UNIVERSIDAD AUSTRAL DE CHILE
SEDE PUERTO MONTT
ESCUELA DE INGENIERIA EN COMPUTACION

DESARROLLO DE SISTEMA DE ADMINISTRACIÓN DE DOCUMENTOS Y GESTIÓN DE CALIDAD UTILIZANDO PROGRAMACIÓN ORIENTADA A ASPECTOS (AOP)

Proyecto de Seminario
de Titulación para optar
al título de Ingeniero en Computación

PROFESOR PATROCINANTE:
Sra. Mónica Gallardo Vargas

LUISA CELIA HERNÁNDEZ ZÚÑIGA

PUERTO MONTT – CHILE
2009

Universidad Austral de Chile
Escuela de Ingeniería en Computación

Los Pinos s/n, Balneario Pelluco
Sede Puerto Montt
Casilla 1327 · Fono: 56 65 260990
Fax: 56 65 277156
Email: ecomputa@uach.cl
www.uach.cl

Puerto Montt, 26 de mayo de 2009

COMUNICACIÓN INTERNA N° 097/09

DE : Sra. Sandra Ruiz Aguilar
DIRECTORA ESCUELA DE INGENIERIA EN COMPUTACION

A : Dr. Renato Westermeier H. – **VICERRECTOR SEDE PUERTO MONTT**
Ms. César Pino Soto **– DIRECTOR ACADEMICO SEDE PUERTO MONTT**
Sra. Cristina Barriga – **JEFE DEPARTAMENTO REGISTRO ACADEMICO**
Sra. Alba Vásquez - **ENCARGADA DE TITULACIÓN SEDE PUERTO MONTT**

C.c : Srta. Luisa Hernández Zúñiga
Sra. Mónica Gallardo Vargas
Sra. Claudia Zil Bontes
Sra. Sandra Ruiz Aguilar

MOTIVO:

Informar a usted, las calificaciones obtenidas por la alumna de Ingeniería en Computación **Srrta. Luisa Celia Hernández Zúñiga** Rut 15.287.432-4, en su informe de Titulación "***Desarrollo de Sistema de Administración de Documentos y Gestión de Calidad Utilizando Programación Orientada a Aspectos (AOP)***"

Prof. Mónica Gallardo Vargas	6.2
Prof. Claudia Zil Bontes	6.3
Prof. Sandra Ruiz Aguilar	6.2
Promedio Seminario	**6.23**

Sin otro particular, le saluda atentamente,

UNIVERSIDAD AUSTRAL DE CHILE
ESCUELA DE INGENIERIA EN COMPUTACION
SEDE PUERTO MONTT

SANDRA RUIZ AGUILAR
DIRECTORA

SRA/mva

PUERTO MONTT, 26 mayo 2009

De : Sra. Mónica Gallardo Vargas
PROFESORA PATROCINANTE

A : Sra. Sandra Ruiz Aguilar
DIRECTORA ESCUELA INGENIERÍA EN COMPUTACIÓN

MOTIVO:

Informar a Usted la calificación obtenida por la alumna ***Luisa Celia Hernández Zúñiga*** en su Seminario de Titulación "**Desarrollo de Sistema de Administración de Documentos y Gestión de Calidad Utilizando Programación Orientada a Aspectos (AOP)**":

NOTA: 6,2

JUSTIFICACION:

Clara especificacion tanto del problema, como de la solucion. Sistema pertinente, original y usa nuevos paradigmas de programacion

OTRAS OBSERVACIONES:

Sra. Mónica Gallardo Vargas
PROFESORA PATROCINANTE

PUERTO MONTT, 26 de Mayo del 2009

De : Sra. Claudia Zil Bontes
PROFESORA INFORMANTE

A : Sra. Sandra Ruiz Aguilar
DIRECTORA ESCUELA INGENIERÍA EN COMPUTACIÓN

MOTIVO:

Informar a Usted la calificación obtenida por el alumno ***Luisa Celia Hernández Zúñiga*** en su Seminario de Titulación "**Desarrollo de Sistema de Administración de Documentos y Gestión de Calidad Utilizando Programación Orientada a Aspectos (AOP)**":

NOTA: 6,3

JUSTIFICACION:

Buen nivel de diseño con incorporación de nuevos paradigmas.
Documentación organizada en adecuada coordinación de actividades entre metodologías.

OTRAS OBSERVACIONES:

Ver observaciones al interior del documento.

Sra. Claudia Zil Bontes
PROFESORA INFORMANTE

PUERTO MONTT, 25 mayo 2009

De : Sra. Sandra Ruiz Aguilar
PROFESORA INFORMANTE

A : Sra. Sandra Ruiz Aguilar
DIRECTORA ESCUELA INGENIERÍA EN COMPUTACIÓN

MOTIVO:

Informar a Usted la calificación obtenida por el alumno ***Luisa Celia Hernández Zúñiga*** en su Seminario de Titulación "**Desarrollo de Sistema de Administración de Documentos y Gestión de Calidad Utilizando Programación Orientada a Aspectos (AOP)**":

NOTA: 6.2

JUSTIFICACION:

Buen desarrollo del tema.

OTRAS OBSERVACIONES:

Sra. Sandra Ruiz Aguilar
PROFESORA INFORMANTE

Dedicado a mis padres por su gran trabajo y sacrificio.

Agradecimientos

Gracias a mi familia por el apoyo en este proceso y por todo lo que me brindaron en toda la época de estudios, por todo sus esfuerzos y por toda su ayuda. A ti mamita que has sido siempre mi principal pilar, por tu ánimo, por tu paciencia y por el cariño de siempre.

Pero sobre todo gracias a mi querido padre, porque el destino quiso que te vayas cuando esto aún no terminaba, porque cada línea de este informe, cada noche, cada día y cada pedacito de esfuerzo esta hecho en nombre de tus sueños y de los míos. Te dedico mis sueños y mi trabajo interpretados en estas líneas, a modo de recompensa por dedicarme tu vida.

Un reconocimiento especial a mi jefe y gran apoyo en este seminario, Don Gustavo Montero Prieto gracias por su ayuda.

Por supuesto muchas gracias a mi profesora Mónica Gallardo por ser mi patrocinante y por su ayuda en todo este proceso.

Muchas gracias también a mis grandes amigos Rolando Martínez, Boris Ayancán y Paulina Ojeda por todo su apoyo y su compañía.

En fin gracias a todos y por todo….

ÍNDICE

Síntesis en castellano

Síntesis en inglés

1. Introducción……. 1

2. Objetivos…….. 4

2.1 Objetivos general 4

2.2. Objetivos específicos 4

3. Planteamiento del Problema 6

3.1. Antecedentes 6

3.1.1. Definición del problema 6

3.1.2. Identificación de esfuerzos anteriores 8

3.1.3. Definición de solución 8

3.1.4. Definición del Equipo de Trabajo.....18

3.2. Justificación 19

3.2.1. Situación sin proyecto 19

3.2.2. Situación con proyecto 19

3.3. Delimitación 20

4. Metodología…….....21

5. Recursos…….....24

5.1. Hardware. 24

5.2. Software……. 25

6. Fase de Inicio…….....26

6.1. Modelado de requerimientos de alto nivel, Esquemas de casos de uso de alto nivel.....27
6.2. Identificación de las entidades de negocio y sus relaciones de alto nivel…….....38
6.3 Glosario (Diccionario de datos).....41
6.4 Modelado de arquitectura de alto nivel 42
6.5 Pruebas…… 44
6.5.1 Planificación Inicial de las pruebas.....44
6.6 Levantamiento el entorno del trabajo y herramientas utilizadas .44
6.6.1 Plataforma de Desarrollo.....45
6.6.2 Herramientas de Desarrollo.....48
6.6.3 Elementos de la Programación orientada aspectos 51
7. Fase elaboración.....54
7.5 Modelado de arquitectura 54
7.5.1 Capa de Presentación.....54
7.5.2 Capa de Negocios.....58
7.5.3 Capa de Acceso a Datos.....59
7.6 Prototipo interfaces de usuario 61
8. Fase Construcción 75
8.1 Diagramas de secuencias.....75
8.2 Modelo de Diseño 116
8.2.1 Diagrama de Clases de diseño 116
8.3 Diseño de aspectos 121

8.3.1 Propuesta elegida de Suzuki y Yamamoto.....121
8.3.2 Consideraciones teóricas de Programación Orientada a aspectos……......122
8.3.3 Diagramas de aspectos.....123
8.4 Diseño de la base de datos.....141
8.4.1 Diseño Conceptual.....142
8.4.2 Diseño lógico de la base de datos para base de datos relacionales…….....163
8.4.2.1....Determinar las relaciones para el modelo lógico de los datos.....163
8.4.3 Diseño físico de la base de datos para base de datos relacionales…… 167
8.5 Construcción de la Aplicación.....169
8.6 Pruebas de Unidad.....182
9. Transición.....183
9.1 Pruebas de Aceptación.....183
9.2 Finalizar el paquete de despliegue.....183
10. Conclusiones y recomendaciones.....184
10.1 Conclusiones.....184
11. Bibliografía.....187
12. Anexos.....190
A . Instalación y uso de plugin aspectj en netbeans.....190
B. Pruebas de aceptación del sistema.....195

C. Levantamiento del sistema en ambiente de producción 199

D. Sintaxis de Aspectj.....203

Tablas

Tabla 1: Funcionalidades Módulo Maestro 12

Tabla 2: Funcionalidades Módulo Administración de Proyectos.....15

Tabla 3: Funcionalidades Módulo Control Activo.....17

Tabla 4: Equipo de Trabajo.....18

Tabla 5: Plataforma de Software.....25

Tabla 6: Actividades fase de inicio.....27

Tabla 7: Actores de casos de uso.....28

Tabla 8: Caso de Uso “Generación de un nuevo proyecto”. 30

Tabla 9: Caso de uso “Publicación nuevo documento mediante el sistema” 32

Tabla 10: Caso de uso “Acceso de documentos de biblioteca”. 34

Tabla 11: Caso de uso “Proceso de control activo”. 36

Tabla 12: Caso de uso “Ingreso y modificación de maestros del sistema”. ..38

Tabla 13: Diccionario de datos.....42

Tabla 14: Detalle modelo de clases.....120

Tabla 15: Entidades modelo conceptual.....144

Tabla 16: Relaciones modelo conceptual 147

Tabla 17: Entidad aprobación 147

Tabla 18: Entidad biblioteca.....148

Tabla 19: Entidad control 148

Tabla 20: Entidad correctiva 149
Tabla 21: Entidad detalle_control.....150
Tabla 22: Entidad documento 150
Tabla 23: Entidad apoyo_externo 151
Tabla 24: Entidad documento_version.....151
Tabla 25: Entidad email 152
Tabla 26: Entidad externo.....152
Tabla 27: Entidad mae_area.....153
Tabla 28: Entidad mae_cargo 153
Tabla 29: Entidad mae_centro 154
Tabla 30: Entidad mae_departamento.....154
Tabla 31: Entidad mae_documento_ familia.....155
Tabla 32: Entidad mae_documento_tipo 155
Tabla 33: Entidad mae_etapa.....156
Tabla 34: Entidad mae_perfil 156
Tabla 35: Entidad mae_proyecto_tipo 157
Tabla 36: Entidad mae_tipo_variable.....157
Tabla 37: Entidad mae_usuario 158
Tabla 38: Entidad mae_variable 159
Tabla 39: Entidad mae_modulo 159
Tabla 40: Entidad mae_pagina 159
Tabla 41: Entidad planificacion 160
Tabla 42: Entidad proyecto 160

Tabla 43: Entidad resultado_control 161
Tabla 44: Entidad tarea.....161
Tabla 45: Tipos de Puntos de corte 206

Figuras

Figura 1: Resumen de las funcionalidades del Sistema 9
Figura 2: Caso de uso “Generación de un nuevo proyecto”. 29
Figura 3 : Caso de uso “Publicación de un nuevo documento mediante el sistema”. 31
Figura 4: Caso de uso “Acceso de documentos de biblioteca”. 33
Figura 5: Caso de uso “Proceso de control activo”. 35
Figura 6: Caso de uso “Ingreso y modificación de maestros del sistema”. ...37
Figura 7: Diagrama inicial de dominio. 40
Figura 8: Diagrama de arquitectura de alto nivel. 43
Figura 9: Esquema de clases sin aspectos.....51
Figura 10 : Esquema de utilización de aspectos.....52
Figura 11: Funcionamiento del patrón DTO para transporte de un dato.....61
Figura 12: Esquema general de interfaz de usuario 62
Figura 13: Interfaz de bienvenida al sistema 63
Figura 14: Interfaz de usuario que permite generar nuevos proyectos.....64
Figura 15: Interfaz de usuario que permite ver los documentos existentes en biblioteca.....65

Figura 16: Interfaz de usuario que permite agregar nuevos “documentos a crear” a un proyecto existente.....66
Figura 17: Interfaz de usuario que permite agregar nuevas versiones a un documento existente.....67
Figura 18: Interfaz de usuario que permite asignar usuarios a un grupo de trabajo para un proyecto existente.....68
Figura 19: Interfaz de usuario que permite asignar documentos de apoyo a un proyecto existente.....69
Figura 20: Interfaz de usuario que permite definir tareas a los usuarios pertenecientes al proyecto.....70
Figura 21: Interfaz de usuario que permite modificar o eliminar documentos existentes en biblioteca.....71
Figura 22: Interfaz de usuario que permite crear un nuevo control.....72
Figura 23: Interfaz de usuario que permite planificar un control existente....73
Figura 24: Interfaz de usuario que permite ingresar nuevas áreas al sistema 74
Figura 25: Diagrama de secuencia de ingresar un nuevo proyecto.....76
Figura 26: Diagrama de secuencia de ingresar etapas a un proyecto.....77
Figura 27: Diagrama de secuencia ingresar grupo de trabajo 78
Figura 28: Diagrama de secuencia de ingresar tareas 80
Figura 29: Diagrama de secuencia de ingresar documentos a crear.....81
Figura 30: Diagrama de secuencia de documentos de apoyo 82

Figura 31: Diagrama de secuencia de ingreso de una nueva versión de documento 84
Figura 32: Diagrama de secuencia de ingreso de aprobación de un documento 86
Figura 33: Diagrama de secuencia de acceso a documentos en biblioteca .88
Figura 34: Diagrama de secuencia ingreso de documentos biblioteca……89
Figura 35: Diagrama de secuencia de eliminación de documentos en biblioteca.....91
Figura 36: Diagrama de secuencia de modificación de documentos en biblioteca.....93
Figura 37: Diagrama de secuencia de ingreso de un nuevo control 95
Figura 38: Diagrama de secuencia de ingreso de una nueva planificación ..96
Figura 39: Diagrama de secuencia de ingreso de resultados a una planificación 98
Figura 40: Diagrama de secuencia de ingreso de resultados a una planificación 100
Figura 41: Diagrama de secuencia de ingreso de resultados a una acción correctiva 101
Figura 42: Diagrama de secuencia de ingreso áreas.....103
Figura 43: Diagrama de secuencia de ingreso usuarios 104
Figura 44: Diagrama de secuencia de ingreso de departamentos.....105
Figura 45: Diagrama de secuencia de ingreso de tipos de proyectos.....106
Figura 46: Diagrama de secuencia de ingreso de perfiles de usuario 107

Figura 47: Diagrama de secuencia de ingreso de permisos de perfiles a páginas 109
Figura 48: Diagrama de secuencia de ingreso de centros de cultivo.....110
Figura 49: Diagrama de secuencia de ingreso de cargos.....111
Figura 50: Diagrama de secuencia de ingreso de tipos de variables.....112
Figura 51: Diagrama de secuencia de ingreso de variables 114
Figura 52: Diagrama de Clases de Software de la capa de negocios.....117
Figura 53: Aspectos para manejo de transacciones 125
Figura 54: Diagrama de Clases con el AspectoTransaccionProyecto 126
Figura 55: Diagrama de Clases con el AspectoTransaccionAprobacion 127
Figura 56: Diagrama de Clases con el AspectoTransaccionControl.....128
Figura 57: Diagrama de Clases con el AspectoTransaccionResultados.....129
Figura 58: Diagrama de Clases con el AspectoTransaccionCorrectivas 130
Figura 59: Diagrama de Clases con el Aspecto Notificación 131
Figura 60: Diagrama de Clases con el Aspecto Autentificación.....133
Figura 61: Diagrama de Clases con el Aspecto Autorización 134
Figura 62: Diagrama básico de un aspecto utilizando un entretejido que genera una nueva clase.....135
Figura 63: Diagrama de Aspectos Notificación a Clase Base Tarea.....136
Figura 64: Diagrama de Aspectos Notificación a Clase Base Grupo.....136
Figura 65: Diagrama de Aspectos Notificación a Clase Base Externo.....136
Figura 66: Diagrama de Aspectos Notificación a Clase Base Versión.....137

Figura 67: Diagrama de Aspectos AspectoTransaccionProyecto a Clase Base Proyecto.....137
Figura 68: Diagrama de Aspectos AspectoTransaccionAprobacion a Clase Base Aprobacion.....138
Figura 69: Diagrama de Aspectos AspectoTransaccionControl a Clase Base Control 138
Figura 70: Diagrama de Aspectos AspectoTransaccionResultados a Clase Base Resultado.....139
Figura 71: Diagrama de Aspectos AspectoTransaccionCorrectivas a Clase Base Correctiva 139
Figura 72: Diagrama de Aspectos Autentificación a Clase Base ing_area .140
Figura 73: Diagrama de Aspectos Autorización a Clase Base ing_area.....140
Figura 74: Modelo conceptual de la base de datos.....162
Figura 75: Relación “tieneEtapa” 163
Figura 76: Tablas intermedia entre proyecto y mae_etapa.....164
Figura 77: Relación “tieneEtapa” 164
Figura 78: Tabla intermedia entre mae_peril y página.....164
Figura 79: Relación tipo superclase/subclase.....165
Figura 80: Relación compleja “grupo_trabajo” 166
Figura 81: Modelo físico de la base de datos 168
Figura 82: Ejecución de pruebas creadas para la capa de negocios.....182
Figura 83: Carga de plugin en Netbeans 191
Figura 84: Creación de un nuevo aspecto 192

Figura 85: Activar utilización de AJDE 193
Figura 86: Resultados compilación Aspectj 194
Figura 87: Página inicial de Apache Tomcat.....200
Figura 88: Página gestor de aplicaciones Tomcat 201
Figura 89: Pantalla donde se selecciona el archivo WAR.....202

SÍNTESIS

El presente seminario tiene como finalidad el diseño, análisis e implementación de un sistema Web que permita en resumidas palabras, generar nueva documentación a una empresa, administrar la documentación existente y registrar los controles que se hacen para verificar si los procesos se efectúan según dice la documentación formal, todo esto principalmente orientado a la industria acuícola.

Para el desarrollo del sistema se utiliza como apoyo la metodología RUP Agile y para la creación de la base de datos la metodología Connolly.

Uno de los elementos técnicos más importantes del seminario es incluir en la solución del problema el paradigma de programación orientada a aspectos (AOP), lo que permitirá llevar a la práctica en un problema real los alcances del paradigma y sus herramientas.

Una vez terminado el proyecto se puede contar con un sistema que apoya a todos los procesos productivos con la documentación necesaria actualizada y con un registro de sus controles de calidad efectuados, permitiendo entonces mejorar los procesos, evitar multas y registrar responsables de mejora continúa.

ABSTRACT

The current seminar has as its main objectives the design, analysis and implementation of a web system that allows a company to produce new data, to manage the existing protocols and to record the controls that are carried out to check if the processes are made according to the existing data. All this work is oriented towards the aquaculture field.

To carry out this system, the support of RUP AGILE methodology is needed and for the creation of the database the CONNOLLY methodology.

One of the most important technical elements of the seminar is to include, in solving the problem, the paradigm of programming oriented on aspects (AOP), this will allow to take into practice, in a real problem, the implications of the paradigm and its tools.

When the project is finished, the result is a supporting system for all the productive processes with the necessary updated information and with a record of the quality controls made, taking to improve the processes, avoid legal fines, register responsible people for the continuous improvement.

1. Introducción

Gran porcentaje de los productos que pertenecen al área salmonera y sus derivados son exportados a Asia y Europa. Son justamente las altas exigencias de estos clientes para importar productos alimenticios que obligan a la industria a demostrar que se produce un bien seguro y de calidad.

Para demostrar que realmente se cumplen las condiciones exigidas por los clientes se hace necesario que las empresas se certifiquen con algunas de las normas internacionales existentes como son ISO, IEC, SIGES.

En este marco y con la finalidad de contar con una herramienta para la industria y de acuerdo a las normas legislativas y de mercado de los productos salmonídeos, ha nacido el Sistema Integrado de Gestión (SIGES). Este sistema es un conjunto de herramientas diseñadas por INTESAL (Instituto Tecnológico del Salmón) que consiste principalmente en conductas y estándares verificables, que permiten incorporar sistemas y estándares de calidad, gestión ambiental y seguridad ocupacional de amplio reconocimiento nacional o internacional [INTESAL2004].

Surge la necesidad, entonces, de las empresas de cumplir con los procesos y exigencias que imponen las normas SIGES (o cualquier otro estándar) previo al

paso definitivo de la certificación, lo que conlleva a la necesidad de herramientas informáticas que apoye las labores de elaboración y distribución de la documentación necesaria, así como el registro de la verificación de que los procesos se realizan según están establecidos.

Este seminario de tesis contempla, entonces, la definición y construcción de una herramienta informática que permita servir de apoyo al proceso de certificación permitiendo principalmente la elaboración y administración de documentación y control de calidad de procesos.

En el capítulo 2 se pueden ver el objetivo general y los objetivos específicos, mientras que en capítulo 3 se presentará el planteamiento del problema que se requiere resolver indicando los esfuerzos anteriores y la definición de la solución. En el capítulo 4 se mostrará la metodología que sirve de guía para llevar a cabo el proyecto, indicando una breve descripción de sus fases.

Posteriormente se podrá ver, en el capítulo 5, los recursos ocupados para el desarrollo del proyecto.

A continuación se podrá ver el desarrollo de las actividades de acuerdo a la metodología escogida, estos capítulos 6, 7, 8, y 9 corresponden a la fase de inicio, elaboración, construcción y transición respectivamente.

Finalmente, en el capítulo 10 se pueden ver las conclusiones obtenidas del trabajo de tesis y algunas recomendaciones para trabajos futuros.

2. Objetivos

2.1. Objetivo general

Desarrollar una aplicación Web utilizando el paradigma de programación orientada a aspectos (AOP), que permita principalmente administrar documentación, crear nueva documentación y controlar procesos productivos de la industria acuícola.

2.2. Objetivos específicos

Desde el punto de vista funcional se persiguen los siguientes objetivos:

- Desarrollar un módulo que permita apoyar el proceso de creación de nueva documentación, administrando quienes participarán en la redacción, sus tareas, plazos de entrega y finalmente la aprobación de los nuevos contenidos.
- Desarrollar un módulo que permita centralizar todos los documentos de una compañía, tanto los creados por entidades externas como los generados con la ayuda del propio sistema.
- Desarrollar un módulo que permita crear checklists de verificación de procesos, planificar visitas a terreno, almacenar los resultados de

dichas visitas y las acciones correctivas si se estiman necesarias.

Desde el punto de vista del desarrollo los objetivos son:

- Incorporar la programación orientada a aspectos en las etapas de diseño e implementación, para codificar en forma separada los conceptos comunes de toda la aplicación y obtener así, un código ordenado y fácil de cambiar.

3. Planteamiento del Problema

3.1. Antecedentes

3.1.1. Definición del problema

Para dar cumplimientos a los estándares de calidad se necesita aplicar dentro de la compañía cambios en los procesos, acciones de autodiagnóstico, capacitación y verificación que permiten pasar con éxito las auditorias y lograr la etapa de certificación. En este contexto surge la necesidad de mantener a lo largo de toda la línea de producción la información que permite llevar a cabo los procesos según indican las normas. Es decir, procedimientos, manuales de buenas prácticas, instructivos, etc. deben estar siempre disponibles a toda persona que participe en el proceso productivo, siendo muy importante en este punto la vigencia de esta información. Otra razón que motiva la centralización de información es que entidades externas, como el servicio nacional de pesca, realizan auditorias a los centros de cultivos para verificar que estos tengan la última versión de los documentos, arriesgando multas sino cumplen con este requerimiento.

Actualmente en la empresa acuícola donde se aborda la problemática no existe un sistema informático que apoye en la centralización y orden de documentos

por lo que existe un riesgo constante que la información con que cuentan los usuarios no esté actualizada, no sea íntegra o simplemente no la tengan.

Por otro lado, la generación de la nueva documentación es también un punto muy importante en una industria donde existen muchos procesos críticos y donde estos cambian constantemente. Tampoco existe un sistema informático que apoye este proceso en la empresa estudiada, es decir, la generación de un nuevo manual o instructivo se hace enviándose la información por correo electrónico sin un control de responsabilidades ni tampoco de versiones, lo que lleva a que generalmente el proceso de creación de un documento no llegue a término.

Finalmente, para mantener un proceso de mejora continua se realizan auditorias internas que permiten constatar en la práctica que los procesos establecidos en la documentación se hagan correctamente. Para el manejo de estas auditorias, sus resultados y sus acciones correctivas asociadas es importante contar con un sistema informático que apoye en la labor de planificación, mantención de la información y búsqueda histórica de las visitas a centros de cultivo.

3.1.2. Identificación de esfuerzos anteriores

En la actualidad existen varios softwares, tanto gratuitos como pagados, que permiten administrar proyectos y su documentación, por ejemplo dotproject (http://www.dotproject.net/) y scramject (http://scramject.net/portal/). Pese a ser aplicaciones muy completas en el ámbito de manejo de proyectos, no cumplen completamente con los requerimientos del cliente puesto que este necesita una herramienta que comprenda también la generación interna de documentos (mediante grupos de trabajo, niveles de aprobación, etc) y que ayude al control de los prácticas sobre los procesos al interior de la empresa.

3.1.3. Definición de solución

Ante la problemática antes planteada la alumna propone como proyecto de tesis el diseño lógico, físico e implementación de un sistema informático que sea accesible a todos los actores que participan en los procesos productivos y que les sirva de apoyo para mejorar las prácticas al interior de la cadena de valor y a la vez que permita generar nuevos documentos para contribuir así a un proceso de mejora continua. El sistema debe también apoyar a quienes se encargan del autodiagnóstico de las prácticas, permitiendo registrar las visitas y

controles, además de los responsables e irregularidades encontradas en las auditorias.

A modo de resumen se presenta a continuación un esquema que permite ver la idea básica del sistema:

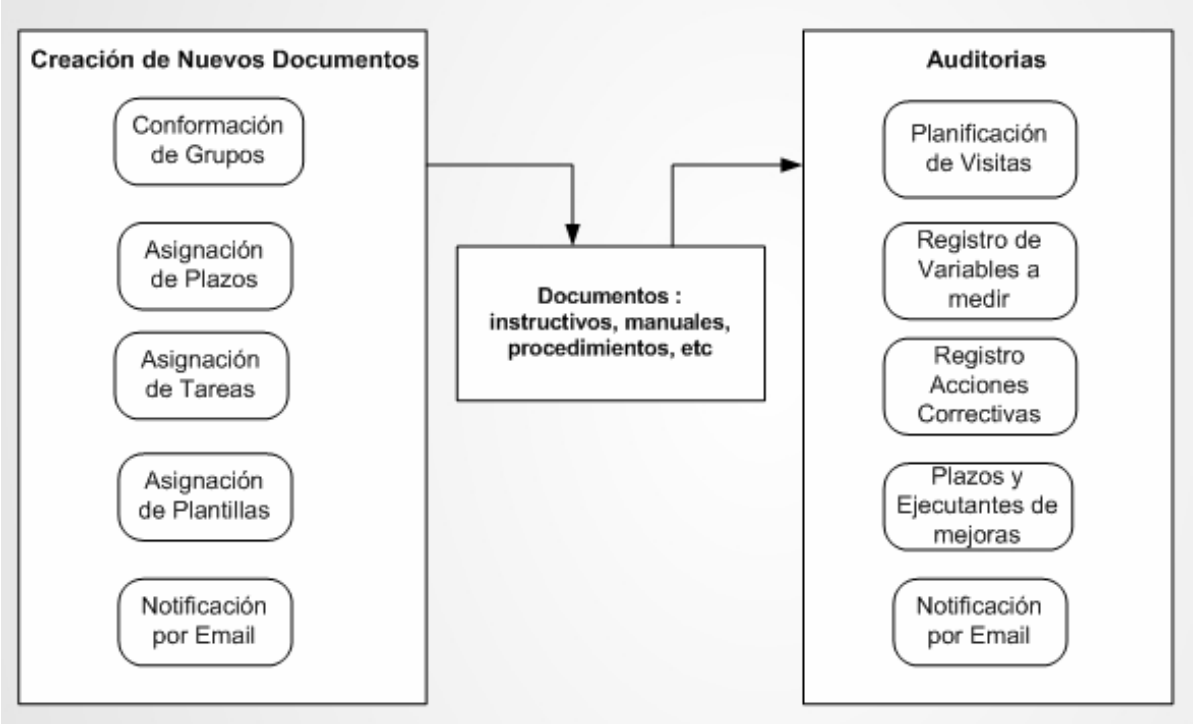


Figura 1: Resumen de las funcionalidades del Sistema

El Sistema se pretende implementar sobre una plataforma Web, de esta manera quedará disponible a todos los usuarios sin límite de ubicación

geográfica. Además la alumna pretende utilizar para la implementación del sistema el nuevo paradigma denominado Programación Orientada a Aspectos (AOP).

La programación orientada a aspectos (AOP) es un paradigma de programación que persigue la modularización fomentando la separación de incumbencias (Separation of Concerns), en especial los que afectan a otros conceptos de la aplicación [Kiczales 1997].

El principio de separación de incumbencias fue identificado en la década de 1970, y plantea que un problema dado involucra varias incumbencias que deben ser identificadas y separadas.

Las incumbencias son los diferentes temas o asuntos de los que es necesario ocuparse para resolver el problema. Una de ellas es la función específica que debe realizar una aplicación, pero también surgen otras como por ejemplo distribución, persistencia, replicación, sincronización, etc. Separando las incumbencias, se disminuye la complejidad a la hora de tratarlas y se puede cumplir con requerimientos relacionados con la calidad como adaptabilidad, mantenibilidad, extensibilidad y reusabilidad [Kicillof 2005].

Los tres elementos constitutivos principales de la AOP son [Asteasuain 2002]:

- Un lenguaje para definir la funcionalidad básica, conocido como lenguaje base o componente. El mismo puede ser un lenguaje imperativo, o no, como por ejemplo C#, Java, C++.
- Uno o varios lenguajes de aspectos, para especificar el comportamiento de los distintos aspectos. Algunos ejemplos son Aspectj, Aspect#, JBossAOP.
- Un tejedor de aspectos, del inglés weaver, que se encarga de combinar los lenguajes en tiempo de ejecución o de compilación.

Para el desarrollo del sistema la alumna estudiará estos elementos y los llevará a la práctica en un problema real, por lo que se pueden llegar a conclusiones objetivas acerca de las herramientas existentes para implementar el paradigma, así como ventajas y desventajas de su utilización.

Desde el punto de vista funcional el sistema se agrupa en módulos los cuales se nombran a continuación:

3.1.3.1. Módulo Administrador de Maestros

El objetivo de este módulo es administrar el conjunto de maestros que permitirán el ingreso de variables estandarizadas que serán utilizadas por los

demás módulos del sistema, de manera de reducir el ingreso erróneo de datos y normalizar las relaciones al interior del sistema.

A continuación se enumeran las aplicaciones que formarán parte de este módulo:

Módulo	**Aplicación**
Maestros	Maestro de usuarios
	Maestro de familias de documentos
	Módulo de administración de perfiles y permisos
	Maestro de tipos de documento
	Maestro de departamentos
	Maestro de Centros de Cultivo
	Maestro de Variables de Control

Tabla 1: Funcionalidades Módulo Maestro

3.1.3.2. Módulo Administración de Documentos

El objetivo de este módulo será por una parte administrar el proceso de generación de documentos al interior de la compañía, y su posterior manejo.

Esto es, permitirá generar un proyecto de elaboración, definiendo grupos de trabajo, tareas y plazos. Además de ello el sistema llevará un registro de las modificaciones y de la evolución de los borradores elaborados, además de implementar un lugar común de discusión y análisis (foro).

Finalmente este módulo administrará el proceso de aprobación del documento final, su publicación a los usuarios externos al grupo de trabajo, y la administración del proceso de entrega e inducción del procedimiento tratado en el documento elaborado.

Con esto el sistema ayudará a una administración centralizada de documentos, permitiendo su clasificación, publicación (con o si restricciones) y divulgación. Por otra parte este mismo módulo permitirá generar proyectos de mejora continua al interior de la compañía, generando los respectivos grupos de trabajos, tareas y plazos, además de guardar los documentos elaborados durante el proceso de diseño e implementación de la mejora.

Módulo	**Aplicación**
Administración de Documentos	Generación nuevo proyecto.
	Ingreso datos generales proyecto.
	Modificar o borrar datos generales.

	Asignación de usuarios que participarán el en proyecto.
	Eliminar usuarios asignados.
	Definición de tareas.
	Modificación y eliminación de tareas.
	Asignación de tareas a participantes.
	Modificar asignaciones.
	Definición de plazos.
	Modificación de plazos ingresados.
	Registro de término de tareas.
	E-mail de notificación de tareas asignadas y recordatorio.
	Eliminar modificación (elimina documento con modificación).
	Foro de discusión y registro de comentarios.
	Registro de comentarios.
	Modificación de comentarios.
	Eliminar comentarios.
	Registro de documentos.
	Subir Documento al sistema (públicos,

	restringidos, borradores, etc.)
	Eliminar documento del sistema.
	Asignación de documentos a proyectos.
	Modificación de asignación de documentos a Proyectos.
	E-mail de notificación de documentos registrados
	Aprobación y liberación de documentos.
	Informe de estado de proyecto.
	Carta Gantt Proyecto.
	Lista de comentarios por proyecto.

Tabla 2: Funcionalidades Módulo Administración de Proyectos

3.1.3.3. Módulo de Control Activo

El objetivo de este módulo será administrar los procesos de fiscalización que se lleven al interior de la empresa, de manera de controlar que los procesos estandarizados en procedimientos, protocolos, instructivos, etc., se estén llevando acabo según se encuentra establecido. Además de ello permitirá

registrar un conjunto de acciones correctivas y sus respectivos responsables, de manera de controlar el proceso de corrección de los problemas encontrados. Este módulo también permitirá registrar los controles de la documentación que debe estar en los centros de cultivos, ya sea por estipulaciones legales o por simple decisión de la compañía.

Módulo	**Aplicación**
Control Activo	Generación de controles de procedimiento
	Generación de nuevo control de procedimiento y documentación
	Modificación de control de procedimiento ya generado
	Definición de variable o documentos a controlar (CheckList)
	Modificación de las variable definidas
	Calendarización de controles.
	Generación de planificación de controles por centro.
	Cambio de fechas y de tipos de control calendarizados

	Impresión de formularios de control (CheckList)
	Registro de variables a controlar.
	Registro de solicitud de acciones correctivas.
	Modificaciones de solicitudes de acciones correctivas.
	Definición de plazos para la aplicación de acciones correctivas.
	Modificación de plazos definidos.
	Asignación de responsables.
	Modificación de responsables.
	Registro de acciones completadas.
	Modificación de registro de acciones completadas.
	Informe de resultado de visitas y controles.
	Informe histórico por centro.
	Informe de acciones correctivas por responsable.

Tabla 3: Funcionalidades Módulo Control Activo

3.1.4. Definición del Equipo de Trabajo

El desarrollo del sistema se llevará a cabo con el siguiente grupo de trabajo:

Equipo de Trabajo	
Nombre	**Función**
Gustavo Montero Prieto	Comunicación con usuarios finales, toma de requerimientos, revisión de las funcionalidades del sistema y de la documentación.
Elvira Montero Prieto	Apoyo en el diseño de interfaces gráficas.
Mónica Gallardo Vargas	Apoyo en la revisión general del sistema y de la documentación.
Luisa Hernández Zúñiga	Diseño lógico, diseño físico e implementación.

Tabla 4: Equipo de Trabajo

Como se ve en la tabla el Sr. Gustavo Montero Prieto es el encargado de la comunicación con los usuarios finales. Es decir, él comunica los requisitos del sistema a la alumna, quien es la encargada de modelarlos e implementarlos.

3.2. Justificación

3.2.1. Situación sin proyecto

La situación sin proyecto es lo que se vive actualmente en la empresa estudiada, es decir, no se lleva un orden de la documentación existente, ni se permite asignar responsables a actores de toda la línea de producción para la generación de nueva documentación. Además al no existir una fuente de documentación centralizada se produce un gran problema con las versiones existentes y con las auditorias de entidades externas que exige que la documentación esté disponible.

Se produce entonces un gran riesgo de no cumplir con las normas que exigen los estándares nacionales e internacionales y perder así clientes que exigen el cumplimiento de dichos estándares

3.2.2. Situación con proyecto

La implementación del sistema permitirá a algunos responsables del área técnica, publicar documentos existentes y aprobados previamente, como también coordinar la creación de nuevos documentos manteniendo el orden y el

control sobre este proceso. Por otro lado se mantendrá el control en el proceso de auditorias a departamentos y procesos de la compañía, para asegurar que realmente las variables claves de los procesos se hacen según está establecido en los documentos. Esto llevará a mejorar las prácticas en la empresa y finalmente lograr cumplir las exigencias de los clientes y de la industria en general.

3.3. Delimitación

La principal limitante que puede tener la utilización del sistema es el uso de éste en zonas apartadas por las dificultades con la conexión a Internet que existe en esos lugares. Estas zonas son principalmente centros de cultivo que muchas veces no tienen conexión a Internet o bien ésta es muy lenta.

4. Metodología

La metodología elegida para desarrollar el sistema es RUP Agile (AUP). Esta metodología es una versión simplificada de Rational Unified Process (RUP) y se describe bibliográficamente como un procedimiento sencillo, fácil de entender para el desarrollo de software de aplicación comercial ágil utilizando las técnicas y los conceptos que aún permanecen fieles a RUP. En este enfoque se aplican técnicas ágiles que incluyen las pruebas impulsadas por el desarrollo (TDD) y Agile Modelo Driven Development (AMDD) [Ambler2005].

RUP Agile (AUP) está compuesta por 4 grandes fases, y su vez estas tienen un conjunto de disciplinas que se llevan a cabo en forma iterativa. En el presente seminario se documentarán sólo las tareas principales de la metodología dejando de lado las que no se consideren de gran relevancia para el entendimiento del proyecto. Otro punto relevante en las actividades documentadas es que se agregarán algunas que tienen relación con la inclusión de la programación orientada a aspectos (AOP) dentro del proyecto, esto es porque ninguna metodología de desarrollo de software incluye formalmente dentro de sus actividades el análisis e implementación de este paradigma.

A continuación se explica brevemente lo que se realizará en cada fase:

- Inicio: El objetivo de esta fase es obtener una comprensión común del alcance del nuevo sistema y definir los requerimientos realizando principalmente diagramas de caso de uso y diagramas de dominio de alto nivel.
- Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- Construcción: Se realizaran los modelos de clases de software, la base de datos y se trabajar ininterrumpidamente en el desarrollo y pruebas del sistema.
- Transición: el sistema se llevará a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

Como se dijo anteriormente a la metodología se le deben agregar algunas actividades para utilizar programación orientada a aspectos (AOP), para ello se estudiarán algunas propuestas que permitan tener en cuenta la influencia de los aspectos en los sistemas en las etapas tempranas del desarrollo de software. Estas propuestas se basan principalmente en extensiones a las técnicas de modelado para sistemas orientados a objetos, centrándose especialmente en el lenguaje de modelado unificado (UML).

Algunas de estas propuestas son:

- Propuesta de Suzuki y Yamamoto [Suzuki1999]
- Propuesta de Herrero [HERRERO 2000]
- Propuesta de Barra, Genova y Llorens [Barra2004]
- Propuesta de Bustos y Eterovic [Bustos2007]
- Propuesta de Basch y Sanchez [Bash2003]

Para el desarrollo del sistema la alumna estudiará las propuestas nombradas anteriormente y aplicará la que considere más apropiada de realizar tomando en cuenta la metodología elegida, el tipo y el tamaño de la problemática o cualquier otro factor que influya en la solución del problema y al estudio del paradigma.

5. Recursos

5.1 Hardware

El Hardware utilizado para el desarrollo del sistema será el siguiente:

Equipo desarrollo:

Portátil Compaq, AMD Turion 64 bits, HD 120 GB, 1,5 Gb RAM, Sistema Operativo Microsoft Windows Professional.

Equipo de Pre-Producción:

PC Dell, Intel Core 2 Duo, HD 80 GB, 2 Gb RAM, Sistema Operativo Microsoft Windows 2003 Server.

5.2 Software

La plataforma de software que se utilizará en el desarrollo del sistema es el siguiente:

Software	Funcionalidad
MySql Server	Gestor de Base de Datos
Navicat Mysql	Cliente para Myslq Server
Mysql Administrator	Administrador para Mysql Server
Mysql Query Browser	Cliente para Myslq Server
JDK	Kit de Desarrollo Java
Netbeans 6.0.1	Interfaz de desarrollo para Java
Apache Tomcat	Servidor de Aplicaciones Java
Power Designer	Diseñador Base de datos
Microlap Desginer for Mysql	Diseñador Base de datos
Microsoft Word	Editor de texto

Tabla 5: Plataforma de Software

6. Fase de Inicio

El propósito de esta fase es establecer una visión común inicial de los objetivos del proyecto, determinar si es viable y decidir si merece la pena llevar acabo algunas investigaciones serias en la fase de elaboración [Larman 2002]. Hay algunas actividades que comienzan en esta fase pero su término es en las siguientes como son los “Prototipos interfaces de usuario” o bien actividades genéricas que se llevan a cabo en todas las fases de la metodología como son las relacionadas con al gestión del proyecto. Entonces esta fase es muy breve, se detallan las siguientes actividades que la alumna documentará en el presente seminario:

Disciplina	Actividades
Modelado	Modelado de requerimientos de alto nivel - Casos de Uso de Alto nivel
	Identificar las entidades de negocio y sus relaciones. (sólo los nombres)
	Hacer un diccionario definiendo los elementos importantes tanto del negocio como técnicos
	Modelado de arquitectura de alto nivel
Entorno	Levantar el entorno del trabajo (Instalar herramientas necesarias, equipos, etc.)

Tabla 6: Actividades fase de inicio

6.1 Modelado de requerimientos de alto nivel, Esquemas de casos de uso de alto nivel

Antes de trabajar en los casos de uso se elabora una lista denominada actor-objetivo, que permite recoger los actores principales y los objetivos de cada uno de ellos [Larman 2002].

Actor	Objetivo
Jefe de proyecto	Llevar a cabo las tareas relacionadas con la administración de proyectos.
Usuario creador de contenidos	Incorporar contenido a un documento nuevo.
Usuario aprobador de documentos	Aprobar documentos para su posterior publicación.
Usuario común	Ver documentos de biblioteca y tareas básicas.
Administrador de documentos	Administrar los documentos existentes en la biblioteca.
Usuario encargado de control de procesos	Administrar las tareas relacionadas con el control de los procesos, esto incluye crear controles, planificar controles e ingresar acciones correctivas.
Administrador	Administrar los maestros del sistema.

Tabla 7: Actores de casos de uso

Luego de definir los actores que se ven involucrados en los procesos, se elaboran uno a uno los casos de uso esenciales que permiten analizar los principales requisitos funcionales del sistema.

6.1.1 Generación de nuevo proyecto

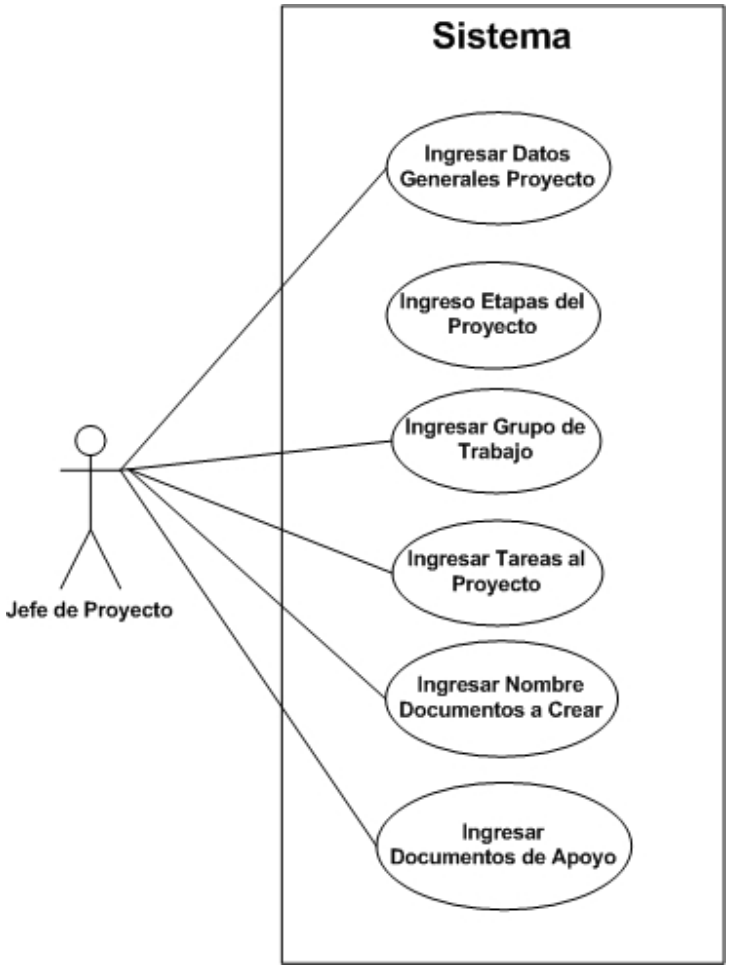


Figura 2: Caso de uso “Generación de un nuevo proyecto”.

Caso de Uso	Generación de nuevo proyecto
Actores	Jefe de proyecto
Tipo	Primario
Descripción	Cuando un usuario de sistema decide tomar el cargo de "Jefe de Proyecto" y generar un nuevo proyecto de documentación, se hará cargo de ingresar el nuevo proyecto al sistema junto a sus etapas, posteriormente decidirá quienes participaran en este proyecto y cuales son las tareas que llevarán a cabo. Finalmente ingresará y subirá documentos que sirvan de apoyo a la realización del proyecto y también debe ingresar los nombres de los documentos que se generarán como producto final.

Tabla 8: Caso de Uso "Generación de un nuevo proyecto".

6.1.2 Publicación de un nuevo documento mediante el sistema

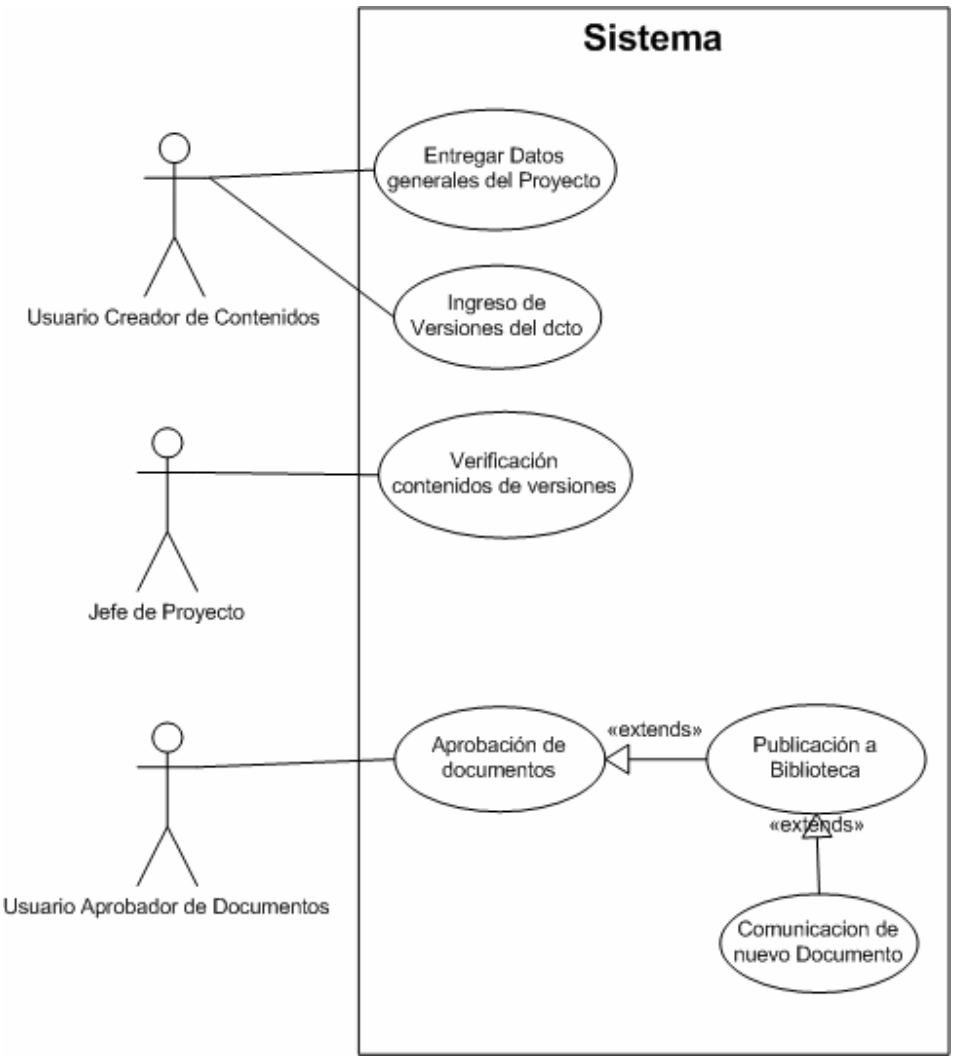


Figura 3 : Caso de uso “Publicación de un nuevo documento mediante el sistema”.

Caso de Uso	Publicación de un nuevo documento mediante el sistema
Actores	Usuario creador de documentos, Jefe de proyecto, usuario aprobador documentos
Tipo	Primario
Descripción	Cuando un usuario es encargado y comunicado que es parte de un proyecto como creador de contenidos, debe verificar los datos del proyecto en el sistema e informarse acerca de las tareas que debe cumplir. Luego dependiendo de la planificación de proyecto el usuario subirá al sistema las versiones de él o los documentos que correspondan, convirtiéndose en un proceso iterativo hasta que el jefe de proyecto determine que el documento está terminado. Posteriormente él o los usuarios que también pertenecen al grupo de trabajo con el cargo de aprobador, determinan que el documento está listo y por lo tanto el sistema lo publica a la biblioteca y comunica a los usuarios que pueden acceder a él.

Tabla 9: Caso de uso “Publicación de un nuevo documento mediante el sistema”.

6.1.3 Acceso a documentos de biblioteca

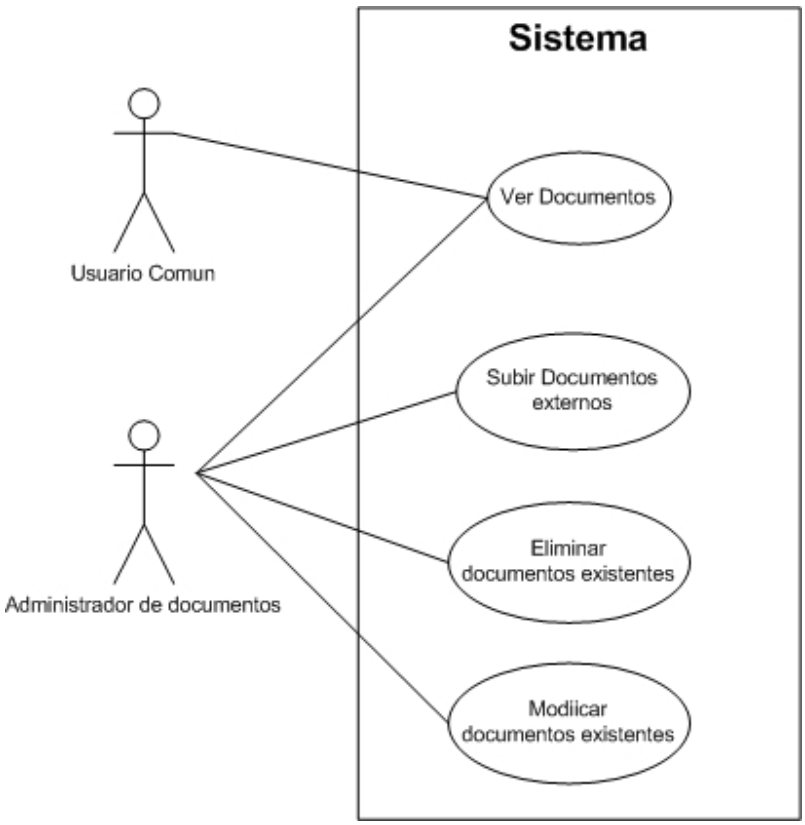


Figura 4: Caso de uso “Acceso de documentos de biblioteca”.

Caso de Uso	Acceso de documentos de biblioteca
Actores	Usuario Común, Administrador de Documentos
Tipo	Primario
Descripción	A la fuente central de documentos, llamada a lo largo de este documento como biblioteca, pueden acceder todos los usuarios del sistema o administradores, pero sólo estos últimos podrán ingresar, eliminar y modificar documentos.

Tabla 10: Caso de uso “Acceso de documentos de biblioteca”.

6.1.4 Proceso de control activo

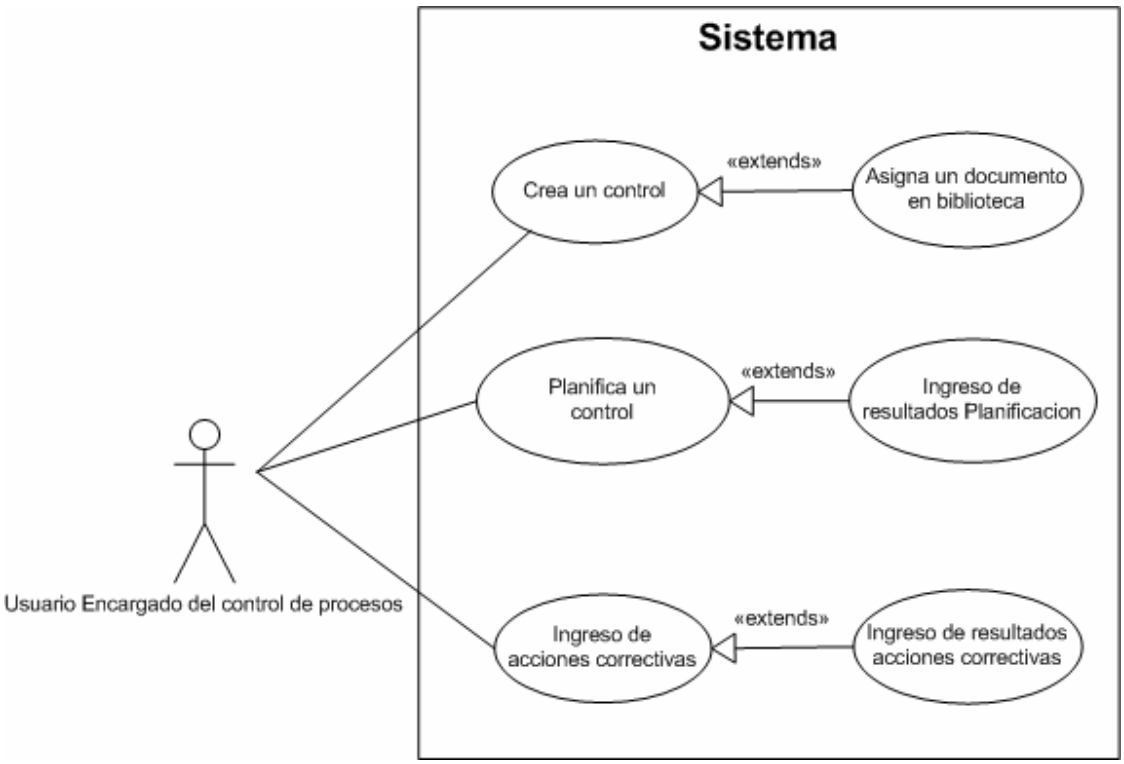


Figura 5: Caso de uso "Proceso de control activo".

Caso de Uso	**Proceso de control activo**
Actores	Usuario encargado de control de procesos
Tipo	Primario
Descripción	Para administrar el proceso de fiscalización al interior de la empresa primero de debe ingresar un control que está

	constituido por un conjunto de variables medibles, a este control se le puede asignar uno o más documentos existentes en la biblioteca. Posteriormente se planificará este control, es decir, se especificará cuando y donde se verificarán o medirán las variables. A esta planificación se le ingresarán resultados y si es necesario acciones correctivas asociadas, para finalizar se ingresaran el estado y ejecución de las acciones correctivas.

Tabla 11: Caso de uso “Proceso de control activo”.

6.1.5 Proceso de ingreso y modificación de maestros del sistema

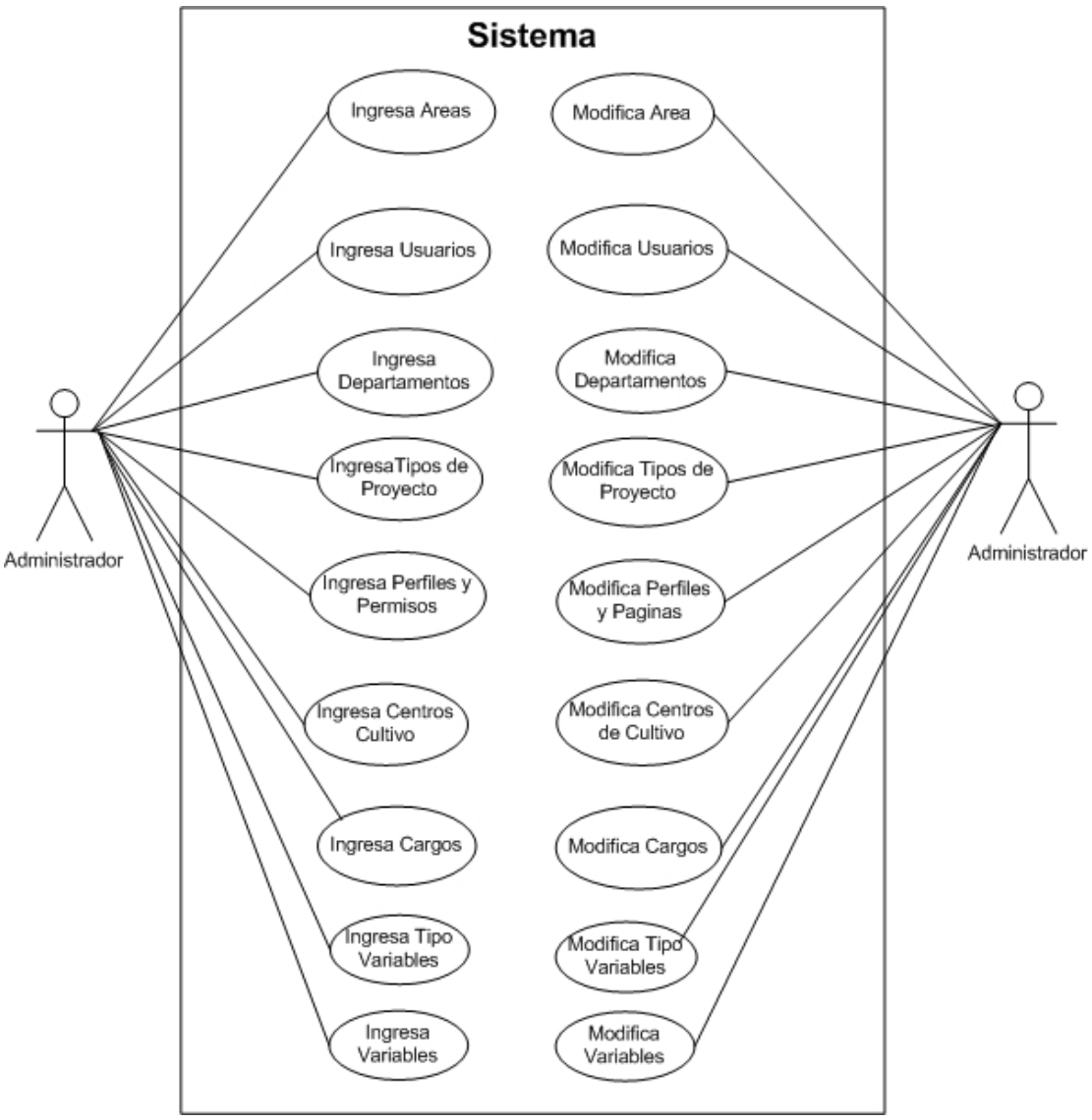


Figura 6: Caso de uso “Ingreso y modificación de maestros del sistema”.

Caso de Uso	Ingreso y modificación de maestros del sistema
Actores	Administrador
Tipo	Primario
Descripción	Para utilizar el sistema es necesario tener las funcionalidades que permitan manejar las variables denominadas maestros. Estas variables permiten la facilidad y rapidez, además de la disminución de errores en el ingreso en los módulos funcionales del sistema. Como se observa en el esquema cada variable debe tener la contar con el proceso de ingreso y modificación.

Tabla 12: Caso de uso "Ingreso y modificación de maestros del sistema".

6.2 Identificación de las entidades de negocio y sus relaciones de alto nivel.

En esta actividad se realiza un estudio del problema para detectar las clases candidatas y sus respectivas relaciones. Esto se hace siguiendo la guía del "modelo de dominio inicial" de [Ambler2006].

La primera parte consiste en identificar las entidades candidatas del modelo del dominio. Las entidades identificadas son las siguientes:

Proyecto

Documento Externo

Documento Interno

Versión

Tareas

Grupo de Trabajo

Documentos de Apoyo

Departamento

Área

Cargos

Usuarios

Formulario de control

Variable de Control

Planificación

Centro de Cultivo

Acciones Correctivas

En la figura 7 se muestra el diagrama de dominio inicial realizado con la notación UML.

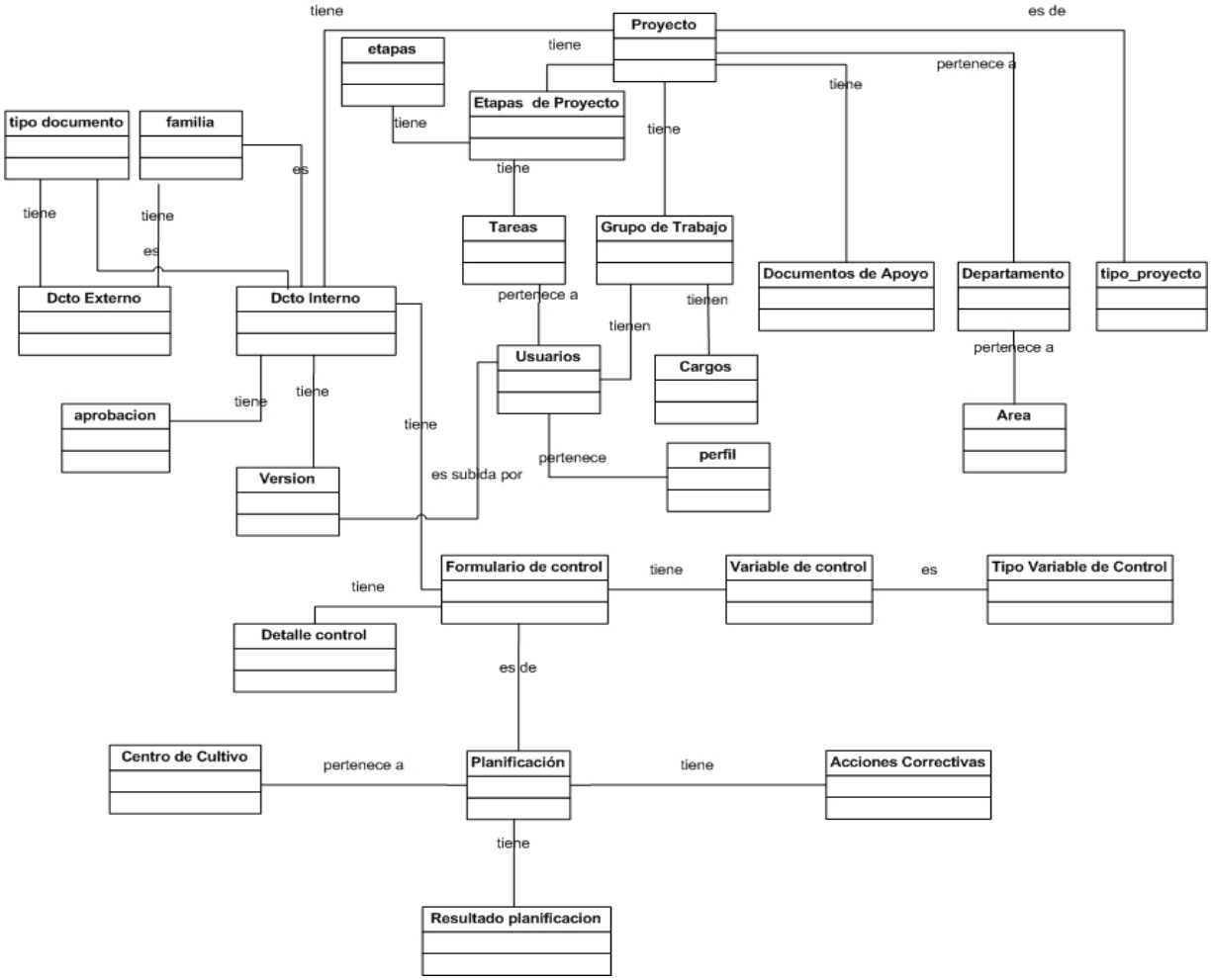


Figura 7: Diagrama inicial de dominio.

6.3 Glosario (Diccionario de datos)

En su forma más simple, el glosario es una lista de los términos relevantes y sus definiciones. Es habitual que un término sea este técnico o del dominio, sea utilizado por diferentes personas con distintos nombres o para distintos usos [Larman 2002].

El diccionario de datos es empezado en la fase de inicio pero puede ser actualizado en cualquier fase del proyecto. He aquí, entonces, el diccionario preeliminar de datos hasta terminada la primera fase del análisis de requisitos:

Alias	**Descripción**
Documentos a crear	Son los documentos asignados a un proyecto que posteriormente tendrán versiones y que luego de un proceso iterativo de actualizaciones y revisiones serán parte de la biblioteca.
Documentos de Apoyo	Son los documentos denominados como plantillas, es todo tipo de recurso agregado al proyecto que sirva a los usuarios del grupo de trabajo para la creación de sus nuevos documentos. Un documento de apoyo puede ser un documento en versión anterior, fotos, videos, etc.

Cargo	Es el cargo que toma un usuario dentro del grupo de trabajo, no se refiere al cargo que un usuario tiene en la empresa o institución donde se utiliza el sistema. Un cargo puede ser por ejemplo: Aprobador, Desarrollador de Contenidos, etc.
Proyecto	Un proyecto es el nombre que se le da a la instancia creada en el sistema que permitirá crear o actualizar documentación. Es decir un proyecto siempre está asociado a documentos, protocolos, minutas, etc.
Versiones	Son las versiones que suben los usuarios pertenecientes al grupo de trabajo de un proyecto. Cada documento a crear puede tener muchas versiones, pero después de la aprobación del documento a crear será siempre la última versión la que aparecerá publicada en la biblioteca.

Tabla 13: Diccionario de datos

6.4 Modelado de arquitectura de alto nivel

El diagrama que se muestra en la figura 8 representa a nivel general la arquitectura del sistema. Se puede observar que al ser un sistema Web la capa

de presentación está compuesta de páginas Web almacenadas en un servidor, las cuales se utilizan para interactuar con el usuario final. En la siguiente capa nombrada como capa de negocios, se implementarán todas las clases que implementan el modelo de dominio y los aspectos comunes de las reglas de negocios, estos elementos (tanto clases como aspectos) serán definidos en la siguiente fase. En última capa llamada capa de acceso a datos o de persistencia, corresponde implementar las clases para realizar mapeo objeto-relacional sobre la base de datos.

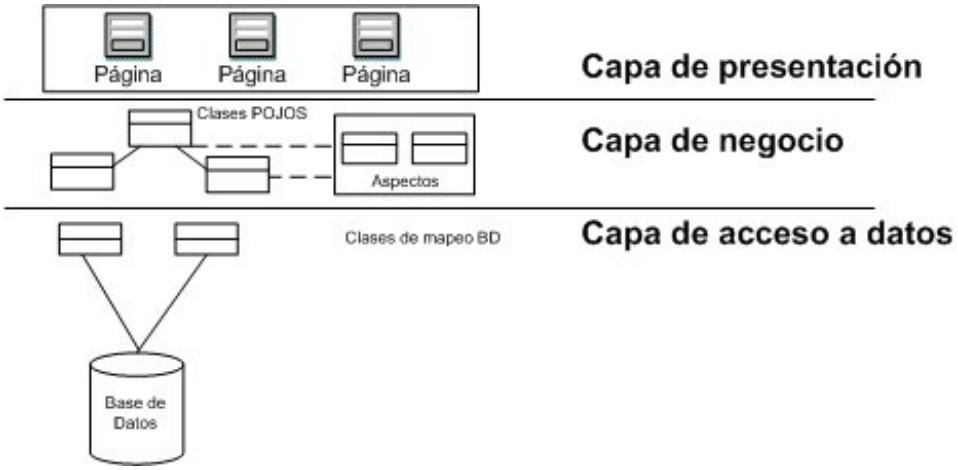


Figura 8: Diagrama de arquitectura de alto nivel.

6.5 Pruebas

6.5.1 Planificación Inicial de las pruebas

En cada fase del desarrollo se realizan un conjunto de pruebas que permiten validar las tareas realizadas.

Algunas de estas pruebas son las siguientes:

- Revisión de código
- Revisión del diseño
- Revisión del modelo
- Pruebas de clases (pruebas de unidad)
- Pruebas en interfaces de usuario
- Pruebas de aceptación (cliente)

6.6 Levantamiento el entorno del trabajo y herramientas utilizadas

El desarrollo de esta actividad es completado a lo largo de todo el proyecto, sin embargo, en esta sección se hará una breve descripción de la plataforma de desarrollo, herramientas y cuales son las principales ventajas de su utilización.

Además se agrega un apartado especial sobre elementos teóricos de la programación orientada a aspectos.

6.6.1 Plataforma de Desarrollo

6.6.1.1 Java (Lenguaje Base)

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. Es un lenguaje muy utilizado por la industria del software y existen muchas tecnologías asociados a él. Además es multiplataforma y la principal ventaja a nivel de desarrollo es la existencia herramientas libres y gratuitas con las cuales se puede trabajar.

6.6.1.2 Aspectj (Lenguaje Orientado a aspectos)

AspectJ [Miles2004] es un lenguaje orientado a aspectos de propósito general, cuya primera versión fue lanzada en 1998 por el equipo conformado por Gregor Kickzales (líder del proyecto), Ron Bodkin, Bill Griswold, Erik Hilsdale, Jim Hugunin, Wes Isberg y Mik Kersten. AspectJ es una extensión compatible de Java para facilitar el uso de aspectos por parte de los programadores de Java.
La elección de este lenguaje por sobre otros analizados (JbossAOP y SpringAOP) está dado principalmente por:

- Se considera a Aspectj como la herramienta más avanzada y probada dentro de las existentes para llevar a cabo un proyecto como el que se pretende implementar.

- Es una de las primeras herramientas lanzadas y por este motivo tiene varias versiones mejoradas.
- Existen herramientas de apoyo que permiten integrar el compilador a herramientas de desarrollo existente como eclipse, netbeans, etc.
- Es posible encontrar en Internet abundante documentación con casos reales de implementación de la programación orientada a aspectos, utilizando Aspectj.

6.6.1.3 Mysql (Sistema de gestión de Base de Datos)

El software MySQL® proporciona un servidor de base de datos SQL (Structured Query Language) muy rápido, multi-threaded, multi usuario y robusto. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido. MySQL es una marca registrada de MySQL AB.

El software de bases de datos MySQL es un sistema cliente/servidor que consiste en un servidor SQL multi-threaded que trabaja con diferentes backends, programas y bibliotecas cliente, herramientas administrativas y un amplio abanico de interfaces de programación para aplicaciones (APIs).

Las principales razones para elegir el gestor de base de datos Mysql son las siguientes:

- Experiencia en otros proyectos con el gestor de base de datos, lo que facilita la implementación y agiliza la codificación.
- Al ser libre y gratuito se muestra como ventaja ante futuros clientes.
- Es multiplataforma.

6.6.1.4 Servidor Web (Tomcat)

Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. Para la elección de esta herramienta se realizaron pruebas levantándose aplicaciones similares tanto en Tomcat 6.0 como en GlassFish obteniéndose las siguientes conclusiones:

- Tomcat se demora menos en los tiempos de respuesta.
- Tomcat ocupa menos recursos a nivel de servidor.
- Tomcat demuestra mayor robustez puesto que nunca fue necesario reiniciarlo mientras que GlassFish si fue necesario el reinicio.

6.6.2 Herramientas de Desarrollo

6.6.2.1 Netbeans

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. Es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos [Netbeans2008].

El principal motivo para usar Netbeans deriva de la facilidad para hacer interfaces gráficas y de cantidad de complementos disponibles que sirven de mucho apoyo a la hora de desarrollar software.

6.6.2.2 Plugin para desarrollo de Aspectos para Netbeans

Es un plugin creado por Ramón Ramos que permite integrar el lenguaje orientado aspectos, Aspectj, al entorno de desarrollo Netbeans. Puede ser

descargado libremente y posee utilidades de seguimiento y ayuda en la integración del Aspectos con el lenguaje base (JAVA).

6.6.2.3 Microolap Database Designer for MySQL

Es un sistema de desarrollo visual previsto para el diseño, modelado, creación, modificación e ingeniería reversa de base de datos Mysql.

6.6.2.4 PowerDesigner

Es un conjunto de herramientas que permiten diseñar, integrar y gestionar los modelos de datos y procesos para los sistemas de información de forma consistente, acelerando el desarrollo de nuevas aplicaciones y facilitando los cambios en las actuales. Basado en técnicas de toma de requerimientos funcionales, diseño de modelo de datos conceptual, lógico y físico; diseño de modelo de procesos de negocio y UML reúne todo lo necesario para el análisis y gestión de cambios en el sistema, facilitando la colaboración entre las áreas de negocio y tecnología.

6.6.2.5 Navicat

Es una solución ideal para la Administración y Desarrollo para MySQL. Este es un programa para cliente de mysql que provee una potente interfase gráfica para la administración, desarrollo y mantenimiento de Bases de Datos.

6.6.2.6 Macromedia Flash

Es una aplicación en forma de estudio de animación que trabaja sobre "fotogramas" destinado a la producción y entrega de contenido interactivo para diferentes audiencias alrededor del mundo sin importar la plataforma. [Adobe2006].

6.6.2.7 Mysqlquerybrowser

Es una herramienta gráfica proporcionada por MySQL AB para crear, ejecutar, y optimizar consultas en un ambiente gráfico, donde el MySQL Administrator está diseñado para administrar el servidor MySQL. Y el MySQL Query Browser está diseñado para ayudarle a consultar y analizar datos almacenados en su base de datos MySQL.

6.6.2.8 Subversión

Subversión es un sistema centralizado para compartir información. En su núcleo está un repositorio, que es un almacén central de datos. El repositorio almacena información en forma de un árbol de ficheros, una jerarquía típica de ficheros y directorios. Cualquier número de clientes se conectan al repositorio, y luego leen o escriben esos ficheros. Al escribir datos, el cliente hace que la información esté disponible para los otros, al leer los datos, el cliente recibe la información de los demás. [Küng2006]

6.6.3 Elementos de la Programación orientada aspectos

Como se dijo anteriormente la programación orientada aspectos viene a complementar la actual y conocida programación orientada a objetos, esto se logra principalmente utilizando unidades modulares e independientes llamadas “aspectos”.

Al tener la siguiente situación:

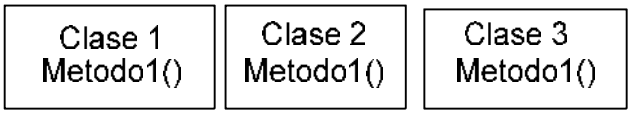


Figura 9: Esquema de clases sin aspectos

Con la utilización de programación orientada a aspectos se transforma en lo siguiente:

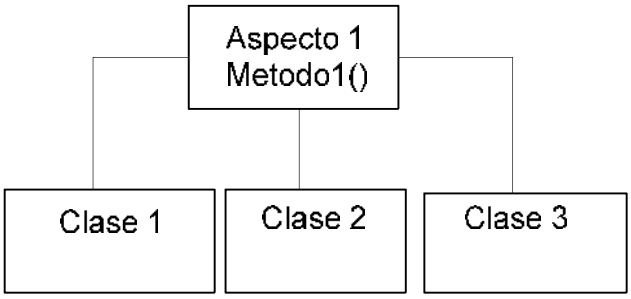


Figura 10 : Esquema de utilización de aspectos

Para lograr lo mostrado en la figura 10 el lenguaje elegido Aspectj extenderá al lenguaje base Java principalmente agregando a la semántica los siguientes elementos básicos [Asteasuain2002].

- Los puntos de enlace (Jointpoints) son puntos bien definidos en la ejecución de un programa, entre ellos podemos citar llamadas a métodos y accesos a atributos.
- Los puntos de corte (Pointcuts) agrupan puntos de enlace y permiten exponer el contexto en ejecución de dichos puntos. Existen cortes primitivos y también definidos por el usuario.
- Los avisos (advices) son acciones que se ejecutan en cada punto de enlace incluido en un corte. Los avisos tienen acceso a los valores expuestos por el corte. Las tres entidades anteriores son dinámicas

porque permiten definir comportamiento adicional que actuará en tiempo de ejecución.

Se puede encontrar un mayor detalle en el anexo D “Sintaxis del lenguaje Aspectj”.

7. Fase elaboración

7.1 Modelado de arquitectura

Como se definió en la fase anterior (inicio), la arquitectura lógica está definida por una arquitectura en 3 capas que son: Presentación, negocios y acceso a datos.

En cada capa se han definido los framework, herramientas y patrones de diseño que se han considerado adecuados y que faciliten la implementación. En el caso de los patrones de diseño se utilizan los pertenecientes a las publicaciones de Sun [Sun2002] denominadas Catálogo de Patrones Principales de J2EE (Core J2EE Patterns).

7.1.1 Capa de Presentación

La capa de presentación de la aplicación está compuesta por las páginas que permiten interactuar con el usuario final.

En esta capa se busca un framework que facilite la generación de pantallas, mapeo de los formularios y sus clases en el servidor, también que facilite la

validación de datos, la conversión de datos, la gestión de errores y en que permita utilizar AJAX. En esta etapa el framework elegido es Java Server Faces (JSF) con componentes Icefaces.

7.1.1.1 Java Server Faces (JSF)

Java Server Faces (JSF) es un framework estándar Java que permite construir aplicaciones Web. Se define por las especificaciones JSR 127, JSR 252 y JSR 276 e implementa el patrón Modelo-Vista-Controlador (MVC).

Dicho modelo MVC es un patrón de arquitectura de desarrollo software que separa las aplicaciones en tres capas diferenciadas: datos (Modelo), interfaz de usuario (Vista) y lógica de control (Controlador). Esto permite una separación entre la interfaz de usuario y la lógica que hace que el mantenimiento de las aplicaciones JSF sea sencillo. JSF es un conjunto de componentes de usuario (UI) que permite construir la capa de vista de las aplicaciones Web. Así, proporciona un conjunto de APIs para desarrollar estos componentes UI y gestionar su funcionamiento mediante el tratamiento de eventos, la validaciones de entrada, la definición de la navegación entre páginas y el soporte de accesibilidad.

Asimismo, JSF trata el interfaz de usuario (Vista) de una forma parecida al estilo de Swing o Visual Basic, realizando su programación a través de componentes y basada en eventos.

Además, las clases de componentes UI incluidas en JSF encapsulan su funcionalidad y no la presentación en un tipo de cliente específico por lo que esto permite poder ser pintados en distintos clientes [Dudnay2004].

Otra de las características importantes de JSF es la inclusión de “Managed Beans” que son clases estándar java que siguen la especificación de Java Bean [Sun1997]. Estas clases permiten controlar del servidor las propiedades y eventos que son mapeados en el código de la página a determinados componentes UI.

Para facilitar el uso de Ajax en las páginas de la aplicación se elige el framework Iceface, puesto que facilitará la creación de interfaces atractivas al usuario ahorrando tiempo en codificación. A continuación se explica brevemente el framework Icefaces.

7.1.1.2 Icefaces

Es una extensión al framework JSF, con la única diferencia que se relaciona principalmente con la fase de despliegue de las páginas. La principal característica de JSF es que permite a los desarrolladores Java EE crear y desplegar fácilmente aplicaciones de Internet Enriquecidas (RIA) utilizando Ajax para esta finalidad.

Adicionalmente, para mejorar las interfaces gráficas se utilizan las siguientes herramientas:

7.1.1.3 CSS

Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

7.1.2 Capa de Negocios

La capa de negocios está compuesta por todas las clases que permiten gestionar las peticiones de la capa de presentación y llevar las respuestas de la capa de acceso a datos. En esta capa está implementado el modelo del dominio.

La capa de negocios está compuesta por:

- **Clases (POJO):** El termino POJO (Plain Old Java Objects) es utilizado en Java para referirse a la utilización de clases comunes y corrientes que no dependen de ningún framework en particular [Richardson2006]. Esto permite trabajar con un modelo de dominio limpio, flexible y simple de desarrollar.

- **Aspectos**: Un aspecto es una unidad modular que se dispersa por la estructura de otras unidades funcionales. Los aspectos existen tanto en la etapa de diseño como en la de implementación. Un aspecto de diseño es una unidad modular del diseño que se entremezcla en la estructura de otras partes del diseño. Un aspecto de programa o de código es una unidad modular del programa que aparece en otras unidades modulares del programa. [Kickzales2007].

Algunos patrones utilizados en la capa de Negocios son:

- **Abstract Factory:** El problema que intenta solucionar este patrón es el de crear diferentes familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas.

7.1.3 Capa de Acceso a Datos

Esta capa está compuesta por las clases encargadas de comunicar la capa de negocios con la fuente externa de datos que en este caso es una base de datos relacional. Son implementadas a nivel de codificación utilizando también clases POJOS que se comunican con la base de datos a través de la especificación JDBC que es utilizada en el desarrollo con Java.

Los patrones utilizados en la capa de acceso a datos son los siguientes:

- **Data Access Object (DAO):** Permite desacoplar la lógica de negocio de la lógica de acceso a datos, de manera que se pueda cambiar la fuente de datos fácilmente.

- **Data Transfer Object (DTO):** EL patrón DTO, es un patrón de diseño que permite optimizar la transferencia de datos a través de las capas de

la aplicación. Una clase DTO está compuesta por todos los atributos de la tabla de la base de datos que representa con sus correspondientes accessors (Setters y Getters).

Entonces las clases de la capa de negocios y capa de acceso a datos instancian las clases dto`s que necesiten para ingresar o devolver datos a través de sus métodos. Por ejemplo en la figura 11 se puede ver un esquema de funcionamiento básico del patrón DTO para ingresar un dato a la base de datos.

Asimismo al hacer el proceso inverso, es decir, al extraer un conjunto de datos (de la fuente externa de datos), también se instancian las dto`s correspondientes, pero en este caso las capas inferiores devuelven un objeto dto que llega finalmente a la capa de presentación.

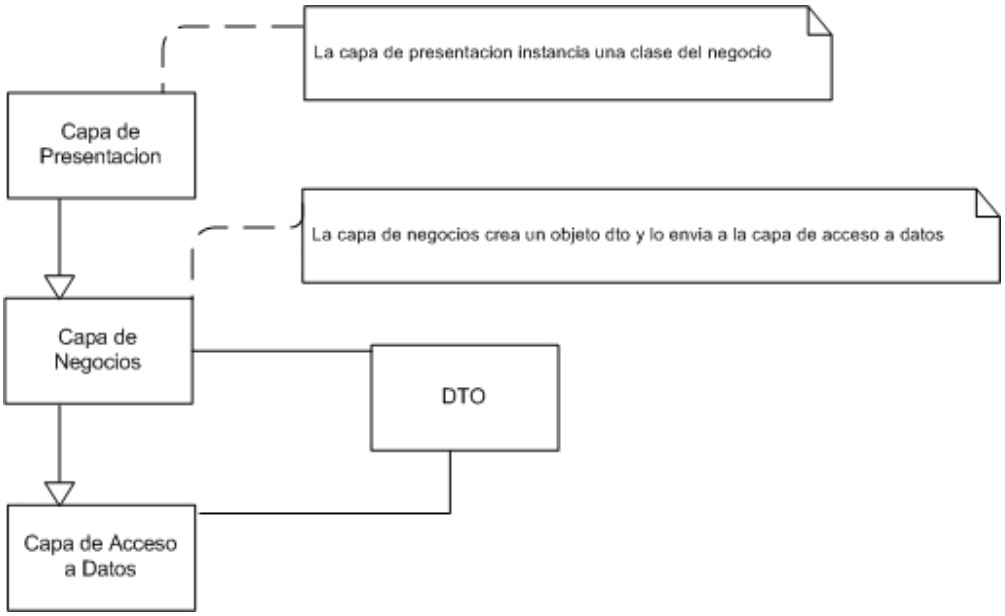


Figura 11: Funcionamiento del patrón DTO para transporte de un dato

7.2 Prototipo interfaces de usuario

Todas las interfaces se componen de los siguientes elementos:

Menú: elemento flash que permite al usuario acceder a las distintas páginas que implementan las funcionalidades del sistema

Titulo: Imagen formato JPG, con el título de la página

Subtítulos: Imagen en formato JPG donde se indica cual es la funcionalidad en concreto que realiza alguna u otra parte de la página. Todas las páginas tienen al menos un subtítulo.

Cuerpo: Elementos que permiten llevar a cabo la funcionalidad misma de la página. Puede estar compuesta por Label, Texbox, Grillas, Combobox, Listas, etc.

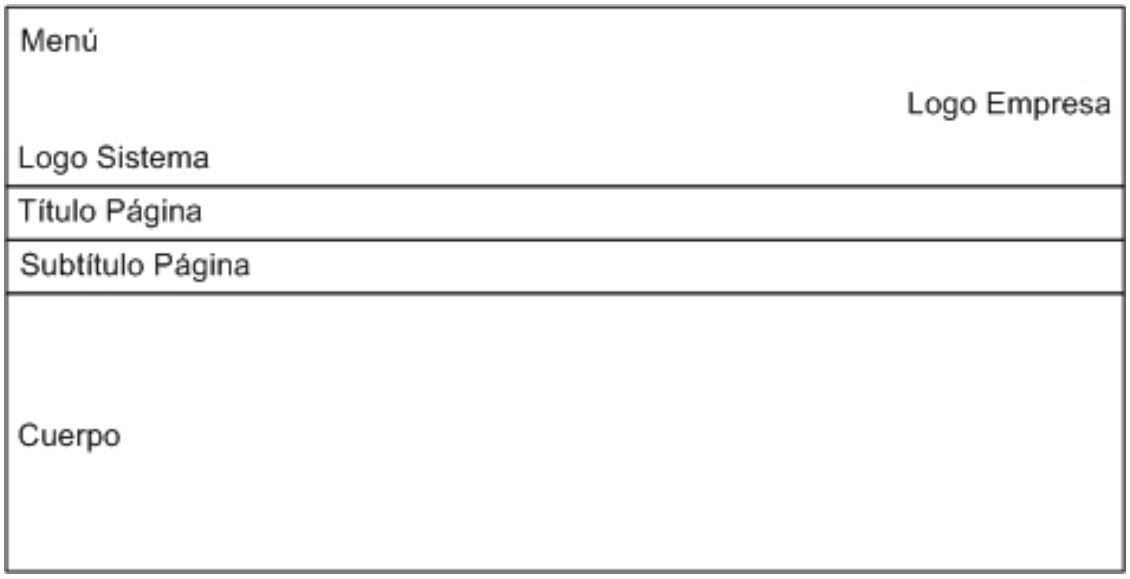


Figura 12: Esquema general de interfaz de usuario

A continuación se muestran las principales interfaces de usuario del sistema.

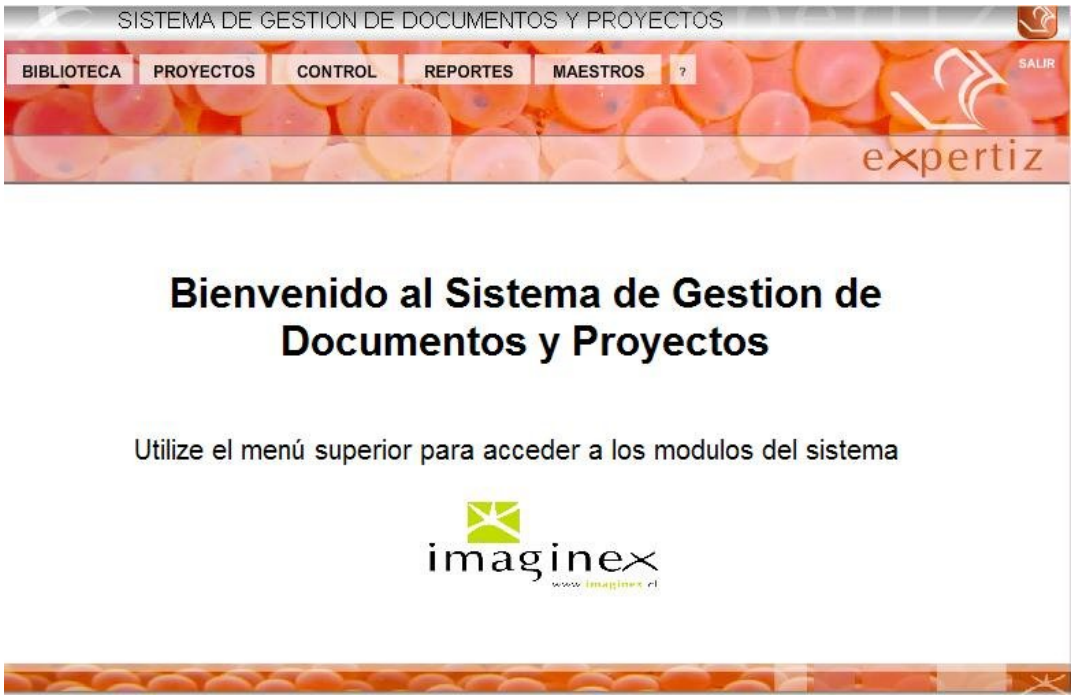


Figura 13: Interfaz de bienvenida al sistema

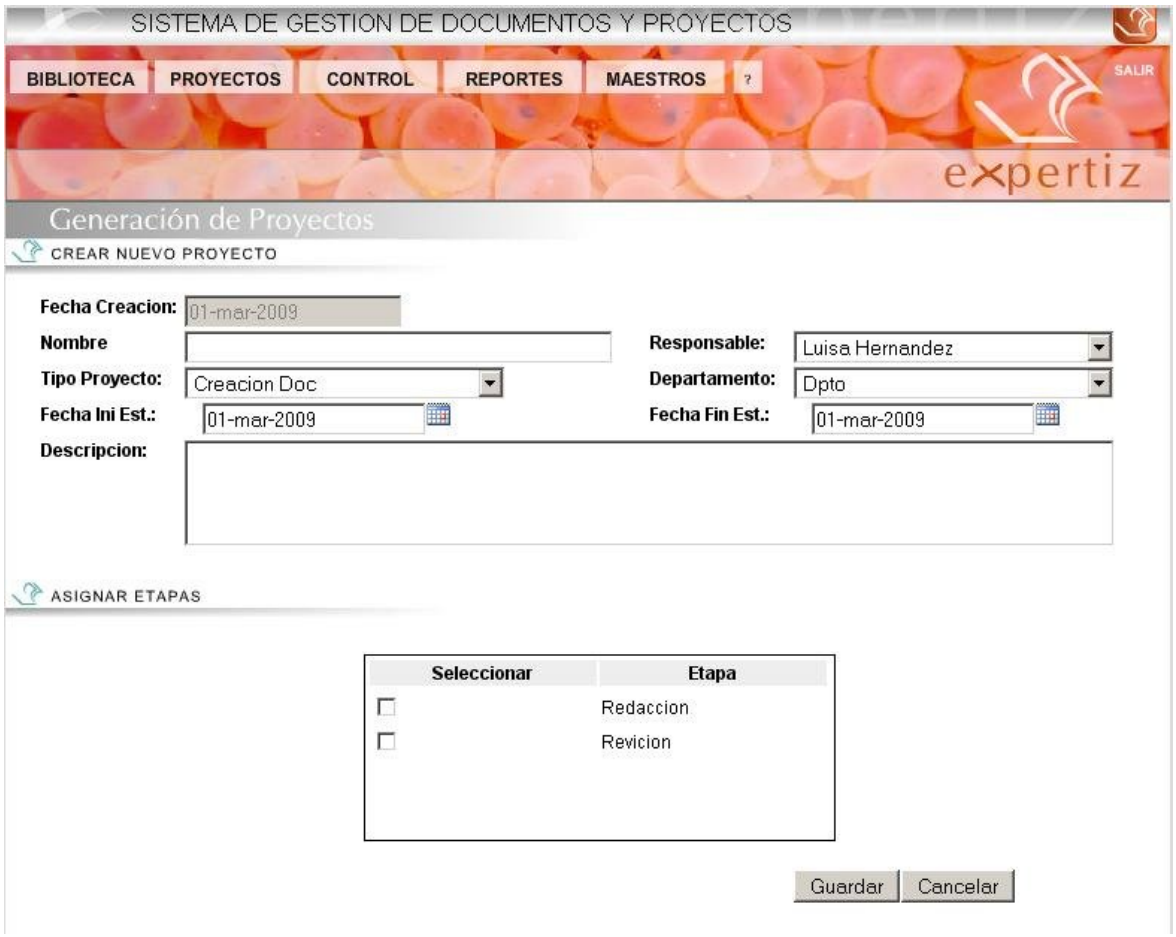


Figura 14: Interfaz de usuario que permite generar nuevos proyectos

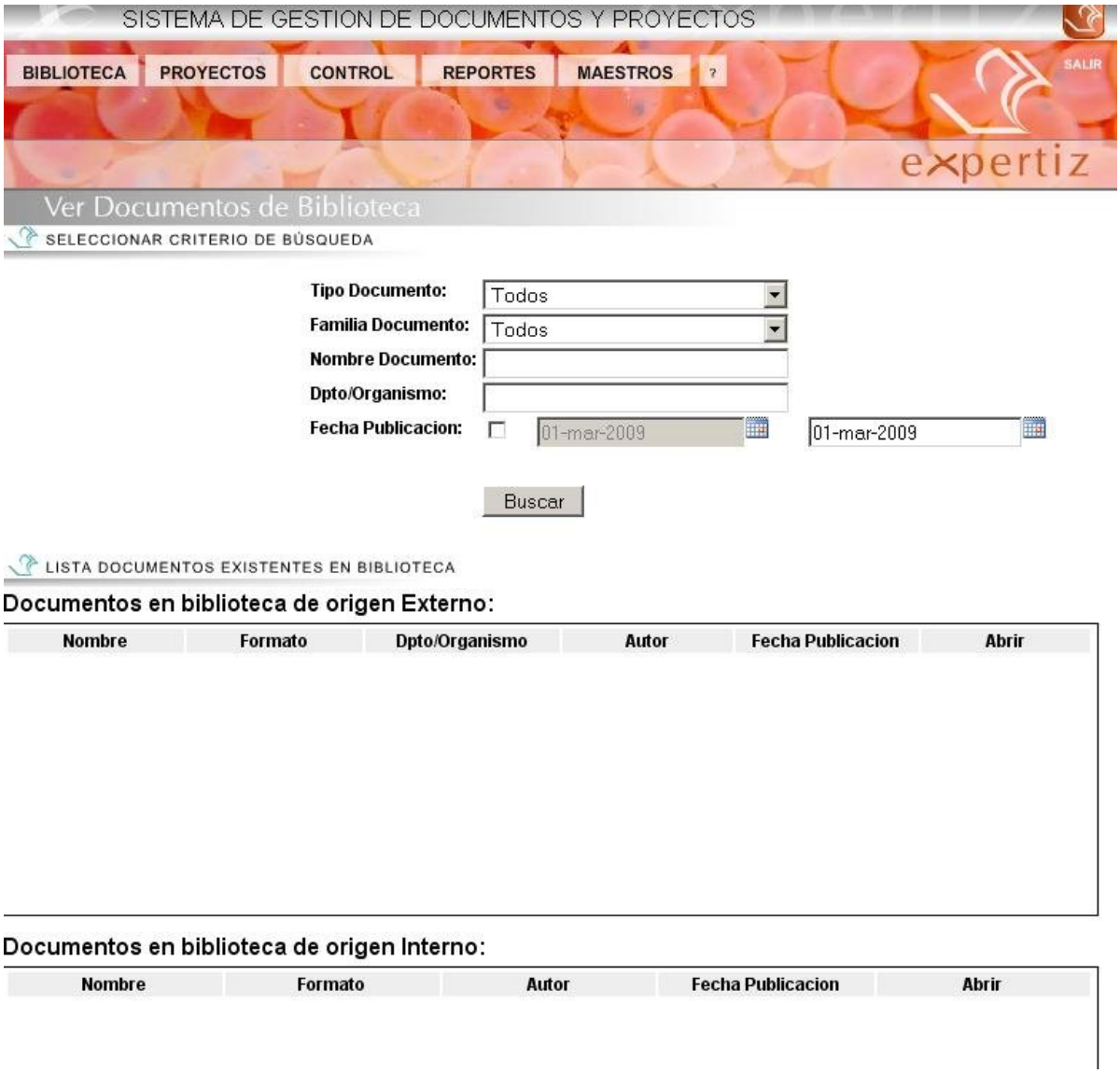


Figura 15: Interfaz de usuario que permite ver los documentos existentes en biblioteca.

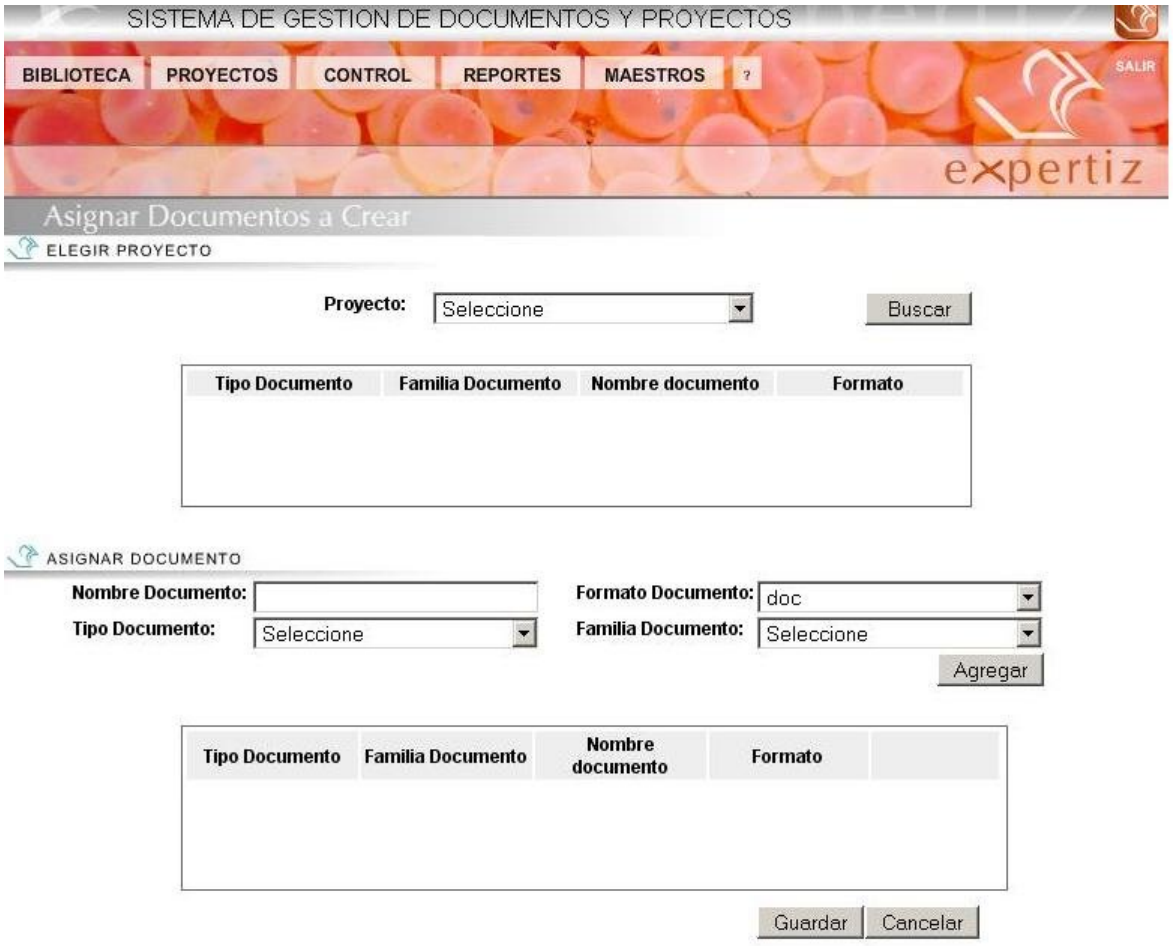


Figura 16: Interfaz de usuario que permite agregar nuevos “documentos a crear” a un proyecto existente

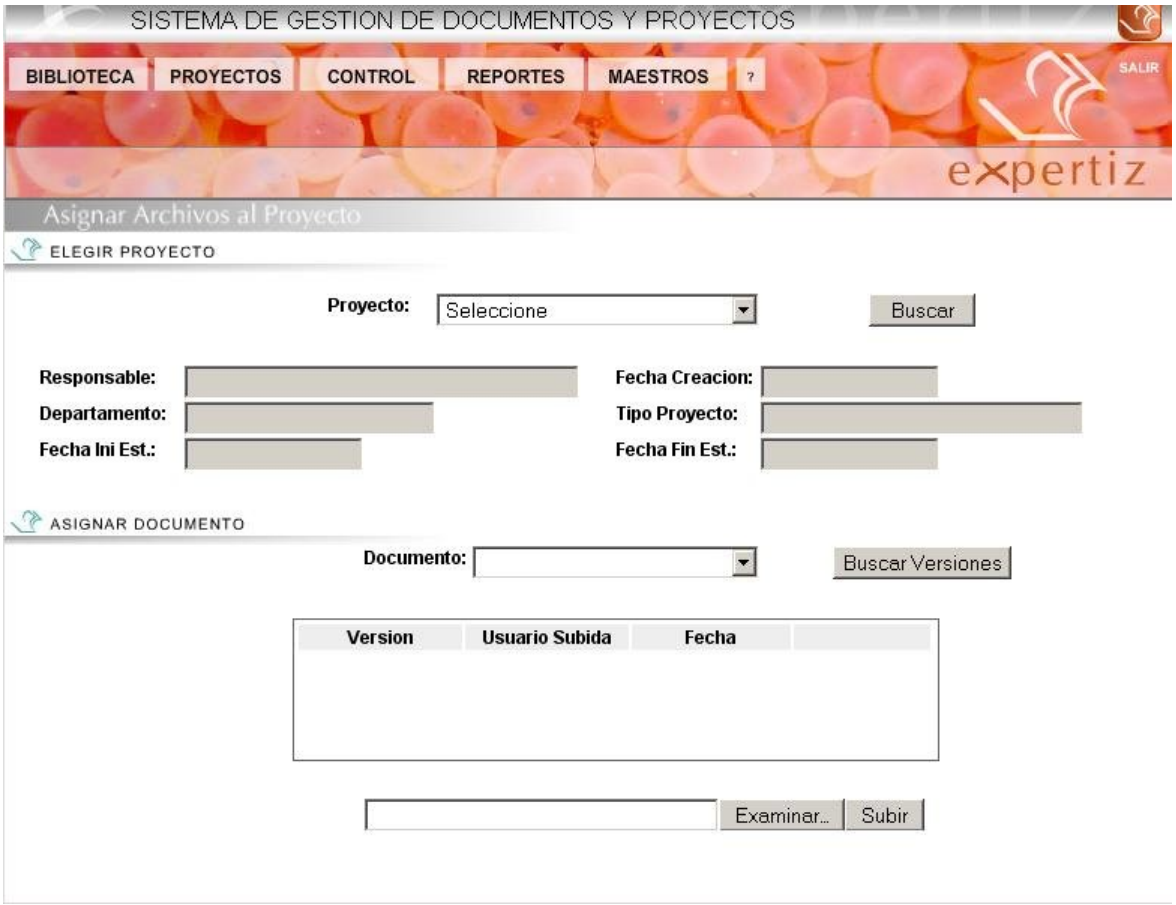


Figura 17: Interfaz de usuario que permite agregar nuevas versiones a un documento existente

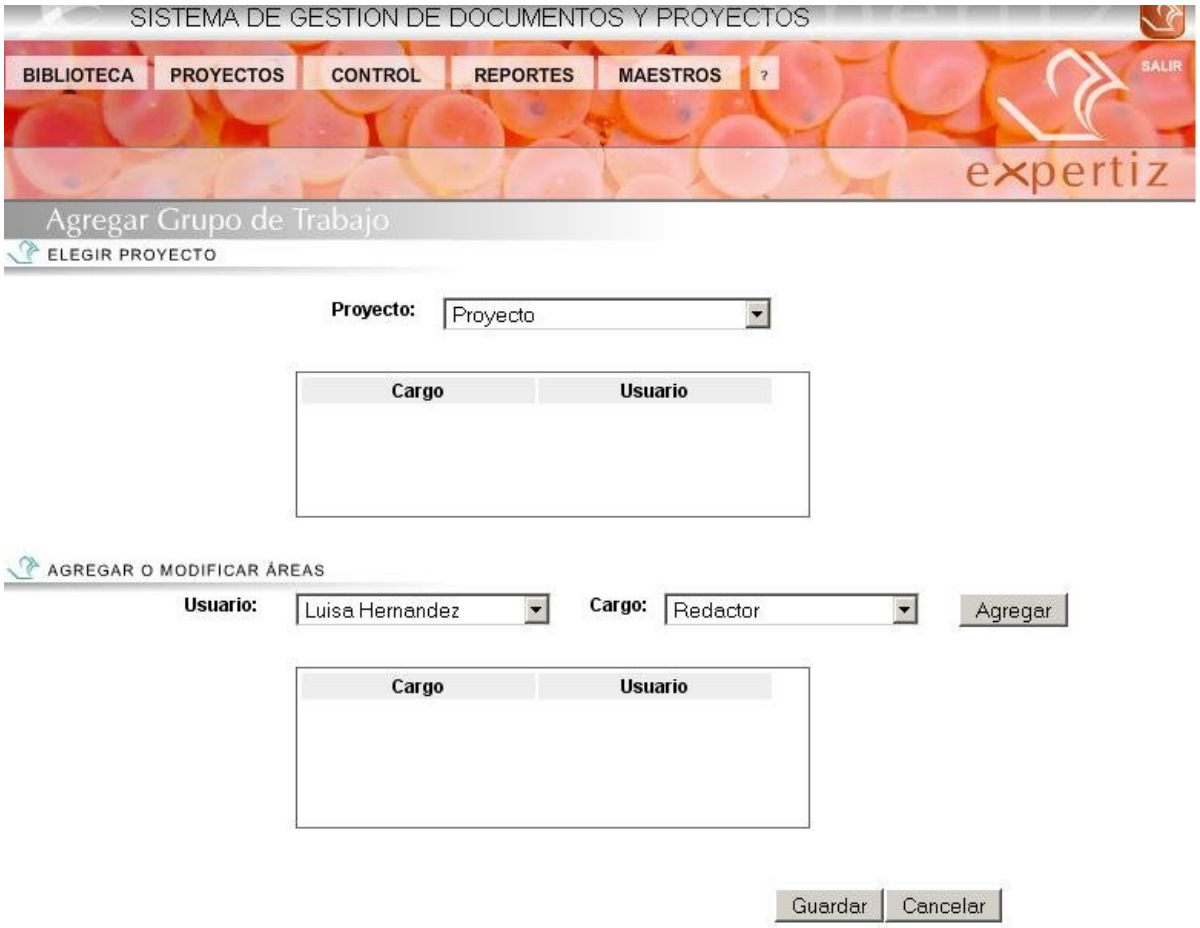


Figura 18: Interfaz de usuario que permite asignar usuarios a un grupo de trabajo para un proyecto existente

Figura 19: Interfaz de usuario que permite asignar documentos de apoyo a un proyecto existente

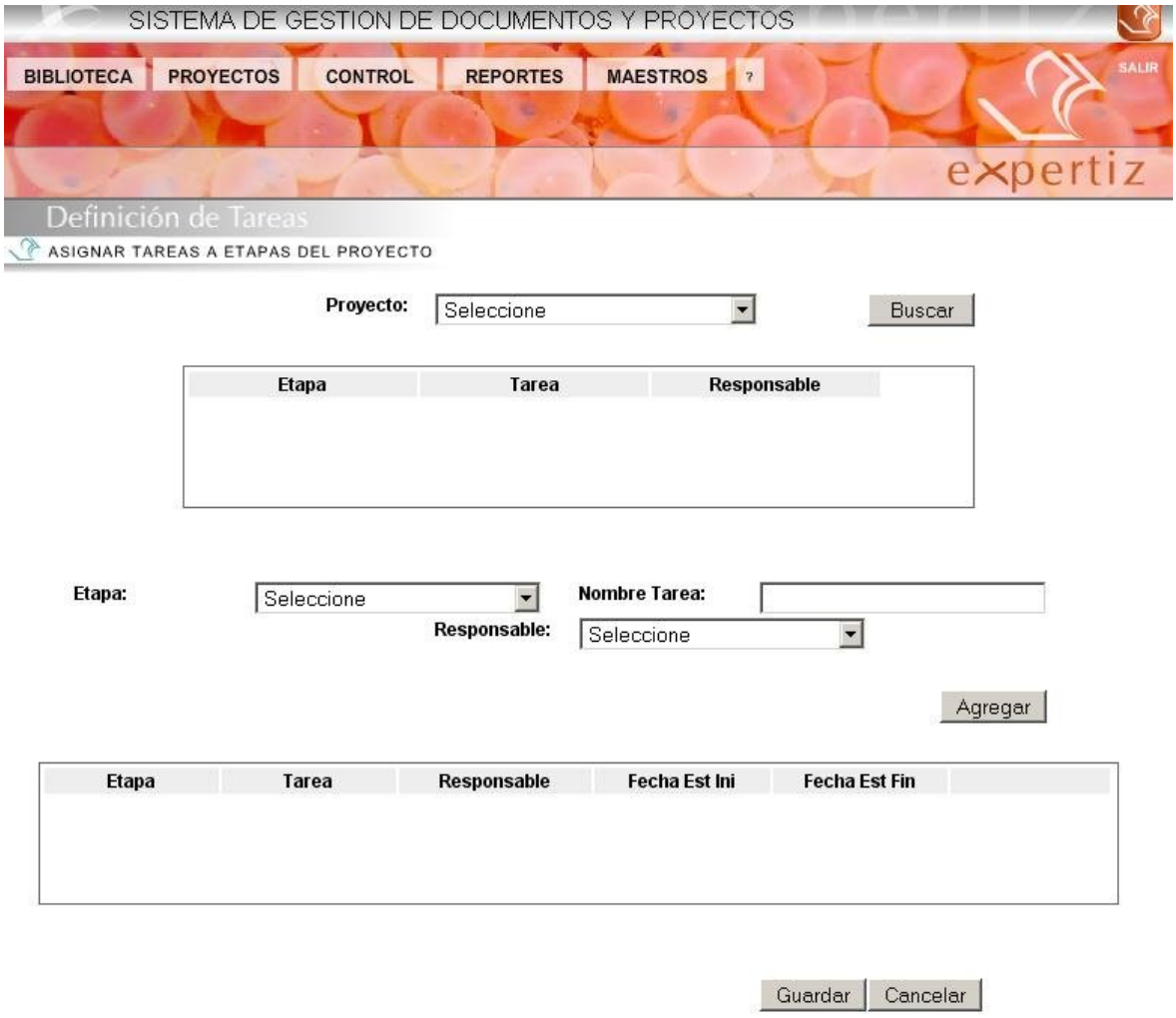


Figura 20: Interfaz de usuario que permite definir tareas a los usuarios pertenecientes al proyecto

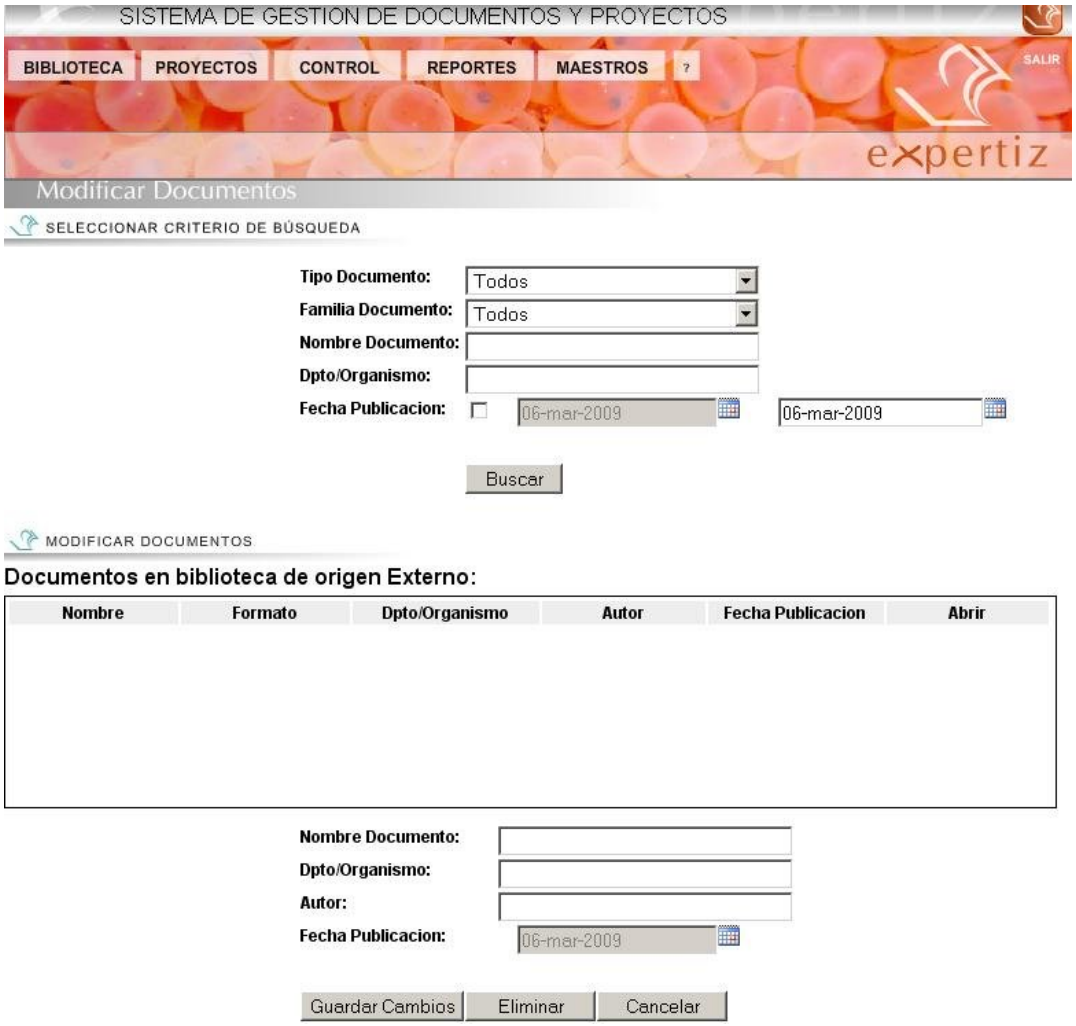


Figura 21: Interfaz de usuario que permite modificar o eliminar documentos existentes en biblioteca

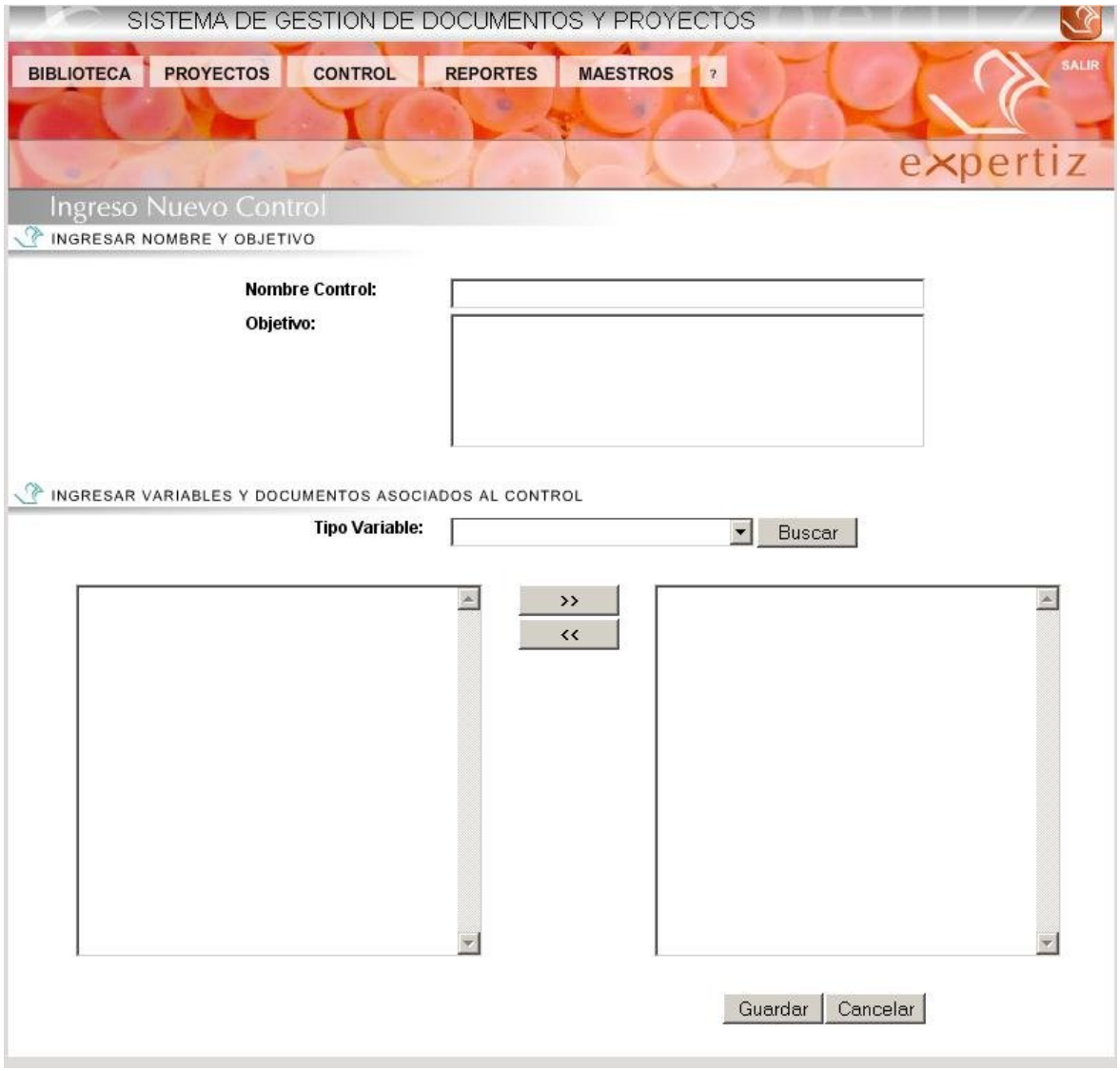


Figura 22: Interfaz de usuario que permite crear un nuevo control

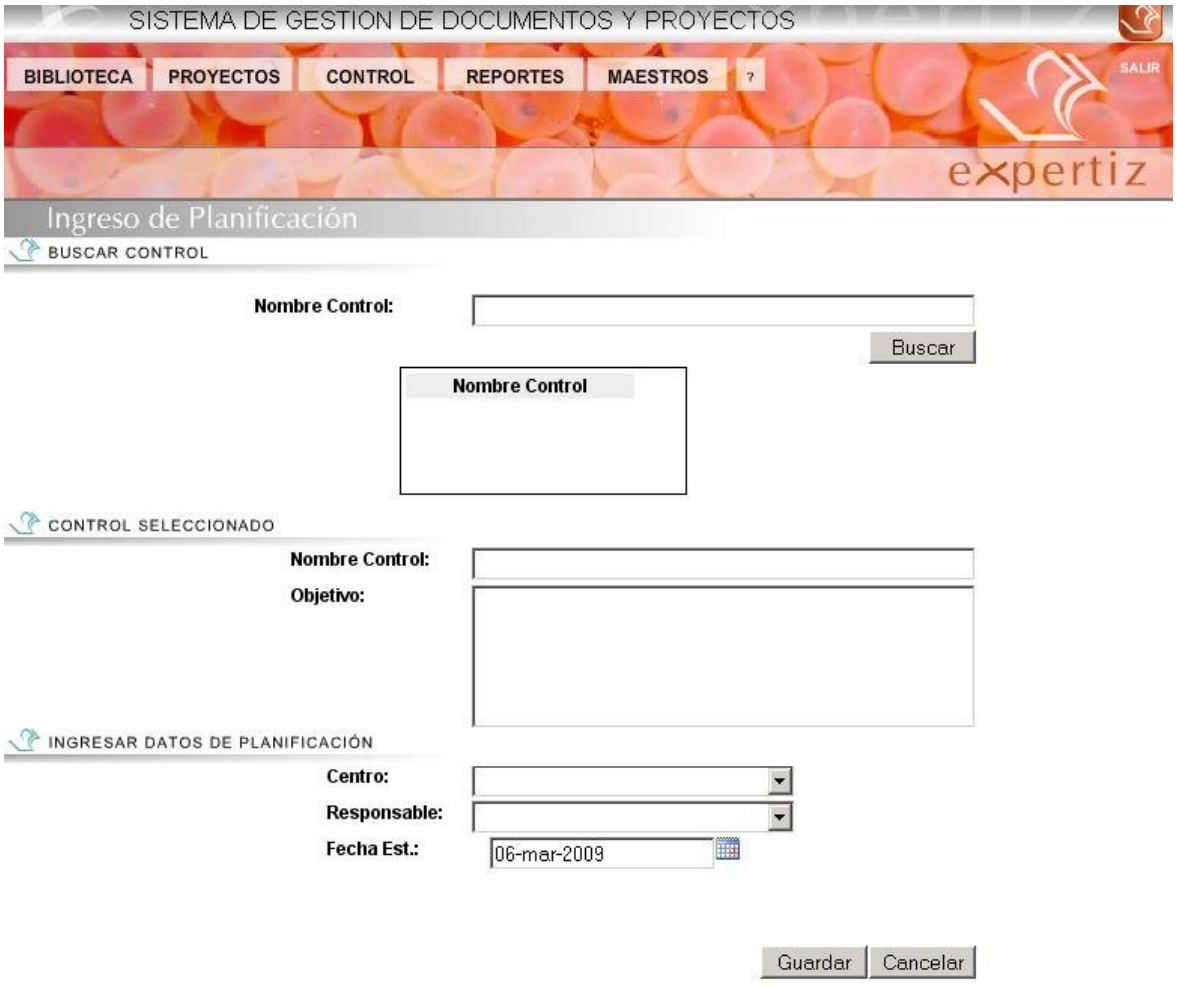


Figura 23: Interfaz de usuario que permite planificar un control existente

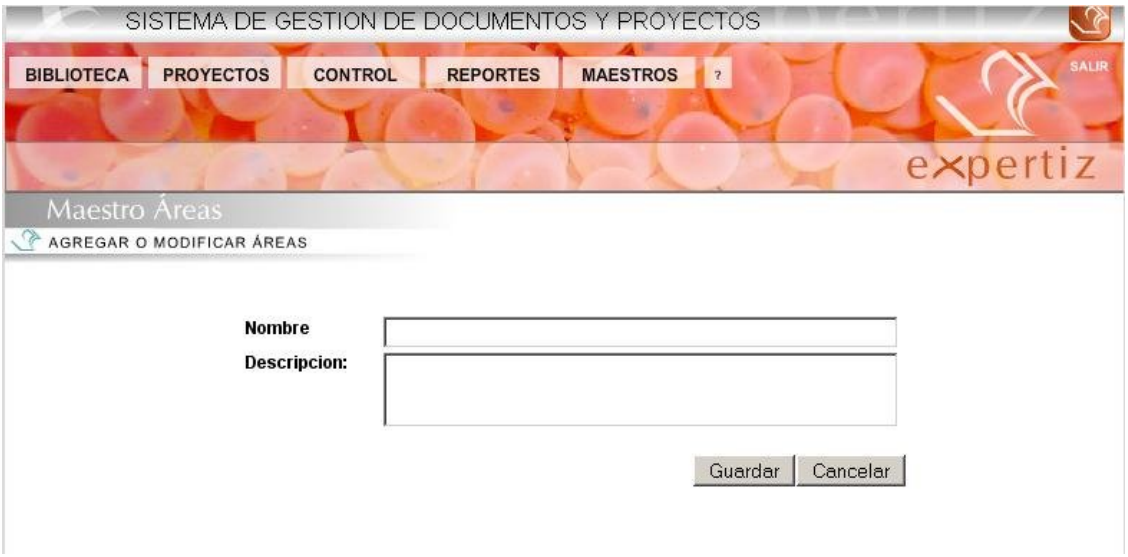


Figura 24: Interfaz de usuario que permite ingresar nuevas áreas al sistema

8. Fase Construcción

8.1 Diagramas de secuencias

Para estudiar la dinámica que siguen los objetos a través del código fuente se realizan los diagramas de secuencia de UML. Estos diagramas se han confeccionado a partir de los casos de uso planteados en la fase de inicio y permiten identificar métodos que deben ser implementados en las diferentes capas del sistema.

8.1.1 Diagrama de secuencia de ingreso de un nuevo proyecto

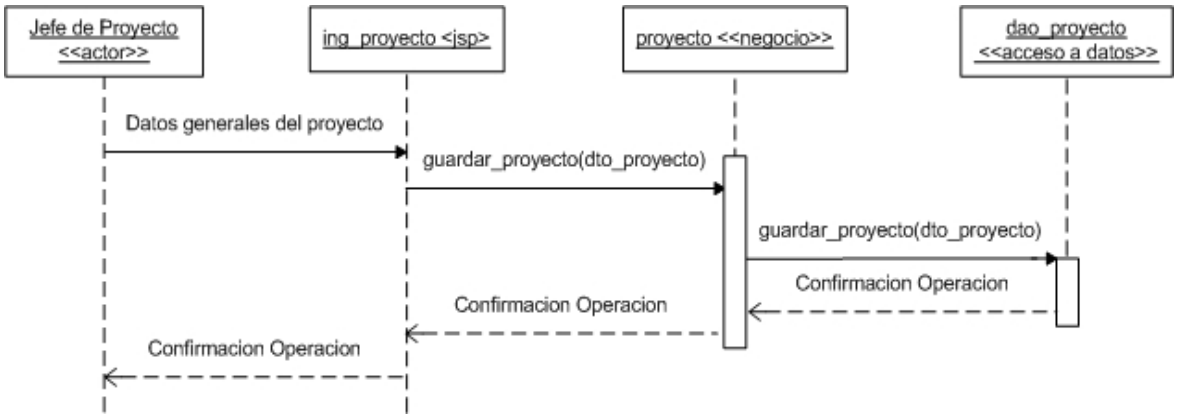


Figura 25: Diagrama de secuencia de ingresar un nuevo proyecto

En la figura 25 se muestra la secuencia de un ingreso de nuevo proyecto basado en el caso de uso “ingreso datos generales del proyecto” de la figura 2. La secuencia de actividades es la siguiente:

1. El actor “jefe de proyecto” ingresa los datos generales del nuevo proyecto en el sistema.
2. La página jsp denominada ing_proyecto llama al método de la lógica del negocio denominado guardar_proyecto.
3. A su vez la clase proyecto de la capa de negocios que llama al método guardar_proyecto de la clase dao_proyecto de la capa acceso a datos.

4. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “jefe de proyecto” que comenzó la secuencia.

8.1.2 Diagrama de secuencia de ingreso de nuevas etapas al proyecto

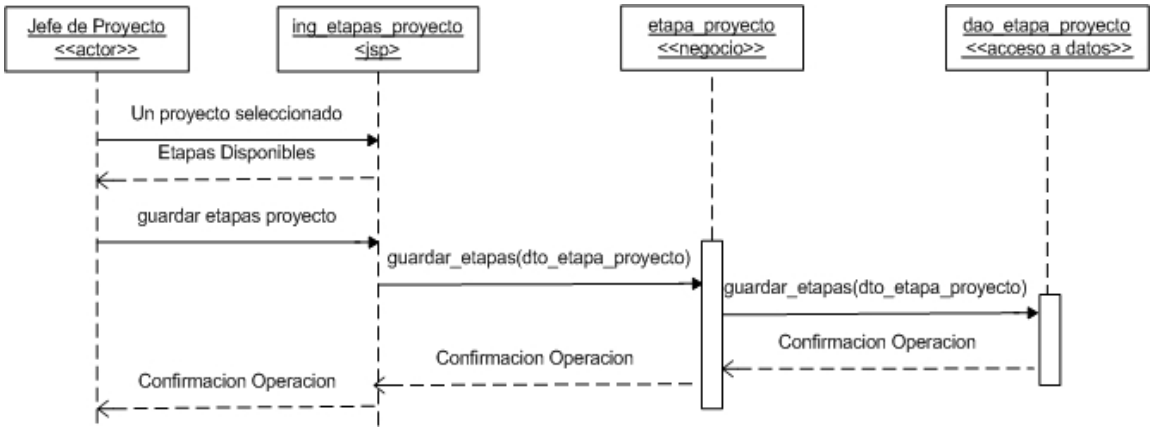


Figura 26: Diagrama de secuencia de ingresar etapas a un proyecto

En la figura 26 se muestra la secuencia de ingresar una etapa a un proyecto existente. Este diagrama de secuencia está basado en el caso de uso “Ingreso de Etapas al Proyecto” de la figura 2. La secuencia de actividades es la siguiente:

1. El actor “jefe de proyecto” selecciona un proyecto existente.
2. La página jsp denominada ing_etapas_proyecto llama al método de la lógica del negocio denominado guardar_etapas.
3. A su vez la clase etapa_proyecto de la capa de negocios llama al método guardar_etapas de la clase dao_etapa_proyecto de la capa acceso a datos.
4. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “jefe de proyecto” que comenzó la secuencia.

8.1.3 Diagrama de secuencia de ingresar un grupo de trabajo a un proyecto

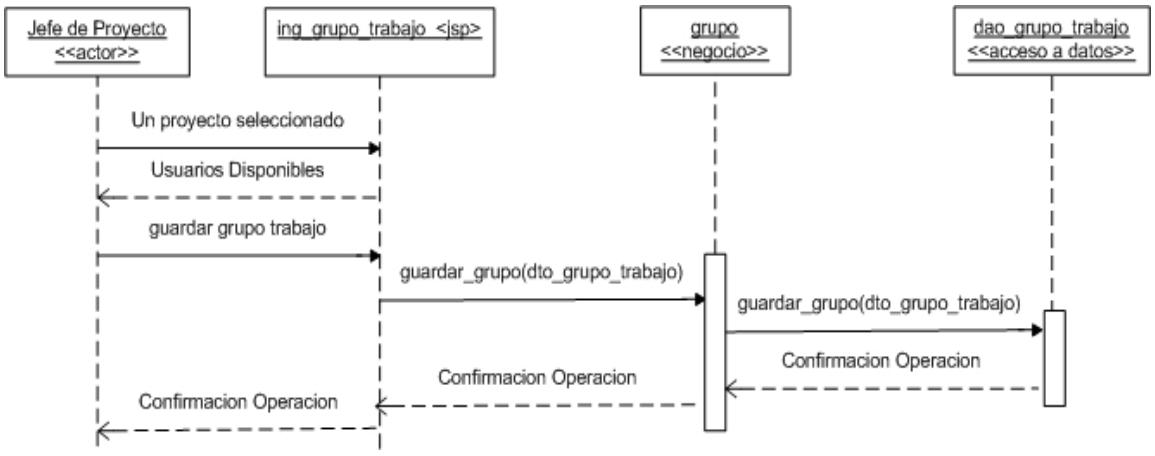


Figura 27: Diagrama de secuencia ingresar grupo de trabajo

En la figura 27 se puede observar la secuencia de ingresar un grupo de trabajo a un proyecto determinado. Este diagrama de secuencia se basa en el caso de uso “Ingresar Grupo de Trabajo” de la figura 2. La secuencia de actividades es la siguiente:

1. El actor “jefe de proyecto” selecciona un proyecto existente.
2. La página jsp denominada ing_grupo_trabajo llama al método de la lógica del negocio denominado guardar_grupo.
3. A su vez la clase grupo de la capa de negocios llama al método guardar_grupo de la clase dao_grupo_trabajo de la capa acceso a datos.
4. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “jefe de proyecto” que comenzó la secuencia.

8.1.4 Diagrama de secuencia de ingresar nuevas tareas a un proyecto

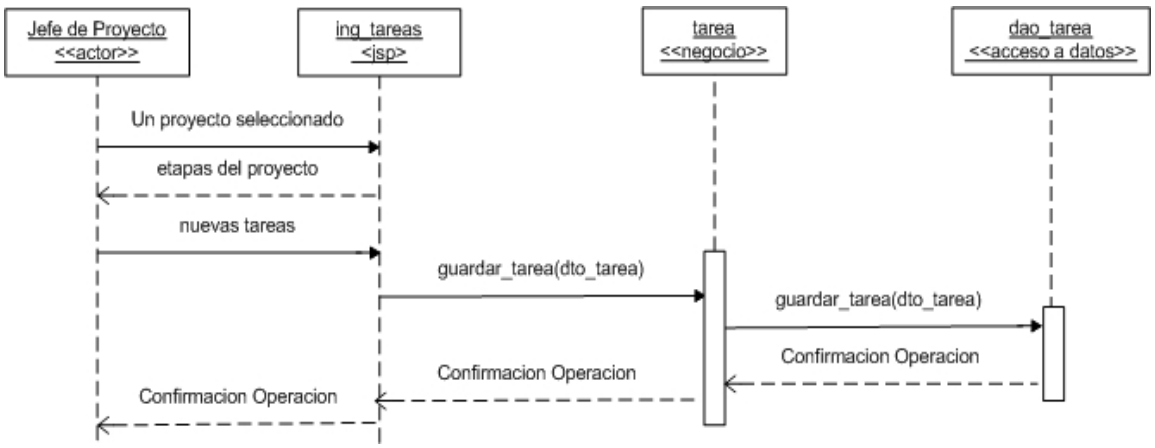


Figura 28: Diagrama de secuencia de ingresar tareas

En la figura 28 se muestra el diagrama la secuencia para ingresar nuevas tareas a un determinado proyecto existente previamente en el sistema. Este diagrama se basa en el caso de uso “Ingresar Tareas al Proyecto” de la figura 2. El detalle de la secuencia es el siguiente:

1. El actor “jefe de proyecto” selecciona un proyecto existente.

2. En la página jsp denominada ing_tarea se muestran las etapas existentes del proyecto.

3. A su vez la clase grupo de la capa de negocios llama al método guardar_tarea de la clase dao_tarea de la capa acceso a datos.

4. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “jefe de proyecto” que comenzó la secuencia.

8.1.5 Diagrama de secuencia de ingresar nuevos documentos a crear a un proyecto

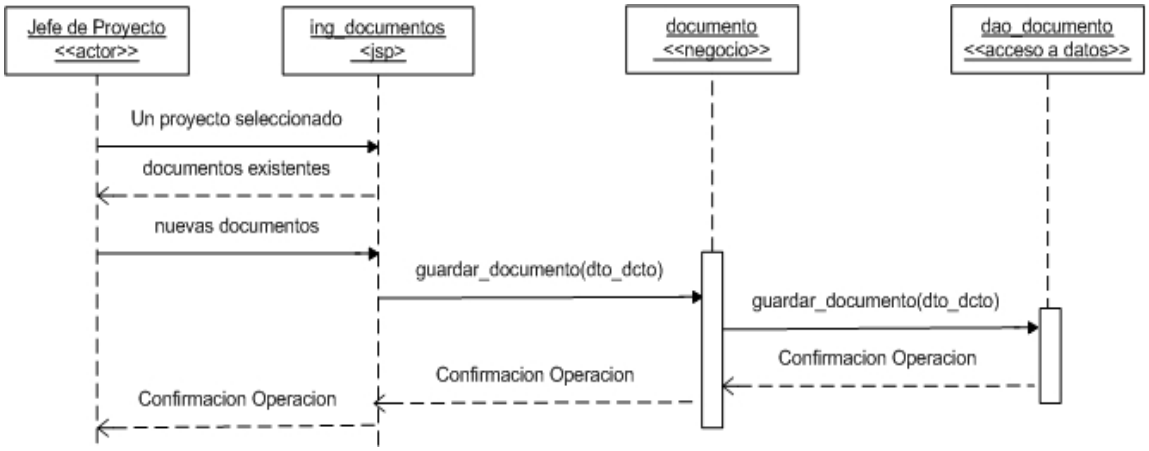


Figura 29: Diagrama de secuencia de ingresar documentos a crear

En la figura 29 se muestra el diagrama la secuencia para ingresar nuevos documentos a un determinado proyecto existente. Este diagrama se basa en el caso de uso “Ingresar Nombre Documentos a Crear” de la figura 2. El detalle de la secuencia es el siguiente:

1. El actor “jefe de proyecto” selecciona un proyecto existente.

2. En la página jsp denominada ing_documentos se muestran los datos de documentos existentes y se llama al método guardar_documento perteneciente a la clase documento de la capa de negocios.

3. A su vez la clase documento de la capa de negocios llama al método guardar_documento de la clase dao_documento de la capa acceso a datos.

4. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “jefe de proyecto” que comenzó la secuencia.

8.1.6 Diagrama de secuencia de ingresar nuevas documentos de apoyo a un proyecto

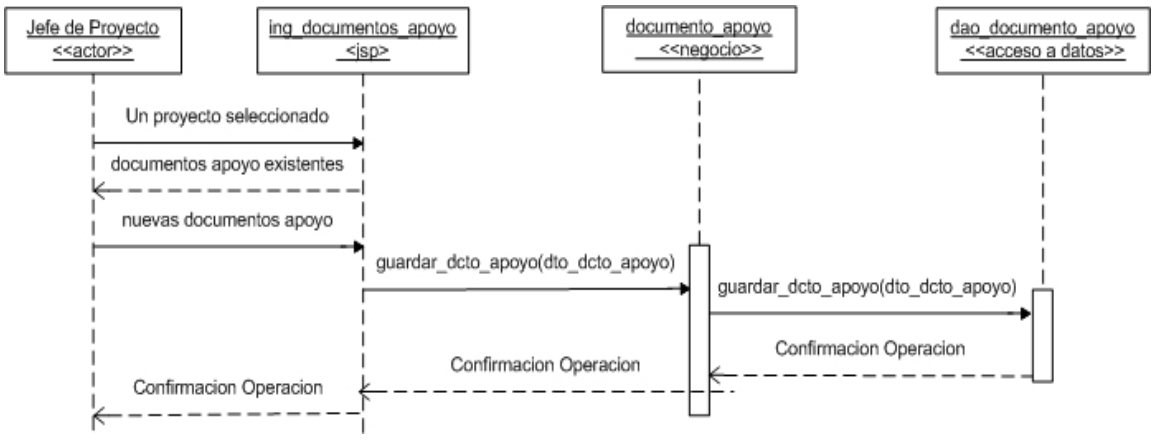


Figura 30: Diagrama de secuencia de documentos de apoyo

En la figura 30 se puede ver el diagrama la secuencia para ingresar documentos de apoyo a un determinado proyecto existente. Este diagrama se basa en el caso de uso “Ingresar Nombre Documentos de apoyo” de la figura 2. El detalle de la secuencia es el siguiente:

1. El actor “jefe de proyecto” selecciona un proyecto existente.
2. En la página jsp denominada ing_documentos_apoyo se muestran los datos de los documentos de apoyo existentes y se llama al método guardar_dcto_apoyo perteneciente a la clase documento_apoyo de la capa de negocios.
3. A su vez la clase documentos_apoyo de la capa de negocios llama al método guardar_dcto_apoyo de la clase dao_documento_apoyo de la capa acceso a datos.
4. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “jefe de proyecto” que comenzó la secuencia.

8.1.7 Diagrama de secuencia de ingreso de nuevas versiones a un documento

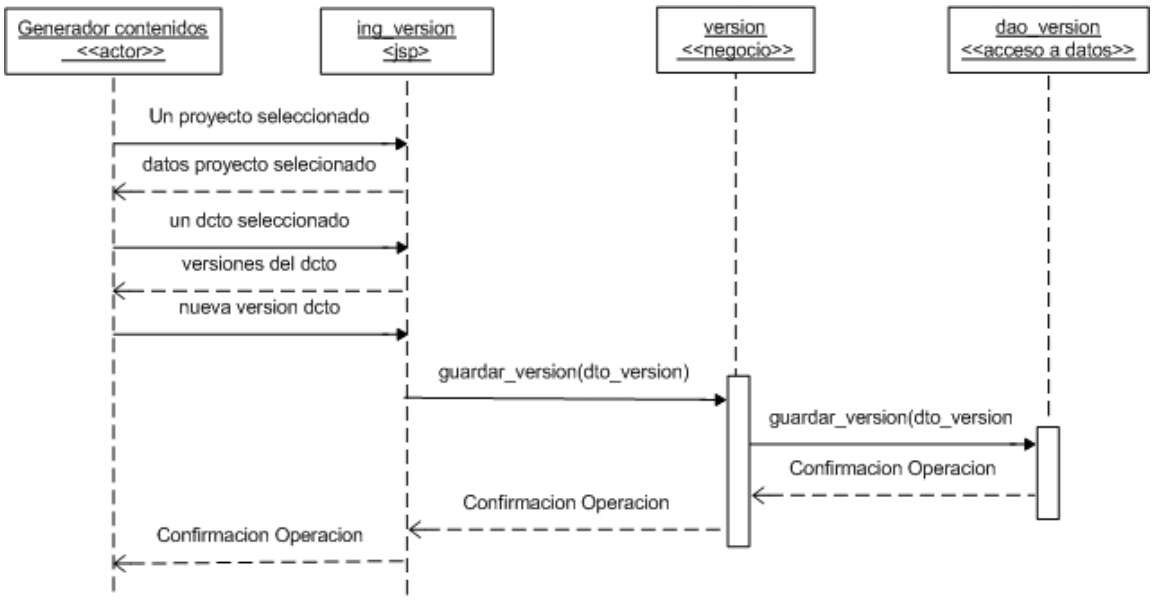


Figura 31: Diagrama de secuencia de ingreso de una nueva versión de documento

En la figura 31 se muestra la secuencia para ingresar nuevas versiones a un determinado documento existente en el sistema. Este diagrama se basa en el caso de uso “Ingreso Versiones del dcto” de la figura 2. El detalle de la secuencia es la siguiente:

1. El actor “generador de contenidos” selecciona un proyecto existente.
2. En la página jsp denominada ing_version se muestran los datos del proyecto seleccionado incluyendo los documentos que pertenecen a él

3. El actor “generador de contenidos” selecciona un documento existente y la página jsp le devolverá todas las versiones existentes del documento seleccionado.

4. El actor “generador de contenidos” ingresa una nueva versión del documento y la página jsp denominada ing_versión llama al método guardar_versión que pertenece a la clase versión de la capa de negocios.

5. A su vez en la clase versión de la capa de negocios llama al método guardar_versión de la capa de acceso a datos.

6. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “generador de contenidos” que comenzó la secuencia.

8.1.8 Diagrama de secuencia de aprobación de un documento

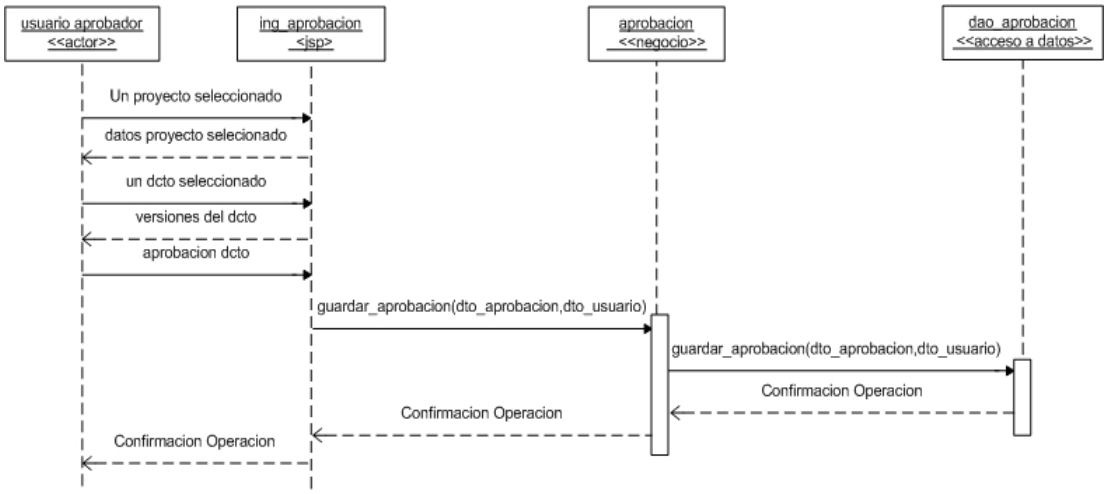


Figura 32: Diagrama de secuencia de ingreso de aprobación de un documento

En la figura 32 se muestra la secuencia de ingreso de la aprobación de un documento. Este diagrama se basa en el caso de uso “Aprobación de documentos” de la figura 2. Los pasos de la secuencia son los siguientes:

1. El actor “usuario aprobador” selecciona un proyecto existente.

2. En la página jsp denominada ing_aprobación se muestran los datos del proyecto seleccionado incluyendo los documentos que pertenecen a él.

3. El actor “usuario aprobador” selecciona un documento existente y la página jsp le devolverá todas las versiones existentes del documento seleccionado.

4. El actor “usuario aprobador” ingresa la aprobación del documento y la página jsp denominada ing_versión llama al método guardar_aprobación que pertenece a la clase aprobación de la capa de negocios.

5. A su vez en la clase aprobación de la capa de negocios llama al método guardar_aprobación de la capa de acceso a datos.

6. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “usuario aprobador” que comenzó la secuencia.

8.1.9 Diagrama de secuencia de acceso a documentos en biblioteca

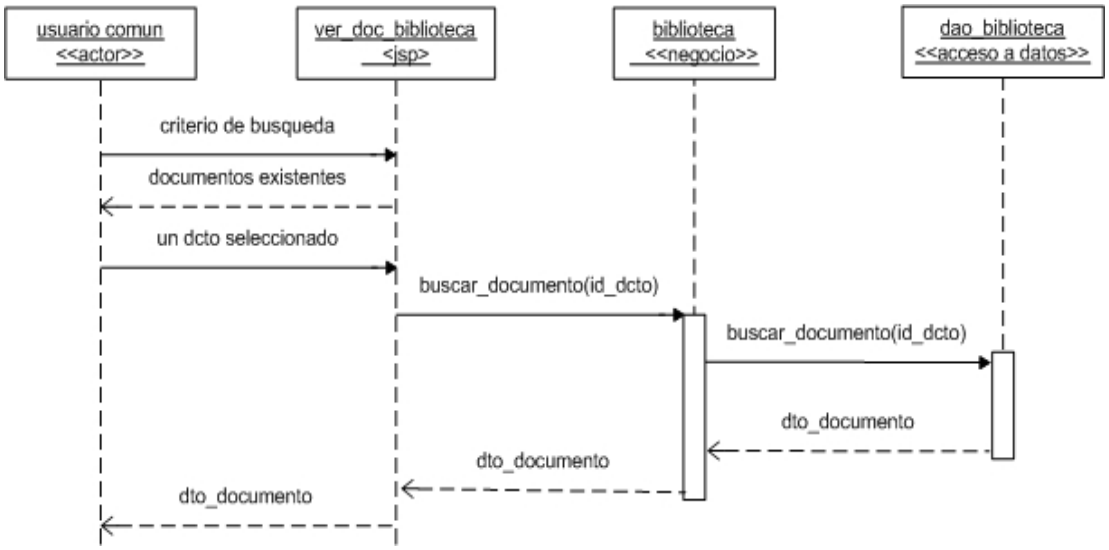


Figura 33: Diagrama de secuencia de acceso a documentos en biblioteca

En la figura 33 se muestra la secuencia para acceder a un documento ya publicado en biblioteca. Este diagrama se basa en el caso de uso “Ver Documentos” de figura 2. Los pasos de la secuencia son los siguientes:

1. El actor “usuario común” selecciona un criterio de búsqueda para documentos existentes en biblioteca.
2. En la página jsp denominada ver_doc_biblioteca se muestran los datos de los documentos encontrados de acuerdo al criterio de búsqueda seleccionado.

3. El actor “usuario aprobador” selecciona un documento y la página jsp denominada ver_doc_biblioteca llama al método buscar_documento que pertenece a la clase biblioteca de la capa de negocios.

4. A su vez en la clase biblioteca de la capa de negocios llama al método ver_doc_biblioteca de la capa de acceso a datos.

6. Se devuelve el documento encontrado a través de las capas, para que finalmente lleguen al actor “usuario común” que comenzó la secuencia.

8.1.10 Diagrama de secuencia de ingreso de documentos en biblioteca

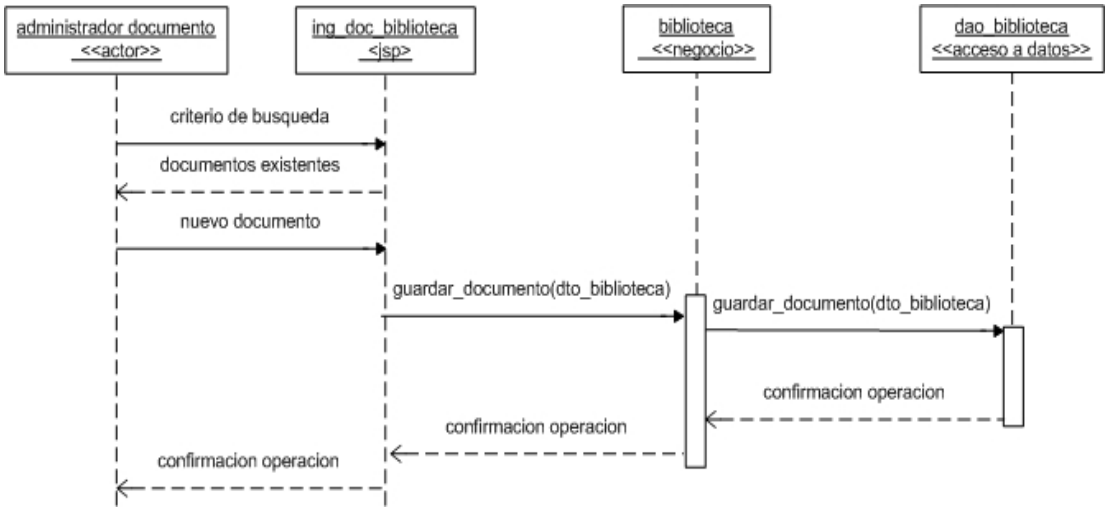


Figura 34: Diagrama de secuencia de ingreso de documentos en biblioteca

En la figura 34 muestra la secuencia para ingresar un nuevo documento a la biblioteca. Este diagrama está basado en el caso de uso “Subir Documentos externos a biblioteca” de la figura 3. Los pasos de la secuencia son los siguientes:

1. El actor “administrador documentos” selecciona un criterio de búsqueda para documentos existentes en biblioteca.
2. En la página jsp denominada ing_doc_biblioteca se muestran los datos de los documentos encontrados de acuerdo al criterio de búsqueda seleccionado.
3. El actor “administrador documentos” ingresa un nuevo documento y la página jsp denominada ing_doc_biblioteca llama al método guardar_documento que pertenece a la clase biblioteca de la capa de negocios.
4. A su vez en la clase biblioteca, de la capa de negocios, llama al método ing_doc_biblioteca de la capa de acceso a datos.
5. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “administrador documentos” que comenzó la secuencia.

8.1.11 Diagrama de secuencia de eliminación de documentos en biblioteca

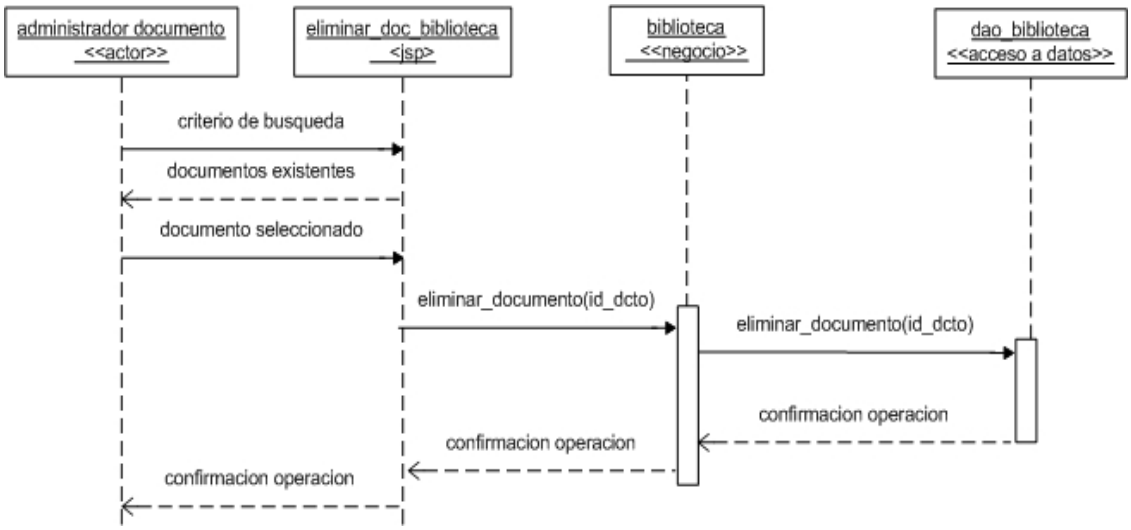


Figura 35: Diagrama de secuencia de eliminación de documentos en biblioteca

En la figura 35 se muestra el diagrama de secuencia que permite eliminar documentos de la biblioteca. Está basado en el caso de uso “Eliminar documentos existentes” de la figura 3. El detalle de la secuencia es el siguiente:

1. El actor “administrador documentos” selecciona un criterio de búsqueda para documentos existentes en biblioteca.

2. En la página jsp denominada eliminar_doc_biblioteca se muestran los datos de los documentos encontrados de acuerdo al criterio de búsqueda seleccionado.

3. El actor “administrador documentos” selecciona un documento y la página jsp denominada eliminar_doc_biblioteca llama al método eliminar_documento que pertenece a la clase biblioteca de la capa de negocios.

4. A su vez en la clase biblioteca de la capa de negocios llama al método eliminar_documento de la capa de acceso a datos.

5. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “administrador documentos” que comenzó la secuencia.

8.1.12 Diagrama de secuencia de modificación de documentos en biblioteca

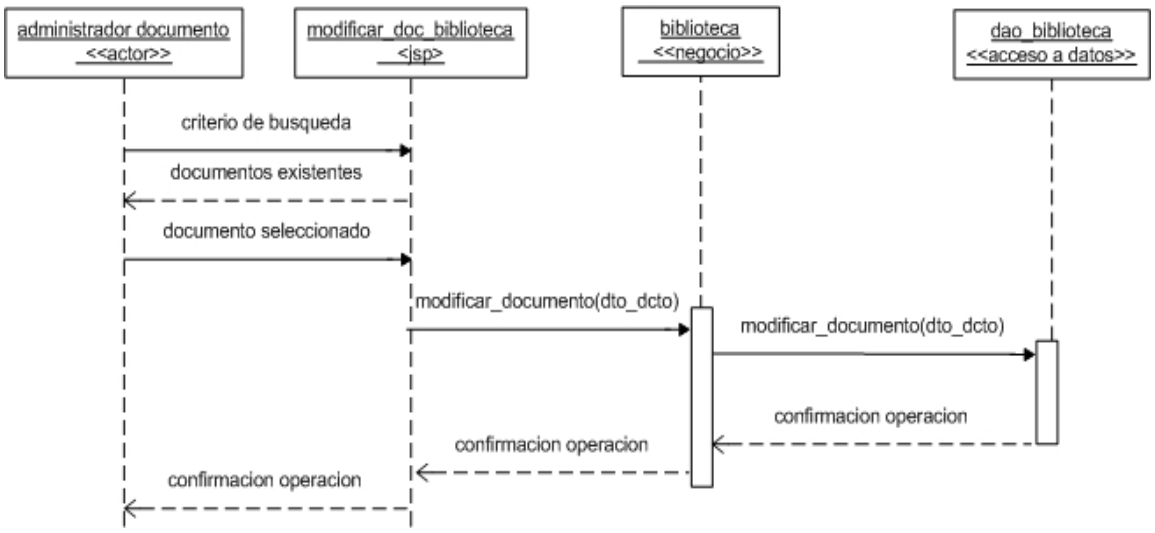


Figura 36: Diagrama de secuencia de modificación de documentos en biblioteca

En la figura 36 se muestra el diagrama de secuencia que representa la modificación de documento de la biblioteca. Este diagrama está basado en el caso de uso “Modificación de Documentos” de la figura 3. El detalle de la secuencia es la siguiente:

1. El actor “administrador documentos” selecciona un criterio de búsqueda para documentos existentes en biblioteca.

2. En la página jsp denominada modificar_doc_biblioteca se muestran los datos de los documentos encontrados de acuerdo al criterio de búsqueda seleccionado.

3. El actor “administrador documentos” selecciona un documento y la página jsp denominada modificar_doc_biblioteca llama al método modificar_documento que pertenece a la clase biblioteca de la capa de negocios.

4. A su vez en la clase biblioteca de la capa de negocios llama al método modificar_documento de la capa de acceso a datos.

5. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “administrador documentos” que comenzó la secuencia.

8.1.13 Diagrama de secuencia de ingreso de un nuevo control

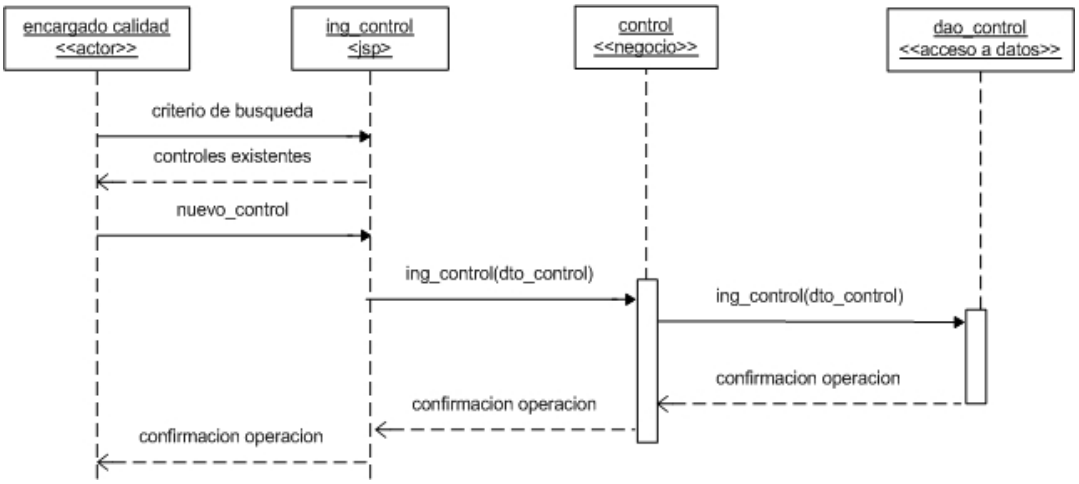


Figura 37: Diagrama de secuencia de ingreso de un nuevo control

En la figura 37 se muestra el diagrama de secuencias de ingreso de un nuevo control de calidad al sistema. Este diagrama está basado en el caso de uso “Crea Control” de la figura 5. Los pasos a seguir en la secuencia son los siguientes:

1. El actor “encargado calidad” selecciona un criterio de búsqueda para controles existentes en el sistema
2. En la página jsp denominada ing_control se muestran los controles encontrados de acuerdo al criterio de búsqueda seleccionado.

3. El actor “encargado calidad” ingresa un nuevo control y la página jsp denominada ing_control llama al método ing_control que pertenece a la clase control de la capa de negocios.

4. A su vez en la clase control de la capa de negocios llama al método ing_control perteneciente a la clase dao_control de la capa de acceso a datos.

5. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “encargado calidad” que comenzó la secuencia.

8.1.14 Diagrama de secuencia de ingreso de una nueva planificación

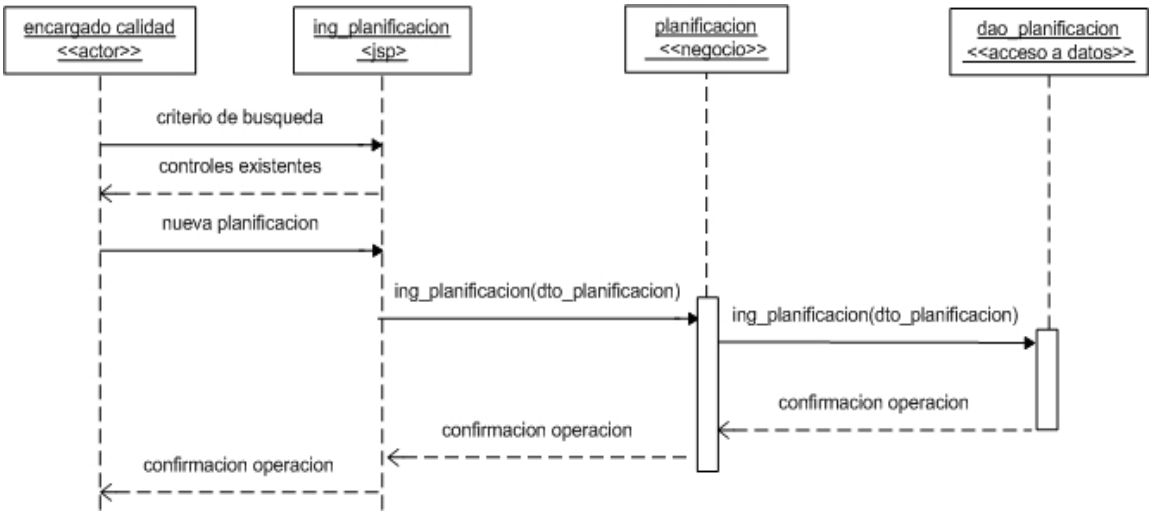


Figura 38: Diagrama de secuencia de ingreso de una nueva planificación

En la figura 38 se muestra el diagrama de secuencia del ingreso de una nueva planificación al sistema. Este diagrama se basa en el caso de uso “Planificar un control” de la figura 5. El detalle de la secuencia es el siguiente:

1. El actor “encargado calidad” selecciona un criterio de búsqueda para controles existentes en el sistema
2. En la página jsp denominada ing_planificación se muestran los controles encontrados de acuerdo al criterio de búsqueda seleccionado.
3. El actor “encargado calidad” ingresa una nueva planificación y la página jsp denominada ing_planificación llama al método ing_planificación que pertenece a la clase planificación de la capa de negocios.
4. A su vez en la clase planificación de la capa de negocios llama al método ing_planificación perteneciente a la clase dao_planificación de la capa de acceso a datos.
5. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “encargado calidad” que comenzó la secuencia.

8.1.15 Diagrama de secuencia de ingreso de resultados a una planificación

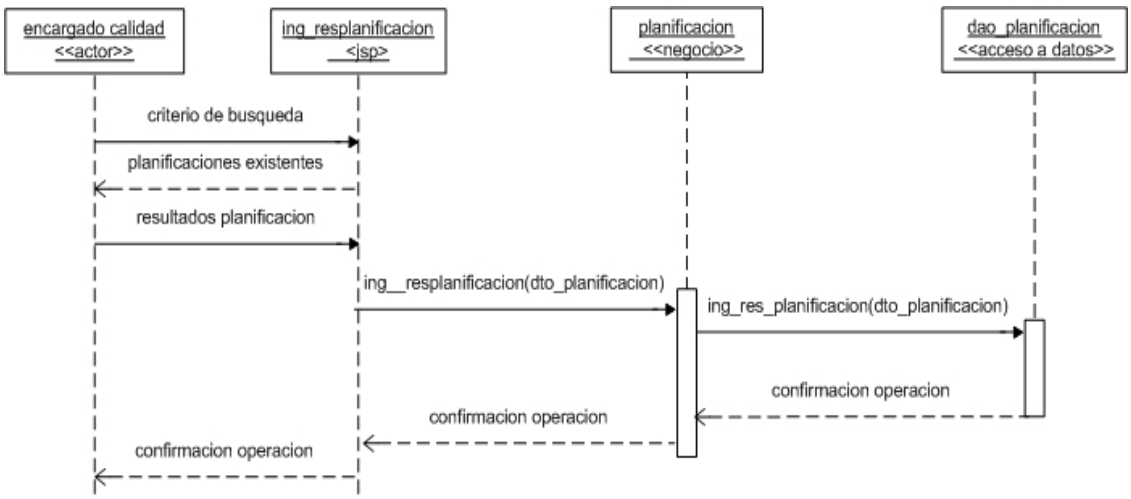


Figura 39: Diagrama de secuencia de ingreso de resultados a una planificación

En la figura 39 muestra el diagrama de secuencia que representa el ingreso de resultados a una planificación existente en el sistema. Está basado en el caso de uso “Ingreso de resultados planificación” de la figura 5 y su detalle es el siguiente:

1. El actor “encargado calidad” selecciona un criterio de búsqueda para planificaciones existentes en el sistema
2. En la página jsp denominada ing_resplanificacion se muestran las planificaciones encontradas de acuerdo al criterio de búsqueda seleccionado.

3. El actor “encargado calidad” ingresa los resultados a la planificación y la página jsp denominada ing_resplanificación llama al método ing_planificación que pertenece a la clase planificación de la capa de negocios.

4. A su vez en la clase planificación de la capa de negocios llama al método ing_resplanificación perteneciente a la clase dao_planificación de la capa de acceso a datos.

5. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “encargado calidad” que comenzó la secuencia.

8.1.16 Diagrama de secuencia de ingreso de una nueva acción correctiva

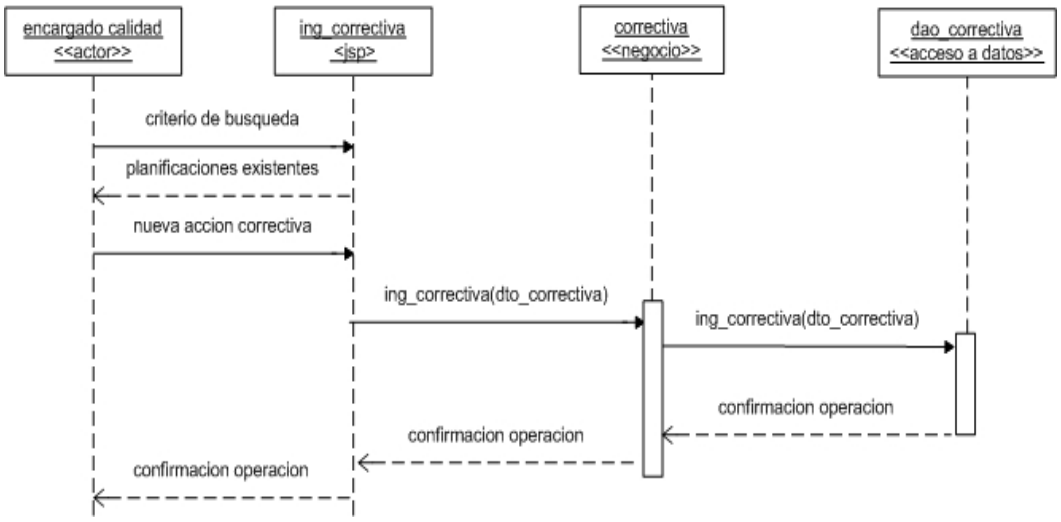


Figura 40: Diagrama de secuencia de ingreso de resultados a una planificación

La figura 40 representa el diagrama de secuencia del caso de uso “Ingreso resultados planificación” de la figura 5. El detalle de la secuencia es el siguiente:

1. El actor “encargado calidad” selecciona un criterio de búsqueda para planificaciones existentes en el sistema.

2. En la página jsp denominada ing_correctiva se muestran las planificaciones encontradas de acuerdo al criterio de búsqueda seleccionado.

3. El actor “encargado calidad” ingresa las acciones correctivas a la planificación y la página jsp denominada ing_correctiva llama al método ing_correctiva que pertenece a la clase correctiva de la capa de negocios.

4. A su vez en la clase correctiva de la capa de negocios llama al método ing_correctiva perteneciente a la clase dao_correctiva de la capa de acceso a datos.

5. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “encargado calidad” que comenzó la secuencia.

8.1.17 Diagrama de secuencia de ingreso de resultados de una acción correctiva

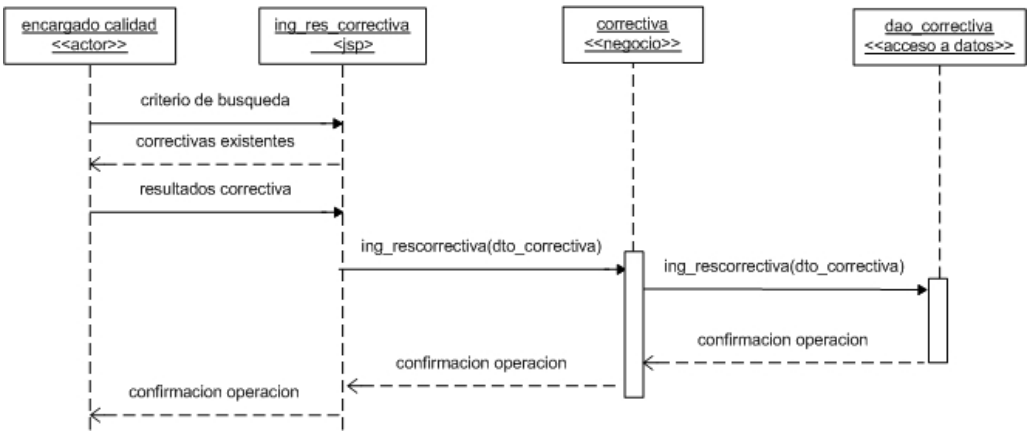


Figura 41: Diagrama de secuencia de ingreso de resultados a una acción correctiva

En el diagrama 41 se muestra la secuencia de ingreso de resultados de una acción correctiva ingresada previamente al sistema. Este diagrama está basado en el caso de uso “Ingreso de resultados acciones correctivas” y su detalle es el siguiente:

1. El actor “encargado calidad” selecciona un criterio de búsqueda para acciones correctivas existentes en el sistema
2. En la página jsp denominada ing_res_correctiva se muestran las acciones correctivas encontradas de acuerdo al criterio de búsqueda seleccionado.
3. El actor “encargado calidad” ingresa los resultados a la acción correctiva seleccionada y la página jsp denominada ing_res_correctiva llama al método ing_rescorrectiva que pertenece a la clase correctiva de la capa de negocios.
4. A su vez en la clase correctiva de la capa de negocios llama al método ing_rescorrectiva perteneciente a la clase dao_correctiva de la capa de acceso a datos.
5. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “encargado calidad” que comenzó la secuencia.

8.1.18 Diagrama de secuencia de ingreso nuevas áreas

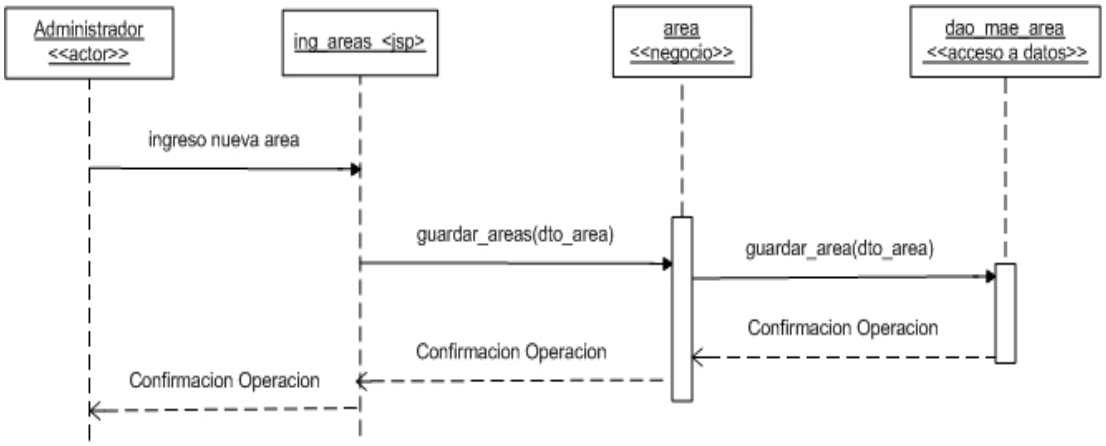


Figura 42: Diagrama de secuencia de ingreso áreas

En la figura 42 se puede ver el diagrama de secuencia que representa el ingreso de nuevas áreas al sistema. Está basado en el caso de uso “Ingresa Área” de la figura 6 y los pasos de la secuencia son los siguientes:

1. El actor “Administrador” ingresa una nueva área y la página jsp denominada ing_área llama al método guardar_área que pertenece a la clase área de la capa de negocios.
2. A su vez en la clase área de la capa de negocios llama al método guardar_área perteneciente a la clase dao_área de la capa de acceso a datos.
3. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia.

8.1.19 Diagrama de secuencia de ingreso nuevos usuarios

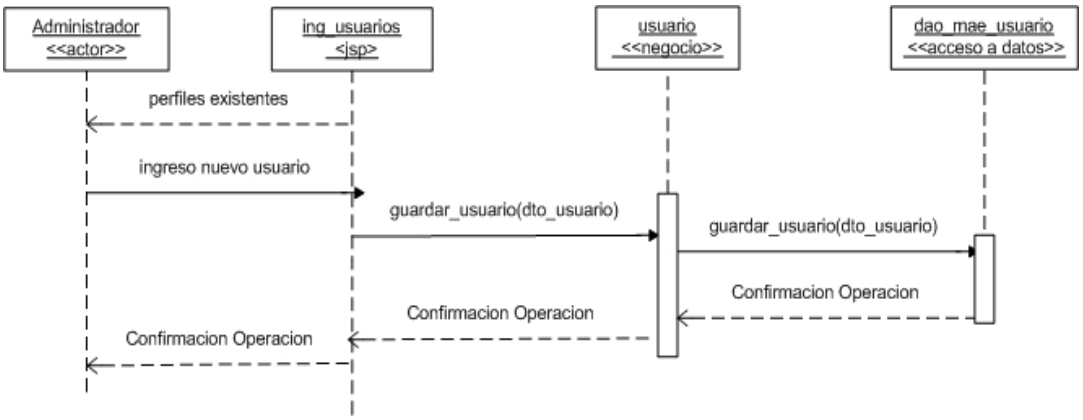


Figura 43: Diagrama de secuencia de ingreso usuarios

En la figura 43 se puede ver el diagrama de secuencia que representa el ingreso de nuevos usuarios al sistema. Está basado en el caso de uso “Ingresa Usuarios” de la figura 6 y los pasos de la secuencia son los siguientes:

1. En la página jsp denominada ing_usuario se muestran los perfiles disponibles para ingresar nuevos usuarios.
2. El actor “Administrador” ingresa un nuevo usuario (incluido el perfil al que pertenece) y la página jsp denominada ing_usuario llama al método guardar_usuario que pertenece a la clase usuario de la capa de negocios.
3. A su vez en la clase usuario de la capa de negocios llama al método guardar_usuario perteneciente a la clase dao_usuario de la capa de acceso a datos.

4. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia.

8.1.20 Diagrama de secuencia de ingreso nuevos departamentos

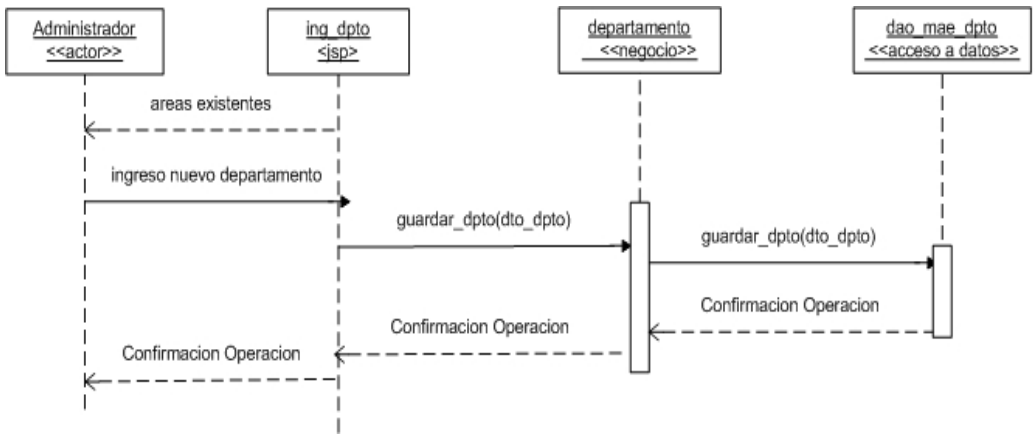


Figura 44: Diagrama de secuencia de ingreso de departamentos

En la figura 44 se muestra el diagrama de secuencia que representa el ingreso de nuevos departamentos al sistema. Está basado en el caso de uso “Ingresa Departamentos” de la figura 6 y los pasos de la secuencia son los siguientes:

1. En la página jsp denominada ing_dpto se muestran las áreas disponibles para ingresar nuevos departamentos.

2. El actor “Administrador” ingresa un nuevo departamento (incluido el departamento al que pertenece) y la página jsp denominada ing_dpto llama

al método guardar_dpto que pertenece a la clase usuario de la capa de negocios.

3. A su vez en la clase usuario de la capa de negocios se llama al método guardar_dpto perteneciente a la clase dao_dpto de la capa de acceso a datos.

4. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia.

8.1.21 Diagrama de secuencia de ingreso nuevos tipos de proyecto

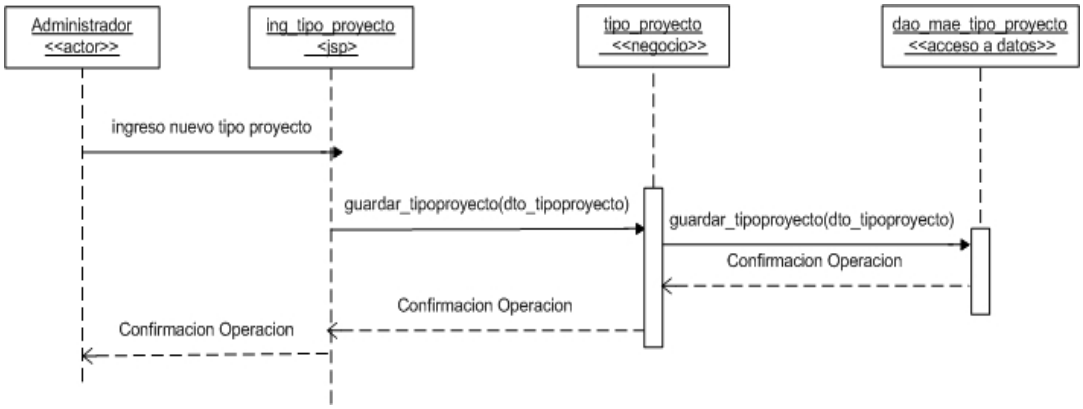


Figura 45: Diagrama de secuencia de ingreso de tipos de proyectos

En la figura 45 se muestra el diagrama de secuencia que representa el ingreso de nuevos tipos de proyecto al sistema. Está basado en el caso de uso

“Ingresa Tipos de Proyectos” de la figura 6 y los pasos de la secuencia son los siguientes:

1. El actor “Administrador” ingresa un nuevo tipo de proyecto y la página jsp denominada ing_tipo_proyecto llama al método guardar_tipoproyecto que pertenece a la clase tipoproyecto de la capa de negocios.

2. A su vez en la clase tipoproyecto de la capa de negocios llama al método guardar_tipoproyecto perteneciente a la clase dao_mae_tipo_proyecto de la capa de acceso a datos.

3. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia.

8.1.22 Diagrama de secuencia de ingreso nuevos perfiles de usuario

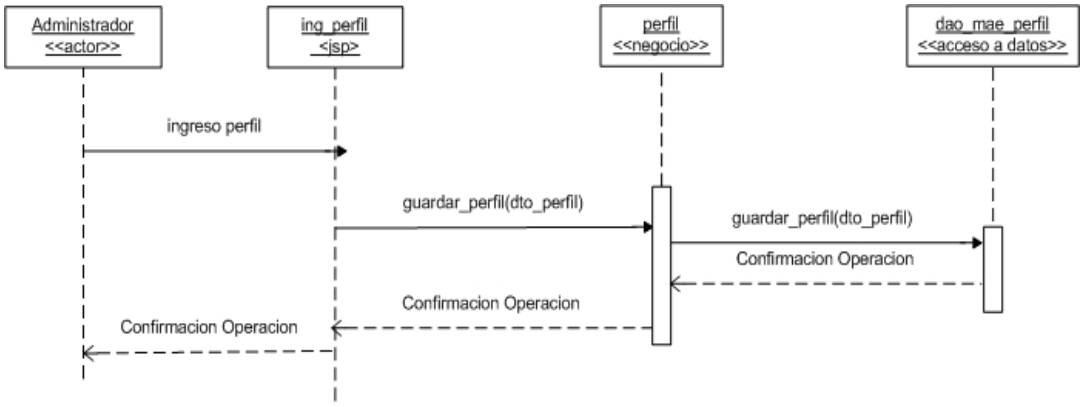


Figura 46: Diagrama de secuencia de ingreso de perfiles de usuario

En la figura 46 se muestra el diagrama de secuencia que representa el ingreso de nuevos perfiles de usuario al sistema. Está basado en el caso de uso “Ingresa Perfiles y Permisos” de la figura 6 y los pasos de la secuencia son los siguientes:

1. El actor “Administrador” ingresa un perfil y la página jsp denominada ing_perfil llama al método guardar_perfil que pertenece a la clase perfil de la capa de negocios.
2. A su vez en la clase perfil de la capa de negocios llama al método guardar_perfil perteneciente a la clase dao_mae_perfil de la capa de acceso a datos.
3. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia.

8.1.23 Diagrama de secuencia de ingreso permisos de perfiles a páginas

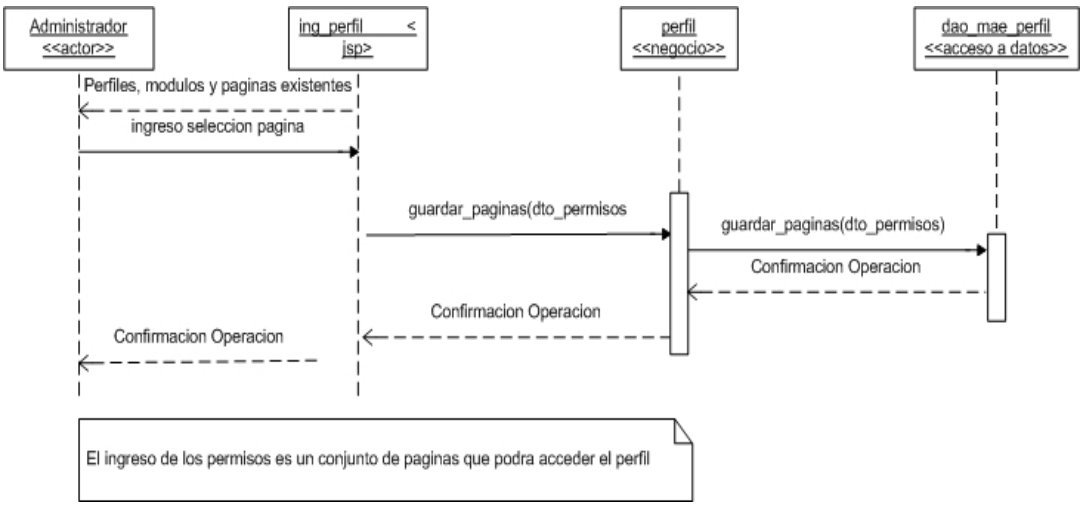


Figura 47: Diagrama de secuencia de ingreso de permisos de perfiles a páginas

En la figura 47 se puede ver el diagrama de secuencia que representa el ingreso permisos de acceso de un determinado perfil a las diferentes páginas del sistema. Está basado en el caso de uso “Ingresa Perfiles y Permisos” de la figura 6 y los pasos de la secuencia son los siguientes:

1. La página jsp denominada ing_perfil muestra los perfiles existentes en el sistema así como también los módulos y las páginas.
2. El actor “Administrador” selecciona un perfil existente y un conjunto de páginas que desea permitir el ingreso.

3. A su vez en la clase perfil de la capa de negocios llama al método guardar_páginas perteneciente a la clase dao_mae_perfil de la capa de acceso a datos.

3. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia.

8.1.24 Diagrama de secuencia de ingreso nuevos centros de cultivo

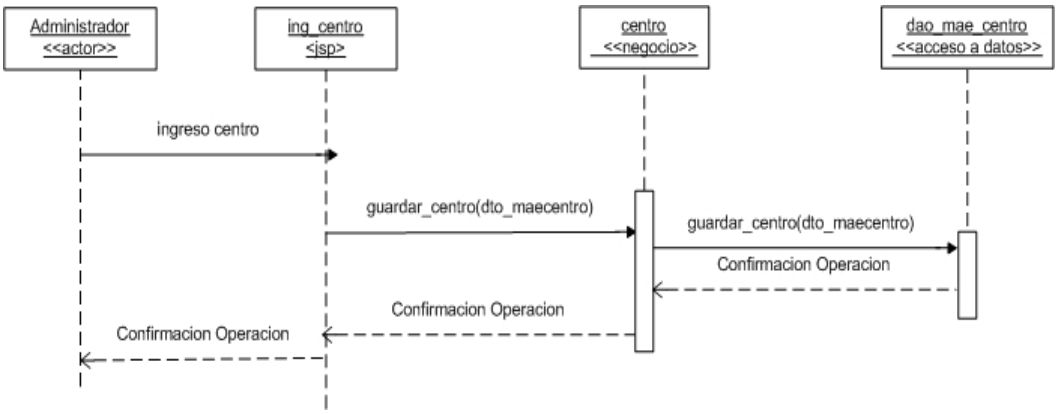


Figura 48: Diagrama de secuencia de ingreso de centros de cultivo

En la figura 48 se muestra el diagrama de secuencia que representa el ingreso de centros de cultivo al sistema. Está basado en el caso de uso “Ingresa Centros Cultivo” de la figura 6 y los pasos de la secuencia son los siguientes:

1. El actor “Administrador” ingresa un centro de cultivo y la página jsp denominada ing_centro llama al método guardar_centro que pertenece a la clase centro de la capa de negocios.

2. A su vez en la clase centro de la capa de negocios llama al método guardar_centro perteneciente a la clase dao_mae_centro de la capa de acceso a datos.

3. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia.

8.1.25 Diagrama de secuencia de ingreso nuevos cargos

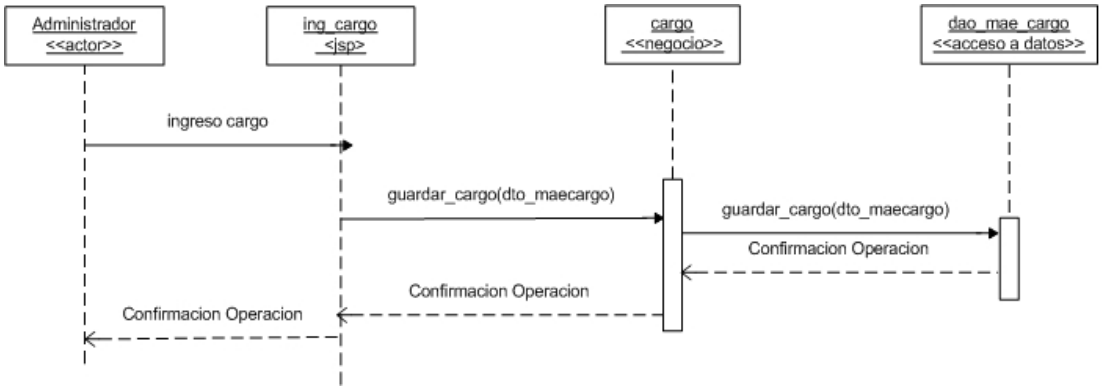


Figura 49: Diagrama de secuencia de ingreso de cargos

En la figura 49 se muestra el diagrama de secuencia que representa el ingreso de cargos al sistema. Está basado en el caso de uso “Ingresa Cargos” de la figura 6 y los pasos de la secuencia son los siguientes:

1. El actor “Administrador” ingresa un cargo y la página jsp denominada ing_cargo llama al método guardar_cargo que pertenece a la clase cargo de la capa de negocios.

2. A su vez en la clase cargo de la capa de negocios llama al método guardar_cargo perteneciente a la clase dao_mae_cargo de la capa de acceso a datos.

3. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia

8.1.26 Diagrama de secuencia de ingreso tipos de variables

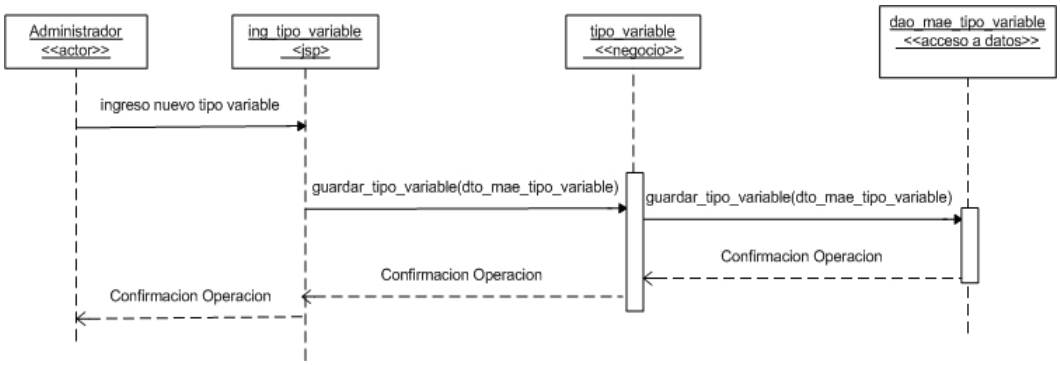


Figura 50: Diagrama de secuencia de ingreso de tipos de variables

En la figura 50 se muestra el diagrama de secuencia que representa el ingreso de tipos de variables al sistema. Está basado en el caso de uso “Ingresa Tipo de Variables” de la figura 6 y el detalle de secuencia son los siguientes:

1. El actor “Administrador” ingresa un nuevo tipo de variables y la página jsp denominada ing_tipo_variable llama al método guardar_tipo_variable que pertenece a la clase tipo_variable de la capa de negocios.
2. A su vez en la clase tipo_variable de la capa de negocios se llama al método guardar_tipo_variable perteneciente a la clase dao_mae_tipo_variable de la capa de acceso a datos.
3. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia

8.1.27 Diagrama de secuencia de ingreso de variables

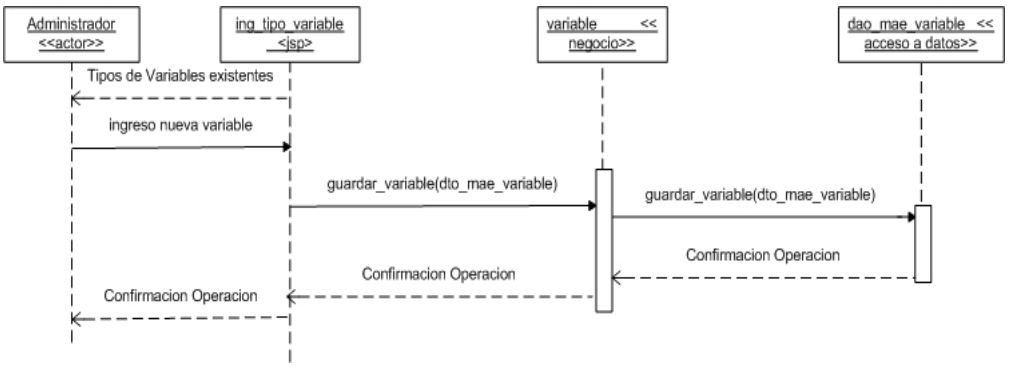


Figura 51: Diagrama de secuencia de ingreso de variables

En la figura 51 se muestra el diagrama de secuencia que representa el ingreso de variables de control al sistema. Está basado en el caso de uso “Ingresa Variables” de la figura 6 y el detalle de secuencia son los siguientes:

1. En la página denominada ing_tipo_variable se muestran los tipos de variables existentes en el sistema
2. El actor “Administrador” ingresa una nueva variable (seleccionando el tipo de variable al que pertenece) y la página jsp denominada ing_variable llama al método guardar _variable que pertenece a la clase variable de la capa de negocios.
3. A su vez en la clase variable de la capa de negocios se llama al método guardar _variable perteneciente a la clase dao_mae_ variable de la capa de acceso a datos.

4. Se devuelven los mensajes de confirmación a través de las capas, para que finalmente lleguen al actor “Administrador” que comenzó la secuencia.

8.2 Modelo de Diseño

8.2.1 Diagrama de Clases de diseño

Basándose en el modelo del dominio inicial representado en la figura 7 y en los distintos diagramas de secuencias identificados en el punto anterior se realiza el modelo de clases de software de la capa de negocios utilizando notación UML y basándose en la perspectiva conceptual [Fowler1999]. En este diagrama (representado en la figura 52) se especifican la multiplicidad de las asociaciones y las clases de la capa de negocios.

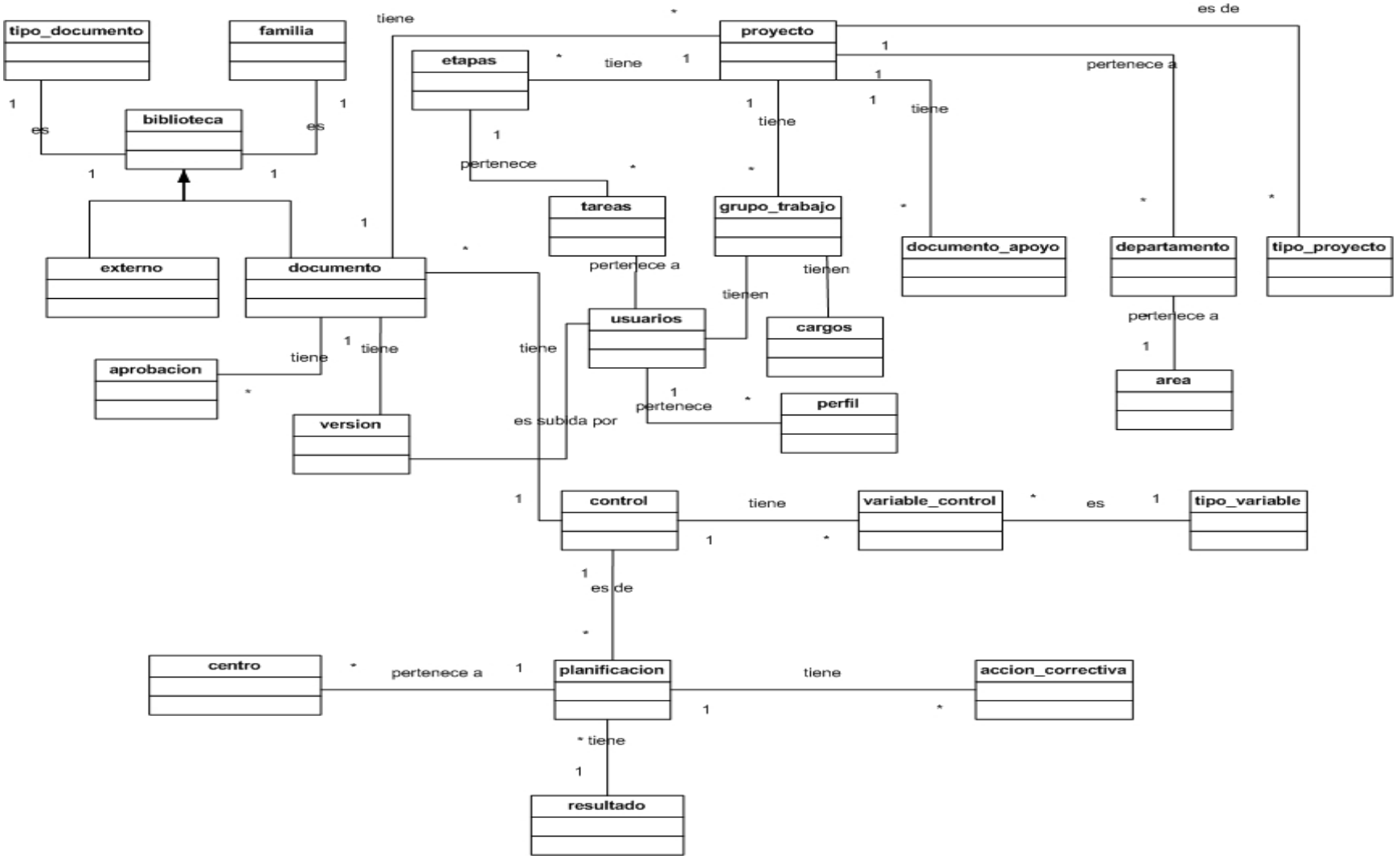


Figura 52: Diagrama de Clases de Software de la capa de negocios

Para complementar la documentación del diagrama de clases se muestra una breve descripción de las clases del diagrama representado en la Figura 50:

Nombre clase	**Descripción**
proyecto	Representa a un proyecto que permite la creación de nuevos documentos.
etapas	Representa a etapas que son usadas como maestros del sistema
grupo_trabajo	Corresponde a un conjunto de usuarios que pertenecen a un proyecto.
tarea	Corresponde a la asignación de una responsabilidad a un usuario dentro de cada proyecto.
documento_apoyo	Corresponde a documentos agregados como apoyo a un proyecto.
departamento	Representa un departamento de la empresa.
tipo_proyecto	Representa los tipos de proyectos que puedan existir en el sistema.

área	Es un área de la empresa que a su vez tiene departamentos.
cargos	Son cargos que puede tener un usuario dentro de un proyecto.
usuario	Representa a una persona dentro del sistema.
perfil	Corresponde a un tipo de usuario.
familia	Es un tipo de clasificación de los documentos tanto externos como internos.
tipo_documento	Es otro tipo de clasificación de los documentos tanto externos como internos.
biblioteca	Es una clase abstracta que permite implementar el patrón Abstract Factory.
externo	Representa un documento agregado al sistema de manera directa.
documento	Representa un documento que debe ser creado con la ayuda del sistema.
versión	Corresponde a una versión de un documento.
aprobación	Corresponde a la aprobación de

	parte de un usuario a un determinado documento.
centro	Representa a un centro de cultivo que pertenece a la empresa.
control	Representa un conjunto de variables que se miden de acuerdo a una planificación.
resultado	Representa los resultados de cada variable de un determinado control.
planificación	Es un control que tiene hora, fecha y responsable para ser medido.
variable_control	Representa una variable para ser medida.
tipo_variable	Representa una clasificación de las variables de control.
acción_correctiva	Es una acción que se estima necesaria para corregir procesos y que se determina después de una planificación.

Tabla 14: Detalle modelo de clases

8.3 Diseño de aspectos

Para incorporar los aspectos en la etapa de diseño se estudiaron distintas propuestas, sin embargo, todas tienen en común utilizar de una u otra manera el lenguaje de modelo unificado (UML).

- Propuesta de Suzuki y Yamamoto [Suzuki1999]
- Propuesta de Herrero [HERRERO2000]
- Propuesta de Barra, Genova y Llorens [Barra2004]
- Propuesta de Bustos y Eterovic [Bustos2007]
- Propuesta de Basch y Sanchez [Bash2003]

8.3.1 Propuesta elegida de Suzuki y Yamamoto.

Debido su facilidad de comprensión, simplicidad, cantidad de documentación y ejemplos encontrados se decidió utilizar la propuesta de Suzuki y Yamamoto.

Esta propuesta consisten en que así como existen clases, interfaces, nodos, y componentes se puede introducir de un nuevo elemento, en UML, llamado “aspecto”. Se reconocen los aspectos como “elementos que describen características de comportamiento y estructurales”. De esta forma un aspecto puede tener cualquier número de atributos y operaciones y puede participar en múltiples asociaciones, generalizaciones y relaciones de dependencia.

Luego de enlazar los aspectos con las clases afectadas en el diagrama de clases se recomienda representar además esta relación en un diagrama de paquetes, mostrando que clases son afectadas por los aspectos definidos [Nieto].

Para hacer los diagramas de la propuesta se debe primero definir algunos elementos teóricos de la programación orientada a aspectos.

8.3.2 Consideraciones teóricas de Programación Orientada a aspectos

Los aspectos describen apéndices al comportamiento de los objetos, hacen referencia a las clases de los objetos y se debe definir en qué punto se van a colocar estos apéndices. Para la ejecución de un programa que está hecho con la base del paradigma de programación a aspectos se debe no sólo unir clases a clases de código fuente sino buscar en todos los puntos de enlace que hay entre los aspectos y el código, a este proceso se llama entretejido o entrelazado.

Un entretejido de aspectos [Matthijs1997] y clases utiliza técnicas de transformación de programas. Sin embargo, la pregunta es cuándo o en qué momento se produce el proceso de tejido. El momento en el cual ocurre el proceso de tejido constituye una de las principales diferencias entre los LOA (lenguajes orientados a aspectos) y tomando en cuenta estas características estos se clasifican en:

- **Entretejido estático:** consiste en la modificación en tiempo de compilación del código fuente, insertando llamadas a las rutinas específicas de los aspectos. Los lugares donde estas llamadas se pueden insertar son los puntos de enlace (*joinpoints*), definidos por cada sistema. El entrelazado estático implica modificar el código fuente de una clase insertando sentencias en estos puntos de enlace. Es decir, que el código del aspecto se introduce en el de la clase, generando una “clase entretejida”.

- **Entretejido dinámico:** consiste en que el entretejido se hace después de la compilación, decir, durante la ejecución.

La mayoría de los lenguajes existentes en la actualidad pertenecen al grupo de lenguajes estáticos, incluyendo el lenguaje utilizado para realizar el presente seminario cuyo nombre es Aspectj. Otros lenguajes estáticos son: AspectSharp, Jboss.

8.3.3 Diagramas de aspectos

Ahora que se ha explicado un poco la teoría de programación orientada a aspectos se retomará la etapa de diseño utilizando la propuesta de Suzuki y Yamamoto. Entonces, la primera parte consiste en incluir los aspectos en el

diagrama de clases de diseño y posteriormente representar cada aspecto con sus respectivas clases entretejidas en un diagrama de aspectos.

8.3.3.1 Diagrama de Clases de Software incluyendo aspectos

Como se dijo anteriormente según la propuesta de Suzuki y Yamamoto se debe introducir el nuevo elemento UML denominado “aspecto” y la relación de estos elementos con las clases en que interfieren.

Para facilitar la representación de los diagramas de aspectos y además porque los aspectos que se pretenden implementar no sólo están relacionados con clases de la capa de negocios sino también a clases de la capa de acceso a datos y a la capa de presentación, se representarán los aspectos en forma unitaria mostrando cada una de las clases que se relacionan en las diferentes capas.

8.3.3.1.1 Aspecto Transacción

El manejo de transacciones es un problema presente en todo desarrollo de software. Para la resolución de este problema en el presente proyecto se ha decidido por el uso de aspectos. La solución está basada en [Laddad2003] y básicamente consiste en utilización del patrón “participant” que consiste en la creación de un aspecto abstracto base que es reutilizado por cada subaspecto en que se requiere manejar transacciones. En el caso del

problema actual los subaspectos considerados se representan en la figura 53.

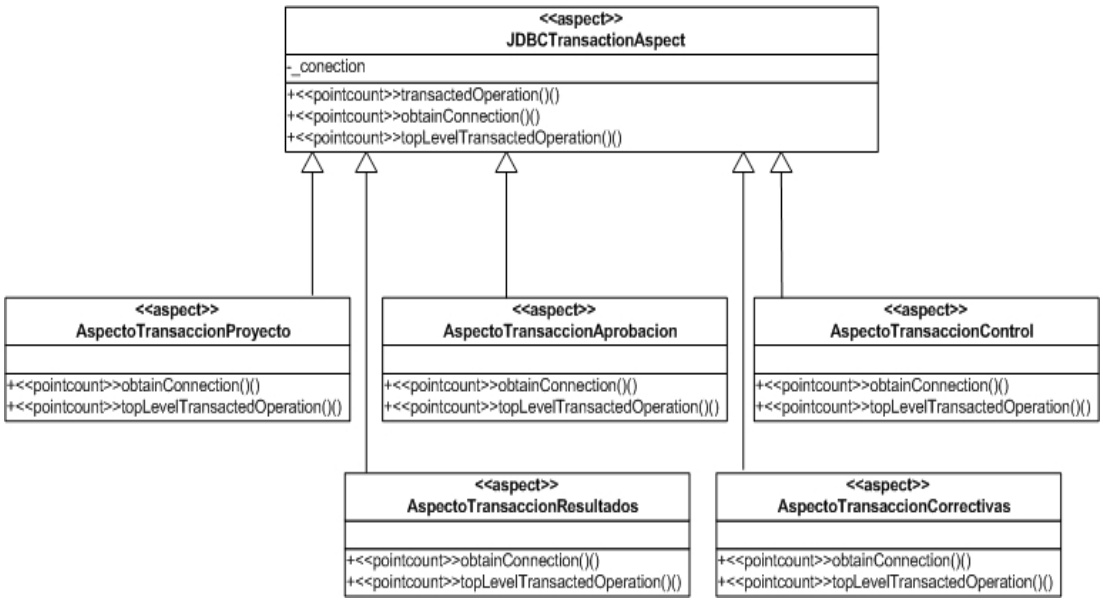


Figura 53: Aspectos para manejo de transacciones

Cada subaspecto representado en la figura está relacionado tanto con clases del negocio como clases de acceso a datos. En los siguientes puntos se mostrarás estas relaciones.

- **AspectoTransaccionProyecto**

  Este aspecto permite manejar la transacción al ingresar un nuevo proyecto y sus etapas. Los puntos de corte del aspecto AspectoTransaccionProyecto están unidos a métodos de la clase proyecto de la capa de negocios, a la clase DatabaseHelper donde se

captura la conexión y a las clases dao_proyecto y dao_etapa_proyecto, de la capa de acceso a datos, donde están las consultas de inserción de tabla respectiva en la base de datos. Este proceso está representado en la figura 54.

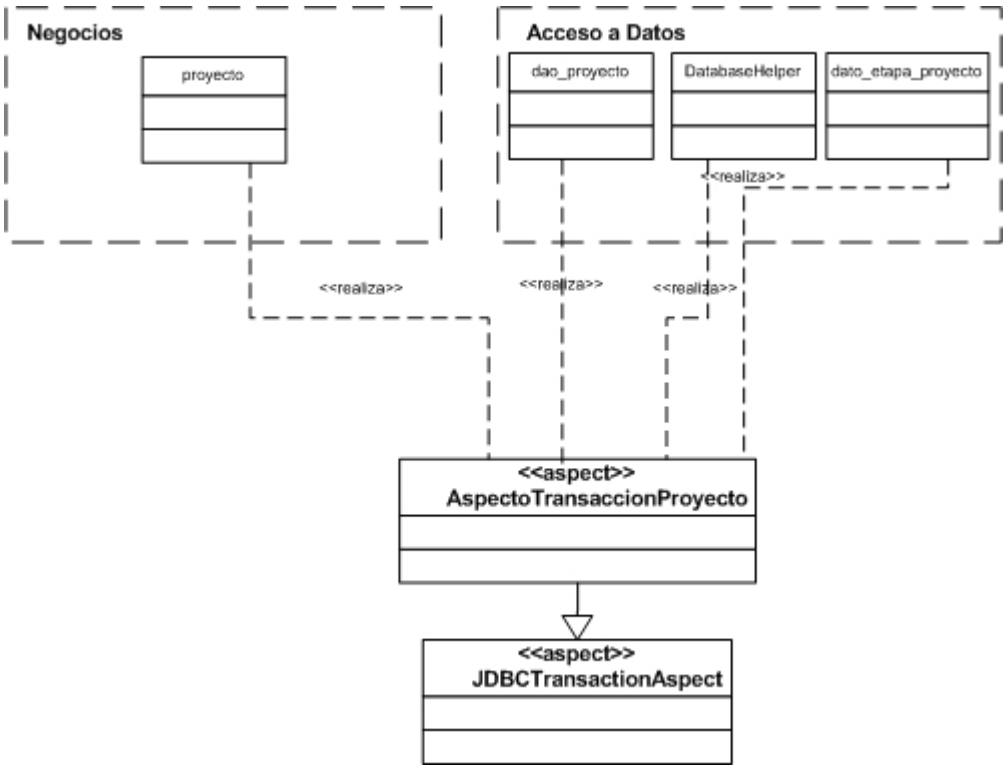


Figura 54: Diagrama de Clases con el AspectoTransaccionProyecto

- **AspectoTransaccionAprobacion**

Este aspecto permite manejar la transacción al guardar la aprobación de un determinado usuario sobre un documento. Los puntos de corte del aspecto AspectoTransaccionAprobacion están unidos a métodos de la

clase aprobacion de la capa de negocios, a la clase DatabaseHelper donde captura la conexión y a las clases dao_aprobacion y dao_documento, de la capa de acceso a datos, donde están las consultas sql de inserción y actualización a las tablas respectivas en la base de datos. Este proceso está representado en la figura 55.

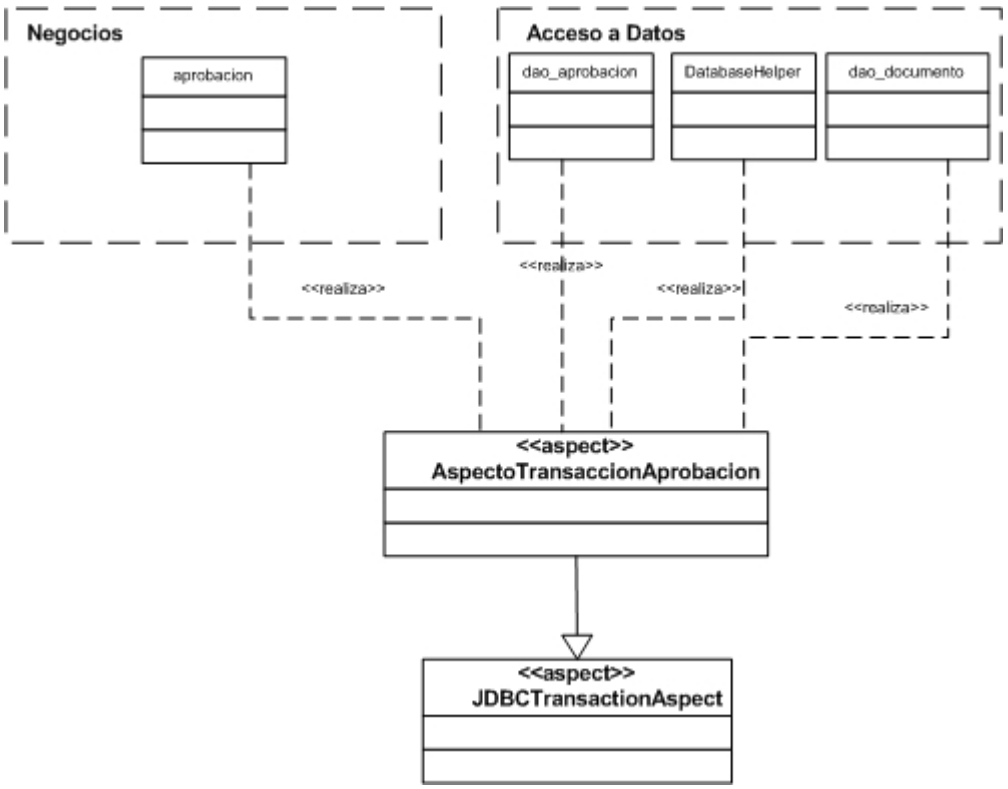


Figura 55: Diagrama de Clases con el AspectoTransaccionAprobacion

- **AspectoTransaccionControl**

Este aspecto permite manejar la transacción al guardar un control y su detalle. Los puntos de corte del aspecto AspectoTransaccionControl

están unidos a métodos de la clase control de la capa de negocios, a la clase DatabaseHelper donde se captura la conexión y a las clases dao_control y dao_detalle_control, de la capa de acceso a datos, donde están las consultas sql de inserción a la tabla respectiva de la base de datos. Este proceso está representado en la figura 56.

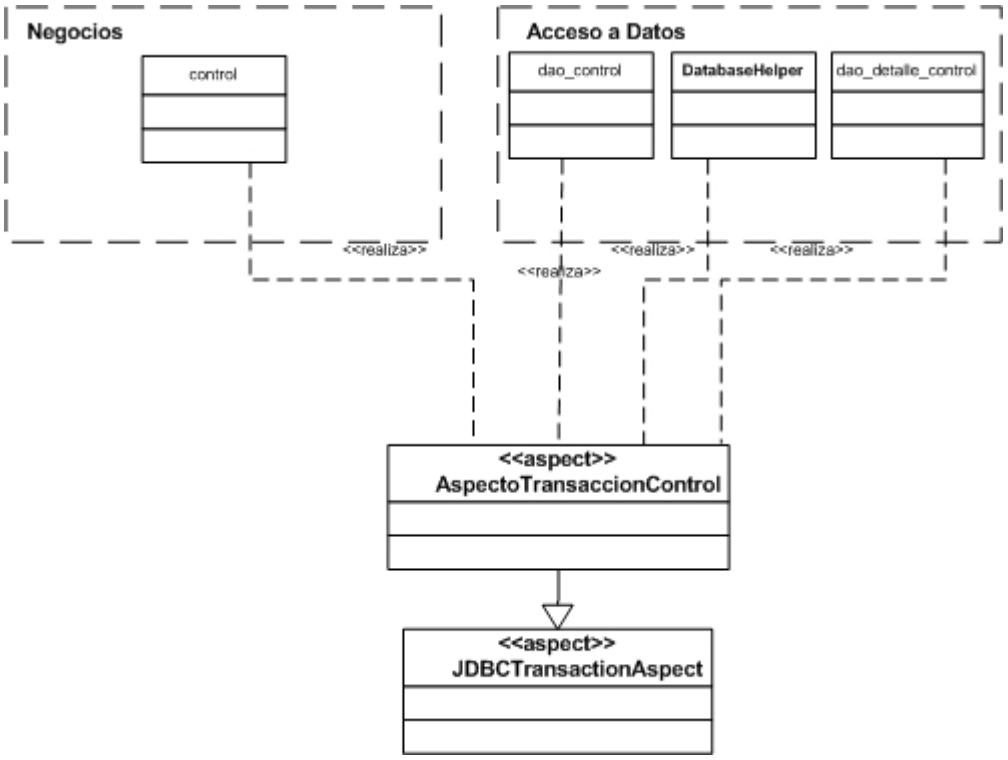


Figura 56: Diagrama de Clases con el AspectoTransaccionControl

- **AspectoTransaccionResultados**

Este aspecto permite manejar la transacción al guardar los resultados de una planificación existente. Los puntos de corte del aspecto

AspectoTransaccionResultados están unidos a métodos de la clase planificacion de la capa de negocios, a la clase DatabaseHelper donde se captura la conexión y a las clases dao_resultado_control y dao_planificacion, de la capa de acceso a datos, donde están las consultas sql de inserción y actualización a las tablas respectivas. Este proceso está representado en la figura 57.

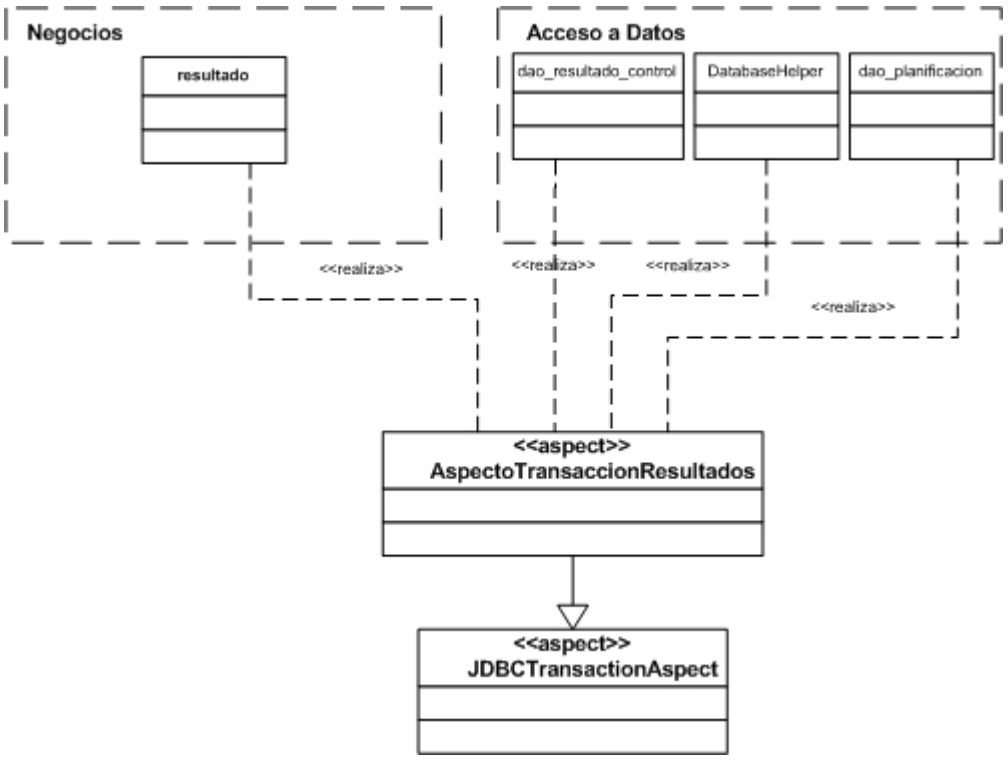


Figura 57: Diagrama de Clases con el AspectoTransaccionResultados

- **AspectoTransaccionCorrectiva**

Este aspecto permite manejar la transacción al guardar acciones correctivas a una planificación existente. Los puntos de corte del aspecto AspectoTransaccionCorrectiva están unidos a métodos de la clase correctiva de la capa de negocios, a la clase DatabaseHelper donde se captura la conexión y a las clases dao_correctiva y dao_planificacion, de la capa de acceso a datos, donde están las consultas sql de inserción y actualización a las tablas respectivas. Este proceso está representado en la figura 58.

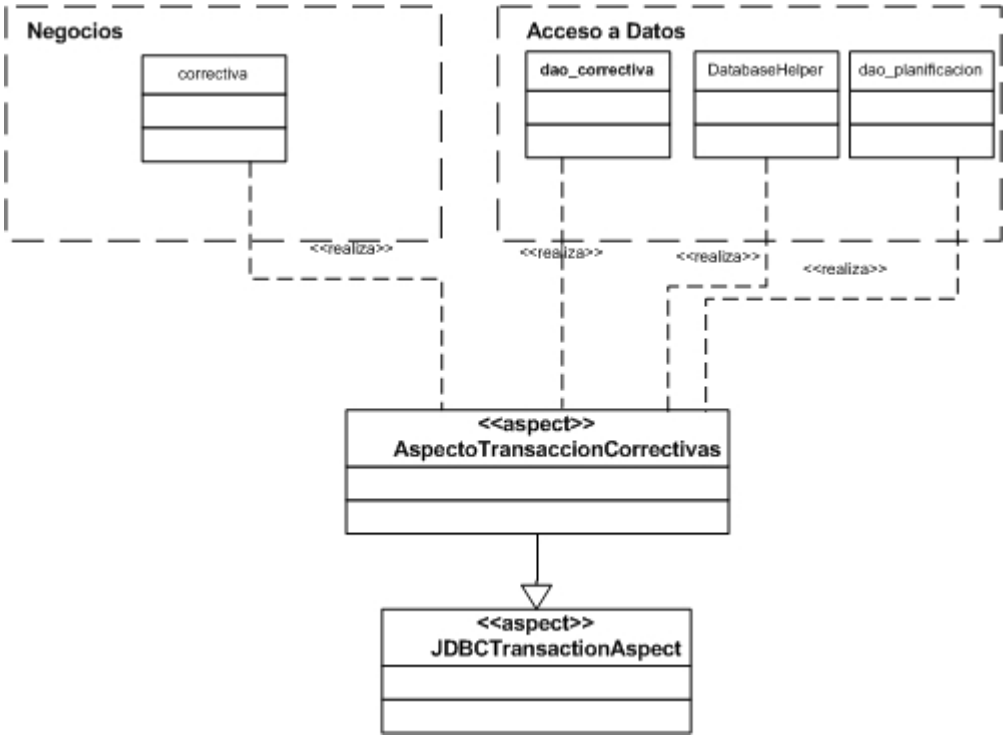


Figura 58: Diagrama de Clases con el AspectoTransaccionCorrectivas

8.3.3.1.2 Aspecto Notificación

La notificación a los usuarios en distintas instancias de la aplicación es parte importante de la solución del problema global. Este es un asunto considerado como transversal, por lo tanto se trabajará como un aspecto, que básicamente unirá las clases donde se requiere hacer una notificación con las clases que permite hacer dicha función. Como se ve en la figura 59 los puntos de corte de este aspecto están relacionados con clases que sólo pertenecen a la capa de negocios.

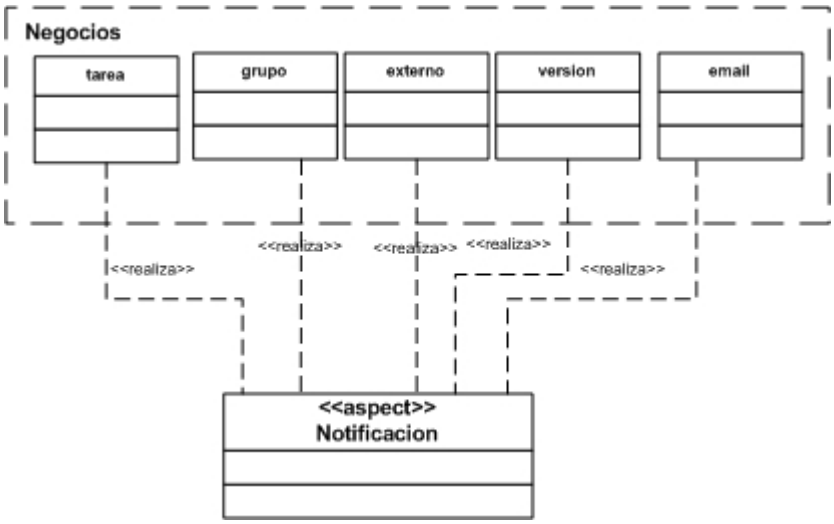


Figura 59: Diagrama de Clases con el Aspecto Notificación

8.3.3.1.3 Aspecto Autentificación

El proceso de autentificación es el proceso en el cual un usuario verifica que tiene permisos para trabajar en el sistema. Es por eso que cada vez que un usuario ingresa a una página del sistema tiene que verificar si el usuario está o no autentificado.

Para este proceso se usará un aspecto que está unido a clases que pertenecen a la capa de presentación, es decir, a las clases Managed Beans que controlan de las páginas. Cada vez que una página se inicia, el aspecto Autentificación verificará si el usuario está autentificado previamente, si no lo ha hecho, se encargará de redireccionar a una página de logeo. Este aspecto está unido a todas las clases controladoras de páginas de la capa de presentación, pero por su gran número en la figura 60 se puede ver la relación sólo con alguna de ellas a modo de ejemplo.

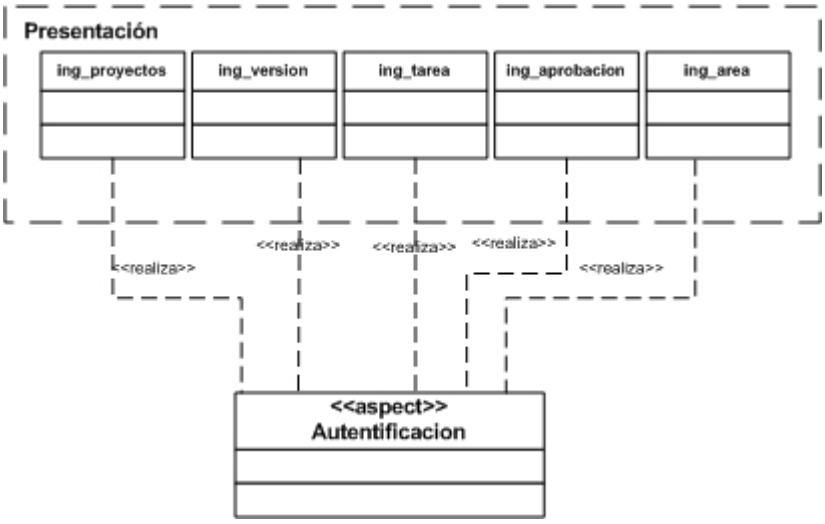


Figura 60: Diagrama de Clases con el Aspecto Autentificación

8.3.3.1.4 Aspecto Autorización

Este aspecto es utilizado para ver si un usuario previamente autentificado tiene los permisos suficientes para ver o no contenidos del sistema. Estos permisos estarán dados previamente por el administrador en una página habilitada para ello, donde a cada perfil podrá tener permisos para que vea o no diferentes interfaces. Al igual que el aspecto de Autentificación, el de Autorización está unido a todas las clases Managed Beans que controlan de las páginas de la capa de presentación, pero a modo de ejemplo se muestran algunas relaciones en la figura 61.

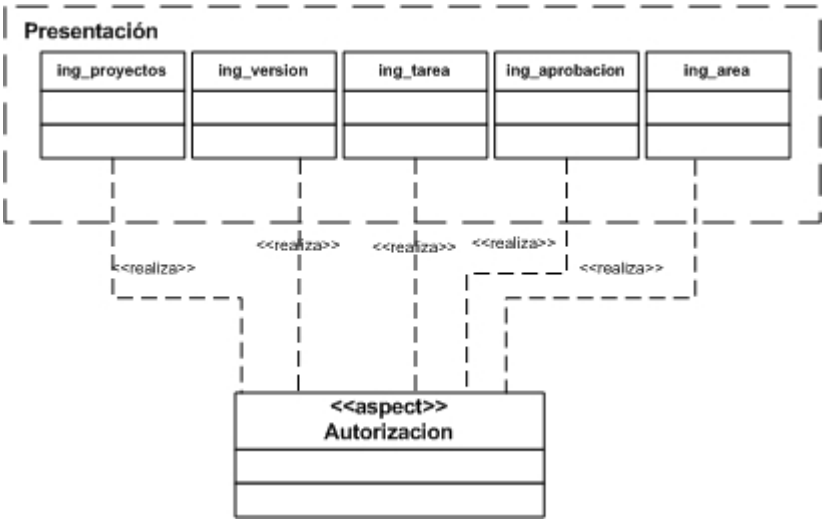


Figura 61: Diagrama de Clases con el Aspecto Autorización

8.3.3.2 Diagramas de aspectos representando la clase entretejida

La forma de estos diagramas depende de cómo se realice el proceso de entretejido del LOA (lenguaje orientado a aspectos) elegido. Como se dijo anteriormente el lenguaje Aspectj genera la clase entretejida juntando el código de la clase base y de los aspectos enlazados a ella. Por lo tanto estos diagramas utilizando la notación de paquetes de UML tendrán la siguiente estructura.

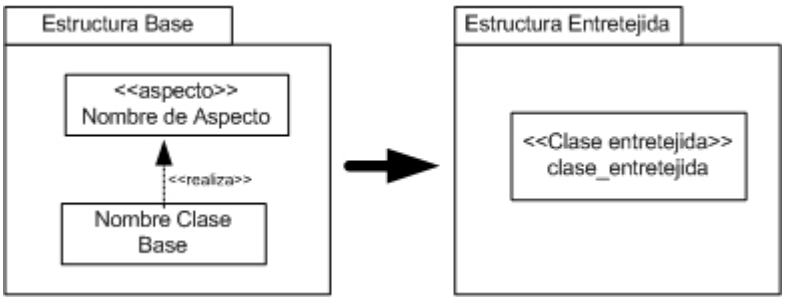


Figura 62: Diagrama básico de un aspecto utilizando un entretejido que genera una nueva clase

Según la figura 62 en la estructura base se debe poner el aspecto junto a la clase a la que está enlazado y en el paquete de estructura entretejida se debe colocar la clase generada después del proceso de entretejido. Si el lenguaje no generará una nueva clase entretejida lógicamente este diagrama sólo tendría uno de los paquetes.

De acuerdo a este diagrama básico se han generado los diagramas de aspectos del sistema. En el caso de los aspectos de autentificación y autorización sólo se mostrará el entrelazado con una clase a modo de ejemplo, sin embargo se da por entendido que estos aspectos interfieren a casi todas las clases controladoras de páginas de la capa de presentación.

8.3.3.2.1 Aspecto Notificación

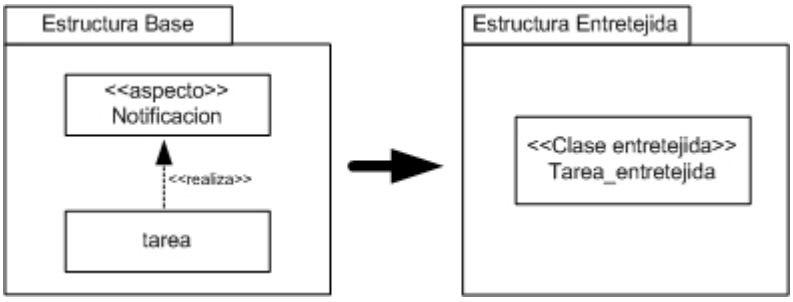


Figura 63: Diagrama de Aspectos Notificación a Clase Base Tarea

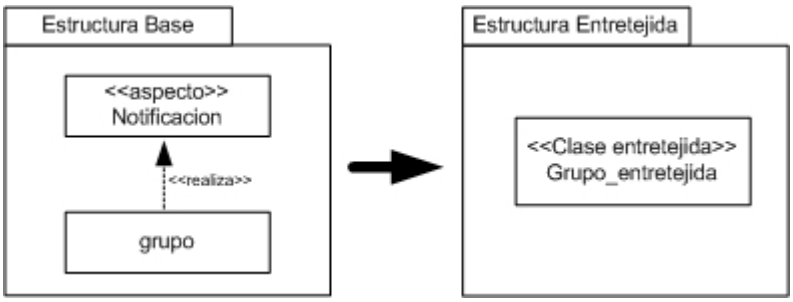


Figura 64: Diagrama de Aspectos Notificación a Clase Base Grupo

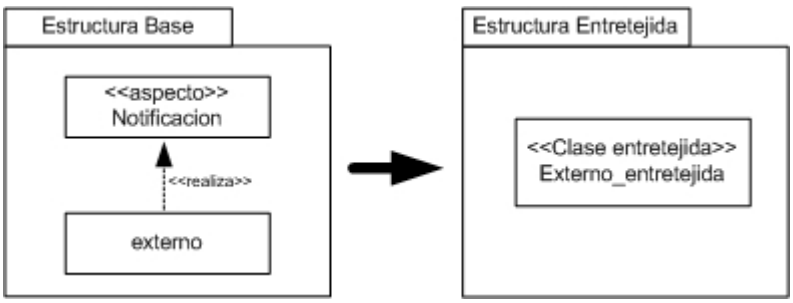


Figura 65: Diagrama de Aspectos Notificación a Clase Base Externo

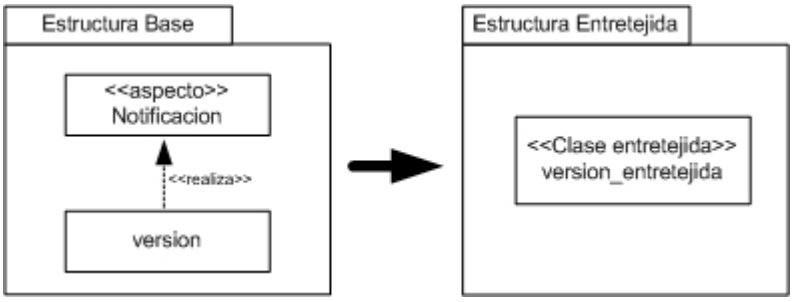


Figura 66: Diagrama de Aspectos Notificación a Clase Base Versión

8.3.3.2.2 Aspecto Transacción

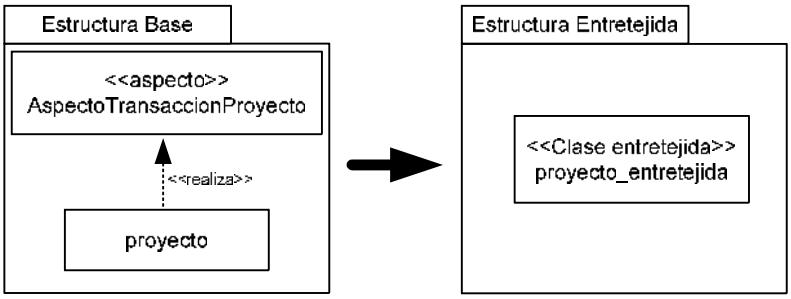


Figura 67: Diagrama de Aspectos AspectoTransaccionProyecto a Clase Base Proyecto

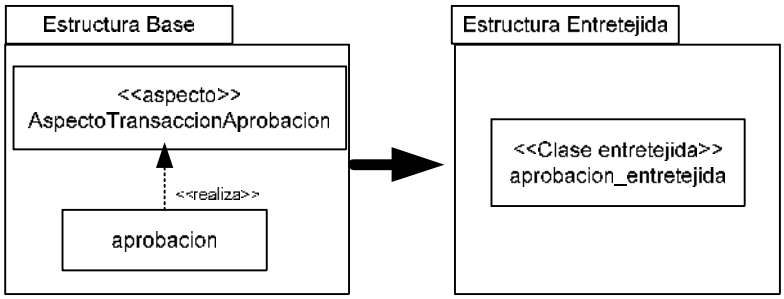


Figura 68: Diagrama de Aspectos AspectoTransaccionAprobacion a Clase Base Aprobacion

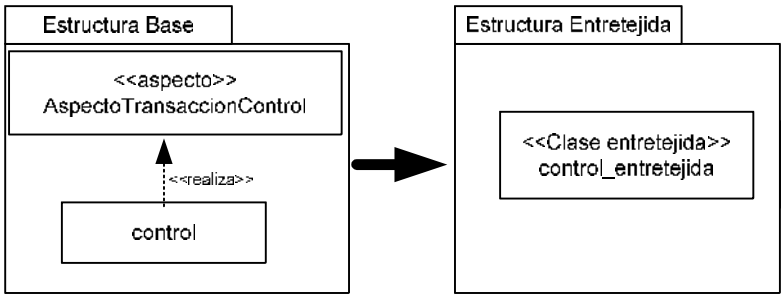


Figura 69: Diagrama de Aspectos AspectoTransaccionControl a Clase Base Control

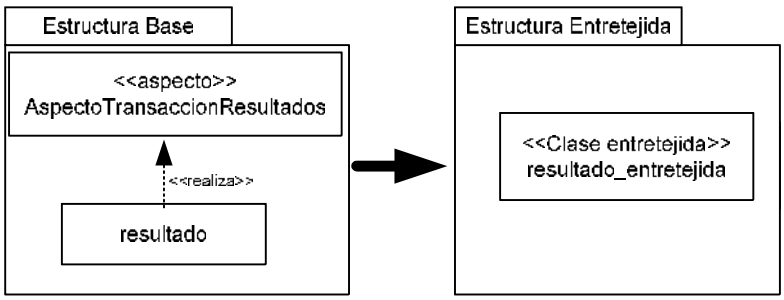


Figura 70: Diagrama de Aspectos AspectoTransaccionResultados a Clase Base Resultado

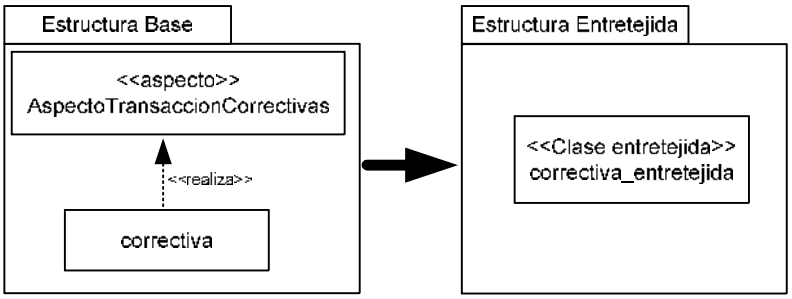


Figura 71: Diagrama de Aspectos AspectoTransaccionCorrectivas a Clase Base Correctiva

8.3.3.2.3 Aspecto Autentificación

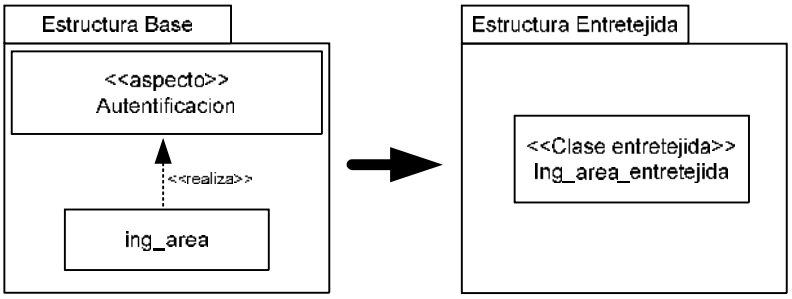


Figura 72: Diagrama de Aspectos Autentificación a Clase Base ing_area

8.3.3.2.4 Aspecto Autorización

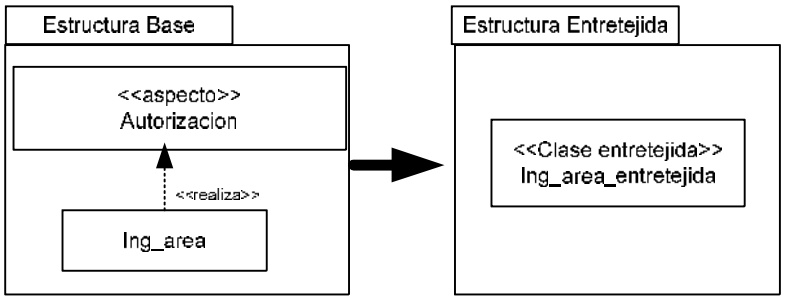


Figura 73: Diagrama de Aspectos Autorización a Clase Base ing_area

8.4 Diseño de la base de datos

Para completar el diseño de la base de datos se ha seguido la metodología presentada en el libro de Connolly y Begg [Connolly2005] que consta de tres importantes fases:

- Diseño conceptual: El proceso de construcción de un modelo de los datos utilizados independiente de las consideraciones físicas.
- Diseño lógico: El proceso de construir un modelo de datos basándose en un modelo específico (relacional).
- Diseño físico: El proceso de generar una descripción de la implementación de la base de datos en un medio secundario, es decir está orientado a un SGBD (sistema gestor de base de datos) especifico.

Cada fase consta de un conjunto de pasos que permiten generar el modelo final de la base de datos. A continuación se muestran las fases de la metodología, con el detalle de los pasos que se estimaron convenientes puesto que según dice la misma metodología se pueden omitir pasos dependiendo del problema.

8.4.1 Diseño Conceptual

8.4.1.1 Identificar los tipos de entidades

En este primer paso del diseño conceptual se señalan las entidades que han sido identificadas inicialmente en el modelo inicial de dominio (figura 7) o posteriormente en el diagrama de clases del sistema (figura 50). Sin embargo existen entidades que no se identificaron en las fases de análisis anteriores ya que están enfocadas más bien a funciones de administración del sistema. Además es necesario indicar que algunos nombres de las entidades del modelo conceptual que son distintos a las de los modelos de las fases anteriores, esto se hizo para seguir la nomenclatura usada a nivel interno en las base de datos.

En la tabla 15 se muestra la lista competa de entidades identificadas con su correspondiente sinónimo (alias) del modelo inicial de dominio o modelo de clases.

Entidad	Alias
aprobacion	aprobacion
biblioteca	biblioteca
control	formulario de control
correctiva	Acciones correctivas

detalle_control	detalle control
documento	Dcto Interno
documento_apoyo	Documentos de Apoyo
documento_version	version
Email	
Externo	Dcto Externo
mae_area	Area
mae_cargo	Cargo
mae_centro	centro
mae_departamento	Departamento
mae_documento_familia	Familia
mae_documento_tipo	Tipo
mae_etapa	Etapa
mae_perfil	Perfil
mae_proyecto_tipo	Tipo Proyecto
mae_tipo_variable	Tipo Variable
mae_usuario	Usuario
mae_variable	Variable de Control
modulo	Módulo del sistema
pagina	Página del sistema
planificacion	Planificación
proyecto	Proyecto

resultado_control	Resultado Planificacion
Tarea	Tarea

Tabla 15: Entidades modelo conceptual

8.4.1.2 Identificar los tipos de relación

En la tabla 16 se muestran las relaciones identificadas con sus respectivos nombres y las entidades que unen.

Nombre	**Entidad 1**	**Entidad 2**	**Descripción**
esTipoVariable	mae_tipo_variable	mae_variable	Una variable es de un Tipo Variable
tienePagina	modulo	pagina	Un módulo tiene páginas
tienePaginaPerfil	mae_perfil	pagina	Un perfil tiene muchas páginas y a su vez las páginas tienen un perfil.
TienePerfil	mae_perfil	mae_usuario	Un usuario tiene un perfil.
perteneceArea	mae_departamento	mae_area	Un Departamento pertenece a una

			determinada área.
perteneceCentro	planificacion	mae_centro	Una planificación pertenece a un centro.
esTipoProyecto	mae_proyecto_tipo	proyecto	Un proyecto es de un Tipo de Proyecto.
tieneVersion	documento	documento_version.	Un documento tiene documento versión (muchas versiones).
tieneDocApoyo	documento_apoyo	proyecto	Un proyecto tiene documentos de apoyo.
tieneDoc	documento	proyecto	Un proyecto tiene documentos.
tieneTarea	proyecto	tarea	Un Proyecto tiene tareas.
esTipoDoc	mae_documento_tipo	biblioteca	Una biblioteca es de un tipo de documento.
esFamiliaDoc	mae_documento_familia	biblioteca	Una biblioteca es de una familia de documento.
perteneceDpto	mae_departamento	proyecto	Un proyecto pertenece a un departamento
esDeControl	control	planificacion	Una planificación es de un determinado

			control.
esdePlanificacion	planificacion	resultado_control	Un resultado de control es de una planificación
tieneDetalleControl	control	detalle_control	Un control tiene un detalle de control
tieneVariable	mae_variable	detalle_control	Un detalle de control tiene variables
esDetalleControl	detalle_control	resultado_control	Un resultado de un control es de un detalle de control
tieneUsuarioResp	control	mae_usuario	Un control tiene un usuario responsable.
hechaPorUsuario	tarea	mae_usuario	Una tarea está hecha por un usuario.
tieneUsuario	mae_usuario	planificacion	Una planificación tiene un usuario.
esSubidaUsuario	documento_version	mae_usuario	Un documento_version es subido por un usuario
esBloqueadoUsuario	mae_usuario	documento_version	Un documento_version es bloqueado por un usuario.

tieneCorrectiva	planificacion	correctiva	Una planificación puede tener o no acciones correctivas.
esDeUsuario	mae_usuario	aprobacion	Una aprobación es de un usuario.
tieneAprobacion	documento	aprobacion	Un documento tiene aprobacion
tieneEtapa	proyecto	mae_etapa	Un proyecto tiene etapas.

Tabla 16: Relaciones modelo conceptual

8.4.1.3 Identificar los atributos con los tipos de entidad y de relación

En este paso se identifican los atributos entidad por entidad, además de los dominios y su respectiva descripción.

Nombre Atributo	Tipo Dato	Descripción
aprobacion_id	Integer	Id autogenerado
fecha_aprobacion	Date	Fecha en que se aprueba el documento

Tabla 17: Entidad aprobación

Nombre Atributo	Tipo Dato	Descripción
biblioteca_id	Integer	Id Autogenerado
biblioteca_nombre	VarChar(25)	Nombre de un documento existen en biblioteca
biblioteca_formato	VarChar(50)	Formato o extensión del documento

Tabla 18: Entidad biblioteca

Nombre Atributo	Tipo Dato	Descripción
control_id	Integer	Id autogenerado
control_nombre	VarChar(25)	Nombre para identificar al formulario de control.
control_objetivo	Text	Objetivo del formulario de control
fecha_generacion	Date	Fecha en que se genera el registro
fecha_modi_ultima	Date	Ultima fecha en que el registro es modificado

Tabla 19: Entidad control

Nombre Atributo	Tipo Dato	Descripción
correctiva_id	Integer	Id autogenerado
correctiva_descripcion	Text	La descripción de la acción correctiva
fecha_ingreso	Date	Fecha en que fue ingresado el registro.
fecha_ejecucion_est	Date	Fecha estimada de ejecución de la acción correctiva.
fecha_ejecucion_real	Date	Fecha en que realmente fue ejecutada la acción correctiva.
estado_correctiva	Short integer	Estado de la acción correctiva.
ejecutor_nombre	VarChar(25)	Nombre de la persona o departamento que debe ejecutar la acción correctiva
ejecutor_cargo	VarChar(25)	Cargo de la persona que debe ejecutar la acción correctiva.

Tabla 20: Entidad correctiva

Nombre Atributo	Tipo Dato	Descripción
detalle_control_id	Integer	Id autogenerado
estado_detalle_control	Short integer	Estado del detalle del control, es decir, si está activo o no.

Tabla 21: Entidad detalle_control

Nombre Atributo	Tipo Dato	Descripción
documento_id	Integer	Id autogenerado
documento_estado	Short integer	Estado del documento, es decir si está aprobado o no.
version_final	Float	Cual es la versión del documento que fue aprobada.
fecha_aprobacion_doc	Date	Fecha de aprobación del documento

Tabla 22: Entidad documento

Nombre Atributo	Tipo Dato	Descripción
apoyo_externo_id	Integer	Id autogenerado
apoyo_externo_nombre	VarChar(40)	Nombre del documento de apoyo.
apoyo_externo_formato	VarChar(20)	Formato o extensión del documento de apoyo.
apoyo_externo_archivo	longblob	Representa los bytes del archivo.
fecha_asignacion	Date	Fecha en que fue asignado el documento de apoyo.

Tabla 23: Entidad apoyo_externo

Nombre Atributo	Tipo Dato	Descripción
documento_version_id	Integer	Id autogenerado
version_num	Decimal	Número de versión.
version_archivo	longblob	Representa los bytes del archivo.
fecha_subida	Date	Fecha en que fue subida la versión.

Tabla 24: Entidad documento_version

Nombre Atributo	Tipo Dato	Descripción
email_id	Integer	Id autogenerado
email_titulo	VarChar(40)	Asunto del email enviado.
email_contenido	Text	Mensaje del email enviado
email_fecha_envio	Date	Fecha del email enviado
email_origen	VarChar(50)	Dirección de origen del email enviado.
email_destino	VarChar(50)	Dirección destino del email enviado.

Tabla 25: Entidad email

Nombre Atributo	Tipo Dato	Descripción
externo_id	Integer	Id autogenerado
externo_autor	VarChar(40)	Autor del documento externo.
externo_origen	VarChar(20)	Origen del documento externo, puede ser un organismo o empresa.
externo_archivo	longblob	Representa los bytes del archivo.
externo_fecha_publicacion	Date	Fecha de publicación del documento.
externo_fecha_subida	Date	Fecha de ingreso del documento al sistema.

Tabla 26: Entidad externo

Nombre Atributo	Tipo Dato	Descripción
area_id	Integer	Id autogenerado
area_nombre	VarChar(25)	Nombre del área.
area_descripcion	Text	Descripción del área.
estado_area	Short integer	Estado del área, es decir, si está activa o inactiva.

Tabla 27: Entidad mae_area

Nombre Atributo	Tipo Dato	Descripción
cargo_id	Integer	Id autogenerado
cargo_nombre	VarChar(25)	Nombre del cargo.
cargo_descripcion	Text	Descripción del cargo.
cargo_estado	Short integer	Estado del cargo, es decir, si está activo o inactivo.

Tabla 28: Entidad mae_cargo

Nombre Atributo	Tipo Dato	Descripción
centro_id	Integer	Id autogenerado
centro_nombre	VarChar(25)	Nombre del centro de cultivo.
centro_email	VarChar(20)	Descripción del centro de cultivo.
estado_centro	Short integer	Estado del centro, es decir, si está activo o inactivo.

Tabla 29: Entidad mae_centro

Nombre Atributo	Tipo Dato	Descripción
departamento_id	Integer	Id autogenerado
departamento_nombre	VarChar(25)	Nombre del departamento.
estado_departamento	Short integer	Estado del departamento, es decir, si está activo o inactivo.

Tabla 30: Entidad mae_departamento

Nombre Atributo	Tipo Dato	Descripción
familia_id	Integer	Id autogenerado
familia_nombre	VarChar(25)	Nombre de la familia de documento.
familia_descripcion	Text	Descripción de la familia de documento.
estado_familia	Short integer	Estado de la familia de documento, es decir, si está activo o inactivo.

Tabla 31: Entidad mae_documento_ familia

Nombre Atributo	Tipo Dato	Descripción
tipo_id	Integer	Id autogenerado
tipo_nombre	VarChar(25)	Nombre de la tipo de documento.
tipo_descripcion	Text	Descripción del tipo de documento.
estado_tipo	Short integer	Estado del tipo de documento, es decir, si está activo o inactivo.

Tabla 32: Entidad mae_documento_tipo

Nombre Atributo	Tipo Dato	Descripción
etapa_id	Integer	Id autogenerado
etapa_nombre	VarChar(25)	Nombre de la etapa.
etapa_descripcion	Text	Descripción de la etapa.
estado_etapa	Short integer	Estado de la etapa, es decir, si está activa o inactiva.

Tabla 33: Entidad mae_etapa

Nombre Atributo	Tipo Dato	Descripción
perfil_id	Integer	Id autogenerado
perfil_nombre	VarChar(25)	Nombre del perfil.
perfilestado	Short integer	Estado del perfil, es decir, si está activo o inactivo.

Tabla 34: Entidad mae_perfil

Nombre Atributo	Tipo Dato	Descripción
proyecto_tipo_id	Integer	Id autogenerado
proyecto_tipo_nombre	VarChar(25)	Nombre del tipo de proyecto.
proyecto_tipo_descripcion	Text	Descripción del tipo de proyecto.
estado_tipo_proyecto	Short integer	Estado del tipo de proyecto, es decir, si está activo o inactivo

Tabla 35: Entidad mae_proyecto_tipo

Nombre Atributo	Tipo Dato	Descripción
tipo_variable_id	Integer	Id autogenerado
tipo_variable_nombre	VarChar(25)	Nombre del tipo variable.
estado_tipo_variable	Short integer	Estado del tipo de variable, es decir, si está activo o inactivo

Tabla 36: Entidad mae_tipo_variable

Nombre Atributo	Tipo Dato	Descripción
usuario_id	Integer	Id autogenerado
usuario_rut	Integer	Rut del usuario ingresado.
usuario_nombre	VarChar(50)	Nombres del usuario ingresado.
usuario_apellido	VarChar(50)	Apellidos del usuario ingresado.
usuario_email	VarChar(50)	Email del usuario.
user	VarChar(12)	Nombre de usuario con que accederá al sistema.
pwd	VarChar(40)	Contraseña encriptada.
usuario_estado	Short integer	Estado del usuario, es decir, si esta activo o inactivo.
digito_rut	Short integer	Digito verificador del Rut.
ultimo_ing	Date	Fecha del último ingreso del usuario.

Tabla 37: Entidad mae_usuario

Nombre Atributo	Tipo Dato	Descripción
variable_id	Integer	Id autogenerado
variable_nombre	VarChar(25)	Nombre de la variable.
unidadmedida	VarChar(10)	Unidad de medida de la variable.
variable_descripcion	Text	Descripción de la variable.
estado_variable	Short integer	Estado de las variables, es decir, si está activo o inactivo.

Tabla 38: Entidad mae_variable

Nombre Atributo	Tipo Dato	Descripción
modulo_id	Integer	Id autogenerado
modulo_nombre	VarChar(10)	Nombre del módulo del sistema.

Tabla 39: Entidad mae_modulo

Nombre Atributo	Tipo Dato	Descripción
pagina_id	Integer	Id autogenerado
pagina_nombre	VarChar(12)	Nombre de la página.

Tabla 40: Entidad mae_pagina

Nombre Atributo	Tipo Dato	Descripción
planificacion_id	Integer	Id autogenerado
fecha_est	Date	Fecha estimada en que se realizará la planificación de control.
estado_planificacion	Short integer	Estado de la planificación, es decir, si está ejecutada o no.
fecha_real	Date	Fecha en que se realizó la planificación.
conclusion	Text	Conclusión final de la planificación de control.

Tabla 41: Entidad planificacion

Nombre Atributo	Tipo Dato	Descripción
proyecto_id	Integer	Id autogenerado
proyecto_nombre	VarChar(25)	Nombre del proyecto.
fecha_creacion	Date	Fecha creación del proyecto.
fecha_fin	Date	Fecha Fin del proyecto.
objetivo	Text	Objetivo del proyecto.
estado_proyecto	Short integer	Estado del proyecto, es decir, si está terminado o en proceso.

Tabla 42: Entidad proyecto

Nombre Atributo	Tipo Dato	Descripción
resultado_id	Integer	Id autogenerado
valor	VarChar(25)	Resultado de la variable medida.

Tabla 43: Entidad resultado_control

Nombre Atributo	Tipo Dato	Descripción
tarea_id	Integer	Id autogenerado
tarea_nombre	VarChar(30)	Nombre de la tarea.
tarea_descripcion	Text	Descripción de la tarea.
fecha_fin_est	Date	Fecha fin estimada de la tarea.
fecha_fin_real	Date	Fecha fin real de la tarea.
estado_tarea	Short integer	Estado de la tarea, es decir, si está terminada o en proceso.

Tabla 44: Entidad tarea

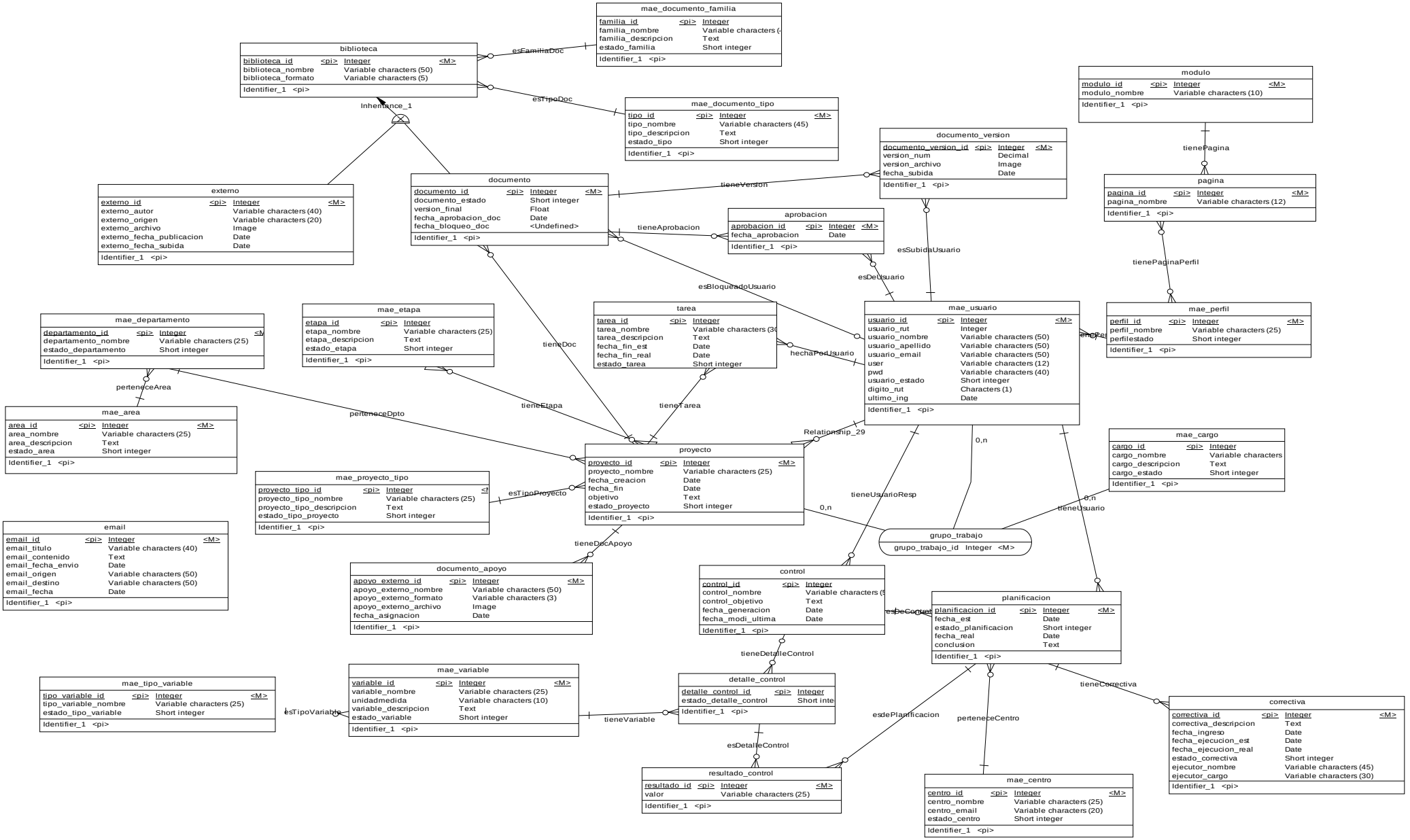


Figura 74: Modelo conceptual de la base de datos

8.4.2 Diseño lógico de la base de datos para base de datos relacionales

8.4.2.1 Determinar las relaciones para el modelo lógico de los datos

8.4.2.1.1 Relaciones muchos a muchos

En el modelo se identifican las dos relaciones muchos a muchos, que se detallan a continuación.

Relación “tieneEtapa”

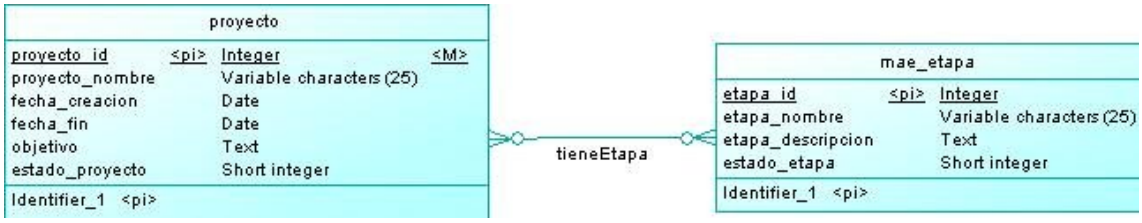


Figura 75: Relación “tieneEtapa”

En este caso un proyecto tiene muchas etapas y la vez una etapa puede estar en muchos proyectos. Es por eso que se crea una tabla intermedia como se muestra en la figura 76.

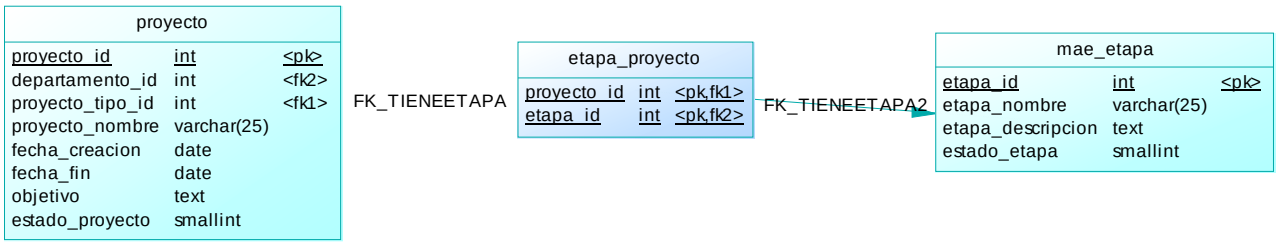


Figura 76: Tablas intermedia entre proyecto y mae_etapa

Relación TienePaginaPerfil

Figura 77: Relación "tieneEtapa"

En la relación llamada **TienePaginaPerfil** un perfil puede acceder a muchas páginas y una página puede ser accedida por muchos perfiles. Es por eso que se crea la tabla intermedia como se muestra en la figura 78.

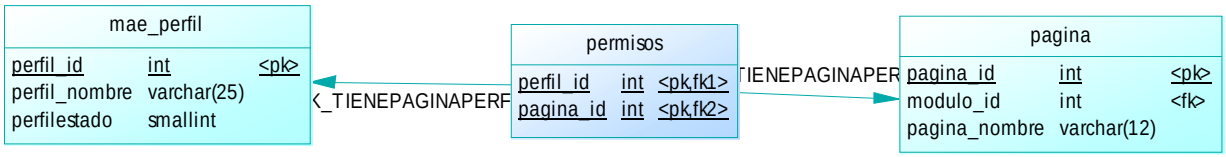


Figura 78: Tabla intermedia entre mae_peril y página

8.4.2.1.2 Tipos de relaciones superclases/subclases

Se identifica una relación de tipo superclase/subclase, donde se involucra las entidades biblioteca, externo y documento.

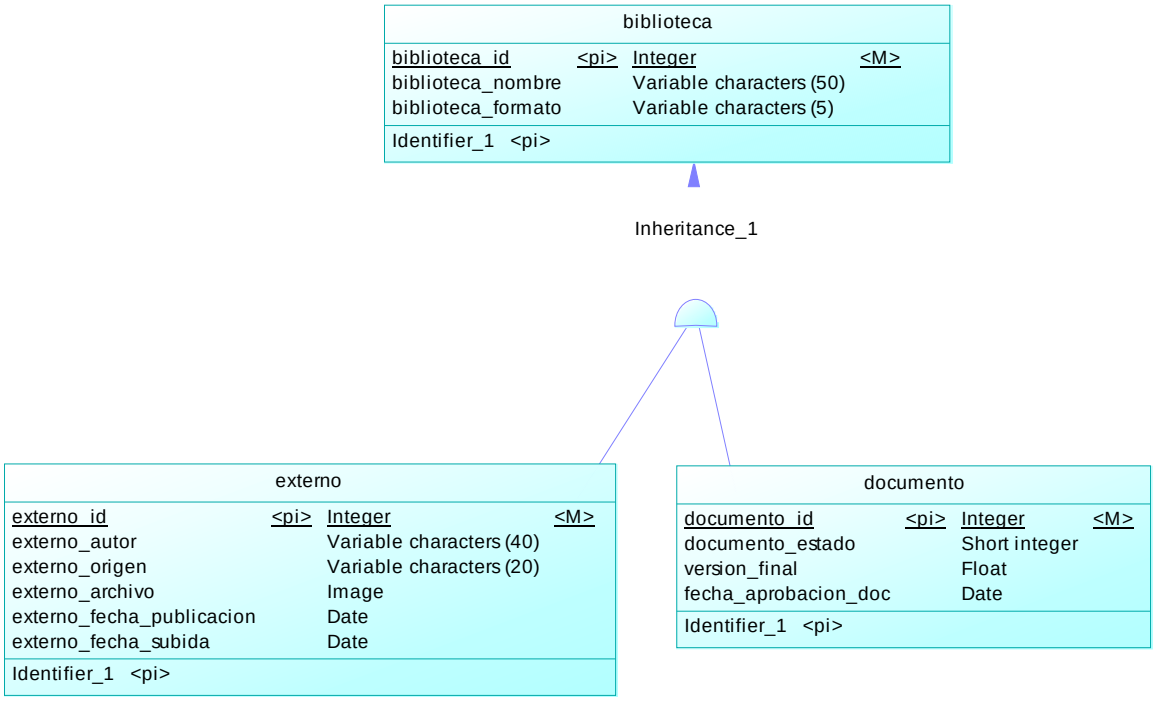


Figura 79: Relación tipo superclase/subclase

8.4.2.1.3 Tipos de Relaciones complejas

Se encuentra una relación compleja en el modelo. En esta relación se llama grupo_trabajo e involucran las entidades: mae_usuario, proyecto y mae_cargo.

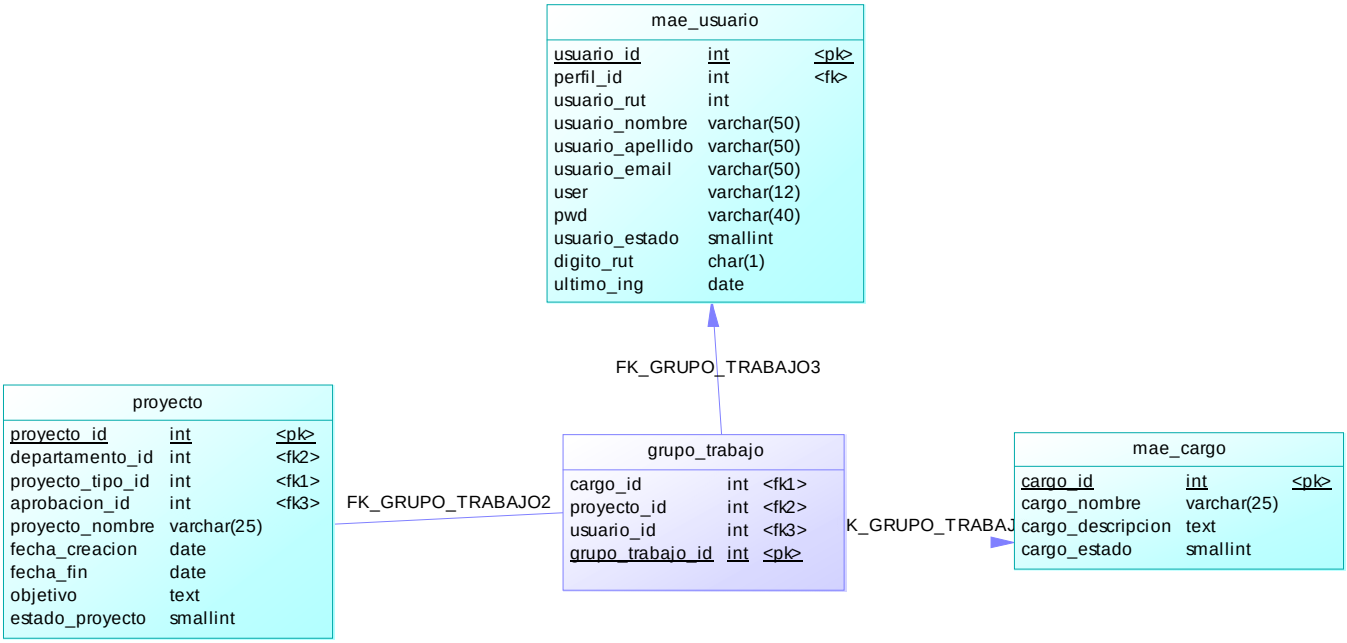


Figura 80: Relación compleja “grupo_trabajo”

8.4.3 Diseño físico de la base de datos para base de datos relacionales

En el diseño físico se traducen las relaciones contenidas en el modelo lógico de los datos a una forma que pueda implementarse en un SGBD (sistema gestor de base de datos) que en este caso es Mysql 5.0. Este proceso se lleva a cabo con la herramienta PowerDesginer.

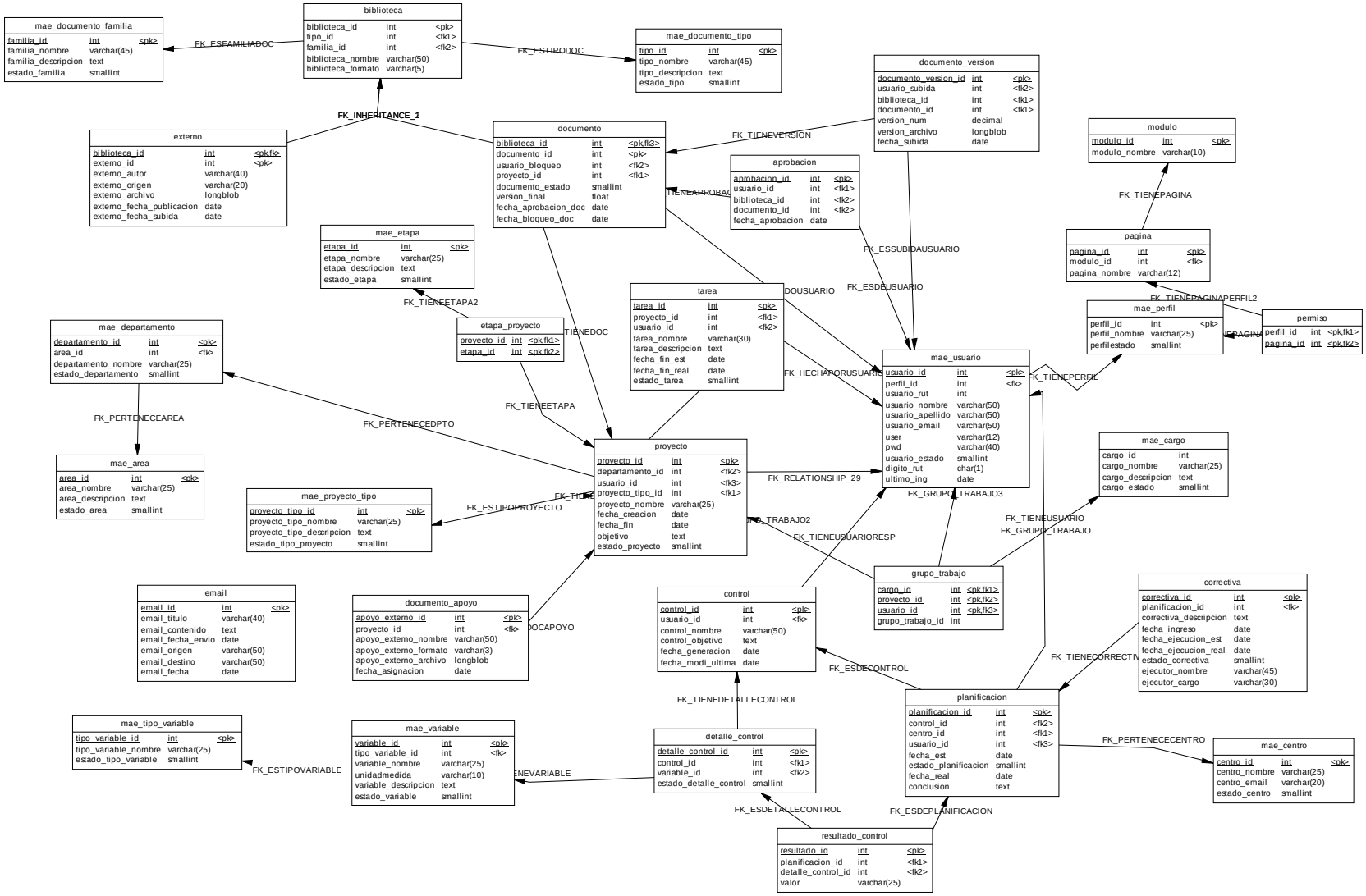


Figura 81: Modelo físico de la base de datos

8.5 Construcción de la Aplicación

Para demostrar la codificación tanto de clases y aspectos se mostrará a continuación paso a paso la ejecución de un proceso determinado. El proceso escogido es el ingreso de un usuario al sistema y el posterior ingreso de un nuevo proyecto y sus etapas. A medida que se avance en el proceso se irá explicando como se instancias las clases en las distintas capas y como intervienen los aspectos en su comportamiento.

Para ingresar al sistema un usuario que está registrado debe autentificarse, es decir, debe iniciar una sesión, ingresando su nombre de usuario y contraseña en la página de inicio (login.jsp). Como se ha dicho anteriormente al utilizar el framework JSF (Java Server FAces) todas las páginas cuentan por el lado del servidor con su respectiva clase Managed Beans que permite manejar los eventos y las propiedades utilizadas por los componentes UI de la página.

El código de la página login.jsp es el siguiente:

```
<jsp:root version="2.1" xmlns:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html" xmlns:ice="http://www.icesoft.com/icefaces/component"
xmlns:jsp="http://java.sun.com/JSP/Page">
   <jsp:directive.page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8"/>
   <f:view>
      <ice:outputHtml id="outputHtml1">
         <ice:outputHead id="outputHead1">
            <ice:outputStyle href="./resources/stylesheet.css" id="outputStyle1"/>
         </ice:outputHead>
         <ice:outputBody id="outoutBody1" style="background-color: rgb(223, 219, 219)">
            <ice:form id="form1">
               <div id="container" >
                  <ice:panelLayout id="panelLayout1" >
                     <ice:outputLabel id="outputLabel1"   value="Usuario:"/>
                     <ice:inputText binding="#{login.inputTextNombre}" id="inputTextNombre"
style="left: 336px; top: 192px; position: absolute; width: 215px"/>
                     <ice:commandButton action="#{login.buttonIngresar_action}"
id="buttonIngresar" style="left: 408px; top: 264px; position: absolute" value="Ingresar"/>
                     <ice:commandButton id="buttonCancelar" style="left: 480px; top: 264px;
position: absolute" type="reset" value="Cancelar"/>
                     <ice:graphicImage height="145" id="graphicImage1" style="left: 0px; top: 0px;
position: absolute" url="/resources/encabezado.png" width="781"/>
                     <ice:graphicImage height="27" id="graphicImage2" style="left: 0px; top: 312px;
position: absolute" url="/resources/pie.jpg" width="781"/>
                     <ice:outputLabel id="outputLabel2"                    style=" left: 240px; top:
216px; position: absolute; "  />
                     <ice:inputSecret binding="#{login.inputSecret1}" id="inputSecret1"     style="left:
336px; top: 216px; position: absolute; width: 215px"/>
                  </ice:panelLayout>
               </div>
            </ice:form>
         </ice:outputBody>
      </ice:outputHtml>
   </f:view>
</jsp:root>
```

Como se puede ver en el código de la página el botón “Ingresar” está mapeado con el método buttonIngresar_action() de la clases login.java. A continuación se muestran las partes más importantes de esta clase.

```
public class login extends AbstractPageBean {
    //se instancias las clases del capa de negocios
    private negocios.usuario u=new negocios.usuario();
    private negocios.perfil p=new negocios.perfil();

    public login() {

    }

    public usuario getU() {
        return u;
    }

    public void setU(usuario u) {
        this.u = u;
    }

    //método que se ejecuta cuando un usuario hace clic en el botón Ingresar
   public void buttonIngresar_action() {
        try
        {
            //se verifica el usuario y la contraseña ingresados por el usuario
           u.verifica_usuario(this.inputTextNombre.getValue().toString(),
                                SHA1(this.inputSecret1.getValue().toString()));
            if (u.getUsuario()!=null)
            {
                //se redirecciona a la página bienvenido si es que el usuario y la contraseña existe
                String javascriptCode = "window.location ='/Expertize/bienvenido.jsp'";
                JavascriptContext.addJavascriptCall(FacesContext.
                getCurrentInstance(),javascriptCode);
                // Además se agrega el verifica a que páginas tiene permiso el usuario
                // y se agrega como lista a SessionBean
                p.lista_permisos(u.getUsuario().get_perfil_id());
                List<String> permisos=new ArrayList();
                for (int i=0;i<p.getPermisos().size();i++)
                {
                    permisos.add(p.getPermisos().get(i).getPagina_id().getPagina_nombre());
                }
                this.getSessionBean1().setPermisos(permisos);
            }
            else
            {
              // Si el usuario no es válido
              this.outputLabelMensaje1.setValue("La combinacion de usuario y contraseña no
              existe.");
            }
        }
        catch(Exception ex){
            this.outputLabelMensaje1.setValue("Ha ocurrido un error al verificar nombre y
            contraseña");
        }
    }

    }
```

La clase login.java redirecciona a la página bienvenido.jsp si es que el usuario que intenta ingresar al sistema está registrado. Esta verificación se hace a través de la instancia de la clase de usuario de la capa de negocios.

Parte del código de esta clase se muestra a continuación.

```
package negocios;
import java.util.List;
public class usuario {
    dto_mae_usuario usuario= new dto_mae_usuario();
    public String  ing_usuario(negocios.dto_mae_usuario u){
        if (negocios.dao_mae_usuario.guarda(u).equals("ok")){
            return "ok";
        }
        else
        {
            return "error";
        }
    }
    public List<negocios.dto_mae_usuario>  lista_usuario()
    {
        List<negocios.dto_mae_usuario> a=negocios.dao_mae_usuario.lista();
        if (a!=null){
            return a;
        }
        else
        {
            return null;
        }
    }
    public void verifica_usuario(String usuario,String password) throws Exception{
          this.usuario= negocios.dao_mae_usuario.autenticacion(usuario, password);
    }

    public dto_mae_usuario getUsuario() {
        return usuario;
    }

    public void setUsuario(dto_mae_usuario usuario) {
        this.usuario = usuario;
    }
}
```

Como se ve, en negocios.usuario se crean objetos dto que son usados en la capa de presentación y se ejecutan los métodos de negocios a través de las clases de la capa de acceso a datos (clases dao). Los métodos que permiten verificar e ingresar un nuevo usuario se muestran en el código siguiente perteneciente a la clase dao_mae_usuario.

```
package dao;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class dao_mae_usuario {

    public static String guarda(dto.dto_mae_usuario mu){
        try{
          Connection conn = DatabaseHelper.getConnection();
          Statement stmt = conn.createStatement();
          String sql ="Insert mae_usuario (usuario_nombre,usuario_rut,usuario_apellido) values
('"mu.get_usuario_nombre()+"','"+ mu.get_usuario_rut()()+"','+ mu.get_usuario_apellido()+" )";
          int rs = stmt.executeUpdate(sql)
          return "ok";
        }
        catch(Exception ex){
            return "error";
        }
    }
//este método verifica si existe un usuario
    public static dto.dto_mae_usuario autenticacion(String usuario,String password) throws
SQLException {
            Connection conn = DatabaseHelper.getConnection();
            Statement stmt = conn.createStatement();
            String sql = "select * from mae_usuario where user='"+usuario+"' and
            pwd='"+password+"'";
            ResultSet rs = stmt.executeQuery(sql);
            dto_mae_usuario u=new dto_mae_usuario();
            while(rs.next()){
                u.set_usuario_id(rs.getInt("usuario_id"));
                u.setUsuario_rut(rs.getInt("usuario_rut"));
                u.set_usuario_nombre(rs.getString("usuario_nombre"));
                u.set_usuario_apellido(rs.getString("usuario_apellido"));
                u.set_email(rs.getString("usuario_email"));
                u.set_usuario_estado(rs.getString("usuario_estado"));
                u.set_digito_rut(rs.getString("digito_rut"));
                dto_mae_perfil mp=new dto_mae_perfil();
                mp.setPerfil_id(rs.getInt("perfil_id"));
                return u;
            }
            return null;

    }
```

Retomando el proceso de ingreso al sistema, si no ocurre ningún problema el usuario es redireccionado a la página bienvenido.jsp mostrada en la figura 13. En esta página, como en todas las del sistema, está disponible para los usuarios el menú flash que permite acceder a las demás funcionalidades que se ofrecen. Este menú se agrega como un fragmento de página a través del control llamado “Page Frament Box” donde se llama a un archivo que tiene una extensión .jspf.

El código que permite incluir un fragmento en una página jsp es el siguiente:

```
<jsp:directive.include file="menu.jspf"/>
```

Mientras que el código del propio fragmento, que carga el elemento swf, es el siguiente:

```
<f:subview id="menu">
        <object type="application/x-shockwave-flash" data="menu.swf" width="780"
         height="130" title="main" standby="main">
                  <param name="movie" value="menu.swf" />
                 <param name="scale" value="noscale" />
                 <param name="quality" value="best" />
         </object>
</f:subview>
```

Luego de la carga de la página bienvenido el usuario podrá escoger lo que desea hacer en el sistema, para seguir el proceso ejemplo la página cargada será ing_proyectos.jsp mostrada en la figura 14.

Al seguir con el proceso, es decir, si el código de los aspectos no redirecciona a la página de login.jsp o a sinpermisos.jsp, el usuario ingresa

los datos necesarios del nuevo proyecto y selecciona las etapas que estime parte del nuevo proyecto.

Parte del código de la página ing_proyectos.jsp se muestra a continuación.

```
<ice:commandButton action="#{ing_proyectos.buttonGuardar_action}" id="buttonGuardar"
partialSubmit="true" value="Guardar"/>

<ice:outputLabel id="outputLabel1"   value="Nombre"/>

<ice:inputText binding="#{ing_proyectos.inputTextNombre}" id="inputTextNombre"
value="#{ing_proyectos.pr.pr._proyecto_nombre}">

<ice:outputLabel binding="#{ing_proyectos.outputLabel2}" id="outputLabel2"
value="Descripcion:"/>

<ice:inputTextarea binding="#{ing_proyectos.inputTextareaDesc1}" id="inputTextareaDesc1"
value="#{ing_proyectos.pr.pr._objetivo}"/>

<ice:commandButton action="#{ing_proyectos.buttonCancelar_action}" id="buttonCancelar">
```

Nótese que siempre en las páginas se usan las propiedades dto pertenecen a clases de la capa del negocio. El código de la clase ing_proyecto se muestra a continuación.

```
public void buttonGuardar_action() {
        try{
            //se buscan las etapas seleccionadas
            int cantidad_seleccionado=0;
            for (int i=0;i<this._list_etapas.length;i++){
                if (this._list_etapas[i].getSeleccionado()==true){
                    cantidad_seleccionado++;
                }
            }
            //se setea las etapas de la clase negocios.proyecto por las etapas seleccionadas
            this.pr.setEtapas(_list_etapas);

            // Se guarda el proyecto y sus etapas
            if (this.pr.ing_proyecto().equals("ok"))
            {
                this.outputMensaje.setValue("El registro se ha guardado satisfactoriamente");
                this.isvisible=true;
            }
            else{
                this.outputMensaje.setValue("Ha ocurrido un error al guardar el registro");
                this.isvisible=true;
            }
        }
        catch(Exception ex)
        {
            this.outputMensaje.setValue("Ha ocurrido un error al guardar el registro");
             this.isvisible=true;

        }
    }
```

A su vez negocios.proyecto instancia las clases de la capa de acceso a datos (dao), esto se puede ver en el siguiente código.

```
public class proyecto {
    private dto.dto_proyecto pr =new dto.dto_proyecto();
    private dto.dto_mae_etapa[] etapas;
    //para devolver proyectos existentes
    private List<dto.dto_proyecto> proyectos;

    public String  ing_proyecto() {
        try{
                int res_guardar=dao.dao_proyecto.guarda(pr);
                for (int i=0;i<etapas.length;i++)
                {
                    if (etapas[i].getSeleccionado())
                    {
                        dto.dto_etapa_proyecto etapa_proyecto=new dto.dto_etapa_proyecto();
                        etapa_proyecto.set_proyecto_id(res_guardar);
                        etapa_proyecto.set_etapa_id(etapas[i].get_etapa_id());
                        dao.dao_etapa_proyecto.guarda(etapa_proyecto);
                    }
                }
                return "ok";
        }
        catch(Exception ex){
          return "error";
        }
    }

    public void lista_proyecto(int estado){
        this.proyectos =dao_proyecto.lista(estado);
    }

}
```

Como se puede ver en la clase negocios.proyecto el método guardar no tiene un manejo de transacciones por si ocurre un error, esto es por que justamente este proceso se lleva a cabo con aspectos.

Como se muestra en las etapas de diseño, para el manejo de transacciones se utilizan dos aspectos, uno abstracto denominado JDBCTransactionAspect y otro implementa el aspecto abstracto que permite definir a que clases se desea manejar la transacción.

El código de JDBCTransactionAspect es el siguiente:

```
import java.sql.*;

public abstract aspect JDBCTransactionAspect  percflow(topLevelTransactedOperation()) {
    private  Connection _connection;
    protected abstract pointcut transactedOperation();
    protected abstract pointcut obtainConnection();

    protected pointcut topLevelTransactedOperation()
    : transactedOperation() && !cflowbelow(transactedOperation());

    Object around() : topLevelTransactedOperation() {
        Object operationResult;
        try {
           //este es el código especifico que se hace la transaccion
            operationResult = proceed();
            if (operationResult=="error")
            {
                _connection.rollback();
            }
            if (_connection != null) {
                _connection.commit();
            }
        }
        catch (Exception ex) {
            if (_connection != null) {
                _connection.rollback();
            }
            throw new TransactionException(ex);
        }
    finally {
            if (_connection != null) {
                _connection.close();
            }
    }
          return operationResult;
    }

Connection around() throws SQLException
: obtainConnection() && cflow(transactedOperation()) {
    if (_connection == null) {
        _connection = proceed();
        _connection.setAutoCommit(false);
    }
    return _connection;
}

public static class TransactionException
extends RuntimeException {
    public TransactionException(Exception cause) {
        super(cause);
    }
}
}
```

Cada vez que se desee manejar una transacción se debe extender el aspecto JDBCTransactionAspect, en el caso del ingreso de proyectos el aspecto creado es AspectTransaccionProyecto y su código es el siguiente:

```
import java.sql.Connection;

public aspect AspectTransaccionProyecto extends JDBCTransactionAspect {
    protected pointcut transactedOperation()
        :  execution(* negocios.proyecto.ing_proyecto(..))|| execution(*
                    dao.dao_proyecto.guarda(..))
                   || execution (* dao.dao_etapa_proyecto.guarda(..));
    protected pointcut obtainConnection()
        : call(Connection DatabaseHelper.getConnection(..));

}
```

Por último se mostrará una clase dto como ejemplo de las existentes en el sistema. Esta clase es dto_proyecto.java.

```
public class dto_proyecto implements java.io.Serializable {

    public dto_proyecto() {

    }

    private int _proyecto_id;
    private String _proyecto_nombre;
    private Date _fecha_creacion;
    private Date _fecha_inicio;
    private Date _fecha_fin;
    private String _objetivo;
    private String estado_proyecto;
    private dto_mae_usuario usuario_id=new dto_mae_usuario();
    private dto_mae_tipo_proyecto tipo_proyecto_id=new dto_mae_tipo_proyecto();
    private dto_mae_departamento departamento_id=new dto_mae_departamento();


    public dto_mae_departamento getDepartamento_id() {
        return departamento_id;
    }

    public void setDepartamento_id(dto_mae_departamento departamento_id) {
        this.departamento_id = departamento_id;
    }

    public dto_mae_tipo_proyecto getTipo_proyecto_id() {
        return tipo_proyecto_id;
    }

    public void setTipo_proyecto_id(dto_mae_tipo_proyecto tipo_proyecto_id) {
        this.tipo_proyecto_id = tipo_proyecto_id;
    }

    public dto_mae_usuario getUsuario_id() {
        return usuario_id;
    }
```

Finalmente se puede resumir que según lo visto con el proceso ejemplo, las peticiones de los usuarios pasan por las clases de distintas capas interviniendo además los aspectos necesarios para que el sistema cuente con seguridad e integridad de datos.

8.6 Pruebas de Unidad

Se realizaron pruebas de unidad para validar el funcionamiento de las clases que pertenecen a la capa de negocios. Estas pruebas se codificaron con la herramienta Junit que es el ambiente de pruebas más conocido en plataforma Java.

En la siguiente figura se pueden ver la ejecución de las pruebas de unidad creadas.

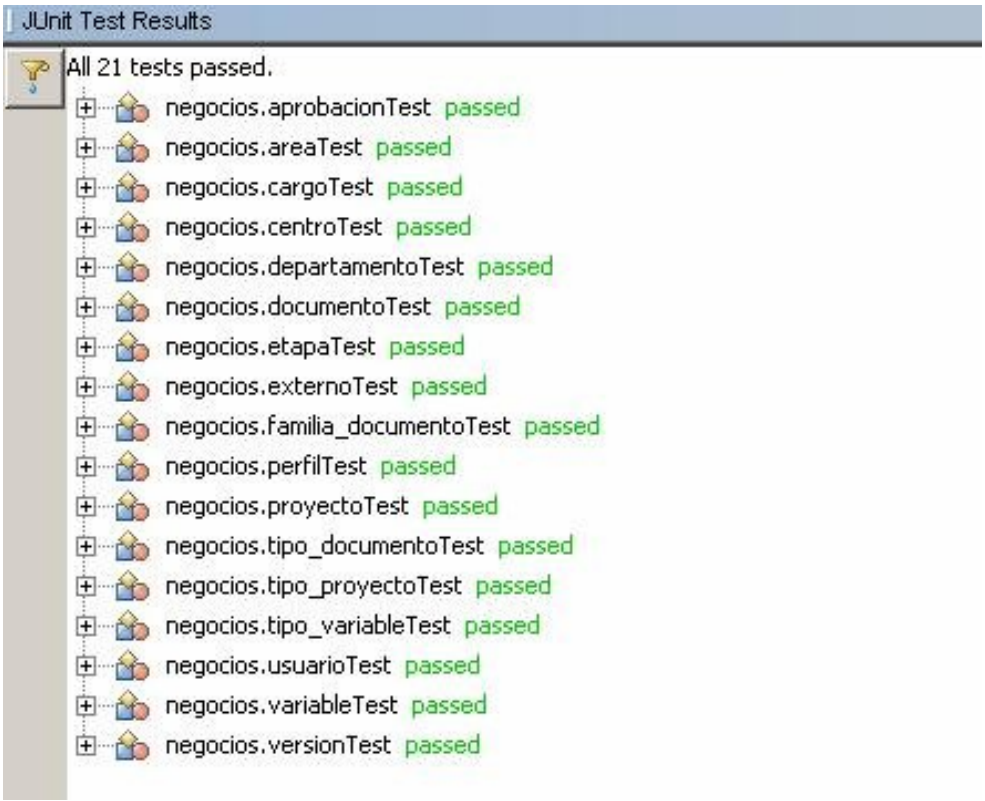


Figura 82: Ejecución de pruebas creadas para la capa de negocios

9. Transición

En esta etapa el sistema se lleva al entorno de preproducción y se hacen pruebas principalmente de aceptación, es decir, validación de parte del cliente que realmente es lo que planteó. Además se lleva a cabo tareas como arreglar errores, finalizar el despliegue, finalizar la documentación y planificar la capacitación a usuarios.

9.1 Pruebas de Aceptación

Como se dijo anteriormente las pruebas de aceptación consisten en que el cliente verifica si realmente el sistema cumple con lo que el requería. Estas pruebas se llevan a cabo de acuerdo a una serie de ejercicios creados para que el usuario administrador del sistema (cliente) valide las funcionalidades y también se familiarice con las interfaces. Estas pruebas de aceptación se pueden ver en el anexo B.

9.2 Finalizar el paquete de despliegue

La implementación del sistema en el servidor de aplicaciones se puede ver en el anexo C.

10. Conclusiones y recomendaciones

10.1 Conclusiones

El objetivo de este seminario era realizar una aplicación Web que permita administrar, crear nueva documentación y controlar procesos productivos en la industria acuícola. Al final del proyecto se puede contar con una herramienta que cumple con las necesidades del cliente permitiendo dar apoyo en estas áreas alcanzando con éxito el objetivo planteado.

Desde el punto de vista de funcionamiento y en base a los objetivos específicos se puede concluir:

- Se desarrolló un módulo cuyas funcionalidades permiten apoyar con el proceso de creación de documentos, estas funcionalidades hechas en Java son fáciles de usar y hechas con mucho apoyo del cliente (usuario final) por lo que se llegó a interfaces hechas a la medida.

- Se desarrolló un módulo que permite centralizar los documentos existentes y publicar los creados con la ayuda del sistema, este módulo permite tener filtros de búsqueda y mantener los documentos existentes para facilitar a los usuarios el uso del sistema. Es el módulo más usado dentro del sistema porque

facilita enormemente al usuario la búsqueda de documentos y la administración de estos.

- Se desarrolló un módulo que permite crear controles checklist para procesos específicos del ciclo productivo y controlar todo el proceso de planificación de estos controles y los resultados obtenidos.

Con respecto a la utilización de la programación orientada a aspectos se puede concluir:

- Que esta técnica cumple con la función que teóricamente plantea, es decir, se puede manejar separadamente lo que se defina como transversal en todo el sistema
- Permite ahorrar tiempo de codificación puesto que si se quiere enviar un correo en otra parte del sistema, por ejemplo, basta con cambiar el código del aspecto agregando cuales son las clases que intervienen en el envío de correo, sin necesidad de cambiar el código de cada una de esas clases.
- Permite tener un código más limpio y modular.
- Se cree que al ser una tecnología aún muy reciente, la curva de aprendizaje es muy alta principalmente por no contar con muchas herramientas que permitan modelar y codificar aspectos.

- Sin embargo, pese a encontrarse en una fase de desarrollo se cree que en el futuro se consolidará como gran apoyo a la programación orientada a objetos.

Para finalizar se puede agregar que el uso de la plataforma de desarrollo Java ha sido un muy buen acierto dentro del proyecto, se han usado herramientas libres como Netbeans, Icefaces y Tomcat que han dado excelentes resultados por su facilidad de uso, abundante documentación y estabilidad en ambiente de producción.

Aunque este proyecto ha sido creado orientado a la industria acuícola los requisitos son comunes en diferentes industrias, prueba de ello es que posterior a este desarrollo se ha adaptado el sistema para el área construcción obteniendo muy buenos resultados.

En una segunda etapa del proyecto se recomienda la localización del sistema, esto no debería llevar mucho tiempo puesto que es una de las características principales de framework utilizado (JSF).

11. Bibliografía

[Adobe2006]
Manual de Referencia Adobe Flash 8
Adobe.com

[Ambler2006]
Ambler, Scott W: Techniques for Successful Evolutionary/Agile Database Development disponible en: www.agiledata.org
2006

[Asteasuain2002]
Asteasuain Fernando, Contreras Bernardo
Programación orientada a aspectos, análisis del paradigma
Tesis de Licenciatura Universidad Nacional del sur
2002

[Basch2003]
Mark Basch, Arturo Sanchez. Incorporating Aspects into the UML
Proceedings of Third International Workshop on Aspect, Disponible en:
http://lglwww.epfl.ch/workshops/aosd2003/papers/Basch-IcorporatingAspectsIntotheUML.pdf
2003

[Barra2004]
Eduardo Barra, Gonzalo Génova, Juan Llorens
An approach to Aspect Modelling with UML 2.0 Computer Science Department University Carlos III (AOM'04)
Disponible en: http://www.ie.inf.uc3m.es/ggenova/pub-uml2004c.html
2004

[Bustos2007]
Alex Bustos, Yadran Eterovic, Modeling Aspect UIT UML´s class, sequence and state diagrams in an industrial setting. Disponible en:
http://blog.radaworks.cl/wp-content/uploads/2007/10/sea2007.pdf
2007

[Connolly2005]
Connolly, Begg.
Un enfoque práctico para diseño, implementación y gestión.
Pearson. Cuarta edición.
2005

[Dudnay2004]
Dudnay, Lehr, Willis, Mattingly.
Mastering Java Server Faces Wiley.

2004

[Fowler2002]
Fowler, Rice, Foemmel, Hieatt, Mee, Stafford.
Patterns of Enterprise Application Architecture.
Addison Wessley.
2002

[Fowler1999]
Martin Fowler, Kendall Scott.
Uml gota a gota
Pearson
1999

[Geary2007]
David Geary, Cay Horstmann
Core JavaServer™ Faces
Sun Microsystems Segunda Edición
2007

[Herrero2000]
J.L. Herrero, M. Sánchez y F. Sánchez, Changing UML metamodel in order to represent separation of concerns. ECOOP'2000 Workshop (Sophia Antipolis, France). Disponible en: http://portal.acm.org/citation.cfm?id=508399
Junio 2000

[INTESAL2004]
Instituto Tecnológico del salmón, Manual de implementación SIGES
Disponible en: http://www.siges-salmonchile.cl/proysiges/
2004

[Larman 2002]
Craig Larman
Uml y Patrones
Pearson Segunda Edición
2002

[Kickzales2007]
Kickzales, Lamping, Mendhekar, Maeda, Lopes, Loingtier, Irwin.
Aspect-Oriented Programming, in Proceedings of the European Conference on Object-Oriented Programming (ECOOP), Finlandia
1997

[Küng2006]
Stefan Küng, Lübbe Onken y Simon Large
"Un cliente de Subversion para Windows"

2006

[Laddad2003]
Laddad Ramnivas
Aspectj in Action, Practical Aspect-Oriented Programming.
Manning Primera edición.
2003

[Matthijs1997]
Matthijs, Joosen, Vanhaute, Robben, Verbaeten.
Artículo “Aspects should not die.” European Conference on Object-Oriented Programming
1997

[Miles2004]
Miles Russ
AspectJ Cookbook
O'Reilly
2004

[Netbeans2008]
Netbeans.org: NetBeans IDE 6.1 Features
Disponible en: http://www.netbeans.org/features/
2008

[Richardson2006]
Richardson Chris.
POJOS in Action, Manning
2006

[Sun1997]
Especificación JavaBeans.
Sun Microsystems, Editor Graham Hamilton
1997

[Sun2002]
Patrones J2EE: Sun Microsystems.
Disponible en: http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html
2002

[Suzuki1999]
Suzuki -Yamamoto
Extending UML withAspects: Aspect Support in the Design Phase. 3rd AOP workshop at ECOOP’99 (Lisbon, Portugal). Disponible en: http://citeseer.ist.psu.edu/suzuki99extending.html
Junio 1999

12. Anexos

12.1 Anexo A :

Instalación y uso de plugin aspectj en netbeans

En la actualidad existe un plugin creado por Ramón Ramos que permite utilizar aspectj en la interfaz de desarrollo Netbeans 6.X.

A continuación se muestra paso a paso como instalar esta herramienta.

1. Descargar el plugin desde la central de plugins de Netbeans.org, específicamente desde la página http://plugins.netbeans.org/PluginPortal/faces/?pluginid=4015.

2. Descomprimir el archivo zip y luego ir a Menu->Tools -> Plugins, en esta ventana seleccionar la pestaña “Descargados”.

3. Presionar el botón “Add Plugins” y elegir los archivos nbm anteriormente descomprimidos.

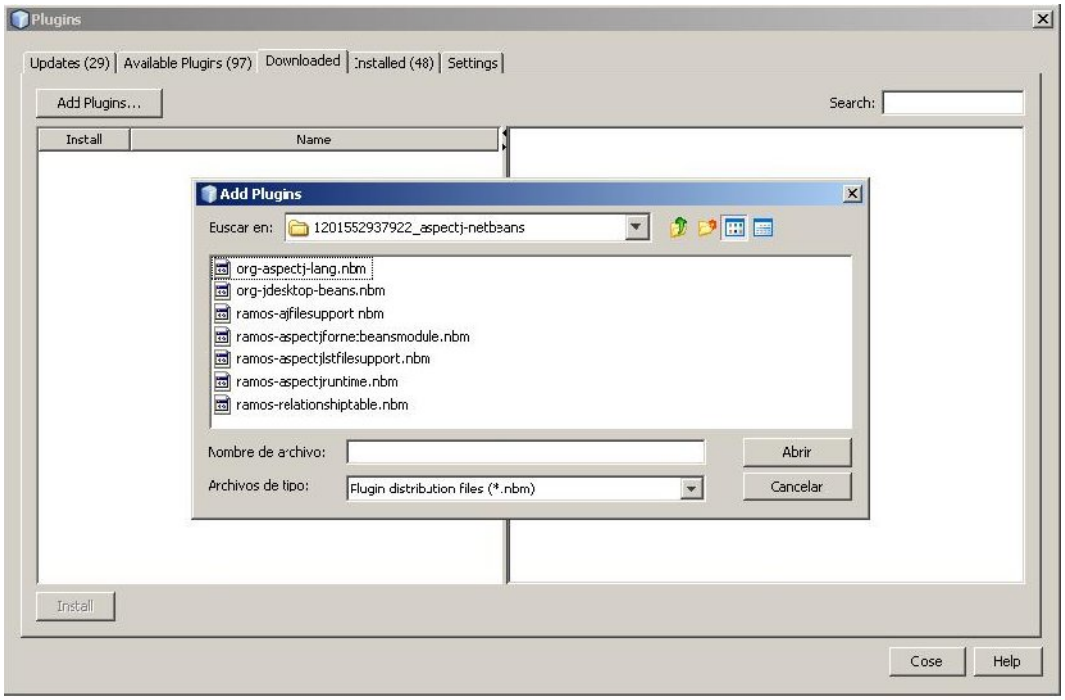


Figura 83: Carga de plugin en Netbeans

4. Una vez elegido los archivos presionar el botón “Install”. El proceso de instalación tardará algunos minutos y una vez terminado se debe reiniciar Netbeans.

Principalmente el plugin instalado permite crear aspectos, compilar los aspectos creados y ayudar a los usuarios indicando gráficamente a que clases interfieren los avisos (advices) de cada aspecto.

1. Como puede verse en la figura al agregar un nuevo elemento aparece ahora un ítem denominado Aspectj. Al seleccionar este ítem se pueden elegir las siguientes alternativas :
    - Empty Aspect File: Crea un nuevo aspecto en el proyecto con la extensión “aj”.
    - Empty Lst File: Crea un archivo vacío con extensión “lst”, que representa un archivo de configuración para la compilación de los aspectos.
    - Source Root Lst File: Crea un archivo con extensión “lst”, pero esta con las variables por defecto para que se efectúe la compilación.

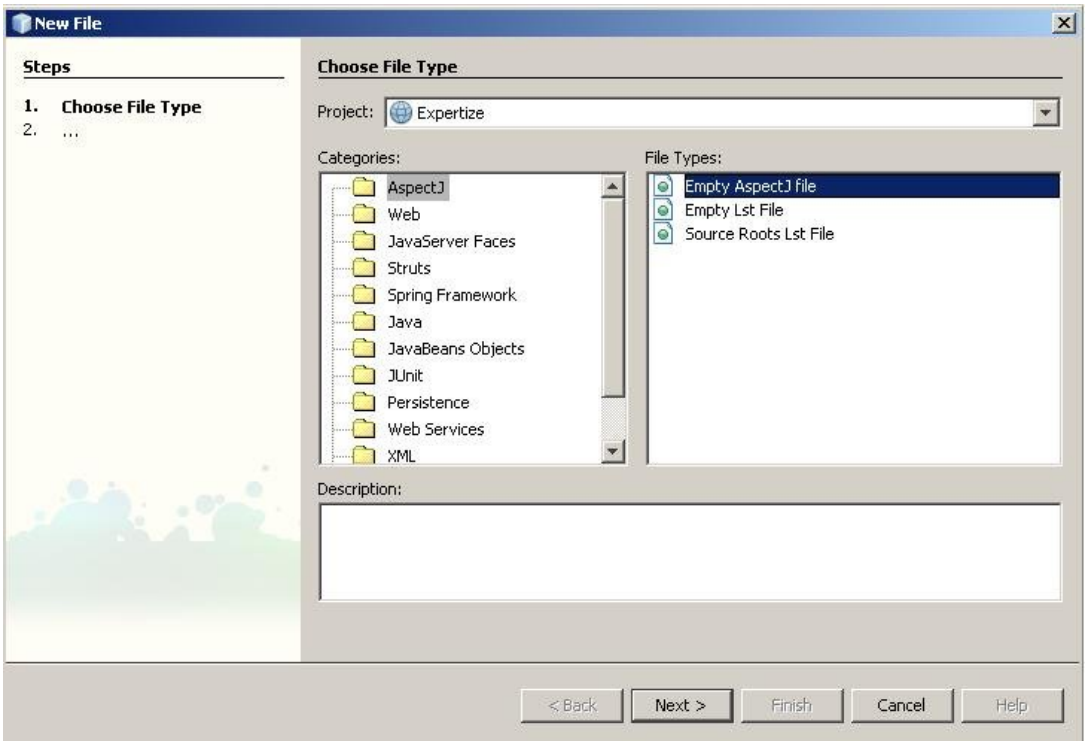


Figura 84: Creación de un nuevo aspecto

2. Luego de agregar un nuevo aspecto se debe activar la utilización de AJDE (AspectJ Development Environment) y posteriormente seleccionar un archivo de configuración. Esto se hace utilizando la nueva opción existente en el menú de Netbeans (Tools -> Aspectj).

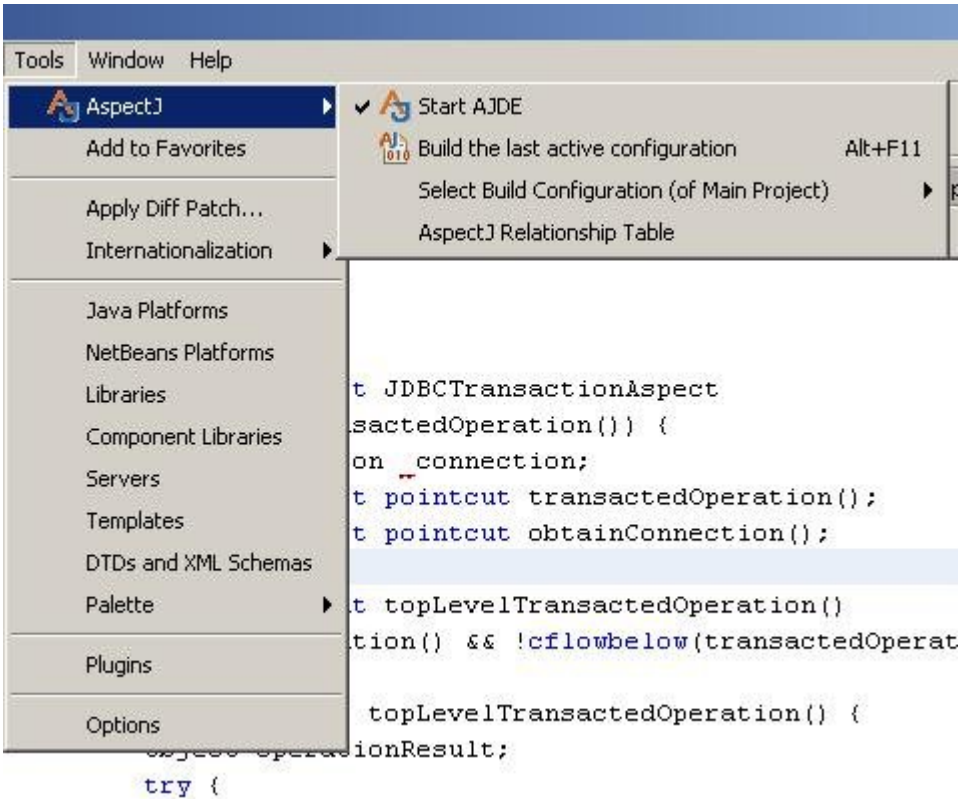


Figura 85: Activar utilización de AJDE

3. Finalmente se puede ver en la ventana de resultados de compilación de aspectj si los aspectos tenían errores, en caso contrario, si el proceso de compilación no falla se puede ver finalmente a que clases

afectan los aspectos compilados en la tabla de relaciones de aspectos.

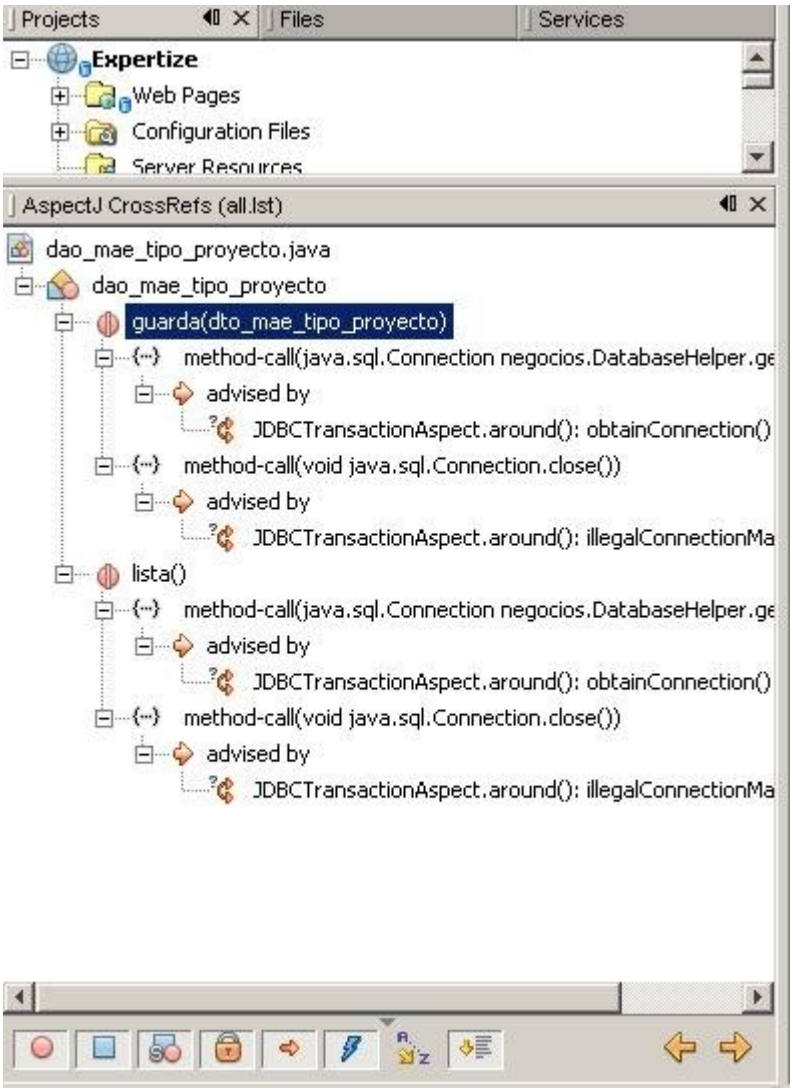


Figura 86: Resultados compilación Aspectj

12.2 Anexo B

Pruebas de aceptación del sistema

Esta es una tarea llevada a cabo antes de la puesta en marcha del sistema. Se realizan un conjunto de pruebas para que el usuario administrador (cliente) valide el buen funcionamiento del sistema y a su vez se familiarice con sus interfaces.

A continuación se muestran las pruebas de aceptación elaboradas.

Descripción	Agregar nuevo Proyecto
Instrucciones	• Colocar como nombre proyecto:"Proyecto de Prueba". • Colocar fecha estimada al proyecto • Elegir un responsable al proyecto • Agregar dos etapas al proyecto • Guardar el proyecto.
Resultados esperados	Un mensaje de éxito del sistema.

Descripción	Agregar grupo de trabajo
Instrucciones	• Elegir proyecto • Agregar usuarios y cargos • Guardar cambios al grupo de trabajo
Resultados esperados	Un mensaje de éxito del sistema.

Descripción	Asignar Tareas
Instrucciones	• Elegir proyecto • Agregar “tarea 1” • Agregar “tarea 2”
Resultados esperados	Un mensaje de éxito del sistema.

Descripción	Agregar documento a crear
Instrucciones	• Elegir tipo de documento:”Procedimientos” • Nombre del documento: Por Ej. “Norma calidad” • Departamento: “Producción Agua Mar”
Resultados esperados	Un mensaje de éxito del sistema.

Descripción	Subir Documento
Instrucciones	• Elegir tipo de documento:”Procedimientos” • Nombre del documento: Por Ej. “Procedimiento Muestreo de peces” • Departamento: “Producción Agua Mar” • Autor: “Trusal”
Resultados esperados	Un mensaje de éxito del sistema.

Descripción	Buscar Documento
Instrucciones	• Elegir tipo de documento:”Procedimientos” • Elegir fecha de publicación • Abrir documento
Resultados esperados	El documento elegido

Descripción	Registrar un nuevo control
Instrucciones	• Ingresar nombre y objetivo • Asignar al menos 3 variables de control • Guardar el control
Resultados esperados	Un mensaje de éxito del sistema.

Descripción	Registrar una planificación
Instrucciones	• Buscar el control ingresado previamente. • Asignar centro, responsable y fecha • Guardar la planificación
Resultados esperados	Un mensaje de éxito del sistema.

12.3 Anexo C

Levantamiento del sistema en ambiente de producción

El sistema corre sobre un servidor Apache Tomcat 6 o superior. Puede ser implementado de distintas maneras, una de ellas es utilizar directamente las funcionalidades de Netbeans para hacer la publicación en el servidor. Otra de las formas es la que se muestra a continuación que consiste resumidamente en hacer la compilación utilizando Netbeans y luego cargarlo al servidor de producción a través de la página de configuración de Apache Tomcat.

Primero se debe tener instalada la versión indicada de Apache Tomcat, si esto es efectivo se acceder al administrador del servidor con un navegador accediendo al puerto especificado en la instalación, por ejemplo: http://localhost:8080/.

Si Apache Tomcat está correctamente instalado debe aparecer la siguiente página:

Figura 87: Página inicial de Apache Tomcat

Posteriormente se debe acceder al link del costado izquierdo de la página “Tomcat Manager” y se debe ingresar nombre de usuario y contraseña del servidor. Si el usuario y contraseña son correctos se pasa al “Gestor de Aplicación de Tomcat” como se muestra en la siguiente pantalla.

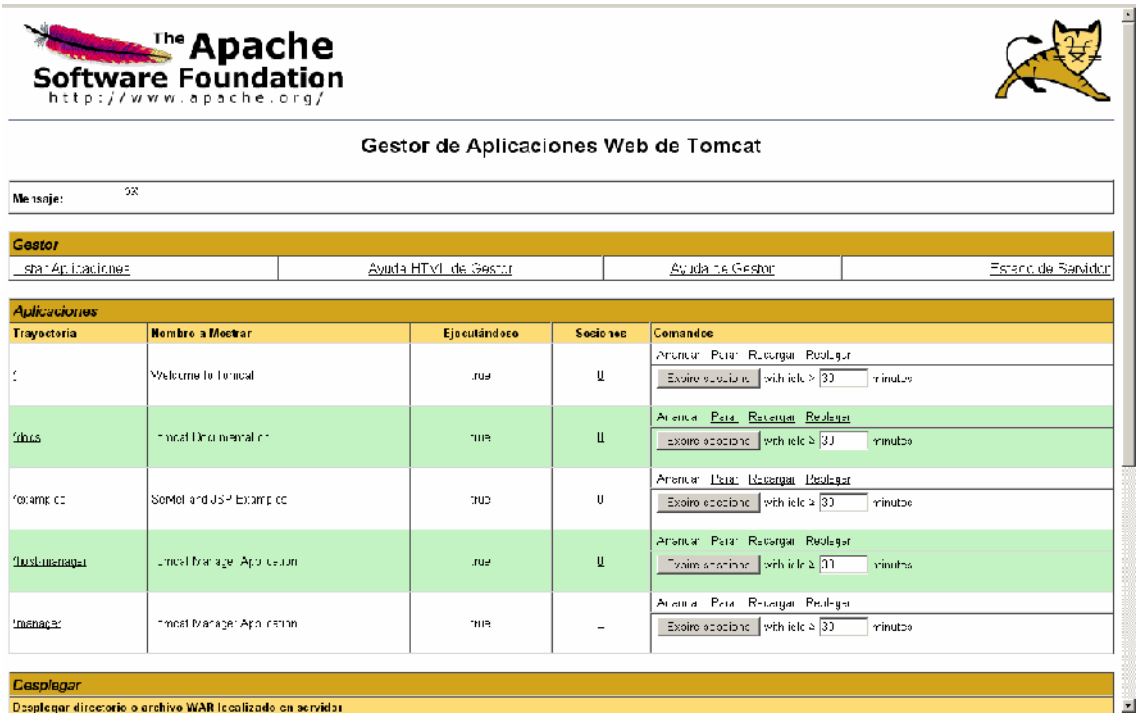


Figura 88: Página gestor de aplicaciones Tomcat

En esta pantalla se puede arrancar, parar, recargar o replegar una aplicación especifica o bien cargar una nueva aplicación con un archivo war generado previamente. Un archivo war es un archivo jar creado por Sun que empaqueta paginas jsp, html, xml y clases compiladas .class. El archivo war generado por Netbeans se puede encontrar en la carpeta del proyecto, específicamente en la ruta nombre_proyecto/dist.

Para la implementación de la aplicación entonces, se carga el archivo war de la aplicación en el gestor de aplicaciones de tomcat. Esto se hace en la parte inferior de la página mostrada figura 89.

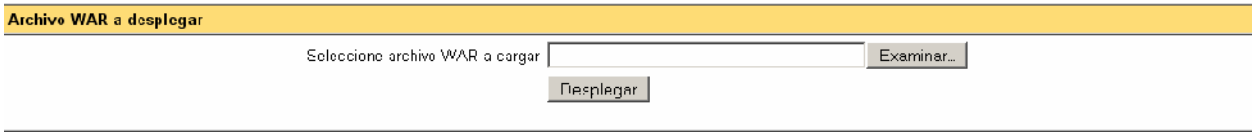


Figura 89: Pantalla donde se selecciona el archivo WAR

Una vez cargada la aplicación se puede ver en la lista de aplicaciones y se puede acceder a través de la dirección y el puerto en el navegador.

12.4 Anexo D

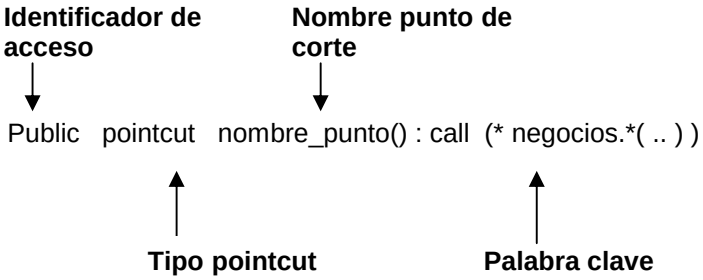
Sintaxis de Aspectj

Aspectj tiene una sintaxis muy simple y fácil de comprender, además que todo código de Java puede ser utilizado en Aspectj. Este anexo está hecho en gran parte según la traducción de [Laddad2003].

Los puntos de corte (pointcuts) capturan o identifican puntos de enlace en el flujo del programa. Una vez definidos con los constructores que proporciona el lenguaje, puede especificarse mediante reglas de entrelazado cómo combinar los comportamientos definidos en los aspectos con el código original. Además los puntos de corte dan acceso al contexto del programa y las acciones se pueden beneficiar de la información contextual para llevar a cabo su cometido.

En AspectJ las pointcuts también se pueden declarar anónimos o con un nombre. Al igual que las clases, un punto de corte anónimo se define en el lugar donde se usa, que bien podría ser en un advice o en la definición de otro pointcut. Los puntos de corte que se definen con un nombre pueden después ser referenciados desde múltiples partes del código, haciéndolos reusables.

Su declaración básica de un punto de corte con nombre es la siguiente:



Los puntos de corte harán referencia a una serie de puntos de enlace (*joinpoints*) en el programa, por lo que se necesita un lenguaje eficiente para definirlos. AspectJ utiliza una sintaxis basada en elementos comodines que sirven para capturar puntos de enlace que comparten características comunes. Dichos elementos son los siguientes [Nieto]:

- * un asterisco denota cualquier número de caracteres excepto el punto.
- .. dos puntos seguidos denota cualquier número de caracteres incluyendo el punto.
- + el símbolo de la suma denota los descendientes de una clase.

Al igual que en Java y en muchos lenguajes las expresiones se pueden combinar para dar lugar a expresiones más complejas. Para ello se utilizan los siguientes operadores unarios y binarios:

- ! la exclamación de cierre es un operador unario que denota todos los puntos de enlace, excepto aquellos que se especifiquen en la expresión que siga. Por ejemplo, !within(aspects.*) denota todos los puntos de enlace excepto aquellos del paquete aspects.

- || dos barras verticales es el operador binario que permite combinar puntos de enlace con nombre y anónimos, de tal forma que se cumpla alguno de los lados de la expresión.

- && es el operador binario que permite combinar puntos de enlace con nombre y anónimos, de tal forma que se cumplan ambos lados de la expresión.

Una vez que se tiene una manera eficiente de definir los puntos de enlace en el código, se necesita una manera de utilizarlos. Para referirse a ellos se puede hacer según el tipo de punto de enlace al que representan, como puede ser llamadas a métodos, ejecución de estos, lectura de atributos, etc., estos tipos se pueden ver en la siguiente tabla:

Categoría	Sintaxis
Ejecución de métodos	execution(MethodSignature)
Llamada a métodos	call(MethodSignature)
Ejecución de constructores	execution(ConstructorSignature)
Llamada a constructores	call(ConstructorSignature)
Inicialización de clases	staticinitialization(TypeSignature)
Lectura de atributos	get(FieldSignature)
Escritura de atributos	set(FieldSignature)
Captura de excepciones	handler(TypeSignature)
Inicialización de objetos	initialization(ConstructorSignature)
Pre-inicialización de objetos	preinitialization(ConstructorSignature)
Ejecución de avisos	adviceexecution()

Tabla 45 : Tipos de Puntos de corte

Con estos elementos es posible interceptar las clases del lenguaje y codificar así los asuntos transversales del sistema.