utilizando el método por puntos de caso de uso, teniendo en cuenta el esfuerzo como el principal factor para esto. También se expuso los principales beneficios que trae consigo dicho sistema. Con todo esto se llegó a la conclusión que resultará factible implementar la aplicación, ya que los beneficios sociales que se alcanzarán son considerables.

CONCLUSIONES GENERALES

- Al culminar el trabajo se llegaron a las siguientes conclusiones: se logró la implementación de una aplicación, que brinda una serie de funcionalidades importantes para la preparación metodológica docente del Departamento Central de Ingeniería y Gestión de Software de la UCI, logrando establecer pautas con respecto a la seguridad de la información almacenada de los miembros del Departamento. Está provisto de un ambiente cómodo, fácil de entender y que cumple los estándares de diseño.

- Para el desarrollo de la propuesta se realizó el análisis de las tecnologías más usadas en la actualidad, concluyéndose en la utilización de PHP 5, como lenguaje de programación, el gestor de bases de datos utilizado fue PostgreSQL, junto a otras tecnologías como las hojas de estilo CSS.

- La aplicación se desarrolló siguiendo la metodología RUP, y se utilizaron representaciones UML para la modelación de todas las fases del proyecto. Se identificaron los procesos principales del negocio, se definieron los requerimientos del sistema, tanto funcionales como no funcionales, estructurándose además, el modelo de casos de uso del sistema.

- Se realizó el diseño del sistema a través de diagramas de clases del diseño, de interacción, de despliegue. Se elaboró el modelo de datos, analizándose finalmente el costo y los beneficios que genera el sistema.

- Por todo lo anterior se concluye que el objetivo propuesto para el presente proyecto ha sido cumplido satisfactoriamente, incluyendo una serie de recomendaciones que deben tenerse en cuenta para el trabajo futuro.

RECOMENDACIONES

Al concluir este trabajo se recomienda lo siguiente para versiones futuras:

- Profundizar en el estudio de los procesos y tareas de los Departamentos Docentes Centrales de la UCI, creándose un grupo de desarrollo en esta dirección.
- Continuar el desarrollo de este sistema, adicionándole nuevas funcionalidades y dándole seguimiento a las actividades de soporte del Departamento, adecuándolo a los cambios que puedan surgir en etapas posteriores, en el proceso de enseñanza de la UCI.
- Permitir la implantación de futuras versiones a nivel central, donde directivos a una mayor escala, en el proceso de formación de la UCI, puedan formar parte del grupo de usuarios directos del sistema, y tener un control más exacto de los resultados en el proceso docente metodológico.
- Implantar el sistema en la Intranet de la Universidad de las Ciencias Informáticas para prestar los servicios implementados dándole solución inmediata a los problemas detectados en la etapa de estudio preliminar.

REFERENCIAS BIBLIOGRÁFICAS

BARRIENTOS ENRÍQUEZ, A. M. *El proceso Unificado de Modelado (RUP)*, [en línea]. 2005. [26/02/07]. Disponible en: http://www.monografias.com/trabajos16/lenguaje-modelado-unificado/lenguaje-modelado-unificado.shtml#PROCESO

BATISTA VELAZQUEZ, Y. *Elaboración de un Sistema de Gestión para el Departamento Docente central de Ingeniería y Gestión de Software*. Tesis (Ingeniero Informático). La Habana, Universidad de las Ciencias Informáticas, 2006. 8. p.

CABEZAS GRANADO, L. M. *PHP 5*. Anaya Multimedia, 2004, 384 p,

CORPORATION, N. *NuSphere PhpED - PHP Editor*, [en línea]. 2007-03-29. [15/11/06]. Disponible en: http://www.nusphere.com/products/phped.htm

DÍAZ, N. *Ventajas de PostgreSQL*, [en línea]. 2003. [5/12/06]. Disponible en: http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html

GUTIÉRREZ., J. J. *¿Qué es un framework Web?.* [en línea]. 2007. [25/10/06]. Disponible en: www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

JACOBSON, I.; G. BOOCH*, et al. El proceso unificado de desarrollo de software*. La Habana, Addison Wesley Longman, 2000, 438 p, Volumen 1

LARMAN, C. *UML Y Patrones. Introducción al Análisis y Diseño Orientado a Objetos*. La Habana, Pretice Hall., 2004, 590 p, Volumen 1

MANAGEMENT, E. I. *AllFusion® ERwin® Data Modeler.*, [en línea]. 2007. [20/01/07]. Disponible en: http://www3.ca.com/solutions/Product.aspx?ID=260

NIELSEN, J. *Web Style Sheets*, [en linea]. 2007. [10/02/07]. Disponible en: http://www.w3.org/Style/

PRESSMAN, R. S. *Ingeniería del Software, Un enfoque práctico*. 5a. ed. La Habana, Mc Graw Hill, 2005, 601 p, Volumen 1,

ROSSI, G.; A. MOREIRA*, et al. UML: el lenguaje estándar para el modelado del software*, [en linea]. Marzo-Abril. [5/02/07]. Disponible en: http://www.ati.es/novatica/2004/168/168-4.pdf

RUMBAUGH, J.; I. JACOBSON*, et al. El Lenguaje Unificado de Modelado. Manual de Referencia*. Addison Wesley 2000,

SÁNCHEZ MATÍAS, E. *Cuba utilizará Software Libre*, [en línea]. 2004. [25/10/06]. Disponible en: http://cronopios.net/Textos/breve_introduccion_al_software_libre.pdf

SOLUTIONS, E. D. M. *EMS PostgreSQL Manager*, [en línea]. 2005-07-05. [10/01/07]. Disponible en: http://www.sqlmanager.net/products/postgresql/manager

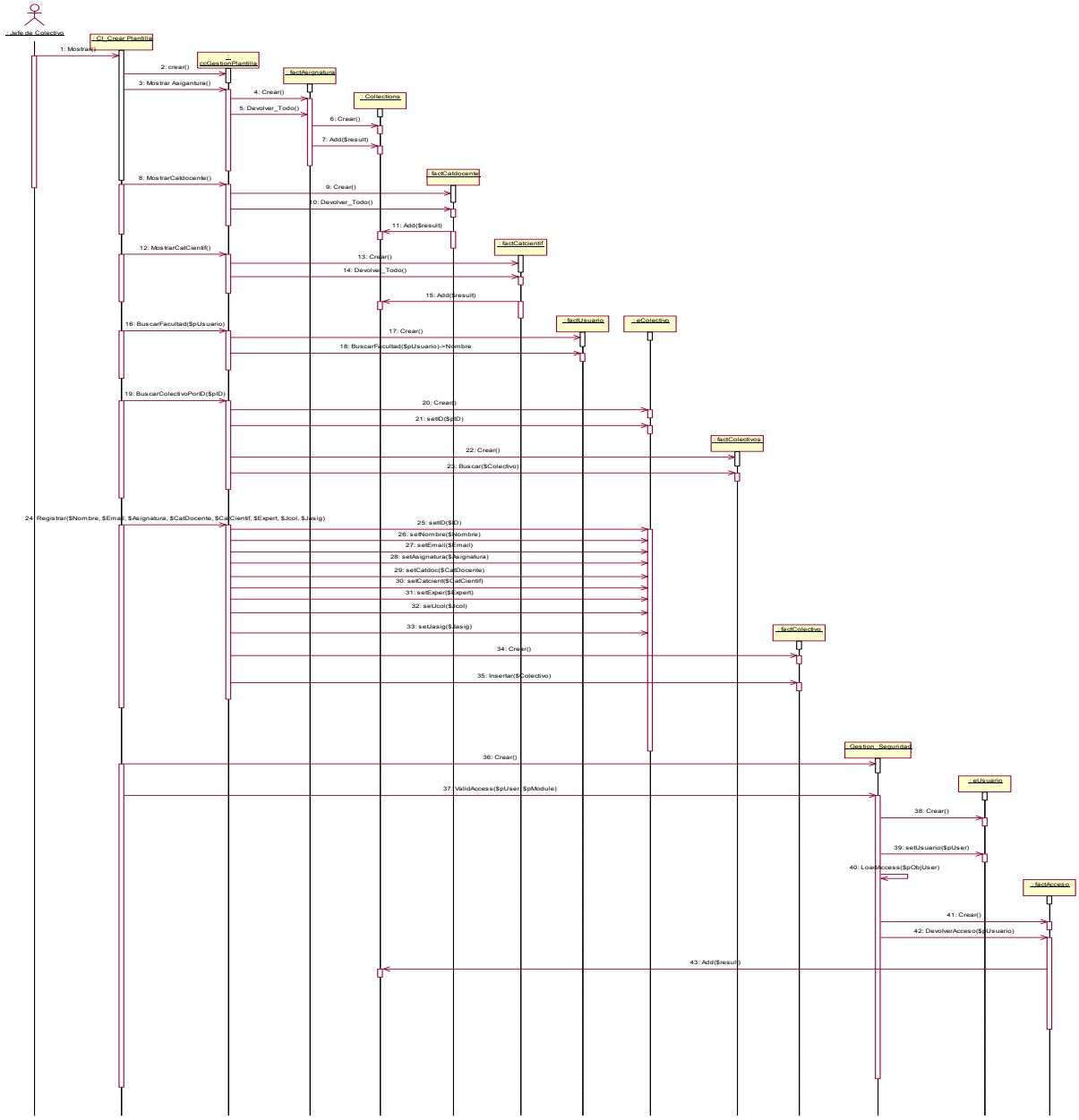
VIAL S, J. M. *Introducción a la programación en PHP*, [en linea]. 2001-2006. [22/02/07]. Disponible en: http://www.aulambra.com/ver2.asp?id=146&tipo=

WIKIMEDIA, F. *Caso de Uso*, [en línea]. 2007. [24/02/07]. Disponible en: http://es.wikipedia.org/wiki/Caso_de_uso

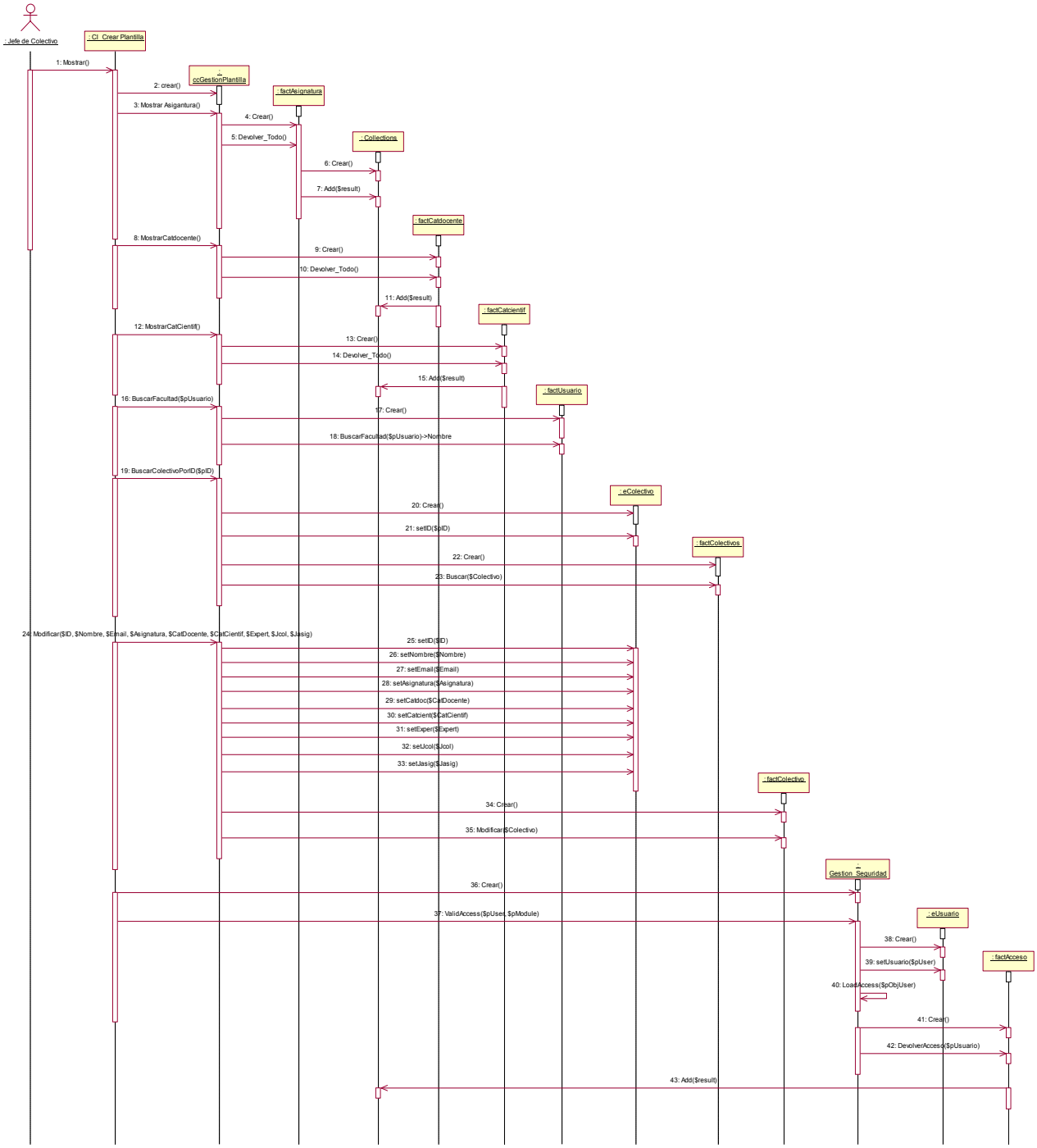
---. *XML*, [en línea]. 20 de marzo 2007. [10/12/06]. Disponible en: http://es.wikipedia.org/wiki/XML

ANEXOS

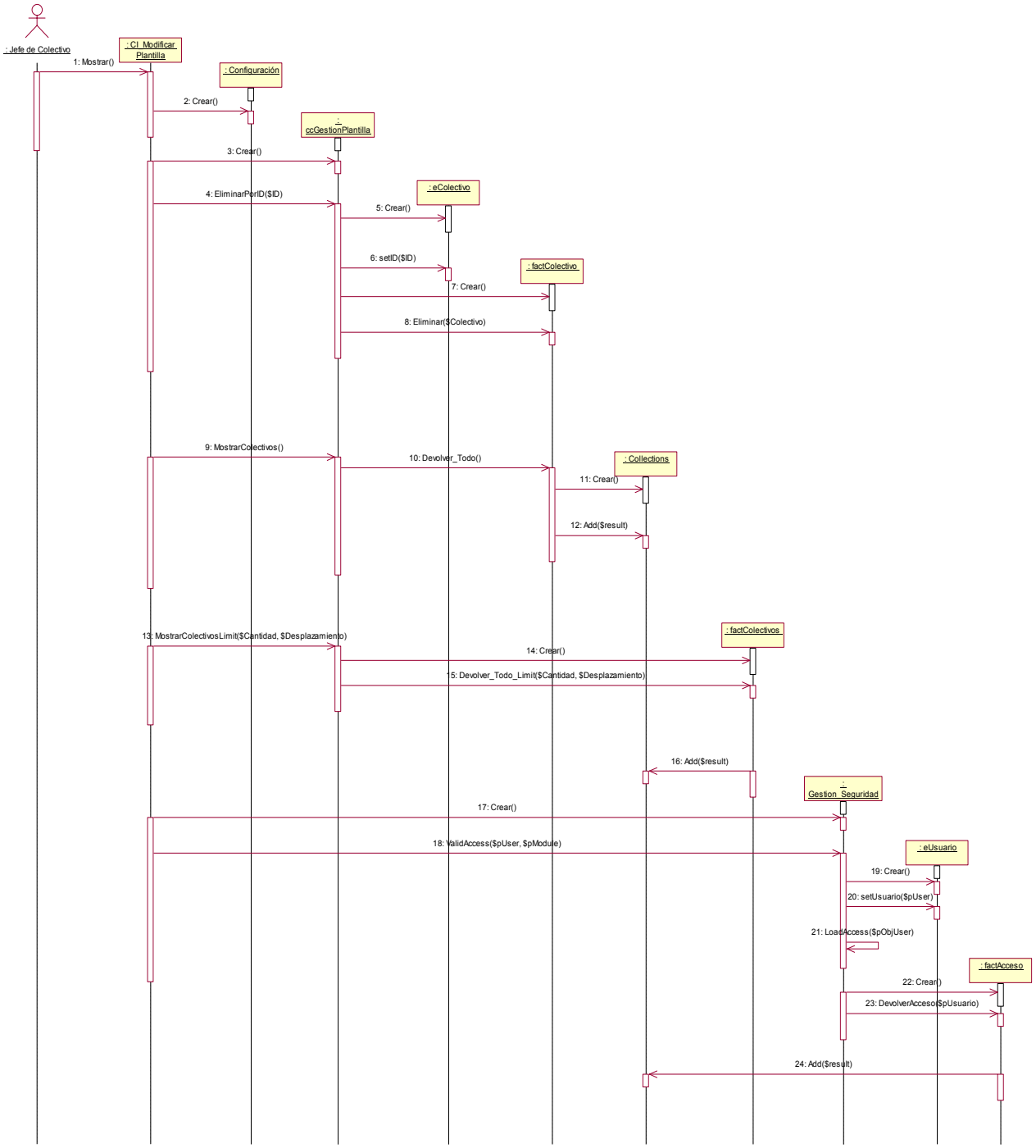
Anexo1



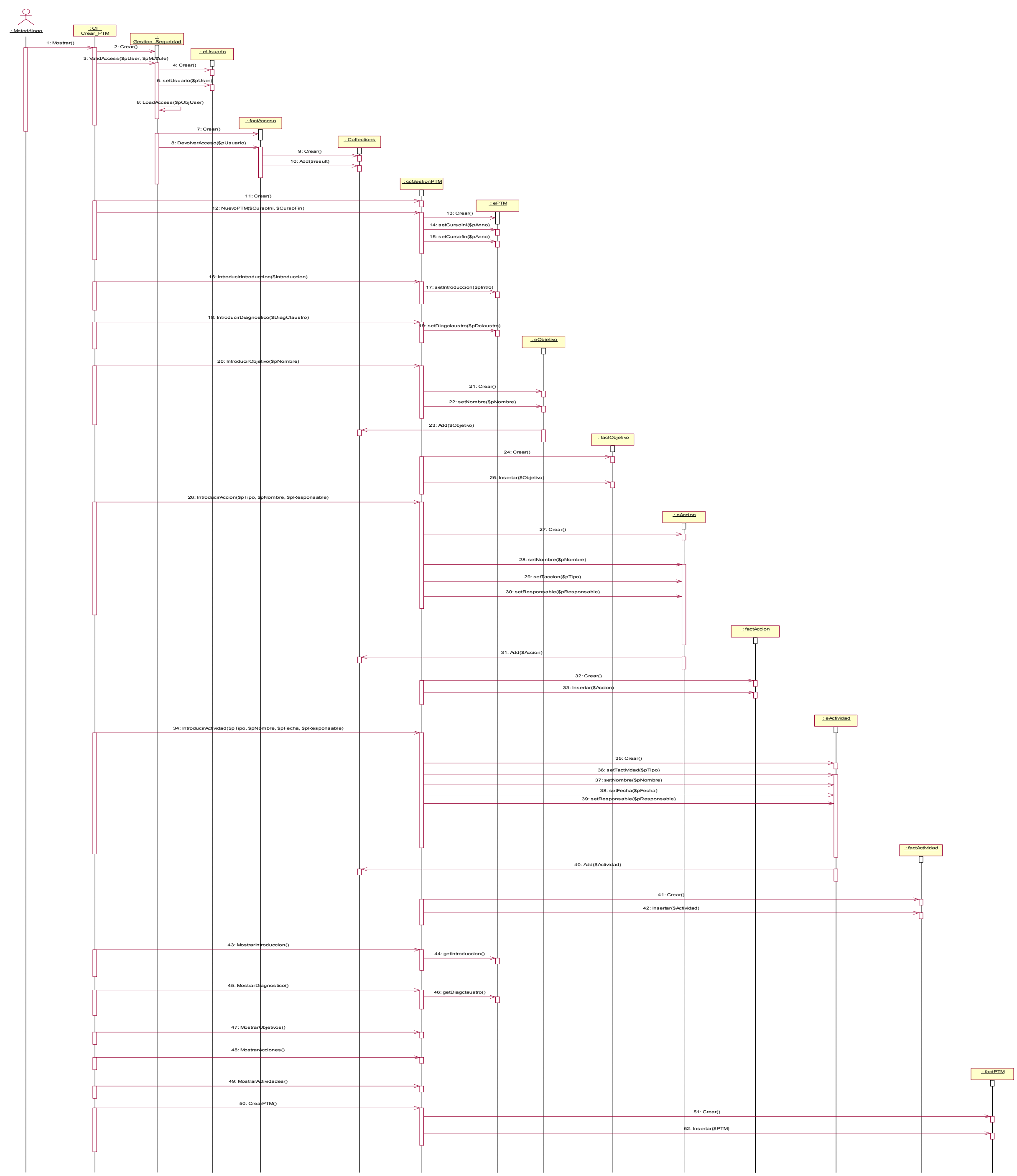
Anexo2


: Jefe de Colectivo
: Cl_Crear Plantilla
: ccGestionPlantilla
: factAsignatura
: Collections
: factCaldocente
: factCatcientif
: factUsuario
: eColectivo
: factColectivos
: factColectivo
Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: crear()
3: Mostrar Asigantura()
4: Crear()
5: Devolver_Todo()
6: Crear()
7: Add($result)
8: MostrarCatdocente()
9: Crear()
10: Devolver_Todo()
11: Add($result)
12: MostrarCatCientif()
13: Crear()
14: Devolver_Todo()
15: Add($result)
16: BuscarFacultad($pUsuario)
17: Crear()
18: BuscarFacultad($pUsuario)->Nombre
19: BuscarColectivoPorID($pID)
20: Crear()
21: setID($pID)
22: Crear()
23: Buscar($Colectivo)
24: Modificar($ID, $Nombre, $Email, $Asignatura, $CatDocente, $CatCientif, $Expert, $Jcol, $Jasig)
25: setID($ID)
26: setNombre($Nombre)
27: setEmail($Email)
28: setAsignatura($Asignatura)
29: setCatdoc($CatDocente)
30: setCatcient($CatCientif)
31: setExper($Expert)
32: setJcol($Jcol)
33: setJasig($Jasig)
34: Crear()
35: Modificar($Colectivo)
36: Crear()
37: ValidAccess($pUser, $pModule)
38: Crear()
39: setUsuario($pUser)
40: LoadAccess($pObjUser)
41: Crear()
42: DevolverAcceso($pUsuario)
43: Add($result)

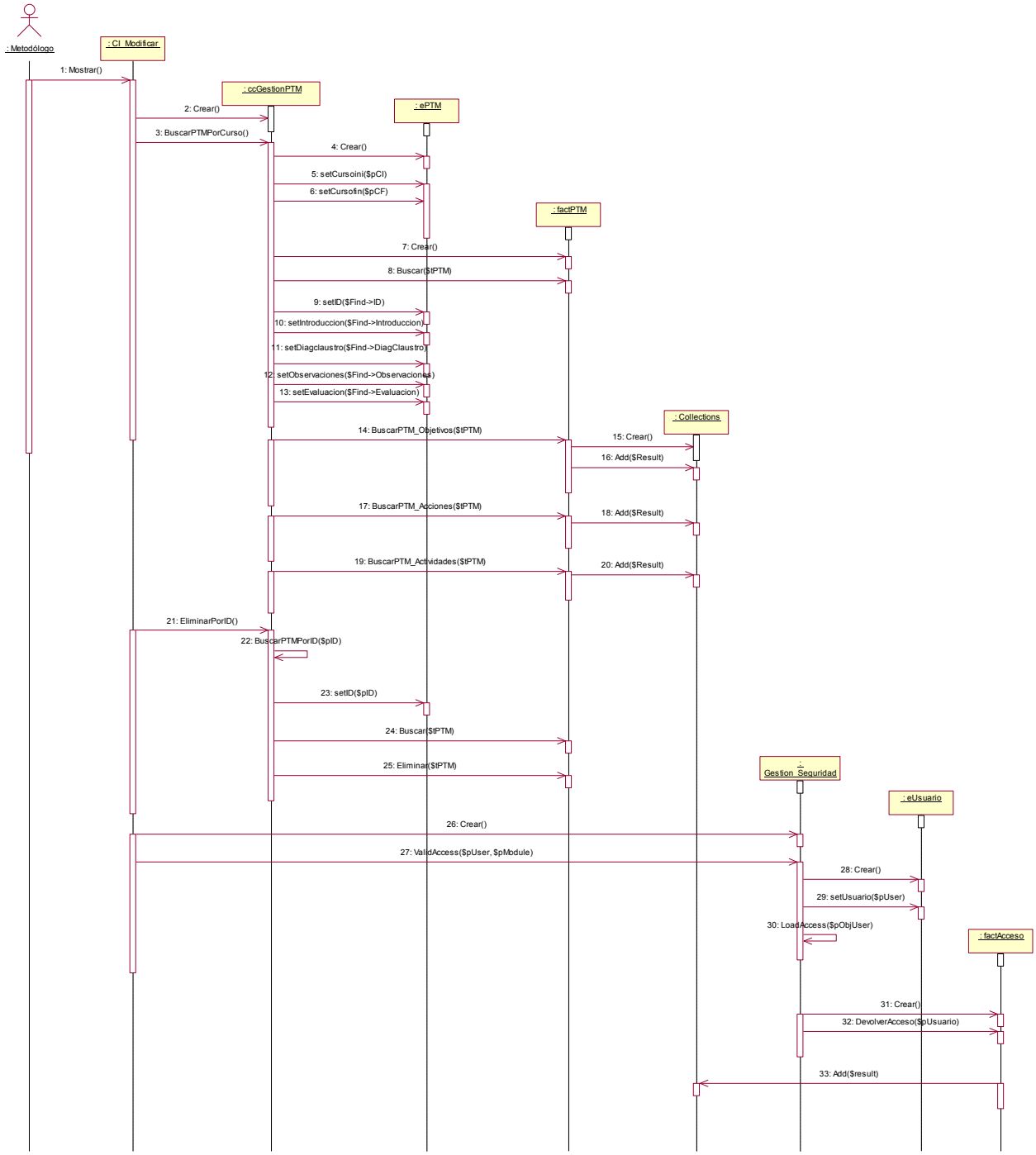
Anexo3


: Jefe de Colectivo
: Cl_Modificar_Plantilla
: Configuración
: ccGestionPlantilla
: eColectivo
: factColectivo
: Collections
: factColectivos
: Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: Crear()
3: Crear()
4: EliminarPorID($ID)
5: Crear()
6: setID($ID)
7: Crear()
8: Eliminar($Colectivo)
9: MostrarColectivos()
10: Devolver_Todo()
11: Crear()
12: Add($result)
13: MostrarColectivosLimit($Cantidad, $Desplazamiento)
14: Crear()
15: Devolver_Todo_Limit($Cantidad, $Desplazamiento)
16: Add($result)
17: Crear()
18: ValidAccess($pUser, $pModule)
19: Crear()
20: setUsuario($pUser)
21: LoadAccess($pObjUser)
22: Crear()
23: DevolverAcceso($pUsuario)
24: Add($result)

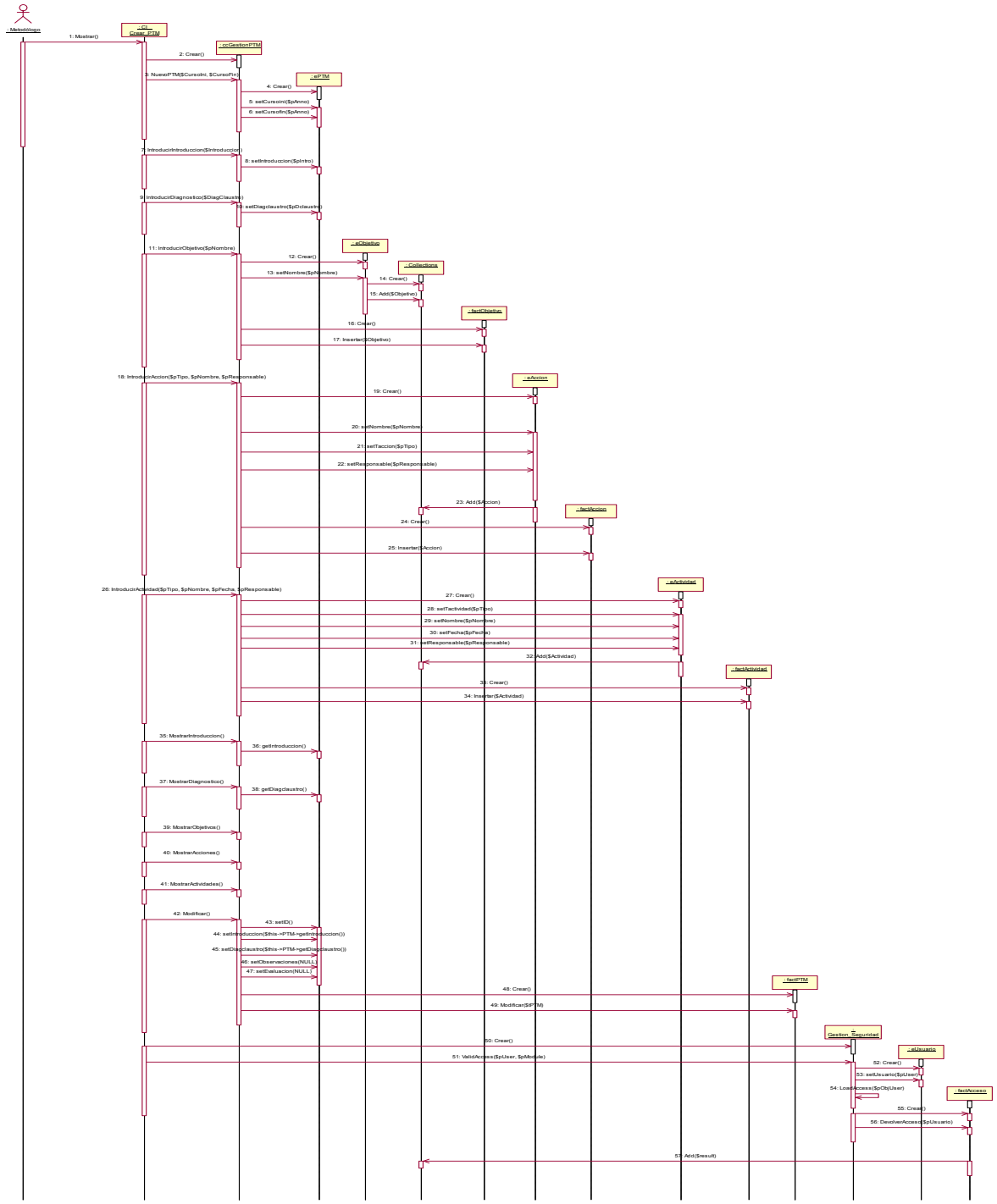
Anexo4


: Metodólogo
: CI_ Crear_PTM
: Gestion_Seguridad
: eUsuario
: factAcceso
: Collections
: ccGestionPTM
: ePTM
: eObjetivo
: factObjetivo
: eAccion
: factAccion
: eActividad
: factActividad
: factPTM
1: Mostrar()
2: Crear()
3: ValidAccess($pUser, $pModule)
4: Crear()
5: setUsuario($pUser)
6: LoadAccess($pObjUser)
7: Crear()
8: DevolverAcceso($pUsuario)
9: Crear()
10: Add($result)
11: Crear()
12: NuevoPTM($CursoIni, $CursoFin)
13: Crear()
14: setCursoini($pAnno)
15: setCursofin($pAnno)
16: IntroducirIntroduccion($Introduccion)
17: setIntroduccion($pIntro)
18: IntroducirDiagnostico($DiagClaustro)
19: setDiagclaustro($pDclaustro)
20: IntroducirObjetivo($pNombre)
21: Crear()
22: setNombre($pNombre)
23: Add($Objetivo)
24: Crear()
25: Insertar($Objetivo)
26: IntroducirAccion($pTipo, $pNombre, $pResponsable)
27: Crear()
28: setNombre($pNombre)
29: setTaccion($pTipo)
30: setResponsable($pResponsable)
31: Add($Accion)
32: Crear()
33: Insertar($Accion)
34: IntroducirActividad($pTipo, $pNombre, $pFecha, $pResponsable)
35: Crear()
36: setTactividad($pTipo)
37: setNombre($pNombre)
38: setFecha($pFecha)
39: setResponsable($pResponsable)
40: Add($Actividad)
41: Crear()
42: Insertar($Actividad)
43: MostrarIntroduccion()
44: getIntroduccion()
45: MostrarDiagnostico()
46: getDiagclaustro()
47: MostrarObjetivos()
48: MostrarAcciones()
49: MostrarActividades()
50: CrearPTM()
51: Crear()
52: Insertar($PTM)

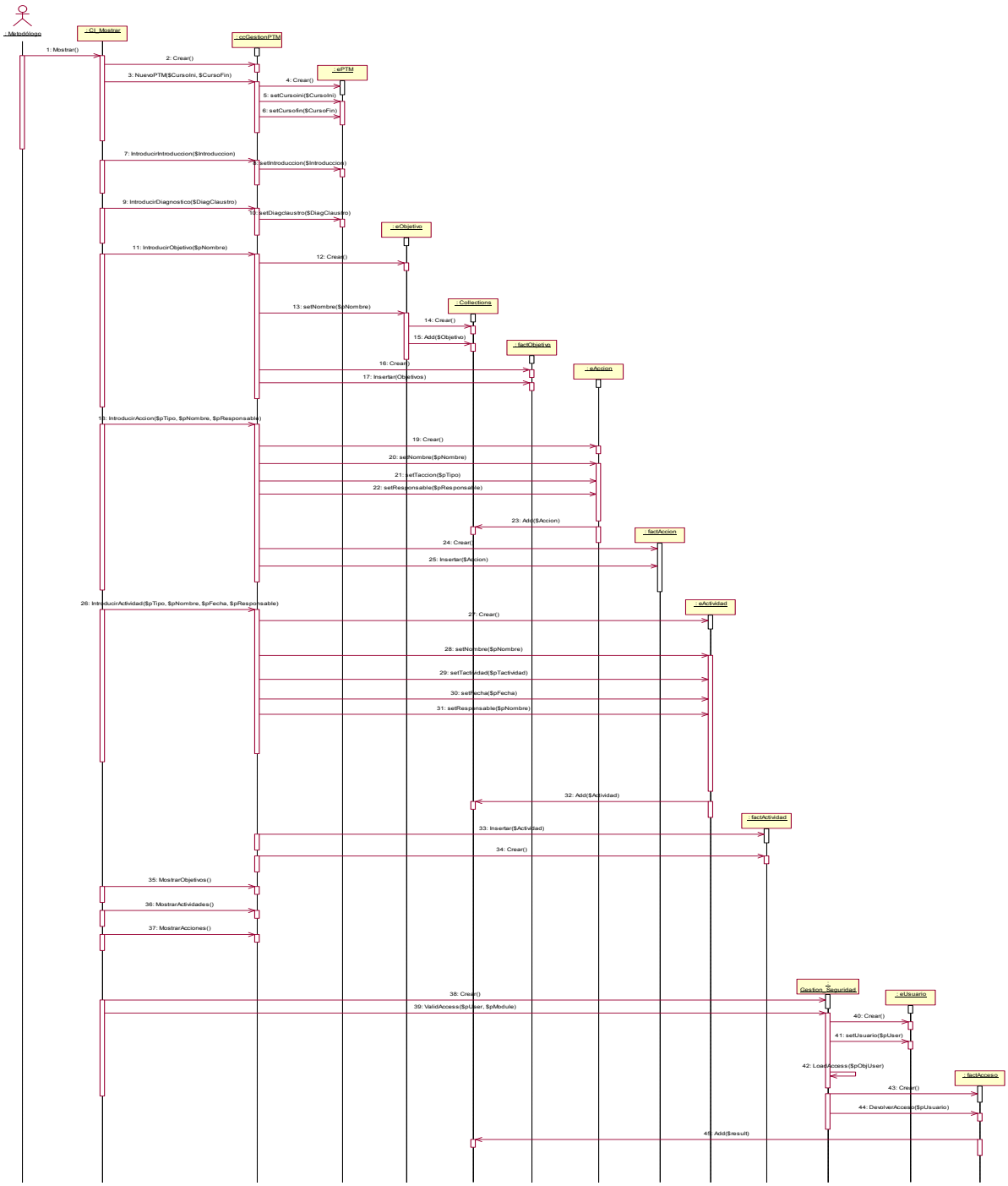
Anexo5


: Metodólogo
: CI_Modificar
: ccGestionPTM
: ePTM
: factPTM
: Collections
: Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: Crear()
3: BuscarPTMPorCurso()
4: Crear()
5: setCursoini($pCI)
6: setCursofin($pCF)
7: Crear()
8: Buscar($tPTM)
9: setID($Find->ID)
10: setIntroduccion($Find->Introduccion)
11: setDiagclaustro($Find->DiagClaustro)
12: setObservaciones($Find->Observaciones)
13: setEvaluacion($Find->Evaluacion)
14: BuscarPTM_Objetivos($tPTM)
15: Crear()
16: Add($Result)
17: BuscarPTM_Acciones($tPTM)
18: Add($Result)
19: BuscarPTM_Actividades($tPTM)
20: Add($Result)
21: EliminarPorID()
22: BuscarPTMPorID($pID)
23: setID($pID)
24: Buscar($tPTM)
25: Eliminar($tPTM)
26: Crear()
27: ValidAccess($pUser, $pModule)
28: Crear()
29: setUsuario($pUser)
30: LoadAccess($pObjUser)
31: Crear()
32: DevolverAcceso($pUsuario)
33: Add($result)

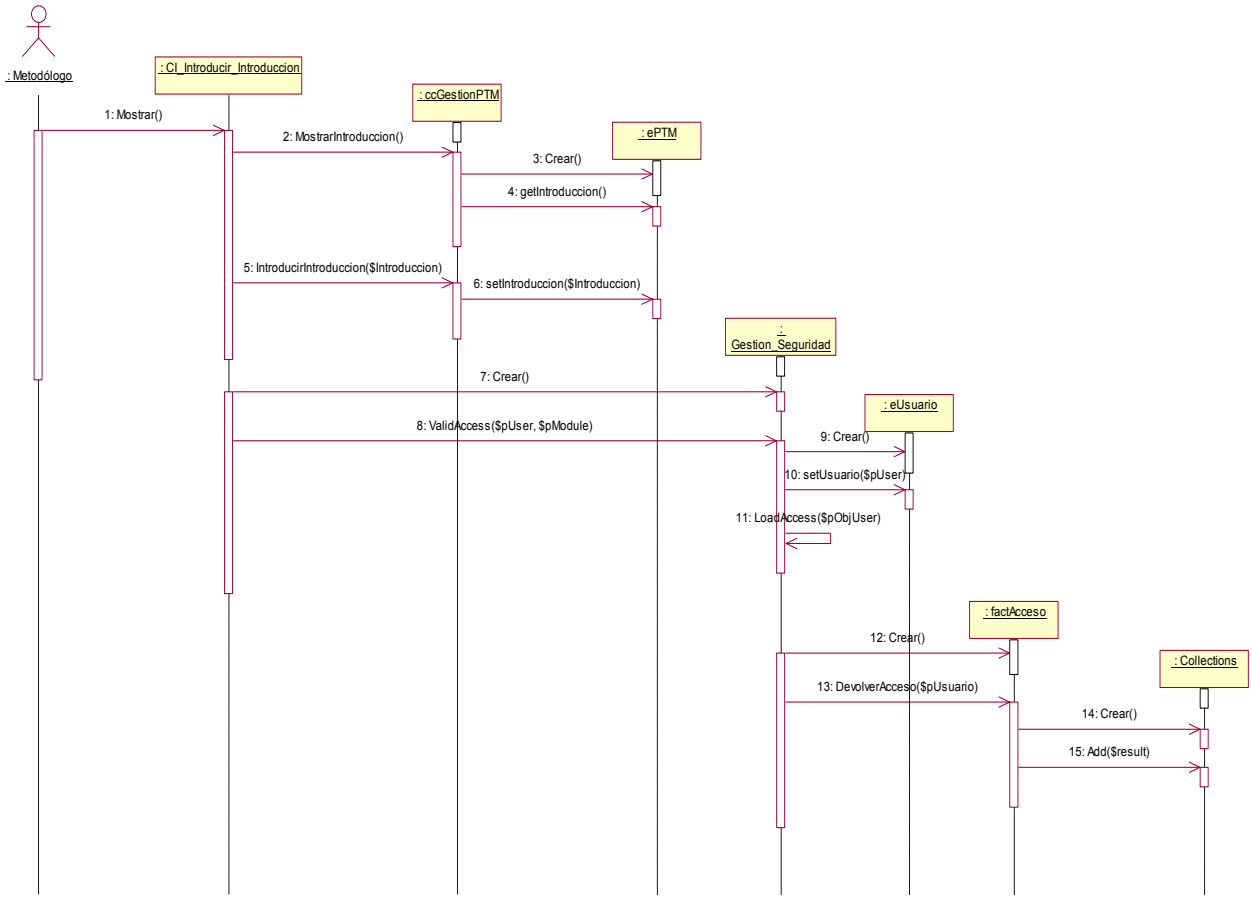
Anexo6


:Metodólogo
:cc Crear_PTM
:ccGestionPTM
:eObjetivo
:eAccion
:eActividad
:ccGestion_Seguridad
:eUsuario
:factAccesso
1: Mostrar()
2: Crear()
3: NuevoPTM($CursoIni, $CursoFin)
4: Crear()
5: setCursoIni($pAnno)
6: setCursoFin($pAnno)
7: IntroducirIntroduccion($pIntroduccion)
8: setIntroduccion($pIntro)
9: IntroducirDiagnostico($DiagClaustro)
10: setDiagclaustro($pDclaustro)
11: IntroducirObjetivo($pNombre)
12: Crear()
13: setNombre($pNombre)
14: Crear()
15: Add($Objetivo)
16: Crear()
17: Insertar($Objetivo)
18: IntroducirAccion($pTipo, $pNombre, $pResponsable)
19: Crear()
20: setNombre($pNombre)
21: setTaccion($pTipo)
22: setResponsable($pResponsable)
23: Add($Accion)
24: Crear()
25: Insertar($Accion)
26: IntroducirActividad($pTipo, $pNombre, $pFecha, $pResponsable)
27: Crear()
28: setTactividad($pTipo)
29: setNombre($pNombre)
30: setFecha($pFecha)
31: setResponsable($pResponsable)
32: Add($Actividad)
33: Crear()
34: Insertar($Actividad)
35: MostrarIntroduccion()
36: getIntroduccion()
37: MostrarDiagnostico()
38: getDiagclaustro()
39: MostrarObjetivos()
40: MostrarAcciones()
41: MostrarActividades()
42: Modificar()
43: setID()
44: setIntroduccion($this->PTM->getIntroduccion())
45: setDiagclaustro($this->PTM->getDiagclaustro())
46: setObservaciones(NULL)
47: setEvaluacion(NULL)
48: Crear()
49: Modificar($PTM)
50: Crear()
51: ValidAccess($pUser, $pModule)
52: Crear()
53: setUsuario($pUser)
54: LoadAccess($pObjUser)
55: Crear()
56: DevolverAccesos($pUsuario)
57: Add($result)

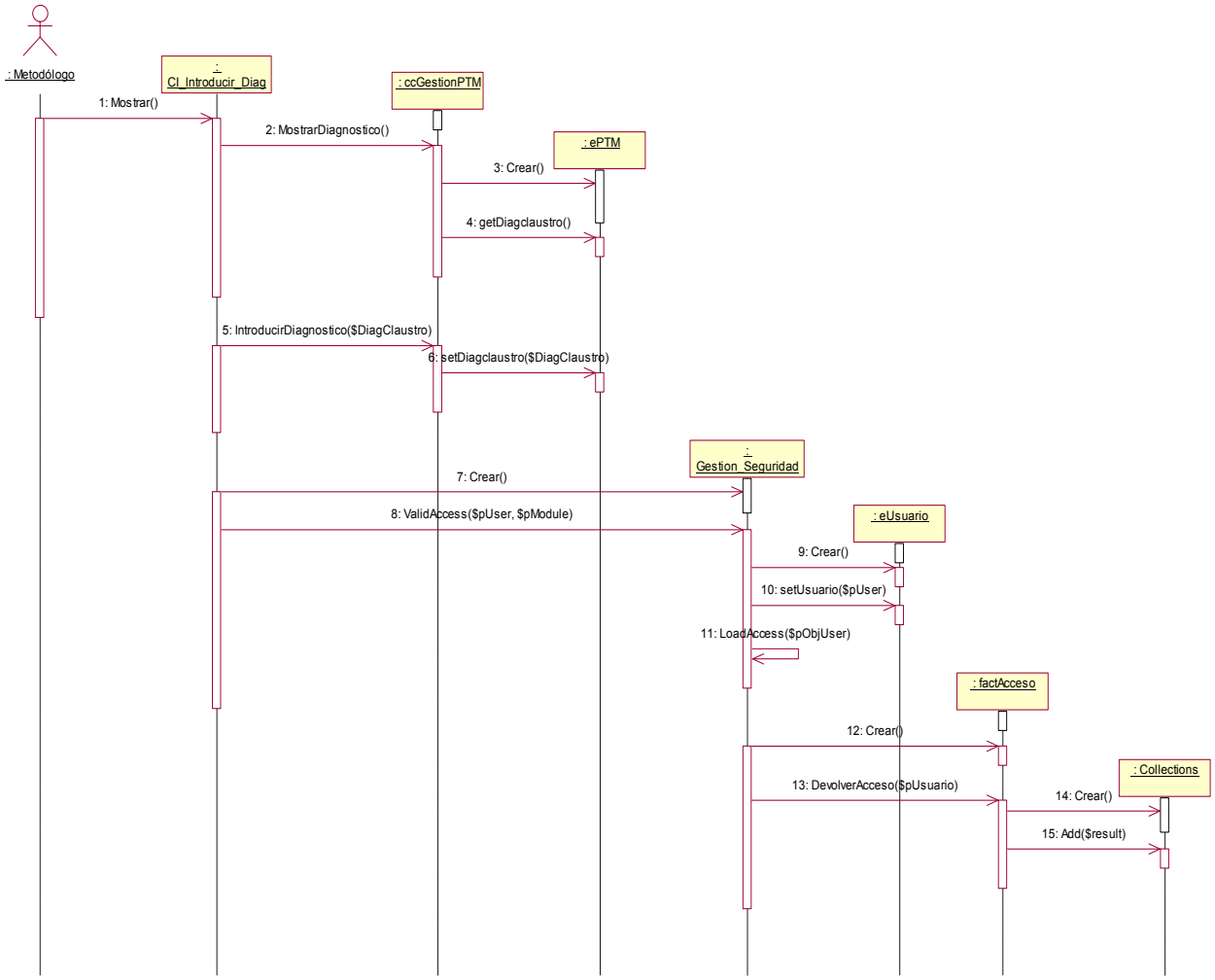
Anexo7


: Metodólogos
: Cl_Mostrar
: ccGestionPTM
: ePTM
: eObjetivo
: Collections
: factObjetivo
: eAccion
: factAccion
: eActividad
: factActividad
: Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: Crear()
3: NuevoPTM($CursoIni, $CursoFin)
4: Crear()
5: setCursoini($CursoIni)
6: setCursofin($CursoFin)
7: IntroducirIntroduccion($Introduccion)
8: setIntroduccion($Introduccion)
9: IntroducirDiagnostico($DiagClaustro)
10: setDiagclaustro($DiagClaustro)
11: IntroducirObjetivo($pNombre)
12: Crear()
13: setNombre($pNombre)
14: Crear()
15: Add($Objetivo)
16: Crear()
17: Insertar(Objetivos)
18: IntroducirAccion($pTipo, $pNombre, $pResponsable)
19: Crear()
20: setNombre($pNombre)
21: setTaccion($pTipo)
22: setResponsable($pResponsable)
23: Add($Accion)
24: Crear()
25: Insertar($Accion)
26: IntroducirActividad($pTipo, $pNombre, $pFecha, $pResponsable)
27: Crear()
28: setNombre($pNombre)
29: setTactividad($pTactividad)
30: setFecha($pFecha)
31: setResponsable($pNombre)
32: Add($Actividad)
33: Insertar($Actividad)
34: Crear()
35: MostrarObjetivos()
36: MostrarActividades()
37: MostrarAcciones()
38: Crear()
39: ValidAccess($pUser, $pModule)
40: Crear()
41: setUsuario($pUser)
42: LoadAccess($pObjUser)
43: Crear()
44: DevolverAcceso($pUsuario)
45: Add($result)

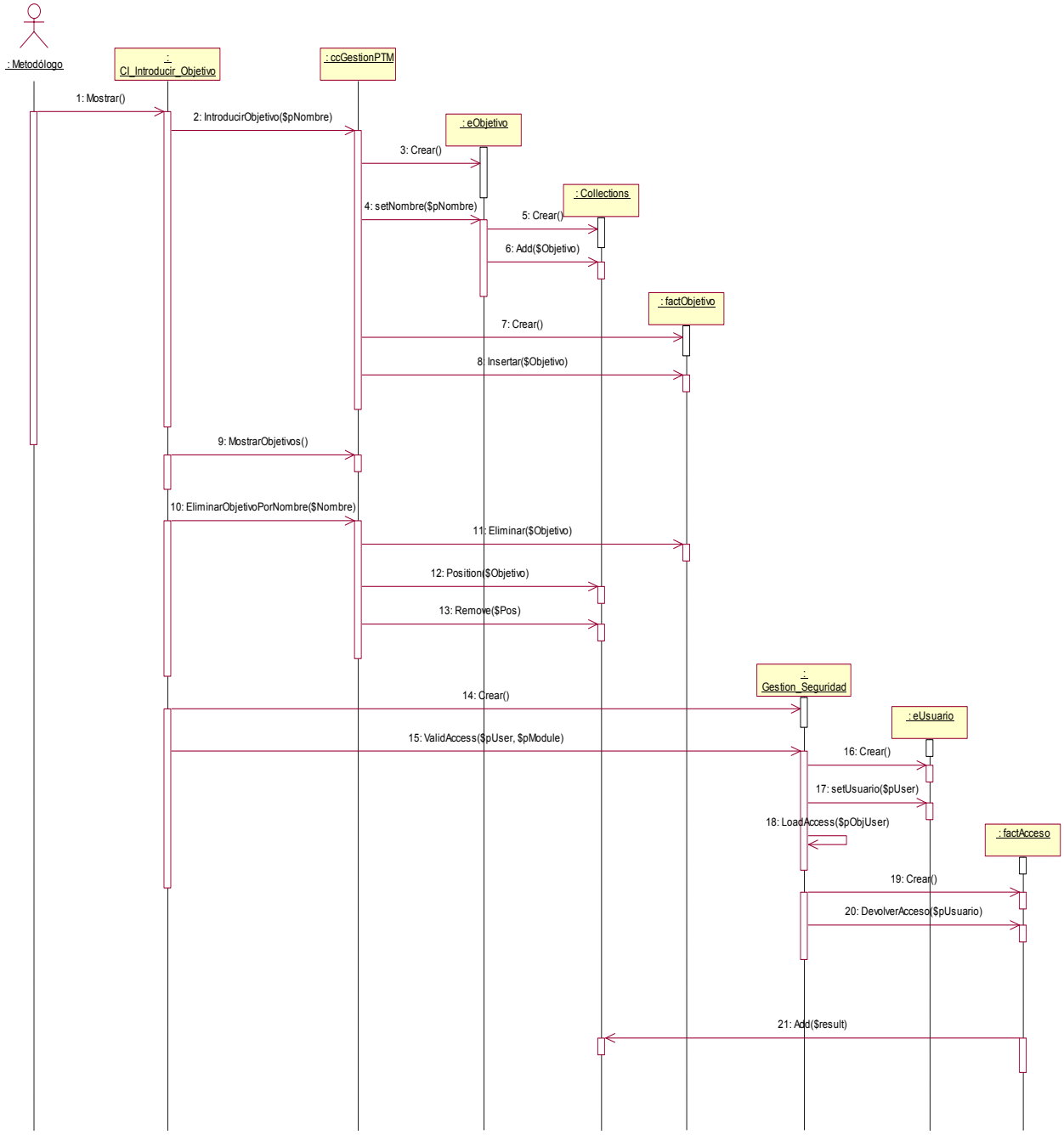
Anexo8


: Metodólogo
: Cl_Introducir_Introduccion
: ccGestionPTM
: ePTM
: Gestion_Seguridad
: eUsuario
: factAcceso
: Collections
1: Mostrar()
2: MostrarIntroduccion()
3: Crear()
4: getIntroduccion()
5: IntroducirIntroduccion($Introduccion)
6: setIntroduccion($Introduccion)
7: Crear()
8: ValidAccess($pUser, $pModule)
9: Crear()
10: setUsuario($pUser)
11: LoadAccess($pObjUser)
12: Crear()
13: DevolverAcceso($pUsuario)
14: Crear()
15: Add($result)

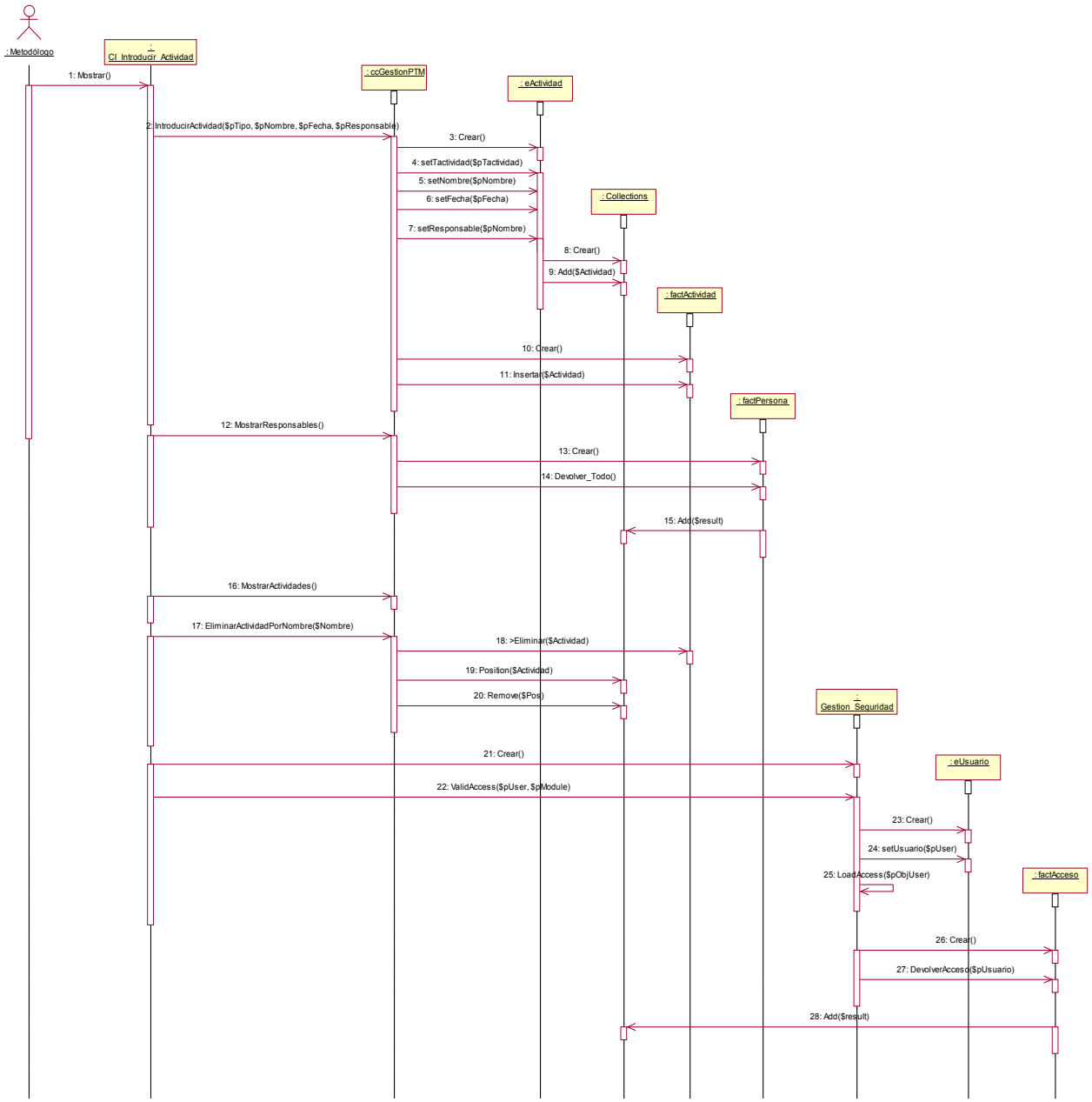
Anexo9


: Metodólogo
: CI_Introducir_Diag
: ccGestionPTM
: ePTM
: Gestion_Seguridad
: eUsuario
: factAcceso
: Collections
1: Mostrar()
2: MostrarDiagnostico()
3: Crear()
4: getDiagclaustro()
5: IntroducirDiagnostico($DiagClaustro)
6: setDiagclaustro($DiagClaustro)
7: Crear()
8: ValidAccess($pUser, $pModule)
9: Crear()
10: setUsuario($pUser)
11: LoadAccess($pObjUser)
12: Crear()
13: DevolverAcceso($pUsuario)
14: Crear()
15: Add($result)

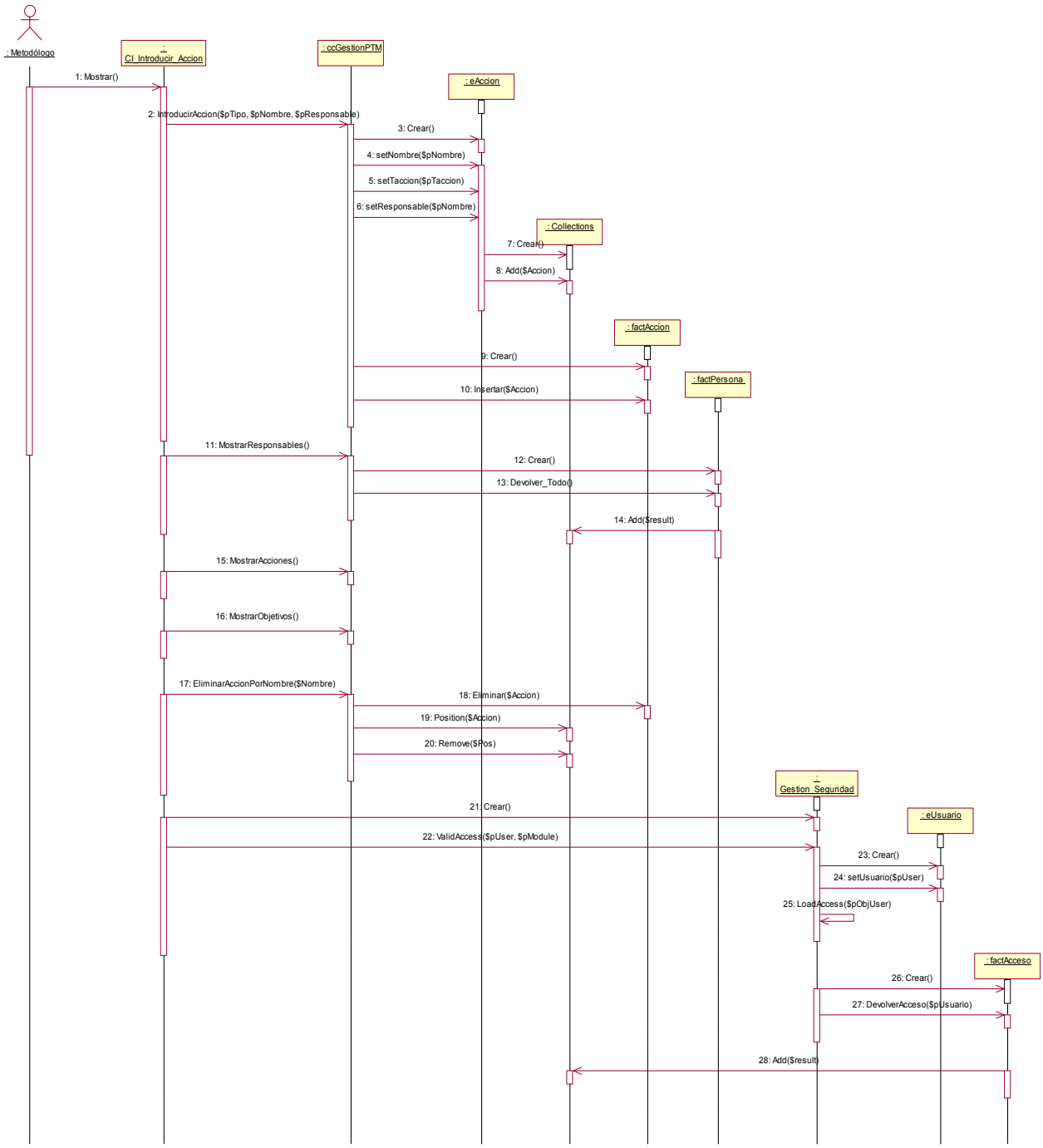
Anexo10


: Metodólogo
: CI_Introducir_Objetivo
: ccGestionPTM
: eObjetivo
: Collections
: factObjetivo
: Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: IntroducirObjetivo($pNombre)
3: Crear()
4: setNombre($pNombre)
5: Crear()
6: Add($Objetivo)
7: Crear()
8: Insertar($Objetivo)
9: MostrarObjetivos()
10: EliminarObjetivoPorNombre($Nombre)
11: Eliminar($Objetivo)
12: Position($Objetivo)
13: Remove($Pos)
14: Crear()
15: ValidAccess($pUser, $pModule)
16: Crear()
17: setUsuario($pUser)
18: LoadAccess($pObjUser)
19: Crear()
20: DevolverAcceso($pUsuario)
21: Add($result)

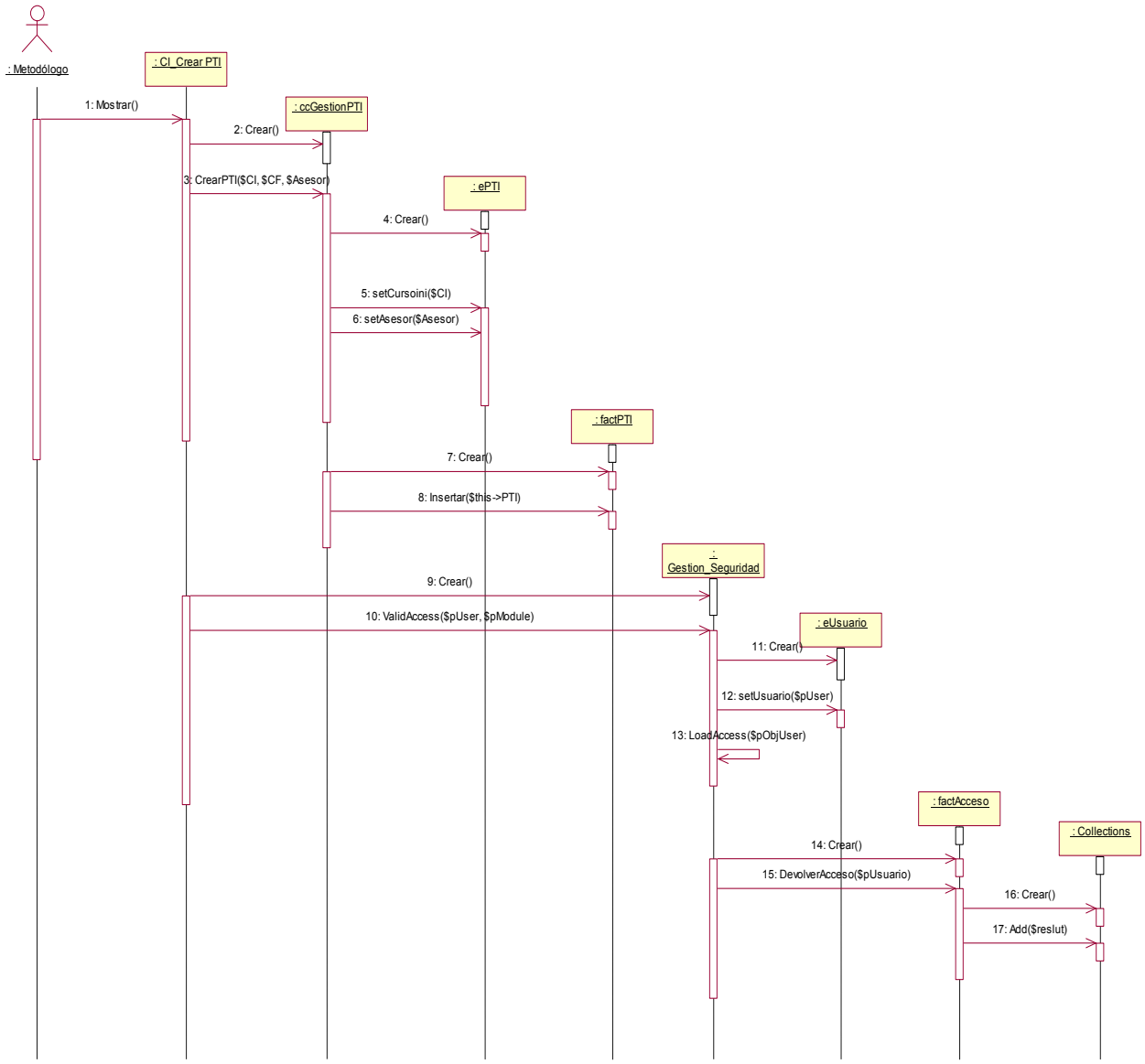
Anexo11


: Metodólogo
CI_Introducir_Actividad
: ccGestionPTM
: eActividad
: Collections
: factActividad
: factPersona
Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: IntroducirActividad($pTipo, $pNombre, $pFecha, $pResponsable)
3: Crear()
4: setTactividad($pTactividad)
5: setNombre($pNombre)
6: setFecha($pFecha)
7: setResponsable($pNombre)
8: Crear()
9: Add($Actividad)
10: Crear()
11: Insertar($Actividad)
12: MostrarResponsables()
13: Crear()
14: Devolver_Todo()
15: Add($result)
16: MostrarActividades()
17: EliminarActividadPorNombre($Nombre)
18: >Eliminar($Actividad)
19: Position($Actividad)
20: Remove($Pos)
21: Crear()
22: ValidAccess($pUser, $pModule)
23: Crear()
24: setUsuario($pUser)
25: LoadAccess($pObjUser)
26: Crear()
27: DevolverAcceso($pUsuario)
28: Add($result)

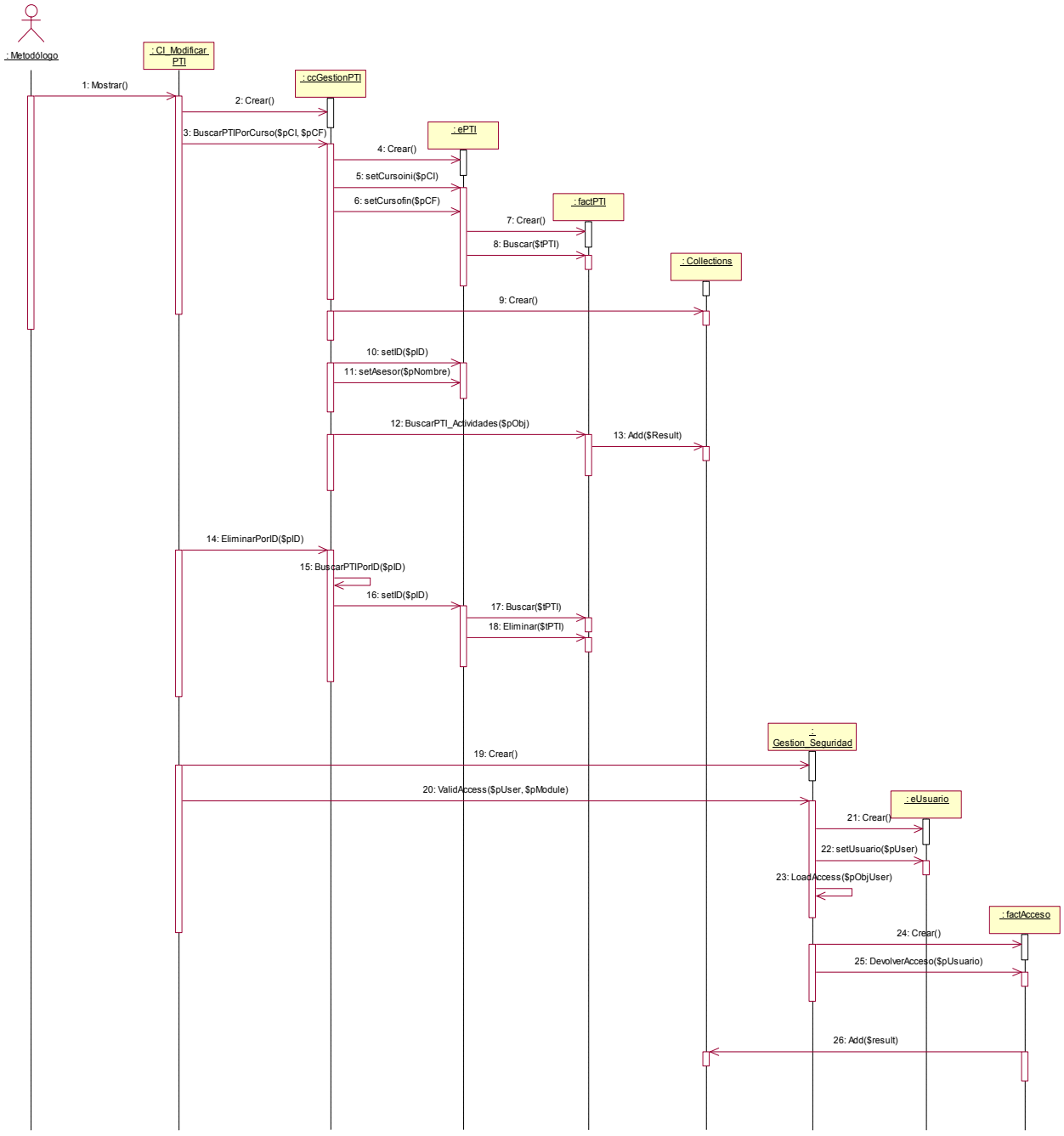
Anexo12


: Metodólogo
: CI_Introducir_Accion
: ccGestionPTM
: eAccion
: Collections
: factAccion
: factPersona
: Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: IntroducirAccion($pTipo, $pNombre, $pResponsable)
3: Crear()
4: setNombre($pNombre)
5: setTaccion($pTaccion)
6: setResponsable($pNombre)
7: Crear()
8: Add($Accion)
9: Crear()
10: Insertar($Accion)
11: MostrarResponsables()
12: Crear()
13: Devolver_Todo()
14: Add($result)
15: MostrarAcciones()
16: MostrarObjetivos()
17: EliminarAccionPorNombre($Nombre)
18: Eliminar($Accion)
19: Position($Accion)
20: Remove($Pos)
21: Crear()
22: ValidAccess($pUser, $pModule)
23: Crear()
24: setUsuario($pUser)
25: LoadAccess($pObjUser)
26: Crear()
27: DevolverAcceso($pUsuario)
28: Add($result)

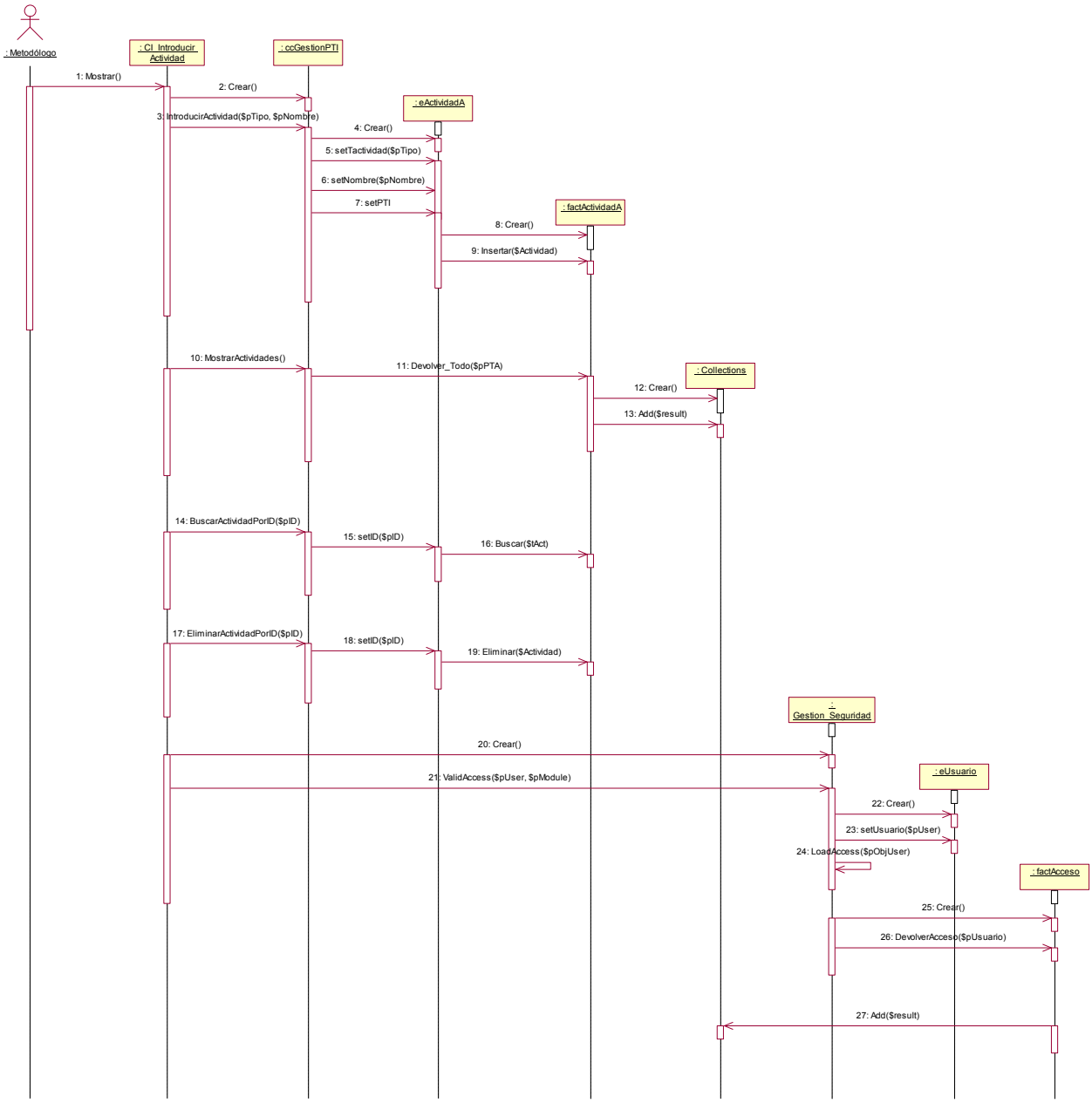
Anexo13


: Metodólogo
: Cl_Crear PTI
: ccGestionPTI
: ePTI
: factPTI
: Gestion_Seguridad
: eUsuario
: factAcceso
: Collections
1: Mostrar()
2: Crear()
3: CrearPTI($CI, $CF, $Asesor)
4: Crear()
5: setCursoini($CI)
6: setAsesor($Asesor)
7: Crear()
8: Insertar($this->PTI)
9: Crear()
10: ValidAccess($pUser, $pModule)
11: Crear()
12: setUsuario($pUser)
13: LoadAccess($pObjUser)
14: Crear()
15: DevolverAcceso($pUsuario)
16: Crear()
17: Add($reslut)

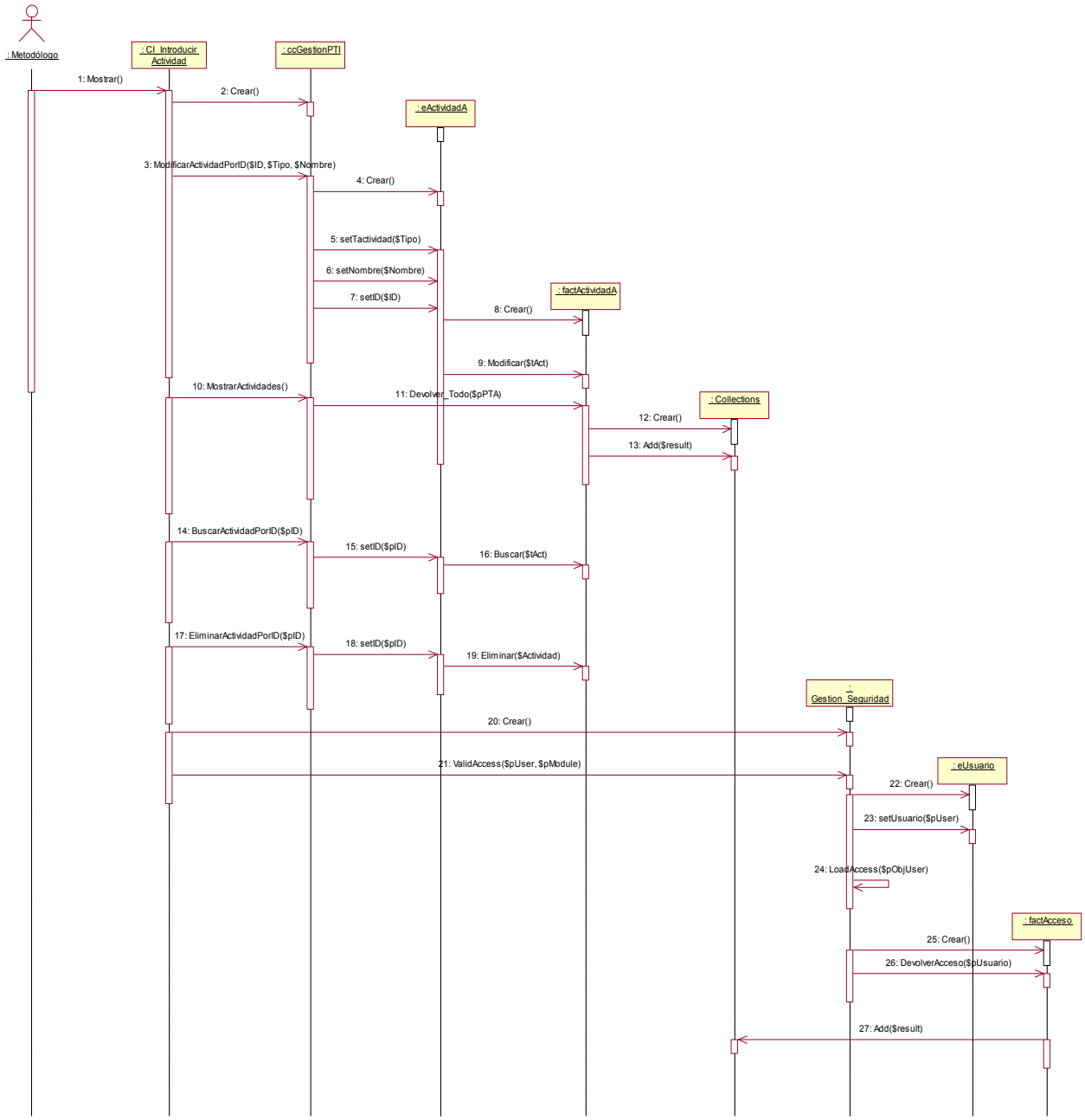
Anexo14


: Metodólogo
: Cl_Modificar PTI
: ccGestionPTI
: ePTI
: factPTI
: Collections
: Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: Crear()
3: BuscarPTIPorCurso($pCI, $pCF)
4: Crear()
5: setCursoini($pCI)
6: setCursofin($pCF)
7: Crear()
8: Buscar($tPTI)
9: Crear()
10: setID($pID)
11: setAsesor($pNombre)
12: BuscarPTI_Actividades($pObj)
13: Add($Result)
14: EliminarPorID($pID)
15: BuscarPTIPorID($pID)
16: setID($pID)
17: Buscar($tPTI)
18: Eliminar($tPTI)
19: Crear()
20: ValidAccess($pUser, $pModule)
21: Crear()
22: setUsuario($pUser)
23: LoadAccess($pObjUser)
24: Crear()
25: DevolverAcceso($pUsuario)
26: Add($result)

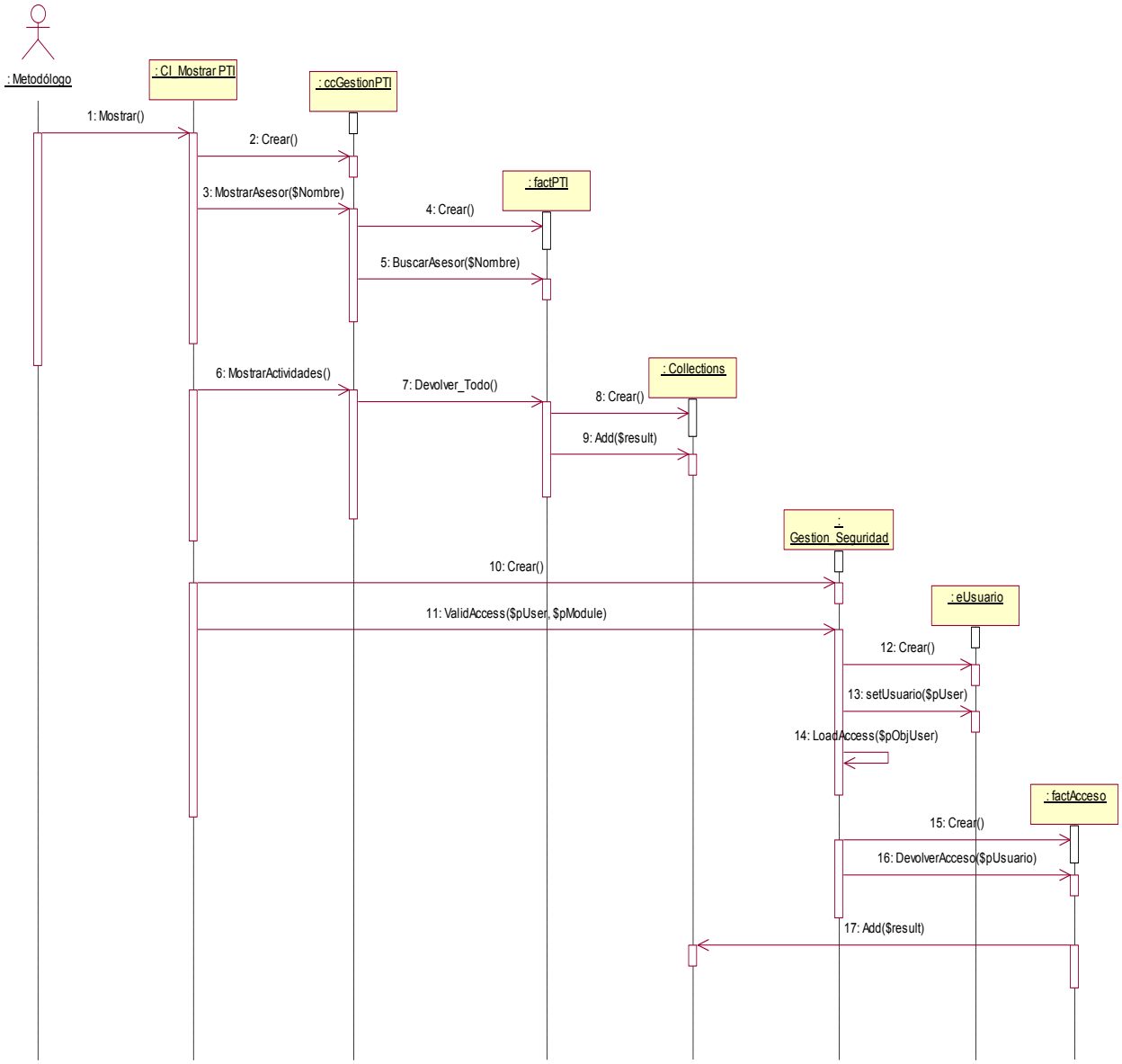
Anexo15


: Metodólogo
: CI_Introducir_Actividad
: ccGestionPTI
: eActividadA
: factActividadA
: Collections
: Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: Crear()
3: IntroducirActividad($pTipo, $pNombre)
4: Crear()
5: setTactividad($pTipo)
6: setNombre($pNombre)
7: setPTI
8: Crear()
9: Insertar($Actividad)
10: MostrarActividades()
11: Devolver_Todo($pPTA)
12: Crear()
13: Add($result)
14: BuscarActividadPorID($pID)
15: setID($pID)
16: Buscar($tAct)
17: EliminarActividadPorID($pID)
18: setID($pID)
19: Eliminar($Actividad)
20: Crear()
21: ValidAccess($pUser, $pModule)
22: Crear()
23: setUsuario($pUser)
24: LoadAccess($pObjUser)
25: Crear()
26: DevolverAcceso($pUsuario)
27: Add($result)

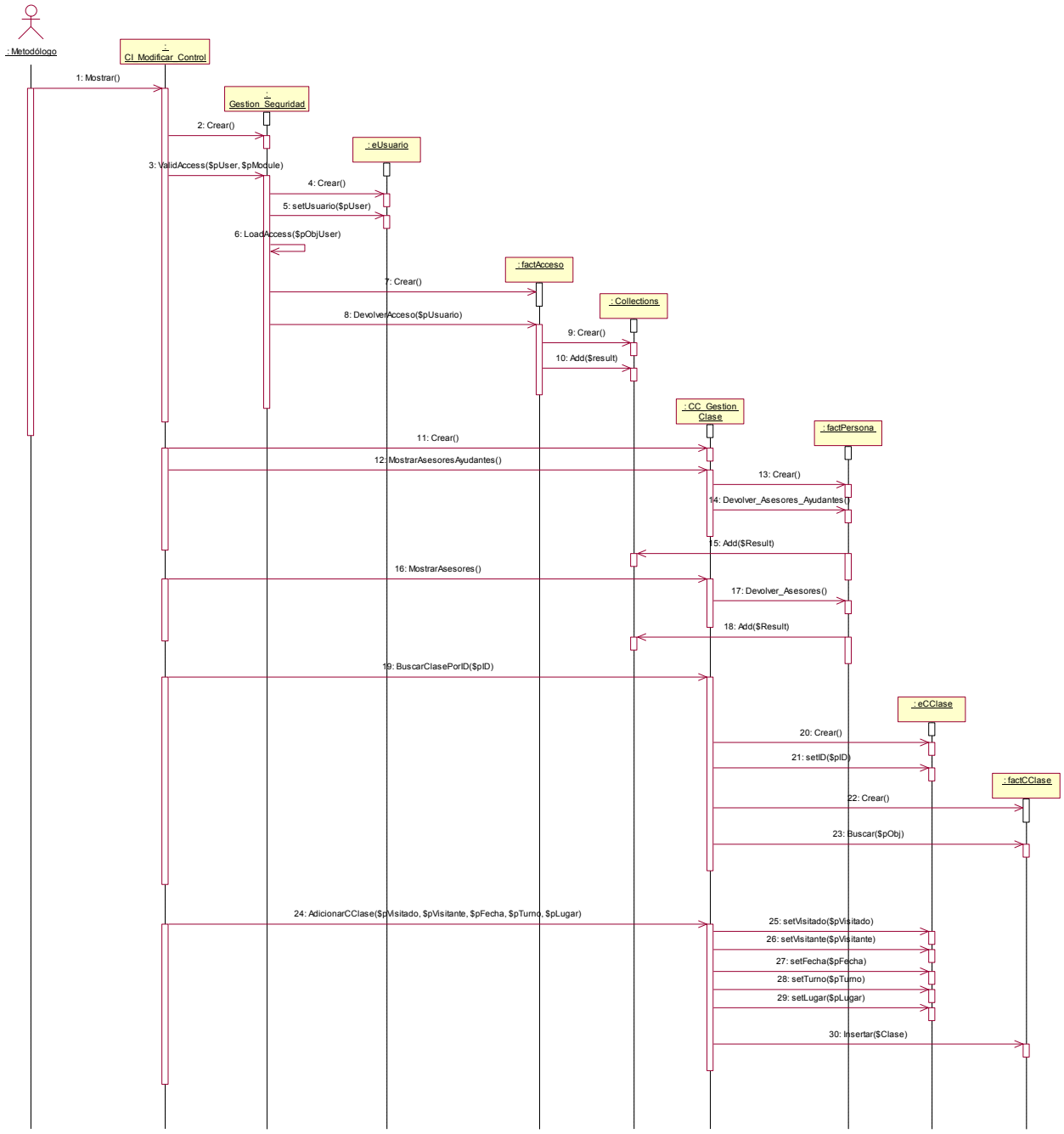
Anexo16


: Metodólogo
: CI_Introducir Actividad
: ccGestionPTI
: eActividadA
: factActividadA
: Collections
: Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: Crear()
3: ModificarActividadPorID($ID, $Tipo, $Nombre)
4: Crear()
5: setTactividad($Tipo)
6: setNombre($Nombre)
7: setID($ID)
8: Crear()
9: Modificar($tAct)
10: MostrarActividades()
11: Devolver_Todo($pPTA)
12: Crear()
13: Add($result)
14: BuscarActividadPorID($pID)
15: setID($pID)
16: Buscar($tAct)
17: EliminarActividadPorID($pID)
18: setID($pID)
19: Eliminar($Actividad)
20: Crear()
21: ValidAccess($pUser, $pModule)
22: Crear()
23: setUsuario($pUser)
24: LoadAccess($pObjUser)
25: Crear()
26: DevolverAcceso($pUsuario)
27: Add($result)

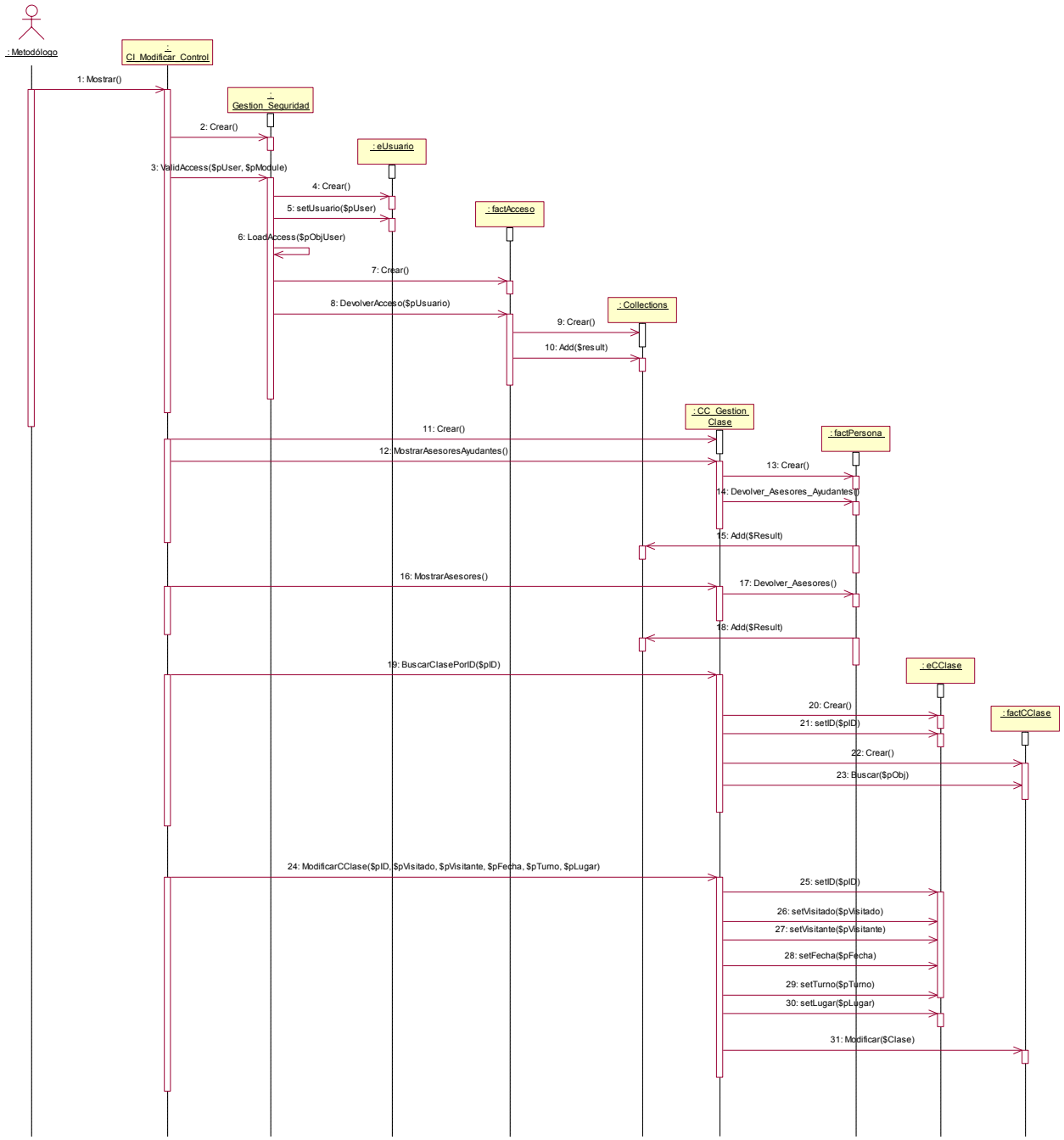
Anexo17


: Metodólogo
: Cl_Mostrar PTI
: ccGestionPTI
: factPTI
: Collections
: Gestion_Seguridad
: eUsuario
: factAcceso
1: Mostrar()
2: Crear()
3: MostrarAsesor($Nombre)
4: Crear()
5: BuscarAsesor($Nombre)
6: MostrarActividades()
7: Devolver_Todo()
8: Crear()
9: Add($result)
10: Crear()
11: ValidAccess($pUser, $pModule)
12: Crear()
13: setUsuario($pUser)
14: LoadAccess($pObjUser)
15: Crear()
16: DevolverAcceso($pUsuario)
17: Add($result)

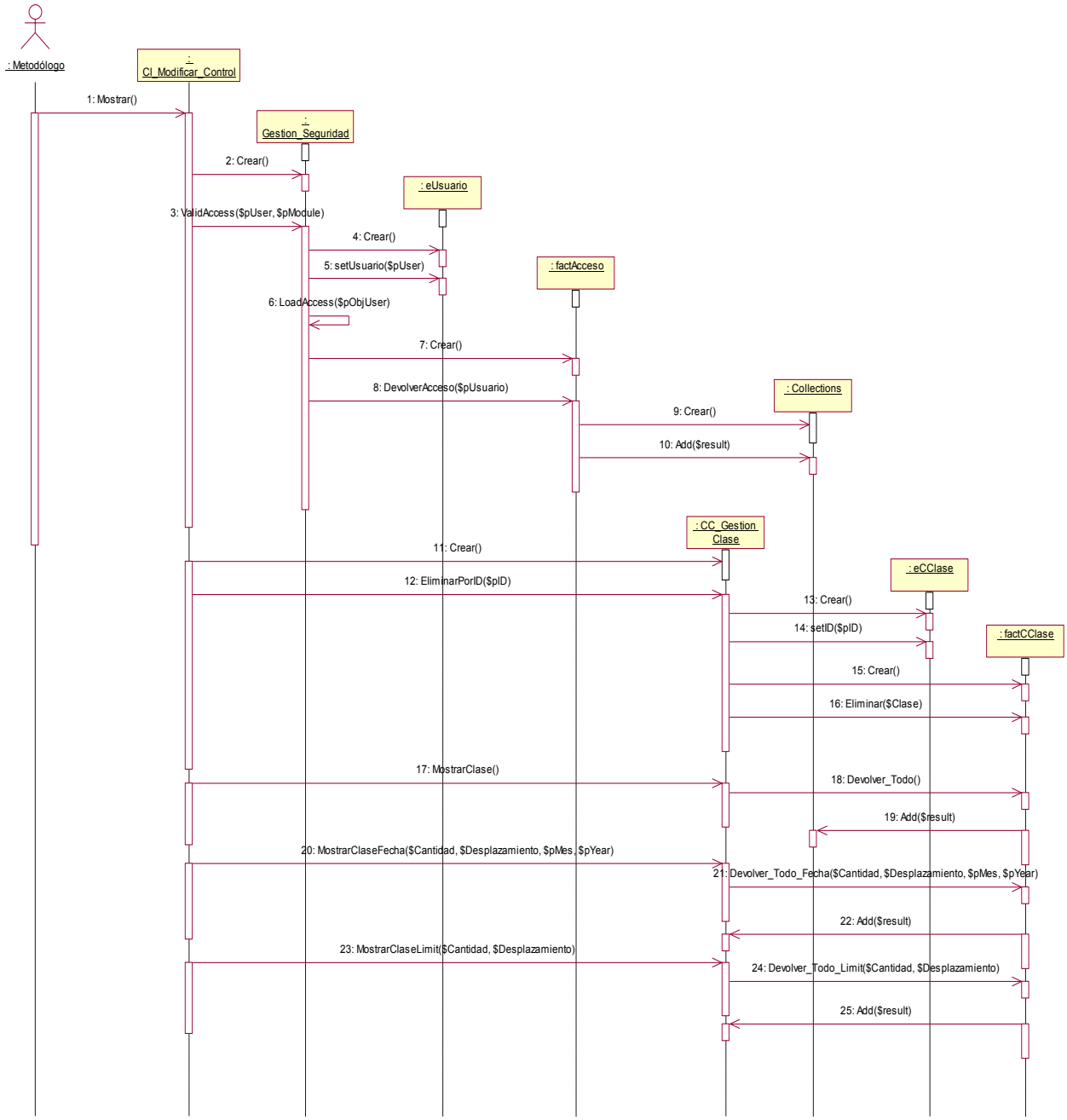
Anexo18


: Metodólogo
: CI_Modificar_Control
: Gestion_Seguridad
: eUsuario
: factAcceso
: Collections
: CC_Gestion_Clase
: factPersona
: eCClase
: factCClase
1: Mostrar()
2: Crear()
3: ValidAccess($pUser, $pModule)
4: Crear()
5: setUsuario($pUser)
6: LoadAccess($pObjUser)
7: Crear()
8: DevolverAcceso($pUsuario)
9: Crear()
10: Add($result)
11: Crear()
12: MostrarAsesoresAyudantes()
13: Crear()
14: Devolver_Asesores_Ayudantes()
15: Add($Result)
16: MostrarAsesores()
17: Devolver_Asesores()
18: Add($Result)
19: BuscarClasePorID($pID)
20: Crear()
21: setID($pID)
22: Crear()
23: Buscar($pObj)
24: AdicionarCClase($pVisitado, $pVisitante, $pFecha, $pTurno, $pLugar)
25: setVisitado($pVisitado)
26: setVisitante($pVisitante)
27: setFecha($pFecha)
28: setTurno($pTurno)
29: setLugar($pLugar)
30: Insertar($Clase)

Anexo19


: Metodólogo
: Cl_Modificar_Control
: Gestion_Seguridad
: eUsuario
: factAcceso
: Collections
: CC_Gestion_Clase
: factPersona
: eCClase
: factCClase
1: Mostrar()
2: Crear()
3: ValidAccess($pUser, $pModule)
4: Crear()
5: setUsuario($pUser)
6: LoadAccess($pObjUser)
7: Crear()
8: DevolverAcceso($pUsuario)
9: Crear()
10: Add($result)
11: Crear()
12: MostrarAsesoresAyudantes()
13: Crear()
14: Devolver_Asesores_Ayudantes()
15: Add($Result)
16: MostrarAsesores()
17: Devolver_Asesores()
18: Add($Result)
19: BuscarClasePorID($pID)
20: Crear()
21: setID($pID)
22: Crear()
23: Buscar($pObj)
24: ModificarCClase($pID, $pVisitado, $pVisitante, $pFecha, $pTurno, $pLugar)
25: setID($pID)
26: setVisitado($pVisitado)
27: setVisitante($pVisitante)
28: setFecha($pFecha)
29: setTurno($pTurno)
30: setLugar($pLugar)
31: Modificar($Clase)

Anexo20


: Metodólogo
: CI_Modificar_Control
: Gestion_Seguridad
: eUsuario
: factAcceso
: Collections
: CC_Gestion Clase
: eCClase
: factCClase
1: Mostrar()
2: Crear()
3: ValidAccess($pUser, $pModule)
4: Crear()
5: setUsuario($pUser)
6: LoadAccess($pObjUser)
7: Crear()
8: DevolverAcceso($pUsuario)
9: Crear()
10: Add($result)
11: Crear()
12: EliminarPorID($pID)
13: Crear()
14: setID($pID)
15: Crear()
16: Eliminar($Clase)
17: MostrarClase()
18: Devolver_Todo()
19: Add($result)
20: MostrarClaseFecha($Cantidad, $Desplazamiento, $pMes, $pYear)
21: Devolver_Todo_Fecha($Cantidad, $Desplazamiento, $pMes, $pYear)
22: Add($result)
23: MostrarClaseLimit($Cantidad, $Desplazamiento)
24: Devolver_Todo_Limit($Cantidad, $Desplazamiento)
25: Add($result)

GLOSARIO DE TÉRMINOS

Artefacto: Producto tangible del proyecto que es producido, modificado y usado por las actividades.

Automatización: Realización de una combinación específica de acciones por una máquina, sin la ayuda de personas.

Bytecodes (código de bytes): Es un conjunto de instrucciones, altamente optimizado, diseñado para ser ejecutado por una maquina virtual que emula al intérprete Java.

Código abierto (Open source): Relativo al software para el cual el código fuente está disponible en forma gratuita.

Código fuente: Es un texto escrito generalmente por una persona que se utiliza como base para generar otro código con un compilador o intérprete para ser ejecutado por una computadora.

Componente: Es la unidad de construcción elemental del diseño físico.

Depuración: Es el proceso de identificar y corregir errores de programación.

Descriptor: Estructura de datos que representa el perfil de documento, una descripción de clase de objeto, un estilo de disposición, un estilo de presentación o una descripción de objeto.

E-commerce: Aplicación informática dirigida a realizar transacciones comerciales (sitio web de comercio electrónico).

Full – cycle: Ciclo completo.

FTP (File Transfer Protocol): Permite recibir y enviar cualquier clase de fichero desde o hacia cualquier ordenador que éste conectado a Internet.

Interfaz de usuario: Es la forma en que los usuarios pueden comunicarse con un computador, y comprende todos los puntos de contacto entre el usuario y el equipo.

Java: Lenguaje de programación diseñado para Internet, utilizado para crear aplicaciones completas o pequeñas aplicaciones (applets) para ser insertados en una página Web.

Modelo Vista Controlador (MVC): Es una arquitectura de software que separa el modelo de datos de una aplicación, la interfaz de usuario, y la lógica de control en tres distintos componentes de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos.

Multiplataforma: Poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

Negocio: Cualquier ambiente o entorno en cual esta enmarcado el problema.

Patrones: Describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de la solución al problema, de forma que puede reutilizarse continuamente.

Requerimientos: Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

Reutilizar código: Es la acción de volver a utilizar el código. La utilidad puede venir para el usuario mediante una acción de mejora o restauración, o sin modificar el código si es útil para un nuevo usuario.

Scripting: Lenguaje interpretado a un lenguaje de programación que fue diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados.

SFTP (Secure File Transfer Protocol): Es una forma segura de transferir ficheros a un servidor ya que los datos circulan encriptados por la red.

Sintaxis: Reglas que gobiernan la estructura de un lenguaje.

Sistema Informático: Es aquel sistema que se encarga del manejo de información en la computadora, a través de la cual el usuario controla las operaciones que realiza el procesador.

Sistema de Gestión: Es el conjunto de políticas y normas relacionadas entre sí que se establecen para el acceso y tratamiento de los recursos de información. Incluye los registros administrativos y los archivos, el soporte tecnológico de los recursos y el público a que se destina.

Sistema Gestor de Contenido: Permite la creación y administración de contenidos principalmente en páginas web.

Software: Conjunto de instrucciones escritas en un determinado lenguaje, que dirigen a un ordenador para la ejecución de una serie de operaciones, con el objetivo de resolver un problema que se ha definido previamente.

Software libre: Aplicaciones informáticas que pueden ser libremente copiadas, distribuidas, estudiadas y modificadas por el usuario, según los parámetros establecidos por su creador.

WebDAV: Es un conjunto de extensiones HTTP independientes de la plataforma, que permiten a los usuarios editar y administrar archivos colaborativamente en servidores Web remotos.