UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"HERRAMIENTA DE ADMINISTRACIÓN DE PROYECTOS PARA MOPROSOFT".

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRA EN INGENIERÍA

(COMPUTACIÓN)

PRESENTA:

BRENDA DANIELA TORRES CASTILLO

DIRECTORA DE TESIS:

M. EN C. MA. GUADALUPE ELENA

IBARGÜENGOITIA GONZÁLEZ

MÉXICO, D. F. **2008**

Gracias, Mamá, Papá, Juan Eduardo y Tía Yolis,
su apoyo incondicional,
es pieza clave de este logro, que también es suyo.

Gracias, Carlos por alegrar mi corazón,
alentarme, saber que nunca lucharé sin ayuda,
y saber que siempre estarás cerca de mí.

Gracias, maestra Lupita por todas sus valiosas enseñanzas,
paciencia y cariño, gracias por todo su apoyo, pero sobretodo por la
oportunidad de aprender de su experiencia.

Gracias, al Posgrado en Ciencia e Ingeniería de la Computación, UNAM,
maestros, compañeros y amigos,
por la gran experiencia que vivimos.

Gracias, a mis sinodales, Dra. Hanna Oktaba, Ana Briseño,
Gloria Quintanilla y al Dr. Fernando Gamboa,
por sus valiosas enseñanzas.

Gracias, a la UNAM por ser parte fundamental
de mi formación académica pero sobretodo humana.

Gracias, al CONACYT por el apoyo que me brindó,
para poder realizar mis estudios de posgrado.

Brenda Daniela Torres Castillo

Cierro mis ojos,
solo por un momento y los momentos se han ido,
todos mis sueños pasan ante mis ojos,
una curiosidad.

Polvo en el viento,
todo lo que son es polvo en el viento,
la misma vieja canción,
solo una gota de agua en el mar infinito.

Todo lo que hacemos,
se desmorona en el suelo,
aunque nos neguemos a ver.

Polvo en el viento,
todo lo que somos es polvo en el viento.

Ahora resiste,
nada dura para siempre,
excepto la tierra y el cielo.

Se escabulle,
y todo tu dinero no comprará otro minuto.

Polvo en el viento,
todo lo que somos es polvo en el viento.

Dust in the Wind ♪
Kansas ♫

Este año ha sido inolvidable, porque el cielo y nosotros ganamos tres hermosos angelitos Abue, Tía María y Tío Polín, la vida sin ustedes cambió en muy poco tiempo, pero sé que están juntos y siempre nos cuidan. Este logro es para ustedes.

Brenda Daniela

Índice

Índice de Tablas

Índice de Figuras

Introducción

El objetivo de este trabajo de tesis es presentar el desarrollo de HeAP MoProSoft (Herramienta de Administración de Proyectos para MoProSoft), que consistirá en una aplicación Web que formará parte del software libre.

Este documento está estructurado en cuatro capítulos, que describiremos a continuación:

Capítulo 1. Antecedentes

En el primer capítulo hablaremos de los dos antecedentes de este trabajo, el primero de ellos es la administración de proyectos de software y el segundo es MoProSoft (Modelo de Procesos para la Industria de Software) [1], siendo estos los que motivaron la realización de esta tesis.

Capítulo 2. Investigación Tecnológica

En el capítulo 2 se encuentra el resultado de la investigación de las tecnologías que utilizamos para desarrollar HeAP MoProsoft, que incluye J2EE (Java 2 Enterprise Edition), el patrón de diseño MVC (Modelo-Vista-Controlador) y el marco de trabajo Struts.

Capítulo 3. Justificación de la Herramienta

El capítulo 3 contiene una breve investigación sobre las herramientas de administración de proyectos utilizadas actualmente, con el fin de compararlas y realizar la justificación para el desarrollo de esta herramienta, este capítulo contiene también la descripción del problema tratado.

Capítulo 4. Desarrollo de la Herramienta

En el capítulo 4 se presenta el desarrollo de HeAP MoProSoft siguiendo el Proceso Unificado de Desarrollo de Software [8].

Conclusiones

Finalmente presentamos las conclusiones de este trabajo, ventajas y desventajas de la herramienta, así como las sugerencias para trabajos futuros.

Capítulo 1. Antecedentes

Introducción

Este trabajo de tesis tiene dos antecedentes muy importantes, el primero de ellos es la Administración de Proyectos de Software y el segundo es MoProSoft (Modelo de Procesos para la Industria de Software) [1], que al ser aprobado como norma mexicana para el desarrollo de software, en agosto del 2005, se convirtió en una guía importante para las PyMES (Pequeñas y Medianas Empresas) de la industria de desarrollo y/o mantenimiento de software, por lo tanto es indispensable contar con herramientas que apoyen su mejor comprensión e implementación.

1.1 Administración de Proyectos de Software

La administración de proyectos de software se ha convertido hoy en día en una gran necesidad para los desarrollares de software debido a su gran complejidad. El hacer tangible lo intangible, es decir, a diferencia de otros proyectos que tienen por objetivo la construcción de un edificio o un automóvil, en los proyectos de desarrollo de software el avance no es visible inmediatamente, lo que dificulta mucho más su correcta administración.

Lo primero que haremos será dar la definición de proyecto y administración de proyectos de forma general y después realizaremos la distinción entre éstos, los proyectos de software y la administración de proyectos de software de forma específica.

¿Qué es un proyecto?

El PMI (Project Management Institute) define proyecto como *"un esfuerzo temporal emprendido para crear un producto o servicio único"* [2]. Partiendo de esta definición podemos analizar varios puntos: el primero de ellos es que un proyecto es *temporal* lo que significa que tiene un inicio y fin bien definidos, un proyecto se termina cuando los objetivos del proyecto son alcanzados. El segundo punto es que un proyecto es *único* porque nunca volveremos a hacer exactamente lo mismo, con la misma gente y ambiente, además de crear un producto o servicio único.

Otra de las características de un proyecto es que su elaboración es progresiva, se siguen un conjunto de pasos durante los cuales se van generando incrementos, por lo que debe de ir

acompañado de una correcta definición del alcance del proyecto y la definición de los pasos a realizar.

Vale la pena mencionar la diferencia entre un proyecto y un trabajo del tipo operacional: un proyecto es temporal y único mientras que el trabajo operacional es de tipo continuo y repetitivo, aunque es cierto que comparten muchas similitudes como: son realizados por gente, se encuentran restringidos en recursos, son planeados, ejecutados y controlados.

¿Qué es la administración de proyectos?

La definición que da el PMI para la administración de proyectos es *"la aplicación del conocimiento, habilidades, herramientas y técnicas para planear las actividades que satisfagan los requisitos de un proyecto"* [2]. La administración de proyectos es realizada a través de la aplicación e integración de los pasos de administración de proyectos de inicio, planeación, ejecución, control y monitoreo y cierre.

Ciclo de Vida de un Proyecto

En el PMBOK se identifican cinco grupos procesos, que no son secuenciales y que constituyen el ciclo de vida del proyecto, estos procesos son:

- En el proceso de *inicio* se autoriza el proyecto, se identifica la necesidad y se describe.
- En el proceso de *planeación* se definen los objetivos, se selecciona la mejor alternativa para dirigir el proyecto y alcanzar sus objetivos.
- En el proceso de *ejecución*, se coordina a la gente o a otros recursos para llevar a cabo el plan.
- En el proceso de *control y monitoreo*, nos aseguramos que los objetivos del proyecto se realicen monitoreándolos y haciendo mediciones del proceso para identificar variaciones de nuestro plan y tomar acciones correctivas en caso de ser necesario.
- En el proceso de *cierre* se formaliza la aceptación del proyecto y llevarlo a un término ordenado.

La definición de proyecto y administración de proyectos vistas anteriormente son muy generales y aplicables a cualquier tipo de proyecto, en nuestro caso nos enfocaremos en los proyectos de software y en la administración de proyectos de software.

¿Qué es la administración de proyectos de software?

Para poder explicar a qué nos referimos cuando hablamos de administración de proyectos de software lo primero que debemos hacer es definir qué es un proyecto de software.

Proyecto de Software

Un proyecto de software crea productos o servicios que además de ser únicos poseen las siguientes características, que los hacen diferentes del resto de los proyectos:

- *Invisibilidad*, el progreso en este tipo de proyectos no es visible inmediatamente.

- *Complejidad*, los productos de software son más complejos que otros productos generados por otros tipos de proyectos, porque se trata de productos intangibles.
- *Conformidad*, los desarrolladores de software deben cumplir con requerimientos poco claros y muchas veces incompletos por parte de los clientes, porque en muchas ocasiones ni el cliente tiene claro lo que quiere y lo que necesita.
- *Flexibilidad*, los sistemas de software son propensos a sufrir grandes cambios.

Estas características traen consigo una serie de problemas que se deben tomar en cuenta al momento de administrar este tipo de proyectos, estos problemas los listamos a continuación:

- Malas estimaciones y planes mal elaborados.
- Falta de medidas y estándares de calidad
- Falta de orientación en la toma de decisiones en la organización
- Falta de técnicas para que el progreso del proyecto sea visible
- Mala o deficiente definición de roles
- Criterios de aceptación incorrectos
- Inadecuada especificación del trabajo
- Los administradores de proyectos ignoran a las tecnologías de la información
- Falta de conocimiento de las áreas de aplicación
- Falta de estándares
- Falta de documentación actualizada
- Actividades con retraso
- Falta de comunicación entre los desarrolladores técnicos y los usuarios
- Falta de comunicación con el líder de proyectos, lo que puede provocar trabajo duplicado
- Falta de compromiso
- Mala definición del alcance por parte de los expertos técnicos
- Cambios en los requisitos
- Cambios al ambiente de software
- Presión para una tener una fecha límite
- Falta de control de calidad
- Administración remota
- Falta de capacitación

Muchos de estos problemas son originados por falta de comunicación entre el equipo de desarrollo y el usuario o entre el líder de proyecto y el equipo de trabajo, etc. El origen del resto de los problemas es la falta de conocimiento en las áreas de TI (Tecnologías de la Información) lo que provoca una mala implementación de estas técnicas y tecnologías.

Lo anterior nos permite observar que para administrar un proyecto de software se debe tener en cuenta que, un proyecto de software posee dos dimensiones: el área de ingeniería y la administración de proyectos. La ingeniería se enfoca en la construcción del sistema de software por lo tanto centra sus esfuerzos en el diseño, pruebas, codificación, entre otras. Mientras que la administración de proyectos se ocupa de la planeación, ejecución y el monitoreo y control de las actividades de ingeniería para satisfacer los objetivos del proyecto en costo, tiempo y alcance.

Por lo tanto la administración de proyectos de software es *"la aplicación del conocimiento, habilidades, herramientas y técnicas para planear las actividades que satisfagan los requisitos de un proyecto de software"*.

Muchas de las técnicas de administración de proyectos son aplicables a la administración de proyectos de software, pero debido a las características de este tipo de proyectos fue necesario desarrollar un marco de referencia especialmente diseñado para la administración de proyectos de software dirigido a la industria mexicana, MoProSoft (Modelo de Procesos para la Industria de Software) [1] fue creado con el propósito de ofrecer las mejores prácticas para la administración de proyectos de software.

1.2 MoProSoft

(Modelo de Procesos para la Industria de Software)

MoProSoft tiene como objetivo brindar a las PyMES dedicadas al desarrollo y/o mantenimiento de software un marco de referencia basado en las mejores prácticas para la administración de proyectos de software, el modelo está basado en procesos considerando la estructura de una empresa: Alta Dirección, Gerencia y Operación las cuales conforman las tres categorías de procesos del modelo.

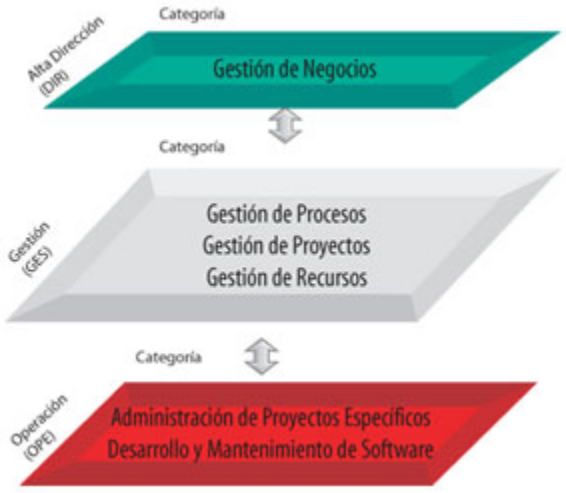


Figura 1.1 Estructura de MoProSoft

En la categoría de Alta Dirección está contenido el proceso de Gestión de Negocio.

La categoría de Gerencia se integra con los procesos de Gestión de Procesos, Gestión de Proyectos y Gestión de Recurso integrado por los siguientes subprocesos: Recursos Humanos, Ambiente de Trabajo, Bienes, Servicios e Infraestructura y Conocimiento de la Organización.

La categoría de Operación contiene los procesos de Administración de Proyectos Específicos y Desarrollo y Mantenimiento de Software.

Cada proceso es desempeñado por uno o varios integrantes de la organización y a cada uno se le asigna un rol específico dependiendo de sus habilidades, el grado de capacitación, y las necesidades del proyecto.

Categorías y sus Procesos [1]

1. Categoría de Alta Dirección

La categoría de Alta Dirección contiene las prácticas relacionadas con la gestión del negocio, además de proporcionar los lineamientos que seguirá la categoría de Gerencia.

1.1 Proceso Gestión de Negocio

El propósito de este proceso es establecer la razón de ser de la organización, sus objetivos y lineamientos, y las condiciones para lograrlos, tomando en cuenta las necesidades de los clientes, analizando los resultados con el fin de proponer cambios y realizar una mejora continua. De esta forma se prepara a la organización para las situaciones de cambio y a su personal para trabajar conforme a los objetivos establecidos.

2. Categoría de Gerencia

Esta categoría incluye las prácticas de gestión de procesos, proyectos y recursos siguiendo los lineamientos dictados por la Categoría de Alta Dirección. A su vez se establecen los lineamientos que seguirá la Categoría de Operación, de la cual se recibe y evalúa información, los resultados obtenidos serán enviados a la Categoría de Alta Dirección.

2.1 Gestión de Procesos

El propósito de Gestión de Procesos es establecer los procesos de la organización, en función de los *Procesos Requeridos* identificados en el *Plan Estratégico*. Así como definir, planificar e implantar las actividades de mejora en los mismos.

2.2 Gestión de Proyectos

El propósito de la Gestión de Proyectos es asegurar que los proyectos contribuyan al cumplimiento de los objetivos y estrategias de la organización.

2.3 Gestión de Recursos

El propósito de Gestión de Recursos es conseguir y dotar a la organización de los recursos humanos, infraestructura, ambiente de trabajo y proveedores, así como crear y mantener la *Base de Conocimiento* de la organización. La finalidad es apoyar el cumplimiento de los objetivos del *Plan Estratégico* de la organización.

2.3.1 Recursos Humanos y Ambiente de Trabajo

El propósito de Recursos Humanos y Ambiente de Trabajo es proporcionar los recursos humanos adecuados para cumplir las responsabilidades asignadas a los roles dentro de la organización, así como la evaluación del ambiente de trabajo.

2.3.2 Bienes, Servicios e Infraestructura

El propósito de Bienes, Servicios e Infraestructura es proporcionar, identificar, seleccionar, proveedores de bienes, servicios e infraestructura que satisfagan los requisitos de adquisición de los procesos y proyectos.

2.3.3 Conocimiento de la Organización

El propósito de Conocimiento de la Organización es mantener disponible y administrar la Base de Conocimiento que contiene la información y los productos generados por la organización.

3. Categoría de Operación

En esta categoría se llevan a cabo las prácticas necesarias para el desarrollo y mantenimiento de software de acuerdo con los lineamientos que se establecieron en la Categoría de Gerencia, a la que se le entrega la información y los productos generados.

3.1 Administración de Proyectos Específicos

El propósito de la Administración de Proyectos Específicos es establecer y llevar a cabo sistemáticamente las actividades que permitan cumplir con los objetivos de un proyecto en tiempo y costo esperados.

3.2 Desarrollo y Mantenimiento de Software

El propósito de Desarrollo y Mantenimiento de Software es la realización sistemática de las actividades de análisis, diseño, construcción, integración y pruebas de productos de software, nuevos o modificados cumpliendo con los requerimientos especificados.

Roles

Como se mencionó anteriormente, en MoProSoft se tienen definidos diferentes roles que se describen a continuación:

Rol	Abreviatura	Descripción
Cliente	CL	Es quien solicita el producto de software y financia el proyecto para su desarrollo o mantenimiento.
Usuario	US	Es el que utilizará el producto de software.
Grupo Directivo	GD	Es el encargado de dirigir la organización y son los responsables de su funcionamiento exitoso.
Responsable de Proceso	RP	Es el encargado de que se lleven a cabo las prácticas de un proceso y del cumplimiento de sus objetivos.
Involucrado	IN	Se refiere a otros roles que pueden ser requeridos por sus habilidades para realizar actividades o tareas específicas. Por ejemplo: Analista, Programador, entre otros.
Responsable de Gestión de Negocio	RGN	Es quien define e implanta exitosamente el proceso de Gestión de Negocio y asegura se realice con éxito.
Responsable de Gestión de Proceso	RGP	Es el encargado de definir e implantar exitosamente el proceso de Gestión de Procesos.
Responsable de Gestión de Proyectos	RGPY	Es el encargado de realizar gestión de proyectos.
Responsable de la Administración del Proyecto Específico	RAPE	Es el que realizará la administración de los proyectos específicos por lo que deberá contar con: capacidad de liderazgo con experiencia en la toma de decisiones, planificación estratégica, manejo de personal, delegación y supervisión, finanzas y desarrollo de software.
Grupo de Gestión	GG	Es el encargado de realizar las propuestas de mejora al Plan Estratégico e implantar los procesos definidos.
Responsable de Gestión de Recursos	RGR	Es el encargado de definir e implantar exitosamente el proceso de Gestión de Recursos y sus subprocesos.
Responsable de Recursos Humanos y Ambiente de Trabajo	RRHAT	Debe tener el conocimiento de las actividades necesarias para implantar exitosamente el subproceso de Recursos Humanos y Ambiente de Trabajo.
Responsable de Bienes, Servicios e Infraestructura	RBSI	Debe tener el conocimiento de las actividades necesarias para implantar exitosamente el subproceso de de Bienes, Servicios e Infraestructura.

Rol	Abreviatura	Descripción
Responsable de Conocimiento de la Organización	RCO	Es el encargado de garantizar la integridad, seguridad y eficiencia de la Base de Conocimiento.
Responsable de Capacitación	RC	Es el encargado de implantar la capacitación solicitada.
Equipo de Trabajo	ET	El Equipo de Trabajo debe contar con conocimiento y experiencia de acuerdo a su rol.
Grupo de Responsables de Procesos	GRP	Debe conocer las necesidades del proceso con respecto a la Base de Conocimiento.
Responsable del Subcontrato	RSC	Es el encargado de administrar los subcontratos por lo que debe tener conocimiento de administración de proyectos.
Responsable de Desarrollo y Mantenimiento de Software	RDM	Es el encargado del proceso de desarrollo y mantenimiento de software por lo tanto debe contar con conocimiento y experiencia en este proceso.
Analista	AN	Es el encargado de la obtención, especificación y análisis de los requerimientos, debe tener conocimiento y experiencia.
Diseñador de Interfaz de Usuario	DU	Es el encargado de realizar el diseño de las interfaces de usuario, debe tener conocimientos de diseño y criterios ergonómicos para las interfaces de usuario.
Diseñador	DI	Es el encargado del diseño de la estructura de los componentes de software.
Programador	PR	Es el encargado de la programación, integración y de realizar las pruebas unitarias.
Responsable de Pruebas	RPU	Es el encargado de la planificación y realización de las pruebas de integración y de sistema.
Revisor	RE	Es el encargado de aplicar las técnicas de revisión y debe tener experiencia en el desarrollo y mantenimiento de software.
Responsable de Manuales	RM	Es el encargado de generar los manuales por lo tanto debe tener experiencia en redacción en el desarrollo y mantenimiento de software.
Evaluador	EV	Es el encargado de aplicar las evaluaciones, debe contar con conocimiento en metodologías de evaluación.

Tabla 1.1 Roles de MoProSoft

Capítulo 2. Investigación Tecnológica

Introducción

En este capítulo veremos el resultado de la investigación de las tecnologías que decidimos utilizar para el desarrollo de HeAP MoProSoft. El criterio fundamental para realizar esta elección fue a partir de la restricción del sistema, que consiste en que debe ser una aplicación Web y formar parte del software de distribución libre, por lo tanto elegimos desarrollar la aplicación en la plataforma J2EE[1] (Java 2 Enterprise Edition) lo que implica utilizar tecnologías basadas en el lenguaje Java[1] para desarrollo Web, como Java servlets y JavaServer Pages (JSP), ya que son una buena alternativa para la programación Web dinámica.

Optamos por utilizar el patrón de diseño MVC (Modelo-Vista-Controlador) [3] por ser uno de los más utilizados en la organización de los sistemas de información, ya que ha demostrado ser un muy buen enfoque para crear aplicaciones Web.

Analizando las tecnologías anteriores decidimos utilizar el marco de trabajo Struts [4], ya que Struts las integra permitiendo desarrollar aplicaciones Web de Java de forma fácil y rápida.

Finalmente para el acceso a datos seleccionamos el patrón Data Access Object (DAO) [5], cuya finalidad es desacoplar la lógica de negocio de una aplicación de la lógica de acceso a datos.

A lo largo de este capítulo profundizaremos más en cada una de estas tecnologías. Comenzaremos por la plataforma J2EE y las tecnologías asociadas a ella, seguiremos con el patrón de diseño MVC ya que Struts está basado en esta arquitectura, estableceremos las bases teóricas de Struts y por último definiremos el patrón DAO.

2.1 J2EE y Tecnologías

La implementación de la plataforma J2EE utilizada para HeAP MoProSoft es una configuración centrada en el Nivel Web (Web-Centric), como se muestra en la Figura 2.1 el contenedor Web se sitúa entre el Nivel Cliente y el Nivel EIS (Enterprise Information System). Este tipo de configuración ofrece beneficios, ya que el Nivel Cliente no se ve afectado por cambios en el Nivel EIS, el cliente no tiene acceso directo a los recursos de este nivel, es decir a la información de la base de datos.

[1] http://www.sun.com/java/

Se decidió utilizar el marco de trabajo Struts, basado en el patrón de diseño MVC (Modelo-Vista-Controlador). Este marco de trabajo reside en el Nivel Web [6].

La plataforma J2EE y las tecnologías utilizadas para el desarrollo del sistema HeAP MoProSoft se muestran en la Figura 2.1.

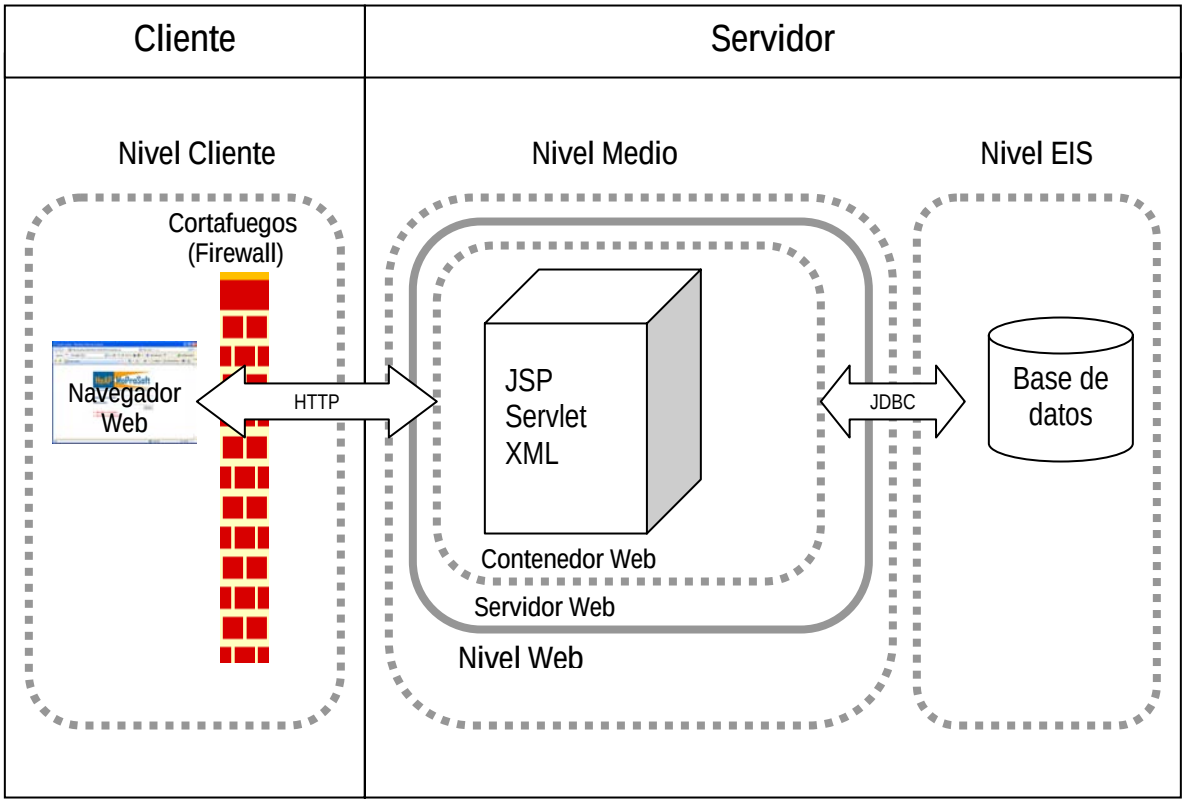


Figura 2.1 Plataforma J2EE y Tecnologías

2.2 Patrón de diseño MVC (Modelo-Vista-Controlador)

El patrón de diseño utilizado para definir la arquitectura es el de Modelo-Vista-Controlador (MVC) [15], este patrón tiene tres componentes clave:

- Modelo: Corresponde a la información, es decir, al manejo de datos y sus funcionalidades lo que propiamente llamamos la “lógica del negocio”
- Vista: Corresponde a la presentación o interacción con el usuario u otro sistema
- Controlador: Corresponde al comportamiento o control del flujo de ejecución

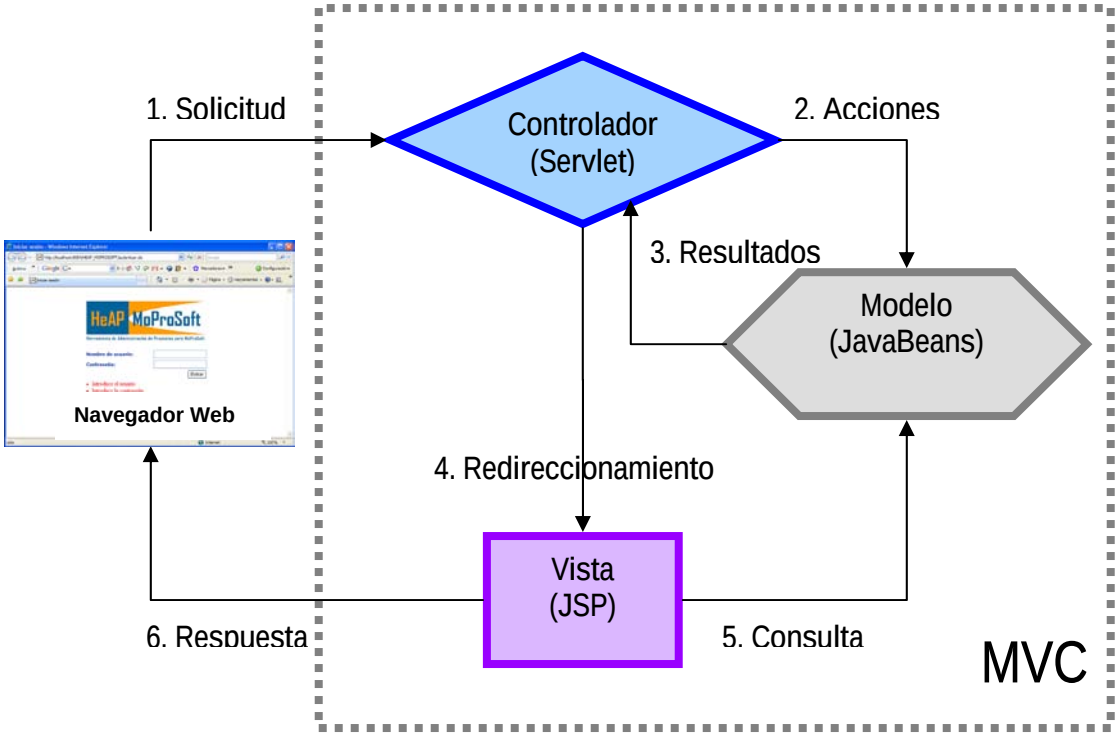


Figura 2.2 Diagrama del patrón MVC (Modelo-Vista-Controlador)

En la aplicación Web el cliente realiza una solicitud al servidor, es procesada por el controlador, quien decide qué acciones del modelo se realizan; el modelo regresa los resultados al controlador y redirecciona a la siguiente vista como respuesta del sistema.

Modelo

El modelo depende del tipo de arquitectura de la aplicación Web, por lo que el modelo puede adoptar muchas formas, generalmente se utilizan JavaBeans con beans de sesión conocidos como objetos de transferencia de datos, se utilizan dentro del nivel Web para crear el contenido dinámico.

Vista

La vista consta generalmente de páginas HTML (contenido estático) y JSP's (contenido estático y dinámico), también es posible el uso de JavaScript ya que no interfiere con el concepto MVC.

Controlador

El controlador por lo general es un servlet Java que realiza las siguientes funciones:

1. Intercepta las peticiones HTTP desde un cliente
2. Traduce cada petición en una operación específica de negocio a llevar a cabo
3. Invoca la operación de negocio o la delega a un manejador
4. Ayuda a seleccionar la siguiente vista a mostrarle al cliente
5. Devuelve la vista al cliente

2.3 Marco de trabajo Struts

Struts es un marco de trabajo que forma parte del software de distribución libre que permite desarrollar aplicaciones Web de forma fácil y rápida, basado en tecnologías como JavaBeans, Java servlets, JavaServer Pages (JSP) y XML. Struts fue creado por Craig R. McClanahan y donado a Apache Software Foundation en el año 2000. La arquitectura de Struts está basada en el patrón de diseño MVC (modelo-vista-controlador).

Del MVC Struts se centra en el controlador, permitiendo que el modelo y la vista se puedan integrar utilizando otras tecnologías. Para la parte del modelo Struts puede interactuar con las tecnologías estándar de acceso a datos como JDBC y EBJ. La parte de la vista puede ser implementada con JSP, JSTL (JSP Standard Tag Library) y JSF (JavaServer Faces).

2.3.1 Struts una implementación del patrón MVC

Struts es un conjunto de colaboración entre clases, Java servlets, JavaServer Pages, basado en la arquitectura del MVC, como se muestra en la figura 2.3.

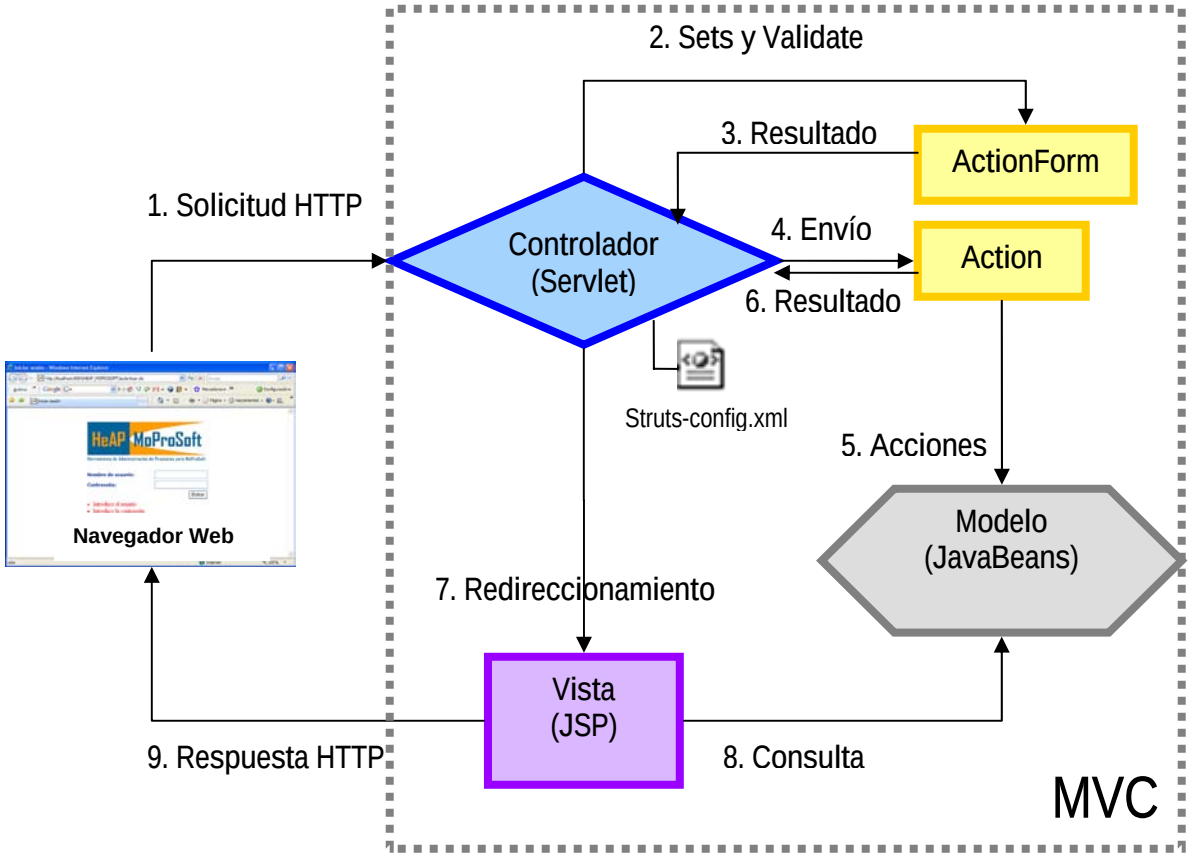


Figura 2.3 Marco de trabajo Struts una implementación del MVC.

El usuario realiza una petición, el navegador Web genera una solicitud, que es recibida por el ActionForm para validarla y enviarla al Controlador (Servlet especializado) para ser atendida, el cual se encarga de analizar la solicitud y llamar al Action correspondiente dependiendo de la configuración definida en el archivo XML (struts-config.xml) y envía los parámetros correspondientes. El Action envía las acciones al modelo para llevar a cabo su tarea. Dependiendo del resultado del Action, el controlador envía la siguiente vista que será entregada al usuario.

2.3.2 Los Paquetes que componen Struts

El marco de trabajo Struts está compuesto por aproximadamente trescientas clases de Java, divididas en aproximadamente ocho paquetes principales ya que se encuentra en constante crecimiento y evolución. A continuación se muestra el diagrama con los ocho paquetes principales de alto nivel con los que cuenta Struts:

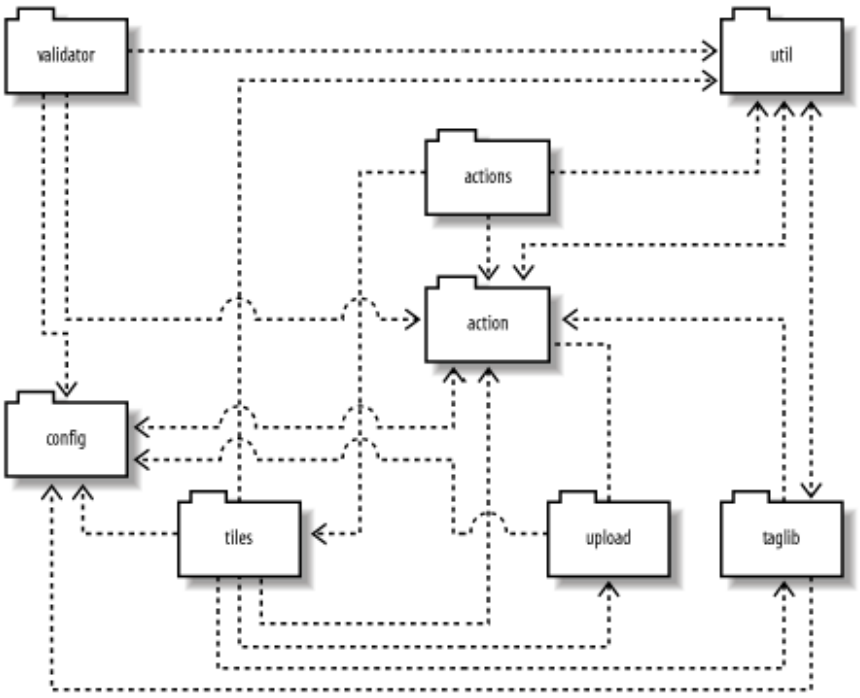


Figura 2.4 Ocho paquetes principales de alto nivel de Struts.

Nombre del paquete	Descripción
`action`	Contiene las clases del controlador, las clases `ActionForm` y `ActionMessage.`
`actions`	Contiene las clases `Action`, como la `DispatchAction` que sirve para extender la funcionalidad de la aplicación
`config`	Incluye a las clases de configuración, que son representaciones en memoria del archivo de configuración de Struts.
`taglib`	Contiene las clases que manejan las etiquetas de la biblioteca de etiquetas de Struts.
`tiles`	Contiene las clases usadas por el marco de trabajo Tiles.

`upload`	Incluye las clases para subir y bajar archivos desde el sistema de archivos local usando el explorador.
`util`	Contiene las clases de utilidades de propósito general utilizadas por el marco de trabajo.
`validator`	Contiene las clases extendidas específicas de Struts cuando se utiliza el Validator. Actualmente las clases e interfaces Validator se encuentran separadas en el paquete `commons`.

Tabla 2.1 Paquetes principales de alto nivel de Struts.

2.3.3 Los componentes del Controlador de Struts

El controlador en las aplicaciones con arquitecturas MVC tiene muchas responsabilidades, pero sus principales funciones son recibir una petición del cliente, invocar una operación de negocio y elegir la vista que será devuelta al cliente.

Los componentes del controlador de Struts son el punto de control de la aplicación Web, reciben una petición del cliente la cual se mapea en alguna operación de negocio y selecciona la siguiente vista como respuesta al usuario basado en la información y en el siguiente estado.

Las responsabilidades del controlador de Struts son implementadas por diferentes componentes, en la figura 2.6 podemos ver el diagrama de clases de los componentes del controlador de Struts que comparten parte de las responsabilidades.

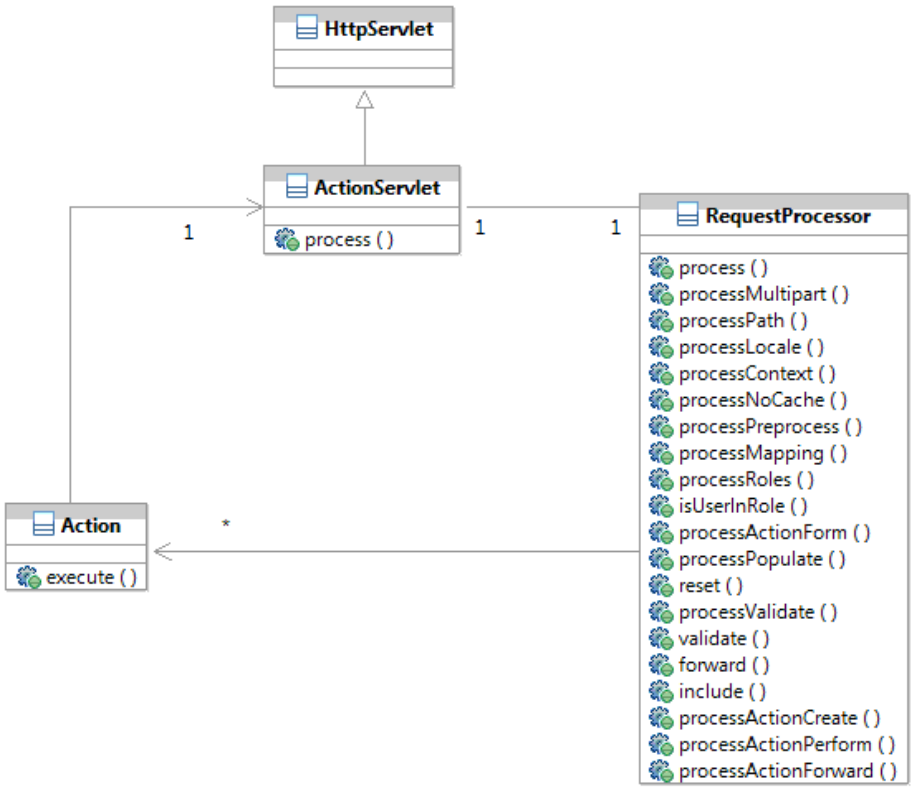


Figura 2.5 Componentes del controlador de Struts.

2.3.3.1 La clase ActionServlet de Struts

La clase `ActionServlet (org.apache.struts.action.ActionServlet)` hereda de la clase `HttpServlet (javax.servlet.http.HttpServlet)`, que es la responsable del manejo de las peticiones HTTP.

La clase `ActionServlet` intercepta todas las solicitudes de la capa del cliente, es decir todas las solicitudes pasan por el `ActionServlet` antes de dirigirse a cualquier parte de la aplicación.

Cuando una instancia de la clase `ActionServlet` recibe un `HttpRequest`, a través de cualquiera de los dos métodos `doGet()`o `doPost()`, se llama el método `process()` para manejar la solicitud.

2.3.3.2 La clase RequestProcessor de Struts

Una instancia de la clase `ActionServlet` llama al método `process()` de la clase `RequestProcessor (org.apache.struts.action.RequestProcessor)` el cual recibe la solicitud actual y los objetos de respuesta.

2.3.3.3 La clase Action de Struts

La clase `Action (org.apache.struts.action.Action)` es el corazón de Struts, es el puente entre la solicitud del cliente y las operaciones del negocio. Cada clase `Action` está diseñada para responder a una sencilla operación del negocio. Una vez que se determina la instancia correcta de la clase `Action` se invoca al método `processActionPerform()`.

El método es responsable de llamar al método `execute()` en la instancia de la clase `Action`.

2.3.3.4 La clase ActionForward de Struts

El método `execute()` de la instancia de la clase `Action` regresa un objeto de la clase `ActionForward`. La clase `ActionForward` representa una abstracción lógica de un recurso Web, generalmente se trata de una página JSP o de un Java servlet. El `ActionForward` envuelve al recurso Web, por lo que existe un menor acoplamiento entre la aplicación y este recurso, los cuales se especifican únicamente en el archivo de configuración (utilizando los atributos de `name`, `path` y `redirect` del elemento `forward`). El `RequestDispatcher` puede llevar cabo tanto un reenvío como un redireccionamiento para una `ActionForward`, dependiendo del valor que tome el atributo `redirect`.

Podemos devolver un ActionForward desde una clase Action de dos maneras, la primera es crearla dinámicamente en la clase Action y la segunda es utilizando el mapeado de acciones configurado en el archivo de configuraciones. Por ejemplo en:

```
return mapping.findForward(“SUCCESS”);
```

En este ejemplo se pasa como parámetro el argumento “Success” al método findForward() de una instancia de la clase ActionMapping, el valor del argumento que recibe el método finfForward() debe coincidir con alguno de los nombres especificados en el apartado global-forwards o uno que sea específico a la acción desde la que se llama.

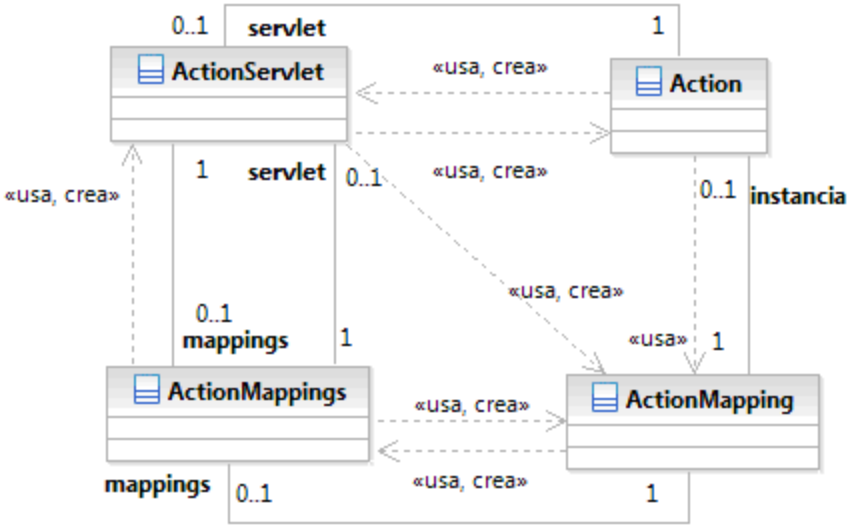


Figura 2.6 Relación entre las clases Action, ActionServlet y ActionMapping.

2.3.4 Los componentes del Modelo de Struts

Para una aplicación Struts el modelo representa los datos y los procesos de negocio de la organización. El modelo incluye a las entidades de negocio y a las reglas que controlan el acceso y modificación de los datos. El marco de trabajo Struts en realidad no aporta demasiado a la forma de implementar el modelo porque no brinda herramientas, pero tampoco limita la utilización de algún patrón para su implementación.

Una arquitectura MVC debe buscar el mayor grado de desacoplamiento entre el modelo (incluyendo la forma de transformar el modelo en datos persistentes) y el resto de la aplicación. Puesto que Struts no proporciona herramientas orientadas a la implementación del modelo, utilizaremos el patrón DAO (del inglés Data Access Object) [5] con este fin.

2.3.4.1 Patrón DAO (Data Access Object)

La finalidad del patrón DAO [5] es desacoplar la lógica del negocio de la lógica de acceso a datos. En el patrón DAO se encuentran implementados dos patrones de diseño estructurales: Bridge [3] y Adapter [3].

El objetivo del patrón DAO es encapsular y ocultar la forma de acceder a la fuente de datos, con el fin de obtener la información sin importar cómo se realiza el acceso a datos o cuál es la fuente de almacenamiento.

El patrón DAO implementa los mecanismos de acceso a datos necesarios para trabajar con la fuente de datos que pueden ser de diferentes tipos (bases de datos relacionales, servicios externos como B2B, CORBA). En nuestro caso, y para el desarrollo de nuestra aplicación, utilizaremos el API (Application Programming Interface) JDBC (Java Database Connectivity) para acceder a los datos de una base de datos relacional, el cual permite acceder y manipular de una forma estándar los datos almacenados en la base de datos. También permite a las aplicaciones J2EE utilizar sentencia SQL, el patrón DAO oculta la complejidad del uso de JDBC a la de presentación o de negocio.

Diagrama de clases

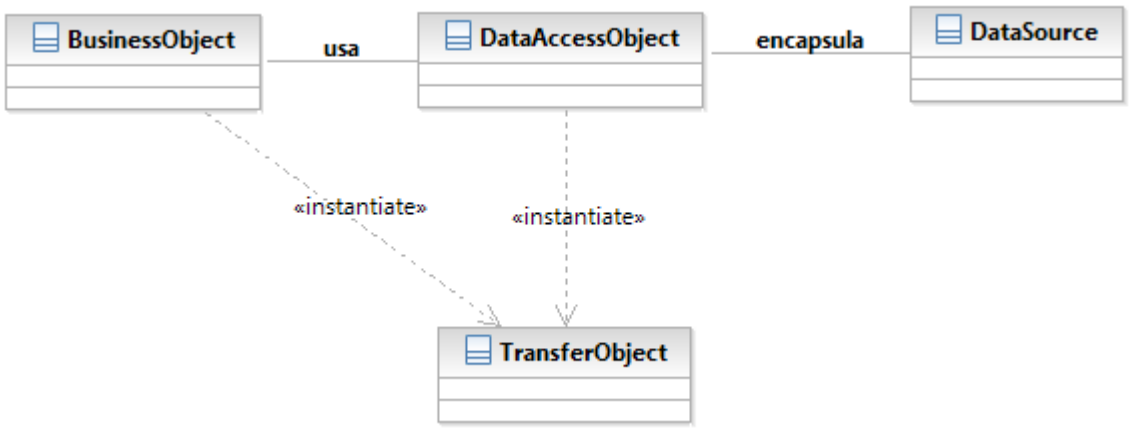


Figura 2.7 Patrón DAO.

BusinesObject

Los BusinesObjects (objeto de negocio) representan los datos del cliente. El objeto que requiere acceder a la fuente de datos y obtener los datos almacenados. Un BusinessObject puede ser implementado como un Bean de sesión o de entidad, o algún otro objeto Java, para ayudar a realizar el acceso a los datos.

DataAccessObject

El principal objeto de este patrón es el Data Acess Object. Se encarga de abstraer el acceso a datos permitiendo un acceso transparente a la fuente de datos. El BusinessObject delega al Data Access Object la responsabilidad de las operaciones de carga y almacenamiento de datos (como create, read, update y delete, también conocidas como operaciones CRUD).

DataSource

Representa una implementación de alguna fuente de datos. Una fuente de datos puede ser una base de datos del tipo relacional, un OODBMS, un repositorio XML, etc. Una fuente de datos también puede ser otro sistema, un servicio (B2B) u otro tipo de repositorios (LDAP).

TransferObject

Es utilizado para transportar datos. El Data Access Object utiliza al Transfer Object para regresar datos al cliente y para recibir los datos que serán actualizados en la fuente de datos.

2.3.5 Los componentes de la Vista de Struts

El marco de trabajo Struts utiliza los componentes de la vista para mostrar dinámicamente el contenido al cliente. Los componentes están basados principalmente en JavaServerPages, y proporcionan soporte en la internacionalización, aceptación de entrada de datos por parte del usuario, validación y gestión de errores, lo que permite una mejor y más fácil implementación de la vista.

En términos generales una vista es una representación del modelo de dominio en una interfaz de usuario. Para el sistema, las vistas se crearan utilizando JavaServer Pages (JSP), por ser la tecnología más ampliamente utilizada para la presentación. Se pueden utilizar componentes adicionales como: páginas HTML, bibliotecas de etiquetas personalizadas JSP, JavaScript y hojas de estilo (CSS), archivos multimedia, paquetes de recursos de mensaje y clases `ActionForm`.

2.3.5.1 La clase ActionForm de Struts

La clase `ActionForm (org.apache.struts.action.ActionForm)` es el componente clave para administrar las tareas que involucran todas aquellas acciones de entrada de datos por parte de los usuarios a través de formularios (nombres de usuario, contraseñas, datos personales, etc), además de proporcionar los mecanismos necesarios para realizar las operaciones de recuperación y validación de los datos, mandar mensajes de error cuando ocurra un fallo al romper una regla de validación.

La clase `ActionForm` recupera los datos del formulario HTML y los transfiere a la clase `Action`. La clase `ActionForm` también actúa como un buffer en el sentido de que si ocurre un error el estado de los datos permanecerá almacenado permitiendo que se lleven a cabo las validaciones necesarias; además de facilitarle al usuario la introducción de información al formulario ya que la información permanecerá en él después de ocurrido el error. También la clase `ActionForm` actúa como un cortafuegos, porque impide la entrada de datos inválidos o sospechosos a la capa del negocio hasta que sean validados mediante las reglas de validación.

Una ventaja más de la clase `ActionForm` es que cuando se devuelven los datos desde la capa de negocio hacia la vista se completa una determinada `ActionForm` para ser utilizada por una página JSP para mostrar los campos de entrada de datos para un formulario HTML.

Una vez que los datos pasaron las reglas de validación la clase `ActionForm` pasa al método `execute()` de la clase `Action`, desde donde se pueden recuperar los datos de la `ActionForm` y pasarlos a la capa del negocio.

Las `ActionForm` pueden ser de petición y sesión. Si se utiliza al alcance de petición, la `ActionForm` está disponible hasta el final del ciclo petición/respuesta y una vez que se le ha devuelto la respuesta al cliente, la `ActionForm` y los datos dentro de ésta dejan de estar accesibles. Cuando es necesario que los datos del formulario se mantengan durante más tiempo que una sola petición, la `ActionForm` debe tener un alcance de sesión. La `ActionForm` de sesión permanecerá en la sesión hasta que se elimine o reemplace con otro objeto o hasta que la sesión expire. Struts no cuenta con un método para limpiar automáticamente los objetos `ActionForm` de ámbito de sesión.

El método `validate()` es invocado por el `RequestProcessor` para cada petición. Para invocarse se necesitan dos cosas, la primera es configurar una `ActionForm` para un mapeado de una acción. Esto significa que el atributo `name` para un elemento `action` debe corresponder al atributo `name` de una de los elementos `form-bean` del archivo de configuración. La segunda condición que se debe cumplir para que el `RequestProcessor` invoque al método `validate()` es que el atributo `validate` en el `mapeado` de acciones debe tener el valor de `true`. El método `validate()` puede devolver un objeto `ActionMessage`, de tipo `ActionError` si se detectaron errores durante la validación, y devuelve `null` en caso de no encontrar errores [7].

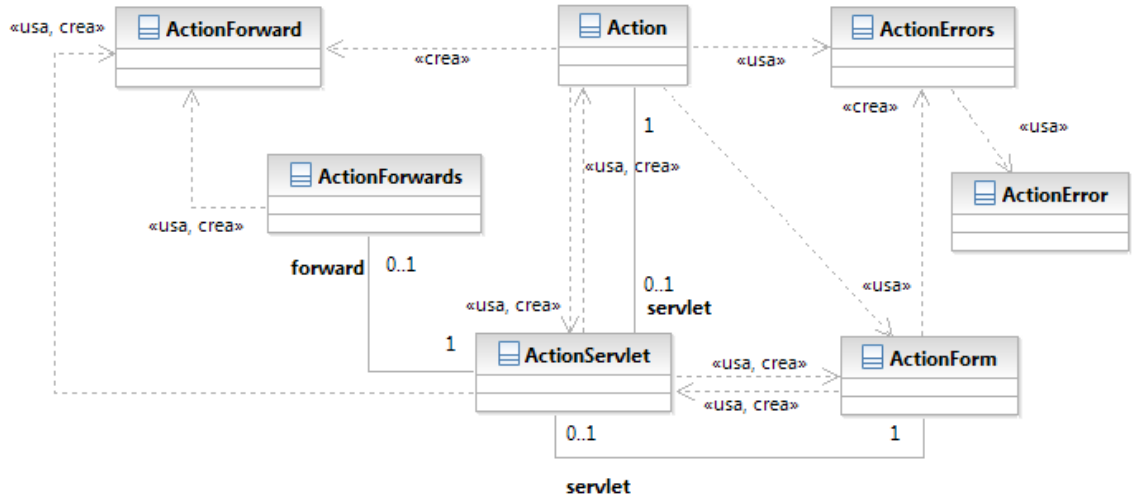


Figura 2.8 Relación entre las clases ActionServlet, Action y ActionForm.

2.3.6 Integración de Struts con la plataforma J2EE

Como se muestra en la Figura 2.9 el marco de trabajo Struts reside en el nivel Web. Las aplicaciones Struts se encuentran albergadas en el contenedor Web haciendo uso de los servicios proporcionados por dicho contenedor.

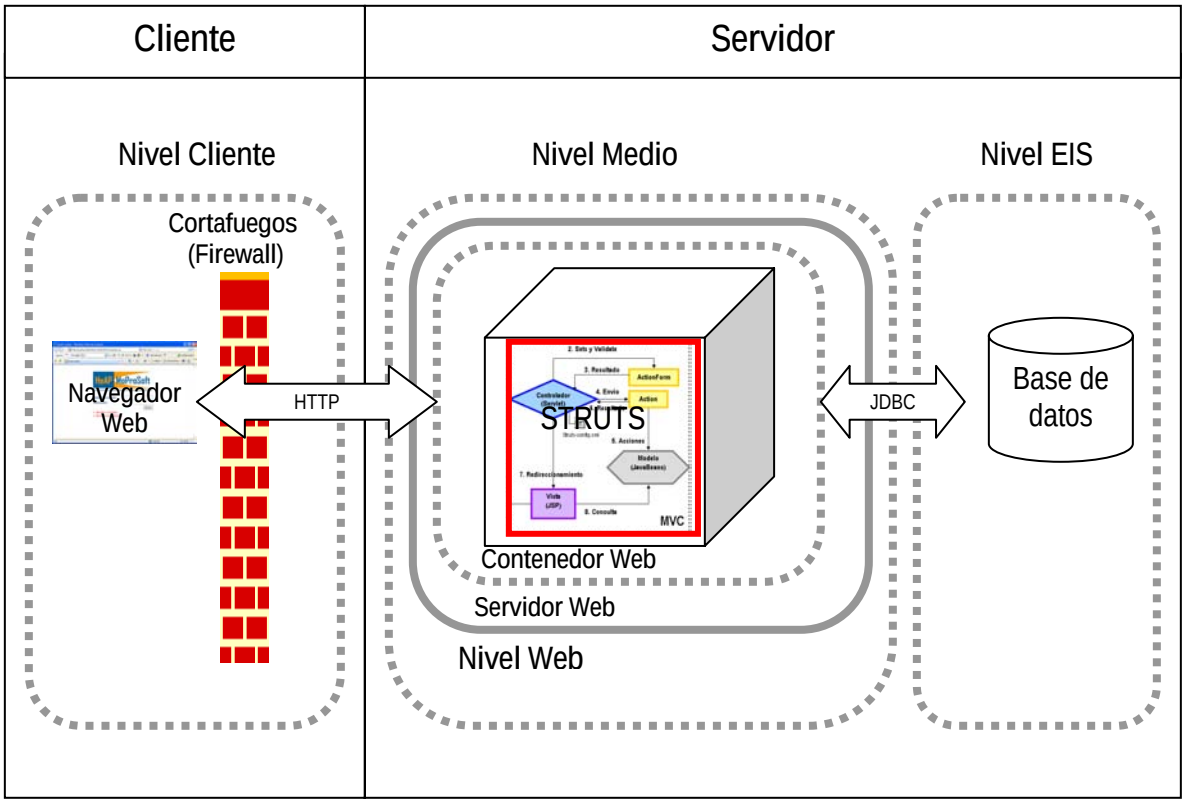


Figura 2.9 Marco de trabajo Struts dentro del Nivel Web.

Struts al estar basado en la tecnología J2EE, Servlet y JSPs, depende de un contenedor Web, que permite que se alberguen y ejecuten tanto como los Servlets, JSPs y las clases Java, gestionando peticiones utilizando los protocolos HTTP y HTTPS. Además de permitir la comunicación con la base de datos relacional, por medio de un driver JDBC (Java Database Connectivity).

En el caso de Struts, el tipo más común de cliente es un navegador Web, pero también es posible tener clientes como dispositivos inalámbricos o applets Java.

Capítulo 3. Justificación de la Herramienta

Introducción

En este capítulo se realizará la justificación de este trabajo de tesis partiendo de una investigación sobre el tipo de herramientas para la administración de proyectos que actualmente se encuentran en el mercado. Realizando también la descripción del problema que surge al no contar con una herramienta de administración de proyectos para MoProsoft.

3.1 Herramientas para la administración de proyectos

Actualmente podemos encontrar una gran variedad de herramientas para administrar proyectos, dedicadas a apoyar en las diversas actividades del proyecto, como la administración del calendario, asignación de recursos, asignación de tareas, administración del alcance, costo, comunicación y sistemas de documentación, entre otros, realizamos una investigación para conocer sus principales características.

Dentro de la amplia gama tenemos herramientas Web o aplicaciones de escritorio, ambas de tipo colaborativo. Para este trabajo nos enfocaremos en las herramientas Web, ya que se desarrollará una herramienta Web colaborativa. Este tipo de herramientas están disponibles a través de Internet por medio de una Intranet o Extranet utilizando un navegador Web. Entre sus ventajas se encuentran:

- *Facilidad de acceso*, es decir, se puede acceder desde cualquier computadora y manteniendo un control de acceso.
- *Multiusuario*, por naturaleza son sistemas que permiten acceso a más de un usuario a la vez.
- *Única instalación*, se instalan una sola vez en el servidor.
- *Colaboración y seguimiento de equipos de trabajo.*

Y también tienen un par de desventajas como:

- *Respuesta lenta*, en comparación con las aplicaciones de escritorio las herramientas Web tienen un tiempo de respuesta menor, además de depender de la velocidad de conexión a Internet.
- *Gráficos limitados*, las herramientas Web se encuentran limitadas en capacidad gráfica en comparación con las aplicaciones de escritorio que cuentan con muchos más recursos de este tipo.
- *Disponibilidad limitada*, la información del proyecto únicamente se encuentra disponible si el servidor está en línea y si hay una conexión a Internet.

Como ejemplos de este tipo de herramientas podemos encontrar @task[1], Ace Project[2], EasyProjects.NET[3]. La siguiente tabla muestra la comparación entre estas herramientas:

Características	@task	Ace Project	EasyProjects.NET
Administración de Proyectos			
Administración de Tareas	☺	☺	☺
Asignación y reasignación de tareas	☺	☺	☺
Progreso de tareas	☺	☺	☺
Dependencias entre tareas	☺	☺	☺
Calendario	☺	☺	☺
Ajuste en el calendario	☺	☺	☺
Time Lines	☺	☺	
Eventos	☺	☺	☺
Diagramas de Gantt	☺	☺	☺
WBS	☺		
Reportes	☺	☺	☺
Estadísticas	☺	☺	☺
Cargas de trabajo	☺	☺	☺
Financieros	☺		
Personalizados	☺	☺	☺
Baseline	☺		
Documentación de administración de proyectos	☺	☺	
Administración de Recursos			
Detalle de los recursos	☺	☺	☺
Timesheets	☺	☺	☺
Correos electrónicos	☺	☺	☺
Costos	☺		☺
Colaboración			
Calendario del equipo	☺	☺	☺
Timelines	☺	☺	

[1] http://www.attask.com/
[2] http://www.aceproject.com/
[3] http://www.easyprojects.net/

Ayuda y Soporte			
Manual	☺	☺	☺
Soporte	☺	☺	☺
FAQs	☺	☺	☺
Tutoriales	☺	☺	☺

Tabla 3.1. Características de las herramientas para administración de proyectos[4]

Ninguna de las herramientas antes mencionadas fue diseñada para seguir un modelo de administración de proyectos de desarrollo de software en específico, son herramientas para proyectos genéricos. Además no existen herramientas específicas para la administración de proyectos basadas en MoProSoft, es por esto que se tomó la decisión de desarrollar HeAP MoProSoft, por la necesidad de contar con una herramienta especializada para administrar proyectos de software, tomando como base un modelo diseñado especialmente con este propósito.

3.2 Descripción del problema

Desde la aprobación de MoProSoft como norma mexicana, surgió el interés en las empresas de desarrollo de software de adoptarlo como modelo de procesos. La adopción incluye el aspecto de la administración de proyectos que afecta a las PyMES y áreas internas dedicadas al desarrollo y/o mantenimiento de Software, ya que MoProsoft está dirigido a este sector de la industria.

Una ayuda para la adopción de MoProSoft es proponer herramientas necesarias para la administración de los proyectos, brindando los mecanismos necesarios para realizar una correcta gestión de los documentos generados en los procesos de Gestión de Proyectos y Administración del Proyecto Específico de las categorías de Gerencia y Operación respectivamente, según lo que marca MoProSoft en estos procesos.

La herramienta que se desarrollará tendrá por nombre Herramienta de Administración de Proyectos para MoProSoft (HeAP MoProSoft). La idea de desarrollar esta herramienta surgió como un proyecto, para el curso de Ingeniería de Software Orientada a Objetos, llamado “Tlamatini HeAP MoProSoft”, este proyecto sirvió como antecedente de este trabajo de tesis en el que retomamos las ideas y el esfuerzo que ya se había realizado.

A diferencia de otras herramientas que no han sido diseñadas especialmente para la implementación de MoProSoft y de realizar las actividades de administración de proyectos manualmente. HeAP MoProSoft será una herramienta diseñada especialmente para la Administración de proyectos para empresas dedicadas al desarrollo y/o mantenimiento de software, que quieren adoptar MoProSoft como su modelo de procesos favoreciendo su implementación, reduciendo el tiempo y proporcionando las plantillas con la información mínima requerida para generar los documentos que

[4] http://project-management-software-review.toptenreviews.com/

estos procesos requieren, apoyando a los roles de MoProSoft: que son *Responsable de Gestión de Proyectos (RGPY)* y *Responsable de la Administración del Proyecto Específico (RAPE)*.

Este sistema en su primera versión considera a esos dos usuarios y cubrirá algunas de las actividades, de nivel 1 y 2 de capacidad de procesos, correspondientes a los procesos de *Gestión de Proyectos* y *Administración del Proyecto Específico* de las categorías de *Gerencia* y *Operación* respectivamente. Las actividades que este sistema cubrirá fueron elegidas por ser las mínimas requeridas para comenzar con las prácticas de administración de proyectos según lo que marca MoProSoft.

Para el proceso de Gestión de Proyectos de la Categoría de Gerencia se realizarán las siguientes actividades, el usuario responsable de esto será el *RGPY*:

- Generar el Plan de Proyectos realizando las siguientes actividades:
    - Generar el Registro del Proyecto
    - Generar la Descripción del Proyecto
    - Asignar Responsable de Administración del Proyecto Específico
- Administración del personal
- Consultar el Plan del Proyecto Específico

Para el proceso de Administración del Proyecto Específico de la Categoría de Operación se realizarán las siguientes actividades, siendo responsable el usuario *RAPE*:

- Generar el Plan del Proyecto Específico que está conformado de la siguiente manera:
    - Definir Ciclos y Actividades
    - Registrar Tiempo Estimado
    - Registrar Costo Estimado
    - Generar el Plan de Adquisiciones y Capacitación
    - Conformar el Equipo de Trabajo
    - Generar el Plan de Manejo de Riesgos
    - Generar el Protocolo de Entrega

3.2.1 Formatos

Para cada una de las actividades anteriores se definió la información que se debe proporcionar, integrando de esta manera un conjunto de formatos:

Para el Plan de Proyectos se requiere de estos campos:

Generar el Registro del Proyecto

Campo	Descripción
Identificador	Nombre corto, acrónimo o alias para el proyecto.
Nombre del proyecto	Nombre completo del proyecto.
Referencia al contrato	En el caso de que exista un contrato se debe especificar el nombre o número del contrato que se encuentra relacionado con este proyecto.
Cliente	El nombre del cliente (persona o empresa) del proyecto.
Contacto 1	Nombre completo de un contacto por parte del cliente (Ejem. Sponsor, Administrador del Proyecto).
Datos del contacto 1	Los datos que pueden incluirse son correo electrónico, teléfono, dirección entre otros.
Contacto 2	Nombre completo de un segundo contacto por parte del cliente (Ejem. Sponsor, Administrador del Proyecto).
Datos del contacto 2	Los datos que pueden incluirse son correo electrónico, teléfono, dirección entre otros.
Fecha inicio	Fecha en la que inicia formalmente el proyecto.
Fecha fin	Fecha estimada para el término del proyecto.

Tabla 3.2 Formato Registro del Proyecto

Generar la Descripción del Proyecto

Campo	Descripción
Propósito	Describir el propósito del proyecto.
Objetivos	Detallar los objetivos del proyecto.
Restricciones	Especificar las restricciones del proyecto.
Criterios de aceptación	Especificar los criterios de aceptación.

Tabla 3.3 Formato Descripción del Proyecto

Asignar Responsable de Administración del Proyecto Específico

Campo	Descripción
RAPE	Asignar al Responsable de la Administración del Proyecto Específico (RAPE).

Tabla 3.4 Formato Asignar Responsable de Administración del Proyecto

Para el Plan del Proyecto Específico se requieren de las actividades necesarias para llenar los campos:

Definir Ciclos y Actividades

Campo	Descripción
Identificador	El identificador de la actividad puede ser un número.
Ciclo	Es un número entero que indica el número de ciclo.
Nombre	Es el nombre de la actividad que se realizará.
Métodos, técnicas y herramientas	En este campo se indican los métodos, técnicas y herramientas que se utilizarán para llevar a acabo esta actividad.
Fecha inicio	Es la fecha en la que inicia la actividad.
Fecha fin	Es la fecha en la que debe terminar dicha actividad.
Esfuerzo	El esfuerzo que tendrá la actividad calculado en horas.

Tabla 3.5 Formato Ciclos y Actividades

Registrar Costo Estimado

Campo	Descripción
Hardware	Costo del Hardware
Software	Costo del Software
Viajes	Costo por viáticos
Reubicación	Costos por reubicación
Subcontratos	El costo de los subcontratos
Mantenimiento	Costos generados por mantenimiento
Capacitación	Costos por la capacitación para el equipo de trabajo.
Espacio	Costo por la renta de los inmuebles.
Recursos humanos	Costo de los recursos humanos
Comunicaciones	Costo de los sistemas y medios de comunicación.
Administración de la configuración	Costo por las herramientas para realizar la administración de la configuración.
Soporte a la documentación	Costo por el soporte y realización de documentación.
Otros	Otros costos generados

Tabla 3.6 Formato Costo Estimado

Generar el Protocolo de Entrega

Campo	Descripción
Descripción	La descripción para explicar en que consistirá el protocolo de entrega.

Tabla 3.7 Formato Protocolo de Entrega

Generar el Plan de Adquisiciones

Campo	Descripción
Adquisición	Tipo de adquisición
Unidades	Número de unidades adquiridas
Costo unitario estimado	El costo unitario que se estima costará la adquisición.
Costo unitario real	El costo unitario con el que se adquirió
Fecha requerido	Indicar la fecha en la que fue requerido
Fecha adquirido	Indicar la fecha en la que fue adquirido.
Estado de la adquisición	El estado en el que se encuentra la adquisición (Adquirido, Cancelado, Suspendido)

Tabla 3.8 Formato Plan de Adquisiciones

Generar el Plan Manejo de Riesgos

Campo	Descripción
Riesgo	Nombre del riesgo
Descripción	Una descripción detallada del riesgo
Tipo	Tipo de riesgo que se trata (Interno o Externo)
Fecha de identificación	La fecha en la que fue identificado el riesgo.
Afectación	La afectación del riesgo en caso de presentarse
Decisión	Las decisiones que se tomarán con respecto al riesgo.
Plan de Mitigación	Las acciones preventivas que se tomarán para evitar que el riesgo se presente.
Plan de Contingencia	Las acciones correctivas que se tomarán una vez que se presente el riesgo.
Probabilidad	La probabilidad de que el riesgo se presente (Alta, Media, Baja).
Impacto	El impacto del riesgo (Alto, Medio, Bajo).
Estado	El estado del riesgo (Identificado, Evaluado, En mitigación, En contingencia, Cerrado).

Tabla 3.9 Formato Plan de Manejo de Riesgos

Plan de Capacitación

Campo	Descripción
Descripción	Una descripción explicando de qué se tratará la capacitación.
Responsable	El responsable de llevar a cabo la capacitación.
Recursos humanos	Los recursos humanos que asistirán a la capacitación.
Fecha inicio	La fecha en la que inicia la capacitación.
Fecha fin	La fecha en la que termina la capacitación
Lugar	El lugar en el que se realizará la capacitación
Duración	Y la duración de la capacitación

Tabla 3.10 Formato Plan de Capacitación

Capítulo 4. Desarrollo de la Herramienta

Introducción

Para el desarrollo de HeAP MoProSoft utilizamos el Proceso Unificado de Desarrollo de Software, (PU) [8], es un proceso dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. El proceso utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) [9], un lenguaje visual, que nos permite modelar los diferentes esquemas de un sistema de software.

A lo largo del capítulo detallaremos el desarrollo del sistema a lo largo de los cinco flujos de trabajo del Proceso Unificado de Desarrollo de Software, iniciando por el detalle de los requisitos.

4.1 Requisitos

El flujo de trabajo de requisitos tiene como objetivo desarrollar el modelo de casos de uso del sistema que se va a construir delimitándolo y capturando la funcionalidad que tendrá desde el punto de vista del usuario. Este modelo es muy importante ya que modela lo que será el sistema en su totalidad, se debe desarrollar utilizando el lenguaje del usuario por ser el punto de acuerdo entre el desarrollador, el usuario y el cliente.

La forma adecuada de crear el modelo de casos de uso es utilizando casos de uso, porque proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales, que serán la entrada para generar el modelo de análisis, diseño y pruebas.

Este modelo muestra la interacción de los actores con los casos de uso, para hacer el modelo de casos de uso, lo primero que haremos será desarrollar el diagrama general de casos de uso, identificando a los tipos de usuarios que tendrá el sistema, cada usuario será representado por un actor, el siguiente paso es identificar todas las acciones que llevarán a cabo dichos actores, una secuencia de acciones que le dan al actor un resultado de valor, será una caso se uso.

Una vez que se tiene el diagrama general de casos de uso, el siguiente paso es detallar cada caso de uso, identificando las relaciones uso que pueden ser de extensión o de inclusión, dependiendo del tipo de comportamiento que se esté modelando.

Un caso de uso de inclusión aporta un nuevo comportamiento al caso de uso base, en este tipo de relaciones se sigue el flujo del caso de uso base, se ejecuta el caso de uso de inclusión y se continúa con el flujo del caso de uso base. Un caso de uso de extensión se utiliza para extender el comportamiento de un caso de uso base, este tipo de relaciones las podemos utilizar para modelar la parte de un caso de uso que el usuario puede ver como un comportamiento opcional del sistema o un subflujo que se ejecuta sólo bajo ciertas condiciones. En una relación de extensión se sigue el

flujo del caso de uso base y al llegar al punto de extensión se ejecuta el caso de uso de extensión sin regresar al flujo del caso de uso base.

El prototipo del sistema se diseñará utilizando la información que se definió al detallar los casos de uso. El diseño del prototipo es muy importante ya que nos servirá como un medio más tangible de comunicación con el cliente y con el usuario, además se trata de una técnica muy efectiva para validar el funcionamiento del sistema.

Para el desarrollo de este sistema utilizaremos como prototipo las interfaces del sistema “Tlamatini HeAP MoProSoft” desarrollado anteriormente como proyecto del curso de la materia de Ingeniería de Software Orientada a Objetos de la maestría.

4.1.1 Diagrama general de casos de uso

El diagrama general de casos de uso que se obtuvo al identificar la interacción entre actores y casos de uso se muestra en la figura 4.1.

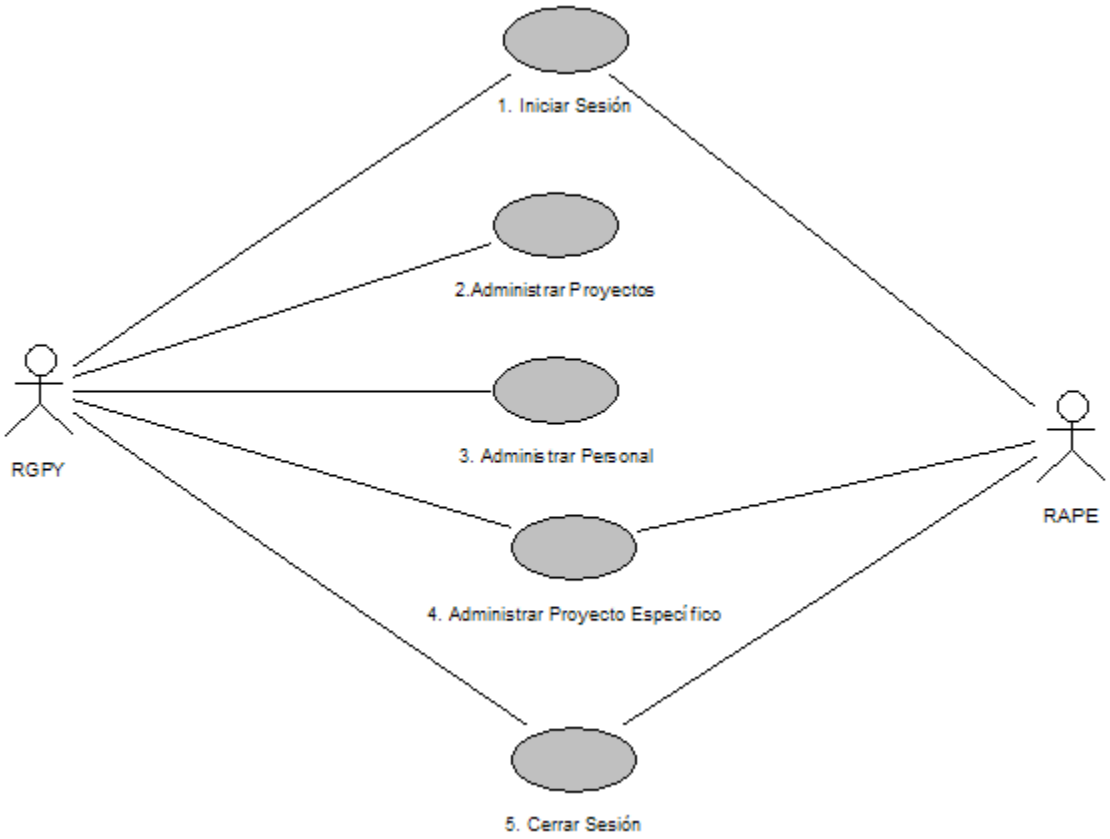


Figura 4.1 Diagrama general de casos de uso

El Sistema HeAP MoProSoft contará con dos tipos de usuarios el *Responsable de Gestión de Proyectos (RGPY)* y *Responsable de la Administración del Proyecto Específico (RAPE)*. Estos tipos de usuarios se representan mediante los actores RGPY y RAPE respectivamente.

El Sistema HeAP MoProSoft está integrado por cinco casos de uso:

1. Iniciar Sesión

   Este caso de uso es para autenticar a los usuarios del sistema (RGPY y RAPE).

2. Administrar Proyectos

   En el caso de uso Administrar Proyectos se lleva a cabo toda la administración del Plan de Proyectos por parte del RGPY, es decir se gestionarán los documentos que integran este plan.

3. Administrar Personal

   En el caso de uso Administrar Personal el RGPY llevará a cabo las altas, bajas, cambios, consulta de usuarios y equipo de trabajo.

4. Administrar Proyecto Específico

   El caso de uso Administrar Proyecto Específico especifica como se administran los documentos que integran el Plan del Proyecto Específico, por parte del RAPE.

5. Cerrar Sesión

   El caso de uso Cerrar Sesión indica como los usuarios (RGPY y RAPE) terminarán su sesión.

De estos cinco casos de uso se eligió el caso de uso *2. Administrar Proyectos* para explicar el proceso de desarrollo de software, a lo largo de este capítulo, por ser el caso de uso en donde se gestionan los proyectos, de la misma forma se desarrollaron los cuatro casos de uso restantes (ver Apéndice A[1]).

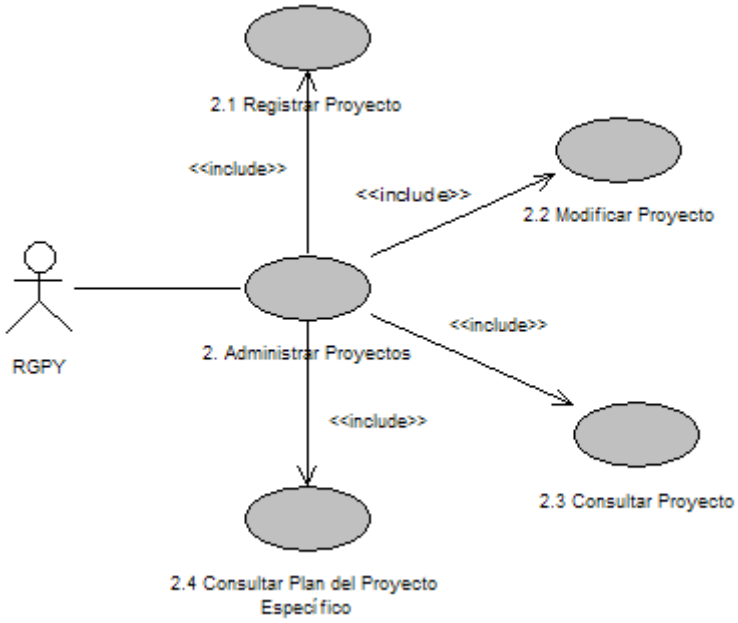
A continuación se realizará el detalle del caso de uso *2. Administrar Proyectos*, indicando los casos de uso con relaciones de inclusión o de extensión.

[1] El Apéndice A se encuentra en la versión electrónica de la tesis.

4.1.2 Detalle del caso de uso 2. Administrar Proyectos

El detalle del caso se realizará utilizando como formato la siguiente tabla que contiene la siguiente información:

- Diagrama del caso de uso detallado, se debe incluir el diagrama del caso de uso detallado.
- Nombre del caso de uso, se indica el nombre del caso de uso que se está detallando.
- Tipo, se indica de que tipo es este caso de uso puede ser básico, extensión o inclusión.
- Actor o actores que participan en este caso de uso
- Descripción, una descripción detallada de lo que realiza el caso de uso.
- Precondiciones, las precondiciones que se necesitan cumplir para que se lleve a cabo este caso de uso.
- Poscondiciones, se detallan las condiciones que se esperan una vez que se ejecutó este caso de uso.
- Flujo de eventos, se describen los flujos de caso de uso que son de dos tipos: básico y alterno.

Diagrama del caso de uso	
	
Definición	
Caso de uso:	2. Administrar proyectos
Tipo:	Básico
Actor:	*RGPY*
Descripción:	En este caso de uso se administran los proyectos, se realiza el registro, modificación y consulta de los proyectos, para el registro se genera el Plan de Proyectos que contiene el Registro y la Descripción del proyecto, así como la asignación del *RAPE*, durante la modificación se podrá actualizar la información contenida en el Registro y en la Descripción del proyecto, además de actualizar el estado del proyecto, la consulta consiste en consultar los datos del Registro y la Descripción del proyecto. También el *RGPY* podrá consultar el Plan del Proyecto Específico de cada uno de los proyectos registrados.
Precondiciones:	Se debió haber ejecutado el caso de uso 1.Iniciar sesión como *RGPY*.

<table>
<tr><td>Poscondiciones:</td><td colspan="3">El RGPY realizó alguna de las siguientes actividades: el registro de un proyecto, la consulta del Plan del Proyecto Específico.</td></tr>
<tr><td colspan="4">Flujo de eventos</td></tr>
<tr><td colspan="4">Flujo básico:</td></tr>
<tr><td colspan="2">Actor</td><td colspan="2">Sistema</td></tr>
<tr><td>Paso</td><td>Acciones</td><td>Paso</td><td>Acciones</td></tr>
<tr><td>1</td><td>El RGPY selecciona la opción “Administrar proyectos”</td><td>2</td><td>El sistema le muestra la interfaz principal de administración de proyectos en dónde se listan los proyectos registrados en el sistema ordenados pro fecha de registro [AE01].</td></tr>
<tr><td>3</td><td>El RGPY selecciona alguna de las siguientes opciones: “”Registrar“, “Modificar”, “Consultar” y “Consultar el Plan del Proyecto Específico”</td><td>4</td><td>El sistema da respuesta a la opción seleccionada por el RGPY mostrándole las interfaces correspondientes a esa opción [AE01] [AG01] [AG02].</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4">Flujos alternos:</td></tr>
<tr><td>Identificador</td><td>Nombre</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>AG01</td><td>Salir del sistema</td><td colspan="2">El RGPY sale del sistema al elegir la opción “Salir”</td></tr>
<tr><td>AG02</td><td>Cerrar sesión</td><td colspan="2">El sistema cierra la sesión cuando el RGPY elige la opción “Cerrar sesión” y le permite elegir otro de sus roles para ingresar al sistema</td></tr>
<tr><td>AE01</td><td>Error de conexión con la base de datos.</td><td colspan="2">El sistema manda un mensaje de error al RGPY indicándole que ha sucedido un error de conexión con la base de datos.</td></tr>
</table>

Tabla 4.1 Detalle del caso de uso 2. Administrar Proyectos

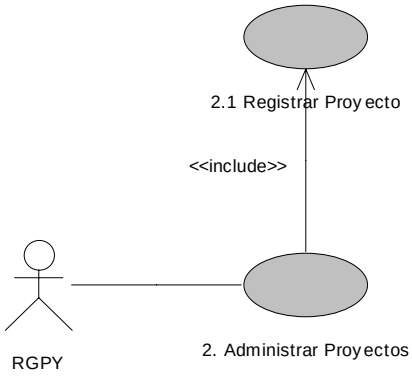
El diseño de la pantalla en donde se listan los proyectos registrados, se muestra en la figura 4.2.

Pantalla “Proyectos”

Esta pantalla es la que se muestra cuando se selecciona la opción Plan de Proyectos, muestra un listado de proyectos con datos básicos como identificador y nombre. En esta pantalla da acceso a las opciones de Registrar, Consultar y Modificar Proyecto.

Figura 4. 2 Pantalla “Proyectos”

Caso de uso: 2.1 Registrar proyecto

Diagrama del caso de uso


Definición	
Caso de uso:	2.1 Registrar proyecto
Tipo:	Inclusión
Actor:	*RGPY*
Descripción:	En el caso de uso Registrar proyecto el *RGPY* se realiza el Registro y la Descripción del proyecto, la asignación del *RAPE* generando con esto el Plan de Proyectos.
Precondiciones:	El *RGPY* seleccionó la opción "Registrar proyecto".
Poscondiciones:	El *RGPY* realizó el registro de un nuevo proyecto dentro del sistema.

Flujo de eventos

Flujo básico:

Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Registrar proyecto comienza cuando el *RGPY* selecciona la opción "Registrar proyecto".	2	El sistema le muestra la interfaz de registro del proyecto.
3	El *RGPY* introduce los datos para realizar el Registro y la Descripción del proyecto y elige la opción "Guardar".	4	El sistema valida los datos y si son correctos se almacenan en la base de datos; el sistema regresará al listado de los proyectos con el nuevo proyecto incluido [AG01] [AG02] [AE01] [AE02].

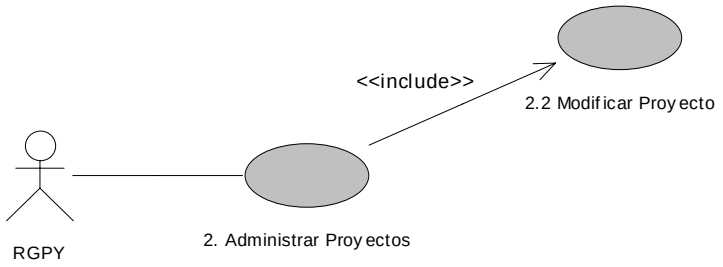
Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RGPY* sale del sistema al elegir la opción "Salir"
AG02	Cerrar sesión	El sistema cierra la sesión cuando el *RGPY* elige la opción "Cerrar sesión" y le permite elegir otro de sus roles para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RGPY* indicándole que ha sucedido un error de conexión con la base de datos.
AE02	Datos inválidos	El sistema manda un mensaje de error indicando que los datos son inválidos.

Tabla 4.2 Detalle del caso de uso 2.1 Registrar Proyecto

Caso de uso: 2.2 Modificar proyecto

Diagrama del caso de uso



Definición

Caso de uso:	2.2 Modificar proyecto
Tipo:	Inclusión
Actor:	*RGPY*
Descripción:	En el caso de uso Modificar proyecto el *RGPY* podrá modificar los datos del proyecto que se encuentra registrado en el sistema, incluyendo el estado del proyecto.
Precondiciones:	El *RGPY* seleccionó la opción "modificar" del listado de proyectos, eligiendo el proyecto que desea modificar.
Poscondiciones:	El *RGPY* realizó la modificación de los datos del proyecto que seleccionó.

Flujo de eventos

Flujo básico:

Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Modificar proyecto comienza cuando el *RGPY* selecciona la opción "modificar" del listado de proyectos eligiendo el proyecto que desea modificar.	2	El sistema le muestra la interfaz que contiene los datos del proyecto que seleccionó, listos para se modificados. [AE01]
3	El *RGPY* introduce los datos que quiere modificar.	4	El sistema valida los datos y si son correctos modifica el registro en la base de datos [AE02].
		5	El sistema regresa a la interfaz del listado de personas registradas [AG01] [AG02].

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RGPY* sale del sistema al elegir la opción "Salir"
AG02	Cerrar sesión	El sistema cierra la sesión cuando el *RGPY* elige la opción "Cerrar sesión" y le permite elegir otro de sus roles para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RGPY* indicándole que ha sucedido un error de conexión con la base de datos.
AE02	Datos inválidos	El sistema manda un mensaje de error indicando que los datos son inválidos.

Tabla 4.3 Detalle del caso de uso 2.2 Modificar Proyecto

El diseño de la pantalla para registrar o modificar los datos del registro de un proyecto se muestran en la figura 4.3.

Pantalla “Registrar o modificar proyecto”

En esta pantalla se podrá registrar o modificar un proyecto.

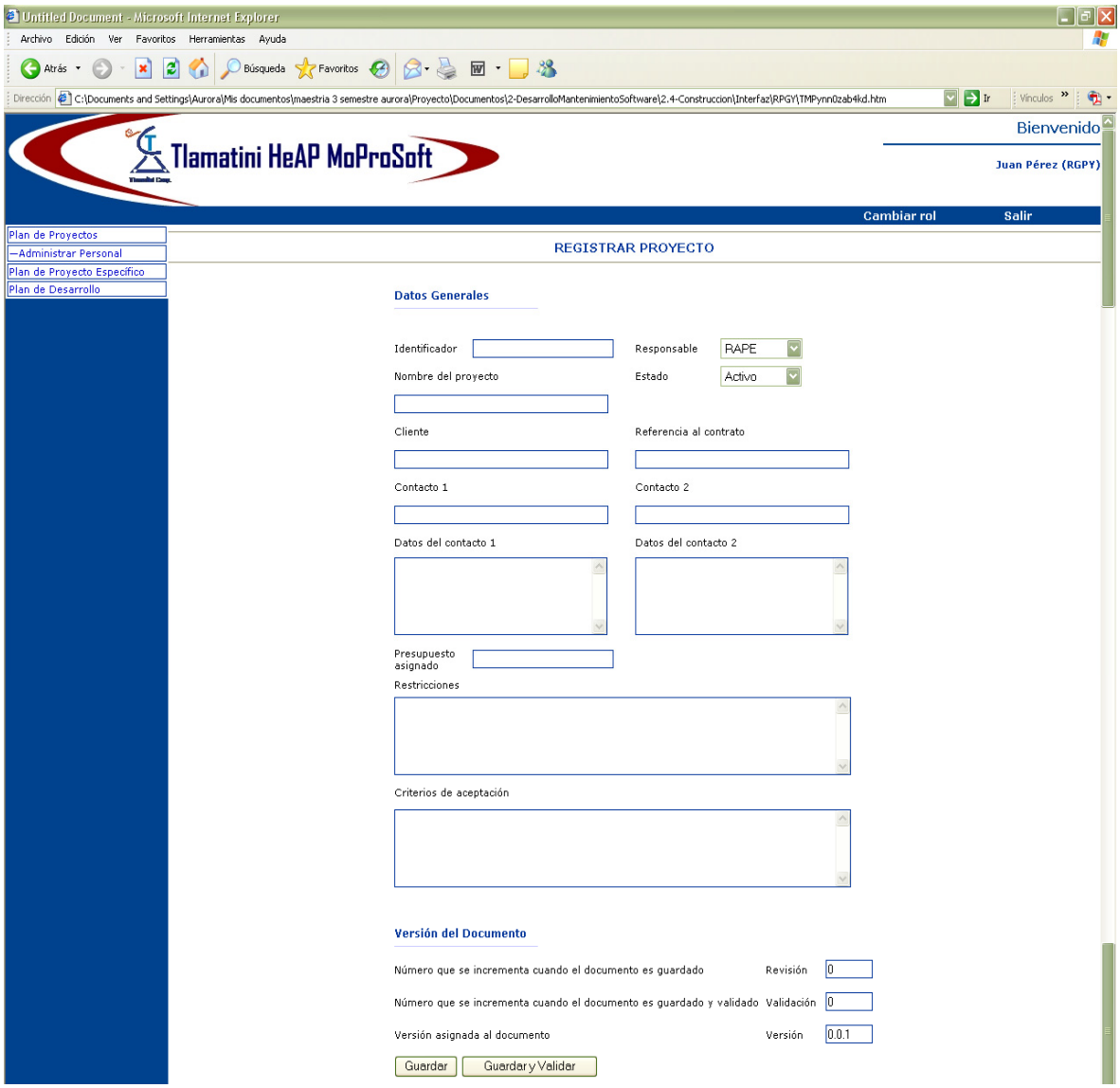
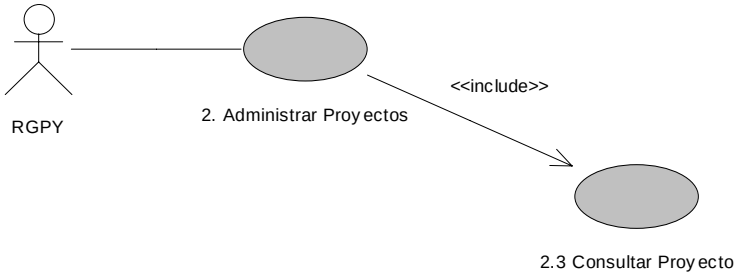


Figura 4.3 Pantalla “Registrar o modificar proyecto”

Caso de uso: 2.3 Consultar proyecto

Diagrama del caso de uso


Definición	
Caso de uso:	2.3 Consultar proyecto
Tipo:	Inclusión
Actor:	*RGPY*
Descripción:	En el caso de uso Consultar proyecto el *RGPY* podrá consultar los datos de los proyectos que se encuentran registrados en el sistema.
Precondiciones:	El *RGPY* seleccionó la opción "consultar" del listado de proyectos, eligiendo el proyecto que desea consultar.
Poscondiciones:	El *RGPY* realizó la consulta de los datos del proyecto que seleccionó.

Flujo de eventos

Flujo básico:

Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Consultar proyecto comienza cuando el *RGPY* selecciona la opción "consultar" del listado de proyectos eligiendo el proyecto que desea consultar.	2	El sistema le muestra la interfaz que contiene los datos del proyecto que seleccionó. [AE01]
3	El *RGPY* elige la opción "regresar al listado" y seguir con la consulta.	4	El sistema regresa a la interfaz del listado de proyectos registrados [AG01] [AG02].

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RGPY* sale del sistema al elegir la opción "Salir"
AG02	Cerrar sesión	El sistema cierra la sesión cuando el *RGPY* elige la opción "Cerrar sesión" y le permite elegir otro de sus roles para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RGPY* indicándole que ha sucedido un error de conexión con la base de datos.

Tabla 4.4 Detalle del caso de uso 2.3 Consultar Proyecto

El diseño de la pantalla para consultar el registro de un proyecto se muestra en la figura 4.4.

Pantalla “Consultar proyecto”

En esta pantalla se podrá consultar los datos de un proyecto.

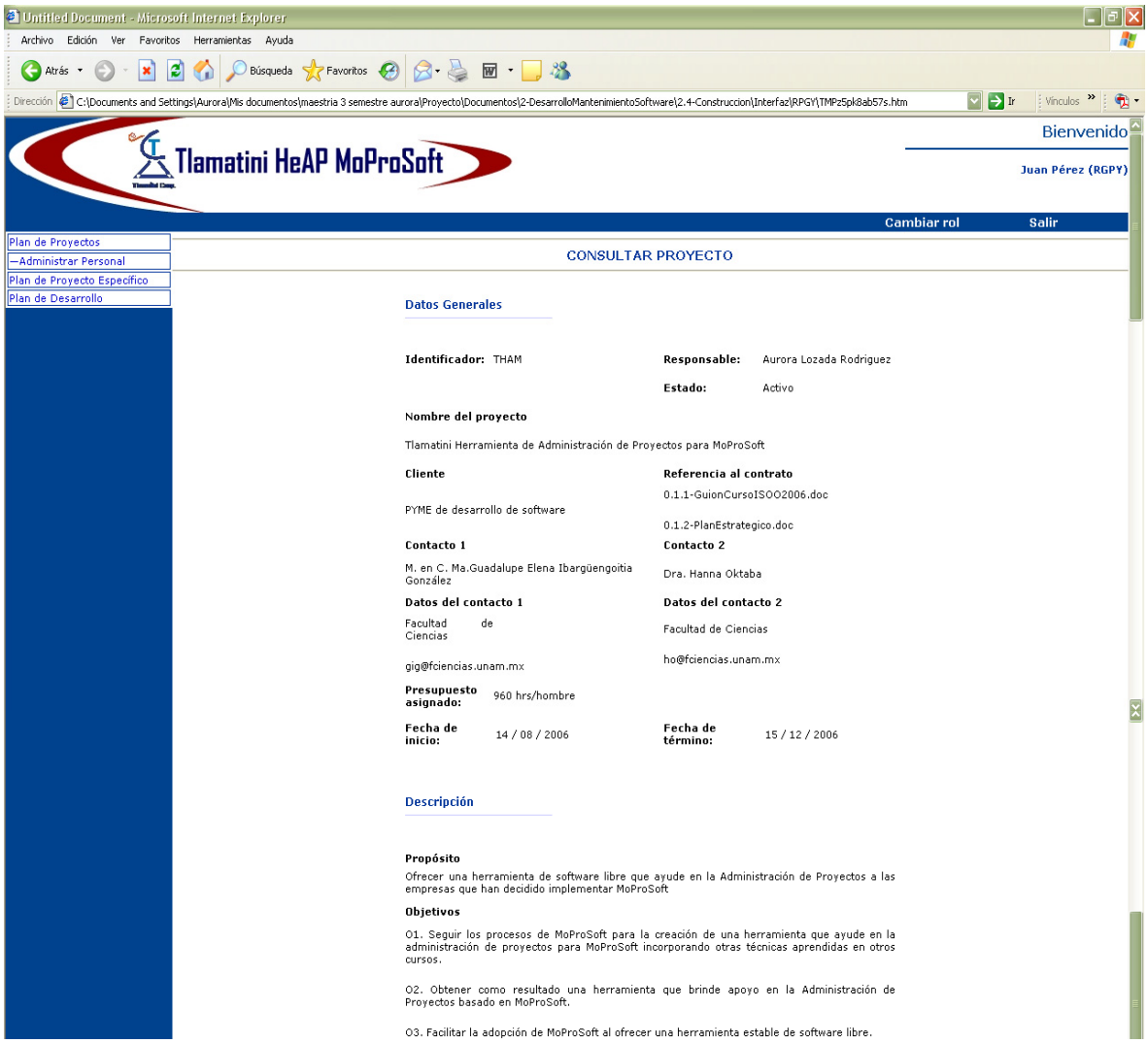
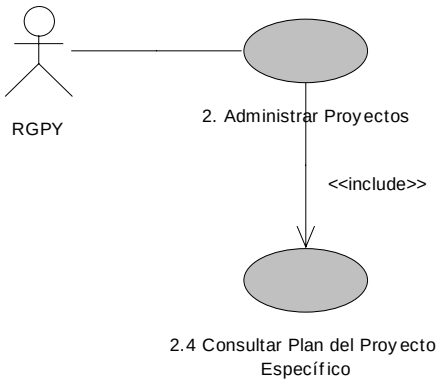


Figura 4.4 Pantalla “Consultar proyecto”

Caso de uso: 2.4 Consultar Plan del Proyecto Específico

Diagrama del caso de uso


Definición	
Caso de uso:	2.4 Consultar Plan del Proyecto Específico
Tipo:	Inclusión
Actor:	*RGPY*
Descripción:	En el caso de uso Consultar Plan del Proyecto Específico el *RGPY* podrá consultar los datos del Plan del Proyecto Específico de los proyectos que se encuentran registrados en el sistema.
Precondiciones:	El *RGPY* seleccionó la opción "Consultar el Plan del Proyecto Específico" del listado de proyectos, eligiendo el proyecto del que desea consultar el plan.
Poscondiciones:	El *RGPY* realizó la consulta del Plan del Proyecto Específico del proyecto que seleccionó.

Flujo de eventos

Flujo básico:

Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Consultar Plan del Proyecto Específico comienza cuando el *RGPY* selecciona la opción "Consultar el Plan del Proyecto Específico" del listado de proyectos eligiendo el proyecto del que desea consultar el plan.	2	El sistema le muestra la interfaz que contiene los datos del Plan del Proyecto Específico del proyecto que seleccionó. [AE01]
3	El *RGPY* elige la opción "regresar al listado" y seguir con otra actividad.	4	El sistema regresa a la interfaz del listado de proyectos registrados [AG01] [AG02].

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RGPY* sale del sistema al elegir la opción "Salir"
AG02	Cerrar sesión	El sistema cierra la sesión cuando el *RGPY* elige la opción "Cerrar sesión" y le permite elegir otro de sus roles para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RGPY* indicándole que ha sucedido un error de conexión con la base de datos.

Tabla 4.5 Detalle del caso de uso 2.4 Consultar Plan del Proyecto Específico

El diseño de la pantalla para consultar el plan de proyectos específicos se muestra en la figura 4.5.

Pantalla “Proyectos específicos”

Esta pantalla le muestra el RGPY la lista de los proyectos registrados en el sistema, y de los cuales podrá seleccionar alguno y consultar la información que comprende al Plan del Proyecto Específico.

Figura 4.5 Pantalla “Proyectos específicos”

4.2 Análisis

En la sección anterior describimos los requisitos para el desarrollo del sistema HeAP MoProsoft; ahora los analizaremos a partir del modelo de casos de uso, refinándolos y estructurándolos con el objetivo de desarrollar el modelo de análisis que nos ayude a conseguir una comprensión más precisa de los requisitos para generar una arquitectura estable y sólida.

4.2.1 Diagramas de clases del análisis

De acuerdo con el Proceso Unificado para el desarrollo de los diagramas de clases se debe analizar cada caso de uso para identificar tres estereotipos de clases: entidad, interfaz y control.

- Entidad: Este tipo de clases se utilizan para modelar la información correspondiente al dominio del problema que posee una vida larga y que es a menudo persistente.

- Interfaz: Las clases de interfaz se utilizan para modelar la interacción entre el sistema y sus actores (usuarios u otros sistemas externos), un objeto de este tipo son las interfaces de usuario o pantallas.

- Control: Son las clases que implementan el comportamiento o control de la lógica de los casos de uso, especificando cuándo y cómo el sistema cambia de estado.

Realizamos este análisis en cada uno de los cinco casos de uso (1. Iniciar Sesión, 2. Administrar Proyectos, 3. Administrar Personal, 4. Administrar Proyecto Específico y 5. Cerrar Sesión) que integran nuestro sistema, para obtener sus diagramas de clases correspondientes, comenzamos por identificar las clases de interfaz, que serán las pantallas que se diseñaron en el detalle de los casos

de uso, con las que los actores interactuarán directamente. Después identificamos las clases de entidad, tomando en cuenta que estas clases representan la información que estará almacenada en el sistema a corto y largo plazo. Finalmente las clases de control las identificamos porque llevan a cabo los cambios del control, la coordinación, la secuencia de las transacciones y en ocasiones operaciones complejas de la lógica del negocio.

Se eligió el caso de uso 2. Administrar Proyectos para explicar el desarrollo del análisis, diseño y pruebas, a lo largo de este trabajo de tesis, de la misma forma se desarrollaron los cuatro casos de uso restantes (ver Apéndice A[2]).

4.2.1.1 Diagramas de clases del análisis del caso de uso 2. Administrar Proyectos

Los diagramas de clases del análisis se realizaron identificando los tres estereotipos (interfaz, control e interfaz) explicados anteriormente.

Las clases de interfaz se obtuvieron de las pantallas que se diseñaron en el detalle de los casos de uso, cada pantalla corresponde con una clase de interfaz (ver figuras 4.2, 4.3, 4.4 y 4.5).

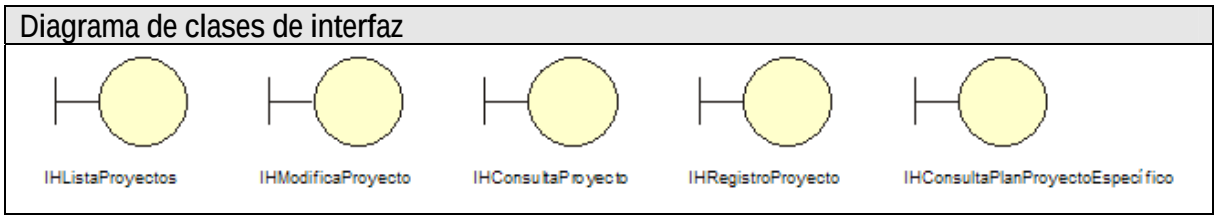


Figura 4.6 Diagrama de clases de interfaz

La clase de interfaz IHListaProyectos se utiliza para que interaccione el actor RGPY quien podrá consultar la lista de los proyectos que estén registrados en el sistema (ver figura 4.2).

IHModificaProyecto le permite al actor RGPY modificar los datos registrados del proyecto (ver figura 4.3).

En la clase IHConsultaProyecto el actor RGPY puede consultar la información registrada del proyecto (ver figura 4.4).

IHRegistroProyecto el actor RGPY interactúa a través de esta clase para realizar el registro de un proyecto (ver figura 4.3).

IHConsultaPlanProyectoEspecífica esta pantalla le permite al actor RGPY consultar el plan del proyecto específico, esta clase está detallada en el caso de uso 4. Administrar Proyecto Específico (ver figura 4.5).

[2] El Apéndice A se encuentra en la versión electrónica de la tesis.

Diagrama de clases de entidad

Figura 4.7 Diagrama de clases de entidad

La clase de entidad Proyecto representa a los Proyectos, que son gestionados por el actor RGPY utilizando las clases de interfaz que definimos anteriormente.

Las clases Persona y Rol, representan a las personas y sus roles dentro del sistema.

Diagrama de clases de control

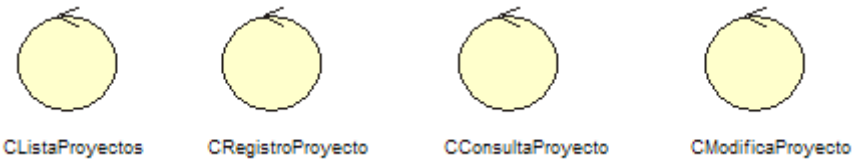


Figura 4.8 Diagrama de clases de control

La clase de control CListaProyectos es la encargada se listar los proyectos que se presentarán en la clase de interfaz IHListaProyectos.

CRegistroProyecto recibe los datos proporcionados por el actor RGPY a través de la clase de interfaz IHRegistroProyecto y los guarda en el sistema.

CConsultaProyecto realiza la consulta de los datos del proyecto presentados por la clase de interfaz IHConsultaProyecto.

CModificarProyecto permite realizar la modificación de los datos del proyecto registrados mediante la clase de interfaz IHModificaProyecto.

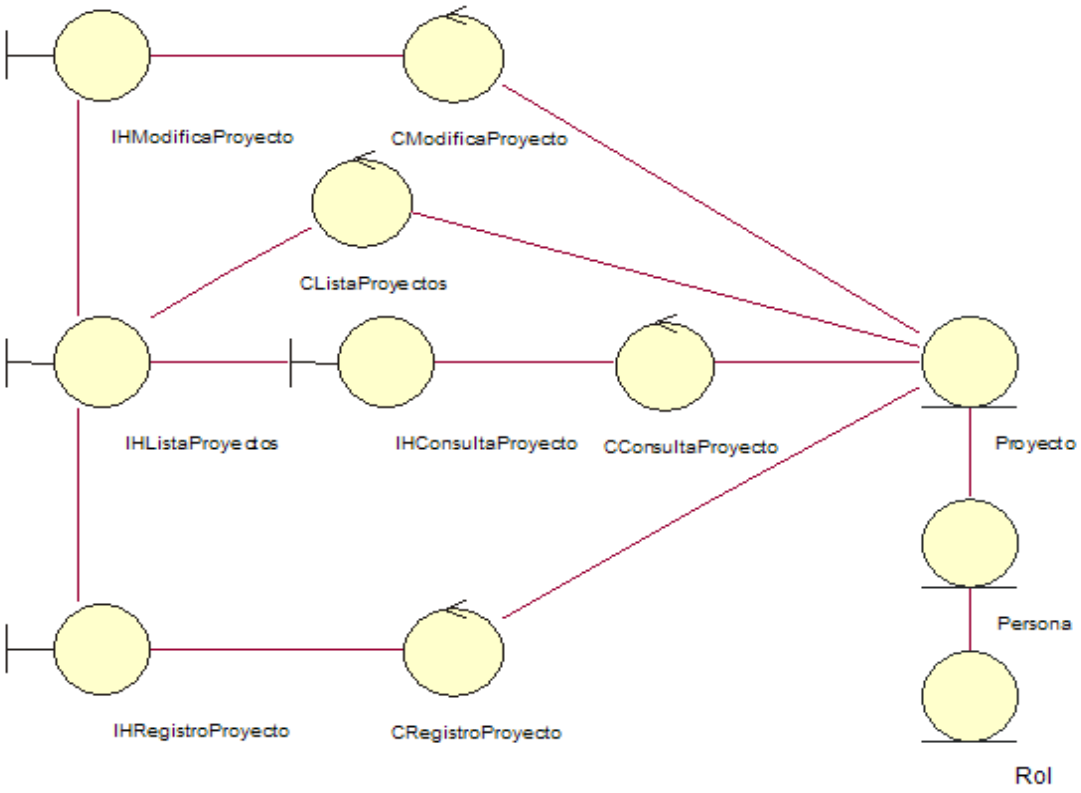


Figura 4.9 Diagrama de clases de interfaz, control y entidad del caso de uso 2. Administrar Proyectos.

El diagrama anterior muestra la relación entre las clases de interfaz, control y entidad de este caso de uso, lo que nos permitirá realizar un mejor diseño de la arquitectura del sistema.

De forma similar se hace para los demás casos de uso y se presentan en el Apéndice A[3].

4.3 Diseño

En el diseño se realiza el modelado del sistema para encontrar su forma, refinando y partiendo del modelo de análisis como entrada, en este momento tomamos en cuenta las consideraciones de ambiente en el que se desarrollará la implementación considerando los requisitos no funcionales y las restricciones relacionadas con las tecnologías de desarrollo (lenguajes de programación, sistemas operativos, componentes reutilizables, tecnologías de distribución y concurrencia, tecnologías de interfaz de usuario, etc.).

El modelo de diseño es un modelo físico porque se trata de un plano para la implementación que incluye los requisitos o subsistemas individuales, interfaces y clases. El modelo de diseño se debe desarrollar pensando en que la implementación debe descomponerse en partes más manejables.

[3] El Apéndice A se encuentra en la versión electrónica de la tesis.

4.3.1 Descripción de la arquitectura del diseño

La arquitectura de software consiste en dividir el software en partes y las relaciones entre ellas, lo que nos permitirá organizar de manera efectiva el desarrollo del sistema. La elección adecuada de la arquitectura depende de los requisitos del sistema y uno de los principales requisitos para el sistema HeAP MoProSoft es que sea una aplicación Web y que forme parte del software de distribución libre por lo que se decidió utilizar una arquitectura Java 2 Enterprise Edition (J2EE), la cual cuenta con un conjunto de especificaciones para desarrollar aplicaciones Web.

Como se explicó en el Capítulo 2, la figura 4.10 nos muestra la arquitectura de struts, ahora le integraremos en el modelo la implementación del patrón DAO.

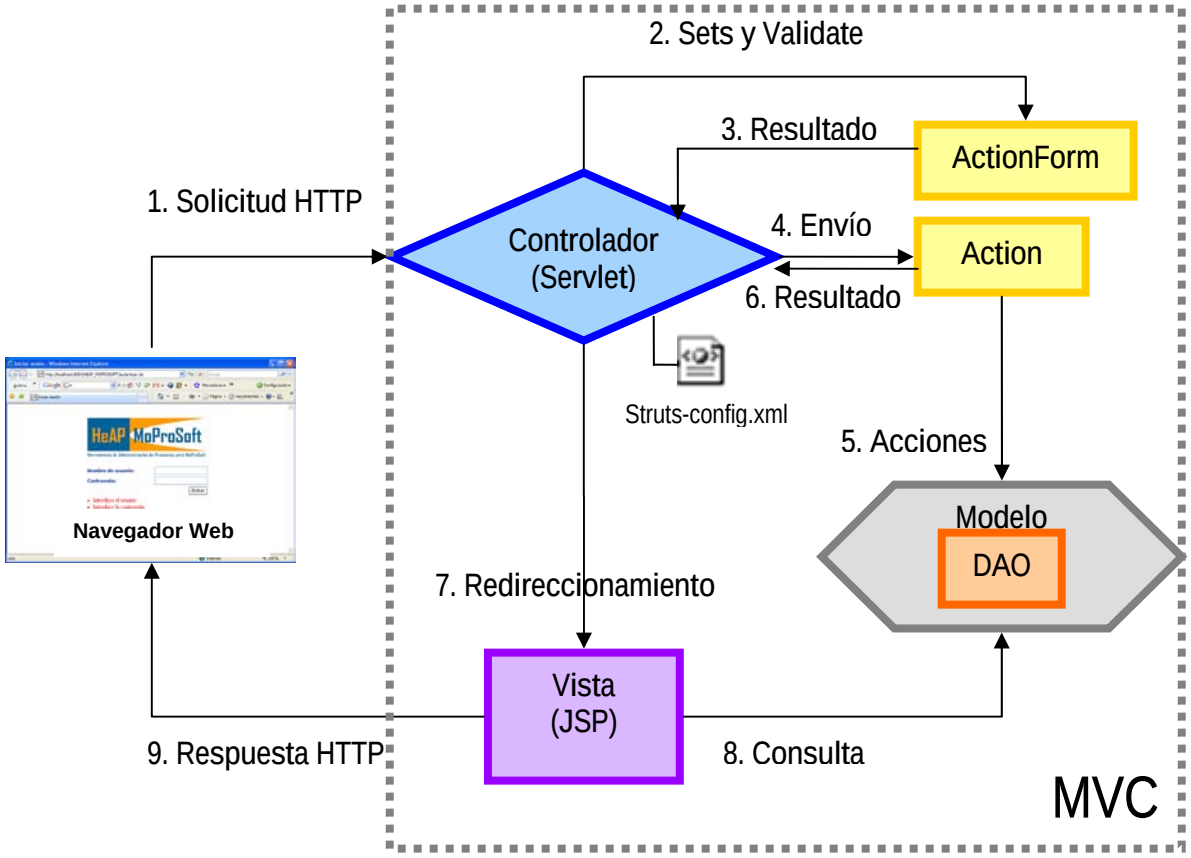


Figura 4.10 Arquitectura Struts y DAO del sistema

Finalmente la arquitectura que definimos se muestra en su totalidad en la figura 4.11, utilizando una arquitectura cliente servidor, en donde en la parte del cliente tenemos el navegador Web que desplegará la aplicación, del lado del servidor tendremos el contenedor Web Apache Tomcat[4] que albergará los Servlets y las páginas JSP de Java. En el servidor también estará el servidor de base de datos PostgreSQL 8.1.

[4] http://tomcat.apache.org/

La comunicación del entre el cliente y el servidor se realiza utilizando el protocolo HTTP, el navegador Web manda una solicitud vía HTTP al servidor, el servidor la recibe y la procesa, ya del lado del servidor el nivel medio se comunica con el nivel EIS, por medio de un JCBC para tener acceso a la fuente de datos y consultar la información solicitada, finalmente una vez procesada la información el servidor le manda la respuesta al cliente.

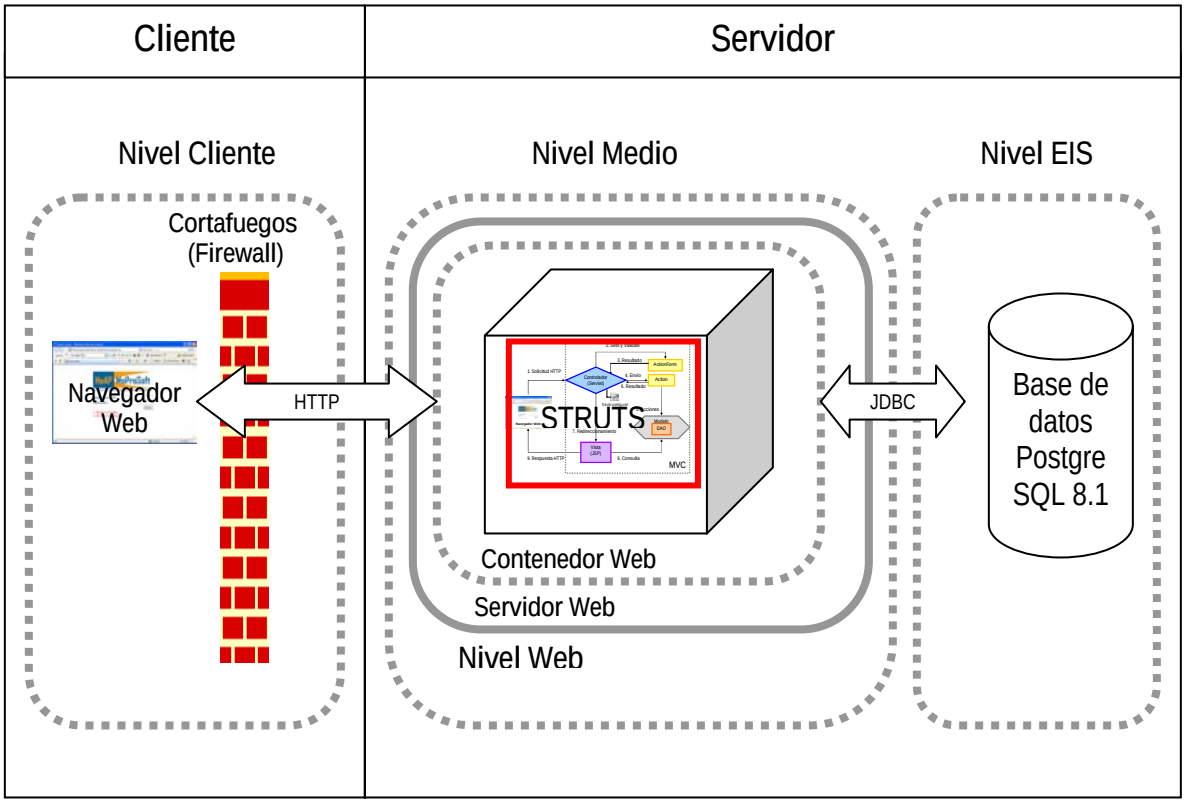


Figura 4.11 Arquitectura Web del sistema

4.3.2 Diagramas de clases del diseño

El modelo de diseño se realiza a partir del modelo del análisis y de acuerdo a la arquitectura que definimos. Las clases del análisis (interfaz, control y entidad) que identificamos en el modelo del análisis, fueron divididas según el marco de trabajo Struts y el patrón DAO que describimos en el capítulo 3.

4.3.2.1 Diagramas de clases del diseño del caso de uso 2. Administrar Proyectos

Los diagramas de clases del diseño son una abstracción de una clase que se codificará durante la implementación, por lo tanto se diseñan tomando en cuenta la arquitectura definida, el lenguaje que se utilizará, la visibilidad de los atributos y las operaciones, lo métodos deben corresponder de manera directa en la implementación.

Los diagramas de clases del diseño se obtuvieron mapeando las clases del análisis en clases propias de Struts y del patrón DAO. El mapeo se realizó de la siguiente forma:

Clases propias de Struts:

- Las clases del análisis de interfaz se dividieron en dos clases: un JSP y una clase ActionForm.
- Las clases de análisis de control corresponden a una clase Action.

Clases implementando el patrón DAO:

- Las clases de entidad se dividieron en dos clases: una clase Bean y una DAO.

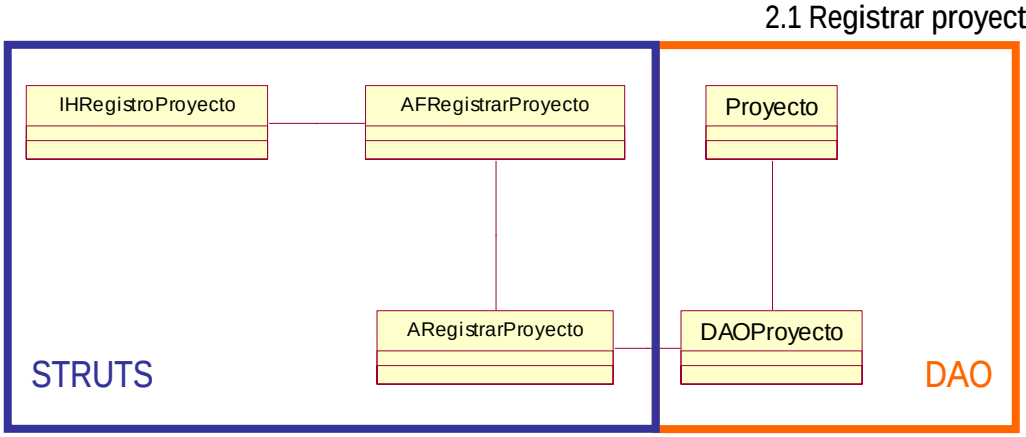


Figura 4.12 Diagrama de clases del diseño del caso de uso 2.1 Registrar Proyecto.

La siguiente tabla muestra el mapeo de las clases del análisis a las clases del diseño, de acuerdo con la implementación de Struts y del patrón DAO.

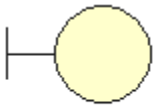
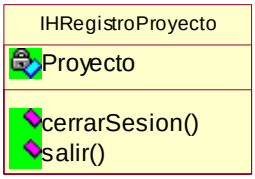
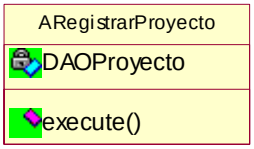
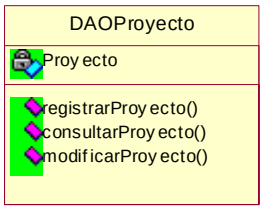
Clases del análisis	Clases del diseño	
 IHRegistroProyecto	IHRegistroProyecto	IHRegistroProyecto esta clase es un JSP de Java.
	AFRegistrarProyecto	AFRegistrarProyecto es una clase ActionForm de Struts.
 CRegistroProyecto	ARegistrarProyecto	ARegistrarProyecto es una clase Action de Struts, es decir el Servlet controlador.
 Proyecto	Proyecto	Proyecto es una clase JavaBean.
	DAOProyecto	DAOProyecto es la clase que se encarga de las operaciones CRUD (Create, Read, Update, Delete) para la información almacenada en la base de datos.

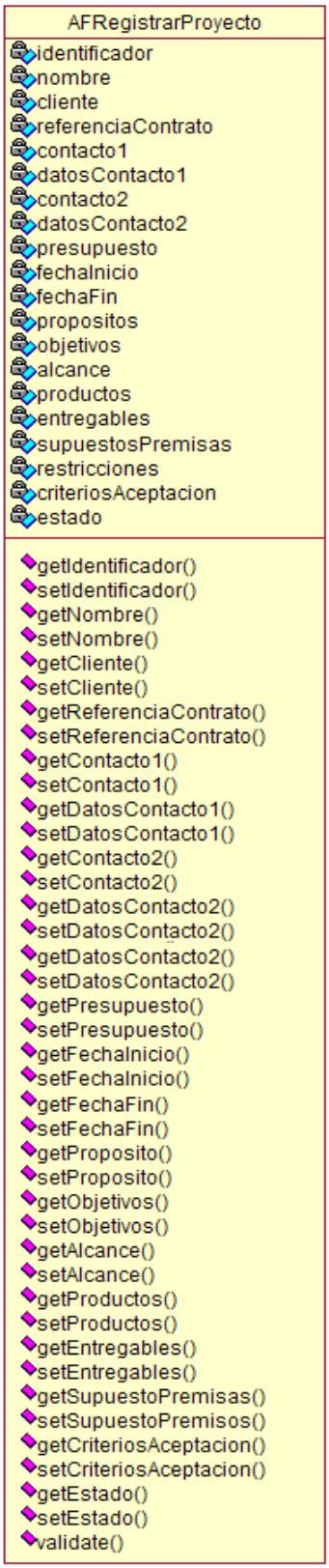
Tabla 4.6 Mapeo de las clases del análisis a las clases del diseño del caso de uso 2.1 Registrar Proyecto.

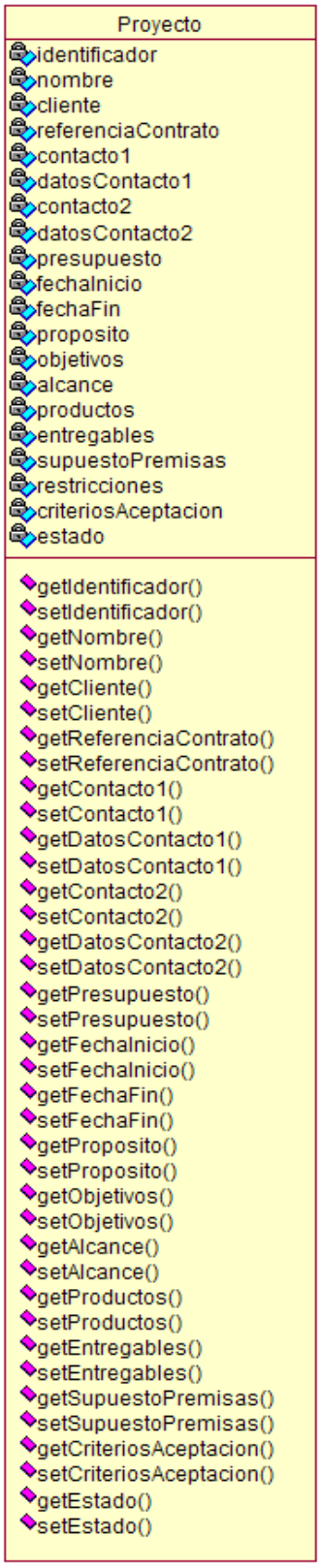
Las siguientes imágenes son las clases del diseño ya con sus atributos y métodos de este caso de uso:











2.2 Modificar proyecto

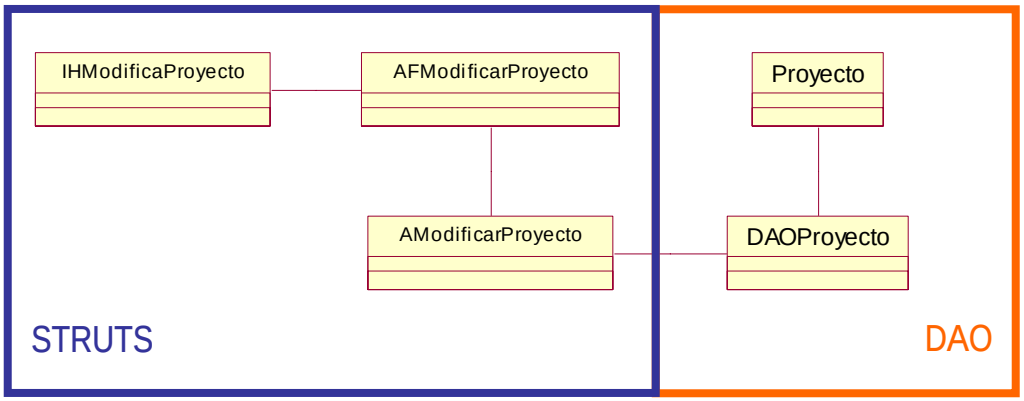


Figura 4.13 Diagrama de clases del diseño del caso de uso 2.2 Modificar Proyecto

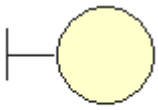
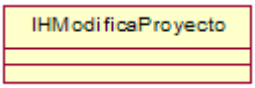
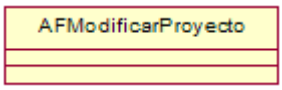
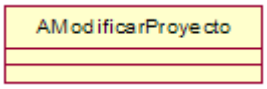
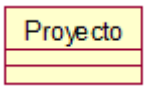
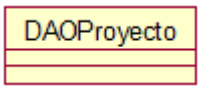
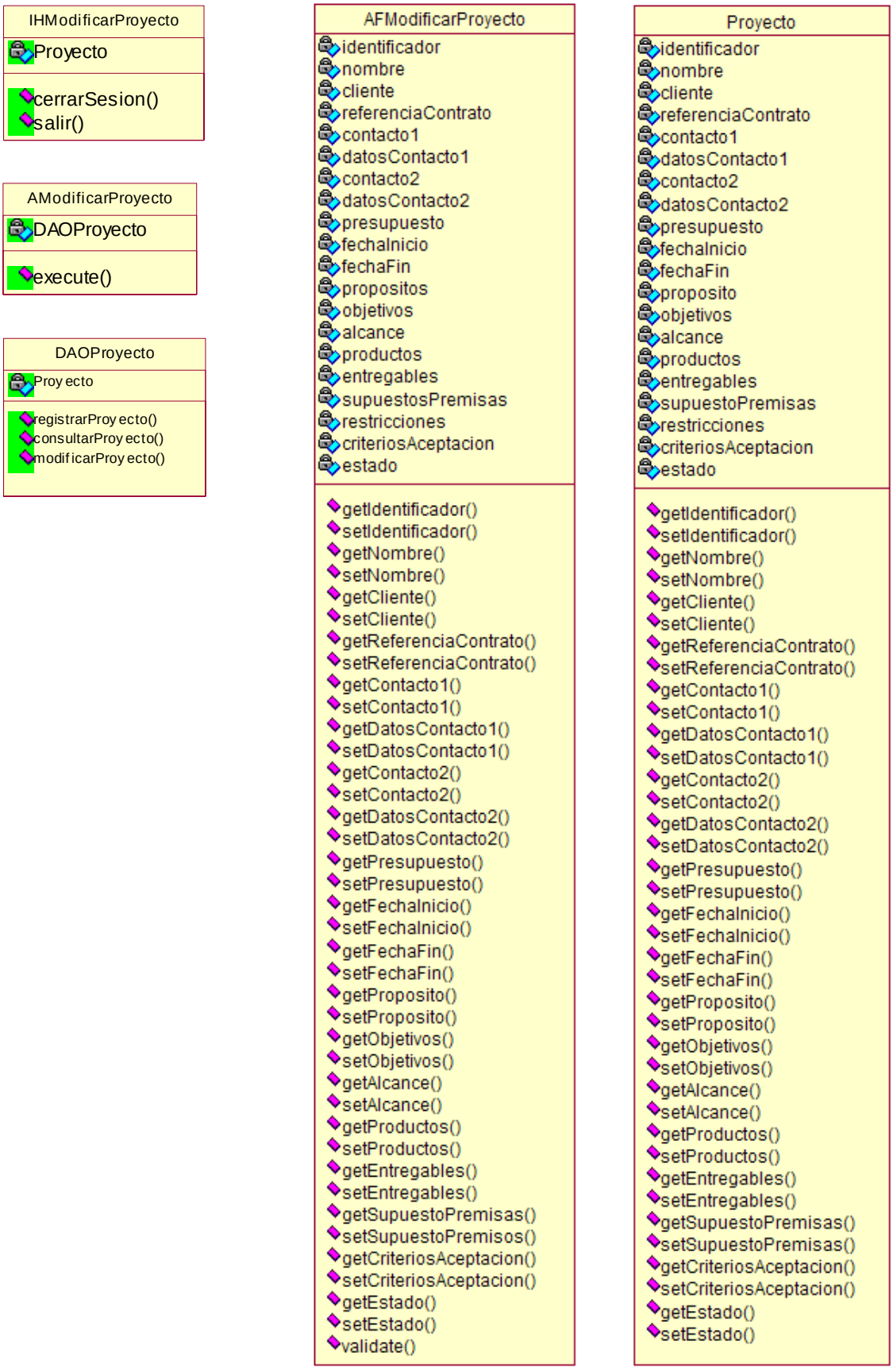
Clases del análisis	Clases del diseño	
 IHModificaProyecto	 IHModificaProyecto	IHModificaProyecto esta clase es un JSP de Java.
	 AFModificarProyecto	AFModificarProyecto es una clase ActionForm de Struts.
 CModificaProyecto	 AModificarProyecto	AModificarProyecto es una clase Action de Struts, es decir el Servlet controlador.
 Proyecto	 Proyecto	Proyecto es una clase JavaBean.
	 DAOProyecto	DAOProyecto es la clase que se encarga de las operaciones CRUD (Create, Read, Update, Delete) para la información almacenada en la base de datos.

Tabla 4.7 Mapeo de las clases del análisis a las clases del diseño del caso de uso 2.2 Modificar Proyecto.

Las siguientes imágenes son las clases del diseño ya con sus atributos y métodos para este caso de uso:


IHModificarProyecto
Proyecto
cerrarSesion()
salir()
AModificarProyecto
DAOProyecto
execute()
DAOProyecto
Proyecto
registrarProyecto()
consultarProyecto()
modificarProyecto()
AFModificarProyecto
identificador
nombre
cliente
referenciaContrato
contacto1
datosContacto1
contacto2
datosContacto2
presupuesto
fechaInicio
fechaFin
propositos
objetivos
alcance
productos
entregables
supuestosPremisas
restricciones
criteriosAceptacion
estado
getIdentificador()
setIdentificador()
getNombre()
setNombre()
getCliente()
setCliente()
getReferenciaContrato()
setReferenciaContrato()
getContacto1()
setContacto1()
getDatosContacto1()
setDatosContacto1()
getContacto2()
setContacto2()
getDatosContacto2()
setDatosContacto2()
getPresupuesto()
setPresupuesto()
getFechaInicio()
setFechaInicio()
getFechaFin()
setFechaFin()
getProposito()
setProposito()
getObjetivos()
setObjetivos()
getAlcance()
setAlcance()
getProductos()
setProductos()
getEntregables()
setEntregables()
getSupuestoPremisas()
setSupuestoPremisos()
getCriteriosAceptacion()
setCriteriosAceptacion()
getEstado()
setEstado()
validate()
Proyecto
identificador
nombre
cliente
referenciaContrato
contacto1
datosContacto1
contacto2
datosContacto2
presupuesto
fechaInicio
fechaFin
proposito
objetivos
alcance
productos
entregables
supuestoPremisas
restricciones
criteriosAceptacion
estado
getIdentificador()
setIdentificador()
getNombre()
setNombre()
getCliente()
setCliente()
getReferenciaContrato()
setReferenciaContrato()
getContacto1()
setContacto1()
getDatosContacto1()
setDatosContacto1()
getContacto2()
setContacto2()
getDatosContacto2()
setDatosContacto2()
getPresupuesto()
setPresupuesto()
getFechaInicio()
setFechaInicio()
getFechaFin()
setFechaFin()
getProposito()
setProposito()
getObjetivos()
setObjetivos()
getAlcance()
setAlcance()
getProductos()
setProductos()
getEntregables()
setEntregables()
getSupuestoPremisas()
setSupuestoPremisas()
getCriteriosAceptacion()
setCriteriosAceptacion()
getEstado()
setEstado()

2.3 Consultar proyecto

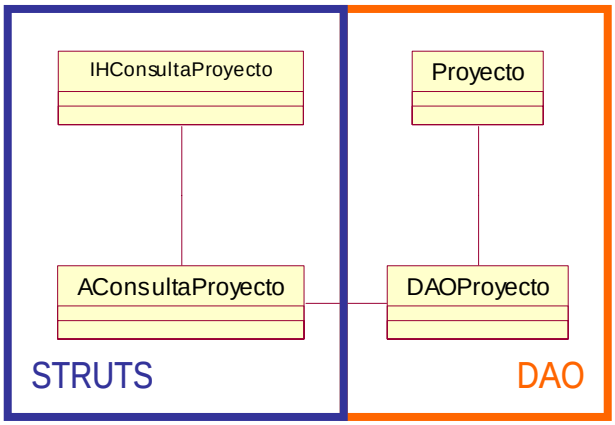


Figura 4.14 Diagrama de clases del diseño del caso de uso 2.3 Consultar Proyecto.

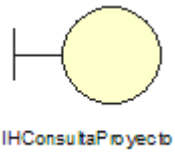
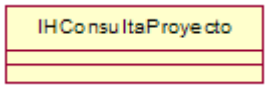
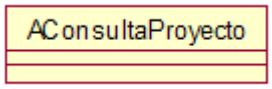
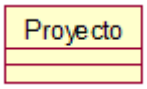
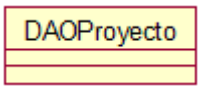
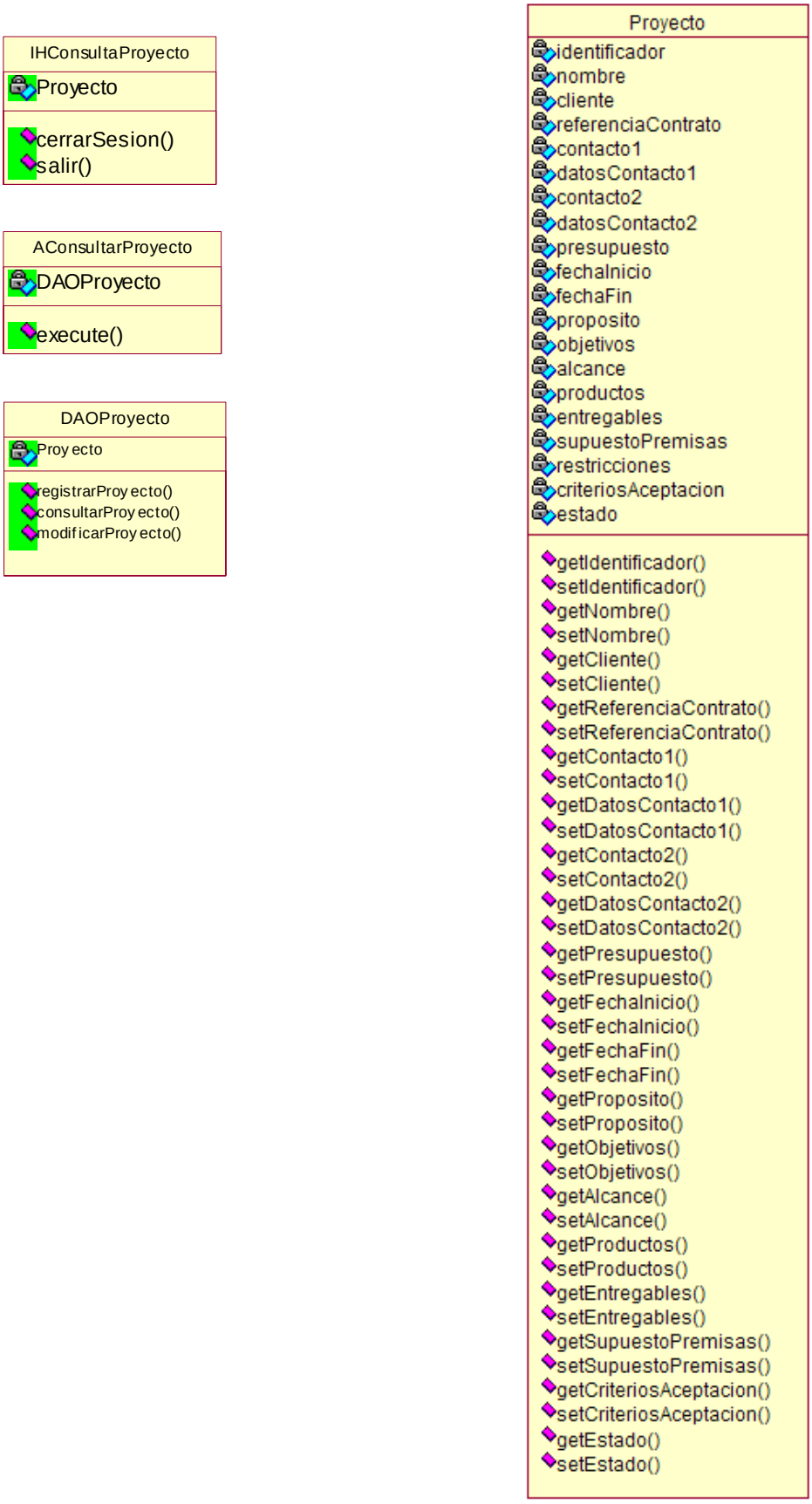
Clases del análisis	Clases del diseño	
		IHConsultaProyecto esta clase es un JSP de Java.
		AConsultaProyecto es una clase Action de Struts, es decir el Servlet controlador.
		Proyecto es una clase JavaBean.
		DAOProyecto es la clase que se encarga de las operaciones CRUD (Create, Read, Update, Delete) para la información almacenada en la base de datos.

Tabla 4.8 Mapeo de las clases del análisis a las clases del diseño del caso de uso 2.3 Consultar Proyecto.

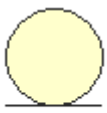
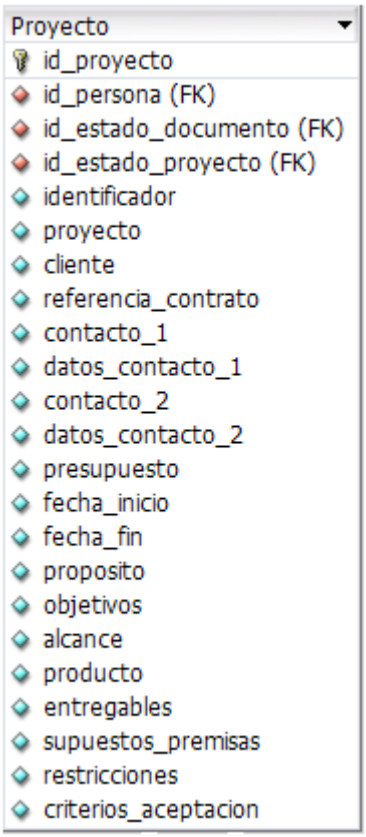
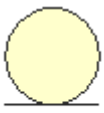
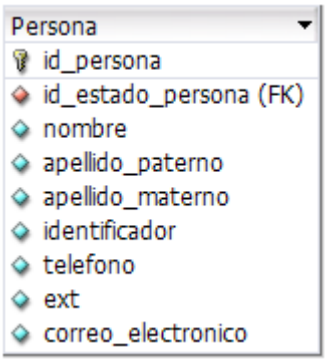

IHConsultaProyecto
Proyecto
cerrarSesion()
salir()
AConsultarProyecto
DAOProyecto
execute()
DAOProyecto
Proyecto
registrarProyecto()
consultarProyecto()
modificarProyecto()
Proyecto
identificador
nombre
cliente
referenciaContrato
contacto1
datosContacto1
contacto2
datosContacto2
presupuesto
fechaInicio
fechaFin
proposito
objetivos
alcance
productos
entregables
supuestoPremisas
restricciones
criteriosAceptacion
estado
getIdentificador()
setIdentificador()
getNombre()
setNombre()
getCliente()
setCliente()
getReferenciaContrato()
setReferenciaContrato()
getContacto1()
setContacto1()
getDatosContacto1()
setDatosContacto1()
getContacto2()
setContacto2()
getDatosContacto2()
setDatosContacto2()
getPresupuesto()
setPresupuesto()
getFechaInicio()
setFechaInicio()
getFechaFin()
setFechaFin()
getProposito()
setProposito()
getObjetivos()
setObjetivos()
getAlcance()
setAlcance()
getProductos()
setProductos()
getEntregables()
setEntregables()
getSupuestoPremisas()
setSupuestoPremisas()
getCriteriosAceptacion()
setCriteriosAceptacion()
getEstado()
setEstado()

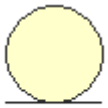
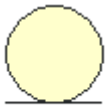
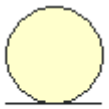
4.3.3 Diseño de la base de datos

En el desarrollo de sistemas las bases de datos tienen un papel fundamental, por lo que se requiere de una decisión estratégica para realizar la elección más apropiada del tipo de base de datos que se utilizará. Para el desarrollo de HeAP MoProSoft, decidimos utilizar el modelo de base de datos del tipo relacional por ser un modelo ampliamente conocido y utilizado, además de cumplir con nuestras necesidades.

Para diseñar la base de datos realizamos una traducción del modelo del dominio de problema a un modelo compuesto por tablas en donde las clases de entidad que identificamos durante el análisis se convirtieron a tablas de la base de datos.

La siguiente tabla muestra la clase de entidad y la tabla de la base de datos que le corresponde:

Clases de entidad	Tablas de la base de datos	Clases de entidad	Tablas de la base de datos
 Proyecto		 Persona	

Clases de entidad	Tablas de la base de datos	Clases de entidad	Tablas de la base de datos
 Rol	Rol id_rol rol identificador	 Usuario	Usuario id_usuario id_persona (FK) usuario contrasena
 Equipo de Trabajo	Equipo_Trabajo id_equipo_trabajo id_rol (FK) id_persona (FK) id_proyecto (FK) id_estado_documento (FK) version fecha_version	 Riesgo	Riesgo id_riesgo id_proyecto (FK) id_estado_documento (FK) id_tipo_riesgo (FK) id_probabilidad_riesgo (FK) id_estado_riesgo (FK) id_impacto_riesgo (FK) version fecha_version riesgo descripcion fecha_identificacion afectacion decision plan_mitigacion plan_contingencia
 Costo Estimado	Costo_Estimado id_costo_estimado id_proyecto (FK) id_estado_documento (FK) moneda hardware software viajes reubicacion subcontratos mantenimiento capacitacioin espacio recursos_humanos comunicaciones administracion_configuracion soporte_documentacion otros total version fecha_version	 Adquisición	Adquisicion id_adquisicion id_estado_documento (FK) id_proyecto (FK) id_estado_adquisicion (FK) id_tipo_adquisicion (FK) adquisicion unidades costo_unitario_estimado costo_unitario_real fecha_requerido fecha_adquirido version fecha_version

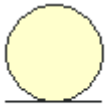
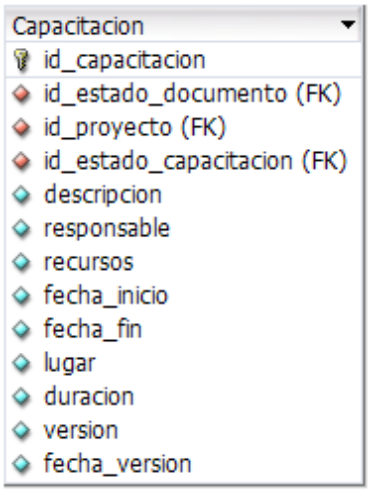
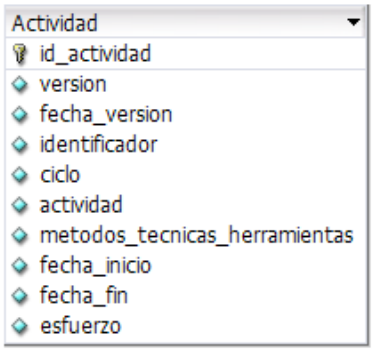
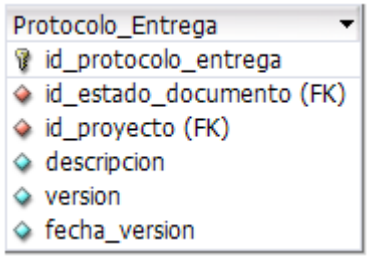
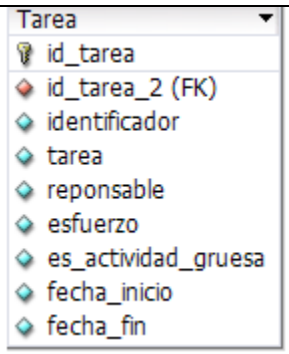
Clases de entidad	Tablas de la base de datos	Clases de entidad	Tablas de la base de datos
 Capacitación		 Actividad	
 Protocolo de Entrega		 Tarea	

Tabla 4.9 Tabla de clases de entidad y tablas de la base de datos.

A continuación se muestra el diseño completo de la base de datos:

4.3.3.1 Diseño lógico de la base de datos

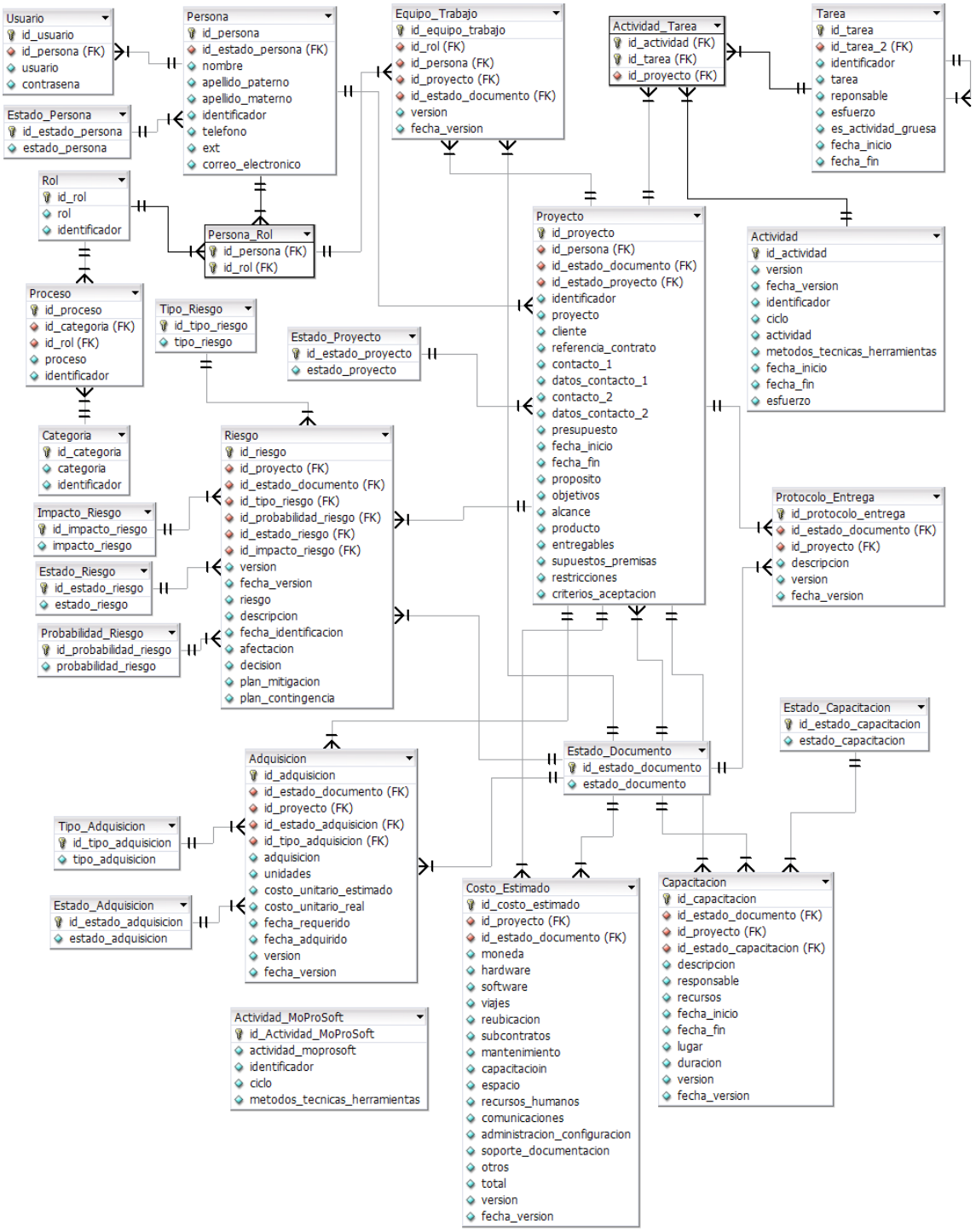


Figura 4.15 Diseño lógico de la base de datos

El sistema manejador de base de datos que utilizamos es PostgreSQL 8.1 [10] por formar parte del software de distribución libre ya que esta es una de las restricciones de la herramienta.

PostgreSQL es un sistema manejador de base de datos del tipo objeto-relacional (ORDBMS), soporta SQL y ofrece algunas características como:

- Queries complejos
- Llaves foráneas
- Triggers
- Vistas
- Integridad transaccional
- Control de concurrencia multiversión

PostgreSQL es un ORDBMS multiplataforma y por formar parte del software libre, puede ser usado modificado y distribuido para cualquier tipo de uso sin costo alguno. Sus ventajas son:

- Es software libre por lo que no genera costo por licencias.
- Es confiable y cuenta con excelente estabilidad.
- Es Multiplataforma.
- Soporta grandes volúmenes de información.
- El código fuente está disponible para realizar las modificaciones necesarias.
- Se diseñó pensando en un bajo costo por mantenimiento.
- Cuenta con un excelente soporte.

4.3.4 Diagrama de paquetes

Para desarrollar el sistema se definieron tres paquetes: Modelo, Vista y Controlador para agrupar las clases, quedando como se muestra en el siguiente diagrama:

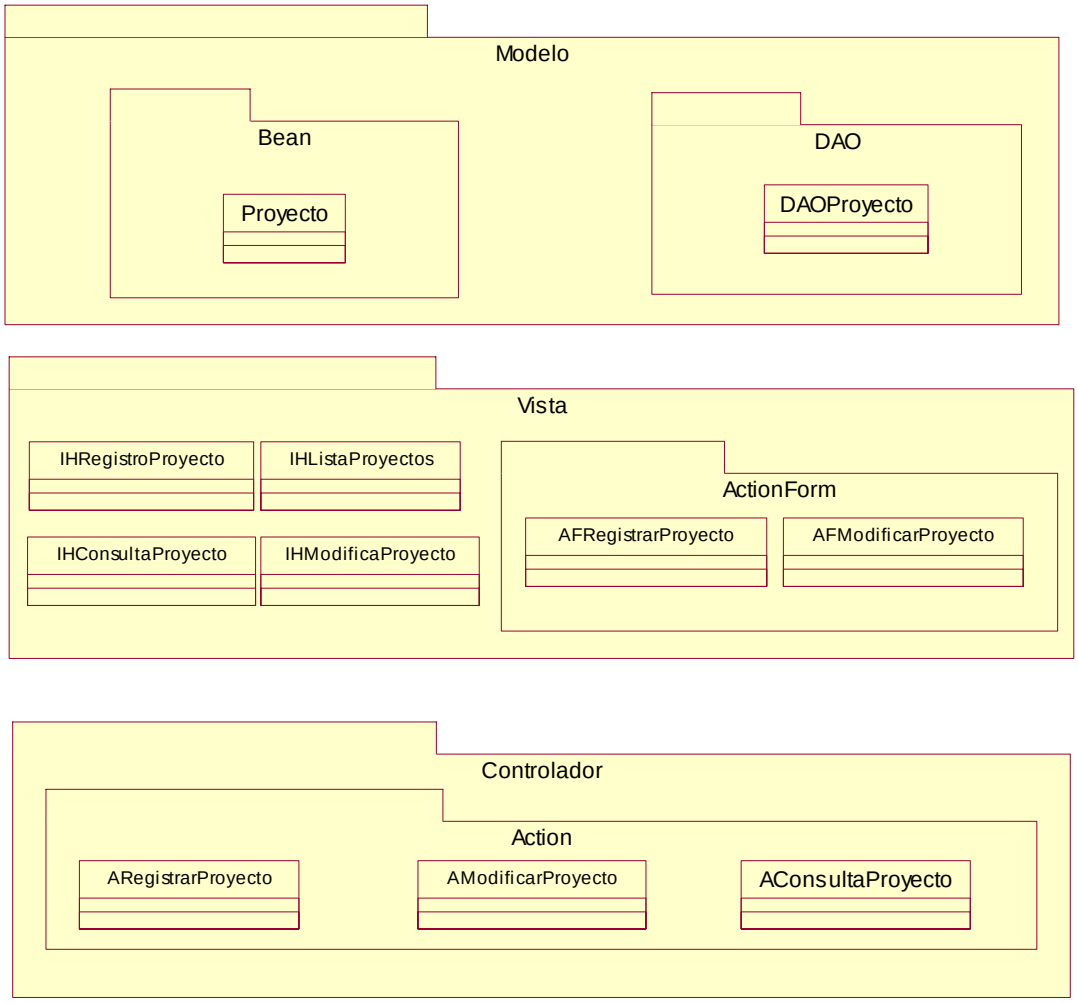


Figura 4.16 Diagrama de paquetes

4.3.5 Diagrama de distribución

En el siguiente diagrama de distribución se muestran la relación entre los elementos que integran el sistema.

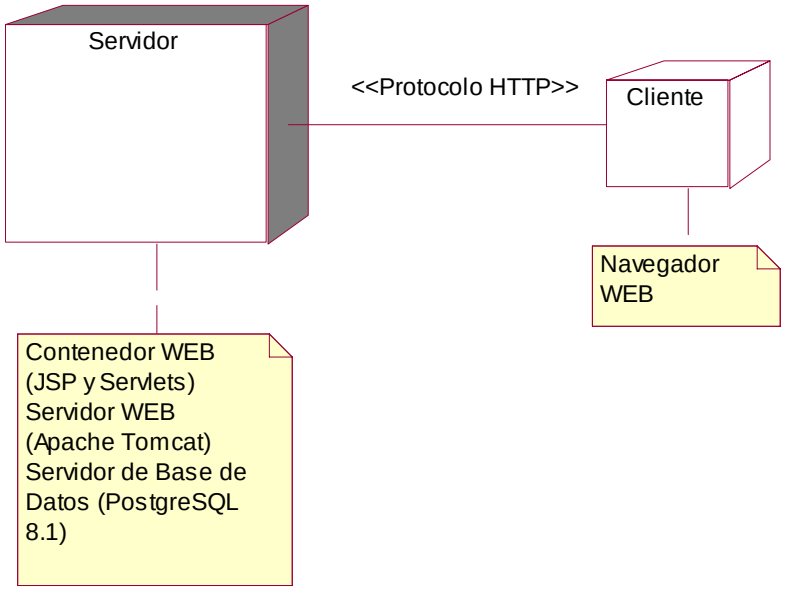


Figura 4.17 Diagrama de distribución

4.4 Implementación

En la implementación tomamos los diagramas que obtuvimos en el diseño, para codificar las clases. Para el desarrollo de la aplicación se decidió utilizar el Integrated Development Environment (IDE) NetBeans 5.5 [11] de Sun Microsystems, perteneciente al software de distribución libre, excelente para desarrollo utilizando tecnologías Java WEB ya que soporta la plataforma J2EE y el marco de trabajo Struts, además de contar con la tecnología necesaria para conexión con bases de datos en este caso con PostgreSQL, mediante un Java DataBase Connectivity (JDBC). Netbeans es un IDE multiplataforma (Windows, Linux, Mac OS X y Solaris). Netbeans 5.5 trae incluido el marco de trabajo Struts 1.2.9, que es la versión que utilizamos.

La organización de la estructura de archivos que definimos se muestra en la siguiente figura:

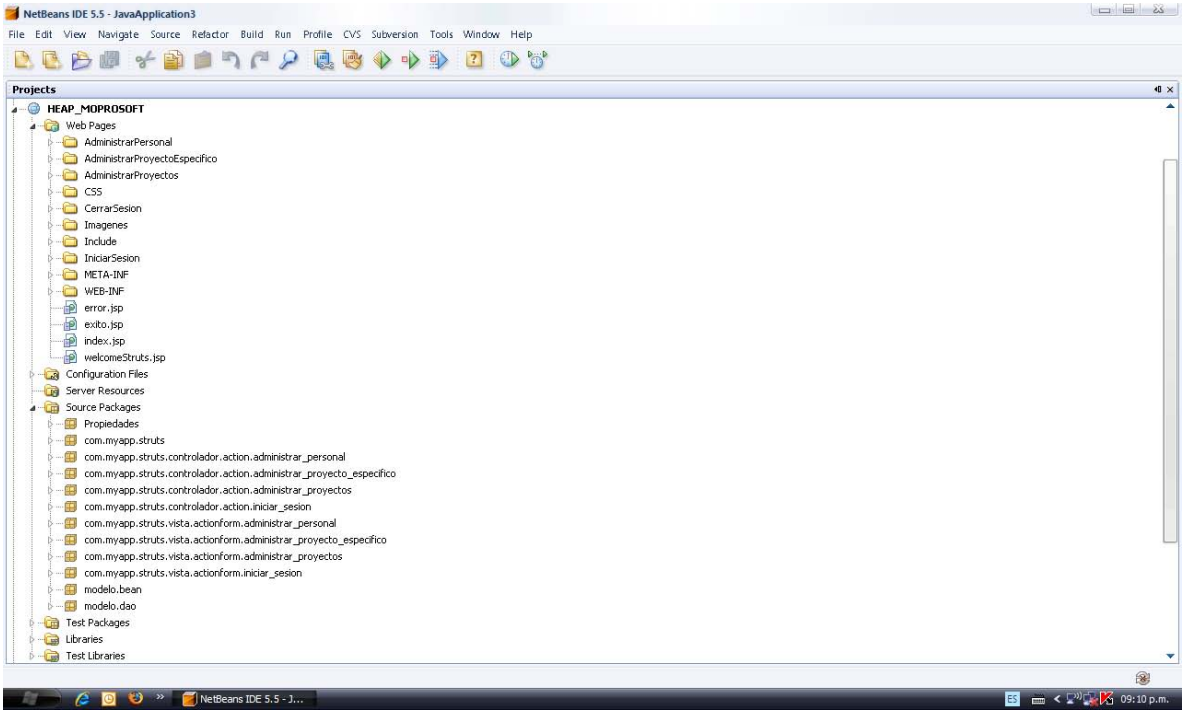


Figura 4.18 Estructura de archivos en NetBeans.

En la carpeta Web Pages se crearon cinco carpetas, una para cada caso de uso, en donde se almacenaron los JSPs que son las clases de interfaz y forman parte de la vista. La carpeta WEB-INF contiene los archivos XML de configuración (web.xml) y navegación (struts-config.xml), que forman parte del controlador. La siguiente figura muestra la estructura de esta carpeta y sus subcarpetas:

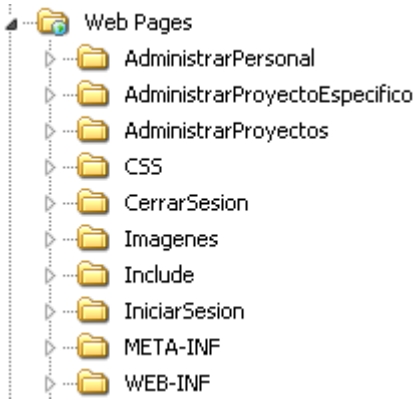


Figura 4.19 Carpeta Web Pages

La carpeta Source Packages contiene el paquete com.myapp.struts para crear las clases Action y ActionForm de Struts, para tener una estructura más organizada adicionalmente se crearon los siguientes paquetes:

Un paquete de controlador y dentro un paquete action (com.myapp.struts.controlador.action). En el paquete action se crearon cinco paquetes, un paquete para cada caso de uso, para almacenar las clases action utilizadas en cada caso de uso, son las clases de control que integran el controlador.

Un paquete de vista y dentro un paquete actionform (com.myapp.struts.vista.actionform). En el paquete actionform también se crearon cinco paquetes, un paquete para cada caso de uso, que contiene las clases actionform utilizadas en cada caso de uso, son parte de las clases de interfaz y parte de la vista.

La siguiente tabla muestra esta distribución de paquetes:

com	myapp	struts	controlador	action	iniciar_sesion
					administrar_proyectos
					administrar_personal
					administrar_proyecto_especifico
					cerrar_sesion
			vista	actionform	iniciar_sesion
					administrar_proyectos
					administrar_personal
					administrar_proyecto_especifico
					cerrar_sesion

Tabla 4.10 Distribución de paquetes de Struts en NetBeans.

La carpeta Source Packages contiene también un paquete al que nombramos modelo y dentro creamos dos paquetes: bean (modelo.bean) y dao (modelo.dao). En el paquete bean se encuentran las clases del tipo JavaBean y el paquete dao contiene las clases DAO que implementamos para el acceso a datos, son clases de entidad y forman parte del modelo. En la siguiente figura se muestra la estructura de estos paquetes:

Figura 4.20 Source Packages

4.5 Pruebas

La realización de pruebas es parte fundamental del desarrollo de sistemas por lo que se debe probar a lo largo de todo el proceso de desarrollo de software. De manera general las pruebas se clasifican en *pruebas de verificación y validación*. Durante las pruebas de verificación se revisa que se esté construyendo el sistema de manera correcta y en la validación se revisa que se esté construyendo lo que el cliente realmente quiere, es decir el sistema correcto, con el fin de garantizar la satisfacción del cliente. La validación con el cliente se lleva a cabo durante la definición de requisitos para garantizar que sean correctos, los prototipos se prueban con el cliente para validar la funcionalidad deseada. Las pruebas finales van encaminadas a certificar la calidad final del sistema. De aquí en adelante nos enfocaremos en las pruebas de verificación del sistema.

Prueba de Unidad

A lo largo de la implementación y conforme se va codificando el sistema de manera simultanea se van realizando pruebas de unidad, en donde probamos métodos y clases individualmente. Las pruebas de unidad que se realizaron son de dos tipos:

1. *Prueba estructural o de caja blanca*, la cual tiene como objetivo revisar el “cómo” se realizan las cosas conociendo el diseño interno de la unidad.
2. *Prueba de especificación o de caja negra*, cuyo objetivo es saber “qué” hace la unidad, esta prueba se realiza enviando entradas y revisando que la salida sea la esperada.

Prueba de Integración

Después de haber realizado las pruebas de unidad, es decir de haber probado de manera individual métodos y clases, debemos realizar las pruebas de integración que consisten en ir integrando todas las unidades en unidades más grandes hasta tener el sistema completo. Las pruebas de integración se realizan sobre los casos de uso, probando primero los flujos básicos para después centrarse en los flujos alternos. De igual manera primero se prueban los casos de uso básicos y después los casos de uso de extensión o inclusión.

Prueba de Sistema

Una vez que se han terminado de probar los casos de uso de manera aislada y conforme al modelo de requisitos, se prueba el sistema entero ejecutando varios casos de uso en paralelo sometiéndolos a diferentes escenarios [12].

Antes de hacer las pruebas se realiza el plan de pruebas con el fin de saber exactamente que probar y poder detectar fallas. Se establecen rangos y tipos de valores de entrada para establecer las salidas para cada caso de prueba, la siguiente sección muestra el plan de pruebas que se diseñó para este caso de uso.

4.5.1 Plan de Pruebas de Unidad

El objetivo del plan de pruebas de unidad es planear las pruebas ejecutarlas y evaluarlas, este plan se realizó por clase, probando cada uno de sus métodos para finalmente probar de manera general la clase completa.

Pruebas estructurales o de caja blanca

Durante la codificación de las clases se realizaron pruebas estructurales o de caja blanca para verificar el código escrito revisando la sintaxis y algoritmos, la siguiente tabla muestra los objetos que se probaron y las acciones que se realizaron:

Objeto de prueba	Acción
Sintaxis	Se revisó la sintaxis y cuando se encontraron errores se corrigieron.
Ciclos	Se revisó que el inicio y fin fueran correctos y que se realizaran dentro de los rangos establecidos.
Flujo de datos	Se verificaron las trayectorias que deben seguir los datos.
Bucles y recursiones	Se revisaron que los bucles y las recursiones se terminaran de manera correcta.
Manejo de errores y de excepciones	Se realizaron las verificaciones necesarias de los casos es los que se producían errores y excepciones y que se lanzaran de manera correcta.
Estructuras de datos	Se revisó que las estructuras de datos estuvieran implementadas de manera correcta,
Acceso y conexión a la base de datos	Se verificaron las conexiones a la base de datos, para que se accediera y cerrara la conexión de manera correcta

Tabla 4.11 Tabla de objetos de prueba.

Pruebas de especificación o de caja negra

Caso de prueba	Resultado esperado	Resultado obtenido
Clase: DAOProyecto		
Método: registrarProyecto(Proyecto proyecto)		
registrarProyecto(proyecto)	Proyecto registrado en la base de datos	Proyecto registrado en la base de datos
Método: modificarProyecto(Proyecto proyecto)		
modificarProyecto(proyecto)	Proyecto modificado correctamente	Proyecto modificado correctamente
Método: consultarProyecto(int idProyecto)		
consultarProyecto(idProyecto)	El proyecto consultado	El proyecto consultado
Método: getConexion()		
getConexion()	Se realizó la conexión a la base de datos	Se realizó la conexión a la base de datos

Tabla 4.12 Tabla de casos de prueba.

Conclusiones

El desarrollo de este trabajo de tesis cumplió con el objetivo de crear una herramienta para la administración de proyectos basada en MoProSoft, y que formara parte del software libre.

La experiencia de seguir el Proceso Unificado para el desarrollo de software fue muy satisfactoria porque permitió retomar el trabajo en todo momento ya que siempre se contó con un proceso documentado y ordenado.

La utilización del marco de trabajo Struts, fue una buena elección, ya que permite desarrollar aplicaciones Web con una arquitectura muy bien definida y estructurada, el desarrollo del sistema fue muy ágil una vez que se superó la curva de aprendizaje. La implementación del patrón MVC, a través de Struts, permite un excelente mantenimiento de la aplicación ya que el modelo, la vista y el controlador se encuentran claramente separados, lo que permite que la realización de cambios, agregar nuevas funcionalidades en cualquiera de los tres niveles se realice casi insospechadamente para el resto de la aplicación. Otra de las ventajas de haber trabajado con Struts es facilidad de integración, es decir, en cada iteración que se implementaba un nuevo caso de uso, se realizó muy fácilmente.

Los primeros pasos implementando Struts se convirtieron en un reto debido a la curva de aprendizaje adicional que surge cuando no se tiene experiencia implementando marcos de trabajo.

La principal ventaja de HeAP MoProSoft es contar con una herramienta diseñada especialmente para apoyar en las actividades de administración de proyectos siguiendo MoProSoft. Otras ventajas son las que nos ofrece el que sea una aplicación Web, como facilidad de acceso pues únicamente es necesario contar con una conexión a Internet y con cualquier navegador Web. HeAP MoProSoft es una herramienta que forma parte del software de distribución libre.

Entre las desventajas de esta herramienta, por ser una aplicación Web, se encuentran la velocidad de acceso y de respuesta ya que esta depende de la velocidad de la conexión a Internet, es decir del ancho de banda con el que se cuente.

Entre los trabajos futuros se encuentra agregar la Categoría de Alta Dirección y el Proceso de Gestión de Negocio, la Categoría de Gerencia con el Proceso de Gestión de Procesos y Gestión de Recursos y finalmente la Categoría de Operación el Proceso de Desarrollo y Mantenimiento de Software. Y robustecer los procesos implementados en esta tesis.

La integración con dos trabajos de tesis titulados: *"Sistema de tableros de control para gestión de proyectos para MoProSoft"* [13] y *"Sistema Multi-Agente de apoyo para el proceso de Administración de Proyectos Específicos de MoProSoft"* [14]. Al incluir el sistema Multi-Agente se tendrá una guía y apoyos para realizar las actividades del proceso. Los tableros de control permitirán visualizar el desempeño de los proyectos de la empresa.

Al término de estas tres tesis, incluyendo ésta, se contará con una herramienta diseñada especialmente para facilitar la adopción de MoProSoft, reduciendo el tiempo de aprendizaje de las empresas para la implementación de los procesos de *Gestión de Proyectos* y *Administración del Proyecto Específico*, proporcionando plantillas y herramientas necesarias, para generar y administrar la documentación correspondiente a cada proceso.

Referencias y bibliografía

[1] Documento MoProSoft por Niveles de Capacidad de Procesos Versión 1.3 Agosto 2005 http://www.software.net.mx

[2] PMBOK 2004 http://www.pmi.org/

[3] Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides, ***Design Patterns***, Addison-Wesley, 2000.

[4] http://struts.apache.org/

[5] http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

[6] Eales Peter, Houston Kelli, Kozaczynski Wojtek, ***Building J2EE Applications with the Rational Unified Process***, Addison-Wesley, 2004.

[7] Cavaness Chuck, ***Jakarta Struts***, Anaya O'Reilly, 2005.

[8] Jacobson Ivar, Booch Grady, Rumbaugh James, ***El Proceso Unificado de Desarrollo de Software***, Addison Wesley, Madrid, España 2006.

[9] Jacobson Ivar, Booch Grady, Rumbaugh James, ***El Lenguaje Unificado de Modelado***, 2ª edición, Addison Wesley, Madrid, España, 2006.

[10] http://www.postgresql.org

[11] http://www.netbeans.org

[12] Weitzenfeld Alfredo, ***Ingeniería de Software Orientada a Objetos con UML***, Java e Internet, Thomson, 2005.

[13] Cruz S. Rafael, ***"Sistema de tableros de control para gestión de proyectos para MoProSoft"***, Tesis de maestría en Ingeniería de la Computación, UNAM 2008.

[14] Lozada R. Aurora, ***"Sistema Multi-Agente de apoyo para el proceso de Administración de Proyectos Específicos de MoProSoft"***, Tesis de maestría en Ingeniería de la Computación, UNAM 2007.

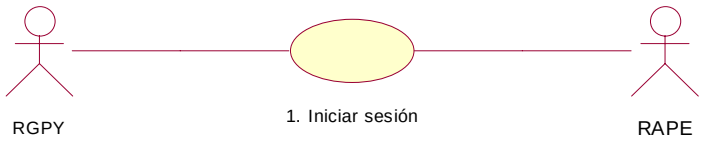
[15] Stelting Stephen, Maassen Olav, ***Patrones de Diseño Aplicados a JAVA***, Sun Microsystems, Prentice Hall, España 2003.

Apéndice A

En el presente apéndice se presenta el detalle de los cuatro casos de uso restantes:
1. Iniciar Sesión
3. Administrar Personal
4. Administrar Proyecto Específico
5. Cerrar Sesión

Detalle del caso de uso
1. Iniciar Sesión

Diagrama del caso de uso



Definición

Caso de uso:	1. Iniciar Sesión
Actor:	*RGPY y RAPE* Nota: Nos referiremos a ellos con el nombre genérico de *Usuario*.
Descripción:	El caso de uso Iniciar sesión nos muestra como el *Usuario* ingresa al sistema autenticándose mediante nombre de usuario y contraseña.
Precondiciones:	1. El *Usuario* debe tener una conexión a Internet y haber cargado la página del sistema en su navegador 2. El *Usuario* debe estar registrado en el sistema y tener un nombre de usuario y contraseña válidos
Poscondiciones:	1. El Usuario ingresó al sistema con un rol determinado

Flujo de eventos

Flujo básico:

Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	Este caso de uso se inicia cuando el *Usuario* introduce su nombre de usuario y contraseña	2	El sistema valida el nombre de usuario y contraseña [AE01]
		3	El sistema muestra la lista de roles permitidos para el *Usuario*
4	El *Usuario* elige uno de los roles de la lista con el cual desea ingresar	5	Este caso de uso termina cuando el sistema permite el ingreso al *Usuario* con el rol que eligió [AG01] [AG02]

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *Usuario* sale del sistema al elegir la opción “Salir”
AG02	Cambiar rol	El sistema lo muestra la lista de roles permitidos para el *Usuario*, para ingresar al sistema
AE01	Datos incorrectos	El sistema manda un mensaje de error indicándole al *Usuario* que el nombre de usuario o la contraseña son incorrectos

Tabla A.1 Detalle del caso de uso 1. Iniciar Sesión

Pantalla “Iniciar Sesión”

Esta pantalla autentica a los usuarios que quieran ingresar al sistema. Esta pantalla además garantiza la seguridad restringiendo la entrada total a cualquier persona no autorizada ya que no muestra ningún componente del sistema.

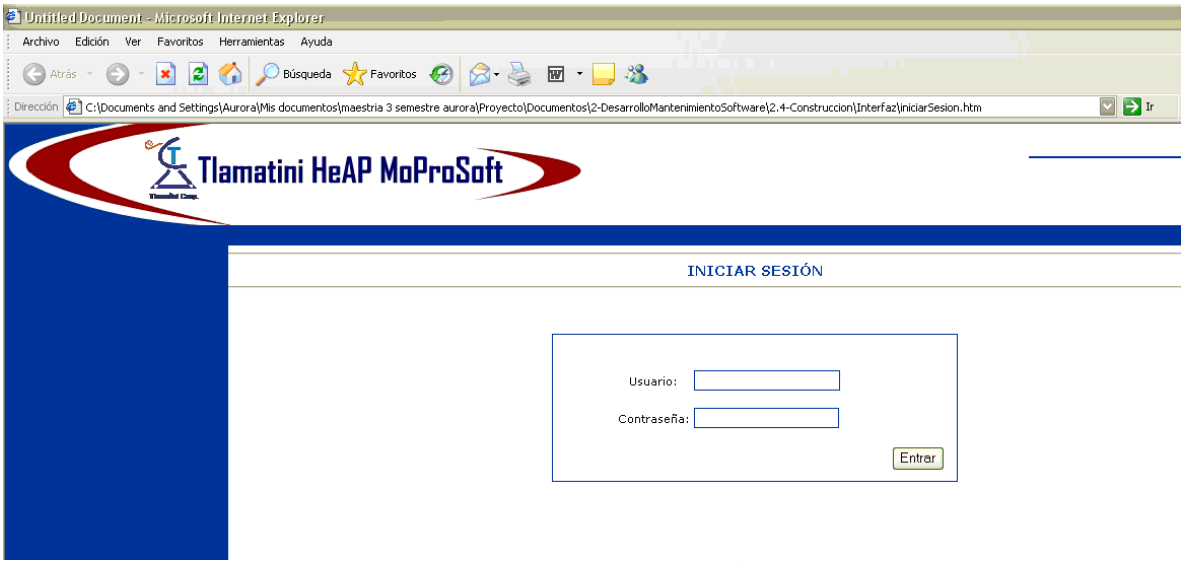


Figura A.1 Pantalla “Iniciar Sesión”

Pantalla “Seleccionar Rol”

Esta pantalla muestra los posibles roles con los cuales puede el usuario entrar al sistema.

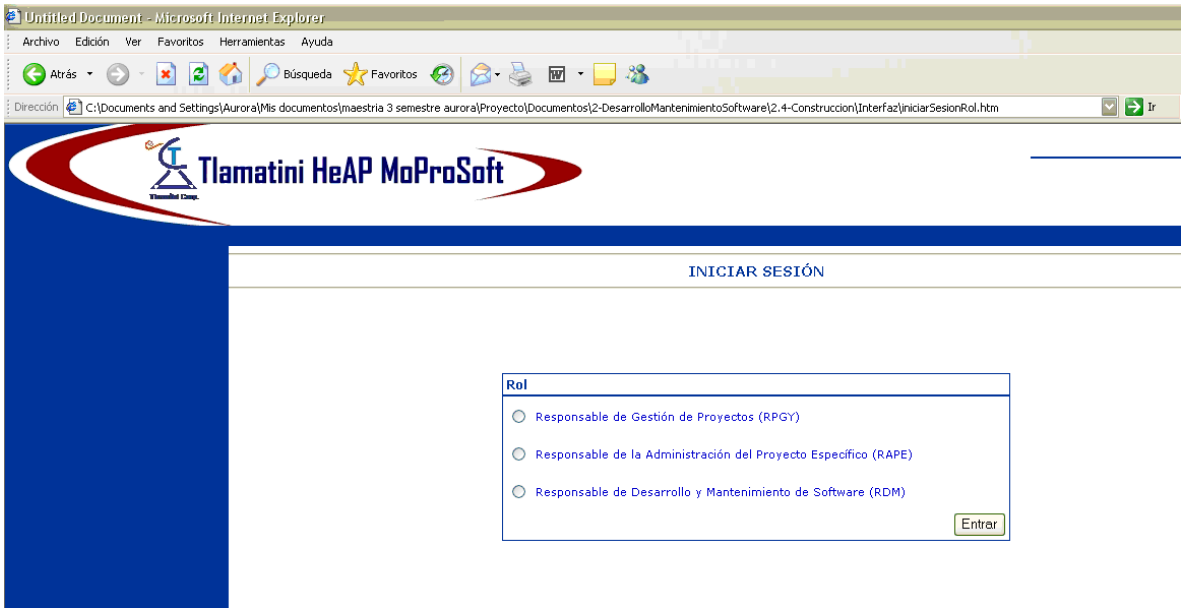


Figura A.2 Pantalla “Seleccionar Rol”

Diagramas de clases del análisis del caso de uso 1. Iniciar Sesión

Diagrama de clases de interfaz

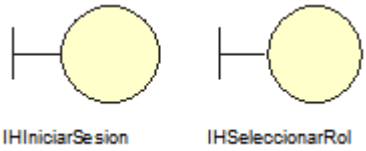
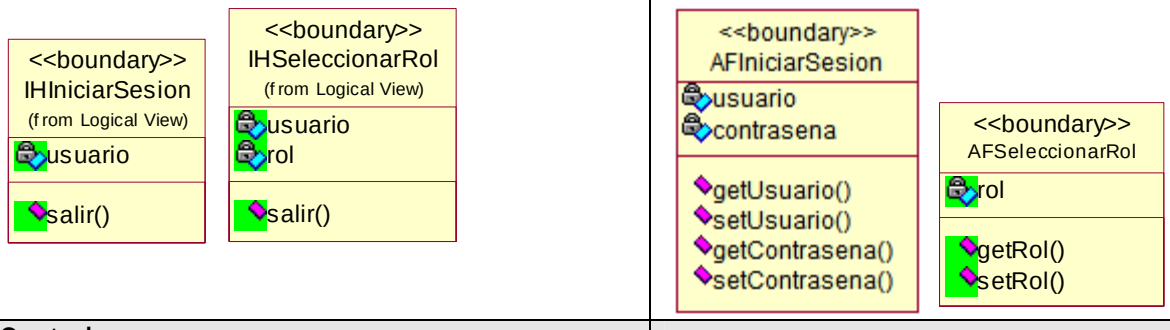
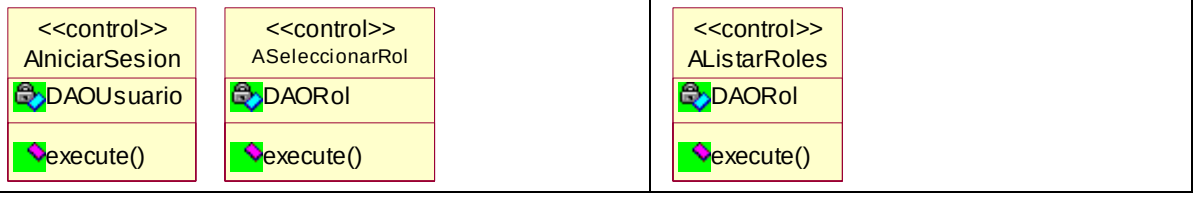



Diagrama de clases de entidad

Diagrama de clases de control


Figura A.3 Diagrama de clases del análisis del caso de uso 1. Iniciar Sesión

Diagramas de clases del diseño del caso de uso 1. Iniciar Sesión

Iniciar Sesión	
Interfaz	
	
Control	
	

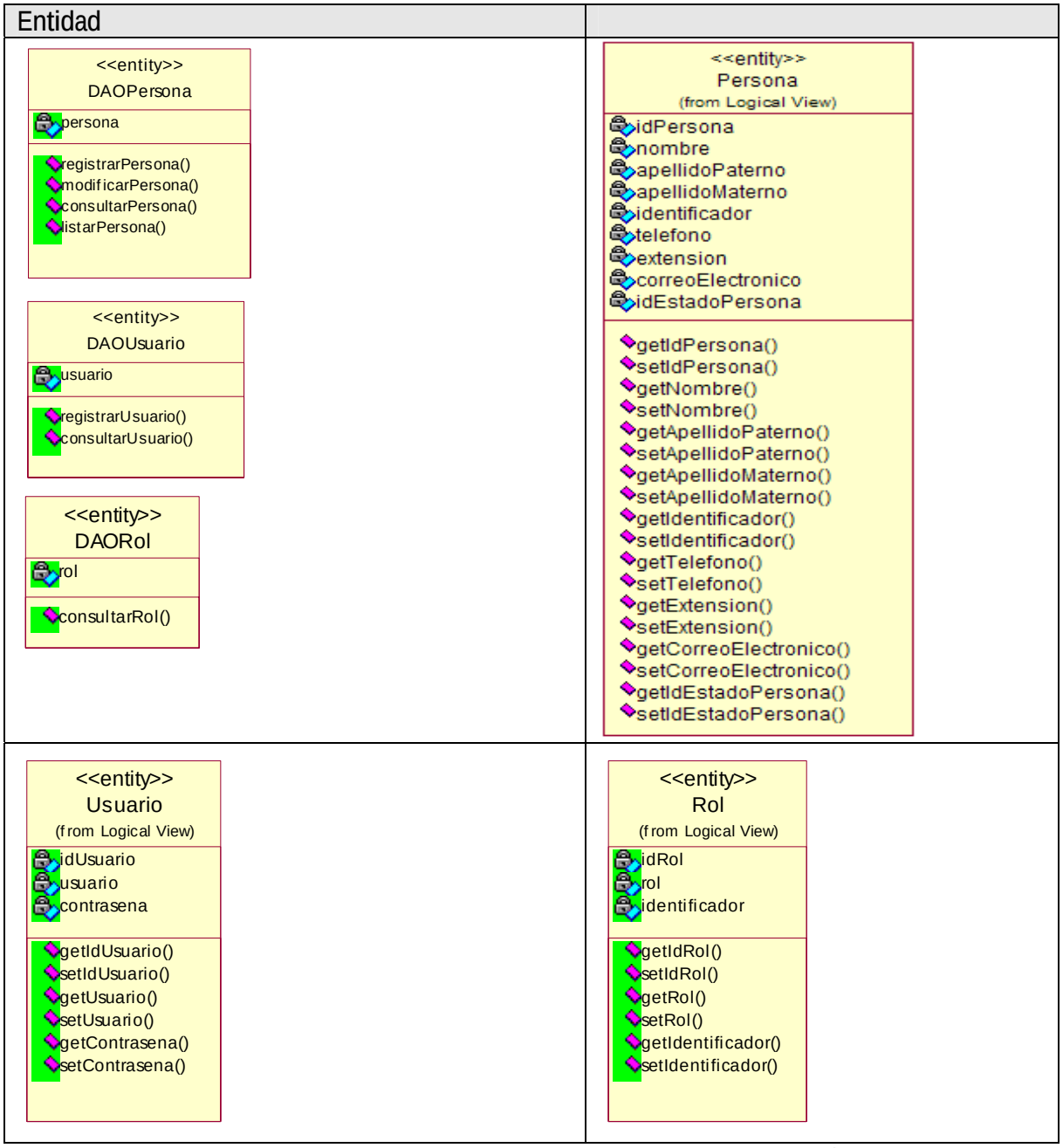
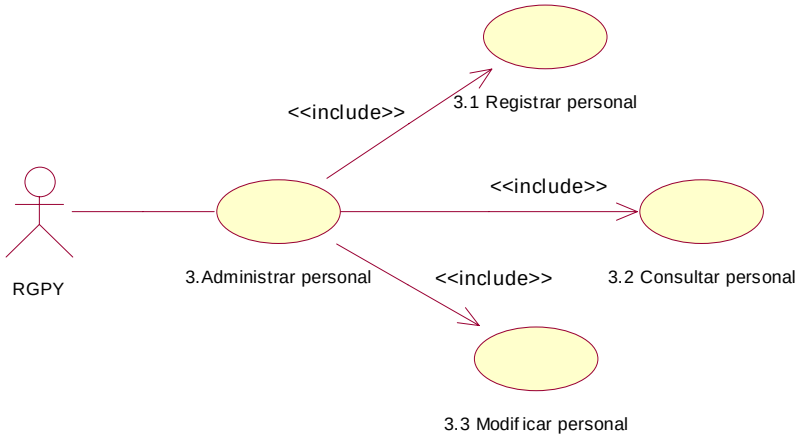


Figura A.4 Diagrama de clases del diseño del caso de uso 1. Iniciar Sesión

Detalle del caso de uso
3. Administrar Personal

Diagrama del caso de uso



Definición

Caso de uso:	3. Administrar personal
Tipo:	Básico
Actor:	*RGPY*
Descripción:	En el caso de uso Administrar personal el RGPY realiza las tareas relacionadas con la consulta, registro y modificación del personal.
Precondiciones:	El *RGPY* ejecutó el caso de uso 1. Iniciar sesión y seleccionó la opción "Administración de personal".
Poscondiciones:	El *RGPY* realizó alguna de las tareas tales como: Consultar, Registrar o Modificar personal.

Flujo de eventos

Flujo básico:

Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	Este caso de uso se inicia cuando el *RGPY* selecciona la opción "Administración de personal" del menú principal del sistema.	2	El sistema le muestra las interfaces que correspondan a la administración de personal.
		3	El sistema presenta la lista de personas registradas en el sistema ordenadas alfabéticamente [AE01].
4	El *RGPY* selecciona alguna de las tareas como: Consultar, Registrar o Modificar personal.	5	El sistema da respuesta a la petición del usuario presentándole las interfaces de la tarea que seleccionó. [AG01] [AG02]

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RGPY* sale del sistema al elegir la opción "Salir"
AG02	Cambiar rol	El sistema lo muestra la lista de roles permitidos para el *Usuario*, para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RGPY* indicándole que ha sucedido un error de conexión con la base de datos.

Tabla A.2 Detalle del caso de uso 3. Administrar Personal.

Pantalla “Administrar Personal”

Esta pantalla muestra el listado de las personas registradas en el sistema, así como las opciones de Registrar, Eliminar y Modificar personas.

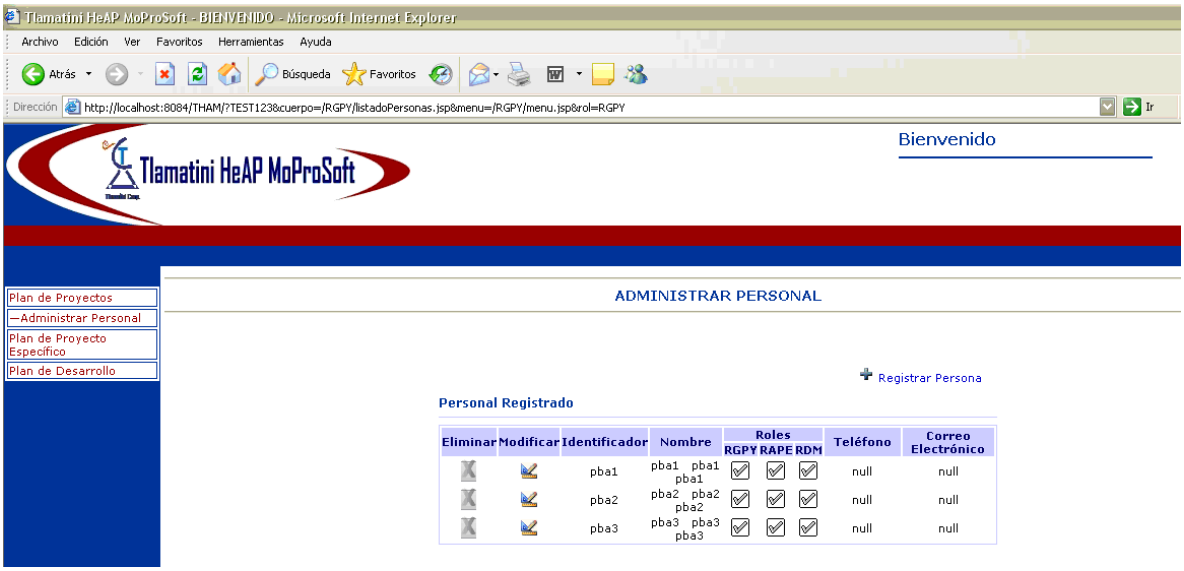
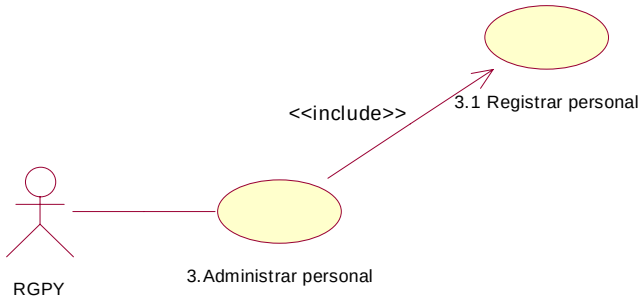


Figura A.5 Pantalla “Administrar Personal”

Caso de uso: 3.1 Registrar Personal

Diagrama del caso de uso


Definición	
Caso de uso:	3.1 Registrar personal
Tipo:	Inclusión
Actor:	*RGPY*
Descripción:	En el caso de uso Registrar personal el *RGPY* podrá realizar el registro del personal.
Precondiciones:	El *RGPY* seleccionó la opción “Registrar persona”.
Poscondiciones:	El *RGPY* realizó el registro de una nueva persona dentro del sistema.

Flujo de eventos			
Flujo básico:			
Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Registrar persona comienza cuando el *RGPY* selecciona la opción "Registrar persona".	2	El sistema le muestra la interfaz de registro de persona.
3	El *RGPY* introduce los datos personales de la persona; si es el caso le asigna un rol, nombre de usuario y contraseña.	4	El sistema valida los datos y si son correctos se almacenan en la base de datos; el sistema regresará al listado de las personas con la nueva persona incluida [AG01] [AG02] [AE01] [AE02].

Flujos alternos:		
Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RGPY* sale del sistema al elegir la opción "Salir"
AG02	Cambiar rol	El sistema lo muestra la lista de roles permitidos para el *Usuario*, para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RGPY* indicándole que ha sucedido un error de conexión con la base de datos.
AE02	Datos inválidos	El sistema manda un mensaje de error indicando que los datos son inválidos.

Tabla A.3 Detalle del caso de uso 3.1 Registrar Personal.

Pantalla "Registrar Persona"

Esta pantalla muestra los datos que deben ser capturados para poder registrar usuarios y personas en el sistema.

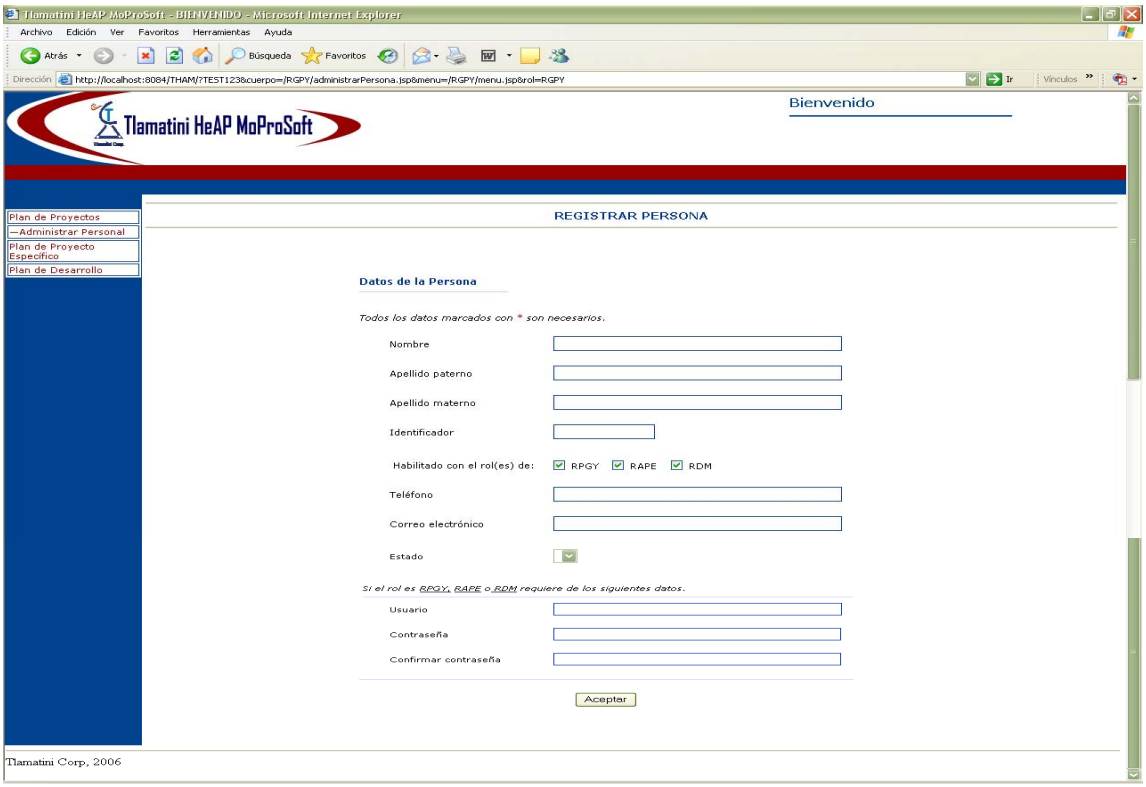


Figura A.6 Pantalla "Registrar Persona"

Caso de uso: 3.2 Consultar Personal

Diagrama del caso de uso


Definición	
Caso de uso:	3.2 Consultar personal
Tipo:	Inclusión
Actor:	*RGPY*
Descripción:	En el caso de uso Consultar personal el *RGPY* podrá consultar los datos de las personas que se encuentran registradas en el sistema.
Precondiciones:	El *RGPY* seleccionó la opción "consultar" del listado de personas, eligiendo a la persona que desea consultar.
Poscondiciones:	El *RGPY* realizó la consulta de los datos de la persona que seleccionó.

Flujo básico:

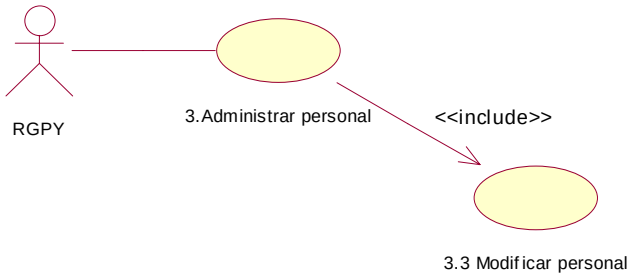
Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Consultar persona comienza cuando el *RGPY* selecciona la opción "consultar" del listado de personas eligiendo a la persona que desea consultar.	2	El sistema le muestra la interfaz que contiene los datos de la persona que seleccionó. [AE01]
3	El *RGPY* elige la opción "regresar al listado" y seguir con la consulta.	4	El sistema regresa a la interfaz del listado de personas registradas [AG01] [AG02].

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RGPY* sale del sistema al elegir la opción "Salir"
AG02	Cambiar rol	El sistema lo muestra la lista de roles permitidos para el *Usuario*, para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RGPY* indicándole que ha sucedido un error de conexión con la base de datos.

Tabla A.4 Detalle del caso de uso 3.2 Consultar Personal.

Caso de uso: 3.3 Modificar Personal

Diagrama del caso de uso


Definición	
Caso de uso:	3.3 Modificar personal
Tipo:	Inclusión
Actor:	*RGPY*
Descripción:	En el caso de uso Modificar personal el *RGPY* podrá modificar los datos de las personas que se encuentran registradas en el sistema.
Precondiciones:	El *RGPY* seleccionó la opción "modificar" del listado de personas, eligiendo a la persona que desea modificar.
Poscondiciones:	El *RGPY* realizó la modificación de los datos de la persona que seleccionó.

Flujo de eventos			
Flujo básico:			
Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Modificar persona comienza cuando el *RGPY* selecciona la opción "modificar" del listado de personas eligiendo a la persona que desea modificar.	2	El sistema le muestra la interfaz que contiene los datos de la persona que seleccionó listos para se modificados. [AE01]
3	El *RGPY* introduce los datos que quiere modificar.	4	El sistema valida los datos y si son correctos modifica el registro en la base de datos [AE02].
		5	El sistema regresa a la interfaz del listado de personas registradas [AG01] [AG02].

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RGPY* sale del sistema al elegir la opción "Salir"
AG02	Cambiar rol	El sistema lo muestra la lista de roles permitidos para el *Usuario*, para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RGPY* indicándole que ha sucedido un error de conexión con la base de datos.
AE02	Datos inválidos	El sistema manda un mensaje de error indicando que los datos son inválidos.

Tabla A.5 Detalle del caso de uso 3.3 Modificar Personal.

Pantalla “Consultar, Modificar Persona”

Esta pantalla muestra los datos que pueden ser consultados o modificados de una persona registrada en el sistema.

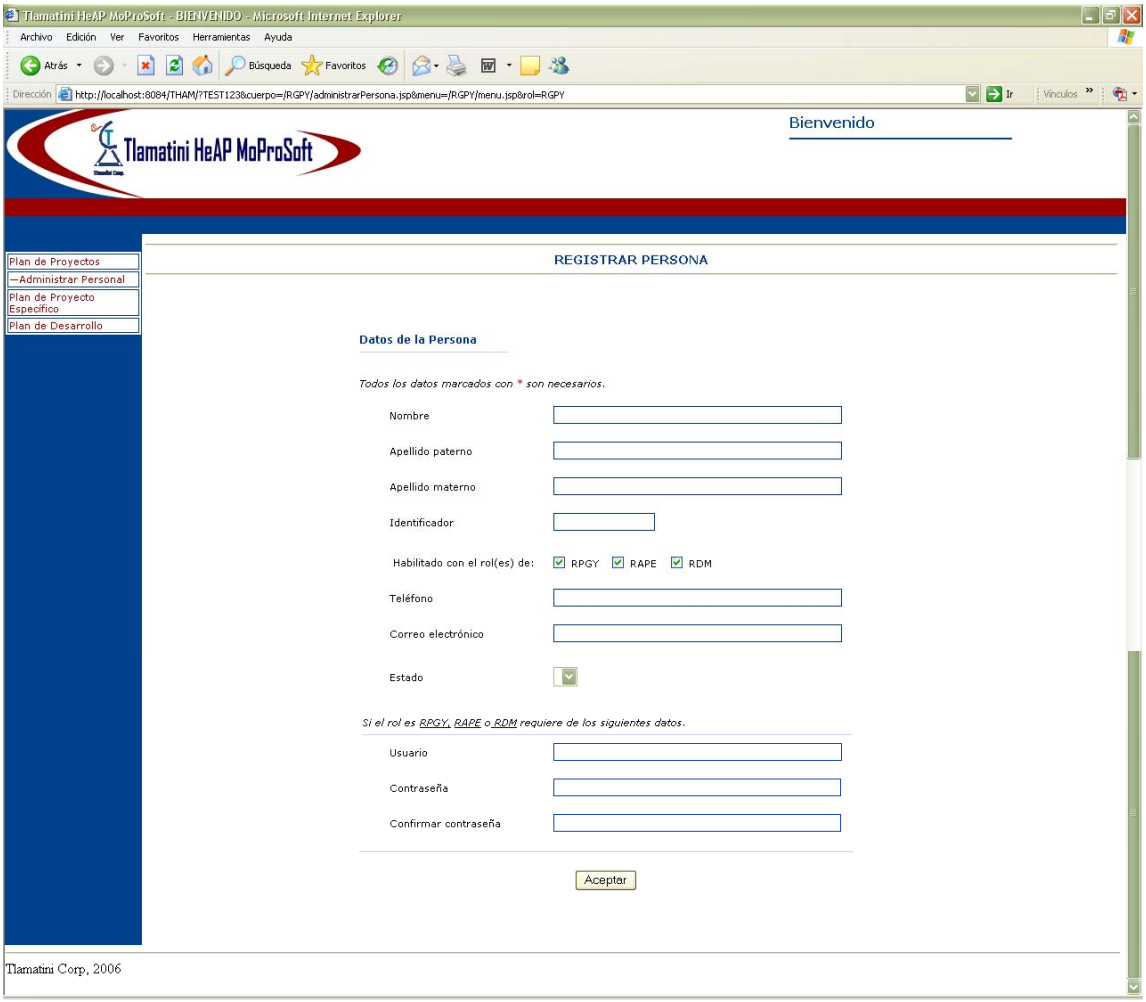


Figura A.7 Pantalla “”Modificar Persona”

Diagramas de clases del análisis
3. Administrar Personal

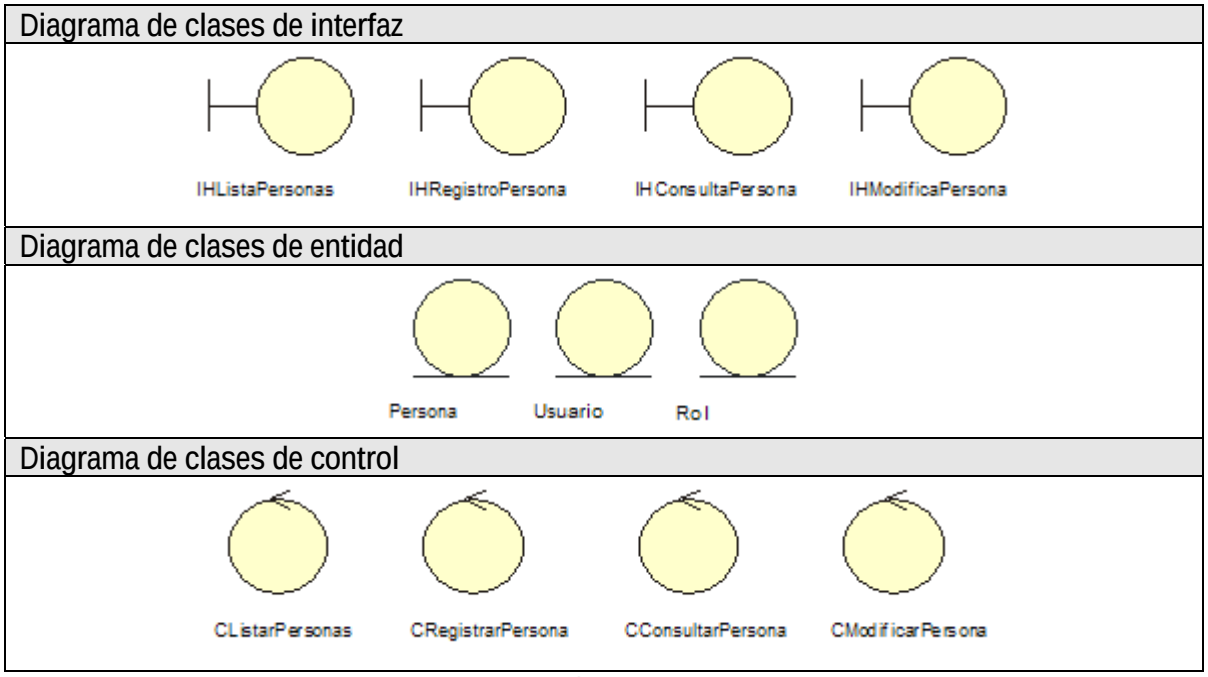
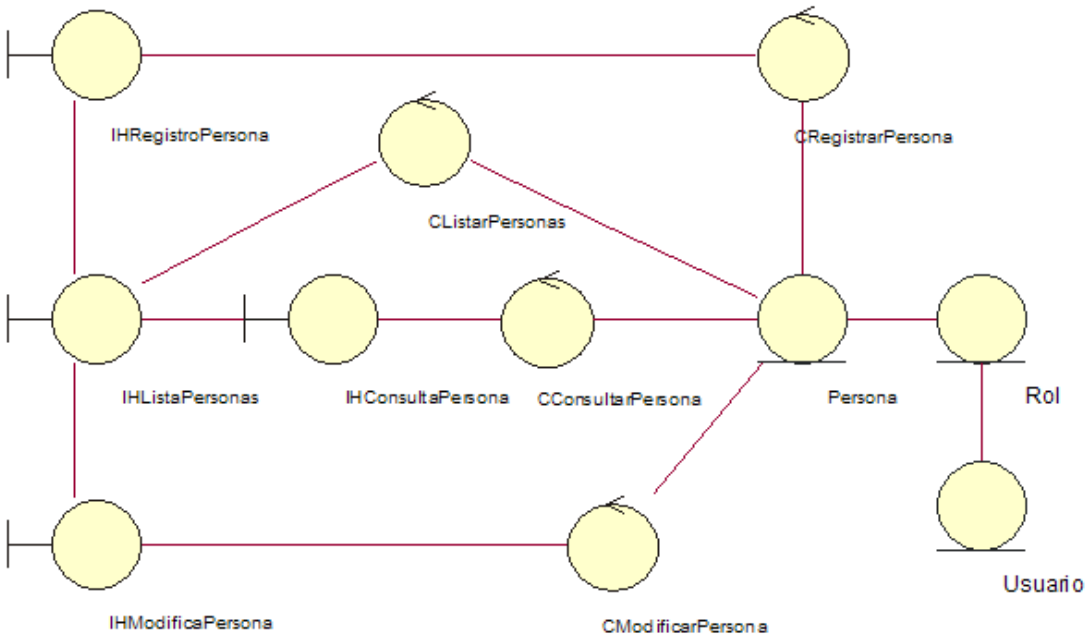


Figura A.8 Diagrama de clases del análisis del caso de uso 3. Administrar Personal

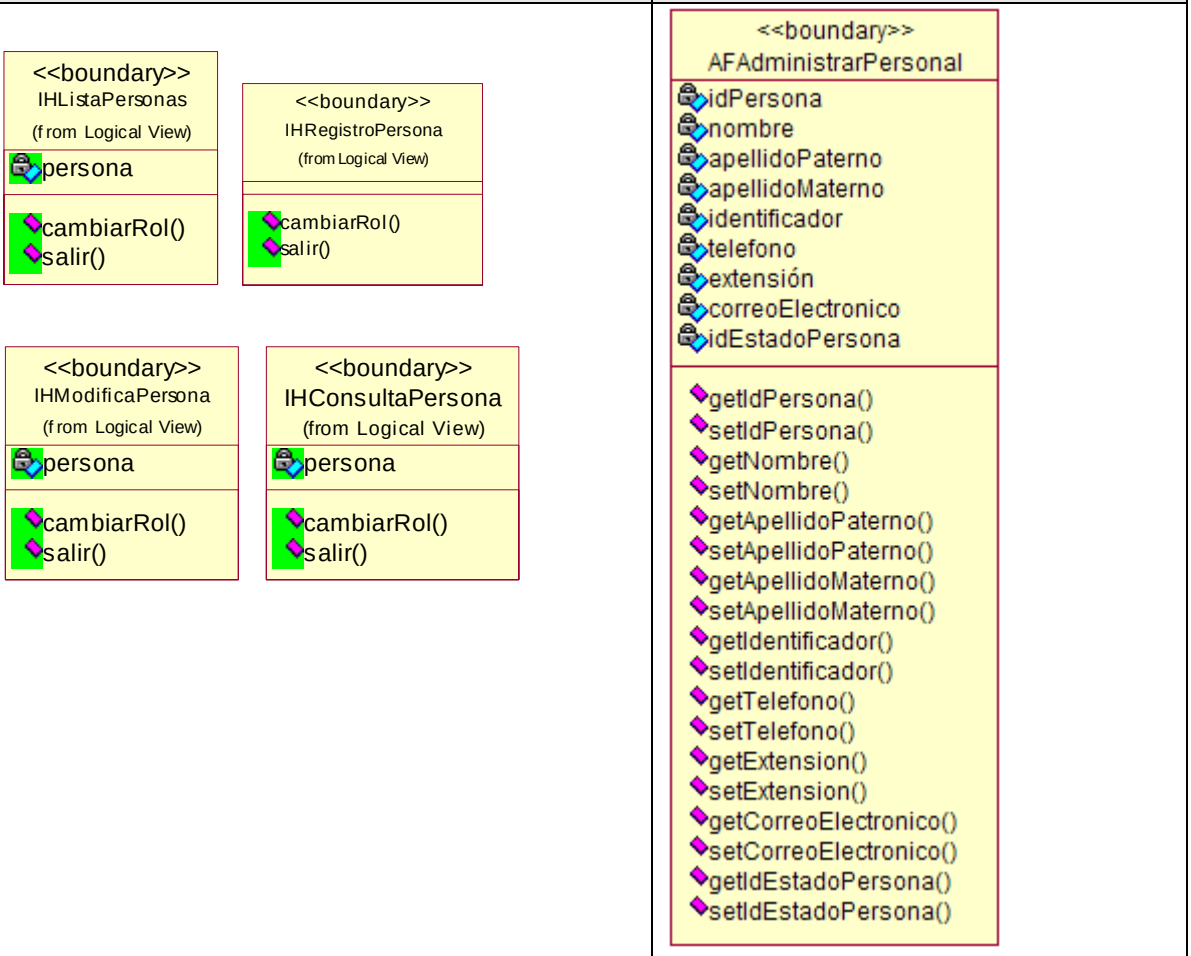
Diagrama de clases del análisis



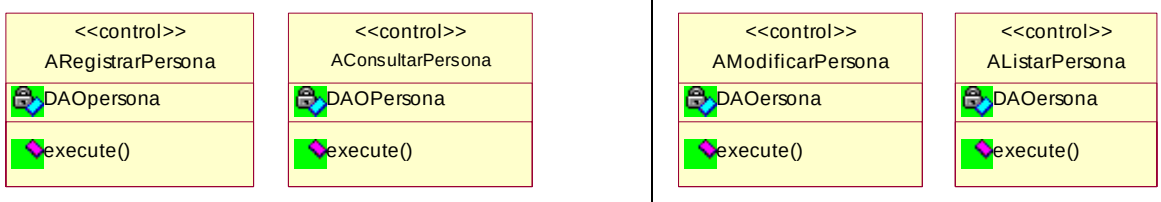
Diagramas de clases del diseño
3. Administrar Personal

Administrar Personal

Interfaz



Control



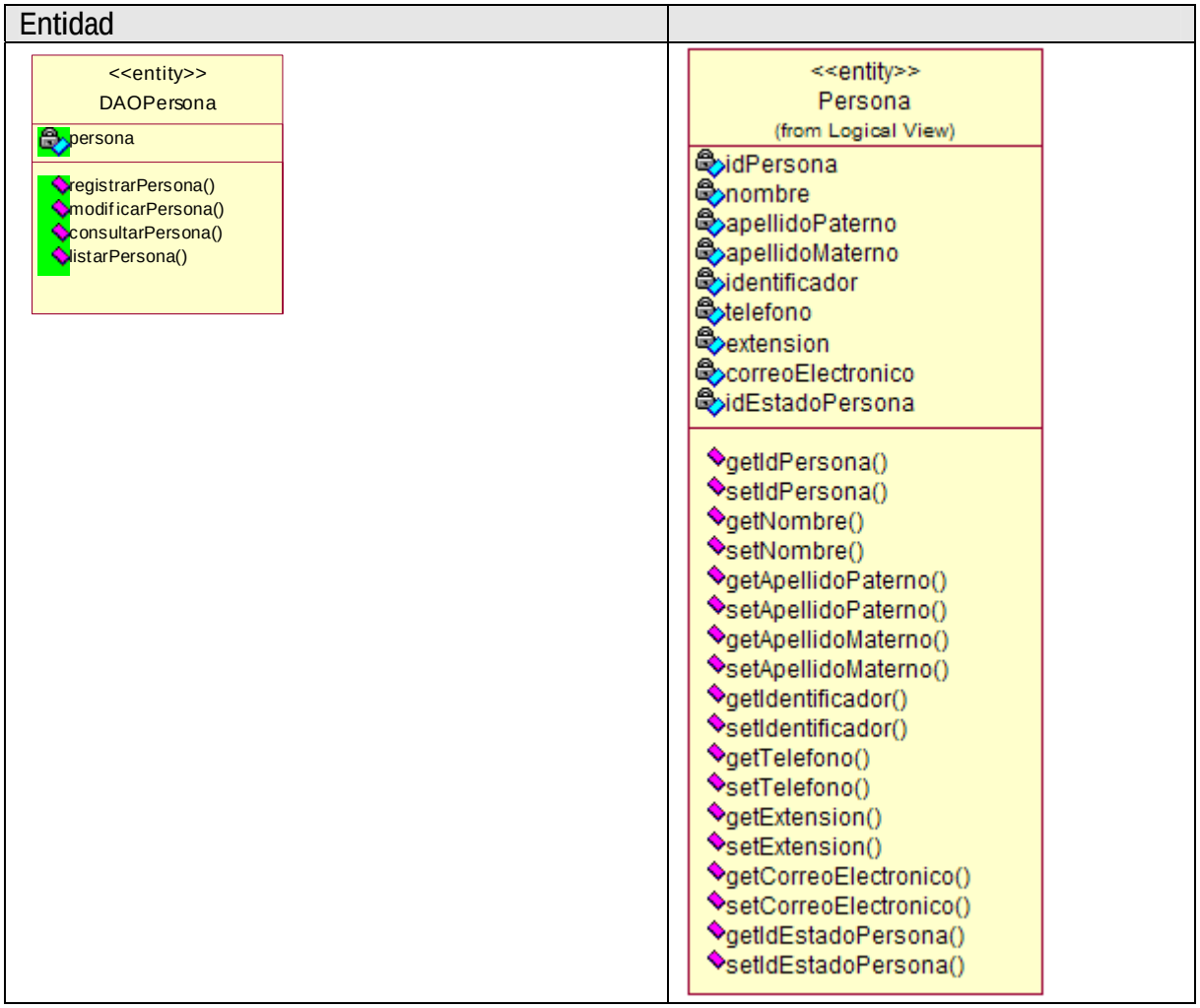
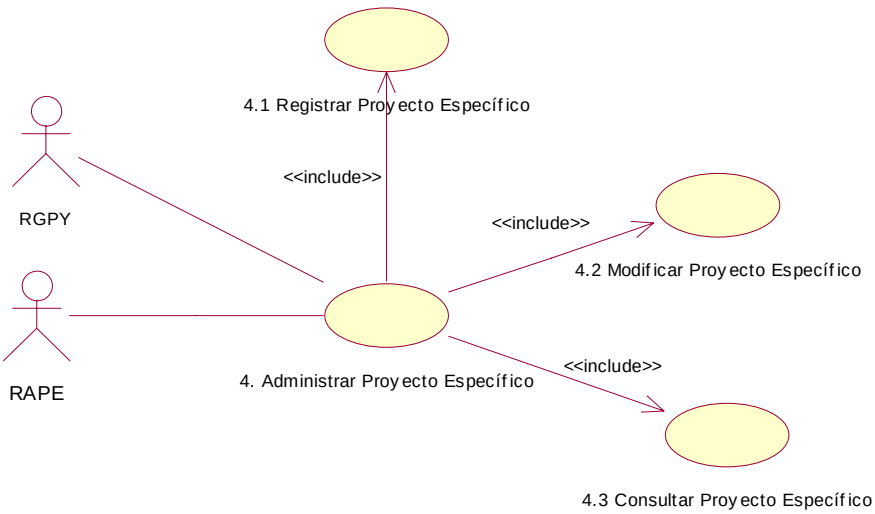


Figura A.9 Diagrama de clases del diseño del caso de uso 3. Administrar Personal

Detalle del caso de uso
4. Administrar Proyecto Específico

Diagrama del caso de uso


Definición	
Caso de uso:	2. Administrar Proyecto Específico
Tipo:	Básico
Actor:	*RGPY y RAPE*
Descripción:	En este caso de uso el *RAPE* administra el plan del proyecto específico, y el *RGPY* únicamente puede consultarlo, el cual está integrado por el plan de actividades, el equipo de trabajo, el costo estimado, el protocolo de entrega, las adquisiciones, la capacitación y los riesgos, se realiza el registro, modificación y consulta del plan del proyecto específico, para el registro se genera el Plan de Proyecto Específico registrando cada uno de los siete planes listados anteriormente, durante la modificación se podrá actualizar la información proporcionada durante el registro, la consulta consiste en consultar los datos del Plan del Proyecto Específico y los siete planes que lo integran
Precondiciones:	Se debió haber ejecutado el caso de uso 1.Iniciar sesión como *RAPE* o *RGPY.*
Poscondiciones:	El *RGPY* realizó alguna de las siguientes actividades: el registro, la consulta o la modificación del Plan del Proyecto Específico.

Flujo de eventos			
Flujo básico:			
Actor		Sistema	
Paso	Acciones	Paso	Acciones
		1	El sistema le muestra la interfaz principal de administración de proyectos específicos en dónde se listan los proyectos registrados en el sistema ordenados alfabéticamente [AE01].
2	El *RAPE* selecciona de la lista el proyecto al que desea ingresar para administrar el Plan del Proyecto Específico.	3	El sistema despliega la pantalla del Plan Específico del Proyecto seleccionado por el *RAPE*, mostrándole las opciones: Actividades,

			Adquisiciones, Capacitación, Costo Estimado, Equipo de Trabajo, Protocolo de Entrega, Riesgos, para administrar el Plan del Proyecto Específico, si el usuario que ingresó es el RGPY, el sistema únicamente le muestra las opciones de consulta.
4	El *RAPE* selecciona alguna de las opciones que le presenta el sistema, dependiendo del plan que desee administrar, el *RGPY* únicamente podrá consultar.	5	El sistema da respuesta a la opción seleccionada por el *RAPE* o *RGPY* mostrándole las interfaces correspondientes a esa opción [AE01] [AG01] [AG02].

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RAPE* o *RGPY* sale del sistema al elegir la opción "Salir"
AG02	Cambiar rol	El sistema lo muestra la lista de roles permitidos para el *Usuario*, para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RAPE* o *RGPY* indicándole que ha sucedido un error de conexión con la base de datos.

Tabla A.6 Detalle del caso de uso 4. Administrar Proyecto Específico

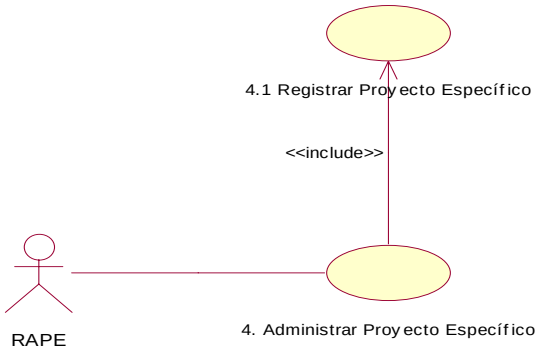
El diseño de la pantalla en donde se listan los proyectos registrados para el *RAPE*, se muestra en la figura A.1.

Pantalla "Proyectos"

Esta pantalla de inicio del Plan del Proyecto Específico, muestra un listado de proyectos con datos básicos como identificador y nombre. En esta pantalla da acceso para consultar el Plan del Proyecto Específico de cada proyecto registrado.

Figura A.10 Pantalla "Proyectos Específicos"

Caso de uso: 4.1 Registrar Proyecto Específico

Diagrama del caso de uso


Definición	
Caso de uso:	4.1 Registrar Proyecto Específico
Tipo:	Inclusión
Actor:	*RAPE*
Descripción:	En el caso de uso Registrar proyecto el *RAPE* se realiza el registro del Plan del Proyecto Específico el cual se encuentra integrado por los siguientes planes: Actividades, Adquisiciones, Capacitación, Costo Estimado, Equipo de Trabajo, Protocolo de Entrega, Riesgos.
Precondiciones:	El *RAPE* seleccionó el proyecto de la lista de sus proyectos registrados.
Poscondiciones:	El *RAPE* realizó el registro de un Plan del Proyecto Específico en el sistema.

Flujo básico:

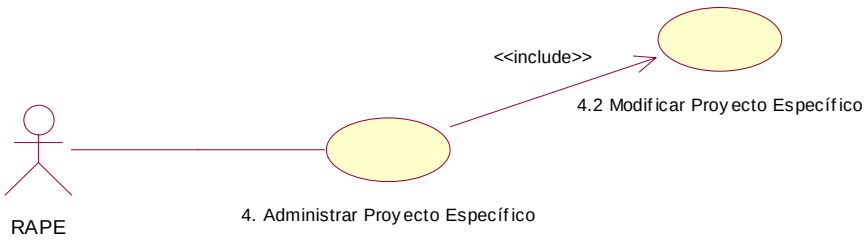
Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Registrar Proyecto Específico comienza cuando el *RAPE* selecciona alguna de las opciones del menú, para registrar alguno de los siguientes planes Actividades, Adquisiciones, Capacitación, Costo Estimado, Equipo de Trabajo, Protocolo de Entrega, Riesgos.	2	El sistema le muestra la interfaz de registro del plan seleccionado por el *RAPE*.
3	El *RAPE* introduce los datos para realizar el registro del plan y elige la opción “Guardar”.	4	El sistema valida los datos y si son correctos se almacenan en la base de datos [AG01] [AG02] [AE01] [AE02].

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RAPE* sale del sistema al elegir la opción “Salir”
AG02	Cambiar rol	El sistema lo muestra la lista de roles permitidos para el *Usuario*, para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RAPE* indicándole que ha sucedido un error de conexión con la base de datos.
AE02	Datos inválidos	El sistema manda un mensaje de error indicando que los datos son inválidos.

Tabla A.7 Detalle del caso de uso 4.1 Registrar Proyecto Específico

Caso de uso: 4.2 Modificar Proyecto Específico

Diagrama del caso de uso


Definición	
Caso de uso:	4.2 Modificar Proyecto Específico
Tipo:	Inclusión
Actor:	*RAPE*
Descripción:	En el caso de uso Modificar Proyecto Específico el *RAPE* podrá modificar los datos del Plan del Proyecto Específico, el cual se encuentra integrado por los siguientes planes: Actividades, Adquisiciones, Capacitación, Costo Estimado, Equipo de Trabajo, Protocolo de Entrega, Riesgos.
Precondiciones:	El *RAPE* seleccionó el proyecto de la lista de sus proyectos registrados.
Poscondiciones:	El *RAPE* realizó la modificación del Plan del Proyecto Específico del proyecto que eligió.

Flujo de eventos

Flujo básico:

Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Modificar Proyecto Específico comienza cuando el *RAPE* selecciona alguna de las opciones del menú, para modificar alguno de los siguientes planes Actividades, Adquisiciones, Capacitación, Costo Estimado, Equipo de Trabajo, Protocolo de Entrega, Riesgos y selecciona la opción "Modificar".	2	El sistema le muestra la interfaz que contiene los datos del plan que seleccionó, listos para ser modificados. [AE01]
3	El *RAPE* introduce los datos que quiere modificar.	4	El sistema valida los datos y si son correctos modifica el registro en la base de datos [AE02].
		5	El sistema regresa a la interfaz de consulta del plan con los datos modificados [AG01] [AG02].

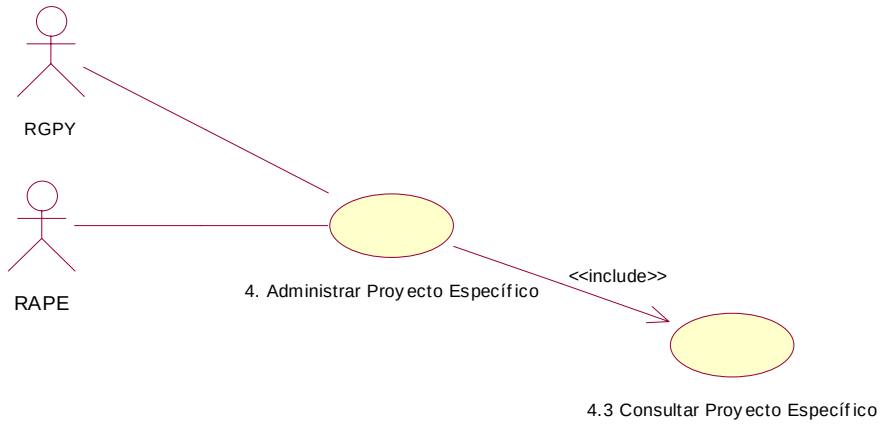
Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RAPE* sale del sistema al elegir la opción "Salir"
AG02	Cambiar rol	El sistema lo muestra la lista de roles permitidos para el *Usuario*, para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RAPE* indicándole que ha sucedido un error de conexión con la base de datos.
AE02	Datos inválidos	El sistema manda un mensaje de error indicando que los datos son inválidos.

Tabla A.8 Detalle del caso de uso 4.2 Modificar Proyecto Específico

Caso de uso: 4.3 Consultar Proyecto Específico

Diagrama del caso de uso



Definición	
Caso de uso:	4.3 Consultar Proyecto Específico
Tipo:	Inclusión
Actor:	*RGPY, RAPE*
Descripción:	En el caso de uso Consultar Proyecto Específico el *RGPY* o el *RAPE* podrán consultar el Plan del Proyecto Específico, el cual se encuentra integrado por los siguientes planes: Actividades, Adquisiciones, Capacitación, Costo Estimado, Equipo de Trabajo, Protocolo de Entrega, Riesgo, que se encuentra registrado en el sistema.
Precondiciones:	El *RGPY* o el *RAPE* seleccionaron el proyecto de la lista de proyectos.
Poscondiciones:	El *RGPY* o el *RAPE* realizaron la consulta del Plan del Proyecto Específico.

Flujo de eventos

Flujo básico:

Actor		Sistema	
Paso	Acciones	Paso	Acciones
1	El caso de uso Consultar Proyecto Específico comienza cuando el *RGPY* o el *RAPE* seleccionan alguna de las opciones del menú, para consultar alguno de los siguientes planes Actividades, Adquisiciones, Capacitación, Costo Estimado, Equipo de Trabajo, Protocolo de Entrega, Riesgos y selecciona la opción "Consultar".	2	El sistema le muestra la interfaz que contiene los datos del plan que seleccionó. [AE01] [AG01] [AG02]
3	El *RGPY* o el *RAPE* eligen alguna otra opción del menú.		

Flujos alternos:

Identificador	Nombre	Respuesta del sistema
AG01	Salir del sistema	El *RGPY* o el *RAPE* sale del sistema al elegir la opción "Salir"
AG02	Cambiar rol	El sistema lo muestra la lista de roles permitidos para el *Usuario*, para ingresar al sistema
AE01	Error de conexión con la base de datos.	El sistema manda un mensaje de error al *RGPY* o al *RAPE* indicándole que ha sucedido un error de conexión con la base de datos.

Tabla A.9 Detalle del caso de uso 4.3 Consultar Proyecto Específico

El diseño de la pantalla para registrar, modificar, consultar o eliminar los datos del registro del Plan del Proyecto Específico se muestran en las siguientes figuras:

Pantalla “Registrar, Modificar, Consultar o Eliminar el Plan del Proyecto Específico”

En esta pantalla se podrá registrar o modificar el Plan del Proyecto Específico.

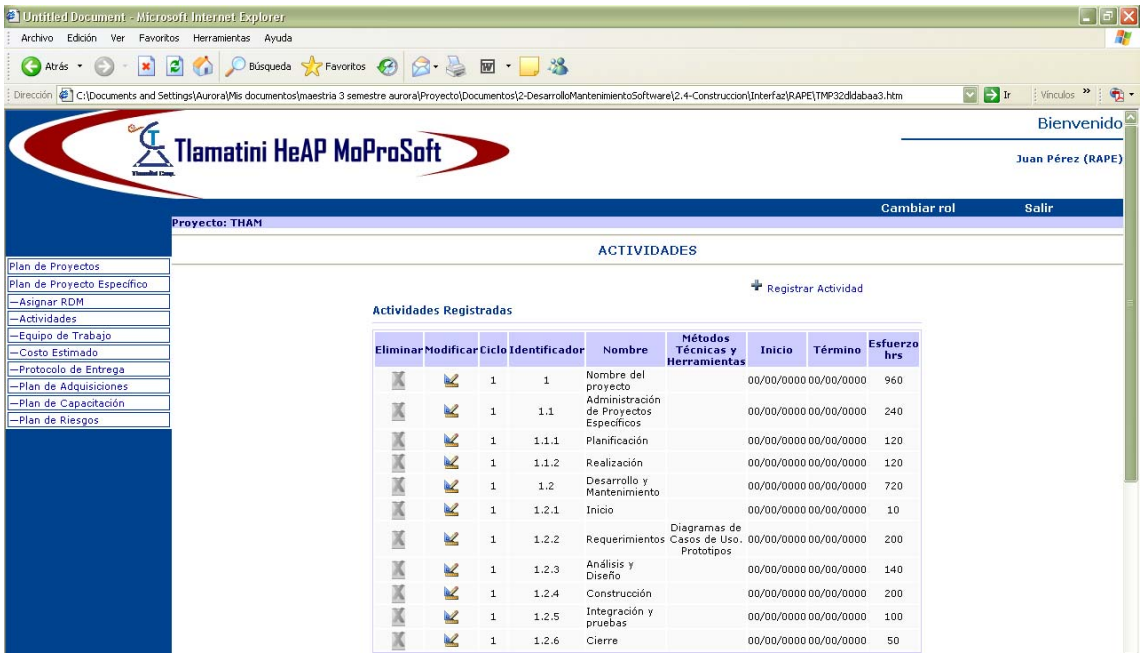


Figura A.11 Pantalla “Actividades”

Pantalla “Adquisiciones”

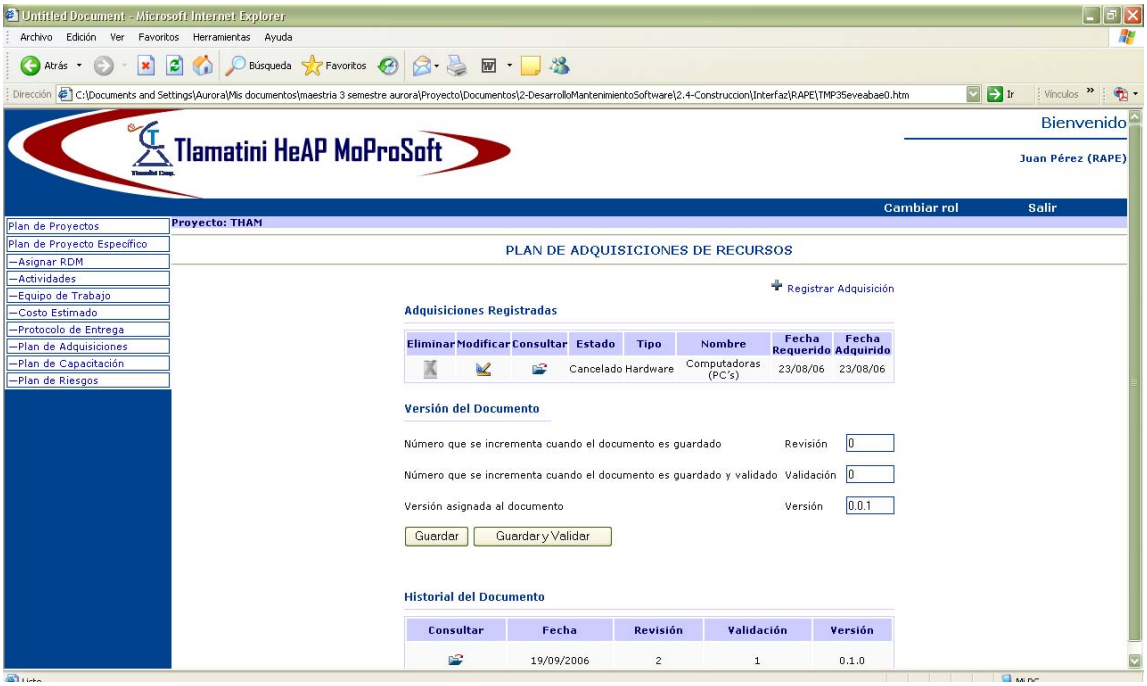


Figura A.12 Pantalla “Adquisiciones”

Pantalla “Capacitación”

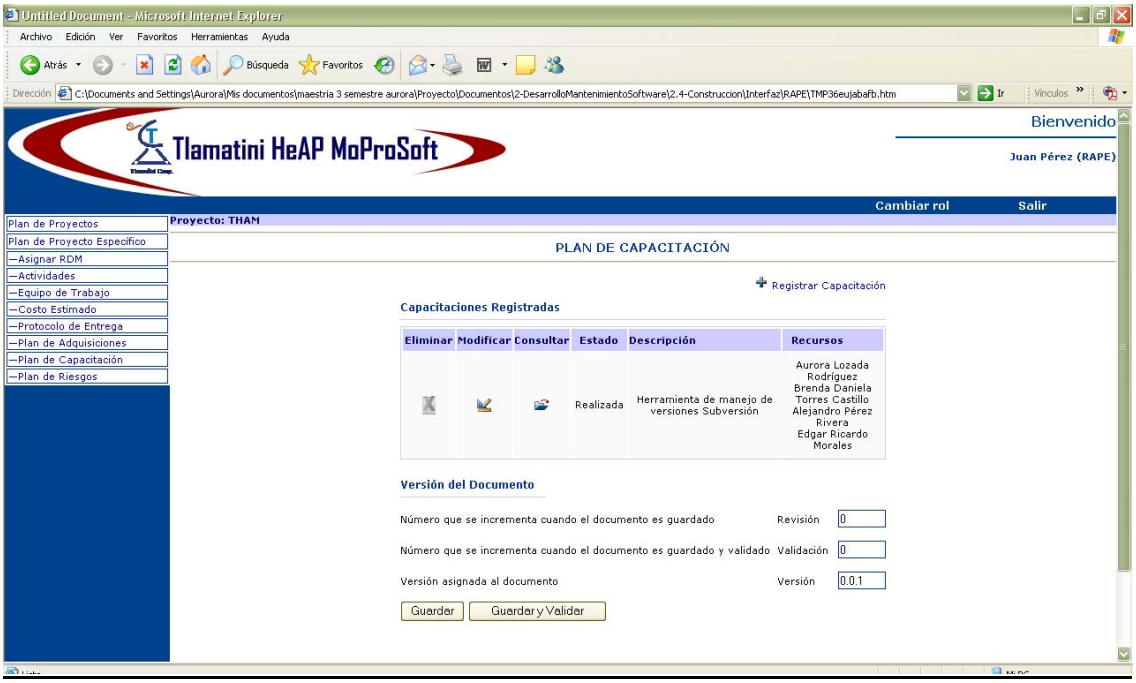


Figura A.13 Pantalla “Capacitación”

Pantalla “Costo Estimado”

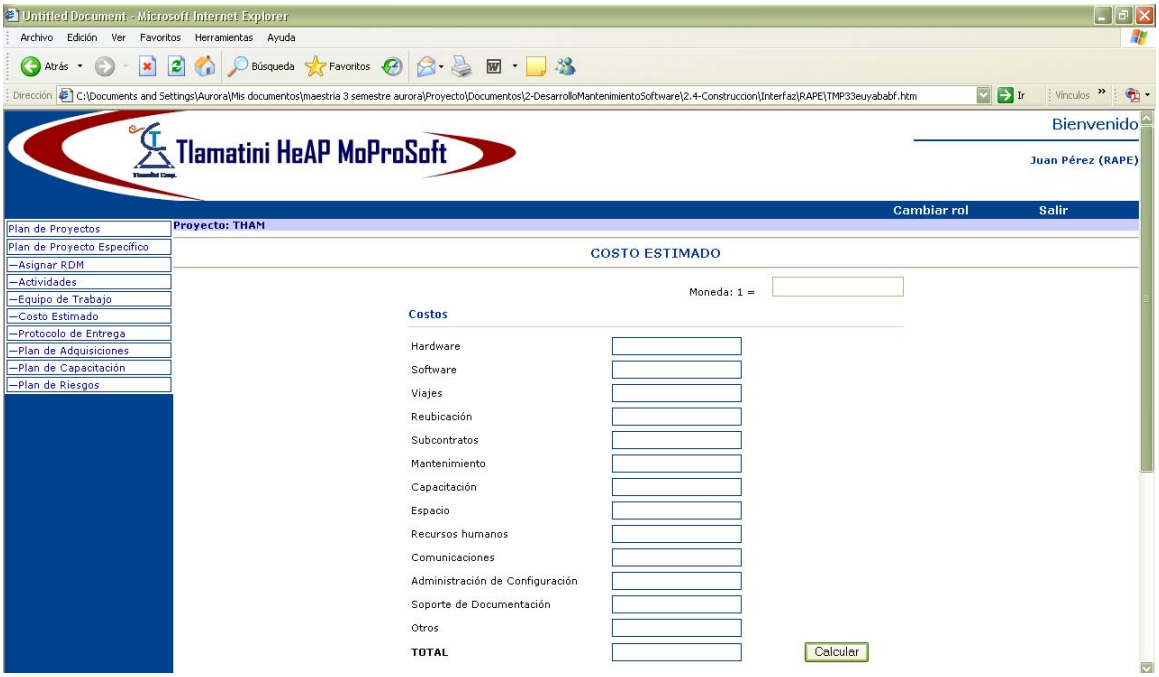


Figura A.14 Pantalla “Costo Estimado”

Pantalla “Equipo Trabajo”

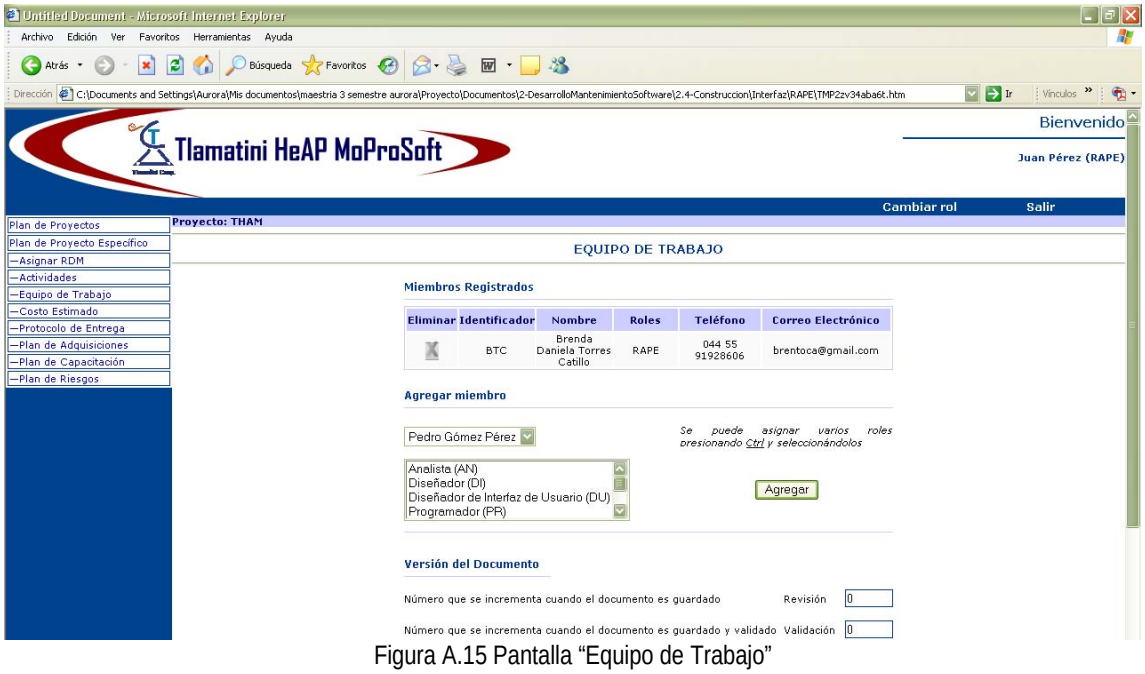


Figura A.15 Pantalla “Equipo de Trabajo”

Pantalla “Protocolo Entrega”

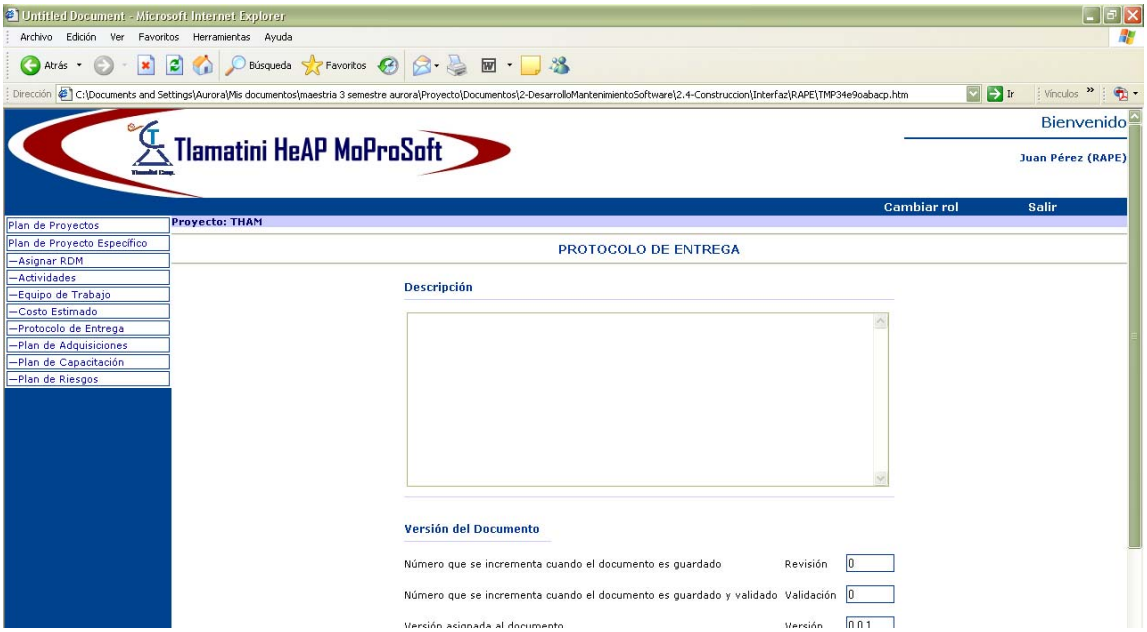


Figura A.16 Pantalla “Protocolo de Entrega”

Pantalla “Riesgos”

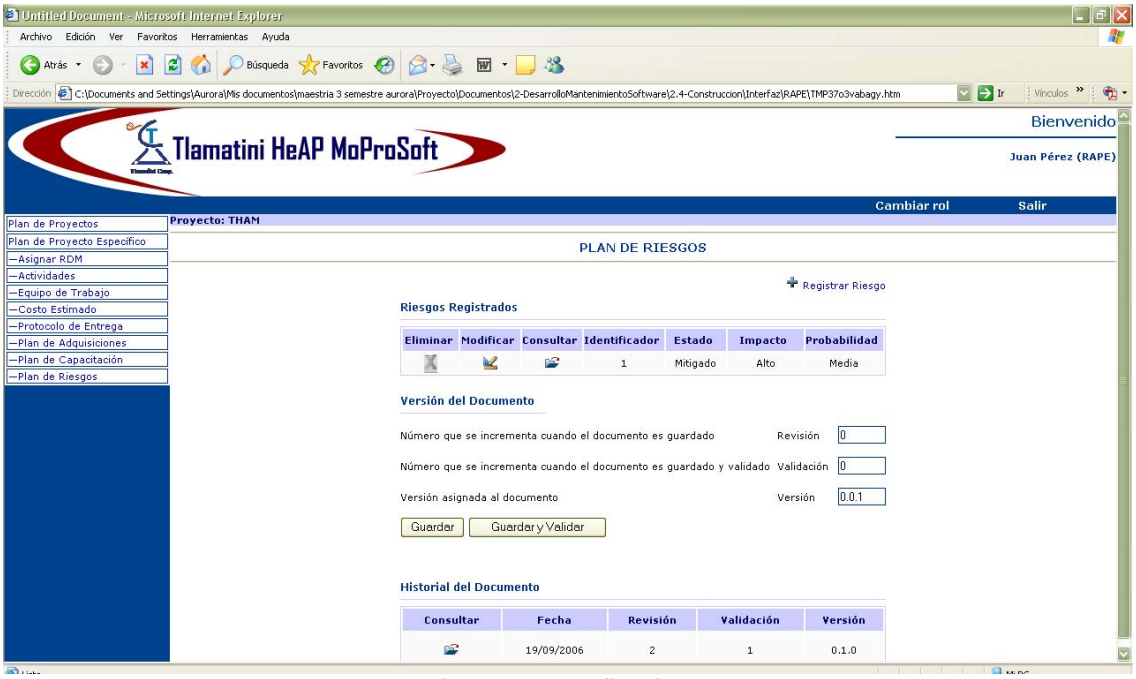
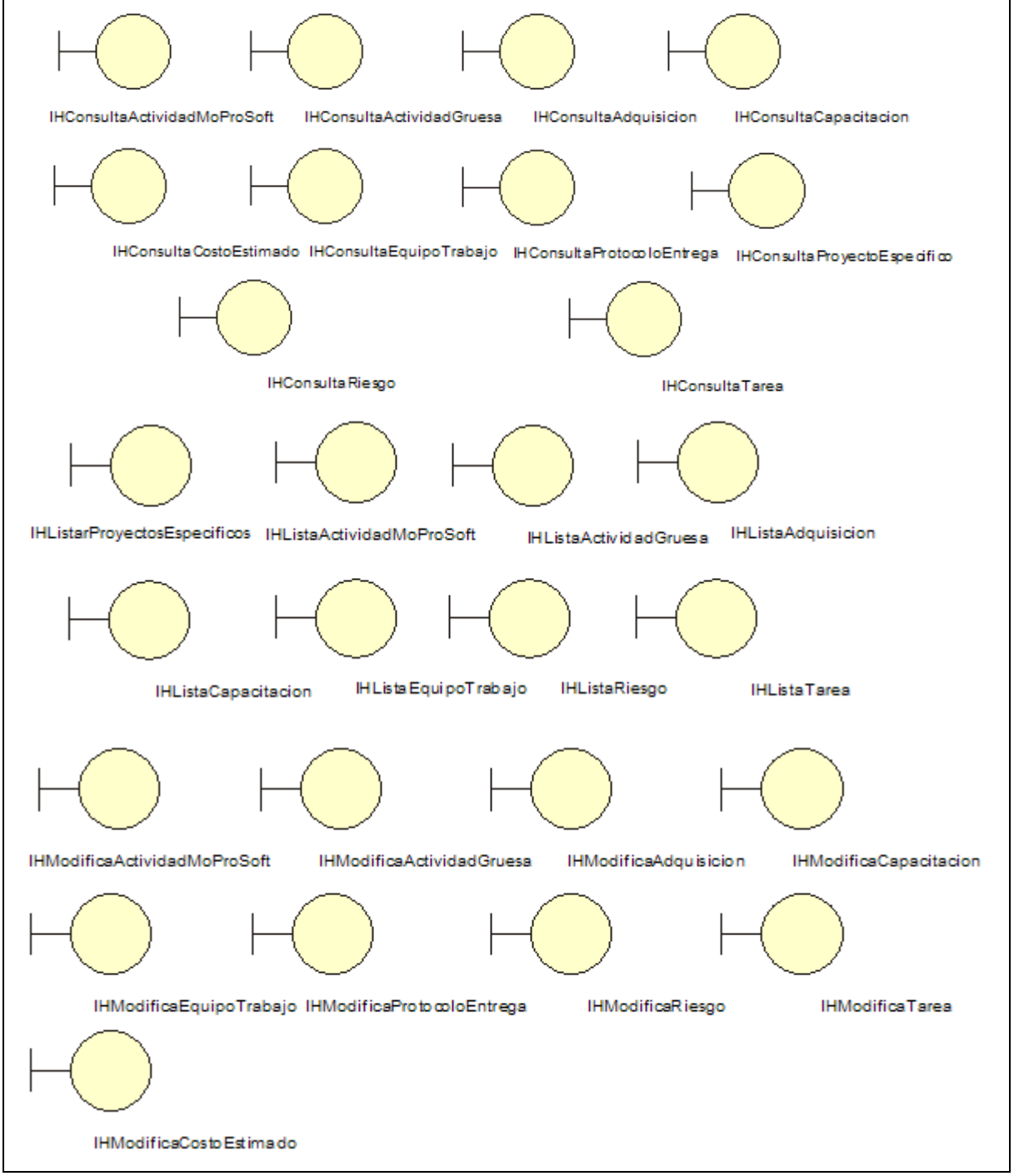


Figura A.17 Pantalla “Riesgo

Diagramas de clases del análisis
4. Administrar Proyecto Específico

Diagrama de clases de interfaz



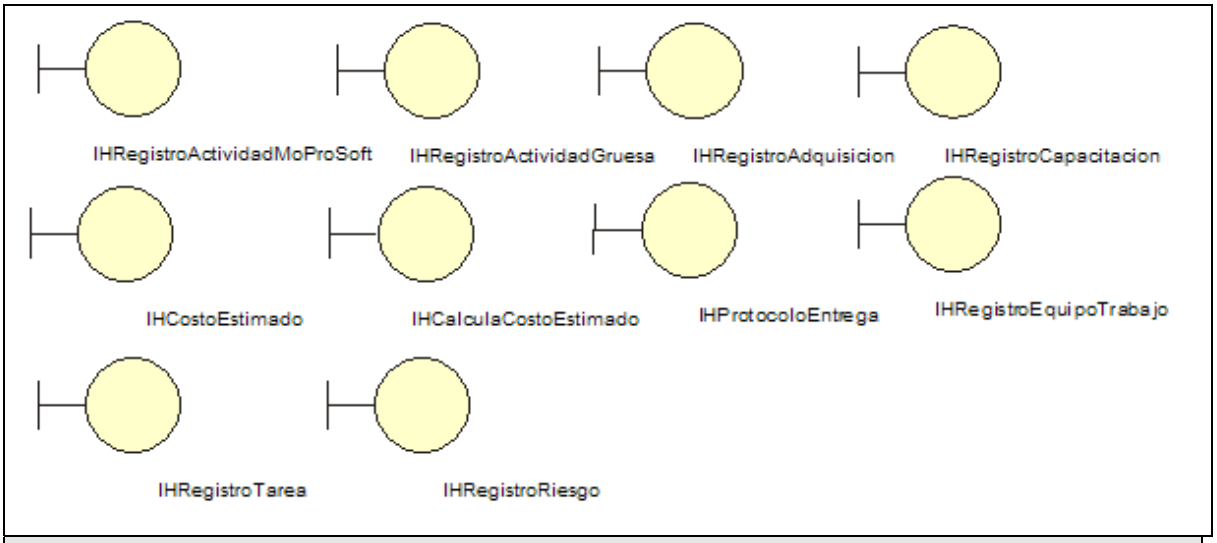


Diagrama de clases de entidad

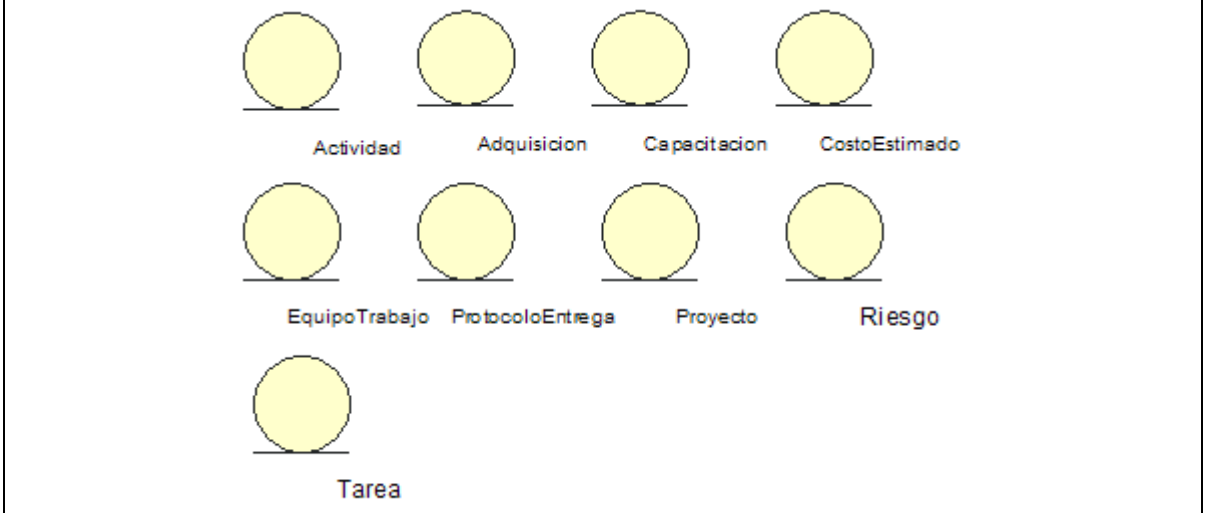
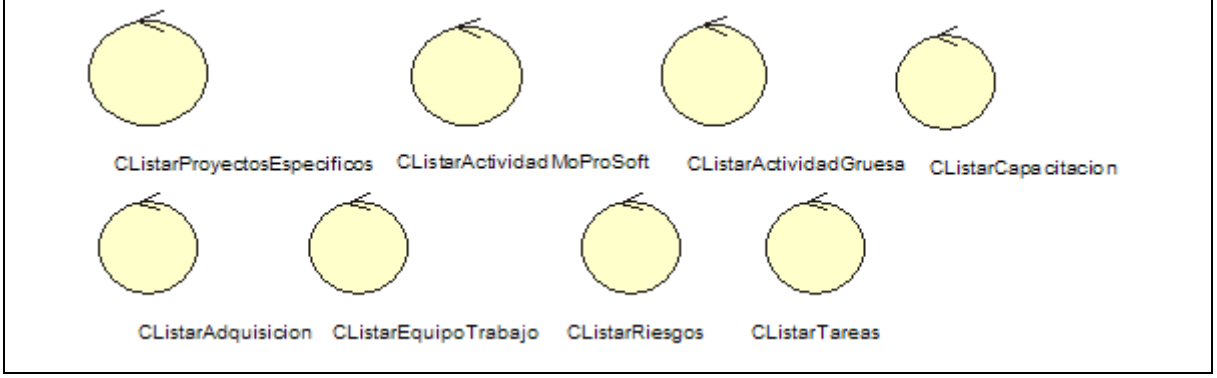
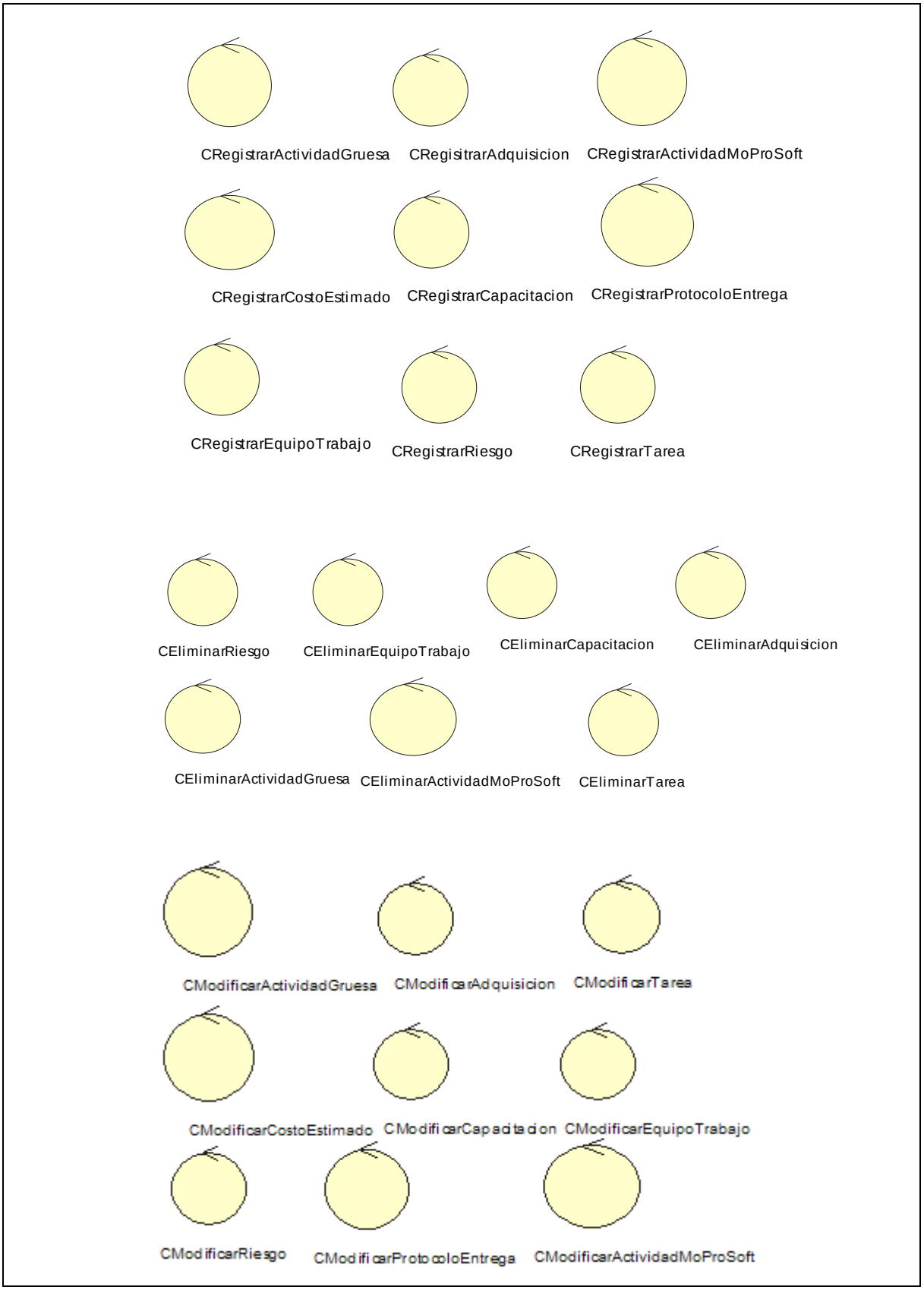


Diagrama de clases de control




CRegistrarActividadGruesa
CRegisitrarAdquisicion
CRegistrarActividadMoProSoft
CRegistrarCostoEstimado
CRegistrarCapacitacion
CRegistrarProtocoloEntrega
CRegistrarEquipoTrabajo
CRegistrarRiesgo
CRegistrarTarea
CEliminarRiesgo
CEliminarEquipoTrabajo
CEliminarCapacitacion
CEliminarAdquisicion
CEliminarActividadGruesa
CEliminarActividadMoProSoft
CEliminarTarea
CModificarActividadGruesa
CModificarAdquisicion
CModificarTarea
CModificarCostoEstimado
CModificarCapacitacion
CModificarEquipoTrabajo
CModificarRiesgo
CModificarProtocoloEntrega
CModificarActividadMoProSoft

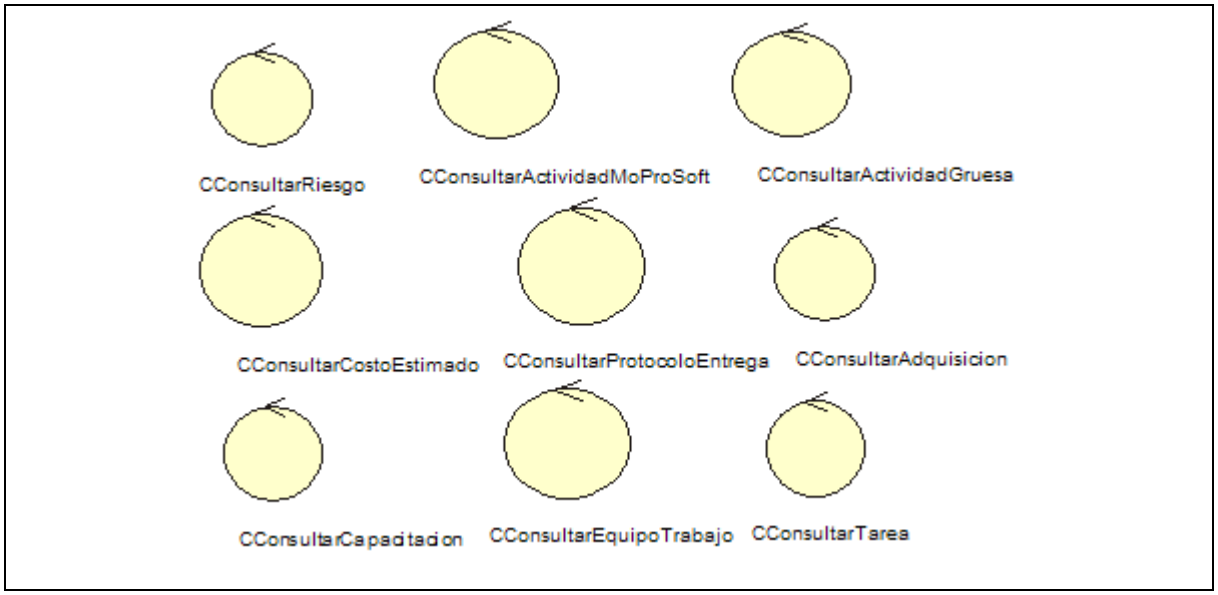
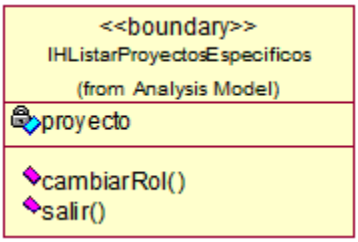


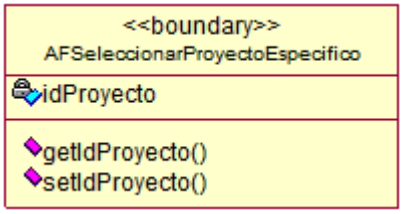
Figura A.18 Diagrama de clases del análisis del caso de uso 4. Administrar Proyecto Específico

Diagramas de clases del diseño
4. Administrar Proyecto Específico

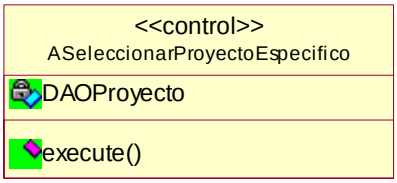
Proyecto Específico

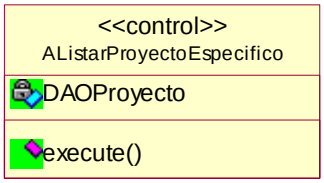
Interfaz



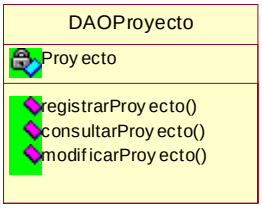


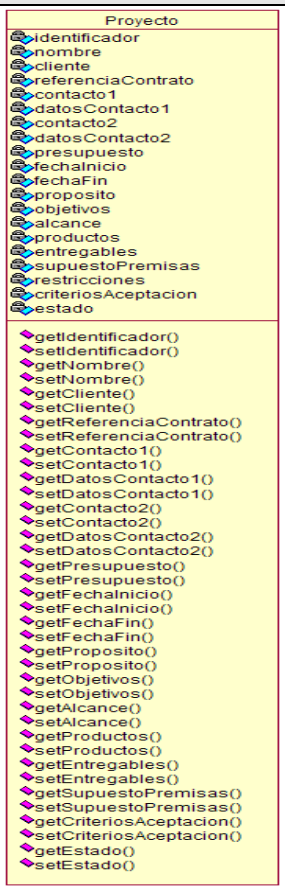
Control





Entidad

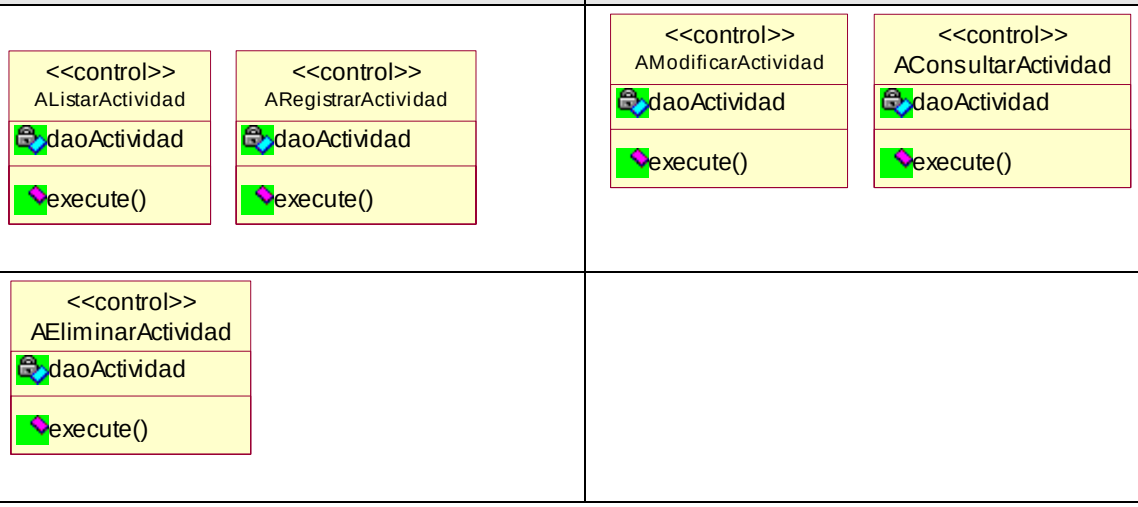


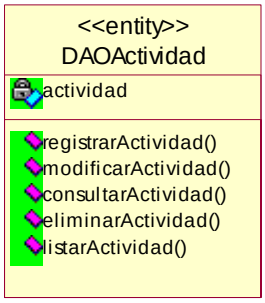
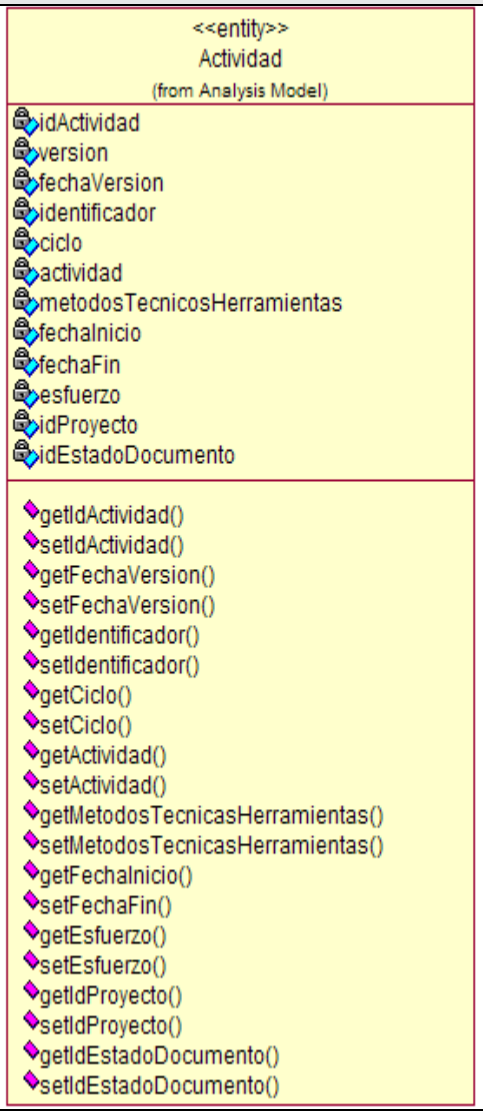


Actividad MoProSoft

Interfaz

Control



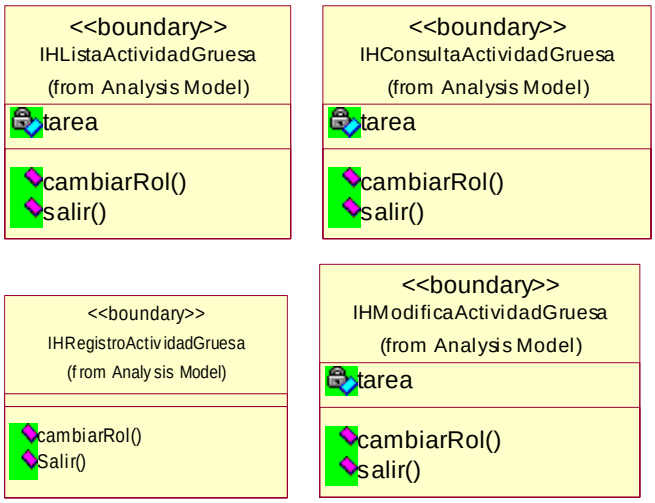
Entidad	
	

<<entity>>
DAOActividad
actividad
registrarActividad()
modificarActividad()
consultarActividad()
eliminarActividad()
listarActividad()

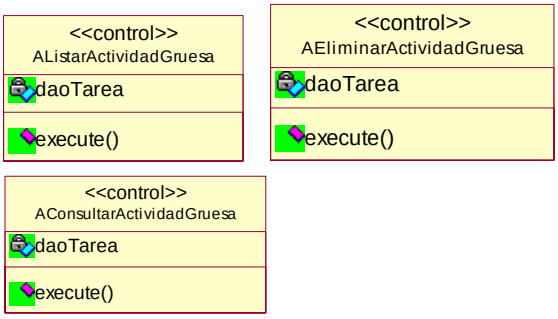
<<entity>>
Actividad
(from Analysis Model)
idActividad
version
fechaVersion
identificador
ciclo
actividad
metodosTecnicosHerramientas
fechaInicio
fechaFin
esfuerzo
idProyecto
idEstadoDocumento
getIdActividad()
setIdActividad()
getFechaVersion()
setFechaVersion()
getIdentificador()
setIdentificador()
getCiclo()
setCiclo()
getActividad()
setActividad()
getMetodosTecnicasHerramientas()
setMetodosTecnicasHerramientas()
getFechaInicio()
setFechaFin()
getEsfuerzo()
setEsfuerzo()
getIdProyecto()
setIdProyecto()
getIdEstadoDocumento()
setIdEstadoDocumento()

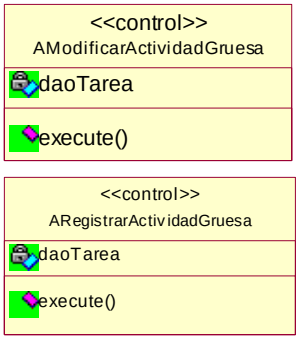
Actividad Gruesa

Interfaz

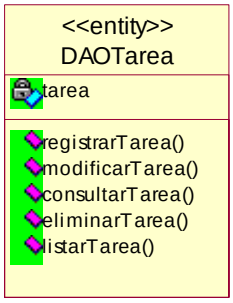


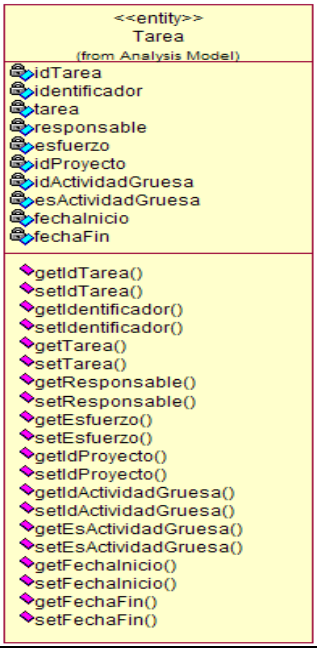
Control





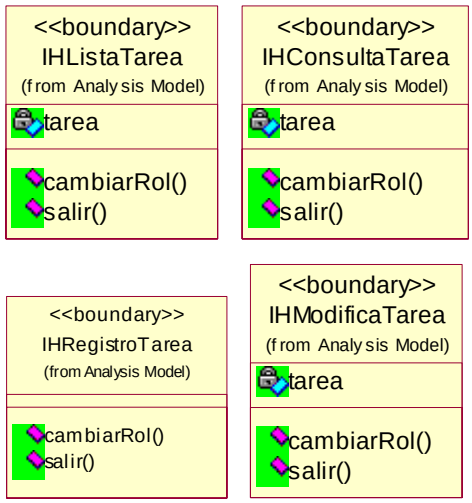
Entidad



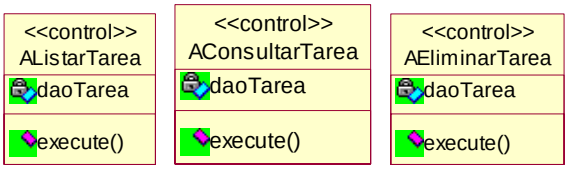


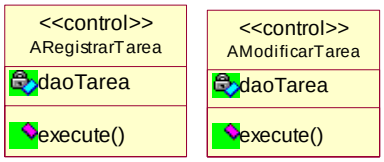
Tarea

Interfaz

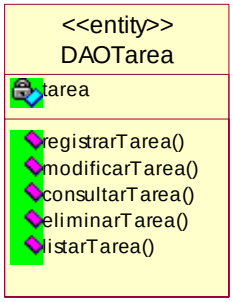


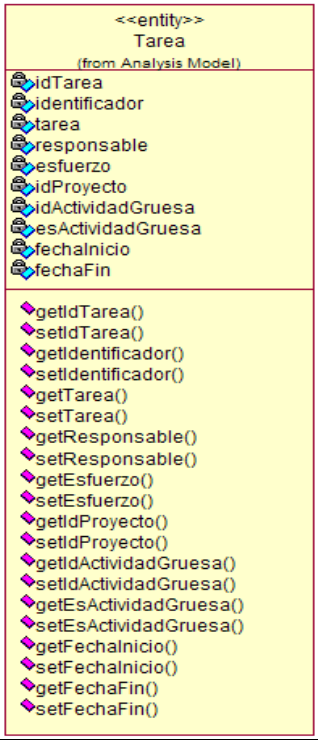
Control





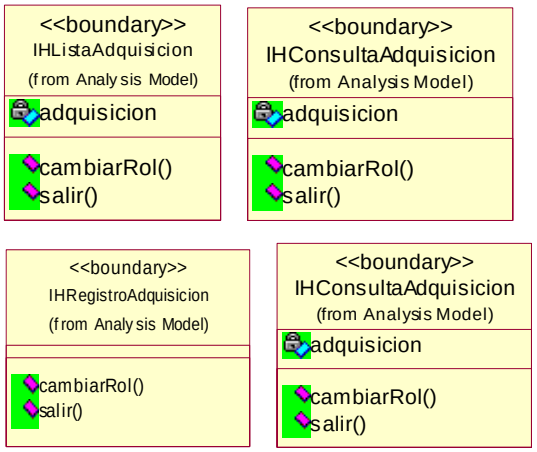
Entidad

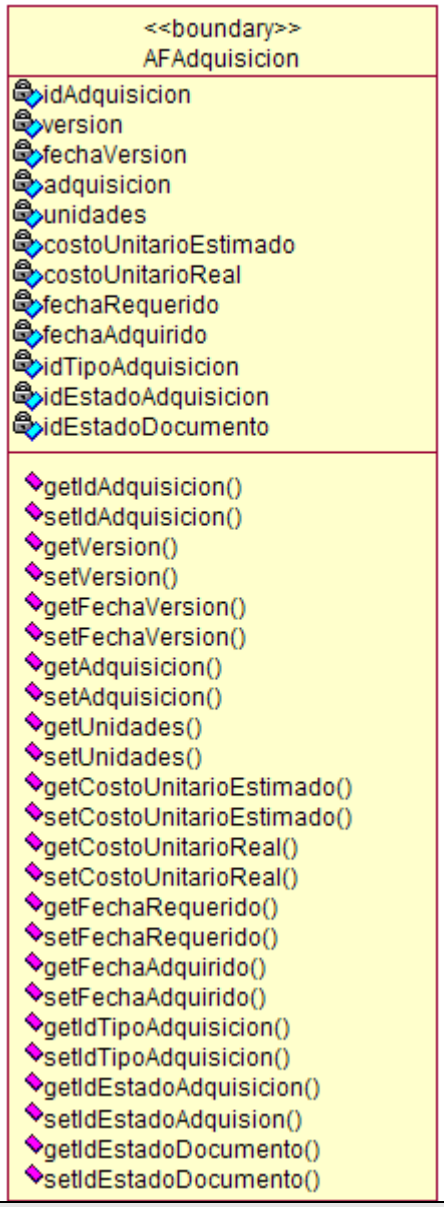




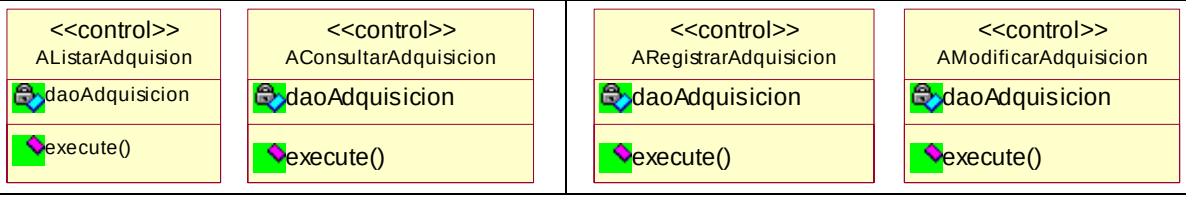
Adquisición

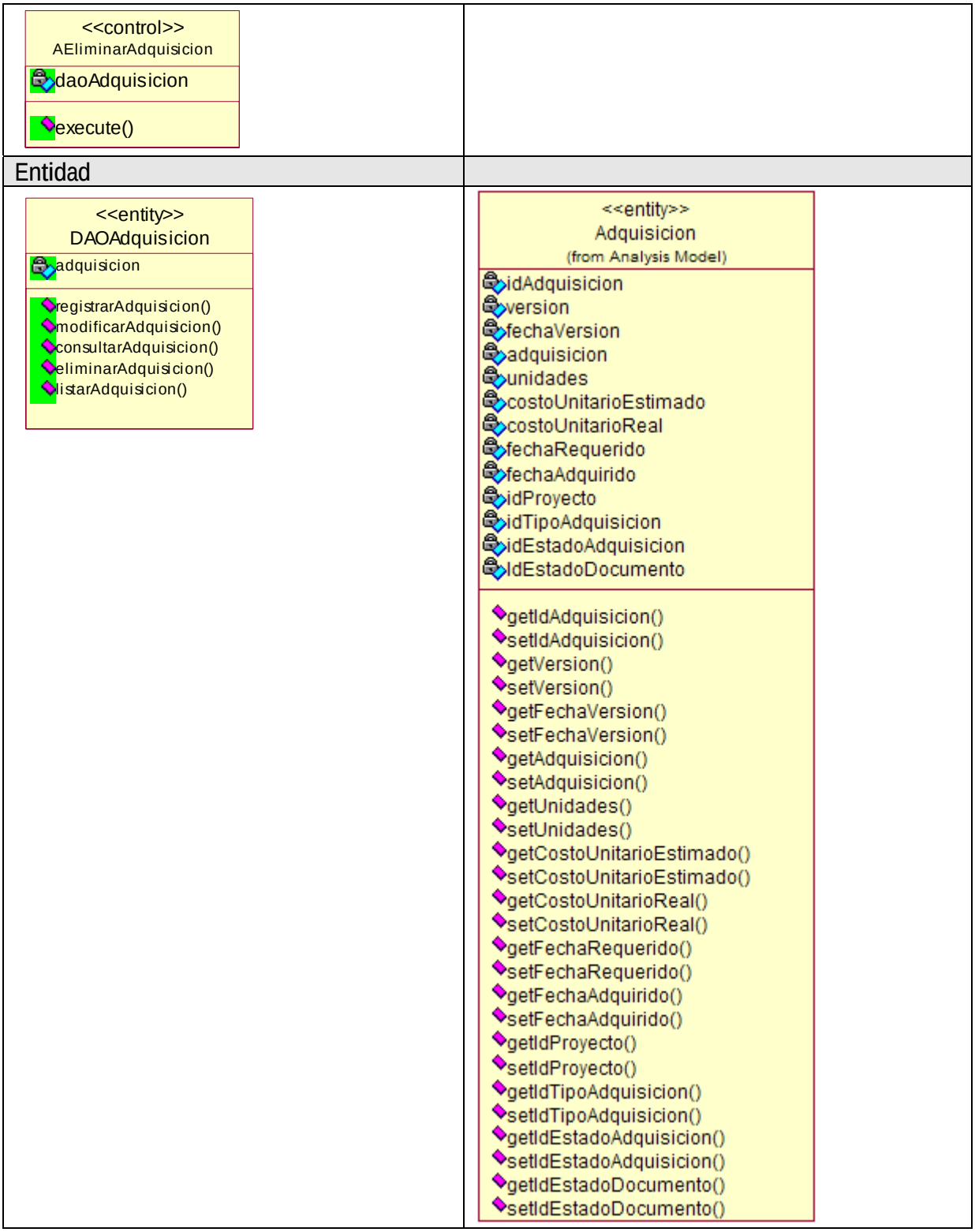
Interfaz





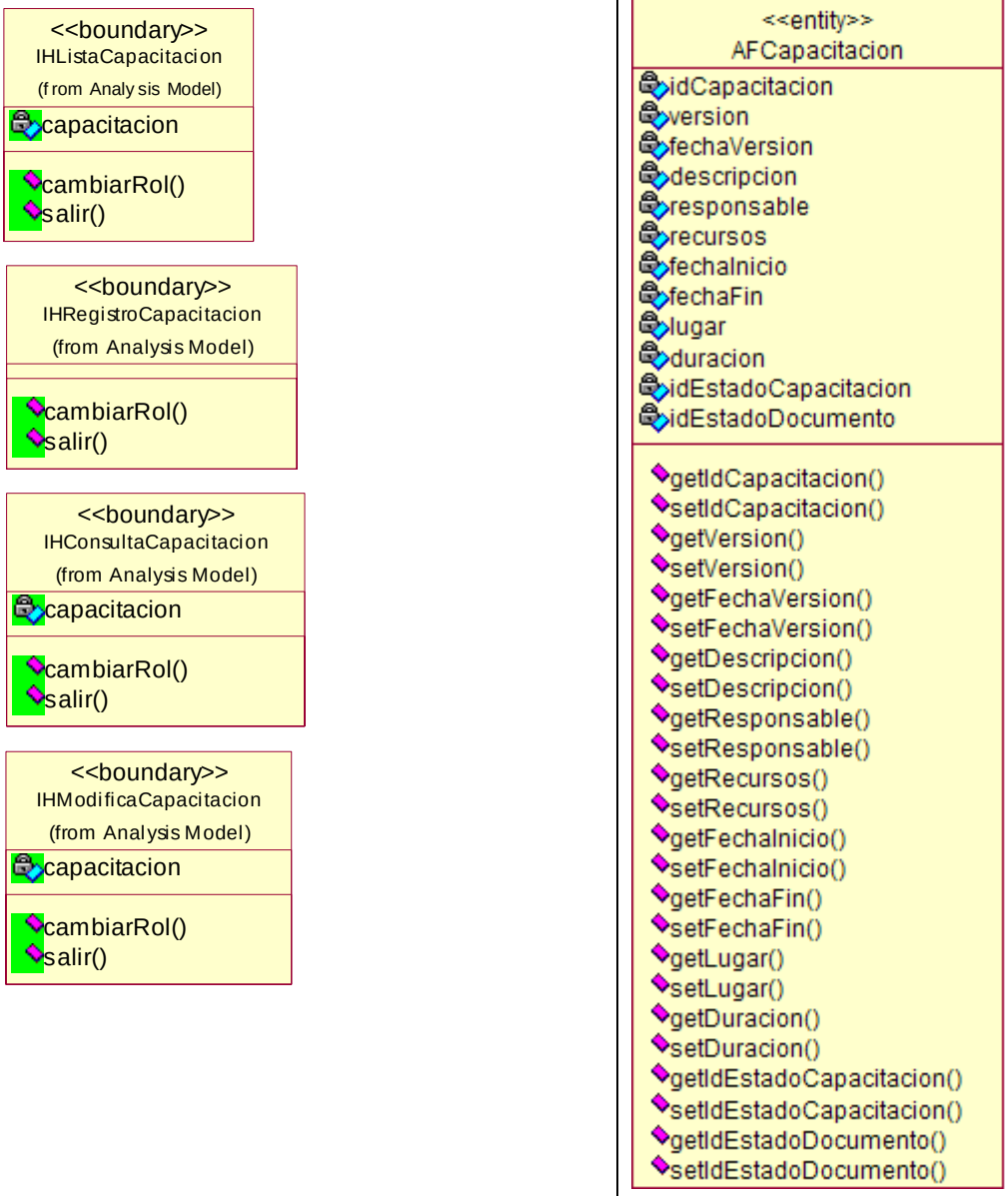
Control



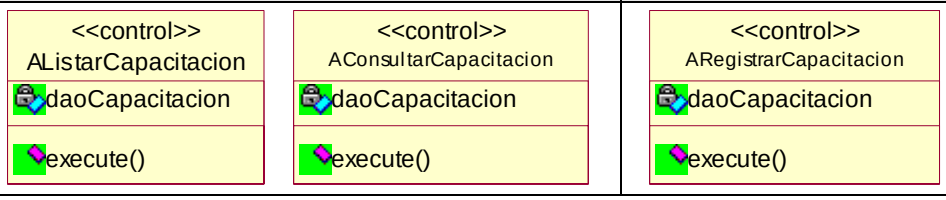

<<control>>
AEliminarAdquisicion
daoAdquisicion
execute()
Entidad
<<entity>>
DAOAdquisicion
adquisicion
registrarAdquisicion()
modificarAdquisicion()
consultarAdquisicion()
eliminarAdquisicion()
listarAdquisicion()
<<entity>>
Adquisicion
(from Analysis Model)
idAdquisicion
version
fechaVersion
adquisicion
unidades
costoUnitarioEstimado
costoUnitarioReal
fechaRequerido
fechaAdquirido
idProyecto
idTipoAdquisicion
idEstadoAdquisicion
IdEstadoDocumento
getIdAdquisicion()
setIdAdquisicion()
getVersion()
setVersion()
getFechaVersion()
setFechaVersion()
getAdquisicion()
setAdquisicion()
getUnidades()
setUnidades()
getCostoUnitarioEstimado()
setCostoUnitarioEstimado()
getCostoUnitarioReal()
setCostoUnitarioReal()
getFechaRequerido()
setFechaRequerido()
getFechaAdquirido()
setFechaAdquirido()
getIdProyecto()
setIdProyecto()
getIdTipoAdquisicion()
setIdTipoAdquisicion()
getIdEstadoAdquisicion()
setIdEstadoAdquisicion()
getIdEstadoDocumento()
setIdEstadoDocumento()

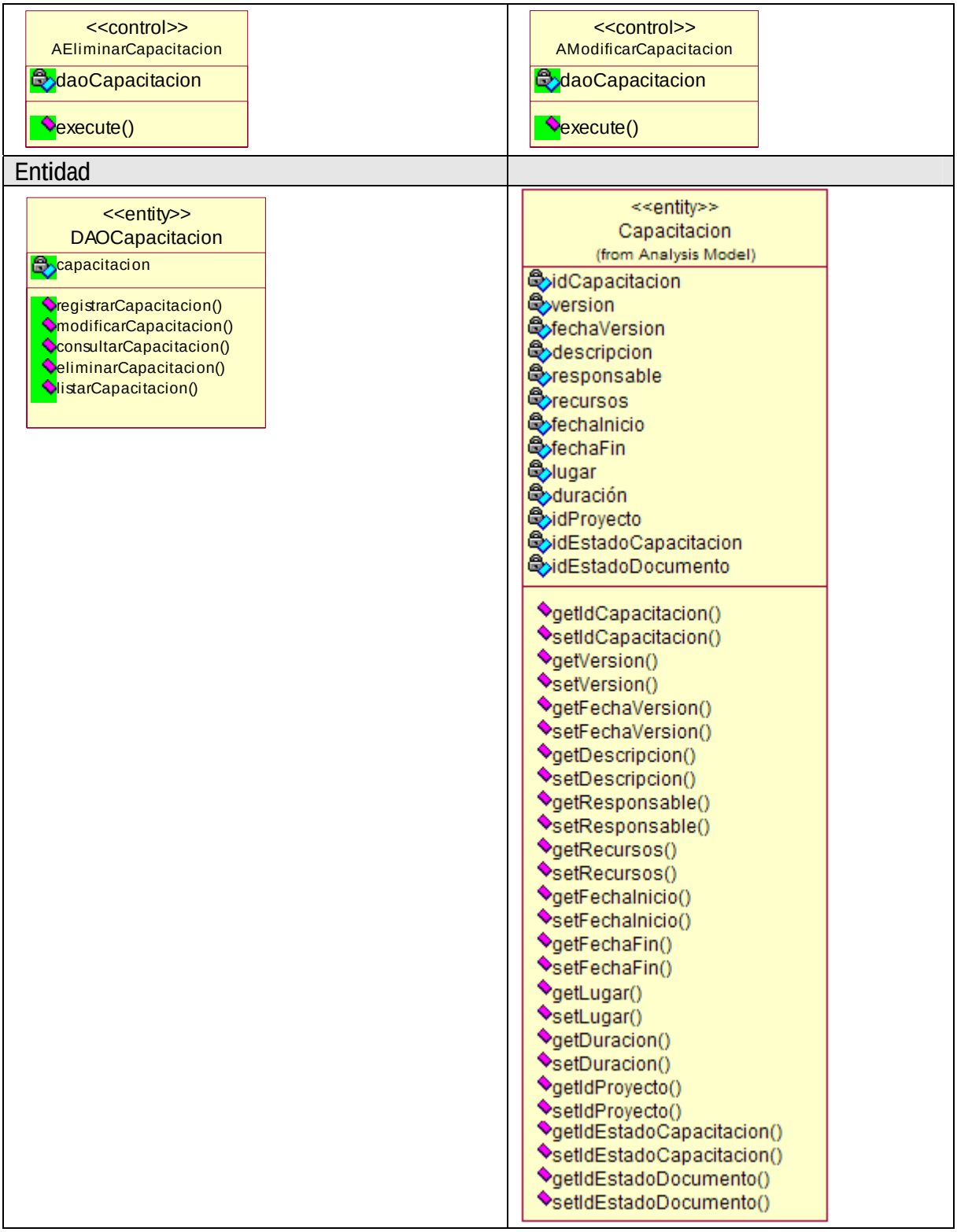
Capacitación

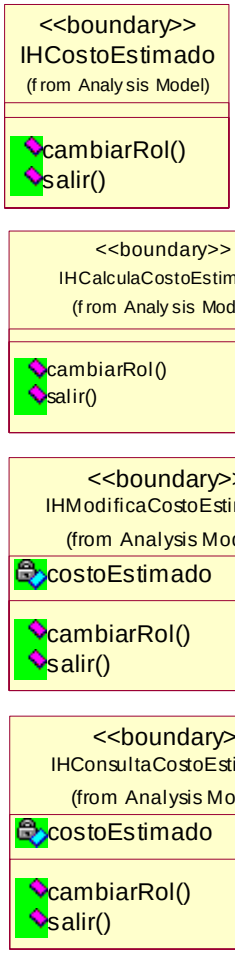
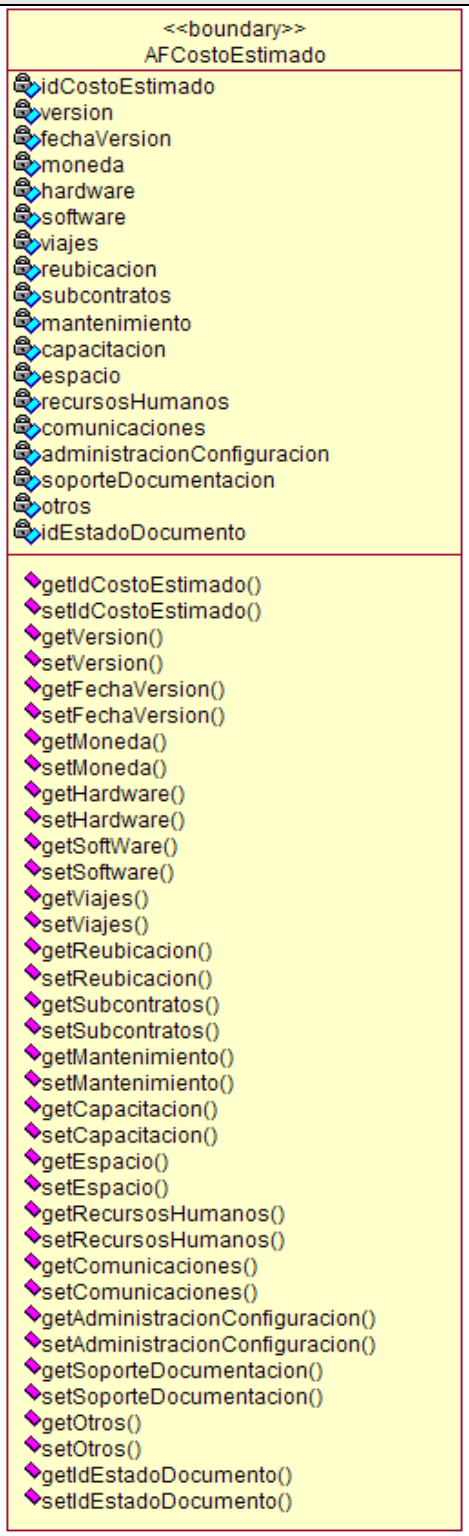
Interfaz

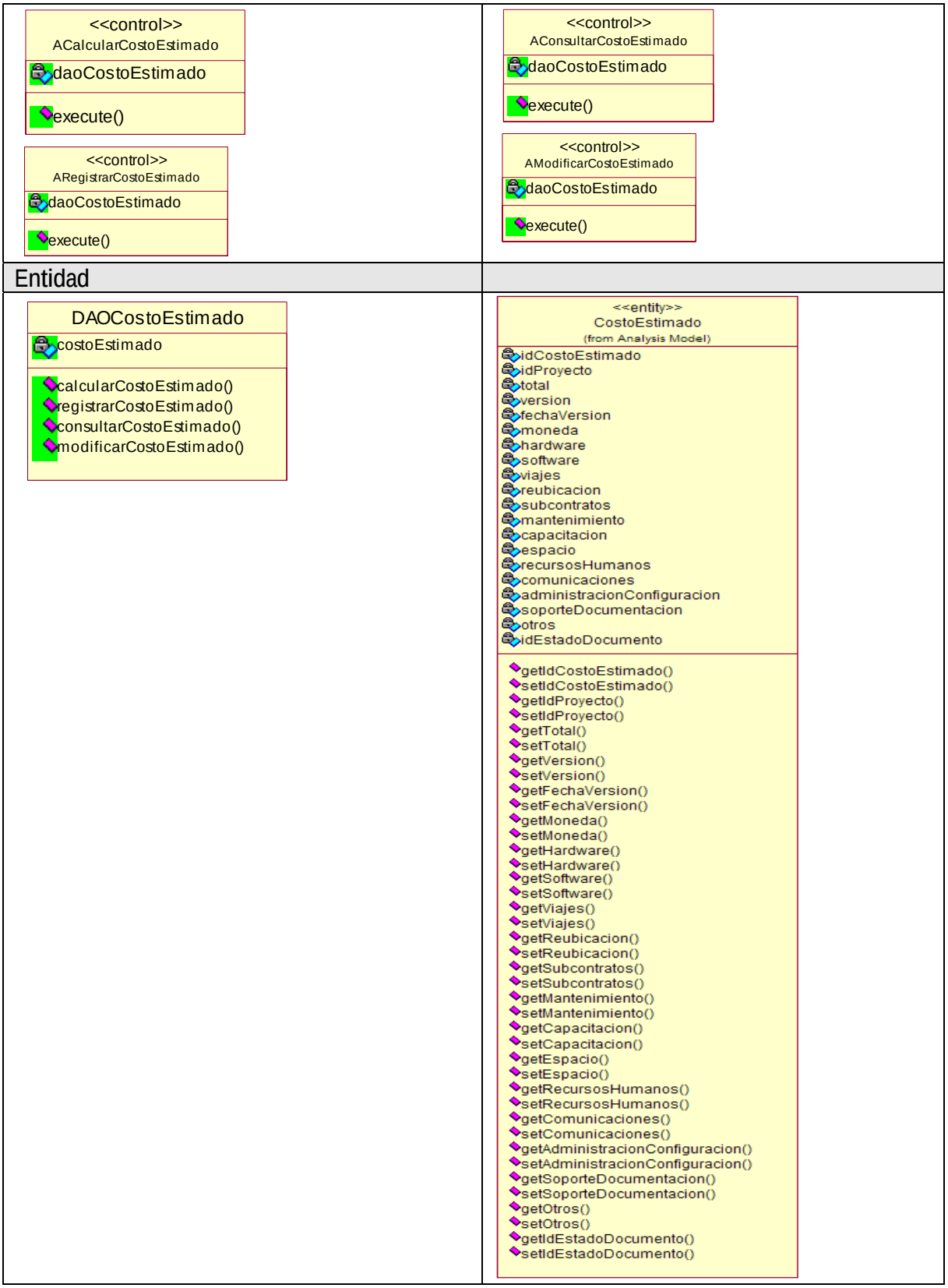


Control



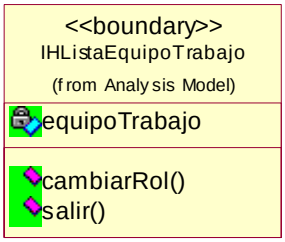

<<control>>
AEliminarCapacitacion
daoCapacitacion
execute()
<<control>>
AModificarCapacitacion
daoCapacitacion
execute()
Entidad
<<entity>>
DAOCapacitacion
capacitacion
registrarCapacitacion()
modificarCapacitacion()
consultarCapacitacion()
eliminarCapacitacion()
listarCapacitacion()
<<entity>>
Capacitacion
(from Analysis Model)
idCapacitacion
version
fechaVersion
descripcion
responsable
recursos
fechaInicio
fechaFin
lugar
duración
idProyecto
idEstadoCapacitacion
idEstadoDocumento
getIdCapacitacion()
setIdCapacitacion()
getVersion()
setVersion()
getFechaVersion()
setFechaVersion()
getDescripcion()
setDescripcion()
getResponsable()
setResponsable()
getRecursos()
setRecursos()
getFechaInicio()
setFechaInicio()
getFechaFin()
setFechaFin()
getLugar()
setLugar()
getDuracion()
setDuracion()
getIdProyecto()
setIdProyecto()
getIdEstadoCapacitacion()
setIdEstadoCapacitacion()
getIdEstadoDocumento()
setIdEstadoDocumento()

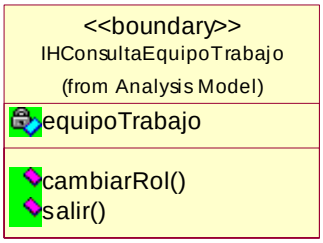
Costo Estimado	
Interfaz	
	
Control	

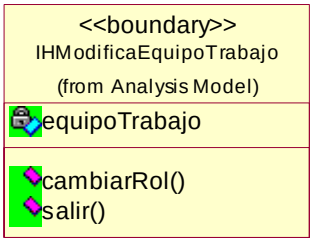

<<control>>
ACalcularCostoEstimado
daoCostoEstimado
execute()
<<control>>
AConsultarCostoEstimado
daoCostoEstimado
execute()
<<control>>
ARegistrarCostoEstimado
daoCostoEstimado
execute()
<<control>>
AModificarCostoEstimado
daoCostoEstimado
execute()
Entidad
DAOCostoEstimado
costoEstimado
calcularCostoEstimado()
registrarCostoEstimado()
consultarCostoEstimado()
modificarCostoEstimado()
<<entity>>
CostoEstimado
(from Analysis Model)
idCostoEstimado
idProyecto
total
version
fechaVersion
moneda
hardware
software
viajes
reubicacion
subcontratos
mantenimiento
capacitacion
espacio
recursosHumanos
comunicaciones
administracionConfiguracion
soporteDocumentacion
otros
idEstadoDocumento
getIdCostoEstimado()
setIdCostoEstimado()
getIdProyecto()
setIdProyecto()
getTotal()
setTotal()
getVersion()
setVersion()
getFechaVersion()
setFechaVersion()
getMoneda()
setMoneda()
getHardware()
setHardware()
getSoftware()
setSoftware()
getViajes()
setViajes()
getReubicacion()
setReubicacion()
getSubcontratos()
setSubcontratos()
getMantenimiento()
setMantenimiento()
getCapacitacion()
setCapacitacion()
getEspacio()
setEspacio()
getRecursosHumanos()
setRecursosHumanos()
getComunicaciones()
setComunicaciones()
getAdministracionConfiguracion()
setAdministracionConfiguracion()
getSoporteDocumentacion()
setSoporteDocumentacion()
getOtros()
setOtros()
getIdEstadoDocumento()
setIdEstadoDocumento()

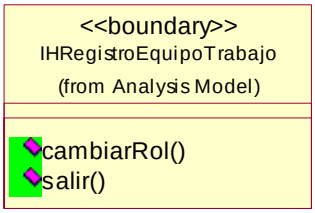
Equipo Trabajo

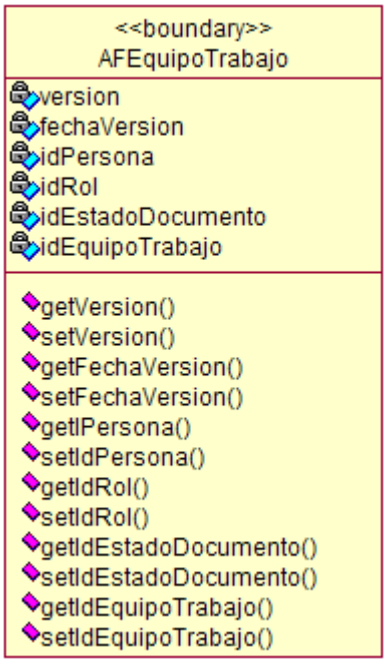
Interfaz



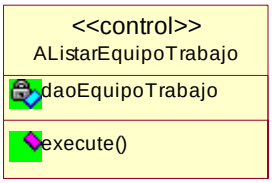


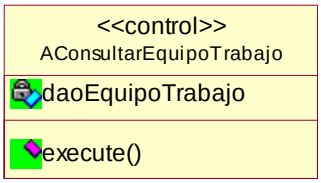


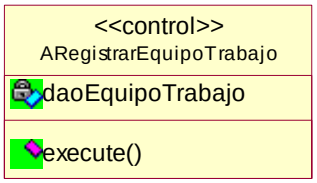


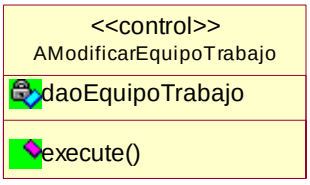


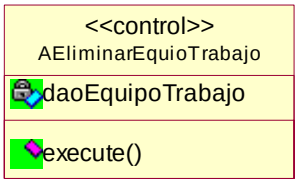
Control

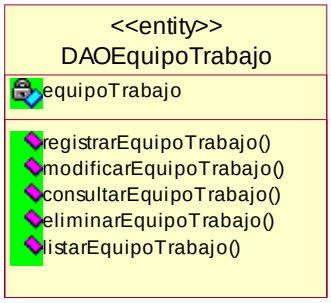
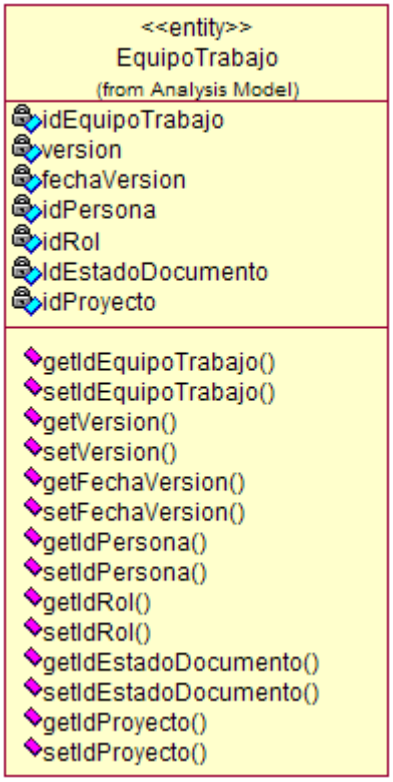










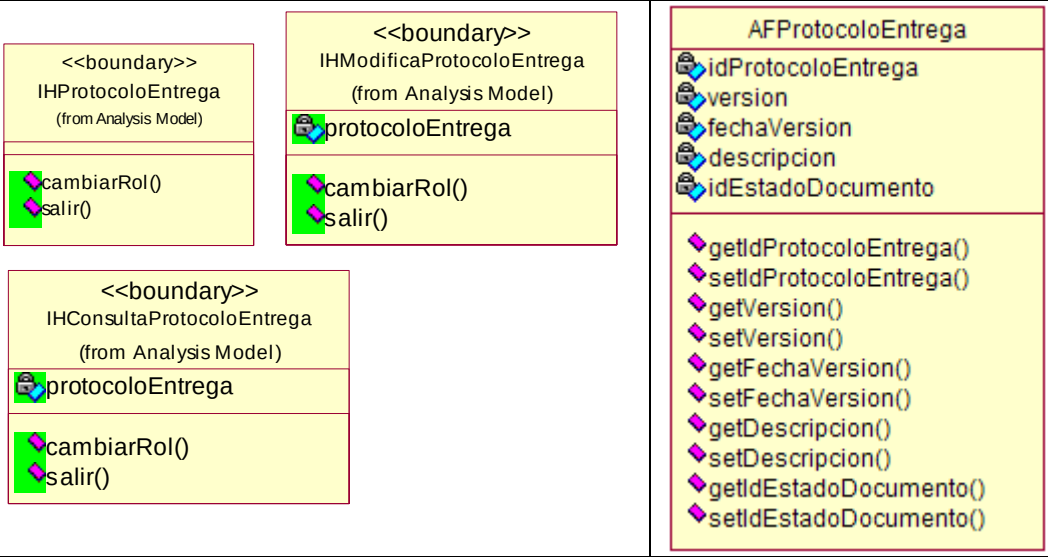
Entidad	
	

<<entity>>
DAOEquipoTrabajo
equipoTrabajo
registrarEquipoTrabajo()
modificarEquipoTrabajo()
consultarEquipoTrabajo()
eliminarEquipoTrabajo()
listarEquipoTrabajo()

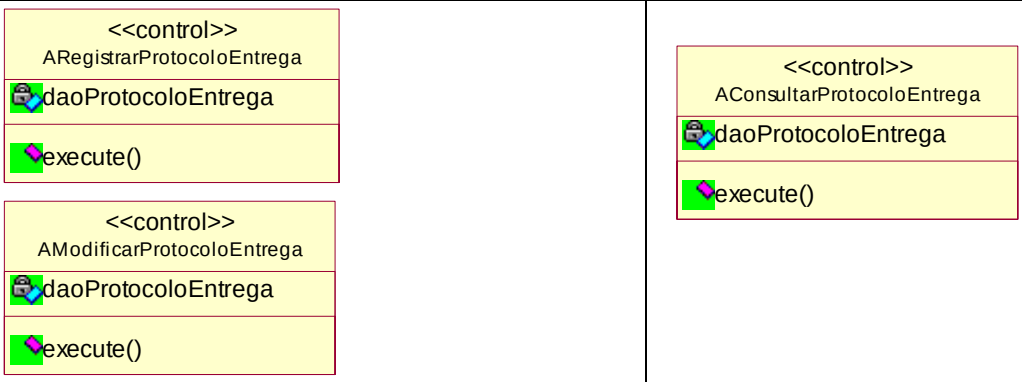
<<entity>>
EquipoTrabajo
(from Analysis Model)
idEquipoTrabajo
version
fechaVersion
idPersona
idRol
IdEstadoDocumento
idProyecto
getIdEquipoTrabajo()
setIdEquipoTrabajo()
getVersion()
setVersion()
getFechaVersion()
setFechaVersion()
getIdPersona()
setIdPersona()
getIdRol()
setIdRol()
getIdEstadoDocumento()
setIdEstadoDocumento()
getIdProyecto()
setIdProyecto()

Protocolo de Entrega

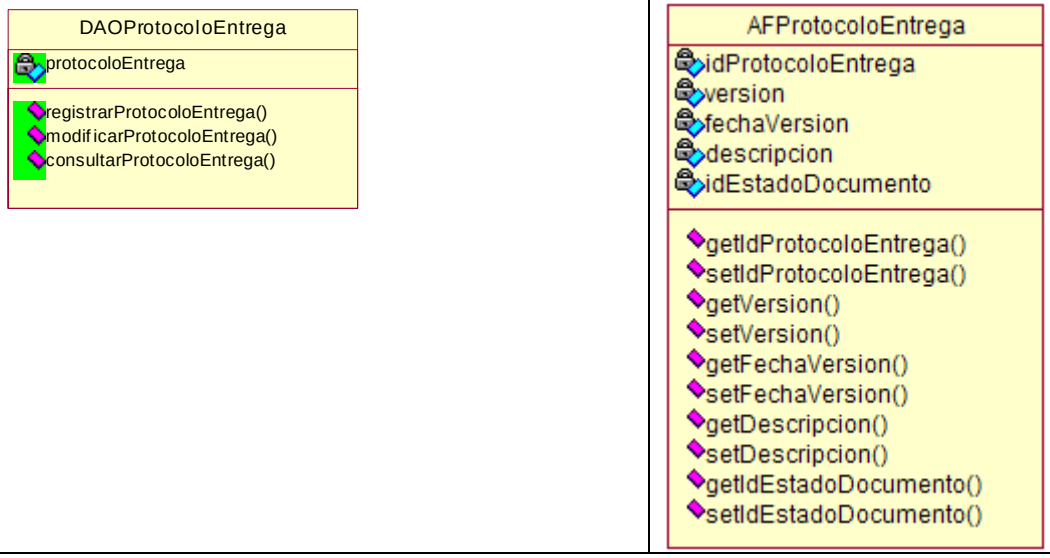
Interfaz



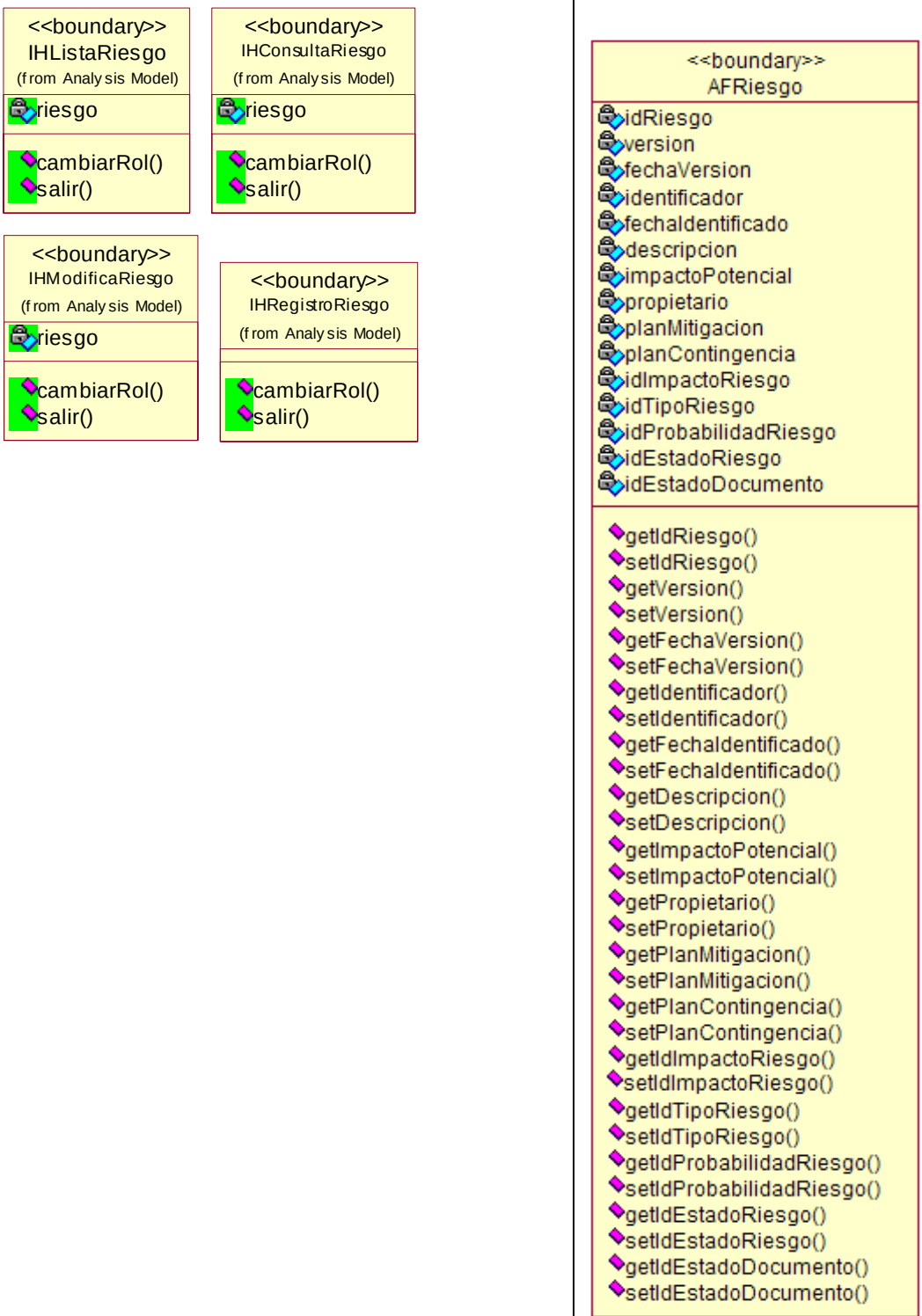
Control



Entidad



Riesgo	
Interfaz	



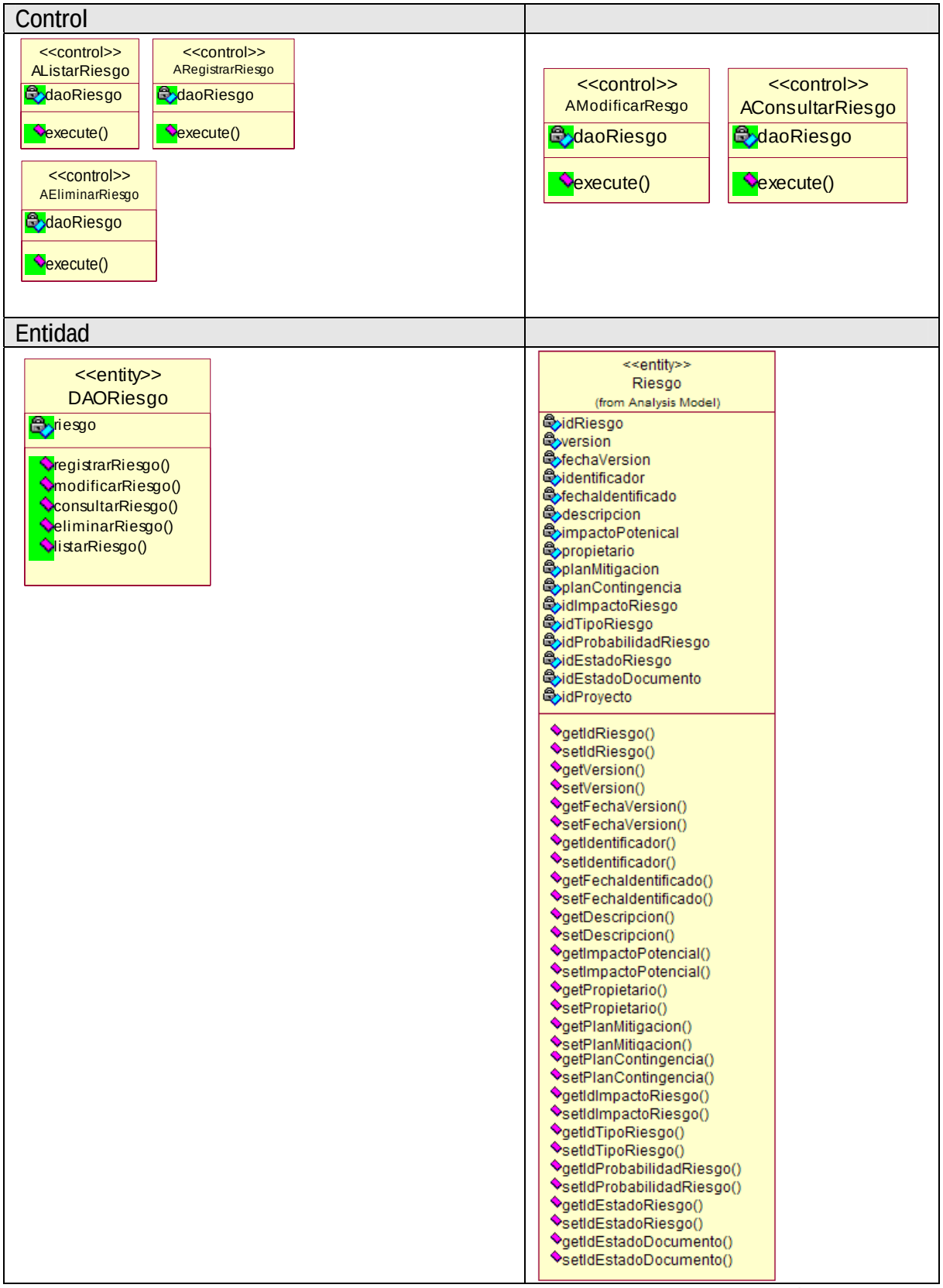


Figura A.19 Diagrama de clases del diseño del caso de uso 4. Administrar Proyecto Específico

Detalle del caso de uso
5. Cerrar sesión

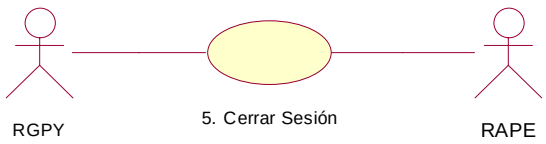
<table>
<tr><th colspan="4">Diagrama del caso de uso</th></tr>
<tr><td colspan="4"> 
</td></tr>
<tr><th colspan="4">Definición</th></tr>
<tr><td>Caso de uso:</td><td colspan="3">5. Cerrar Sesión</td></tr>
<tr><td>Actor:</td><td colspan="3">RGPY y RAPE
Nota: Nos referiremos a ellos con el nombre genérico de Usuario.</td></tr>
<tr><td>Descripción:</td><td colspan="3">El caso de uso Cerrar Sesi{on nos muestra como el Usuario sale del sistema o cambia de rol.</td></tr>
<tr><td>Precondiciones:</td><td colspan="3">3. El Usuario debe tener una conexión a Internet y haber cargado la página del sistema en su navegador
4. El Usuario ingresó al sistema</td></tr>
<tr><td>Poscondiciones:</td><td colspan="3">2. El Usuario Salió del sistema</td></tr>
<tr><th colspan="4">Flujo de eventos</th></tr>
<tr><td colspan="4">Flujo básico:</td></tr>
<tr><td colspan="2">Actor</td><td colspan="2">Sistema</td></tr>
<tr><td>Paso</td><td>Acciones</td><td>Paso</td><td>Acciones</td></tr>
<tr><td>1</td><td>Este caso de uso se inicia cuando el Usuario selecciona la opción “salir” o “cambiar rol”.</td><td>2</td><td>El sistema cierra la sesión del usuario, en el caso de haber elegido la opción “salir”, si por el contrario eligiera la opción “cambiar rol” el sistema le mostraría la pantalla para seleccionar el rol con el que quiere ingresar al sistema y le muestra la lista de roles permitidos para el Usuario.</td></tr>
<tr><td>3</td><td>El Usuario elige uno de los roles de la lista con el cual desea ingresar</td><td>4</td><td>Este caso de uso termina cuando el sistema permite el ingreso al Usuario con el rol que eligió [AG01] [AG02]</td></tr>
<tr><td colspan="4">Flujos alternos:</td></tr>
<tr><td>Identificador</td><td>Nombre</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>AG01</td><td>Salir del sistema</td><td colspan="2">El Usuario sale del sistema al elegir la opción “Salir”</td></tr>
<tr><td>AG02</td><td>Cambiar rol</td><td colspan="2">El sistema lo muestra la lista de roles permitidos para el Usuario, para ingresar al sistema</td></tr>
<tr><td>AE01</td><td>Datos incorrectos</td><td colspan="2">El sistema manda un mensaje de error indicándole al Usuario que el nombre de usuario o la contraseña son incorrectos</td></tr>
</table>

Tabla A.10 Detalle del caso de uso 5. Cerrar Sesión.

Pantalla “Salir”

Esta pantalla es la que se muestra cuando el usuario elige la opción “Salir”.

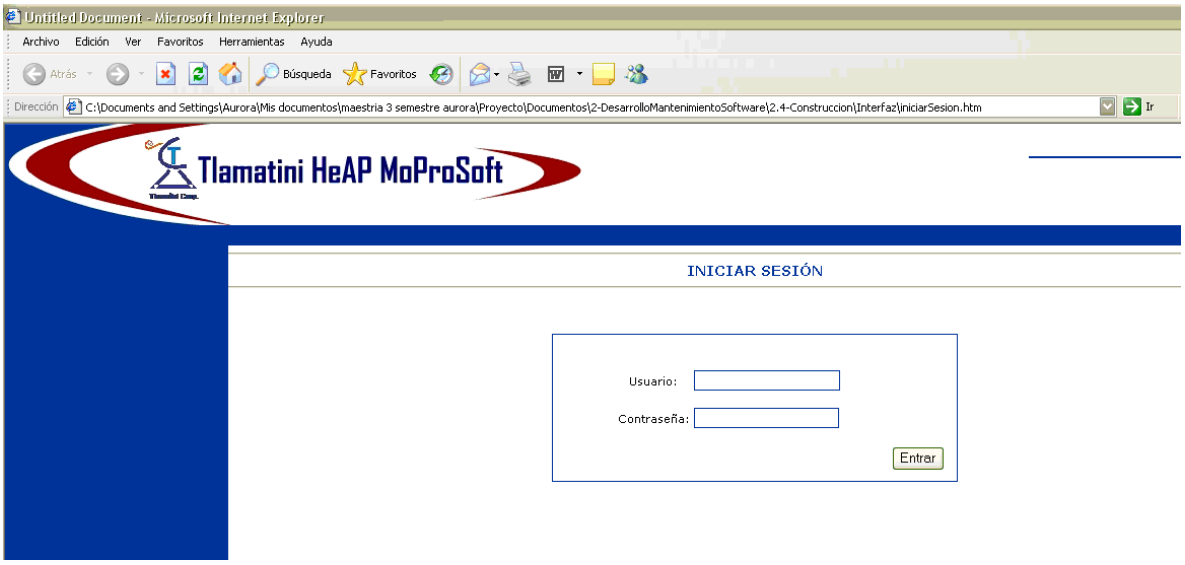


Figura A.20 Pantalla “Salir”

Pantalla “Seleccionar Rol”

Esta pantalla es la que se muestra cuando se elige la opción “Cambiar Rol”.

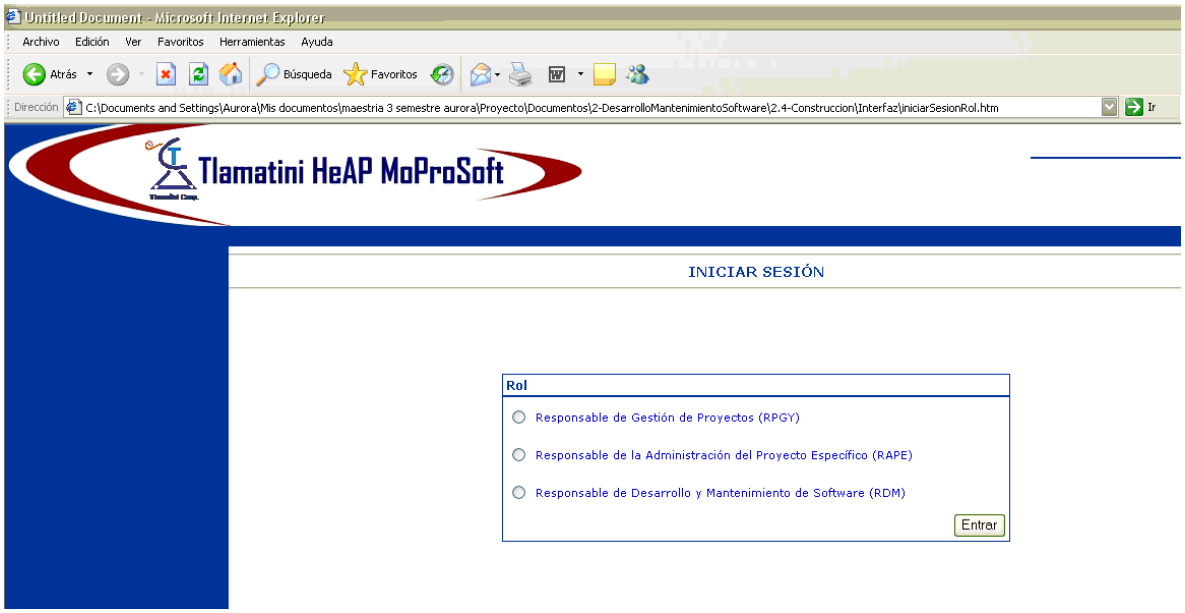


Figura A.21 Pantalla “Cambiar Rol”

Diagramas de clases del análisis
5. Cerrar Sesión

Diagrama de clases de interfaz

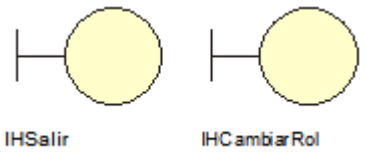



Diagrama de clases de control


Figura A.22 Diagrama de clases del análisis del caso de uso 5. Cerrar Sesión.

Diagramas de clases del diseño
5. Cerrar Sesión

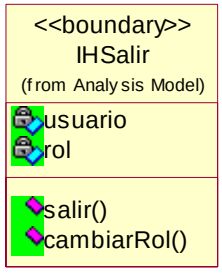
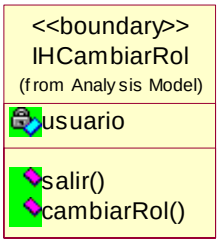
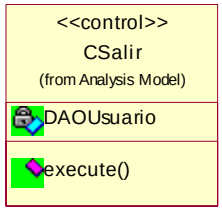
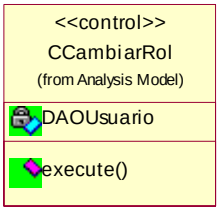
Iniciar Sesión	
Interfaz	
	
Control	
	

Figura A.23 Diagrama de clases del diseño del caso de uso 5. Cerrar Sesión.