

UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

“Red colaborativa para el desarrollo de proyectos online”

DIRECTOR: Daniel Fernández Lanvin



Vº Bº del Director del
Proyecto

AUTOR: Armando Ramírez Vila

Agradecimientos

Quisiera, con la más plena sinceridad, dedicar este proyecto a mi director por confiar en mí, a mis amigos y compañeros por estar siempre presentes y a mi familia por apoyarme durante tres largos y duros años.

¡Muchas gracias!

Resumen

Este proyecto ha consistido en el desarrollo de un sistema distribuido, que soporta entre otras, una aplicación web enriquecida, cuyo fin es permitir a sus usuarios desarrollar sus tareas de forma conjunta. La aplicación ofrece al usuario una simbiosis del concepto de proyecto y de red social, ofreciendo las funcionalidades típicas de una red social para un fin distinto, el desarrollo de actividades con unos objetivos, unas limitaciones y una duración. El sistema ofrece una red de contactos comunicados tanto por mensajería instantánea como por mensajes que pueden crear proyectos tanto individualmente como en grupo. Las acciones de los usuarios sobre los proyectos se basan resumiendo en gestionar recursos, referencias, discutir y planificar. Además, el sistema ayuda a los contactos a organizar reuniones presenciales, fuera ya del marco de un proyecto.

Se ha desarrollado una arquitectura distribuida verticalmente en n niveles, con el fin de obtener escalabilidad y flexibilidad, como muestra de la potencia proporcionada por dicha arquitectura se ha desarrollado un cliente de escritorio para la administración del sistema.

Para el desarrollo del sistema se ha usado tecnología JEE6, apostando por la especificación ante los frameworks de terceros o estándares de facto, no sin tener que recurrir a ellos. Como implementación de JEE6 se ha apostado por las ofertas *Open Source* de *JBoss Community*, de entre las especificaciones usadas destacan, a mi modo de ver CDI y EJB, claves en la arquitectura y el diseño de todo el software.

A la tecnología JEE se ha añadido el *Google Web Toolkit*, enriquecido con un framework de componentes visuales para dicha tecnología, *GXT*, y con otro proyecto de *JBoss* para el desarrollo de aplicaciones basadas en *GWT*, *Errai*.

Se usa la infraestructura de mensajería instantánea de *Google*, y Apache Subversion para la gestión de los recursos de los usuarios, que en su más bajo nivel de abstracción son ficheros.

Los objetivos de este proyecto son netamente académicos, lejos de lo que pueda parecer, no existe un cliente real así como tampoco un plan de negocio. Dichos objetivos consisten pues, en realizar un proyecto con un perfil técnico alto, que reafirme los conocimientos adquiridos durante los 3 años de carrera y complemente mi cartera técnica mediante la incorporación tanto de nuevas tecnologías, herramientas de desarrollo y middleware como de aptitudes relacionadas con la arquitectura y el diseño del software, campos en los que tengo particular interés.

Otro de los mencionados objetivos académicos pasa por el desarrollo de un proyecto orientado al usuario, con lo que se persigue reafirmar y desarrollar habilidades de ingeniería del software que se requieren en este tipo de proyectos.

Palabras Clave

- JEE.
- EJB.
- CDI.
- GWT.
- GXT.
- Proyecto.
- Contacto.
- Mensaje.
- Recurso.
- Tarea.

Abstract

This project is about to develop a distributed system, which gives support to an enrichment web application and others, the application gives to users the functionality of develop their activities in groups.

Application mentioned above, offers to users a merge between project and social network concepts, giving them all functionalities of a typical social network in order to develop projects.

The system offers a contacts network full communicated, both instantaneous messaging and just messaging. The actions that users can do on their projects are resources management, references management, discussions and scheduling. Moreover users can plan meetings, out of a concrete project scope.

It has been developed a vertical distributed architecture, separated in n levels in order to get scalability and flexibility. As sign of advantages of this architecture was developed a desktop based client for system management tasks.

This system is based on Java EE 6 technology, doing a bet for specifications against third part frameworks or in fact standards, although not without these.

The choice of the Java EE6 implementation was based on JBoss Community projects. Most relevant technologies are, in my opinion CDI and EJB, very important for all architecture and design of the software.

JEE technology was complemented with Google Web Toolkit, and this was enrichment with a framework of widgets for this technology, GXT, and with another JBoss project for GWT based applications, Errai.

It uses Google's infrastructure for instantaneous messaging and Apache Subversion for the management of user resources, which are files in their lowest abstraction level.

The objectives of this project are clearly academics and intellectuals. Far from it may seem, there is not a real client neither a business plan. The mentioned objectives consist on a high technical profile's project and confirming knowledge acquired during the last three years career and completes my technical charter by incorporate new technologies, development tools, middleware and skills related with architecture and design of software, areas which I am very interested on.

Another of the academic objectives mentioned above, is about to develop a user oriented project, in order to reaffirm and gets the Software engineering skills that are narrowly related with these kinds of projects.

Keywords

- JEE.
- CDI.
- EJB.
- GWT.
- GXT.
- Project.
- Contact.
- Message.
- Resource.
- Task.

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO.....	23
1.1 RESUMEN DE LA MOTIVACIÓN, OBJETIVOS Y ALCANCE DEL PROYECTO	23
1.2 RESUMEN DE TODOS LOS ASPECTOS.....	24
1.2.1 <i>Resumen General</i>	24
1.2.2 <i>Análisis</i>	25
1.2.3 <i>Diseño</i>	25
1.2.4 <i>Implementación</i>	25
CAPÍTULO 2. INTRODUCCIÓN.....	27
2.1 JUSTIFICACIÓN DEL PROYECTO.....	27
2.2 OBJETIVOS DEL PROYECTO.....	28
CAPÍTULO 3. ASPECTOS TEÓRICOS.....	31
3.1 JEE 6	31
3.1.1 <i>EJB 3.1</i>	32
3.1.2 <i>JPA 2.0</i>	34
3.1.3 <i>CDI 1.0</i>	35
3.1.4 <i>Servlet 3.0</i>	36
3.1.5 <i>JBoss AS 7</i>	37
3.2 SPRING SECURITY.....	39
3.3 GOOGLE WEB TOOLKIT.....	41
3.3.1 <i>Patrón MVP</i>	41
3.3.2 <i>Patrón Composite</i>	42
3.3.3 <i>EXT GWT</i>	42
3.3.4 <i>Errai</i>	43
3.4 SWING.....	44
3.4.1 <i>Swingx</i>	44
3.4.2 <i>Appframework</i>	44
3.5 PROTOCOLO XMPP	45
3.5.1 <i>Google Talk Server</i>	45
3.5.2 <i>Smack API</i>	45
3.6 APACHE SUBVERSION	47
3.6.1 <i>Visual SVN Server</i>	47
3.6.2 <i>SVN KIT</i>	48
3.7 POSTGRESQL	49
CAPÍTULO 4. PLANIFICACIÓN DEL PROYECTO Y RESUMEN DE PRESUPUESTOS	51
4.1 PLANIFICACIÓN.....	51
4.1.1 <i>Perfiles del Proyecto</i>	51
4.1.2 <i>Tareas del Proyecto</i>	52
4.1.3 <i>Diagrama de Gant</i>	52
4.2 RESUMEN DEL PRESUPUESTO	54
CAPÍTULO 5. ANÁLISIS	55
5.1 DEFINICIÓN DEL SISTEMA	55

5.1.1	<i>Introducción</i>	55
5.1.2	<i>Descripción</i>	55
5.2	REQUISITOS DEL SISTEMA.....	61
5.2.1	<i>Obtención de los Requisitos del Sistema</i>	61
5.2.2	<i>Especificación y Diagramas de Casos de Uso</i>	74
5.3	CLASES DEL ANÁLISIS.....	129
5.3.1	<i>Diagrama de Clases del Análisis</i>	129
5.3.2	<i>Descripción de las Clases del Análisis</i>	130
5.4	ANÁLISIS DE INTERFACES DE USUARIO	136
5.4.1	<i>Descripción de la Interfaz</i>	136
5.4.2	<i>Descripción de del Comportamiento</i>	146
5.5	ESPECIFICACIÓN DEL PLAN DE PRUEBAS	147
5.5.1	<i>Pruebas Unitarias</i>	147
5.5.2	<i>Pruebas de Integración</i>	147
5.5.3	<i>Pruebas del Sistema</i>	149
5.5.4	<i>Pruebas de Usabilidad</i>	149
CAPÍTULO 6. DISEÑO DEL SISTEMA.....		151
6.1	ARQUITECTURA DEL SISTEMA.....	151
6.1.1	<i>Patrones Arquitectónicos</i>	151
6.1.2	<i>Diagramas de Componentes</i>	153
6.1.3	<i>Diagramas de Despliegue</i>	161
6.1.4	<i>Diagramas de Paquetes</i>	164
6.2	DISEÑO DE CLASES	169
6.2.1	<i>Patrones de Diseño</i>	169
6.2.2	<i>Inyección de Dependencias</i>	173
6.2.3	<i>Programación Orientada a Aspectos</i>	173
6.2.4	<i>Diagramas de Clases</i>	174
6.3	DIAGRAMAS DE SECUENCIA	217
6.3.1	<i>Gestión de Usuarios</i>	217
6.3.2	<i>Gestión de Contactos</i>	218
6.3.3	<i>Gestión de Proyectos</i>	220
6.3.4	<i>Gestión de Recursos</i>	221
6.3.5	<i>Autenticación</i>	222
6.3.6	<i>Historial</i>	223
6.4	DIAGRAMAS DE ACTIVIDAD Y DE ESTADO	225
6.4.1	<i>Añadir Contacto</i>	225
6.4.2	<i>Crear Recurso</i>	226
6.4.3	<i>Ciclo de Vida de un Proyecto</i>	228
6.4.4	<i>Ciclo de Vida de una Tarea</i>	229
6.4.5	<i>Ciclo de Vida de una Discusión</i>	230
6.4.6	<i>Ciclo de Vida de una Reunión</i>	231
6.5	DISEÑO DE LA INTERFAZ.....	232
6.5.1	<i>Sección Principal</i>	232
6.5.2	<i>Sección de Información</i>	232
6.5.3	<i>Sección de Chat</i>	235
6.5.4	<i>Sección de Notificaciones</i>	236
6.5.5	<i>Sección de Mensajes</i>	237
6.5.6	<i>Sección de Contactos</i>	238

6.5.7	<i>Sección de Reuniones</i>	239
6.5.8	<i>Sección de Proyectos</i>	240
6.6	ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS	248
6.6.1	<i>Pruebas Unitarias</i>	248
6.6.2	<i>Pruebas de Integración y del Sistema</i>	248
6.6.3	<i>Pruebas de Usabilidad</i>	254
CAPÍTULO 7. IMPLEMENTACIÓN DEL SISTEMA		259
7.1	LENGUAJES DE PROGRAMACIÓN	259
7.2	CONVENIOS ADOPTADOS	260
7.2.1	<i>Convenios de Codificación</i>	260
7.2.2	<i>Convenios de Nombrado</i>	260
7.2.3	<i>Convenios de Log</i>	261
7.3	HERRAMIENTAS Y PROGRAMAS USADOS PARA EL DESARROLLO.....	262
7.3.1	<i>Eclipse Indigo</i>	262
7.3.2	<i>Apache Maven</i>	263
7.3.3	<i>API VIZ</i>	264
7.3.4	<i>Graphviz</i>	264
7.3.5	<i>PgAdmin</i>	265
7.3.6	<i>Enterprise Architect</i>	265
7.3.7	<i>Notepad ++</i>	265
7.4	CREACIÓN DEL SISTEMA	266
7.4.1	<i>Problemas Encontrados</i>	266
7.4.2	<i>Métricas</i>	268
7.4.3	<i>Descripción Detallada de las Clases</i>	270
CAPÍTULO 8. DESARROLLO DE LAS PRUEBAS		271
8.1	PRUEBAS UNITARIAS	271
8.1.1	<i>Pruebas Unitarias Capa de Persistencia</i>	271
8.1.2	<i>Pruebas Unitarias Capa de Negocio</i>	274
8.1.3	<i>Pruebas Unitarias Capa de Presentación</i>	278
8.2	PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA	280
8.3	PRUEBAS DE USABILIDAD	287
8.3.1	<i>Resultados Preguntas de Carácter General</i>	287
8.3.2	<i>Preguntas Cortas sobre la Aplicación y Observaciones</i>	288
8.3.3	<i>Cuestionario para el Responsable de las Pruebas</i>	288
8.3.4	<i>Comprobación Heurísticos de Usabilidad</i>	289
CAPÍTULO 9. MANUALES DEL SISTEMA		295
9.1	MANUAL DE INSTALACIÓN	295
9.1.1	<i>Instalación de PostgreSQL</i>	295
9.1.2	<i>Instalación de visual SVN Server</i>	304
9.1.3	<i>Instalación de JBoss AS7</i>	306
9.1.4	<i>Instalación de las aplicaciones</i>	311
9.2	MANUAL DE USUARIO	312
9.2.1	<i>Registrarse en la Aplicación</i>	312
9.2.2	<i>Rasgos Generales</i>	313
9.2.3	<i>Perfil del Usuario</i>	313
9.2.4	<i>Chat de la Aplicación</i>	314
9.2.5	<i>Consultar Notificaciones</i>	315

9.2.6	<i>Consultar Mensajes</i>	315
9.2.7	<i>Contactos</i>	317
9.2.8	<i>Reuniones</i>	318
9.2.9	<i>Proyectos</i>	320
9.2.10	<i>Proyecto</i>	321
CAPÍTULO 10. CONCLUSIONES Y AMPLIACIONES		327
10.1	CONCLUSIONES.....	327
10.2	AMPLIACIONES	328
10.2.1	<i>Control de Versiones de los Recursos</i>	328
10.2.2	<i>Editor de Documentos Online</i>	328
10.2.3	<i>Applet para Subir Carpetas</i>	328
10.2.4	<i>Exportar Recursos a Google Docs.</i>	328
10.2.5	<i>Diagrama de Gant</i>	329
10.2.6	<i>Relacionar Recursos con Tareas</i>	329
10.2.7	<i>Histórico de Acciones de Usuario</i>	329
CAPÍTULO 11. PRESUPUESTO		331
CAPÍTULO 12. REFERENCIAS BIBLIOGRÁFICAS.....		333
12.1	LIBROS Y ARTÍCULOS	333
12.2	REFERENCIAS EN INTERNET.....	334
CAPÍTULO 13. APÉNDICES		335
13.1	GLOSARIO Y DICCIONARIO DE DATOS	335
13.2	CONTENIDO ENTREGADO EN EL CD-ROM	337
13.2.1	<i>Contenidos</i>	337
13.2.2	<i>Estructura de Directorios de "cnpd"</i>	338
13.2.3	<i>Código Ejecutable e Instalación</i>	338
13.2.4	<i>Ficheros de Configuración</i>	339
13.3	ÍNDICE ALFABÉTICO	341
13.4	CÓDIGO FUENTE	343
13.4.1	<i>Ejemplos Capa de Persistencia</i>	343
13.4.2	<i>Ejemplos Entidades</i>	346
13.4.3	<i>Ejemplos Capa de Negocio</i>	352
13.4.4	<i>Ejemplos Capa de Presentación</i>	357
13.4.5	<i>Ejemplos Cliente de Escritorio</i>	363
13.4.6	<i>Ejemplos Patrones de Diseño</i>	367
13.4.7	<i>Ejemplos Interceptores</i>	371

Índice de Figuras

Figura 1 : Arquitectura de JEE 6	32
Figura 2: Ciclo de Vida de un Bean de Sesión sin Estado.....	34
Figura 3: Ciclo de Vida de un Servlet	36
Figura 4 Configuración con Spring Security.....	39
Figura 5: Arquitectura Spring Security	40
Figura 6: Patón MVP	42
Figura 7: Ejemplo de GXT	43
Figura 8: Ejemplo de Swingx.....	44
Figura 9 Ejemplo Smack API	46
Figura 10: Apache Subversion	47
Figura 11: SVN KIT	48
Figura 12: Tareas del proyecto	52
Figura 13: Diagrama de Gant.....	53
Figura 14 : Actores del sistema.....	74
Figura 15: Casos de uso primarios	76
Figura 16: Caso de uso Gestionar Usuarios.....	77
Figura 17: Caso de uso Gestionar Contactos.....	81
Figura 18 : Gestionar Proyectos.....	84
Figura 19: Cas de uso Gestionar Recursos	89
Figura 20: Añadir usuario con éxito	94
Figura 21: Añadir usuario sin éxito cuenta de Google invalida	95
Figura 22: Añadir usuario sin éxito cuenta de Google repetida	96
Figura 23: Modificar usuario con éxito	97
Figura 24: Modificar usuario sin éxito cuenta de Google Invalida.....	98
Figura 25 : Modificar usuario sin éxito cuenta de Google repetida	99
Figura 26: Eliminar usuario con éxito	100
Figura 27: Eliminar usuario con éxito – Administrador	100
Figura 28: Buscar usuario con éxito con buscador	101
Figura 29: Buscar usuario con éxito manualmente	102
Figura 30: Añadir usuario como contacto con éxito.....	103
Figura 31: Añadir usuario como contacto sin éxito	104
Figura 32: Ver perfil de contacto	105
Figura 33: Comunicarse con contacto con éxito en chat general.	106
Figura 34: Comunicarse con éxito en chat de proyecto	107
Figura 35: Comunicarse por mail con éxito.....	108
Figura 36: Comunicarse con contacto por mensajes	108
Figura 37: Eliminar contacto con éxito	109
Figura 38: Crear proyecto con éxito.....	110
Figura 39: Incluir contactos con éxito	111
Figura 40: Incluir contactos sin éxito	112
Figura 41: Excluir Contacts con éxito	113
Figura 42: Cambiar gestor con éxito	114
Figura 43: Publicar proyecto con éxito	115
Figura 44: Eliminar proyecto con éxito	116

Figura 45: Crear discusión con éxito.....	117
Figura 46: Votar en discusión con éxito	118
Figura 47: Abandonar proyecto con éxito	119
Figura 48: Crear nuevo recurso con éxito.....	120
Figura 49: Importar desde local con éxito	121
Figura 50: Importar desde Google Docs con éxito	122
Figura 51: Editar recurso online con éxito	124
Figura 52: Renombrar recurso con éxito	125
Figura 53: Mover recurso con éxito	125
Figura 54: Descargar recurso con éxito	126
Figura 55: Subir recurso con éxito.....	127
Figura 56: Eliminar recurso con éxito	128
Figura 57: Diagrama de clases del Análisis	129
Figura 58: Sección principal - Noticias.....	137
Figura 59: Sección principal - Mensajes	138
Figura 60: Sección principal- contactos	139
Figura 61: Sección principal - proyectos	139
Figura 62: Sección de Contacto	140
Figura 63: Sección del Proyecto - inicio	141
Figura 64: Sección del Proyecto - árbol	141
Figura 65: Sección del Proyecto – recursos libres.....	142
Figura 66: Sección del proyecto – referencias	143
Figura 67: Sección del proyecto - planificación	143
Figura 68: Sección de Proyecto - discusiones	144
Figura 69: Sección del Proyecto - discusiones	145
Figura 70: Diagrama de navegabilidad	146
Figura 71: Patrón N Tiers – Dos niveles	152
Figura 72: Patrón N Tiers – Tres niveles	152
Figura 73: Patrón N Tiers – Cuatro niveles	152
Figura 74: Patrón Layers	153
Figura 75: Diagrama de componentes internos	154
Figura 76: Diagrama de componentes	156
Figura 77: Especificaciones de los componentes del sistema	157
Figura 78: Diagrama Despliegue de Alto Nivel	161
Figura 79: Diagrama de despliegue de Bajo Nivel	163
Figura 80: Diagrama de paquetes global	164
Figura 81: Diagrama paquetes business	166
Figura 82: Diagrama de paquetes web	167
Figura 83: Diagrama de paquetes client	168
Figura 84: Patrón DAO	169
Figura 85: Patrón Template Method	170
Figura 86: Patrón Singleton.....	170
Figura 87: Patrón Facade	171
Figura 88: Patrón Session Facade.....	172
Figura 89: Patrón Adapter	172
Figura 90: Diagrama de clases- Paquete model.....	175
Figura 91: Diagrama de clases- Paquete persistence.....	176
Figura 92: Diagrama de clases – Paquete prsistence.impl	177
Figura 93: Diagrama de clases - persistence.exception	178

Figura 94: Diagrama de clases - Paquete business	179
Figura 95: Diagrama de clases – Paquete business.exception.....	181
Figura 96: Diagrama de clases – UsersManager.....	182
Figura 97: Diagrama de clases – ContactsManager.....	183
Figura 98: Diagrama de clases – ProjectsManager.....	184
Figura 99: Diagrama de clases - ResourcesManager	185
Figura 100: Diagrama de clases – Paquete shared.model	186
Figura 101: Diagrama de clases – Paquete shared.exception	187
Figura 102: Diagrama de clases – Paquete shared.qualifiers	188
Figura 103: Diagrama de clases- Paquete shared.remote	189
Figura 104: Diagrama de clases – Paquete server.remote	190
Figura 105: Diagrama de clases server.mapper	191
Figura 106: Diagrama de clases server.smack.....	192
Figura 107: Diagramas de clases – Paquete server.security	193
Figura 108: Diagrama de clases – Paquete server.util.....	194
Figura 109: Diagrama de clases – Paquete client	195
Figura 110: Diagrama de clases – Paquete client.event	196
Figura 111: Diagrama de clases – MainPresenter	198
Figura 112: Diagrama de clases - ConversationsPresenter.....	199
Figura 113: Diagrama de clases - InfoPresenter.....	200
Figura 114: Diagrama de clases – NotificationsPresenter	201
Figura 115: Diagrama de clases – MessagesPresenter	202
Figura 116: Diagrama de clases – ContactsPresenter.....	203
Figura 117: Diagrama de clases – MeetingsPresenter.....	204
Figura 118: Diagrama de clases – ProjectsPresenter.....	205
Figura 119: Diagrama de clases – ProjectPresenter	207
Figura 120: Diagrama de clases – ManagementPresenter	208
Figura 121: Diagrama de clases – ResourcesPresenter	210
Figura 122: Diagrama de clases – ReferencesPresenter.....	211
Figura 123: Diagrama de clases – DiscussionsPresenter	212
Figura 124: Diagrama de clases – MilestonesPresenter	213
Figura 125: Diagrama de clases – TasksPresenter.....	214
Figura 126: Diagrama de clases - Paquete client.....	216
Figura 127: Diagrama de secuencia – Eliminar usuario.....	217
Figura 128: Diagrama de Secuencia – Añadir Petición	218
Figura 129: Diagrama de Secuencia – Añadir Contacto.....	219
Figura 130: Diagrama de secuencia – Eliminar Proyecto.....	220
Figura 131: Diagrama de Secuencia – Crear Recurso	221
Figura 132: Diagrama de secuencia – Autenticación.....	222
Figura 133: Diagrama de secuencia – Envío de evento GWT para cambiar el historial	223
Figura 134: Diagrama de secuencia - Cambio de valor del historial	224
Figura 135: Diagrama de Actividad - Añadir Contacto	225
Figura 136: Diagrama de Actividad – Crear Recurso en Presentación	226
Figura 137: Diagrama de Actividad Crear Recurso en Negocio.....	227
Figura 138: Diagrama de estados - Ciclo de vida de un proyecto	228
Figura 139: Diagrama de estados – Ciclo de vida de una tarea	229
Figura 140: Diagrama de estado – Ciclo de vida de una discusión	230
Figura 141: Diagrama de estados – ciclo de vida de una reunión.....	231
Figura 142: Diseño de Sección Principal	232

Figura 143: Diseño de Sección de Información.....	233
Figura 144: Diseño de Sección de Información – Estudios y Trabajos	233
Figura 145: Diseño de la sección de Información – Gestión del perfil.....	233
Figura 146: Diseño de Sección de Información – Modificación de Datos	234
Figura 147: Diseño de Sección de Información – Modificación de Trabajos	234
Figura 148: Diseño de Sección de Chat.....	235
Figura 149: Diseño de Sección de Conversaciones	236
Figura 150: Diseño de Sección de Notificaciones	236
Figura 151: Diseño de Sección de Notificaciones – Filtros	237
Figura 152: Diseño de Sección de Notificaciones - Movimientos.....	237
Figura 153: Diseño de Sección de Notificaciones – Detalles	237
Figura 154: Diseño de Sección de Mensajes	238
Figura 155: Diseño de Sección de Mensajes - Responder mensaje	238
Figura 156: Diseño de la sección de Contactos.....	238
Figura 157: Diseño Sección de Contactos – Listado de Contactos.....	239
Figura 158: Diseño Sección de Contactos – Búsqueda de Contactos	239
Figura 159: Diseño de Sección de Reuniones	239
Figura 160: Diseño Sección de Reuniones – Listado de Reuniones	240
Figura 161: Diseño de Sección de Proyectos	240
Figura 162: Diseño de Sección de Proyectos – Listado de Proyectos	240
Figura 163: Diseño de Sección de Proyectos – Búsqueda de Proyectos	241
Figura 164: Diseño Sección de Proyectos – Creación de Proyecto	241
Figura 165: Diseño Sección de Proyecto.....	242
Figura 166: Diseño Sección de Proyecto - Administración	242
Figura 167: Diseño de Sección de Recursos	243
Figura 168: Diseño Sección de Referencias	243
Figura 169: Diseño Sección de Referencias - Listado	244
Figura 170: Diseño de Sección de Referencias – Añadir Referencias	244
Figura 171: Diseño Sección de Discusiones	244
Figura 172: Diseño Sección de Discusiones – Listado	245
Figura 173: Diseño Sección de Discusiones – Nueva Discusión.....	245
Figura 174: Diseño Sección de Hitos	246
Figura 175: Diseño Sección de Tareas	246
Figura 176: Diseño Sección de Tareas – Listado de Tareas	246
Figura 177: Diseño Sección de Tareas – Nueva Tarea	247
Figura 178: Diseño Sección de Tareas – Resumen de Tareas	247
Figura 179: Solución Problema Plugin GWT	267
Figura 180: Ejemplo Javadoc.....	270
Figura 181: Desarrollo Pruebas Unitarias – Persistencia	272
Figura 182: Desarrollo Pruebas Unitarias – Persistencia II	273
Figura 183: Desarrollo de Pruebas Unitarias – Capa de Negocio I	274
Figura 184: Desarrollo de Pruebas Unitarias – Capa de Negocio II	275
Figura 185: Desarrollo de Pruebas Unitarias – Capa de Negocio II	276
Figura 186: Desarrollo de Pruebas Unitarias – Capa de Negocio II	276
Figura 187: Desarrollo de Pruebas Unitarias – Capa de Negocio III	277
Figura 188: Desarrollo de Pruebas Unitarias – Capa de Negocio IV	278
Figura 189: Desarrollo Pruebas Capa de Presentación I	279
Figura 190: Desarrollo Pruebas Capa de Presentación II	279
Figura 191: Instalación PostgreSQL.....	295

Figura 192: Instalación PostgreSQL – Directorio de instalación	296
Figura 193 : Instalación PostgreSQL – Directorio de datos.....	296
Figura 194: Instalación de PosgreSQL – Contraseña	297
Figura 195: Instalación de PostgreSQL- Puerto	297
Figura 196: Instalación de PostgreSQL -Configuración.....	298
Figura 197: Instalación de PostgreSQL - Confirmación.....	298
Figura 198: Instalación de PostgreSQL - Proceso	299
Figura 199: Instalación de PosgresSQL - Fin	299
Figura 200: Configuración de PostgreSQL - PgAdmin3	300
Figura 201: Configuración de PostgreSQL – Contraseña	300
Figura 202: Configuración de PostgreSQL – PgAdmin iniciado	301
Figura 203: Configuración de PostgreSQl – Usuario y contraseña	302
Figura 204: Configuración de PosgreSQL – Creación de la base de datos	302
Figura 205: Configuración de PostgreSQL – Conexiones remotas 1	303
Figura 206: Instalación de Visual SVN Server.....	304
Figura 207: Instalación Visual SVN Server – Tipo de instalación	304
Figura 208: Instalación de Visual SVN Server – Configuración	305
Figura 209: Instalación de Visual SVN Server – Fin	305
Figura 210: Instalación AS7 – Interfaces de Red	306
Figura 211: Instalación AS7 – Driver PostgreSQL	307
Figura 212: Instalación AS7 – Datasource	307
Figura 213: Instalación AS7 – Java Mail	308
Figura 214: Instalación AS7 – Conexión Gmail	308
Figura 215: Instalación AS7 – Subsistema remoting	309
Figura 216: Instalación AS7 – Conexión con negocio	309
Figura 217: Instalación AS7 – Seguridad Conexión con Negocio	310
Figura 218: Aplicación de gestión de JBossAS7	311
Figura 219: Manual de Usuario- Pantalla Inicial	312
Figura 220: Manual de Usuario – Registrarse en la Aplicación I.....	312
Figura 221: Manual de Usuario – Rasgos Generales	313
Figura 222: Manual de Usuario - Perfil del Usuario	314
Figura 223: Manual de Usuario – Chat de la aplicación.....	314
Figura 224: Manual de Usuario – Consultar Notificaciones.....	315
Figura 225: Manual de Usuario – Consultar Mensajes	316
Figura 226: Consultar Mensajes – Responder Mensajes	316
Figura 227: Manual de Usuario –Contactos	317
Figura 228: Manual de Usuario – Consultar Contactos	317
Figura 229: Manual de Usuario – Consultar Peticiones.....	318
Figura 230: Manual de Usuario – Buscar Usuarios	318
Figura 231: Manual de Usuario – Consultar Reuniones	319
Figura 232: Manual de Usuario – Detalles de Reunión	319
Figura 233: Manual de Usuario – Crear Nueva Reunión	320
Figura 234: Manual de Usuario – Crear Nueva Reunión 2.....	320
Figura 235: Manual de Usuario – Consulta de Proyectos	321
Figura 236: Manual de Usuario Crear Proyecto	321
Figura 237: Manual de Usuario - Recursos.....	322
Figura 238: Manual de Usuario - Referencias	323
Figura 239: Manual de Usuario – Hitos.....	324
Figura 240: Manual de Usuario - Tareas	324

Figura 241: Manual de Usuario – Tareas I	325
Figura 242: Manual de Usuario – Resumen de Tareas.....	325

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Cuando la consecución de unos objetivos determinados se hace más y más difícil, las personas tenemos el instinto de juntar esfuerzos para obtener mejores resultados, pero a menudo suele pasar que estos grupos, se encuentran con que al juntar los esfuerzos no se obtienen los resultados esperados.

El objetivo de este proyecto es ofrecer una herramienta a estos grupos de personas que centralice todos los recursos que muchas veces inconscientemente usan para paliar los problemas de trabajar en equipo.

Es importante destacar que no se trata de una herramienta profesional de gestión de proyectos, nada más lejos, se trata de una utilidad que centraliza funcionalidades cotidianas usadas para desarrollar tareas entre varias personas, funcionalidades tales como mensajería instantánea, correos electrónicos o alojamientos de ficheros compartidos, todo esto maquillado para guiar a un usuario sin un concepto de proyecto tan fuerte como el que pueda tener un ingeniero, durante el desarrollo de un esfuerzo temporal para alcanzar un objetivo, algo menos que la definición de proyecto.

Todo esto colocado en el marco de un proyecto final de carrera, que no es capaz de dar una solución empresarial a este problema, da como resultado un proyecto que tiene como objetivos dar una solución que aunque no alcance los elevados niveles de calidad requeridos para sacar al mercado un sistema como el planteado, sirva como instrumento académico para completar mi perfil profesional reafirmando y adquiriendo conocimientos, tanto tecnológicos como teóricos, permita realizar un ejercicio de ingeniería mediante el diseño de una solución e ilustre que la solución propuesta aunque carente de calidad técnica, resuelve el problema propuesto y podría en un contexto empresarial solucionar sus motivaciones en un entorno real.

1.2 Resumen de Todos los Aspectos

En primer lugar se realizará un resumen general de todos los aspectos destacados del proyecto, en el cual se mencionará, a grandes rasgos y sin profundizar demasiado en aspectos técnicos, las tecnologías a utilizar para el desarrollo del Proyecto la arquitectura global del sistema y las características de la aplicación que vera el usuario final.

En segundo lugar se realizará un resumen de cada uno de los aspectos más importantes del proyecto: Análisis, Diseño e Implementación.

1.2.1 Resumen General

Se trata de un sistema distribuido, basado en el patrón de despliegue N-Tiers, con una arquitectura física abierta desde un mínimo de 3 servidores, reales o virtuales, que alberguen por separado el nivel de datos, el nivel de negocio y en la cima la aplicación web.

El sistema está desarrollado con tecnología JEE 6, esto es usando las especificaciones de esta plataforma. Las especificaciones que más marcan el sistema son EJB 3.1, CDI 1.0 y JPA 2.0.

En otro orden cabe resaltar que la aplicación web se sale del marco de la plataforma y esta implementada con tecnología GWT (Google Web Toolkit), con algunos componentes más, entre ellos un proyecto de JBoss Community que permite llevar parte del JSR 299 (CDI) al lado del cliente e integrarlo con GWT.

Para la gestión de ficheros el sistema se apoya en un servidor de control de versiones Apache Subversion mientras que para la mensajería instantánea usa el servidor de Google Talk.

Funcionalmente el sistema ofrecerá una aplicación web Enriquecida, con un aspecto similar a una aplicación de escritorio, mediante la cual el usuario podrá crear proyectos, gestionarlos, y mantener una red de contactos que le permita comunicarse y colaborar con ellos.

La aplicación se estructura en una sección principal donde el usuario verá sus mensajes, sus contactos y sus proyectos, además podrá hablar con los contactos conectados al sistema.

Desde la sección de contactos podrá ver los proyectos y contactos con los que está relacionado cada uno de sus contactos, además de enviarles mensajes. Desde la sección de mensajes podrá leer y contestar los mensajes que le envíen sus contactos. Y finalmente podrá acceder a los proyectos que haya creado así como por supuesto crear y aceptar invitaciones a proyectos.

En cada proyecto el usuario tendrá opciones relacionadas con la gestión de los recursos del proyecto, que para el perfil de usuario se espera que sean mayormente ficheros y documentos. Además de gestionar recursos desde un proyecto se podrá discutir y planificar aspectos del proyecto.

1.2.2 Análisis

En este apartado se especifica de manera detallada las características y funcionalidades que debe satisfacer el sistema que se desarrollara, además se desarrollan una serie de modelos para entender la funcionalidad a desarrollar y como se espera que interactúe el usuario final durante el disfrute de dichas funcionalidades. Se plasma un modelo del sistema en términos de orientación a objetos de alto nivel, o análisis orientado objetos. Finalmente se realiza un esbozo de la interfaz de usuario que se le pueda mostrar a un usuario final aunque no sea definitivo. Todo esto se toma como punto de partida para comenzar el diseño.

1.2.3 Diseño

El diseño recoge la definición de una solución técnica con todos los detalles que debería necesitar un desarrollador para implementar el sistema. Es la continuación natural del análisis por lo que los modelos expuestos en el son un reflejo aumentado de los expuestos en el análisis.

Se comienza por una descripción de la arquitectura del sistema, para lo que se usan diagramas de componentes, paquetes y despliegue, además de explicar los patrones arquitectónicos aplicados, se incluyen modelos de la lógica del sistema, basados principalmente en diagramas de secuencia y de estados, se expone el diseño de clases mediante diagramas de clases y una previa explicación de los patrones de diseño aplicados y finalmente se expone el proceso de diseño de la interfaz de usuario y se especifican técnicamente las pruebas a realizar.

1.2.4 Implementación

En la implementación se describen todas las particularidades del desarrollo del sistema. Se exponen los convenios seguidos durante el desarrollo del sistema y que deberán ser tenidos en cuenta por futuros e hipotéticos desarrolladores. Se expone una breve explicación de cada una de las herramientas con las que ha sido construido el sistema y finalmente se exponen algunos detalles del proceso de creación, tales como problemas encontrados y algunas métricas.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Existe un gran número de personas que se enfrentan a diario al reto de desarrollar actividades de forma conjunta, sin unos conocimientos ni unas herramientas que le faciliten la solución de los problemas que se les presentan durante esta labor. Aunque este perfil no trabajé habitualmente en proyectos de gran envergadura, si no en prácticas o trabajos, se enfrentan a los mismos problemas que presenta un proyecto de ingeniería, a su nivel.

Este proyecto pretende acercar al perfil mencionado nociones propias de la gestión de proyectos, como son el propio concepto de proyecto, hito, tarea o recurso, de la mano de otros ya conocidos como es el de contacto.

En resumen el proyecto pretende ofrecer a los usuarios una red de contactos sobre la cual desarrollar sus proyectos, por muy banales que sean. No se pretende desarrollar una herramienta profesional de gestión de proyectos, de hecho lo que se intenta es ofrecer un número pequeño de las funcionalidades que estas ofrecen abstraídas para un perfil no profesional, como pueden ser estudiantes de cualquier titulación o incluso estudiantes de ingeniería que recién comienzan.

Por otro lado como estudiante, que también soy, en mi proyecto final de carrera veo una oportunidad genial para sentar los cimientos de mi perfil profesional y cartera tecnológica, por lo que he decidió seleccionar un conjunto de tecnologías que entiendo me pueden aportar valor, y aplicarlas al desarrollo de una solución a los problemas antes mencionados.

2.2 Objetivos del Proyecto

Los objetivos de este proyecto son netamente **académicos**, no existe un plan de negocio ni un cliente real. Partiendo de un problema que a mi modo de ver existe, y de unos clientes hipotéticos, se ha desarrollado un proyecto con los siguientes objetivos:

Diseñar mediante el uso de técnicas de Ingeniería del Software una solución a los problemas planteados, con el fin de reforzar las habilidades adquiridas en este campo durante la carrera.

- a. **Diseño de una solución:** Desarrollar software siguiendo unas instrucciones estrictas puede ser tan complejo, o tan simple como se quiera, pero en cualquier caso siempre será más complejo desarrollar también dichas instrucciones, teniendo en cuenta todo lo que ellas puedan implicar y que al mismo tiempo cumplan algo primordial en una solución , *ser una solución*.

Uno de los objetivos de este proyecto es someterme al esfuerzo de pensar una solución a un problema concreto y transformarla en un conjunto de instrucciones, también llamadas requisitos.

- b. **Aplicación de técnicas de Ingeniería del Software:** Otro de los objetivos del proyecto es que el desarrollo de la solución se haga mediante el uso de técnicas de Ingeniería del software, tales como Análisis Orientado objetos o Diseño orientado a objetos.
2. Desarrollar una arquitectura y un diseño que además de soportar la solución diseñada, presenten un nivel de complejidad tal que permita reforzar las habilidades adquiridas durante la carrera en estos campos, particularmente en lo que a arquitecturas distribuidas respecta.
 - a. **Arquitectura:** La arquitectura obviamente tiene que ser válida para la solución, no obstante además de valida se plantea como un objetivo, el desarrollo de una arquitectura distribuida, que involucre a varios elementos de diversas índoles.
 - b. **Diseño:** Al igual que la arquitectura, el diseño tiene que permitir la consecución de los objetivos, no obstante y en línea con los requisitos académicos que se vienen describiendo, el diseño del software debe mantener unos niveles de calidad altos, aplicando patrones de diseño clásicos y modernos a lo largo de toda la arquitectura y para fines tan diversos como la construcción de interfaces de usuario, o el acceso a sistemas de base de datos.
3. Aplicar al desarrollo de la solución diseñada un conjunto de tecnologías con una complejidad y un nivel de aceptación en el mercado alto.
 - a. **Uso de JEE 6 Full profile:** Uno de los objetivos tecnológicos del proyecto es el aprendizaje de la plataforma Java EE 6, concretamente se pretende ganar

experiencia en uso de las especificaciones JPA 2.0 (JSR 317) , CDI 1.0 (JSR 299) y EJB 3.1 (JSR 318). Es un objetivo de este proyecto también llegar a ser capaz de relacionar las especificaciones con los componentes subyacentes que las implementan, cuando aplique.

- b. **Uso de alguna tecnología RIA :** Se pretende adquirir experiencia en el uso de alguna tecnología para el desarrollo de aplicaciones enriquecidas para internet, debido a la dura curva que estas presentan, se precisa mucho tiempo de experiencia para realizar un uso productivo de dichas tecnologías, y un proyecto final de carrera se presenta una oportunidad perfecta para adquirir dicha experiencia.

Capítulo 3. Aspectos Teóricos

En este apartado se puede encontrar toda la información que un ingeniero de software que no está al corriente de las tecnologías usadas para este proyecto podría querer consultar.

No se cubren por tanto aspectos tales como lenguajes de programación o librerías estándar de estos, esta enfocados en las tecnologías que pueden resultar novedosas para un ingeniero, dado que el proyecto está construido sobre la plataforma JEE 6, será a esta a la que más esfuerzo dedicaré.

3.1 JEE 6

JEE es una especificación que define una plataforma sobre la que construir aplicaciones empresariales en Java, en esta sexta edición se introduce el concepto de perfil, que no es más que una configuración concreta de la plataforma enfocada a construir un determinado tipo de aplicaciones. Todos los perfiles comparten un conjunto de servicios en común, hasta la fecha se manejan dos perfiles JEE 6 full profile y JEE web profile, el segundo más reducido.

Los distintos servicios que nos ofrece la plataforma nos llegan en forma de especificaciones con sus respectivas implementaciones, luego cada perfil engloba una serie de especificaciones, así como lo recogido por la propia especificación JEE 6 (JSR 316), algunas de las especificaciones que incluye la plataforma son JSR-315 (Servlet 3.0) , JSR-317 (JPA 2.0), JSR-318 (EJB 3.1), JSR-319 (Interceptors 1.1), JSR-907 (JTA 1.1) , JSR-299 (CDI 1.0), JSR-311 (JAX-RS 1.1) , JSR-303 (Bean Validation), JSR-245 (JSP 2.2) o JSR 314 (JSF 2.0).

Otro concepto primordial en la plataforma desde su incepción es el de contenedor, un contenedor es una o varias implementaciones de una o varias APIs de JEE, de forma que nuestra aplicación sea desarrollada para encajar en el seno de dicho contenedor, la aplicación solo deberá conocer las APIs, cuyas implementaciones le serán provistas por el contenedor, consiguiendo así que el desarrollador se despreocupe de esto. En la Figura 1 se puede ver la arquitectura de la plataforma.

En este proyecto se apuesta por una implementación full profile de la plataforma, de la mano de JBoss Community y su fantástico AS7, salvando la capa de presentación en la que se excluye el uso de la especificación Servlet 3, apostando por tecnologías RIAs de terceros que no son compatibles a día de hoy con el JSR 315.

En los siguientes apartados abordaré algunas de las especificaciones que me parecen más impactantes en el resultado final de este proyecto.

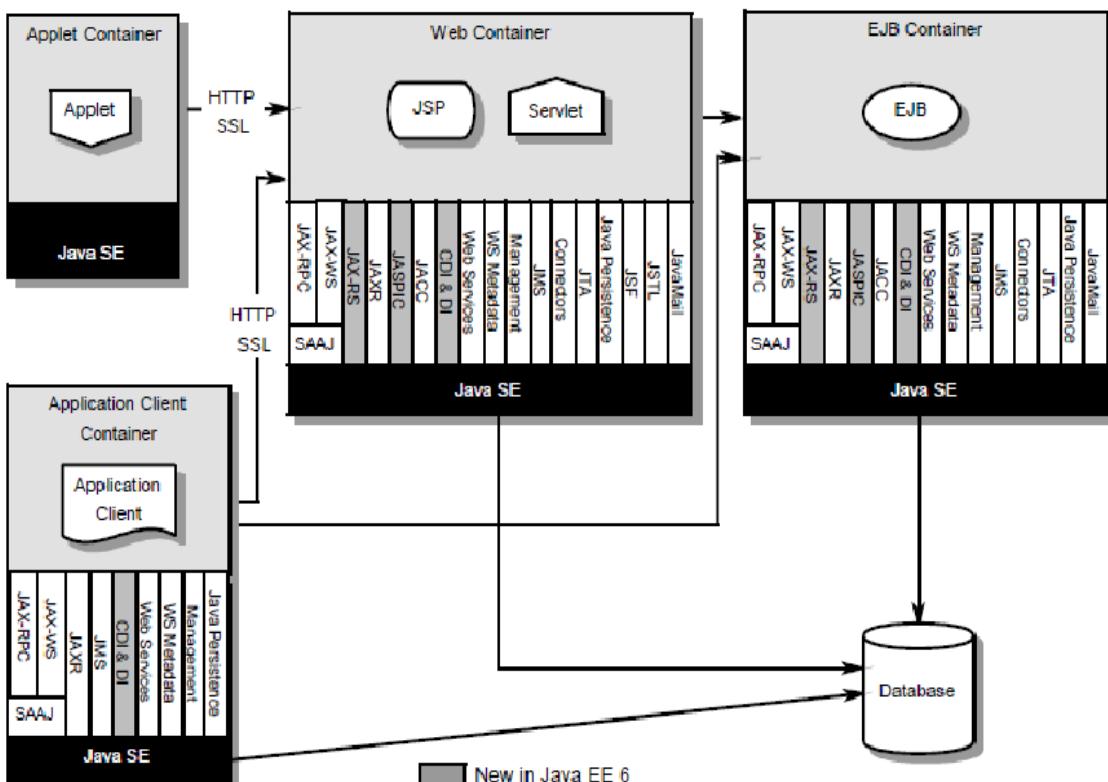


Figura 1 : Arquitectura de JEE 6

3.1.1 EJB 3.1

EJB(Enterprise java beans) es una especificación que define una arquitectura de objetos de servidor que permite al desarrollador de estos delegar aspectos tales como la gestión de las transacciones, la concurrencia o el acceso remoto en un contenedor, conocido en argot como el contenedor de EJBs o EJB container.

La especificación define siete roles, que vienen a definir quien hace que, los roles definidos son:

- Proveedor de beans (Enterprise Beans Provider): El encargado de proporcionar los objetos, estos objetos están pensados para ejecutar lógica de negocio, y pueden ser de sesión, con estado, instancias únicas (singletons), o temporizadores.
- Ensamblador de aplicaciones (Application Assembler): Este es el que se dedica a ensamblar los objetos desarrollados para convertirlos en aplicaciones. Suele ser un experto de dominio.
- Lanzador de aplicaciones (Deployer): Se encarga de desplegar las productos de los roles anteriores en un contenedor, así como de resolver las dependencias de estos.
- Proveedor del Servidor y proveedor del contenedor: Actualmente la especificación asume que ambos roles son desarrollados por el mismo proveedor. Suelen ser proveedores de sistemas operativos y middleware. El proveedor del servidor se refiere

al rol que provee transacciones y objetos distribuidos mientras que el de proveedor del contenedor se refiere al desarrollador de un software que integrado en el servidor, ofrece transacciones, seguridad y acceso remoto a nuestros objetos. La especificación no habla de la interfaz entre el contenedor y el servidor, pero cabe resaltar que no son lo mismo.

- Proveedor de persistencia: Este rol define al proveedor de un sistema de mapeo objeto relacional. Puede ser el proveedor del contenedor de EJBs u otro.
- Administrador del sistema: El administrador tanto del servidor como del contenedor, es quien debe configurar estos y su entorno.

La especificación define tres tipos de Enterprise Java Beans, beans de sesión, beans dirigidos por mensajes y entidades, a su vez define tres tipos de beans de sesión, con estado, sin estado y singlettons.

En este proyecto se hará uso de beans de sesión sin estado, ya que son menos costosos y ofrecen transaccionalidad, acceso remoto y control de concurrencia necesarios.

Cabe destacar que los EJBs son acorde a la especificación Managed Beans 1.0, Managed Beans, esto es que pueden disfrutar de todos los servicios que dicha especificación garantiza, tales como inyección de recursos, de otros beans o interceptores.

La especificación describe en detalle el contrato entre los beans y el contenedor, dicho contrato define entre otras cosas el ciclo de vida de los beans, la Figura 2 muestra el ciclo de vida de un bean de sesión sin estado:

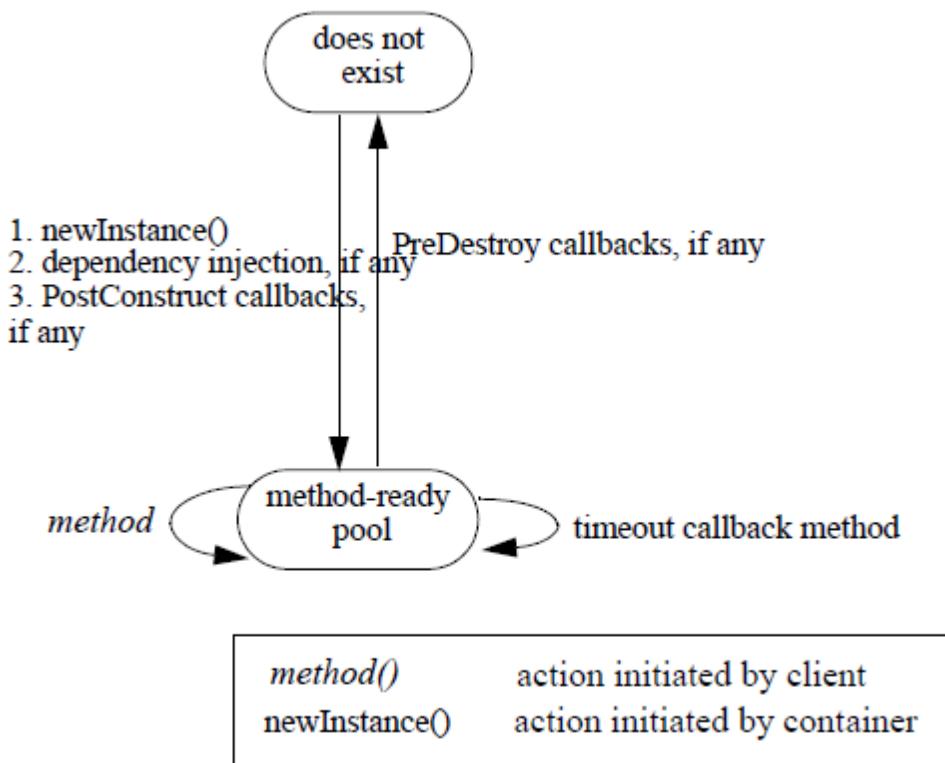


Figura 2: Ciclo de Vida de un Bean de Sesión sin Estado

A la hora de valorar el rendimiento hay que bajar al sitio del administrador del sistema, y lidiar con el contenedor, en donde deberíamos poder configurar los mecanismos de instanciación de los objetos (pool, número de instancias del pool) así como los tiempos de respuesta, un comportamiento típico de los contenedores es la creación de un conjunto de objetos que atienden peticiones, con tiempos de respuesta del orden de minutos.

La implementación de la especificación elegida ha sido JBoss EJB3, proyecto open Source de JBoss Community, que viene integrado en su servidor de aplicaciones.

3.1.2 JPA 2.0

Especificación que define la gestión de la persistencia y el mapeo objeto relacional tanto para Java EE como para java SE. Esta especificación se construye entorno al concepto de entidad, que no es más que una clase java anotada con un conjunto de anotaciones definidas por la plataforma, y que cumple algunas restricciones. Conceptualmente una entidad se podría entender como una versión evolucionada del concepto de entidad del modelo entidad relación clásico. El proveedor de mapeo objeto relacional conocido como ORM se encarga de mapear las entidades de nuestro modelo con sus relaciones al modelo relacional de tablas.

Resumiendo un poco, la especificación define como mapear todos los atributos de la entidad a campos de una relación, como mapear las claves primarias, relaciones muchos a uno, uno a

muchos, uno a uno y muchos a muchos, jerarquías de herencia y composiciones. Para ello define anotaciones que pueden ser sobreescritas en XML.

Un concepto muy importante es el de contexto de persistencia, un contexto de persistencia es un conjunto de instancias de entidades, de las cuales ninguna tiene el mismo valor en su identificador.

El encargado de tratar con el contexto de persistencia es el gestor de entidades o EntityManager. El entityManager está asociado a un contexto de persistencia y define operaciones sobre las entidades para ese contexto de persistencia.

El conjunto de entidades que abarca un entity manager se conoce como unidad de persistencia.

Es muy importante conocer el ciclo de vida de las entidades, que va muy relacionado con concepto de contexto de persistencia. Una entidad puede pasar por 4 fases nueva, gestionada, desactualizada y borrada (new, managed, detached, removed). Esto tiene una serie de implicaciones, por ejemplo las colecciones por defecto, solo pueden ser cargadas en entidades gestionadas. Una entidad gestionada va asociada a un contexto de persistencia y un contexto de persistencia, por defecto solo existe dentro de una transacción. Para salvar esta barrera se define la propagación de un contexto de persistencia, que se consigue estableciendo la propiedad persistenceContextType a EXTENDED, esto hará que el entity manager se mantenga vivo y abierto hasta que sea cerrado, incluso después de una transacción. Esto solo es posible con entityManager(s) gestionados por el contenedor, en estos casos el funcionamiento es que el contenedor abre el entity manager cuando comienza una UserTransaction (JTA) y lo cierra cuando esta acaba.

Para este proyecto la implementación de la especificación elegida ha sido Hibernate, framework ORM muy usado mundialmente que viene integrado desde hace varias versiones en el servidor de aplicaciones de JBoss.

3.1.3 CDI 1.0

Contextos e Inyección de dependencias, definida por la comunidad como la niña mimada de JEE6, es la nueva especificación para inyección de dependencias en java, haciendo uso de las anotaciones definidas por los JSR 333 y 250.

El concepto primordial de la especificación, es el concepto de instancia contextual, que es una instancia gestionada por el contenedor y mapeada a un contexto, otro concepto importante.

Todos los objetos que cumplan con la especificación *Managed Beans 1.0* pueden ser instancias contextuales, las instancias contextuales pueden ser inyectadas en puntos de inyección, otro concepto definido por la especificación. La especificación define como hacer métodos productores de instancias contextuales. Contempla una implementación integrada del patrón *Decorator de Eric Gamma* así como un mecanismo de eventos fantástico, que consiste en definir métodos observadores de eventos, estos métodos son ejecutados por el contenedor

cuando es invocado el método `fire()` del evento, método definido por la interfaz `javax.enterprise.Event`.

La implementación elegida de esta especificación fue Weld, incorporada en el servidor de aplicaciones de JBoss y con una excelente distribución para Java SE.

3.1.4 Servlet 3.0

Especificación que define el empaquetado y el despliegue de las aplicaciones web. Para ello especifica los requisitos que deben cumplir los contenedores web y define las interfaces entre el contendor y la aplicación web.

La especificación define una serie de responsabilidades para el contenedor web tales como la gestión de la sesión, la seguridad o la concurrencia.

Desde el punto de vista del desarrollador, la especificación ofrece tres interfaces para desarrollar aplicaciones, Servlet, Filter y Listener.

Los servlets son clases Java que pueden ser invocados mediante peticiones HTTP, los filtros son clases Java invocadas antes y después de cada petición del usuario mientras que los listeners son escuchadores de eventos, ejecutados por el contenedor cuando tienen lugar los eventos.

Todas estas clases tienen la característica particular y muy importante de que su ciclo de vida es controlado por el contendor, a modo de ejemplo se muestra el ciclo de vida de un Servlet en la Figura 3.

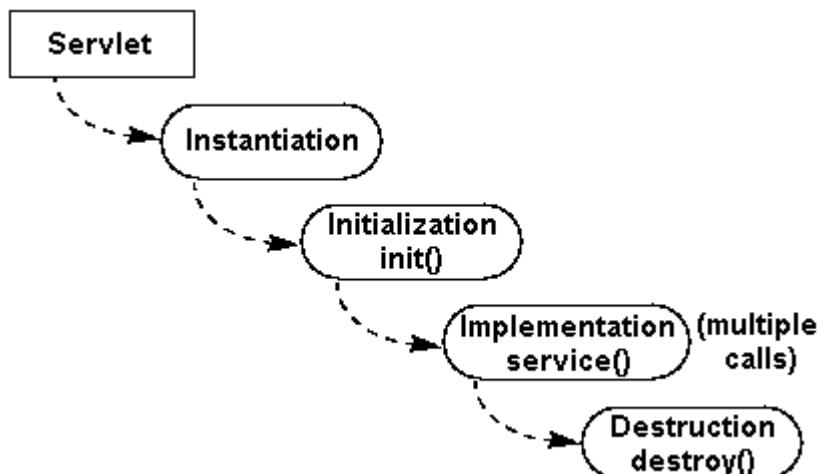


Figura 3: Ciclo de Vida de un Servlet

Además define interfaces para trabajar con la sesión, tener acceso a las peticiones y respuestas de los clientes y obtener acceso al contexto de la aplicación.

La plataforma JEE 6 comprende la versión 6 de la especificación aunque en este proyecto no se han usado ninguna de las virtudes añadidas a la especificación en esta versión.

3.1.5 JBoss AS 7

JEE 6 viene muy de la mano del concepto de *perfil*, que como ya he dicho es una configuración de especificaciones que estandariza a una implementación de la plataforma para un determinado tipo de aplicación. Pues desde *JBoss Community* han sabido reinventarse para encajar en este concepto y mejorar todas las pegas que tradicionalmente se le han puesto a su servidor de aplicaciones. El AS 7.1 ha sido certificado como una implementación de *JEE6 full profile* lo que lo convierte en candidato para este proyecto y sus grandes mejoras sin dudas lo hacen a mis ojos la mejor opción open source de JEE 6 full profile.

3.1.5.1 Administración y configuración

El *JBoss AS7* ofrece una mejoría notable en cuanto a gestión y configuración, cuenta con una aplicación web Enriquecida para la gestión, que ofrece opciones de monitorización y despliegue entre otras. Una estructura de directorios y de ficheros de configuración muy cómoda e intuitiva, dentro de lo que cabe esperar, con ficheros de configuración individuales en función de las funcionalidades. Además existen dos modos de funcionamiento, *standalone mode* y *domain mode*, con repercusión en la facilidad de gestión de las instancias del AS7.

3.1.5.2 Modos de funcionamiento

El modo *standalone* consiste en funcionar con una instancia del AS por proceso, como venía siendo habitual hasta ahora, mientras que el *domain* permite de desde un punto de control múltiples instancias del AS7.

Un dominio es un conjunto de instancias del AS7, estas instancias comparten una política de administración y un punto de control en común, este punto de control es conocido como *Domain Controller*, un dominio puede tener uno o más *Servers Groups*, estos instancias que son gestionadas y configuradas como una.

Es importante destacar que las ventajas del domain mode radican únicamente en la **gestión** de múltiples instancias, y no aportan ventajas en cuanto a funcionalidad aunque sí que puede hacer más cómodo gestionar configuraciones de alto rendimiento en este modo.

3.1.5.3 Carga de clases

Tradicionalmente uno de los aspectos más considerados por los implementadores de servidores de aplicaciones ha sido la carga de clases.

Típicamente la carga de clases se ha implementado jerárquicamente, esto es que un cargador de clases cuando intenta cargar una clase, se la pide al cargador padre, y el padre al padre, si el padre hasta que llega al de más alto nivel, en este punto el de más alto nivel la intenta resolver, si no puede el hijo la intenta, y así hasta abajo, de forma que una clase cargada por el cargador del nivel superior no tiene visibilidad de clases cargadas por el cargador del nivel inferior.

Este sistema puede producir y produce conflictos, para paliarlos y ganar en eficiencia se ha apostado por un sistema modular, basándose en el proyecto *JBoss modules*, en este nuevo modelo las clases se agrupan en módulos, los módulos definen dependencias sobre otros módulos. Los artefactos desplegados en el AS 7 también tratados como módulos.

La precedencia en este sistema de carga de clases es la siguiente, primero se busca en los módulos definidos por el servidor de aplicaciones, luego en busca dependencias definidas por el usuario en un descriptor de estructura de despliegue específico, clases empaquetadas en el despliegue y finalmente dependencias entre despliegues.

Las dependencias sobre la JRE son definidas también por el contenedor en un módulo.

3.1.5.4 Contenedor de Servicios Modular

Otra de las mejoras en el AS 7 viene de la mano del concepto de servicio y de que gran parte de estos servicios, son iniciados bajo demanda, gracias al proyecto *JBoss MSC* (Modular Service Container) se cargan los servicios cuando se despliega un módulo que define una dependencia sobre ese servicio, de esta forma se ahorra tiempo durante el inicio, el tiempo de arranque del AS7 sin despliegues es de 8 segundos por cerca de 1 minuto del AS5.

3.2 Spring Security

Framework basado en ACEGI Security y construido sobre Spring, permite configurar declarativamente todos los aspectos de seguridad tanto de autenticación como de autorización. La gran ventaja es que nos permite redefinir tanto los mecanismos de autenticación como los de autorización.

```
<security:http use-expressions="true" auto-config="false">

    <security:intercept-url pattern="/private/**"
        access="hasRole('ROLE_AUTHOR')"/>

    <security:form-login login-page="/LoadHome.xsp"
        default-target-url="/private/LoadProfile.xsp"
        authentication-failure-url="/LoadHome.xsp" login-processing-url="/login" />

    <security:logout invalidate-session="true"
        logout-success-url="/loadHome.xsp" logout-url="/logout" />
</security:http>
```

Figura 4 Configuración con Spring Security

La configuración de la seguridad puede llegar a ser tan simple como en la Figura 4, aunque en algunos casos se puede hacer necesario extenderla para obtener autenticación contra un sistema remoto más concreto o realizar autorización dinámica. El framework cuenta con facilidades para autenticar contra base de datos, CAS o LDAP.

La funcionalidad se consigue mediante una cadena de filtros que queda definida por el espacio de nombres, estos filtros van realizando diferentes tareas sobre los recursos que han sido mapeados.

La arquitectura del framework queda bien reflejada por la Figura 5.

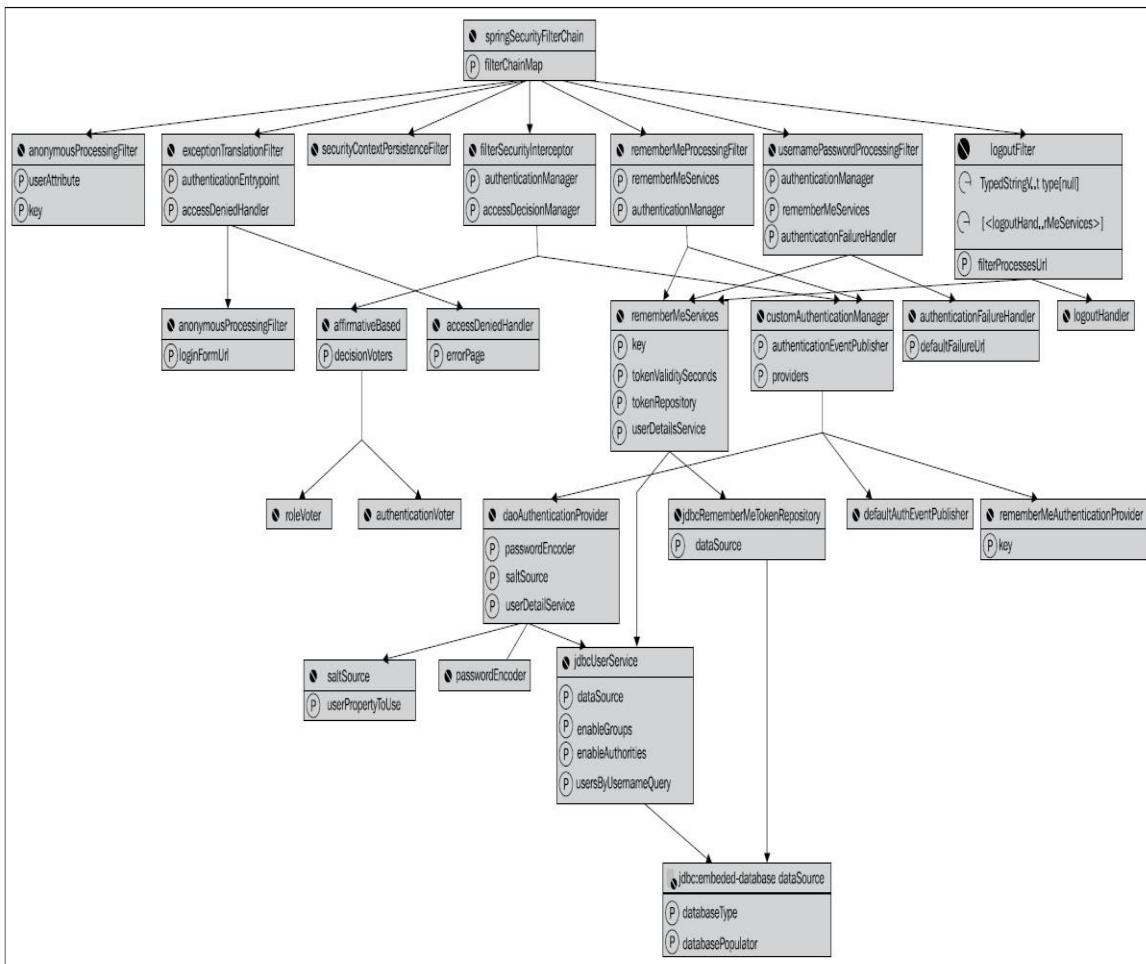


Figura 5: Arquitectura Spring Security

Además de configurar el fichero de contexto de Spring es necesario mapear en el web.xml junto al escuchador del contexto de Spring el filtro delegado de Spring, que Spring sustituirá por una implementación de `javax.servlet.Filter` que este definida como un `bean` con nombre igual al del filtro, como este bean es definido por el espacio de nombres, el nombre es fijo, este filtro se encargara de preparar toda la cadena de filtros.

3.3 Google Web Toolkit

Más conocido por sus siglas, GWT, esta tecnología es la apuesta de Google para el desarrollo de aplicaciones webs enriquecidas basadas en AJAX, se trata de un toolkit que nos permite desarrollar toda nuestra aplicación en java, aplicando todas las buenas prácticas y patrones de diseño del desarrollo de software tradicional, y dejando luego al toolkit la tarea de compilar nuestro código java a código JavaScript, este código JavaScript se ejecutará en el navegador del cliente y se comunicará vía http con otra parte de nuestra aplicación que residirá tal cual la desarrollemos, en el lado del servidor.

Para conseguir esto el toolkit nos ofrece un subconjunto de la biblioteca estándar de java y una serie de componentes visuales predefinidos como tablas, etiquetas o listas.

La comunicación entre cliente y servidor se realiza mediante HTTP, concretamente mediante la invocación de Servlets, para lo que GWT ofrece un API específico.

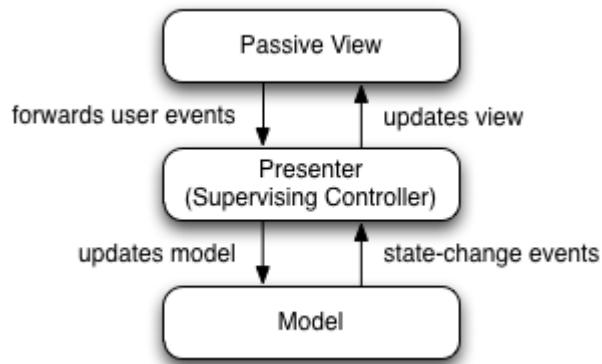
Destaca la posibilidad de definir las interfaces de usuario declarativamente (sin perder la posibilidad de hacerlas o modificarlas programáticamente) así como el manejo del historial del navegador.

El toolkit nos proporciona un plugin para el IDE Eclipse que nos permite además de un desarrollo muy cómodo, ejecutar y depurar nuestra aplicación sobre la máquina virtual sin necesidad de realizar el proceso de compilación, lo que redunda en menos tiempo de desarrollo.

El código JavaScript generado es compatible con múltiples navegadores, para ello GWT compila una permutación por cada par (navegador – Idioma) que sea definido.

3.3.1 Patrón MVP

La arquitectura propuesta por los ingenieros de Google para desarrollar aplicaciones de gran tamaño con GWT se basa en el patrón Modelo Vista Presentador, una variante del patrón Modelo Vista controlador adaptada a este tipo de aplicaciones.

*Figura 6: Patrón MVP*

Las vistas son Widgets de GWT o clases nuestras que agrupen o extiendan a estos, la arquitectura propone que dichas clases deleguen toda la lógica de control en el presentador que será el encargado de interactuar con el modelo y resolver las comunicaciones con el servidor cuando proceda.

3.3.2 Patrón Composite

Los diferentes widgets que provee GWT están desarrollados siguiendo el patrón composite, de forma análoga a como lo hace Swing, GWT define una estructura de composites tal que podamos desarrollar nuestras vistas e integrarlas en la jerarquía de forma que sean tratadas como un widget más.

3.3.3 EXT GWT

EXT GWT consiste en una framework de componentes gráficos (UI widgets) algo más amplio y más avanzados que los proporcionados por GWT, perfectamente integrables. Con EXT GWT tenemos tanto la posibilidad de desarrollar íntegramente la aplicación con sus widgets como de mezclarlos con los propios de GWT.

Entre los componentes más destacados están árboles, rejillas y una API de alto nivel para realizar dibujos y renderizar gráficos que facilita enormemente esta tarea. Además proporciona facilidades para arrastrar y soltar elementos entre componentes.

La muestra uno de los componentes, que puede verse junto al resto en una demo expuesta en la página de Sencha, empresa desarrolladora.



Figura 7: Ejemplo de GXT

3.3.4 Errai

Errai es un framework para desarrollar aplicaciones basadas en GWT que nos ofrece la posibilidad de usar inyección de dependencias en el lado del cliente acorde al JSR 333, además de traer al cliente también una implementación parcial del JSR 299.

Errai ofrece un modelo de programación orientado a mensajes para las comunicaciones entre las partes cliente y servidor de las aplicaciones GWT. Esto lo consigue gracias el uso de un bus de mensajes que no es más que un Servlet, que se encarga de enviar mensajes entre cliente y servidor usando el API provista por GWT y abstrandónos de ella.

Sobre este bus de mensajes Errai nos ofrece otras funcionalidades de más alto nivel como la inyección de dependencias tanto en el lado del servidor como en el lado del cliente, nos permite aprovechar el modelo de eventos propuesto por la especificación CDI, también en el cliente, significa que podremos escuchar eventos, cliente <-> cliente, servidor <-> servidor y servidor <-> cliente.

Errai cuenta con un mecanismo de marshalling que nos permite hacer portables nuestros objetos sin necesidad de hacerlos serializables y con un sistema más avanzado de invocación a métodos cliente servidor que el ofrecido por GWT.

En este proyecto nos hemos abstraído del Bus de mensajes para quedarnos con la inyección de dependencias, los eventos y las llamadas a métodos.

3.4 Swing

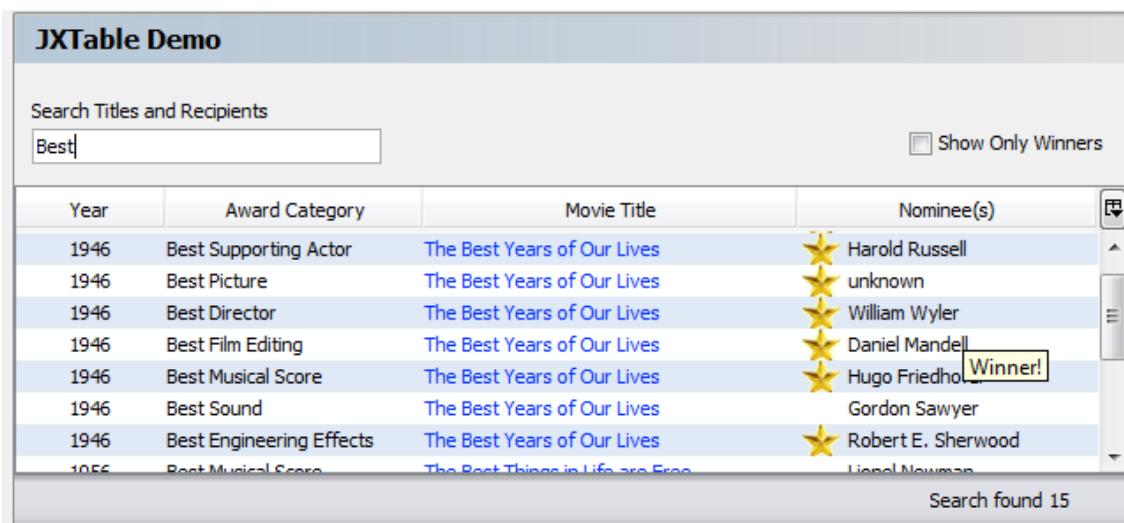
Swing es el principal toolkit para el desarrollo de interfaces gráficas en Java, es parte de la JFC (Java Foundation Classes) y sucede a AWT para ofrecer un conjunto de elementos visuales más avanzados. Es independiente del sistema operativo ya que no usa el API del Sistema para el dibujado de los componentes, a diferencia de otras plataformas como SWT de Eclipse.

En este proyecto se ha usado para la construcción de un cliente de escritorio muy sencillo que ilustre las virtudes de un sistema distribuido.

3.4.1 Swingx

En el marco del proyecto Swing Labs y construido sobre swing, Swingx es una biblioteca de componentes gráficos algo más elaborados que los provisto por Swing, paneles de login, paneles de tareas, rejillas u ordenación son solo algunas de las aportaciones.

La Figura 8 muestra un ejemplo de uno de los componentes provistos.



The screenshot shows a Java Swing application window titled "JXTable Demo". At the top, there is a search bar labeled "Search Titles and Recipients" containing the text "Best". To the right of the search bar is a checkbox labeled "Show Only Winners". Below the search bar is a table with the following columns: "Year", "Award Category", "Movie Title", and "Nominee(s)". The table contains the following data:

Year	Award Category	Movie Title	Nominee(s)
1946	Best Supporting Actor	The Best Years of Our Lives	Harold Russell
1946	Best Picture	The Best Years of Our Lives	unknown
1946	Best Director	The Best Years of Our Lives	William Wyler
1946	Best Film Editing	The Best Years of Our Lives	Daniel Mandell
1946	Best Musical Score	The Best Years of Our Lives	Hugo Friedho
1946	Best Sound	The Best Years of Our Lives	Gordon Sawyer
1946	Best Engineering Effects	The Best Years of Our Lives	Robert E. Sherwood
1955	Best Musical Score	The Best Things in Life are Free	Lionel Newman

At the bottom right of the table, it says "Search found 15".

Figura 8: Ejemplo de Swingx

3.4.2 Appframework

Es la implementación de referencia del JSR 296 Swing Application Framework, que consiste en un conjunto de funcionalidades reutilizables en el desarrollo de aplicaciones basadas en Swing.

Ofrece funcionalidades como la gestión de una sesión para la aplicación, la definición de acciones mediante anotaciones gracias a un contexto de aplicación o la gestión de recursos son algunas de las características que define la especificación.

3.5 Protocolo XMPP

Protocolo extensible de mensajería y presencia (extensible Messaging Presence Protocol), es un protocolo abierto basado en XML para mensajería instantánea.

Sin entrar a ver los distintos esquemas que recogen el protocolo y sus extensiones, algunos conceptos importantes son Roster, Mensaje, Presencia e IQ.

El protocolo define mensajes XML que deben ser enviados entre un cliente y un servidor antes durante y para finalizar una conversación entre dos usuarios de un servicio de mensajería instantánea.

Recoge por tanto la existencia de un servidor y de un cliente, ejemplos de estos son el servidor del servicio de Google Talk ofrecido por Google y las aplicaciones que funcionan como cliente de este, disponibles para escritorio, móvil y web.

3.5.1 Google Talk Server

El servicio de chat de Google Talk que además viene integrado con GMail está basado en el protocolo XMPP, por lo que cuenta con un servidor XMPP. Esto nos permitirá desarrollar un cliente de mensajería instantánea reutilizando la infraestructura de Google, para ello solo debemos desarrollar un cliente XMPP y usar como servidor el servidor XMPP provisto por Google Talk.

3.5.2 Smack API

Biblioteca de java que implementa un cliente del protocolo XMPP, aunque el API es bastante avanzado y soporta varias extensiones del protocolo, en este proyecto se hace un uso muy básico. La Figura 9 muestra un ejemplo de uso que ilustra la simplicidad del API.

A modo de resumen el API nos permite conectar usuarios a un servidor XMPP, enviar mensajes y registrar escuchadores que reciban mensajes.

```
Connection connection = new XMPPConnection("jabber.org");
connection.connect();
connection.login("mtucker", "password");
Chat chat = connection.getChatManager().createChat("jsmith@jivesoftware.com", new MessageListener() {

    public void processMessage(Chat chat, Message message) {
        System.out.println("Received message: " + message);
    }
});

chat.sendMessage("Howdy!");
```

Figura 9 Ejemplo Smack API

3.6 Apache Subversion

Software de Apache para el control de versiones desarrollado en el lenguaje de programación C, actualmente funciona sobre un sistema FSFS, ofrece mediante una API de funciones en C las operaciones para el control de versiones sobre repositorios de ficheros.

Para explotar esta funcionalidad hay varias formas, en este sistema vamos a colocar un servidor Apache 2.0 delante como controlador frontal del acceso al repositorio, el servidor Apache llevará activo su módulo WebDav (mod_dav), lo que le permitirá recibir peticiones en este protocolo sobre ficheros, referenciados mediante URLs HTTP.

Mediante otro módulo llamado mod_dav_svn se mapearan las peticiones WebDav a llamadas de la API de Apache Subversion.

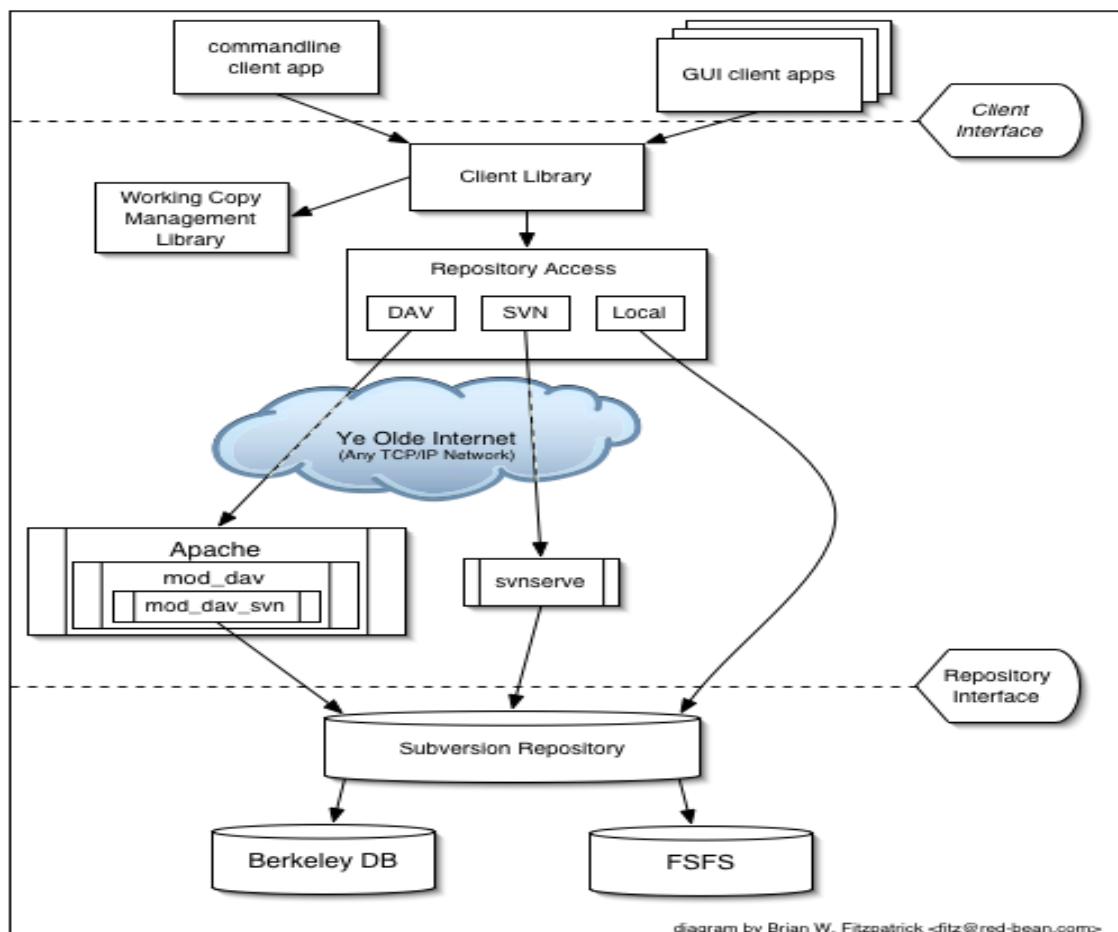


Figura 10: Apache Subversion

3.6.1 Visual SVN Server

Una solución empaquetada y fácil de usar de lo explicado anteriormente es Visual SVN Server, aunque se podría configurar manualmente he preferido usar esta distribución, que posee una

interfaz gráfica para la gestión y configuración de repositorios, y realiza la implementación antes comentada, por lo que permite acceder a los contenidos vía WebDav.

3.6.2 SVN KIT

SVN Kit es una solución evolucionada para el acceso a repositorios SVN, sea vía WebDav o pasando por un SVN Server, en nuestro caso vía WebDav.

Esta biblioteca ofrece dos niveles de abstracción para trabajar con repositorios, una de más bajo nivel (Low-level-API) que permite a los desarrolladores trabajar directamente con el sistema de ficheros y los metadatos del repositorio, y otra de más alto nivel que permite gestionar copias de trabajo como si usásemos un cliente (High-level-API).

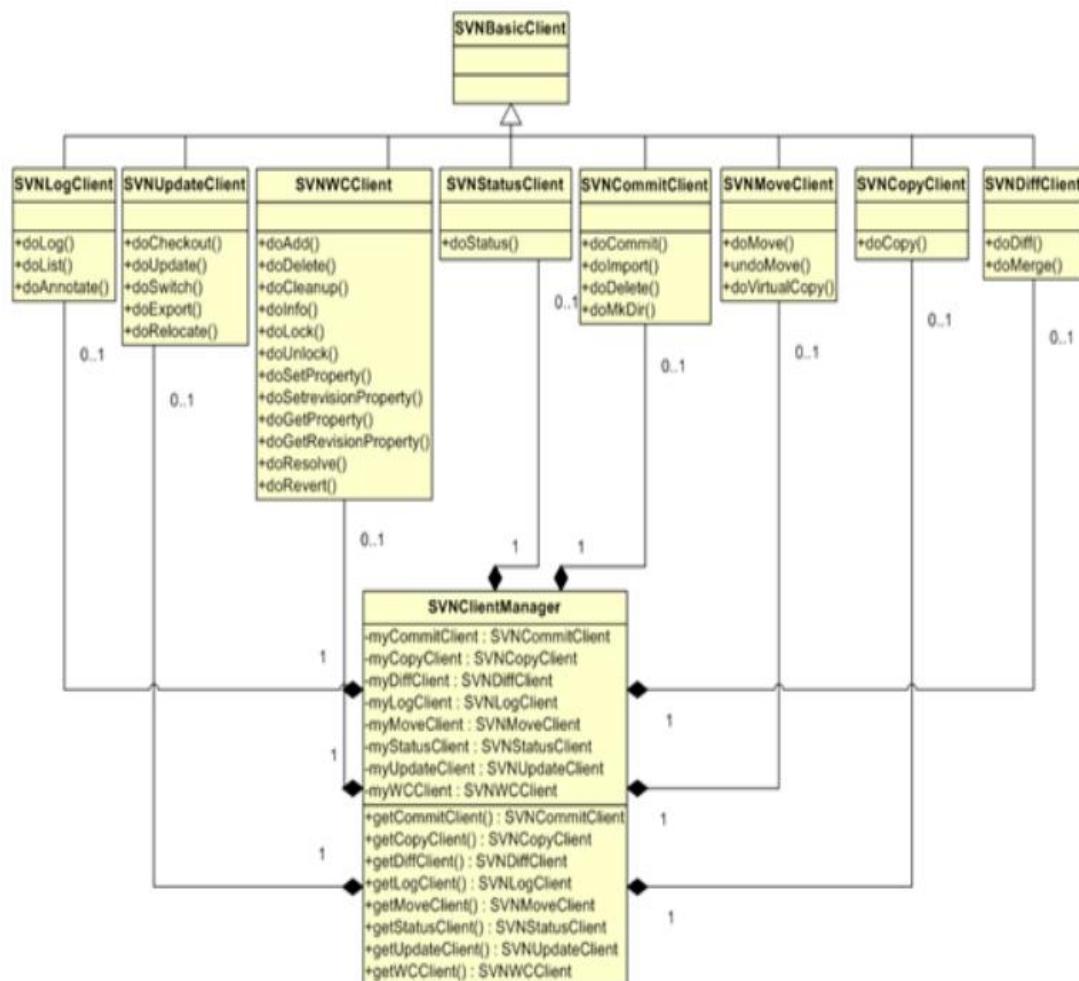


Figura 11: SVN KIT

3.7 PostgreSQL

El sistema de Gestión de bases de datos (SGBD) elegido para el sistema ha sido PostgreSQL, un SGBD relacional, de código libre y gran aceptación en el mercado que cumple todas las propiedades ACID.

Ofrece también un interfaz de gráfica para visualizar y consultar los datos, está disponible tanto para Windows como para Linux y servicios de PaaS como OpenShift lo ofrecen entre sus presentaciones, lo que da fe de su aceptación.

Capítulo 4. Planificación del Proyecto y Resumen de Presupuestos

4.1 Planificación

A continuación se detalla la planificación del proyecto, dando una breve descripción de los perfiles involucrados así como de las principales tareas definidas para la ejecución del proyecto.

Se han definido algo más de 80 tareas, un número relativamente elevado, por lo que se mostrará una vista reducida del árbol de tareas así como del diagrama de Gant.

El proyecto dura 8 meses y es desarrollado íntegramente por 4 personas que se suponen dentro del marco de una empresa pequeña.

4.1.1 Perfiles del Proyecto

Para la planificación se han definido algunos perfiles, que debería tener el equipo del proyecto para que este se desarrollara de una forma rentable.

- **Analista coordinador:** Es el núcleo estratégico del proyecto, encargado de realizar las tareas de análisis y planificación del proyecto, debe contar con un perfil técnico suficientemente alto para estimar, asignar y coordinar las tareas técnicas del resto de los perfiles, así como realizar algunas de estas si fuese necesario.
- **Administrador de sistemas:** Perfil especializado en tareas de administración de sistemas, esto incluye la puesta a punto y el mantenimiento de todo el hardware así como también de todo el middleware. Debe poseer conocimientos avanzados de hardware, redes y middleware, especialmente de servidores de aplicaciones. Además deberá contar con conocimientos de Sistemas de gestión de bases de datos, especialmente de PostgreSQL.
- **Desarrollador experto JEE:** Desarrollador experto en JEE con altas habilidades de diseño, debe dominar un amplio catálogo de patrones de diseño y tener experiencia en el desarrollo con tecnología JEE, especialmente con las tecnologías para el desarrollo de componentes de negocio, EJB y JPA.
- **Desarrollador experto GWT:** Desarrollador experto en desarrollo de aplicaciones enriquecidas, concretamente con altos conocimientos de GWT, además debe contar con conocimientos sobre el desarrollo de aplicaciones web sobre JEE, aunque no se

requieren altos conocimientos, debe conocer bien las especificaciones de JEE que afectan a la presentación, así como tener nociones mínimas las tecnologías que hay por debajo.

4.1.2 Tareas del Proyecto

La Figura 12 muestra una captura de la vista de Gant de la herramienta *Microsoft Project*, en la que se ha excluido el diagrama para visualizar con más detalle el árbol de tareas. Solo se recogen las tareas hamaca, así como un primer nivel de sub tareas, para más detalles puede consultarse el fichero de planificación adjunto en el CD.

- Estudio previo	16 días	jue 01/09/11	jue 22/09/11	
+ Estudio tecnología para ficheros	8 días	jue 01/09/11	lun 12/09/11	
+ Estudio tecnología para chat	5 días	mar 13/09/11	lun 19/09/11	
+ Estudio Google Data APIs	3 días	mar 20/09/11	jue 22/09/11	
Planificación	1 día	vie 23/09/11	vie 23/09/11	1
- Análisis	35 días	vie 23/09/11	jue 10/11/11	
Definición del sistema	7 días	vie 23/09/11	lun 03/10/11	1
Obtención de los requisitos	5 días	mar 04/10/11	lun 10/10/11	18
Revisión de los requisitos	2 días	mar 11/10/11	mié 12/10/11	19
+ Modelado del sistema	13 días	jue 13/10/11	lun 31/10/11	
+ Especificación del sistema	8 días	mar 01/11/11	jue 10/11/11	
Planificación	1 día	vie 11/11/11	vie 11/11/11	17
- Diseño	29 días	vie 11/11/11	mié 21/12/11	
+ Diseño de la arquitectura	5 días	vie 11/11/11	jue 17/11/11	
+ Diseño Busienss tier	8 días	vie 18/11/11	mar 29/11/11	
+ Diseño Web tier	13 días	mié 30/11/11	vie 16/12/11	
+ Diseño Cliente de administración	3 días	lun 19/12/11	mié 21/12/11	
Planificación	1 día	jue 22/12/11	jue 22/12/11	36
- Implementación	83 días	jue 22/12/11	lun 16/04/12	
Definición de herramientas de desarrollo	3 días	jue 22/12/11	lun 26/12/11	36
+ Puesta a punto de los entornos	20 días	mar 27/12/11	lun 23/01/12	
+ Implementación Business tier	22 días	mar 24/01/12	mié 22/02/12	
+ Implementación Web tier	60 días	mar 24/01/12	lun 16/04/12	
+ Implementacion cliente de administración	4 días	lun 19/03/12	jue 22/03/12	

Figura 12: Tareas del proyecto

Es muy importante destacar que la planificación del proyecto no refleja una metodología de desarrollo ni de gestión de proyectos, solamente refleja las tareas que se deberán realizar, quien las realizará y cuánto tiempo costarán.

4.1.3 Diagrama de Gant

La muestra el diagrama de Gant generado por la herramienta Microsoft Project para las tareas definidas, solo se muestran las tareas hamacas y un primer nivel de sub tareas.

Este diagrama refleja de forma más directa la duración de cada tarea y la del proyecto completo. El diagrama de Gant completo se puede ver también en el fichero de planificación adjunto en CD mediante la herramienta Microsoft Project.

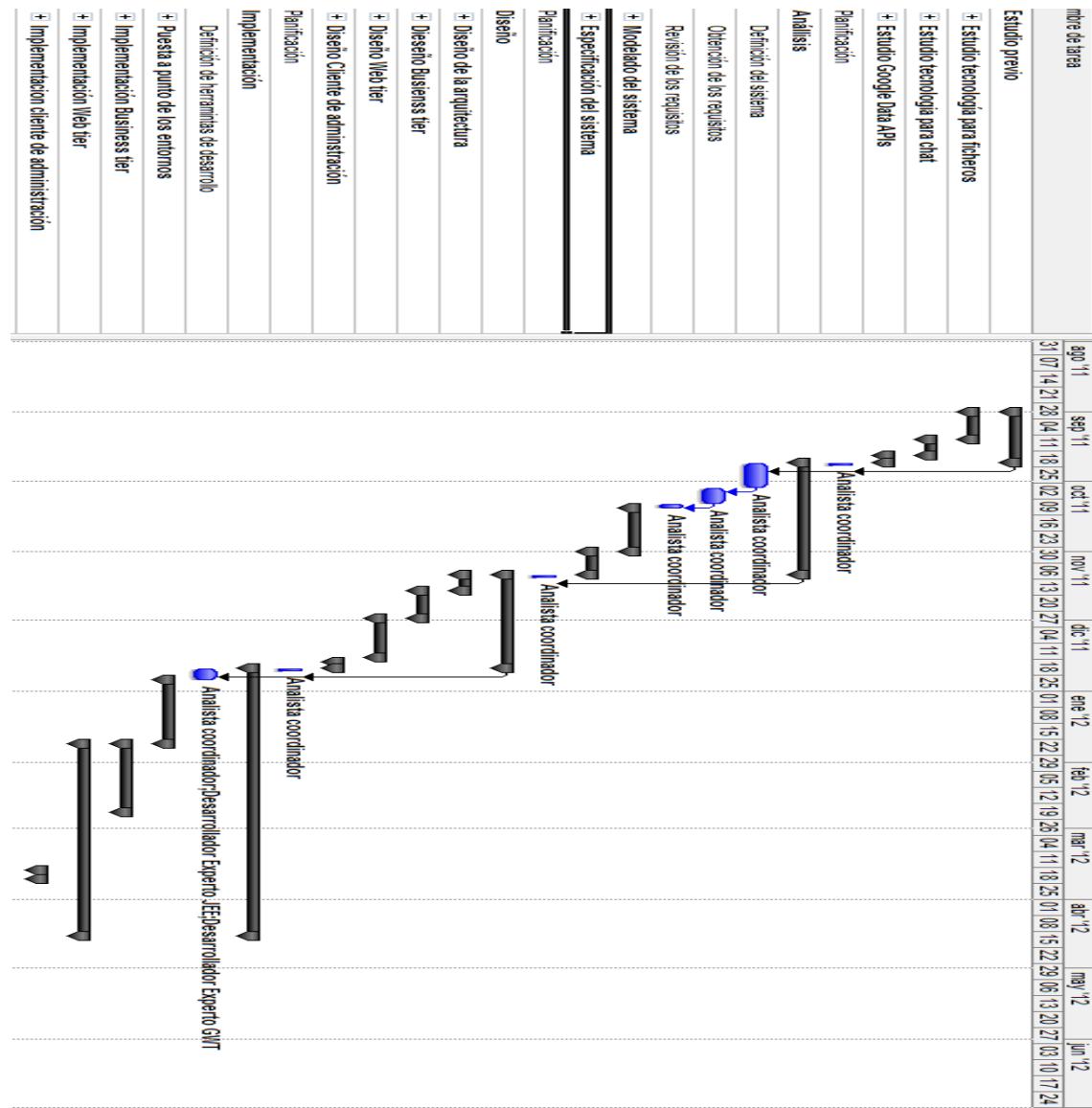


Figura 13: Diagrama de Gant

4.2 Resumen del Presupuesto

En esta sección se resume el presupuesto de costes del proyecto, se han tenido en cuenta el coste de cada recurso humano licencias y hardware. Solo se ha calculado el coste de las horas que los recursos imputan al proyecto, no el coste de todas sus jornadas durante la duración del proyecto.

Ítem	Concepto	Coste Ítem
001	<i>Desarrollo del Sistema</i>	37.976,00 €
002	<i>Licencias de software</i>	2.650,00 €
003	<i>Hardware</i>	11.632,00 €
004	<i>Otros Gastos</i>	2200,00 €
	Subtotal	54.458,72 €
	IVA (18%)	9.802,44 €
	TOTAL	64.260,44 €

Capítulo 5. Análisis

5.1 Definición del Sistema

En este apartado se describe de una forma detallada el sistema que se pretende desarrollar, es una especificación textual de las aplicaciones que compondrán el proyecto.

5.1.1 Introducción

Se pretende crear una aplicación web que provea un marco de trabajo unificado donde desarrollar y compartir proyectos de forma que no se eche en falta ningún recurso externo. Sobre esta aplicación los usuarios podrán crear proyectos o trabajos y compartirlos con otros usuarios.

Los trabajos podrán estar relacionados con diversos campos, un trabajo no será más que una serie de recursos, tales como ficheros, documentos de diversos tipos, carpetas para organizar esos documentos, imágenes o videos.

Habrá recursos contenedores o carpetas, de forma que se podrán tener jerarquías de recursos, pudiéndose mover recursos entre dicha jerarquía. Los recursos podrán ser descargados, y modificados con la herramienta deseada, y vuelto a subir.

El sistema contará con mecanismos para tener a los usuarios comunicados entre sí, para ello se darán tres opciones, correos electrónicos, mensajes internos y mensajería instantánea.

Tanto para la comunicación como para el trabajo conjunto ha de haber un vínculo entre los usuarios, o sea un usuario tendrá una lista de usuarios conocidos o contactos, de los cuales podrá ver la información que estos hayan aportado así como sus proyectos publicados, de esta lista se escogerá con quien establecer una conversación o a quien invitar a un proyecto.

Una vez acabado un proyecto se podrá publicar, esto lo hará visible para todos los usuarios, de forma que podrán acceder a él, mediante una búsqueda o a través de la red de contactos. Cada proyecto tendrá meta información indicada por el usuario en el momento de su creación, esta información servirá además de para describir al proyecto, para realizar búsquedas.

5.1.2 Descripción

5.1.2.1 Primer Contacto

Entrando más en detalle, para acceder a la aplicación será necesaria una cuenta de Google, cuando un usuario acceda por primera vez deberá hacer un proceso de registro, a pesar de que

ya posee las credenciales que usara para entrar al sistema (usuario y contraseña de Google), este no lo dejará pasar si no ha hecho este proceso.

Durante el proceso de registro el usuario aportara la información que aparecerá en su perfil, tanto datos personales como información relacionada con su perfil académico profesional. También aportará el usuario durante esta fase sus credenciales de Google, el sistema validará sus datos, y si son inválidos le dará la oportunidad de repetirlos.

Una vez validados los datos, al usuario se le dará la opción de añadir como contactos a los usuarios del sistema, que tenga como contactos en el servicio de Google Contacts, y que hayan aportado al sistema la misma cuenta de Google (la misma que con la que están relacionados), a estos les llegará una petición que aceptarán o no, a partir de ahí el usuario estará en el sistema y se le irán añadiendo como contactos los usuarios que vayan aceptando la petición.

5.1.2.2 Una Vez Dentro

Una vez en el sistema, el usuario podrá centrarse en su principal objetivo, y del sistema, crear proyectos. El usuario tendrá a su disposición al entrar una serie de opciones distribuidas por secciones dentro de la aplicación.

Tendrá una sección de contactos, donde podrá ver un listado de los usuarios que ya son sus contactos, tendrá un listado de peticiones de otros usuarios que podrá resolver desde aquí, y finalmente tendrá un buscador para localizar usuarios y enviarles una petición.

El sistema ofrecerá al usuario la posibilidad de listar sus contactos de Google Contacts que también son usuarios del sistema, pudiendo seleccionar algunos de estos para enviarles una petición.

Podrá ver los perfiles de sus contactos, así como ver solo los nombres de los contactos de sus contactos que no son contactos de ellos, como nota relevante dentro del perfil de un contacto, además de la información del contacto podrá ver una lista de sus proyectos publicados.

Podrá comunicarse con sus contactos, para ello tendrá una opción análoga a los mensajes de las redes sociales, estos mensajes solo se podrán ver y enviar desde dentro de la aplicación.

Además tendrá la opción de enviar un correo electrónico a un contacto, esto enviará un correo electrónico desde la dirección de Gmail de la cuenta de Google que el usuario haya aportado a la cuenta de Gmail que haya aportado el contacto seleccionado, es simplemente un acceso directo para no salir a enviar un mail, si quiere seguir el hilo de ese mail incluidos rebotes tendrá que ir a su cliente de correo ya que no es el fin del sistema hacer de cliente de correo electrónico, simplemente es un atajo.

Por último, con el fin también de mantener comunicados a los usuarios, el sistema proporciona un cliente de mensajería instantánea, con el que se podrá comunicar con todos los contactos conectados en ese momento. O sea, el usuario vera siempre una lista de los contactos que están conectados, (a este sistema), y podrá iniciar conversaciones con ellos así como atender

las que estos inicien, en principio es un chat simple y asume que los usuarios siempre están disponibles.

Se le dará al usuario la posibilidad de modificar su perfil, un perfil debe tener al menos nombre, apellidos, áreas de interés, estudios y trabajos, todo esto se modificará desde una sección de configuración o administración del perfil, allí también se podrán ver y modificar sus datos de Google.

En su perfil el usuario tendrá un tablón con una serie de notificaciones, ejemplos de notificaciones podrán ser por ejemplo peticiones de contacto, invitaciones a proyecto o la confirmaciones de estas.

De forma transversal al resto de las secciones habrá una sección de reuniones que permitirá concertar reuniones con cualquier número de usuarios, análoga a los eventos de las redes sociales pero con un aspecto más formal, una reunión tendrá un sitio un instante de comienzo, un tema y una descripción, una vez celebrada el creador podrá añadir una conclusión.

5.1.2.3 *Proyectos*

Finalmente el usuario tendrá una sección donde ver sus proyectos, verá un listado de todos los proyectos en los que participa o ha creado, además podrá obviamente crear nuevos proyectos, buscar entre los ya existentes y resolver las invitaciones que le hayan hecho a otros proyectos.

5.1.2.3.1 Accediendo a un Proyecto

Un proyecto tendrá como atributos principales un nombre o título, una descripción, un tema o área y una lista de participantes.

La información principal del proyecto la aportara y modificará el gestor del proyecto, que será por defecto el creador, aunque se podrá ceder este derecho a otro miembro posteriormente.

El gestor es el encargado de incluir y excluir a los miembros del proyecto, al entrar al proyecto se verá la sección inicial la información de este, nombre o título, descripción, temática o área de conocimientos y participantes, además el gestor podrá ir desde aquí a la sección de administración donde podrá cambiar la información, incluir excluir usuarios y ceder el rol de gestor a otro participante.

Dentro de un proyecto se podrá ir a diferentes sub secciones, recursos, referencias, hitos, tareas y discusiones o volver a la descripción del proyecto (sección o vista de inicio del proyecto), en la sección del proyecto además de lo comentado antes habrá al igual que en la ventana principal de la aplicación un chat, pero en este solo participaran los miembros del proyecto.

5.1.2.3.2 Manipulando Recursos

La sección de recursos es el cuerpo del proyecto y consiste en una serie de recursos organizados jerárquicamente.

Como se dijo en la introducción tendremos varios tipos de recursos, ficheros, documentos, directorios o contenedores de recursos, existirá un tipo de recurso none que serán todos aquellos que se añadan al sistema y no encajen en ninguno de los tipos existentes, la tipología de recursos es el punto más flexible a cambios, con lo que se podrá ampliar según necesidades y facilidades.

A efectos de la especificación del sistema, los recursos podrán clasificarse en cuanto su comportamiento en la aplicación, por ejemplo habrán recursos que se puedan visualizar y editar, habrán otros que solo se podrán visualizar y finalmente otros que simplemente estarán ahí. Más formalmente serán editables, visualizables y no visualizables, además cada tipo de recurso tendrá un aspecto distinto.

Cada recurso tendrá opciones, que dependerán del tipo, por ejemplo editar si es editable, eliminar, renombrar, descargar o subir, estas dos últimas opciones permiten al usuario llevarse un recurso a su ordenador, modificarlo y volver a subirlo al sistema.

El proceso de crear un recurso será distinto según el tipo de recurso aunque puede coincidir en varios tipos. En algunos casos se podrá crear el recurso nuevo o importarlo, por ejemplo creando un recurso de tipo file se podrá crear un fichero nuevo o importar el contenido de un fichero desde el equipo, al ser texto plano se podrá escoger cualquier fichero. Un recurso clave es el recurso de tipo folder, un recurso de tipo folder es un recurso contenedor de recursos, que a su vez serán de varios tipos.

Los recursos de tipo file serán editables, contendrán un fichero en texto plano, si se descarga un recurso de estos el resultado será un fichero de texto y como se dijo, al crearlo se podrá importar.

Los recursos de tipo documento comprenden como cabe suponer documentos como pueden ser documentos Word o PDFs, el proyecto no comprende el desarrollo de un editor de documentos, para modificarlos será necesario descargarlos, luego los recursos de tipo documento serán de tipo no editables.

Se dará a demás la opción de importar recursos desde Google Docs, por supuesto de la cuenta que ha aportado el usuario durante el registro.

Además de la sección de recursos habrá otra sección de borradores, donde se podrá tener una colección más amplia de recursos que a lo mejor no son tan importantes y pueden estar apartados, la tipología y la manipulación son las mismas que para los otros recursos, la idea es que estos recursos puedan ser añadidos luego a los recursos del proyecto o que se puedan excluir recursos del proyecto hacia esta zona. Cuando un proyecto es publicado los borradores no estarán visibles.

5.1.2.3.3 Más activos del Proyecto

Además de los recursos, otros activos importantes en un proyecto son sus referencias, las referencias serán links a webs de utilidad para el proyecto, se podrán definir referencias nuevas o añadir referencias que han sido creadas por otros usuarios, para ello se ofrecerá un

buscador de referencias, esto pretende conseguir la colaboración implícita de todos los usuarios de la red colaborativa.

Cada proyecto tendrá también una sección de hitos, aquí se podrá ver en todo momento una descripción del estado del proyecto, definida por el gestor, junto a esta se mostrara el estado esperado del proyecto que será calculado por el sistema a partir de los hitos definidos por los miembros del proyecto, de esta forma, cuando alguien entre a esta pestaña sabrá si el proyecto está parado desde hace seis meses viendo la descripción y viendo la descripción de donde debía estar el proyecto.

Para lograr esto habrá un calendario donde se podrán poner notas a las fechas (estados esperados o hitos), a partir de estas notas se pondrá el mensaje del estado que debería tener el proyecto, este mensaje será el hito con fecha más cercana a la fecha actual. Todos los miembros del proyecto podrán definir hitos, mientras que el estado actual solo lo podrá modificar el gestor. La idea es que para un nuevo proyecto que debe durar una semana Bobby gestor del proyecto defina un hito el primer día, “iniciando Proyecto”, otra hito a los 3 días, “Al 30%” y otro el último día, “Finalizado”.

En estado del proyecto pone “Iniciando”, y si no cambia el estado a los 3 días saldrá:

Estado Actual: “Iniciando”

Estado Deseado: “Al 30%”

En línea con los hitos y con el fin de aplicar un enfoque finalista se creará una sección de tareas, donde un usuario podrá definir tareas, una tarea tendrá por supuesto un nombre, una fecha de comienzo y una duración estimada, los usuarios podrán añadir horas a las tareas, esto aumentara el número de horas trabajadas de la tarea, dato que permitirá saber el porcentaje de la tarea que se ha completado, no obstante para que una tarea aparezca como completada se debe indicar explícitamente, ya que es algo bastante común que las tareas duren más de lo que se pretende.

Una última sección será la de discusiones, allí un usuario podrá crear una discusión, una discusión tendrá un título, unas opciones y una votación que cerrara la discusión pudiendo leerse a posteriori siempre los detalles, en esta pestaña se verá una lista de discusiones por título y estado (abierta, cerrada), cuando un usuario entra a una discusión abierta podrá seleccionar una solución, dando un argumento y así su voto. El gestor podrá dar por cerrada una discusión, esto significa que ya no se podrá votar, pero si se podrá observar las alegaciones y votos, lo más importante es que esto será totalmente anónimo, se sabrán los argumentos y el voto, pero no se sabrá quien fue.

Por ejemplo Bobby miembro participante del Proyecto abre una discusión con título Tecnología de presentación de la aplicación A del proyecto y opciones:

- Struts2
- Spring MVC

- Java Server Faces con Prime Faces
- Google Web Toolkit (GWT)

A continuación vota marcando Spring MVC, y alega un argumento, en la discusión se verá Spring MVC y el argumento, al rato aparecerá debajo Struts2 con su argumento y así con todos los votos, mientras una discusión este abierta no se podrán ver los resultados, una vez que el gestor la cierre todos los miembros podrán ver los resultados.

Falta por hablar de la publicación del proyecto, el gestor de un proyecto tendrá la opción de publicar un proyecto, solo lo podrá hacer el gestor que en principio como hemos dicho será el creador, para ello dispondrá de una opción en la sección de gestión del proyecto.

Hay que valorar con más cuidado la otra cara de un proyecto, o sea el aspecto que tendrá un proyecto para un usuario que no es un participante y que simplemente accede a través de un contacto o mediante el buscador con el fin de consultarla.

Comentamos antes que cada usuario tendrá una lista de contactos y que podría ver su perfil, bien, este perfil para el usuario visitante consta de una serie de información personal más un listado de proyectos, listado de proyectos en los que ha participado el contacto y han sido publicados, cuando un usuario entre a un proyecto ajeno vera tanto la meta información del proyecto como los recursos de este, que no incluyen a los borradores, el resto de secciones del proyecto permanecerán en la intimidad de sus creadores.

El sistema ofrecerá un buscador global de proyectos, mediante el cual el usuario podrá encontrar los proyectos publicados de forma global en la aplicación, he aquí la gran connotación de publicar un proyecto, publicar el proyecto lo hará accesible para todos, tanto conocidos como no conocidos, mientras que si un proyecto permanece sin publicar estará siempre limitado a sus desarrolladores.

5.2 Requisitos del Sistema

Esta sección trata de la obtención así como posterior especificación y modelado de los requisitos del sistema, que se irán agrupando en grupos que posteriormente podrán dar lugar a subsistemas.

Una primera división del sistema ha sido la siguiente:

- Gestión de Usuarios
- Gestión de Contactos
- Gestión de Proyectos
- Gestión de Recursos

Dentro de estos subgrupos se estudiaran las diferentes funcionalidades y particularidades que el sistema debe cumplir.

5.2.1 Obtención de los Requisitos del Sistema

5.2.1.1 Requisitos de Gestión de Usuarios

R1	Gestión de Usuarios.	Requisitos relacionados con la manipulación de Usuarios.
R1.1	Añadir usuarios al Sistema	Los nuevos usuarios realizaran un <u>proceso de registro</u> .
R1.1.1	Aportar Información	El usuario deberá aportar al registrarse cierta información, que aparecerá en su perfil y se usara para el funcionamiento del sistema.
R1.1.1.1	Datos Personales	El usuario aportará nombre y apellidos, fecha de nacimiento, ciudad, y si lo desea foto.
R1.1.1.2	Datos Profesionales	Aportará información acerca de sus estudios y trabajos. Esta información será libre, no habrá ningún tipo de validación.
R1.1.1.3	Datos de Google	La aplicación exige taxativamente que el usuario aporte una cuenta de Google (usuario y contraseña), esta se validará y si no fuese valida se le pedirá repetirla.
R1.1.2	Importar Contactos	Al añadir un nuevo usuario se le mostrará, una lista con los contactos que tiene en Google Contacts y que ya están en el sistema. También el usuario podrá realizar este proceso de importación en otro momento.
R1.2	Eliminar Usuarios del Sistema	El usuario podrá eliminar su perfil en

		cualquier momento o podrá ser eliminado por el sistema.
R1.2.1	Eliminar usuario con proyecto compartido.	Se eliminará su perfil y el proyecto quedará con el resto de los usuarios, si fuera gestor se escogerá otro gestor aleatoriamente.
R1.2.2	Eliminar usuario sin proyecto compartido.	Sus proyectos desaparecerán junto con él.
R1.2.3	Notificar al usuario de la eliminación.	Se le enviará un mail confirmando la eliminación de su perfil del sistema así como de sus datos personales.
R1.3	Modificar Usuarios	Un usuario podrá modificar su información personal desde una <u>sección de administración</u>.
R1.3.1	Modificar Datos Personales	Podrá modificar toda la información personal que ha aportado.
R1.3.2	Modificar Datos Profesionales	El usuario podrá modificar la información profesional que ha aportado, añadiendo nuevos estudios o trabajos, o eliminando algunos de los ya existentes.
R1.3.2	Modificación de su cuenta de Google.	El usuario podrá modificar las credenciales de Google que ha aportado, por otras, siempre y cuando estas sean válidas. Esto es que sean válidas en Google y no haya otro usuario con ellas en el sistema.

5.2.1.2 Requisitos de Gestión de Contactos

R2	Gestión de Contactos	Requisitos sobre la gestión de las relaciones entre usuarios.
R2.1	Añadir Usuario como Contacto	Se podrán añadir usuarios a la lista de contactos, para esto habrá una petición y una confirmación, siempre.
R2.1.1	Importar de Google Contacts	Podrá importar los contactos de Google Contacts que estén en el sistema. Para ello se listaran, y se les enviará una petición, que estos aceptarán o no. En el caso en el que es aceptada, el solicitante recibirá una notificación.
R2.1.2	Agregar usuarios del sistema	Existirá un buscador que permitirá encontrar usuarios en el sistema dados unos parámetros. A estos usuarios se les podrá enviar una petición.
R2.2	Interactuar con Contacto	Cada usuario tendrá una serie de contactos con los que interactuará.
R2.2.1	Ver perfil de contacto	Se podrá visitar el perfil de todos los contactos, en cualquier momento desde la sección de contactos.
R2.2.1.1	Ver información del contacto	En el perfil de un contacto se verá, su información personal, sus estudios, sus empleos, sus contactos y sus proyectos. Los proyectos que se listarán serán solo los que hayan sido publicados.
R2.2.1.2	Ver proyecto de contacto	Al entrar en el proyecto de un contacto, se verá la información del proyecto, y una visión restringida de los recursos del proyecto, en la cual no se verán los borradores ni se podrán modificar los recursos.
R2.2.2	Comunicarse con contacto	Los usuarios se podrán comunicar con sus contactos.
R2.2.2.1	Chatear con contacto	La aplicación proporcionara un sistema por el cual hablar con los contactos conectados.
R2.2.2.1.1	Chat global	En la ventana principal el usuario vera una lista de los contactos conectados, de esta lista el usuario podrá seleccionar con quien establecer una conversación.
R2.2.2.1.2	Chat de proyecto	En la sección de un proyecto habrá otro chat, una lista con los contactos que son miembros de ese proyecto. Esto posibilita la comunicación entre

		usuarios que no sean contacto pero tengan proyectos en común.
R2.2.2.2	Mandar mensaje a contacto	Habrá un sistema de mensajes internos de formato sencillo, este mecanismo permite que un usuario envíe un mensaje a un contacto y que este lo responda.
R2.2.2.3	Enviar mail a contacto	Al enviar un mensaje se podrá indicar que este sea enviado también como correo electrónico. Este correo electrónico usará las direcciones de Gmail que han aportado los usuarios durante el registro.
R2.2.3	Reunirse con contactos	Habrá un sistema de reuniones análogo a los eventos de las redes sociales.
R2.2.3.1	Convocar una reunión	Un usuario podrá convocar una reunión, al convocarla deberá aportar cierta información: <ul style="list-style-type: none"> • Lugar • Fecha y hora • Asunto • Descripción
R2.2.3.2	Invitar Contactos a reunión	El creador de la reunión podrá invitar contactos a la reunión. Los usuarios invitados recibirán una invitación en la sección de reuniones. En la reunión aparecerán como asistentes los usuarios que acepten la invitación. Cada usuario tendrá una lista de las reuniones que tiene, ordenadas por fecha. El usuario que concierta la reunión también la tendrá en su lista.
R2.2.3.3	Finalizar reunión	El usuario creador de la reunión tendrá la responsabilidad de marcarla como concertada. Las reuniones podrán ser eliminadas por el creador sin ser celebradas, en este caso simplemente desaparecerá del listado de reuniones de los participantes.
R2.3	Buscar Usuarios	Se podrá intentar localizar a un usuario que no se tenga como contacto.
R2.3.1	Buscar manualmente	Se podrá recorrer la lista de contactos, viendo que contactos tienen hasta dar con el buscado si se puede.
R2.3.2	Buscar con buscador.	Habrá un buscador para buscar usuarios en el sistema especificando sus datos.

R2.3.2.1	Datos de entrada del buscador	El buscador recibirá como parámetros de búsqueda el nombre y los dos apellidos.
R2.3.2.2	Salida del Buscador	El buscador dará como salida aquellos usuarios que tengan alguno de los tres parámetros. Mostrará el nombre y los dos apellidos de los usuarios encontrados. Permitirá enviar una petición a los resultados.
R2.4	Eliminar Contacto	Se podrán eliminar usuarios de la lista de contactos.
R2.4.1	Confirmar eliminación a usuario	El usuario recibirá confirmación ipso facto de que ha eliminado a un contacto.
R2.4.2	Confirmar eliminación a contacto	El contacto no recibirá confirmación de la eliminación pero ya no tendrá al contacto en su lista. Esta eliminación no afectará a los proyectos, que se tengan en común.

5.2.1.3 Requisitos de Gestión de Proyectos

R3	Gestión de Proyectos.	Requisitos relacionados con la creación, modificación y manipulación de proyectos.
R3.1	Crear Proyecto	Los usuarios podrán crear un nuevo proyecto en cualquier momento.
R3.1.1	Aportar meta información	En el momento de la creación el usuario deberá aportar la meta información del proyecto.
R3.1.1.1	Aportar Titulo	Deberá aportar un título para el proyecto.
R3.1.1.2	Aportar descripción	El usuario deberá aportar una descripción para el proyecto.
R3.1.1.3	Aportar Temática o Área de Cocomimientos.	El usuario escogerá entre una serie de aéreas posible la de su proyecto. (Ingeniería, Literatura, Filosofía,...)
R3.1.2	Invitar contactos	Opcionalmente en esta fase el usuario podrá seleccionar contactos para que se unan al proyecto durante la fase de creación. -Los usuarios invitados recibirán una invitación, que aceptaran o rechazarán. -Los participantes del proyecto deben ser contactos del creador del proyecto, pero no tienen que estar relacionados entre ellos.
R3.1.3	Gestor del proyecto	El proyecto tendrá un gestor, este tendrá más capacidades, por defecto será el creador.
R3.1.3.1	Sesión del rol de gestor	El gestor podrá ceder el rol de gestor del proyecto a otro usuario una vez creado el proyecto, perdiéndolo el.
R3.1.3.2	Cambio forzado	Si el gestor del proyecto desaparece del sistema, entonces se asignará a uno de los usuarios gestor, aleatoriamente.
R3.1.3.3	Capacidades del gestor	El gestor tendrá una serie de privilegios o responsabilidades que no tendrán el resto de los participantes. <ul style="list-style-type: none"> • Eliminar participantes • Eliminar el proyecto • Publicar el proyecto • Otras
R3.1.4	Ámbito del Proyecto	El proyecto tendrá siempre un ámbito con respecto a los usuarios del sistema.
R3.1.4.1	Ámbito Privado	El proyecto no ha sido publicado, es el ámbito por defecto.

		<p>-Los contactos de gestor y participantes no verán el proyecto en la lista de proyectos de estos.</p> <p>-Desde el buscador global no se podrá ni si quiera ver el nombre de este proyecto.</p>
R3.1.4.2	Ámbito Público	<p>Una vez que el gestor publica el proyecto.</p> <p>-Los contactos de gestor y participantes verán el proyecto en la lista de proyectos de estos y podrán acceder a este.</p> <p>-Desde el buscador global se podrá encontrar el proyecto y acceder a él.</p>
R3.1.5	Estructura del Proyecto	<p>Un proyecto tiene la siguiente estructura:</p> <ul style="list-style-type: none"> • Sección de Inicio • Recursos del proyecto • Referencias • Hitos • Discusiones • Tareas
R3.1.5.1	Sección de Inicio	<p>Sección o vista inicial del proyecto, servirá de resumen del proyecto para los participantes.</p> <p>-Mostrará Titulo, Tema y Descripción del proyecto, e verán también los usuarios participantes y el gestor podrá ir a la sección de administración del proyecto</p>
R3.1.5.2	Árbol del proyecto.	<p>El árbol del proyecto es el cuerpo de este propiamente dicho.</p> <p>El cuerpo del proyecto es un árbol de recursos.</p> <p>-Sobre el árbol se pueden gestionar los recursos que se estimen para el proyecto.</p> <p>-Cuando el proyecto adquiera ámbito público el árbol será íntegramente visible.</p>
R3.1.5.3	Recursos libres.	<p>El proyecto tendrá una sección para manejar recursos de forma libre.</p> <p>-Estos recursos estarán al margen del cuerpo del proyecto y no se publicarán.</p> <p>-Se podrán gestionar recursos, y añadir al árbol del proyecto o quitar de esta hacia aquí.</p>
R3.1.5.4	Referencias del Proyecto	<p>Habrá una sección de referencias. Las referencias tendrán un nombre y un URL, pudiendo los usuarios buscar entre las referencias definidas por otros usuarios.</p>
R3.1.5.5	Hitos del proyecto.	<p>Sección donde se gestionara los hitos del proyecto.</p> <p>Se podrán definir hitos sobre un calendario,</p>

		<p>los hitos tendrán un nombre descriptivo y una fecha.</p> <ul style="list-style-type: none"> - El proyecto tendrá un estado actual, que lo asignara el gestor. -El proyecto tendrá un estado esperado que lo asignará el sistema en función de los hitos definidos.
R3.1.5.6	Discusiones del proyecto.	<p>Sección con un listado de discusiones.</p> <ul style="list-style-type: none"> - Una discusión consta de un título, una descripción, unas opciones a discutir y una votación. Cada voto irá acompañado de un argumento o alegación. - Una discusión podrá estar abierta o cerrada. - Para votar escogerá una opción, y se aportará un argumento, esto será anónimo.
R3.1.5.7	Tareas del proyecto	<p>Se podrán definir tareas dentro de un proyecto.</p> <p>Una tarea tendrá un nombre, una fecha de inicio y una duración estimada.</p> <p>Los usuarios podrán añadir horas trabajadas a la cuenta de una determinada tarea.</p> <p>La tarea se podrá marcar como completada de forma independiente al número de horas trabajadas.</p>
R3.2	Administrar proyecto	Habrá una sección de administración o configuración, a la que solo accederá el gestor.
R3.2.1	Modificar Meta información	Desde la sección de administración se podrá cambiar la meta información del proyecto, Titulo, Área o Tema y Descripción.
R3.2.2	Incluir Contactos	A pesar del proceso inicial, en cualquier momento el gestor podrá invitar contactos al proyecto desde la sección de administración.
R3.2.3	Excluir contactos	El usuario gestor podrá excluir a un usuario del proyecto. Desde la sección de administración.
R3.2.4	Ceder rol de gestor	Desde la sección de administración del proyecto el gestor podrá ceder la posición de gestor.
R3.2.5	Publicar proyecto	Desde esta sección se publicara el proyecto, esto lo pondrá disponible para

		<p>todos los usuarios del sistema. Esta acción es irreversible.</p>
R3.2.6	Eliminar proyecto	El usuario gestor podrá eliminar un proyecto desde la sección de administración.
R3.2.6.1	Eliminar Proyecto Individual	La eliminación de un proyecto donde solo está el gestor o creador en este caso. El proyecto se eliminara sin contratiempos.
R3.2.6.2	Eliminar Proyecto Compartido	Si hay varios usuarios, se eliminara también.
R3.2.6.3	Eliminar proyecto con invitaciones en curso	Las invitaciones existentes desaparecerán.
R3.3	Planificar estados del Proyecto	Los usuarios tendrán opciones relacionadas con la planificación del proyecto.
R3.3.1	Crear Hito u objetivo	El usuario gestor, podrá indicar para una fecha un hito u objetivo del proyecto. En la sección de planificación aparecerá el último hito que debió ser superado como el estado deseado del proyecto.
R3.3.2	Establecer estado actual	En la sección de planificación aparecerá encima del último hito superado o estado deseado del proyecto el estado actual. Solo el gestor podrá establecer el estado actual.
R3.4	Discutir en Proyecto	Los usuarios tendrán una sección para llevar a cabo discusiones.
R3.4.1	Crear Nueva Discusión	Cualquier usuario podrá crear una nueva discusión desde la sección de discusiones. Cada discusión tendrá un nombre una descripción y unas opciones.
R3.4.2	Votar en discusión	Cada usuario tendrá un voto en la discusión, será anónimo e irá acompañado de una alegación, una vez que un usuario vote no podrá hacerlo más.
R3.5	Tareas del proyecto	El proyecto tendrá una sección de tareas.
R3.5.1	Crear Tarea	Los usuarios podrán crear tareas dentro de la sección de tareas del proyecto. Cada tarea tendrá un nombre, una fecha de comienzo y una duración estimada.
R3.5.2	Trabajar en Tarea	Los usuarios podrán añadir horas trabajadas a las tareas.
R3.6	Abandonar Proyecto	Un usuario podrá abandonar un proyecto cuando lo desee.

R3.6.1	Abandono de un usuario no gestor.	Un usuario no gestor podrá abandonar en cualquier momento le proyecto.
R3.6.2	Abandono del gestor	Un usuario gestor podrá abandonar cediendo antes el status de gestor a otro, en el caso idóneo, y si no lo hace se seleccionara como gestor otro usuario aleatoriamente que podrá en caso de no estar a gusto cederlo posteriormente.

5.2.1.4 Requisitos de Gestión de Recursos

R4	Gestión de Recursos	Requisitos relacionados con la manipulación de Recursos.
R4.1	Crear Recurso	Se podrán crear recursos, desde cero, importando desde local o desde Google Docs.
R4.1	Eliminar	Todos los recursos tendrán una opción para eliminarlos, con esto se eliminará los recursos de forma permanente.
R4.1.1	Crear nuevo recurso	<p>Se podrán crear recursos desde la nada tanto en los recursos del proyecto como en la sección de borradores.</p> <p>Habrá varios tipos de recursos :</p> <ul style="list-style-type: none"> • Tipo File • Tipo Folder • Tipo Documento • Tipo none <p>También se distinguirá entre recursos editables, visualizables y no visualizables.</p> <p>Un recurso tendrá opciones que dependerán del tipo.</p>
R4.1.1.1	Crear nuevo recurso de tipo folder en blanco.	<p>Un recurso que contiene a otros recursos o está vacío, al descargarse se ve como una carpeta.</p> <p>Se podrá de hecho ser lo más natural crear carpetas vacías.</p>
R4.1.1.2	Crear nuevo Recurso de tipo file en blanco.	<p>Este es un recurso muy básico, abstrae a todo lo que sea un fichero.</p> <p>Se podrá y será bastante habitual crear nuevos ficheros.</p>
R4.1.1.3	Crear nuevo recurso de tipo de documento en blanco.	Estos recursos abstraen Documentos como puede ser documentos Word, PDF y ODT.
R4.1.1.5	Crear nuevo recurso de tipo none	<p>Habrá un tipo de recurso none o desconocido, que es necesario para permitir tráfico de cualquier tipo de ficheros, cualquier fichero se podrá subir como none.</p> <p>Los recursos none no se podrán editar ni se podrán visualizar.</p>
R4.1.2	Importar Recurso Local	Al crear un recurso se dará la opción de importarlo del equipo.
R4.1.2	Excluir hacia la zona de borradores	Todos los recursos tendrán la opción de ser excluidos hacia la zona de recursos libres. Se podrán incluir posteriormente de borradores hacia los recursos del proyecto.

R4.1.3	Importar recurso de Google Docs	Algunos recursos se podrán importar desde Google Docs, escogiendo entre los que se tengan allí y estén soportados por el sistema.
R4.2	Modificar Recurso	Todos los recursos tendrán una serie de opciones para su modificación.
R4.2.1	Editar Contenido Online	Algunos tipos de recurso podrán ser editados mediante un editor online. En principio se podrán editar solo los ficheros.
R4.2.2	Renombrar Recurso	Se podrá renombrar siempre un recurso.
R4.2.3	Mover Recurso	Se podrá mover un recurso dentro del árbol.
R4.2.4	Descargar Recurso	Todos los tipos de recurso podrán ser descargados.
R4.2.5	Subir Recurso	Se podrá volver a subir un fichero para que reemplace el contenido de un recurso.
R4.3	Eliminar Recurso	Se podrá eliminar un recurso en todo momento.
R4.3.1	Eliminar recursos simples	Los ficheros y recursos simples se podrán eliminar sin contratiempos.
R4.3.2	Eliminar recursos de tipo folder	Los recursos de tipo carpeta podrán ser eliminados incluso si contienen otros recursos.

5.2.1.5 Requisitos No funcionales

R4	Gestión de Recursos	Requisitos relacionados con la manipulación de Recursos.
R4.1	Requisitos de Usuario	-Existen algunos requisitos que deben cumplir los usuarios de la aplicación.
R4.1.1	Uso del ordenador	El usuario debe estar familiarizado con el uso del ordenador.
R4.1.2	Conocimientos de la web	Los usuarios deberán tener conocimientos básicos sobre el uso de la web, y estar familiarizados con el uso de aplicaciones web.
R4.1.3	Conocimientos de redes sociales.	Conviene que los usuarios estén relacionados con el uso de redes sociales.
R4.1.4	Conocimientos de gestión de proyectos.	No se requieren conocimientos avanzados de gestión de proyectos, basta con tener unos objetivos.
R4.2	Requisitos de Seguridad	Existen restricciones en cuanto al acceso al sistema.
R4.2.1	El acceso al sistema debe ser restringido.	Se deberá garantizar un mecanismo de autenticación para entrar al sistema.
R4.3	Requisitos Tecnológicos	Existen restricciones acerca de la tecnología.
R4.3.1	Sistema distribuido	El sistema debe estar distribuido de forma que se puedan realizar distintas aplicaciones sobre él.
R4.3.2	Patrón de despliegue en n niveles.	Además de ser un sistema distribuido deberá seguir un patrón de despliegue en n niveles.
R4.3.3	Aplicación enriquecida	La aplicación principal del sistema debe estar construida con alguna tecnología RIA.
R4.3.4	Sistema Operativo	El sistema debe ser portable entre distintos sistemas operativos, particularmente debe poder funcionar en sistemas Windows y en sistemas Linux.

5.2.2 Especificación y Diagramas de Casos de Uso

Para modelar el sistema he contemplado los casos de uso más generales o también llamados casos de uso primarios.

Antes de entrar en los diagramas de casos de uso he puesto por separado los distintos actores del sistema y una breve explicación.

Por último se puede ver una descripción detallada de los distintos escenarios.

5.2.2.1 Actores del Sistema

Los actores que he identificado son los siguientes:

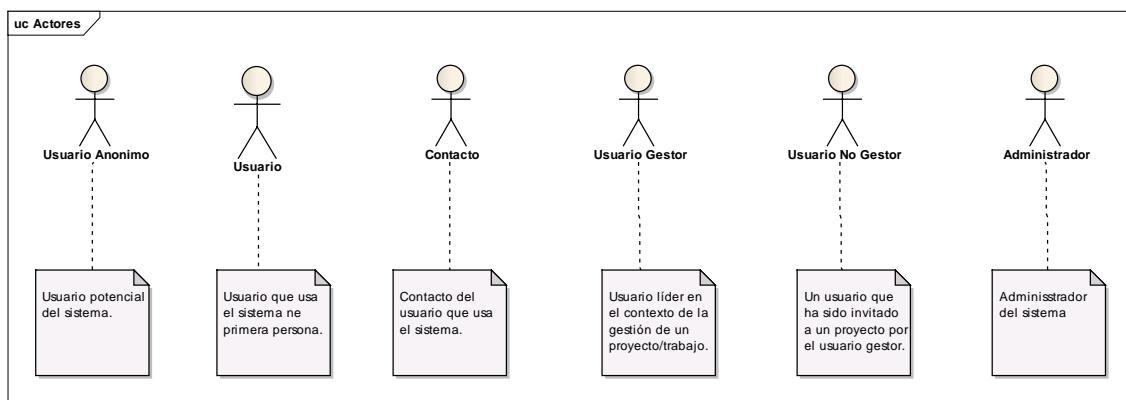


Figura 14 : Actores del sistema

Usuario Anónimo: Usuario potencial del sistema, modela a un usuario que accede a la aplicación por primera vez en su vida, con lo cual deberá realizar un proceso de registro antes de convertirse en un usuario de esta.

Usuario: Es un usuario registrado del sistema, ha hecho el proceso de registro y está en condiciones de usar el sistema. Este actor modela al usuario registrado en un contexto genérico de la aplicación, para contextos más específicos se usarán otros actores.

Contacto: Modela a un usuario que está relacionado o es contacto del usuario que se toma como referencia. Permite modelar interacciones de un usuario con sus contactos, que también son usuarios del sistema, pero en el contexto de esa acción no son los usuarios principales.

Usuario gestor: Es una especialización de un usuario registrado en el sistema, que modela acciones enmarcadas en el contexto de un proyecto, este actor representa al usuario que ha creado el proyecto y tiene por tanto los derechos y deberes relacionados con la administración de este.

Usuario no gestor: Modela lo contrario que el actor anterior, este actor es una especialización de un usuario registrado, en el contexto de un proyecto, pero que no posee derechos de administración sobre dicho proyecto.

Administrador: Este actor modela a la persona que realiza las tareas administrativas del sistema, no es un usuario del sistema en ningún contexto, interactúa con él desde el exterior para asegurarse de que todo vaya bien.

5.2.2.2 Diagramas de Casos de Uso

El siguiente diagrama muestra los casos de uso de más alto nivel del sistema.

Cada uno de estos casos de uso representa un área funcional del sistema que podemos descomponer en más casos de uso o incluso en alguna sub área más específica.

5.2.2.2.1 Casos Generales

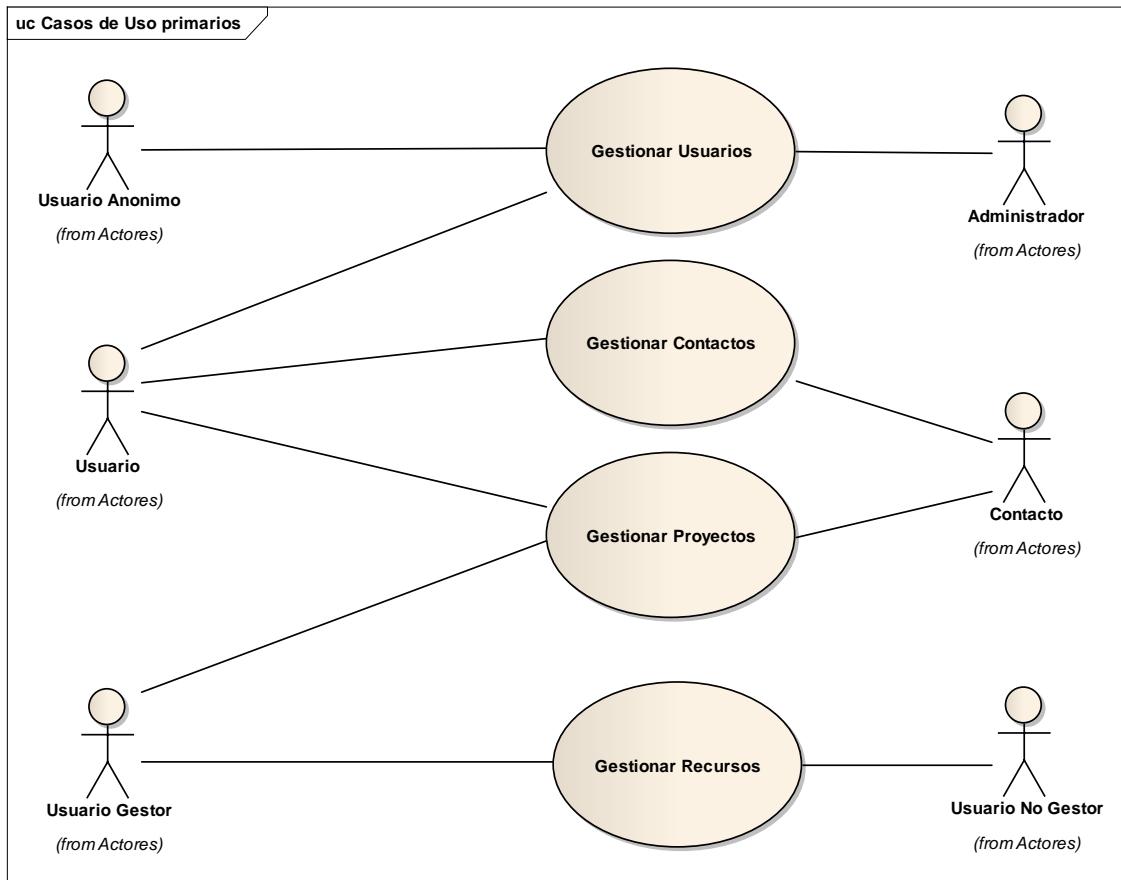


Figura 15: Casos de uso primarios

Cada uno de los casos de uso mostrados en la Figura 15 representa un área funcional del sistema que engloba a su vez otras funcionalidades más concretas representadas con casos de uso más específicos.

A continuación se expondrán cada uno de los casos de uso primarios, así como una definición del caso de uso, los actores que participan y los requisitos cubiertos.

5.2.2.2 Gestionar Usuarios

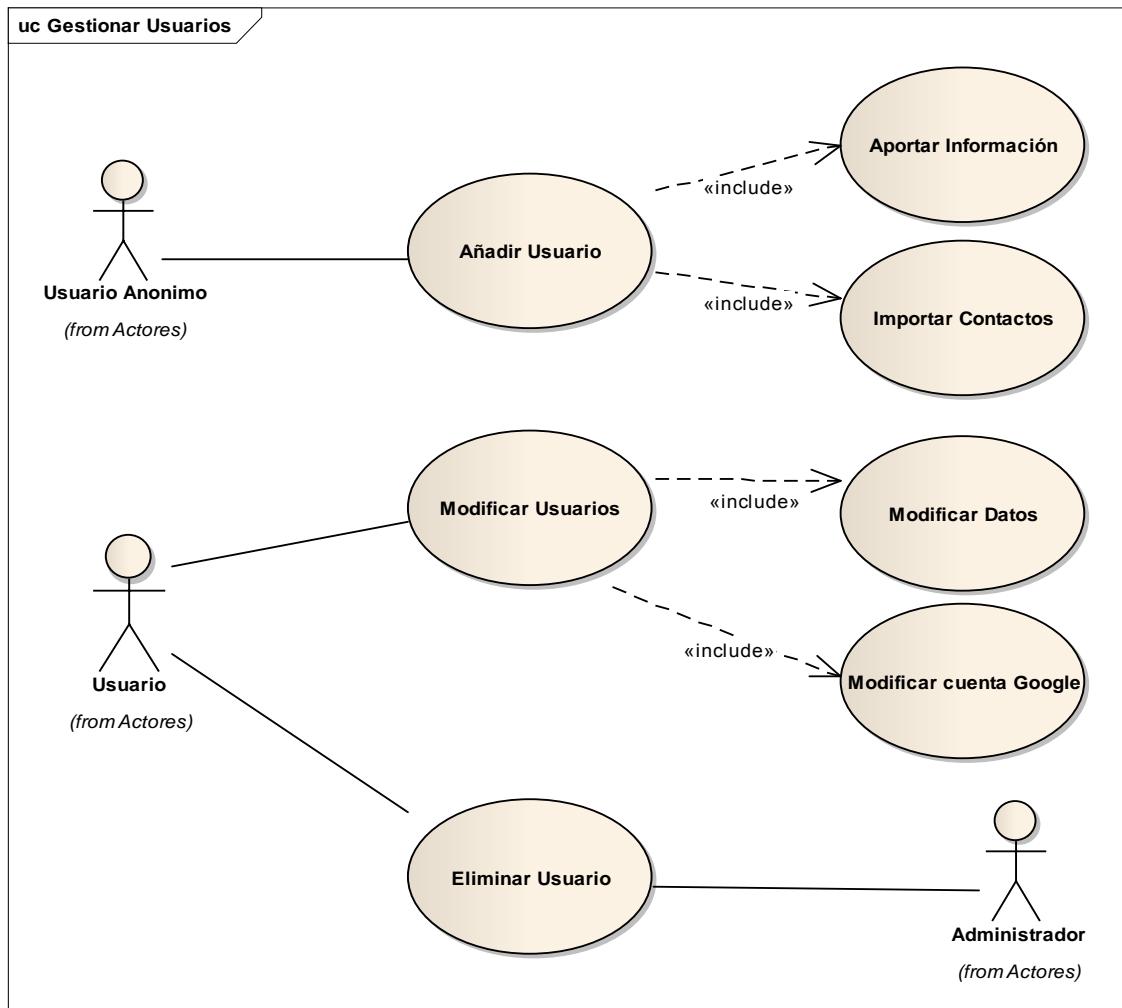


Figura 16: Caso de uso Gestión de Usuarios

Caso de Uso 1.1 : Añadir Usuario	
Definición	<p>Este caso de uso modela la incorporación de un usuario anónimo al sistema. El usuario anónimo realizará un proceso de registro, que consistirá en aportar su información personal, así como una cuenta de Google válida, que esté en funcionamiento y que no la posea ningún usuario que este ya en el sistema. Una vez aportada esta información el usuario podrá importar sus contactos del servicio de contactos de Google, solo los que ya estén en el sistema. Las credenciales de Google aportadas serán las credenciales que deberá aportar el usuario para acceder al sistema una vez completado el registro.</p>

Actores	Usuario Anónimo
Prioridad	Normal
Requisitos Cubiertos	R1.1 Añadir Usuarios al Sistema

Caso de Uso 1.1.1 : Aportar Información	
Definición	<p>Este caso de uso modela el paso de aportar la información durante el proceso de registro, incluido en el caso de uso 1.1.</p> <p>Este caso de uso comprende la aportación por parte del usuario, tanto de datos personales como de su cuenta de Google.</p> <p>Se debe validar toda la información aportada por el usuario.</p> <p>No se le permitirá dejar ningún campo sin completar.</p> <p>No se permitirá usar una cuenta de Google que ya esté en el sistema.</p> <p>Se validará la cuenta de Google comprobando que el usuario y contraseña aportados son correctos.</p> <p>Se pedirá confirmación de la contraseña para evitar errores.</p> <p>Los datos personales comprenden nombre, apellidos, ciudad, fecha de nacimiento y sitio web.</p> <p>Además se le permitirá aportar información académica y profesional, esta información será aportada por el usuario de forma libre.</p> <p>Podrá definir títulos y añadirlos a su lista de títulos.</p> <p>Podrá definir empleos y definirlos a su lista de empleos.</p>
Actores	Usuario Anónimo
Prioridad	Normal
Requisitos Cubiertos	R1.1.1 Aportar Información

Caso de Uso 1.1.2 : Importar Contactos	
Definición	<p>Este caso de uso, sub caso del caso 1.1, modela la acción mediante la cual el usuario importa sus contactos desde el servicio de contactos de Google.</p> <p>Para llegar a este punto se debe pasar con éxito el caso de uso 1.1.1.</p> <p>Se le ofrecerá al usuario un listado de sus contactos de Google Contacts que ya se han registrado en el sistema.</p> <p>Los contactos que el usuario seleccione de este listado recibirán una petición de contacto.</p>

	El usuario podrá pasar por alto este paso sin seleccionar ningún contacto.
Actores	Usuario Anónimo
Prioridad	Normal
Requisitos Cubiertos	R1.1.2 Importar Contactos

Caso de Uso 1.2 : Modificar Usuario	
Definición	Modificación de la información relativa a un usuario registrado del sistema. En cualquier momento un usuario podrá modificar la información aportada durante el proceso de registro. Podrá modificar todos sus datos personales. Podrá modificar los títulos que ha añadido como información académica, quitando los aportados o aportando nuevos. Podrá modificar los empleos aportados eliminándolos o aportando nuevos empleos. Podrá modificar su usuario y su contraseña, aportando otros nuevos, estos deben ser tan bien validos como cuenta de Google.
Actores	Usuario
Prioridad	Normal
Requisitos Cubiertos	R1.3 Modificar Usuarios

Caso de Uso 1.2.1 : Modificar Datos	
Definición	Sub caso de uso del caso 1.2, en este caso el usuario modifica los datos que ha aportado, a modo de descripción personal. Se le mostrarán los datos actuales al usuario. El usuario podrá modificar solo aquellos que desee. Se validará que no deje campos en blanco. Este proceso se podrá repetir tantas veces como el usuario deseé.
Actores	Usuario
Prioridad	Normal
Requisitos Cubiertos	R1.3.1 Modificar Datos Personales

Caso de Uso 1.2.2 : Modificar cuenta de Google	
Definición	En este paso, el usuario modifica la cuenta de Google que ha aportado. La nueva cuenta debe pasar el mismo filtro que la anterior, debe estar activa en el sistema de Google, y no debe pertenecer a ningún usuario del sistema.
Actores	Usuario
Prioridad	Normal
Requisitos Cubiertos	R1.3.2 Modificación de su cuenta de Google

Caso de Uso 1.3 : Eliminar Usuario	
Definición	Modela la eliminación de un usuario del sistema. Este usuario desaparecerá del sistema, así como los proyectos en los que este solo él. El sistema gestionara los proyectos que tenga con otros usuarios para que prevalezcan. La eliminación puede ser tanto voluntaria, autoeliminación, como forzada por el administrador del sistema.
Actores	Usuario, Administrador
Prioridad	Normal
Requisitos Cubiertos	R1.2 Eliminar Usuarios del Sistema

5.2.2.2.3 Gestionar Contactos

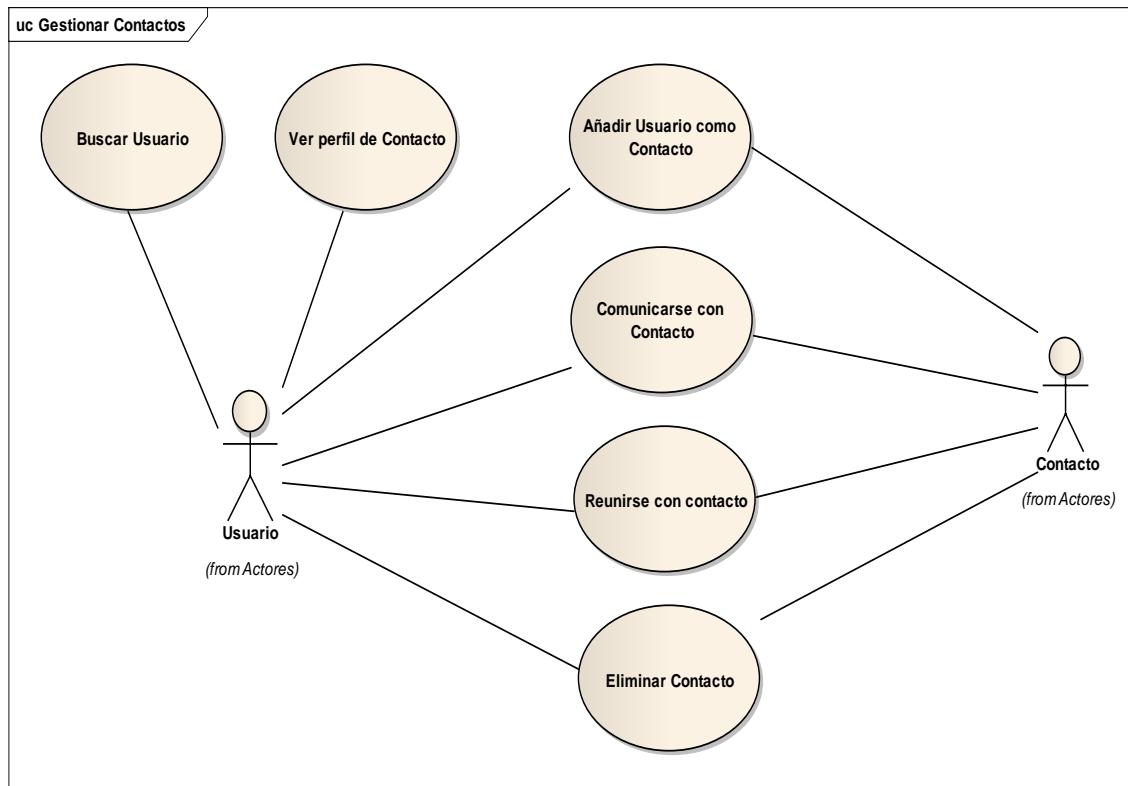


Figura 17: Caso de uso Gestión de Contactos

Caso de Uso 2.1 : Buscar Usuario	
Definición	Se refiere a la acción por parte de un usuario de localizar a otro usuario en el sistema. Existirá un buscador para encontrar usuarios en el sistema. El buscador usará tanto el nombre como los apellidos. El buscador mostrará nombre y apellidos de los usuarios que coincidan parcialmente con los parámetros aportados. El buscador no mostrará los usuarios que ya se estén en la lista de contactos del usuario que realiza la búsqueda.
Actores	Usuario
Prioridad	Normal
Requisitos Cubiertos	R2.3 Buscar Usuario

Caso de Uso 2.2 : Añadir Usuario como Contacto	
Definición	Este caso de uso representa el proceso mediante el cual un usuario intenta añadir a otro usuario del sistema como contacto. El usuario que pretenda agregar a otro usuario a su lista de contactos, debe buscarlo

	<p>y enviar una petición.</p> <p>El usuario pretendido recibirá la petición de contacto.</p> <p>Si el usuario acepta la petición, ambos usuarios tendrán un nuevo usuario en su lista de contactos.</p> <p>Si el usuario rechaza la petición todo quedará como antes.</p> <p>No se podrán enviar peticiones de contacto a usuarios que ya se tengan en la lista de contactos.</p>
Actores	Usuario, Contacto
Prioridad	Normal
Requisitos Cubiertos	R2.1 Añadir Usuario como Contacto

Caso de Uso 2.3 : Ver Perfil de Contacto	
Definición	<p>Modela una interacción de un usuario con su contacto, en la que el usuario simplemente accede a la información de un contacto.</p> <p>El usuario verá toda la información personal de su contacto.</p> <p>El usuario verá toda la información académica de su contacto.</p> <p>El usuario verá toda la información profesional de su contacto.</p> <p>El usuario verá un listado de todos los contactos de su contacto.</p> <p>El usuario verá un listado de todos los proyectos publicados de su contacto.</p> <p>El usuario podrá ver la información de los proyectos publicados de su contacto.</p> <p>El usuario podrá ver todos los recursos de los proyectos publicados de sus contactos.</p>
Actores	Usuario
Prioridad	Normal
Requisitos Cubiertos	R2.2.1 Ver perfil de contacto

Caso de Uso 2.4 : Comunicarse con Contacto	
Definición	<p>El usuario se comunicará con sus contactos usando los medios provistos por el sistema.</p> <p>La comunicación podrá ser mediante un chat global, mediante mensajes internos a la aplicación o mediante correos electrónicos.</p> <p>Cada proyecto contará con un chat interno mediante el cual un usuario se podrá comunicar solo con los participantes de un proyecto.</p>
Actores	Usuario, Contacto
Prioridad	Normal

Requisitos Cubiertos	R2.2.2 Comunicarse con contacto
----------------------	---------------------------------

Caso de Uso 2.5 : Reunirse con Contacto	
Definición	<p>Un usuario podrá concertar una reunión con sus contactos.</p> <p>Un usuario puede crear una reunión e invitar a varios contactos. Estos a su vez pueden aceptar o rechazar esa invitación.</p> <p>Cada usuario tendrá un listado con las reuniones en las que aparece como asistente.</p> <p>Cada usuario tendrá un listado con las invitaciones a reuniones que tiene.</p> <p>El usuario creador de una reunión será asistente automáticamente.</p>
Actores	Usuario, Contacto
Prioridad	Normal
Requisitos Cubiertos	R2.2.3 Reunirse con contacto

Caso de Uso 2.6 : Eliminar Contacto	
Definición	<p>Este caso de uso representa la acción en la que un usuario, elimina a uno de sus contactos de su lista de contactos.</p> <p>El contacto no recibirá notificación de esta eliminación.</p> <p>El usuario no tendrá al contacto en su lista de contactos.</p> <p>El contacto no tendrá al usuario en su lista de contactos.</p> <p>El usuario podrá volver a enviar peticiones al contacto.</p> <p>El contacto podrá enviar peticiones al usuario.</p>
Actores	Usuario, Contacto
Prioridad	Normal
Requisitos Cubiertos	R2.4 Eliminar Contacto

5.2.2.4 Gestionar Proyectos

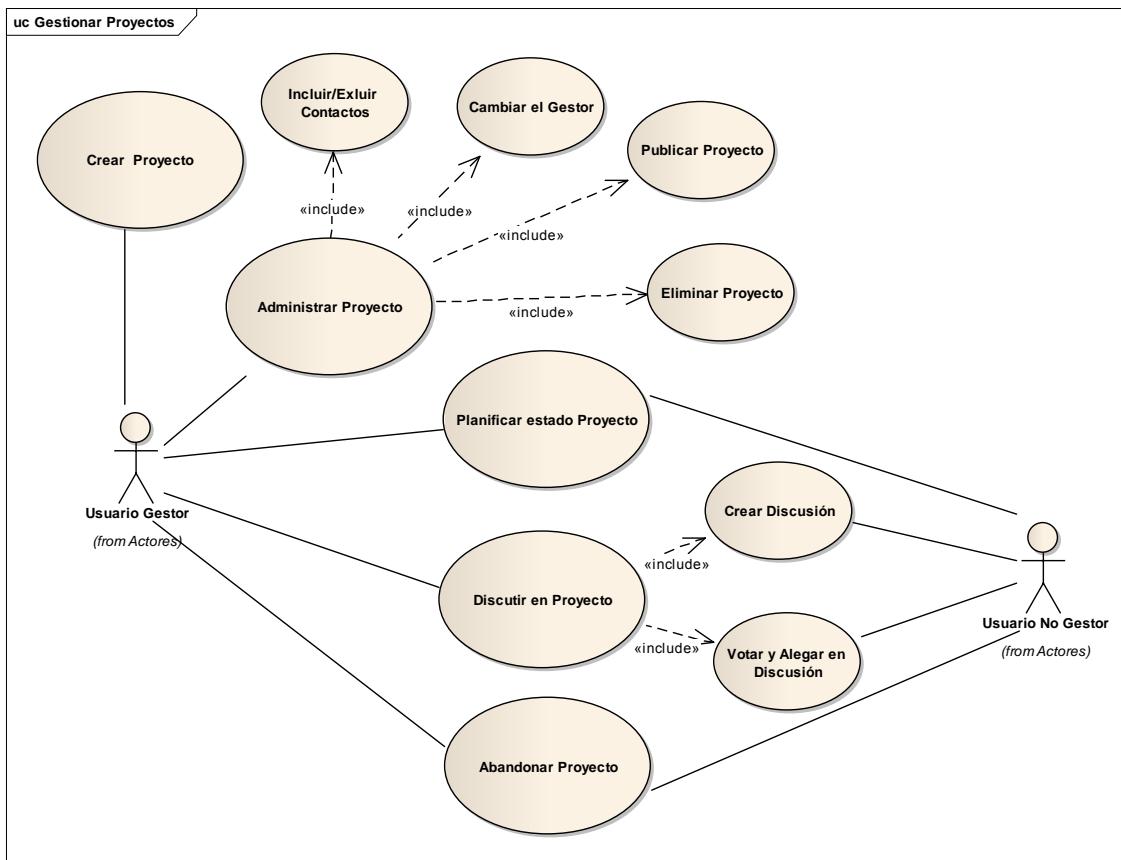


Figura 18 : Gestionar Proyectos

Caso de Uso 3.1 : Crear Proyecto	
Definición	Todos los usuarios del sistema podrán crear un proyecto en cualquier momento. Un usuario podrá crear varios proyectos y participar en otros varios. El usuario creador de un proyecto es automáticamente el gestor de dicho proyecto. Esto le confiere los derechos y deberes relativos a la administración del proyecto. El rol de gestor de proyecto puede cambiar.
Actores	Usuario gestor
Prioridad	Normal
Requisitos Cubiertos	R3.1 Crear Proyecto

Caso de Uso 3.2 : Administrar Proyecto	
Definición	Modela la administración de un proyecto, engloba una serie de posibles acciones. Abarca:

	<ul style="list-style-type: none"> • Incluir y excluir contactos • Modificar Titulo, tema y descripción. • Cambiar gestor • Publicar el Proyecto • Eliminar el proyecto <p>Estas acciones solo pueden ser realizadas por el usuario gestor.</p>
Actores	Usuario gestor
Prioridad	Normal
Requisitos Cubiertos	R3.2 Administrar Proyecto

Caso de Uso 3.2.1: Incluir y Excluir Contactos	
Definición	<p>En este caso de uso el gestor invita a algunos de sus contactos a participar en el proyecto, o bien por el contrario excluye a miembros que ya forman parte del proyecto.</p> <p>Las invitaciones les aparecerán a los contactos del gestor en su lista de invitaciones a proyecto.</p> <p>Los usuarios pueden aceptar o rechazar las invitaciones a un proyecto.</p> <p>Si las aceptan formaran parte del proyecto, si no la rechazan la invitación desaparecerá de su listado de invitaciones.</p> <p>Si un proyecto es eliminado las invitaciones a él desaparecerán de todos los listados.</p> <p>No se podrán invitar contactos que ya sean miembros del proyecto.</p> <p>Los usuarios excluidos podrán ser invitados nuevamente.</p>
Actores	Usuario gestor
Prioridad	Normal
Requisitos Cubiertos	R3.2.2 Incluir Contactos,R3.2.3 Excluir Contactos

Caso de Uso 3.2.2: Cambiar el Gestor	
Definición	El usuario gestor cede el rol de gestor a otro participante del proyecto. Esto es útil si un usuario desea abandonar el proyecto o incluso eliminarse del sistema.
Actores	Usuario gestor
Prioridad	Normal
Requisitos Cubiertos	R3.2.4 Ceder rol de gestor

Caso de Uso 3.2.3 : Publicar Proyecto	
Definición	Acción mediante la cual se consigue dar un ámbito público al proyecto. Es parte de las labores de administración por lo que lo deberá hacer el gestor del proyecto. Una vez publicado el proyecto, estará visible para los contactos de los participantes. También estará accesible desde el buscador de proyectos. Se podrá ver la información del proyecto así como los recursos que no sean borradores.
Actores	Usuario gestor
Prioridad	Normal
Requisitos Cubiertos	R3.2.5 Publicar Proyecto

Caso de Uso 3.2.4 : Eliminar Proyecto	
Definición	Modela la acción en la cual el usuario gestor de un proyecto decide eliminarlo. Solo lo podrá hacer el usuario gestor, además se le debe pedir confirmación ya que es una acción irreversible . A partir de ese momento ninguno de los miembros tendrá el proyecto en su lista de proyectos. Todas las invitaciones a este proyecto desaparecerán de los listados de invitaciones en los que aparezcan.
Actores	Usuario gestor
Prioridad	Normal
Requisitos Cubiertos	R3.2.6 Eliminar Proyecto

Caso de Uso 3.3 : Planificar estado Proyecto	
Definición	Acción mediante la cual los miembros del proyecto definen hitos sobre el calendario. A partir de estos hitos se calculará el estado esperado del proyecto. Se podrán definir hitos sobre todas las fechas

	del calendario. Se podrán mover los hitos de fecha. Se podrán eliminar y renombrar hitos. El estado esperado se recalculará. El gestor establecerá el estado real del proyecto.
Actores	Usuario gestor, Usuario no gestor
Prioridad	Normal
Requisitos Cubiertos	R3.3 Planificar estados del Proyecto

Caso de Uso 3.4 : Discutir en Proyecto

Definición	Acción mediante la cual un participante toma parte en alguna discusión como instigador o como instigado. Cualquier participante podrá crear una discusión, exponiendo un contexto y unas opciones al respecto. El resto escoge una de las opciones dejando además una alegación. Solo se podrá votar una vez.
Actores	Usuario gestor, Usuario no gestor
Prioridad	Normal
Requisitos Cubiertos	R3.4 Discutir en Proyecto

Caso de Uso 3.4.1 : Crear Discusión

Definición	Acción mediante la cual un participante toma parte en alguna discusión como instigador. Cualquier usuario participante de un proyecto podrá crear una nueva discusión. Esta discusión estará abierta hasta que el gestor la cierre. Una vez sea cerrada todos podrán ver sus resultados. El gestor podrá ver los resultados en todo momento. El gestor también podrá votar, solo una vez.
Actores	Usuario gestor, Usuario no gestor
Prioridad	Normal
Requisitos Cubiertos	R3.4 Discutir en Proyecto

Caso de Uso 3.4.2 : Participar en Discusión	
Definición	Acción mediante la cual un participante toma parte en alguna discusión como instigador. Los participantes podrán entrar a todas las discusiones, y si estas están abiertas votar, al votar se les pedirá una alegación. Si ya han votado no podrán volver a hacerlo.
Actores	Usuario gestor, Usuario no gestor
Prioridad	Normal
Requisitos Cubiertos	R3.4 Discutir en Proyecto

Caso de Uso 3.5: Abandonar Proyecto	
Definición	Modela la acción en la que un participante abandona un proyecto, puede ser gestor o no gestor. En caso de que sea el usuario gestor del proyecto, se asignara gestor a uno de los participantes aleatoriamente.
Actores	Usuario gestor, Usuario no gestor
Prioridad	Normal
Requisitos Cubiertos	R3.5 Abandonar proyecto

5.2.2.2.5 Gestionar Recursos

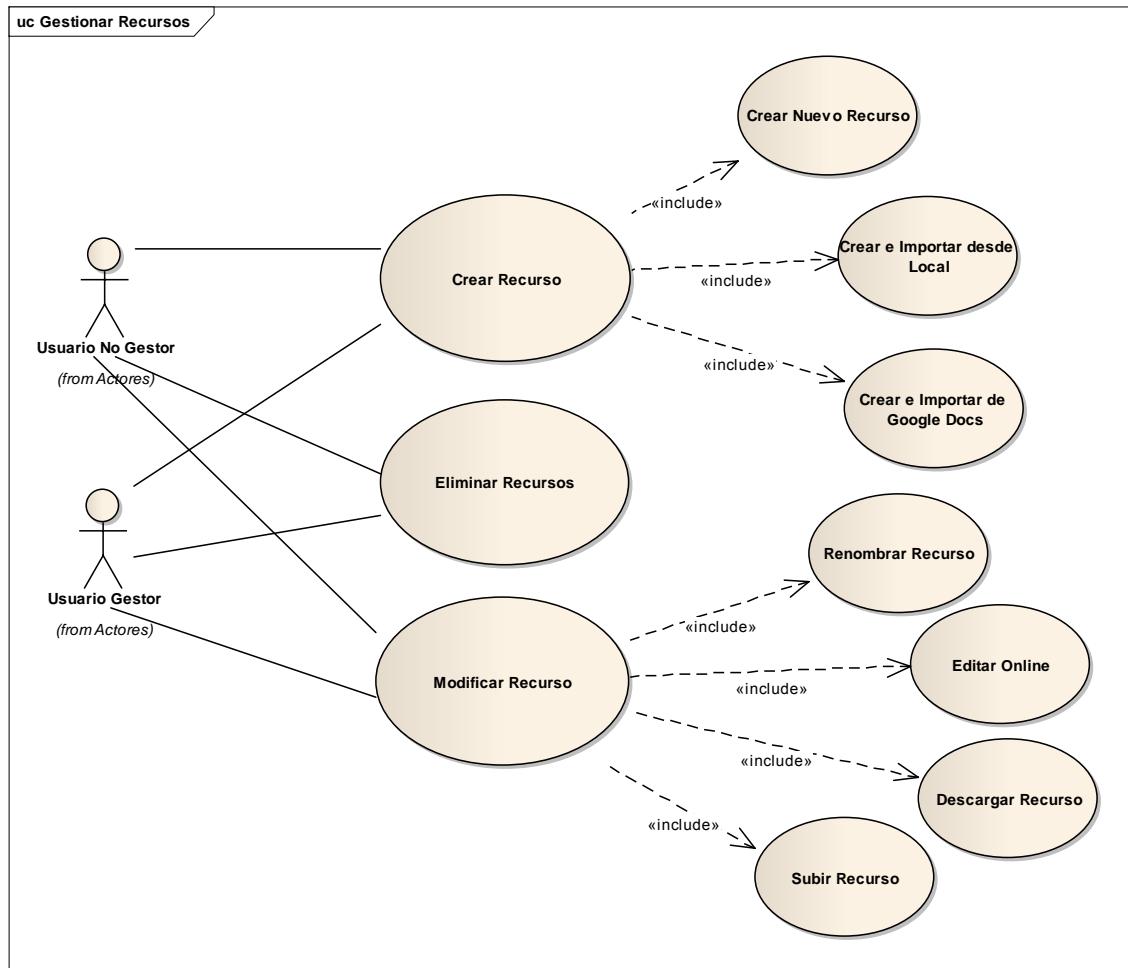


Figura 19: Cas de uso Gestionar Recursos

Caso de Uso 4.1 Crear Recurso

Definición	Modela la acción en la que un usuario participante de un proyecto crea un recurso, todos los participantes pueden crear recursos.
Actores	Usuario gestor, Usuario no gestor
Prioridad	Normal
Requisitos Cubiertos	R4.1 Crear Recurso

Caso de Uso 4.1.1 Crear Nuevo Recurso

Definición	<p>En este caso particular del caso de uso 4.1 el actor crea un recurso de la nada, en contraposición a otros sub casos en los que lo creará reutilizando contenidos de su ordenador o de Google Docs.</p> <p>Un nuevo recurso podrá ser de varios tipos:</p> <ul style="list-style-type: none"> • Tipo folder
------------	---

	<ul style="list-style-type: none"> • Tipo file • Tipo Documento • Tipo Media • Tipo proyecto de eclipse • Tipo none <p>Además dependiendo del tipo de recurso cambiarán las opciones que ofrecerá el recurso.</p>
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal
Requisitos Cubiertos	R4.1.1 Crear nuevo Recurso

Caso de Uso 4.1.2 Crear e Importar desde Local	
Definición	<p>Al igual que en el caso anterior se crea un recurso, pero en este caso el recurso se crea a partir de algún contenido del pc del usuario participante.</p> <p>En este caso se podrán crear recursos de cualquiera de los tipos enumerados anteriormente.</p> <p>La restricción es que según el tipo de recurso que se esté creando se podrán incluir unos contenidos u otros del pc del usuario creador.</p>
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal
Requisitos Cubiertos	R4.1.2 Importar Recurso Local

Caso de Uso 4.1.3 Crear e Importar de Google Docs	
Definición	<p>Al crear un recurso, podemos importarlo desde los que tenemos en Google Docs.</p> <p>El sistema nos mostrará los recursos que tenemos en Google Docs, con la cuenta que hemos aportado al sistema y usamos para autenticarnos.</p>
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal
Requisitos Cubiertos	R4.1.3 Importar recurso de Google Docs

Caso de Uso 4.2 Modificar Recurso	
Definición	Define la modificación de un recurso por parte de los usuarios del sistema.
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal
Requisitos Cubiertos	R4.2 Modificar Recurso

Caso de Uso 4.2.1 Editar Online	
Definición	Modela la acción de editar un recurso online.

	Esto solo se podrá hacer sobre algunos tipos de recursos.
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal
Requisitos Cubiertos	R4.2.1 Editar Contenido Online

Caso de Uso 4.2.2 Renombrar Recurso	
Definición	Modela la acción de cambiar el nombre a un recurso. Todos los recursos tendrán esta opción.
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal
Requisitos Cubiertos	R4.2.2 Renombrar Recurso

Caso de Uso 4.2.3 Mover Recurso	
Definición	Modela la acción de mover un recurso dentro del árbol de recursos. Todos los recursos tendrán una opción para ser movidos, el usuario seleccionará dicha opción que le pedirá la nueva ubicación dentro del árbol. Solo serán ubicaciones validas recursos contenedores.
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal
Requisitos Cubiertos	R4.2.3 Mover Recurso

Caso de Uso 4.2.4 Descargar Recurso	
Definición	Define la acción mediante la cual un usuario se descarga un recurso, con el fin de modificarlo en local. Cada tipo de recurso tendrá una representación distinta al descargarse, en todos los casos una representación relacionada con el tipo de contenido al que abstrae el recurso.
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal
Requisitos Cubiertos	R4.2.4

Caso de Uso 4.2.5 Subir Recurso	
Definición	En este caso el usuario selecciona uno de los recursos, y de este la opción de subir. Esta opción le permitirá subir contenido de su ordenador.
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal

Requisitos Cubiertos	R4.2.5 Subir Recurso
----------------------	----------------------

Caso de Uso 4.3 Eliminar Recurso	
Definición	Define la eliminación de un recurso por parte de un usuario participante en el proyecto al que pertenece el recurso. Todos los recursos podrán ser eliminados. Si se trata de un recurso contenedor se eliminarán también todos los recursos contenidos.
Actores	Usuario gestor, Usuario No gestor
Prioridad	Normal
Requisitos Cubiertos	R4.3 Eliminar Recurso

5.2.2.3 Análisis de Escenarios

En este apartado analizamos los distintos escenarios que se dan en cada caso de uso, apoyándonos en diagramas de secuencia que modelan las distintas secuencias que se pueden dar en los distintos escenarios. Estos diagramas son modelos de análisis que persiguen modelar el funcionamiento del sistema con él fin de detectar funcionalidades críticas así como excepciones que se puedan dar en estas.

5.2.2.3.1 Gestión de Usuarios

Caso de uso 1.1: Añadir Usuario

Escenario 1.1.a : Añadir Usuario con éxito	
Precondiciones	El usuario debe aportar información personal y una cuenta de Google.
Postcondiciones	El usuario registrado podrá acceder al sistema sin problemas con sus credenciales de Google.
Excepciones	Que la cuenta de Google no esté activa. Que ya exista un usuario dado de alta con esa cuenta de Google. En ambos casos se le informará y se le pedirá que ingrese otra. Que haya un error de conexión durante la importación de contactos. En este caso se le dará a elegir al usuario entre seguir sin importar o abandonar el proceso para realizarlo luego.
Iniciado por	Usuario Anónimo
Finalizado por	Usuario Anónimo
Descripción	El usuario anónimo llega a la pantalla de registro, rellena un formulario con su información personal y su cuenta de Google, si todo va bien realiza el proceso de importación de contactos y después irá directamente a la pantalla de inicio de la aplicación.
Corresponde al Requisito:	R1.1 Añadir Usuario

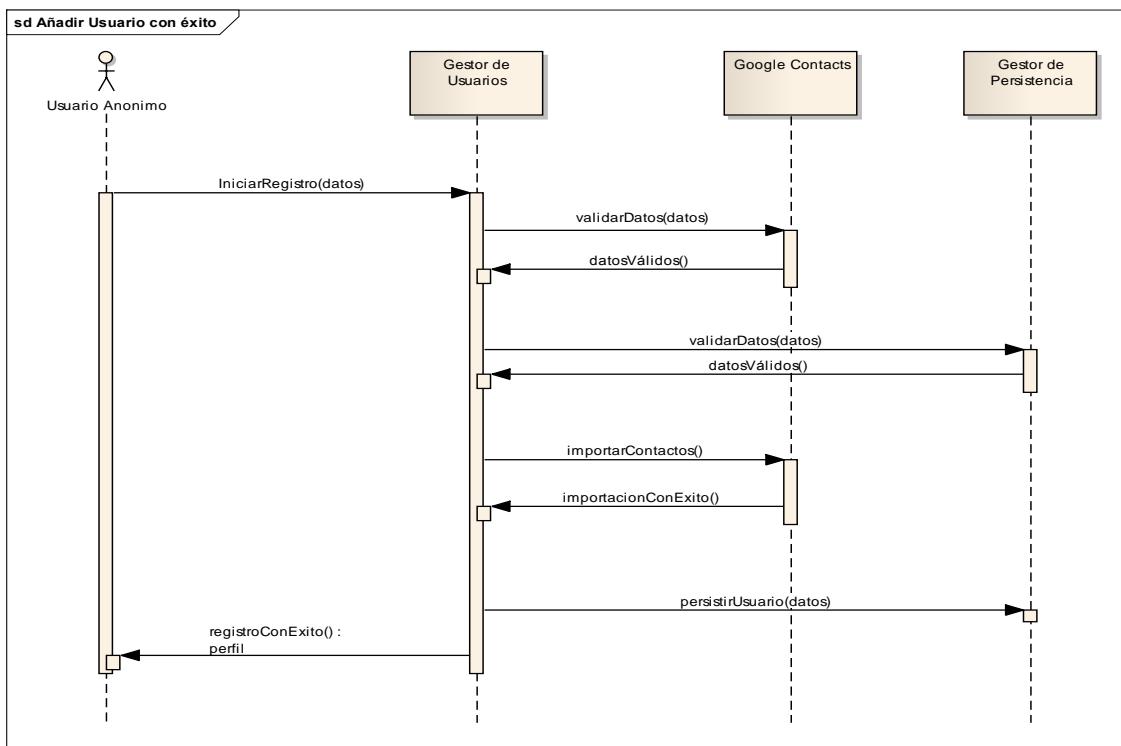


Figura 20: Añadir usuario con éxito

Escenario 1.1.b : Añadir Usuario sin éxito – Cuenta de Google Inválida	
Precondiciones	El usuario debe aportar información personal y una cuenta de Google.
Postcondiciones	Se le mostrará al usuario un mensaje de error.
Excepciones	Ninguna
Iniciado por	Usuario Anónimo
Finalizado por	Usuario Anónimo
Descripción	Durante el proceso de registro se comprueba que la cuenta de Google que el usuario ha aportado no es inválida, y se le da un error.
Corresponde al Requisito:	R1.1 Añadir Usuario

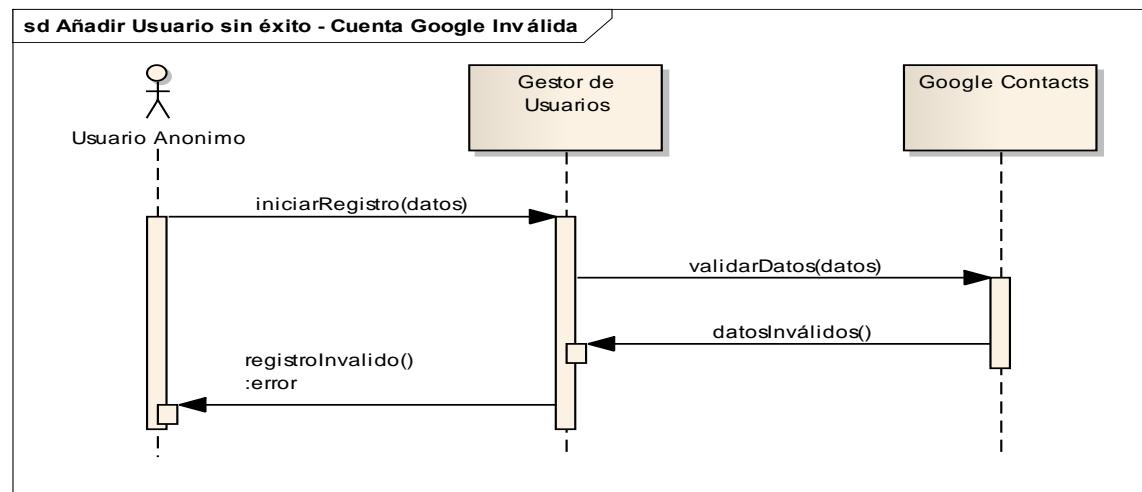


Figura 21: Añadir usuario sin éxito cuenta de Google invalida

Escenario 1.1.c : Añadir Usuario sin éxito - Cuenta de Google repetida

Precondiciones	El usuario debe aportar información personal y una cuenta de Google.
Postcondiciones	Se le mostrará al usuario un mensaje de error.
Excepciones	Ninguna
Iniciado por	Usuario Anónimo
Finalizado por	Usuario Anónimo
Descripción	Durante el proceso de registro después de validar la cuenta de Google aportada por el usuario, se comprueba que el usuario ya existe en el sistema con esa cuenta.
Corresponde al Requisito:	R1.1 Añadir Usuario

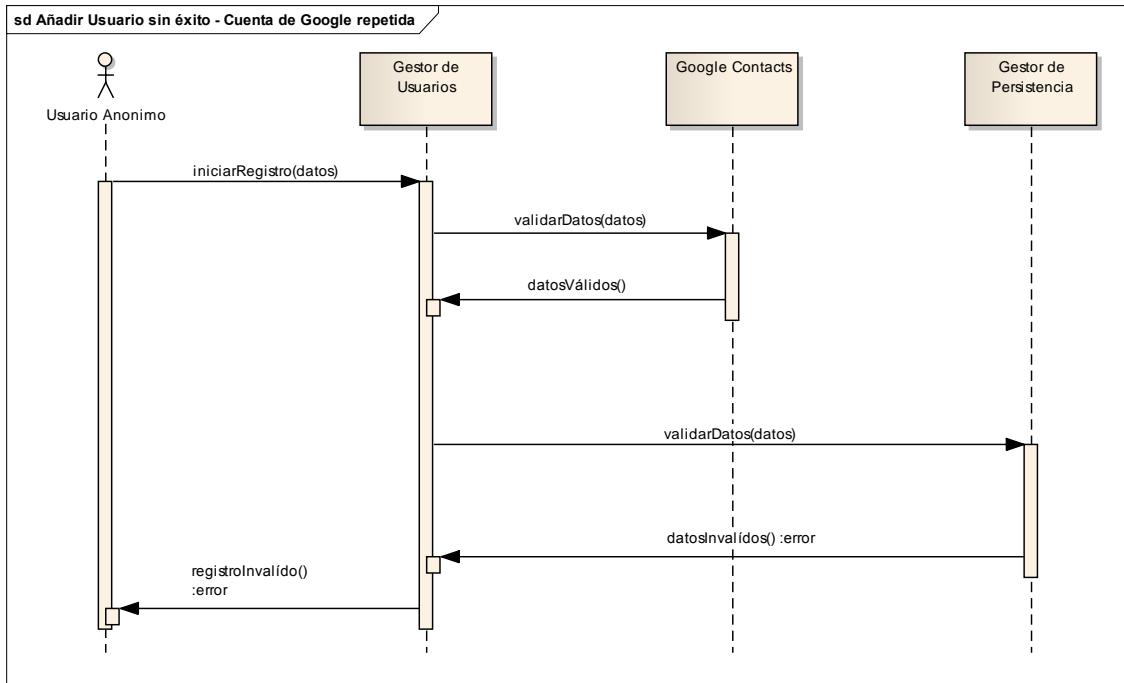


Figura 22: Añadir usuario sin éxito cuenta de Google repetida

Caso de uso 1.2: Modificar Usuario

Escenario 1.2.a : Modificar Usuario con éxito	
Precondiciones	Se deben introducir los nuevos datos, sin dejar ninguno en blanco, (si con el valor anterior).
Postcondiciones	La información del usuario debe quedar modificada en el sistema.
Excepciones	<ul style="list-style-type: none"> -Que el modificador meta algún dato en blanco. En este caso se mostrara un mensaje de error sin perder los datos que son buenos. Que el modificador introduzca una cuenta de Google de Incorrecta. En este se le mostrara un mensaje de error grave diciéndole que la cuenta no es válida o ya está en uso.
Iniciado por	Usuario
Finalizado por	Usuario
Descripción	<p>El modificador inicia el proceso de modificación, este proceso muestra los datos existentes previamente.</p> <p>Entonces el modificador modifica algunos o todos y acepta.</p>
Corresponde al Requisito:	R1.3 Modificar Usuarios

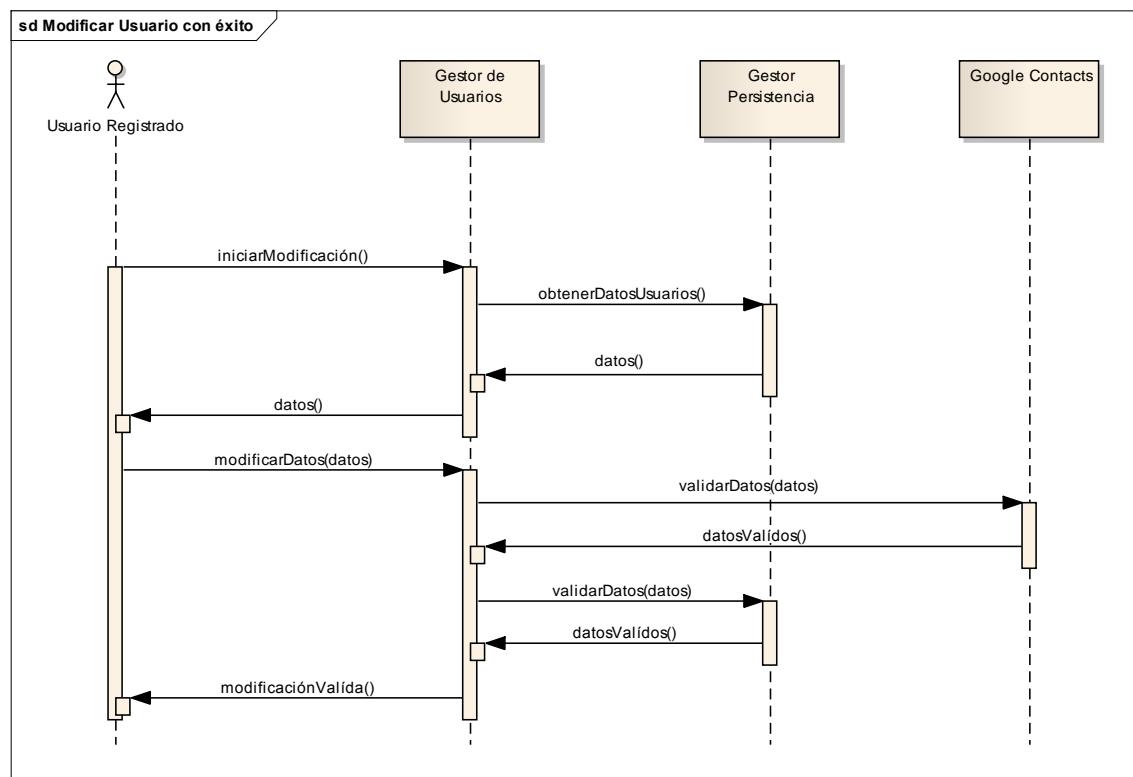


Figura 23: Modificar usuario con éxito

Escenario 1.2.b : Modificar Usuario sin éxito cuenta de Google Inválida

Precondiciones	Se deben introducir los nuevos datos, sin dejar ninguno en blanco, (si con el valor anterior).
Postcondiciones	El usuario recibirá un mensaje de error, que le comunicara que la cuenta es inválida.
Excepciones	Que el modificador meta algún dato en blanco. En este caso se mostrara un mensaje de error sin perder los datos que son buenos.
Iniciado por	Usuario
Finalizado por	Usuario
Descripción	El modificador inicia el proceso de modificación, este proceso muestra los datos existentes previamente. Entonces el modificador intenta modificar algunos o todos y recibe un mensaje de error.
Corresponde al Requisito:	R1.3 Modificar Usuarios

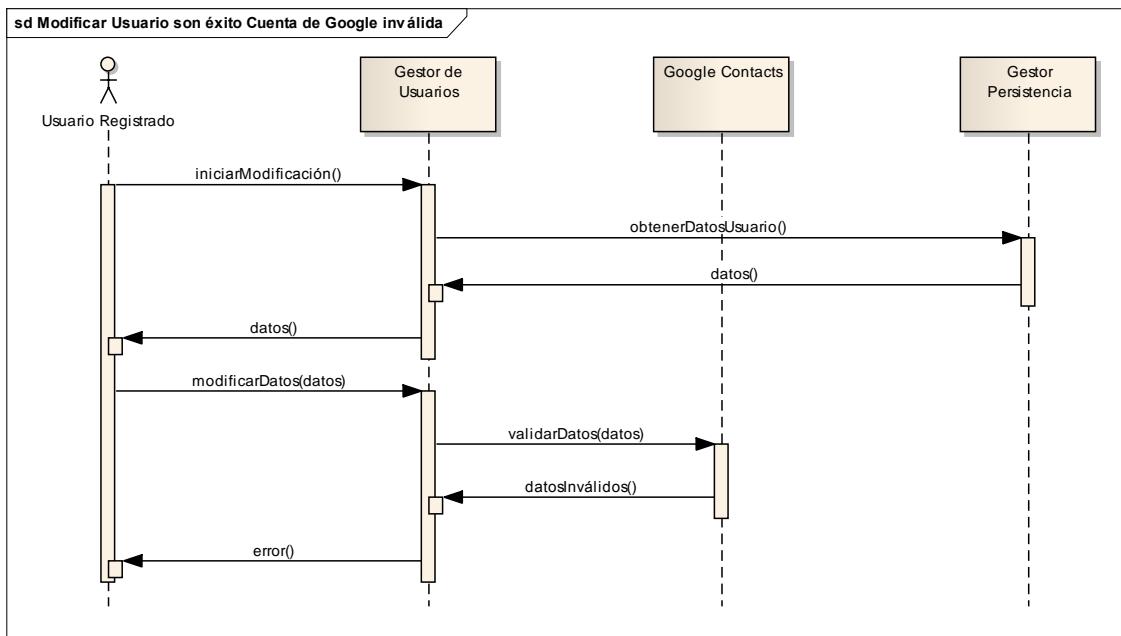


Figura 24: Modificar usuario sin éxito cuenta de Google Inválida

Escenario 1.2.bc : Modificar Usuario sin éxito cuenta de Google repetida	
Precondiciones	Se deben introducir los nuevos datos, sin dejar ninguno en blanco, (si con el valor anterior).
Postcondiciones	El usuario recibirá un mensaje de error, que le comunicara que la cuenta ya está en el sistema.
Excepciones	-Que el modificador meta algún dato en blanco. En este caso se mostrara un mensaje de error sin perder los datos que son buenos.
Iniciado por	Usuario
Finalizado por	Usuario
Descripción	El modificador inicia el proceso de modificación, este proceso muestra los datos existentes previamente. Entonces el modificador intenta modificar alguno o todos y recibe una mensaje de error.
Corresponde al Requisito:	R1.3 Modificar Usuarios

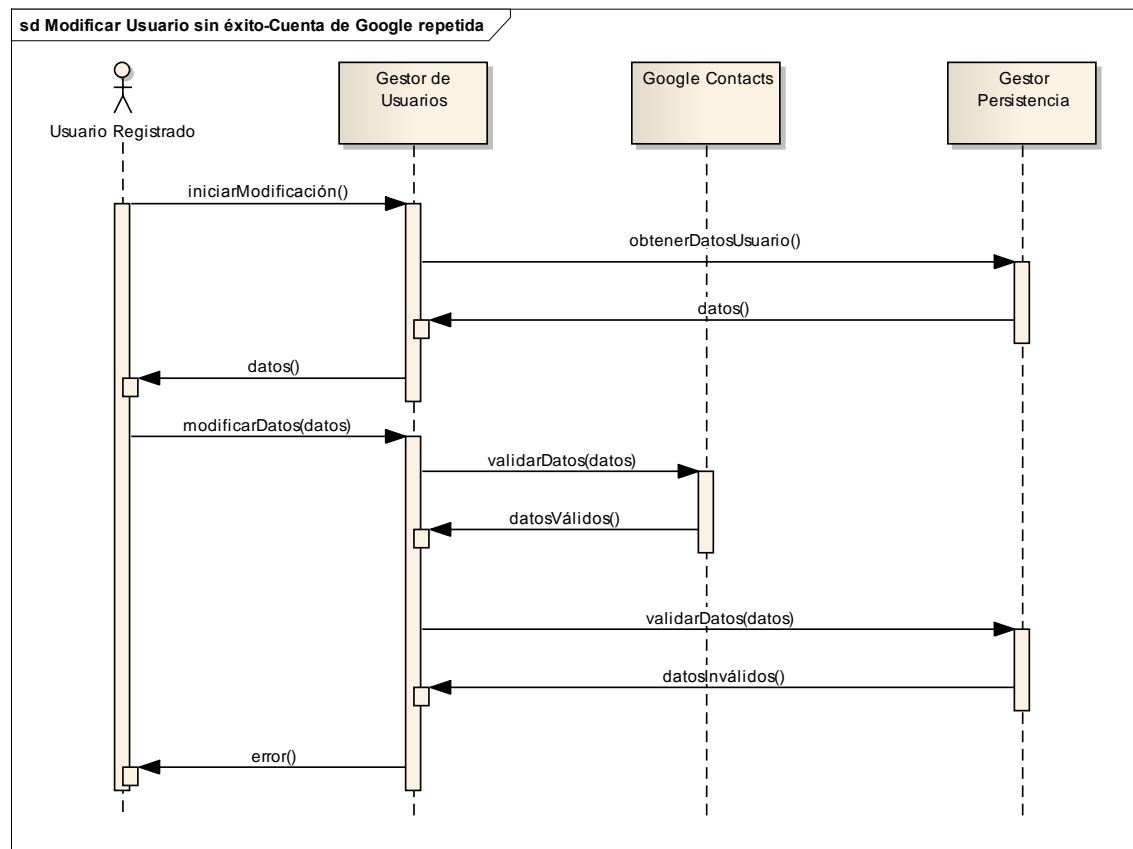


Figura 25 : Modificar usuario sin éxito cuenta de Google repetida

Caso de uso 1.3: Eliminar Usuario

Escenario 1.3.a : Eliminar Usuario con éxito	
Precondiciones	El usuario a eliminar debe pertenecer al sistema.
Postcondiciones	El usuario quedara eliminado del sistema.
Excepciones	-Mientras el administrador esta eliminando a un usuario, este está en el sistema. En ese caso en cuanto el usuario refresque su navegador ira a la pantalla inicial, que no lo dejara entrar. -El usuario eliminado era participante de un proyecto. En ese caso el proyecto no debe sufrir cambios lamentables a menos que el usuario fuera el único, en cuyo caso desaparecerá.
Iniciado por	Usuario, Administrador
Finalizado por	Usuario, Administrador
Descripción	El usuario decide eliminarse, va a la sección de administración y se auto elimina, previa confirmación. Cabe la posibilidad que la eliminación la realice el administrador desde su aplicación de administración.

	También puede ser que el sistema auto elimine al usuario, después de un periodo con su cuenta en funcionamiento parcial por haber perdido su cuenta de Google.
Corresponde al Requisito:	R1.2 Eliminar Usuario

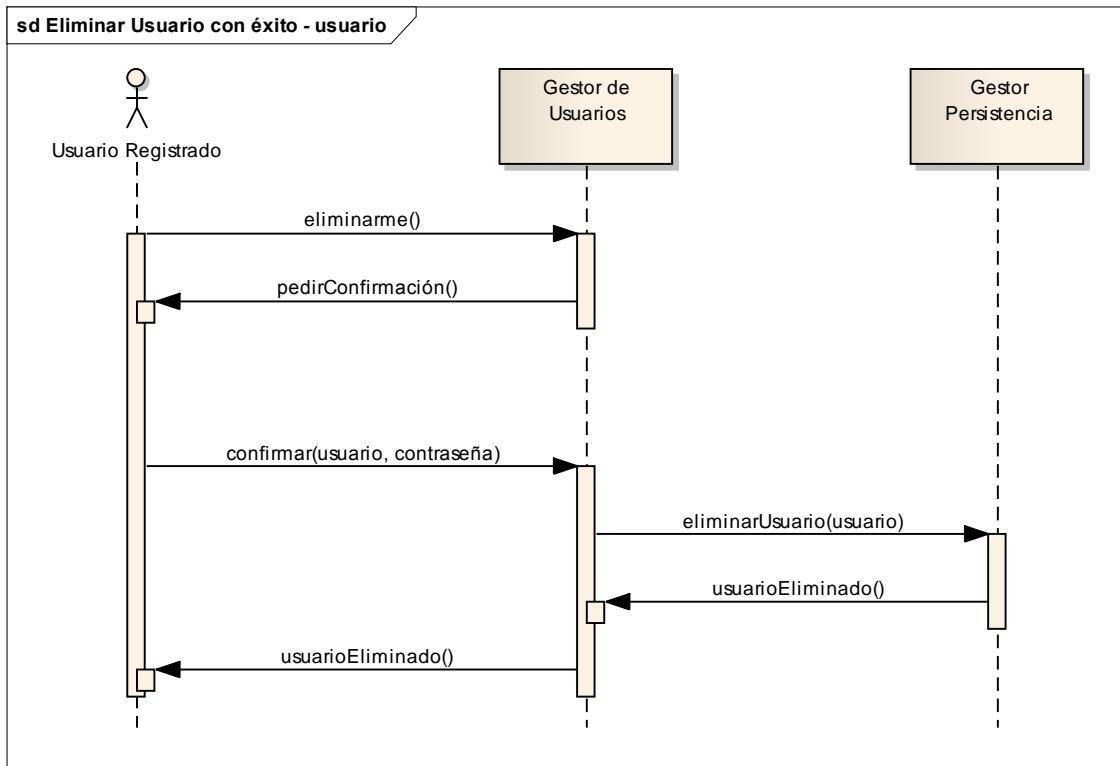


Figura 26: Eliminar usuario con éxito

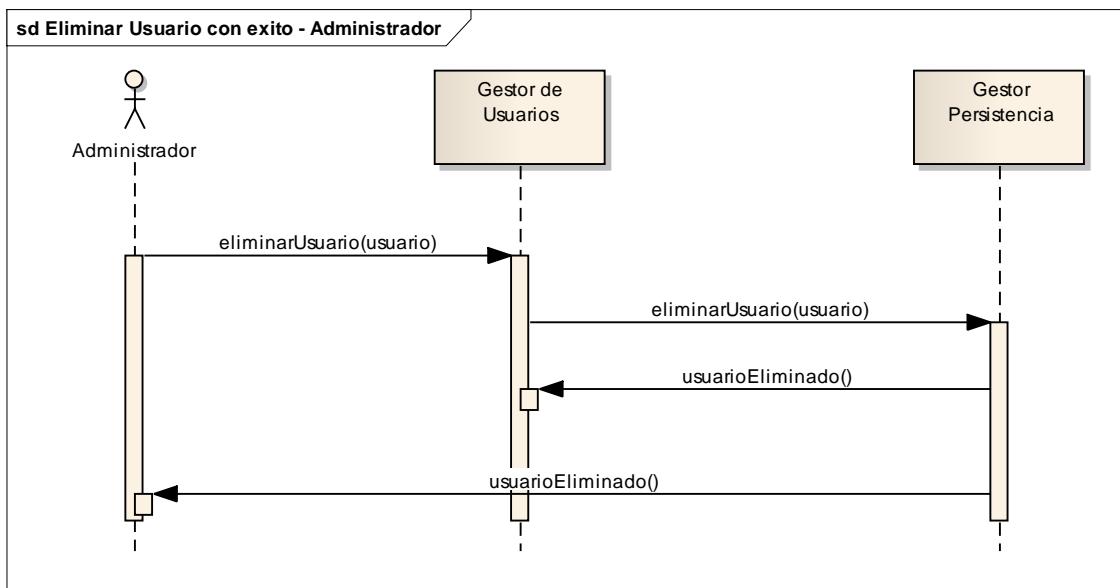


Figura 27: Eliminar usuario con éxito – Administrador

5.1.1.1 Gestionar Contactos

Caso de uso 2.1: Buscar Usuario

Escenario 2.1.a : Buscar Usuario con éxito	
Precondiciones	El usuario buscador debe pertenecer al sistema.
Postcondiciones	El usuario buscador debe obtener una salida (éxito) con resultados o vacía.
Excepciones	Que no se introduzcan parámetros de búsqueda. Se dará un error al usuario.
Iniciado por	Usuario
Finalizado por	Usuario
Descripción	El usuario inicia un proceso de búsqueda, con el fin de encontrar a otro usuario del sistema . Podrá recorrer la lista de sus contactos o utilizar el buscador del sistema.
Corresponde al Requisito:	R2.3 Buscar Usuarios

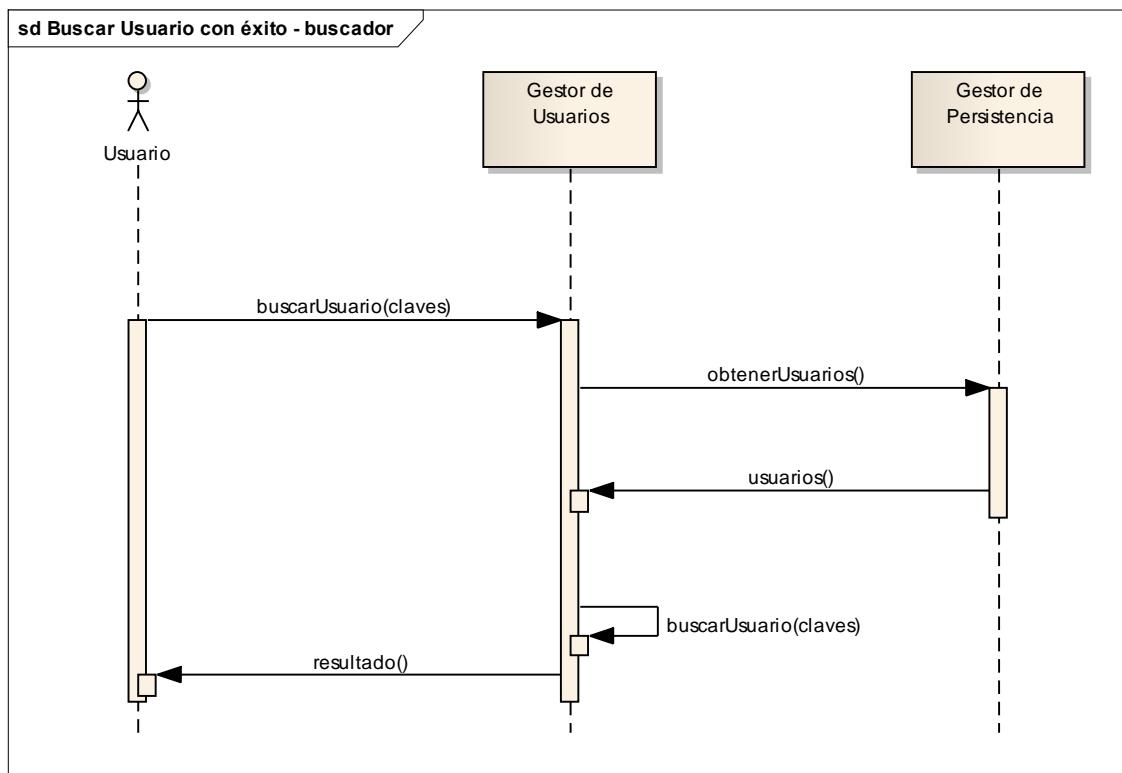


Figura 28: Buscar usuario con éxito con buscador

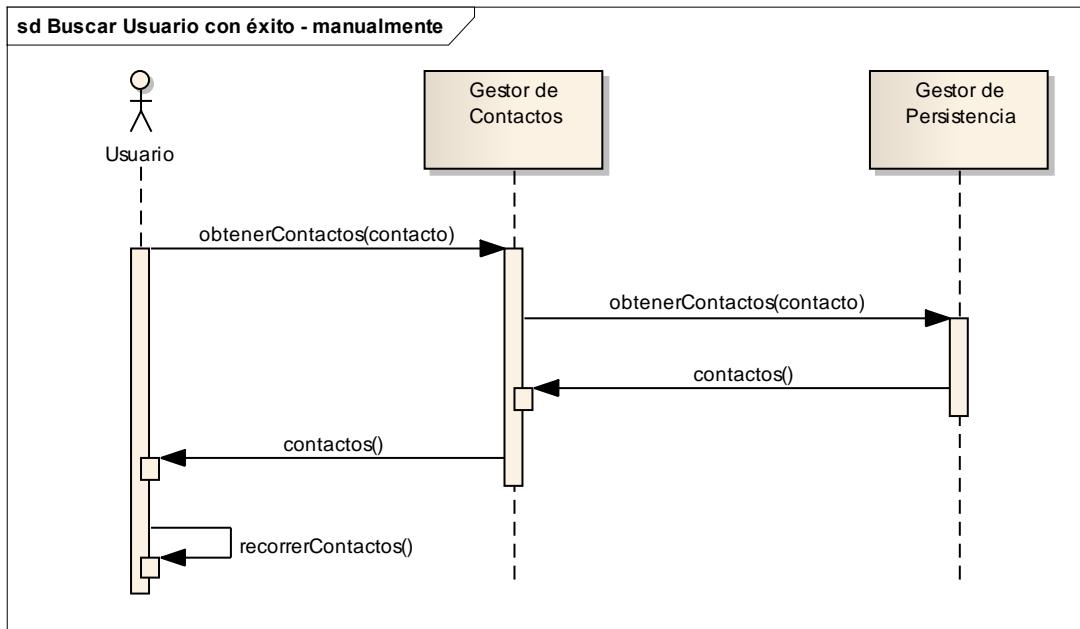


Figura 29: Buscar usuario con éxito manualmente

Caso de uso 2.2: Añadir Usuario como contacto

Escenario 2.2.a : Añadir Usuario como contacto con éxito	
Precondiciones	Tanto el usuario que añade como el añadido deben pertenecer al sistema.
Postcondiciones	El usuario a añadir debe quedar como contacto del usuario que añade.
Excepciones	En caso de cualquier error durante el proceso, se le notificará al usuario que añade que no ha podido hacer la petición.
Iniciado por	Usuario
Finalizado por	Contacto
Descripción	Un usuario le hace una petición a otro usuario, que ha localizado previamente y esta la acepta, entonces se añade a la lista de contactos y se notifica al usuario que hizo la petición.
Corresponde al Requisito:	R2.1 Añadir Usuario como contacto

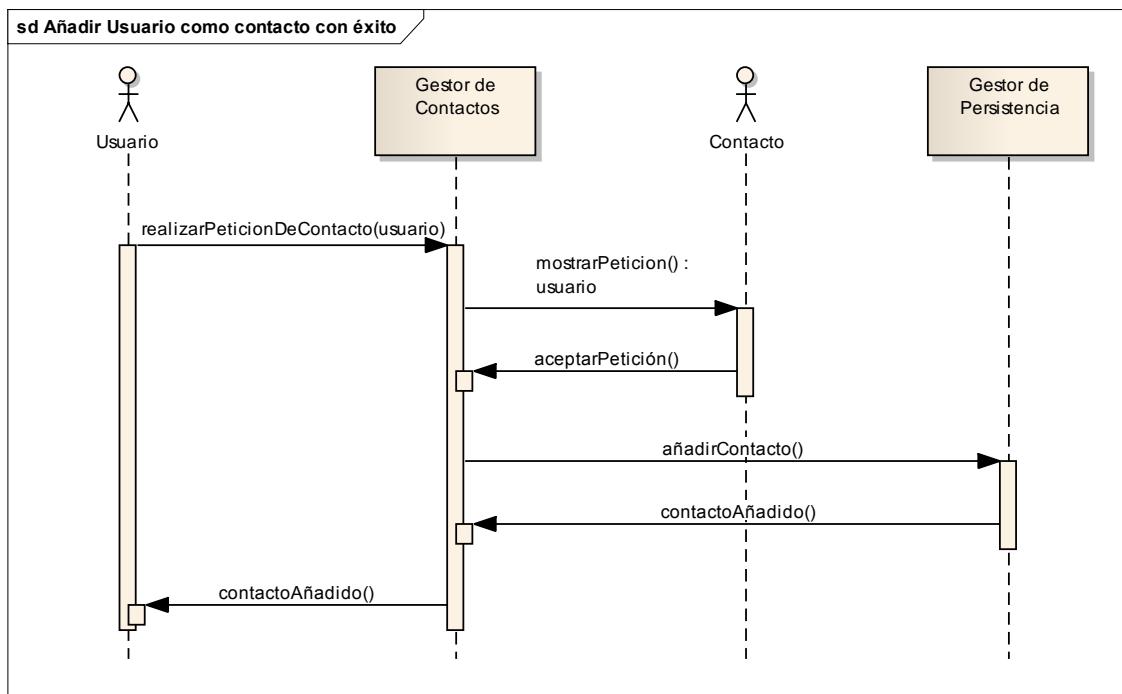


Figura 30: Añadir usuario como contacto con éxito

Escenario 2.2.b : Añadir Usuario como contacto sin éxito	
Precondiciones	Tanto el usuario que añade como el usuario a añadir deben pertenecer al sistema.
Postcondiciones	El usuario a añadir no debe quedar como contacto del usuario que añade. El usuario que añade no recibirá confirmación pero tampoco rechazo.
Excepciones	En caso de cualquier error durante el proceso, se le notificara al usuario que añade que no ha podido hacer la petición.
Iniciado por	Usuario
Finalizado por	Contacto
Descripción	Un usuario le hace una petición a otro usuario, que ha localizado previamente y este la rechaza.
Corresponde al Requisito:	R2.1 Añadir Usuario como contacto

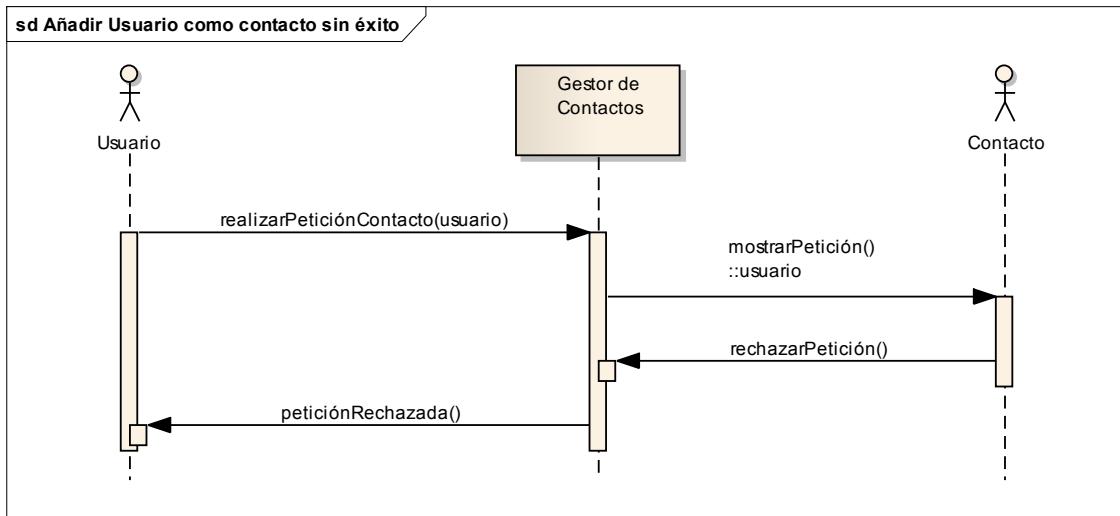


Figura 31: Añadir usuario como contacto sin éxito

Caso de uso 2.3: Ver perfil de Contacto

Escenario 2.3.a : Ver perfil de Contacto con éxito	
Precondiciones	Tanto el usuario como el contacto deben pertenecer al sistema.
Postcondiciones	El usuario debe ver la información de su contacto.
Excepciones	Que el contacto sea eliminado mientras se está viendo su información. En ese caso cuando el usuario visitante actualice ya no podrá verlo más.
Iniciado por	Usuario
Finalizado por	Usuario
Descripción	El usuario entra al área de otro usuario, donde puede ver su información personal, profesional, proyectos y contactos.
Corresponde al Requisito:	R2.2 Interactuar con Contacto

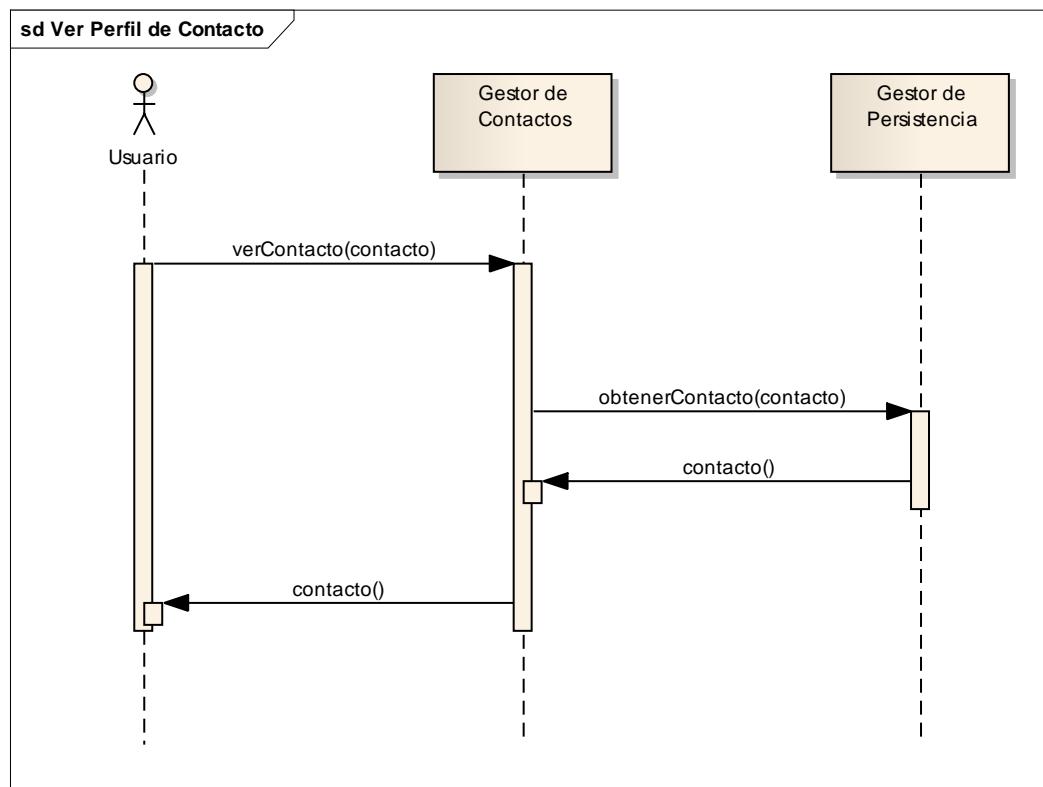


Figura 32: Ver perfil de contacto

Caso de uso 2.4: Comunicarse con contacto

Escenario 2.4.a : Comunicarse con éxito en chat general	
Precondiciones	El usuario debe pertenecer al sistema y estar dentro de este (conectado) al igual que el contacto.
Postcondiciones	Los usuarios se comunicaran mediante mensajería instantánea.
Excepciones	El servidor de mensajería instantánea deja de funcionar. En este caso el panel de chat debe quedar deshabilitado, mostrando un mensaje de error.
Iniciado por	Usuario, Contacto
Finalizado por	Usuario, Contacto
Descripción	Un usuario hace clic en un contacto del chat general y se le abre una ventana pequeña por la que podrá enviar mensajes al contacto.
Corresponde al Requisito:	R2.2 Interactuar con Contacto

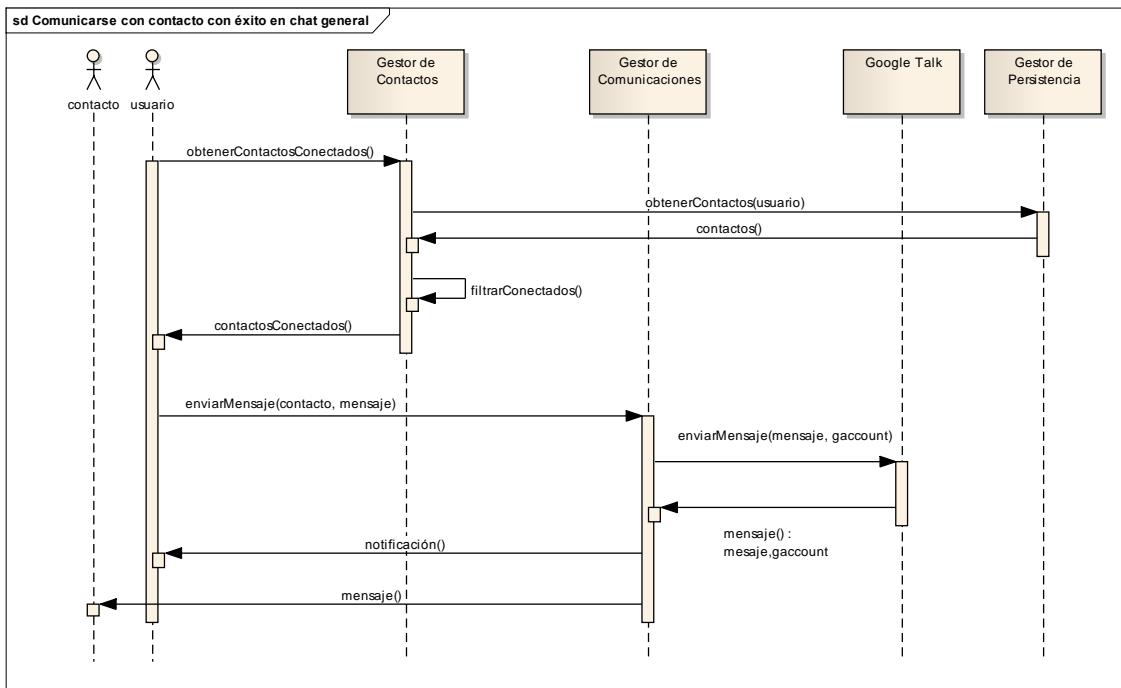


Figura 33: Comunicarse con contacto con éxito en chat general.

Escenario 2.4.b : Comunicarse con éxito en chat de proyecto	
Precondiciones	Ambos usuario debe pertenecer al sistema y estar dentro de este (conectado), además deben tener abierto el proyecto en el que trabajan juntos.
Postcondiciones	Los usuarios se comunicaran mediante mensajería instantánea.
Excepciones	El servidor de mensajería instantánea deja de funcionar. En este caso el panel de chat debe quedar deshabilitado, mostrando un mensaje de error.
Iniciado por	Usuario, Contacto
Finalizado por	Usuario, Contacto
Descripción	El usuario entra a uno de sus proyectos, allí tiene un panel con un listado de los participantes
Corresponde al Requisito:	R2.2 Interactuar con Contacto

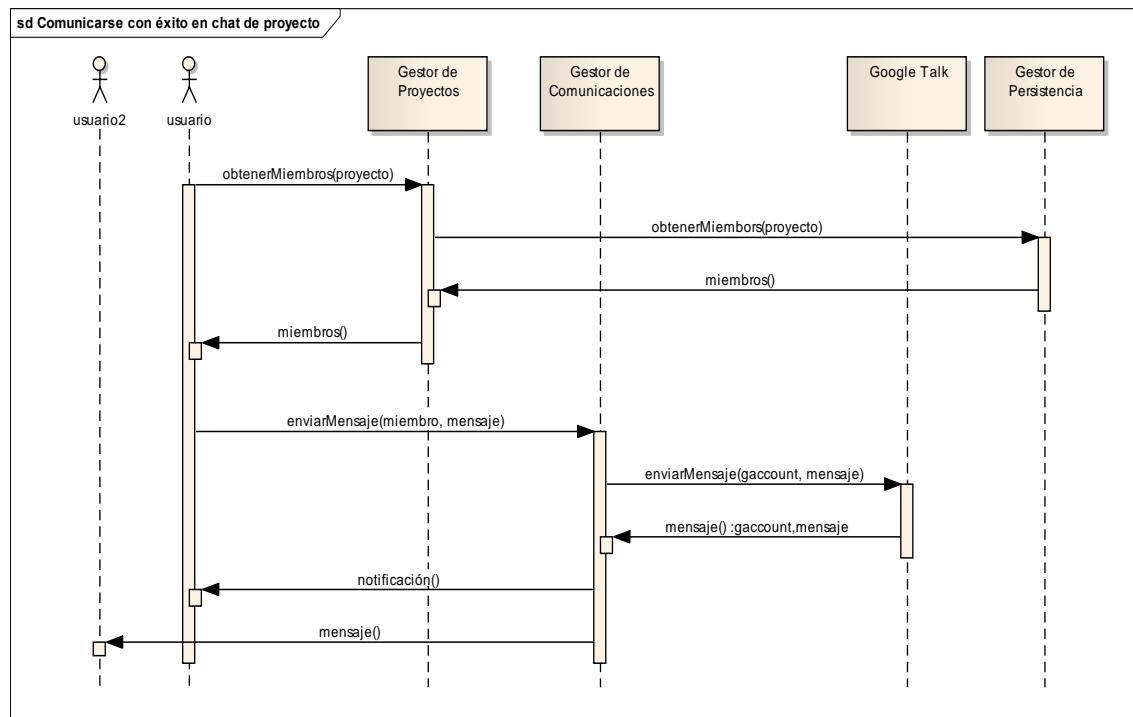


Figura 34: Comunicarse con éxito en chat de proyecto

Escenario 2.4.c: Comunicarse por mail con éxito	
Precondiciones	Tanto el usuario como él contacto deben pertenecer al sistema y tener su cuenta activa (cuenta de Google correcta).
Postcondiciones	El contacto recibirá un mail en su cuenta de Gmail.
Excepciones	El usuario o el contacto no tiene de Google en funcionamiento. En este caso se le indicara al usuario que intenta enviar el mail del problema.
Iniciado por	Usuario
Finalizado por	Contacto
Descripción	Un usuario escoge la opción de enviar un mail a un contacto, bien en el perfil del contacto o desde la sección de mensajes. Redacta el texto y lo envía indicando el contacto al que va dirigido, no una dirección de correo.
Corresponde al Requisito:	R2.2 Interactuar con Contacto

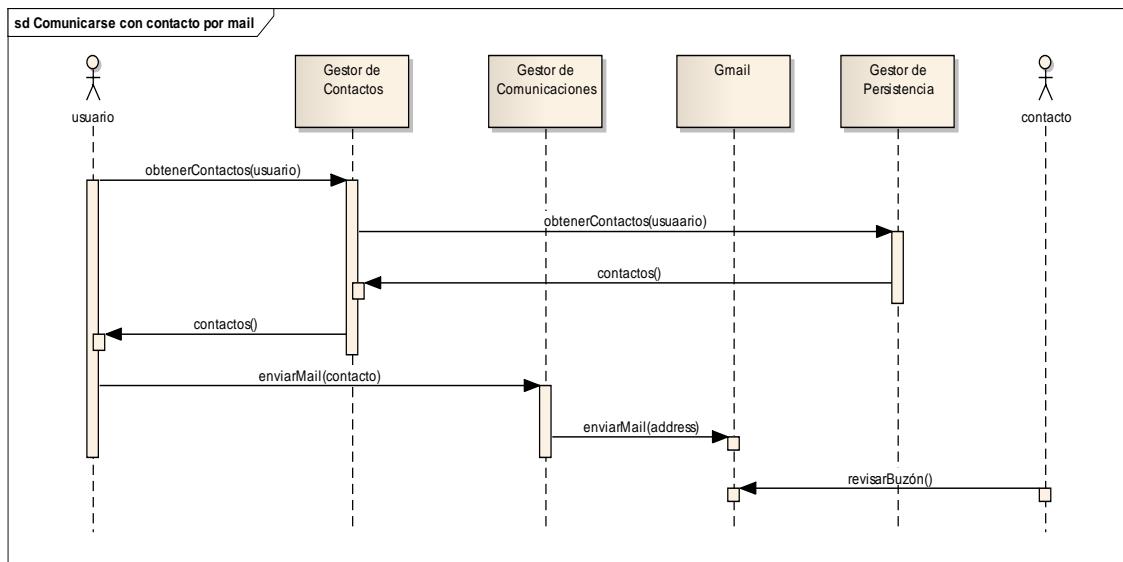


Figura 35: Comunicarse por mail con éxito

Escenario 2.4.d: Comunicarse por mensajes con éxito	
Precondiciones	El usuario debe pertenecer al sistema.
Postcondiciones	El contacto recibirá un mensaje en su bandeja de mensajes.
Excepciones	-
Iniciado por	Usuario
Finalizado por	Contacto
Descripción	El usuario escoge la opción de enviar mensaje desde el área del contacto o desde la sección de mensajes, escribe el texto e indica el contacto.
Corresponde al Requisito:	R2.2 Interactuar con Contacto

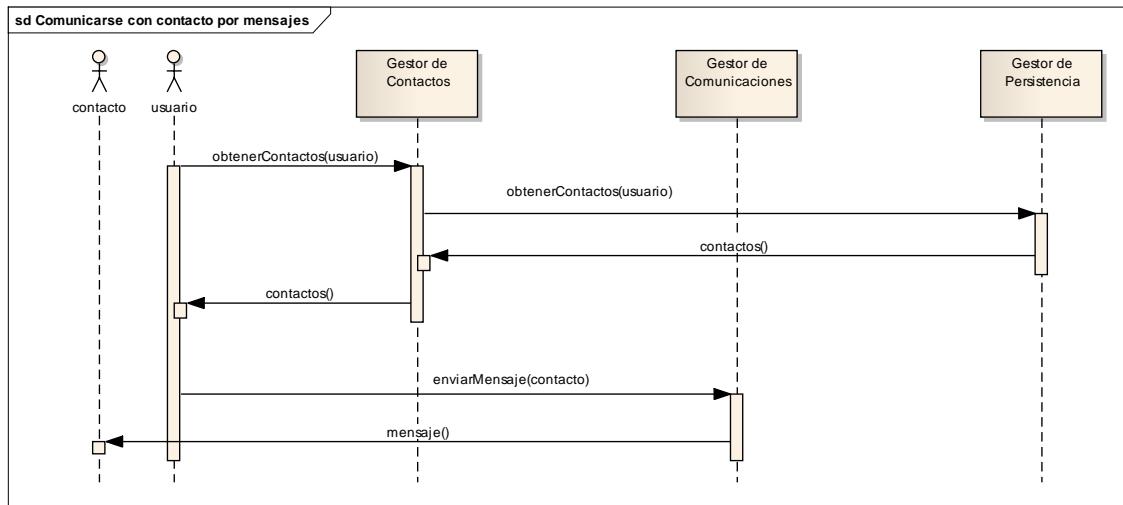


Figura 36: Comunicarse con contacto por mensajes

Caso de uso 2.5: Eliminar Contacto

Escenario 2.5.a : Eliminar Contacto con éxito	
Precondiciones	El contacto a eliminar debe estar en el sistema y en la lista de contactos del usuario.
Postcondiciones	El contacto no debe estar en la lista de contactos del usuario una vez eliminado. Se debe notificar a ambos.
Excepciones	El contacto desaparece del sistema al tiempo que lo estoy eliminando. En este caso el sistema me notificara que el usuario ha sido eliminado del sistema.
Iniciado por	Usuario
Finalizado por	Usuario, Contacto
Descripción	-El usuario selecciona a uno de sus contactos y lo elimina de su lista de contactos.
Corresponde al Requisito:	R2.4 Eliminar Contacto

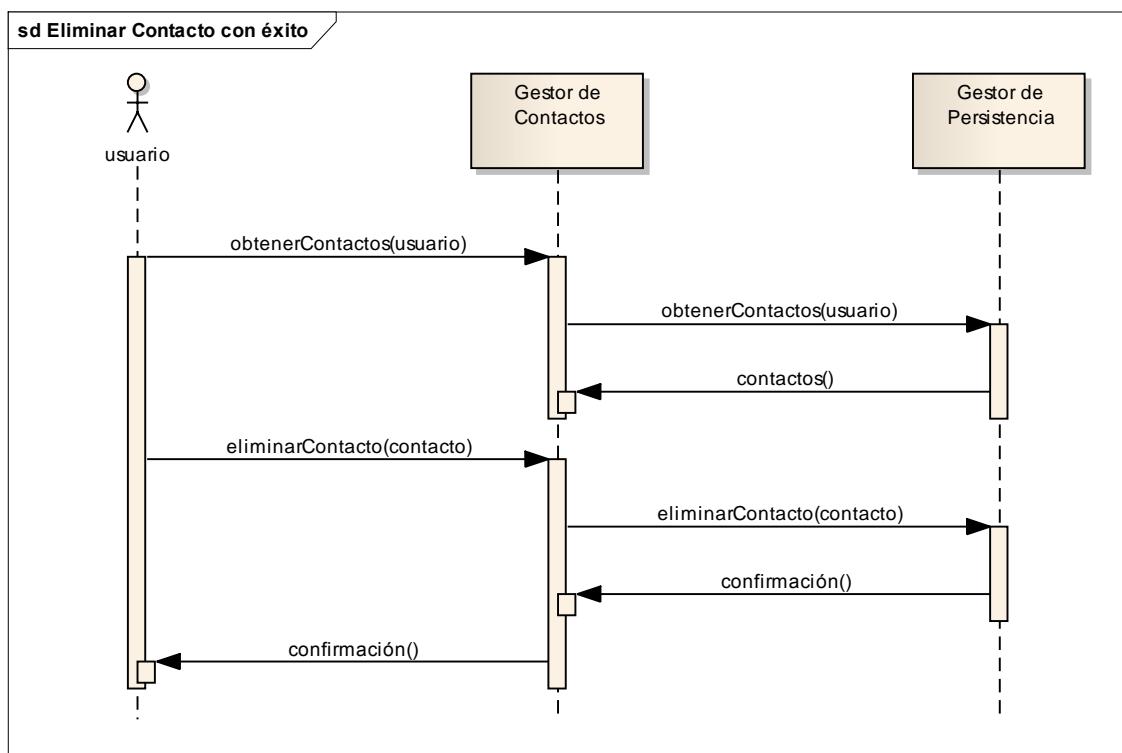


Figura 37: Eliminar contacto con éxito

5.1.1.2 Gestionar Proyectos

Caso de uso 3.1: Crear Proyecto

Escenario 3.1.a : Crear Proyecto con éxito	
Precondiciones	El usuario debe pertenecer al sistema y tener su cuenta de Google activa.
Postcondiciones	Al terminar el usuario debe tener un proyecto nuevo con los participantes que haya añadido en el momento de la creación de haberlo hecho.
Excepciones	El usuario invita a un contacto que es borrado concurrentemente. En este caso el sistema le notificará.
Iniciado por	Usuario gestor
Finalizado por	Usuario gestor
Descripción	Un usuario decide crear un nuevo proyecto, aporta la información de este. Opcionalmente puede invitar contactos en esta fase.
Corresponde al Requisito:	R3.1 Crear Proyecto

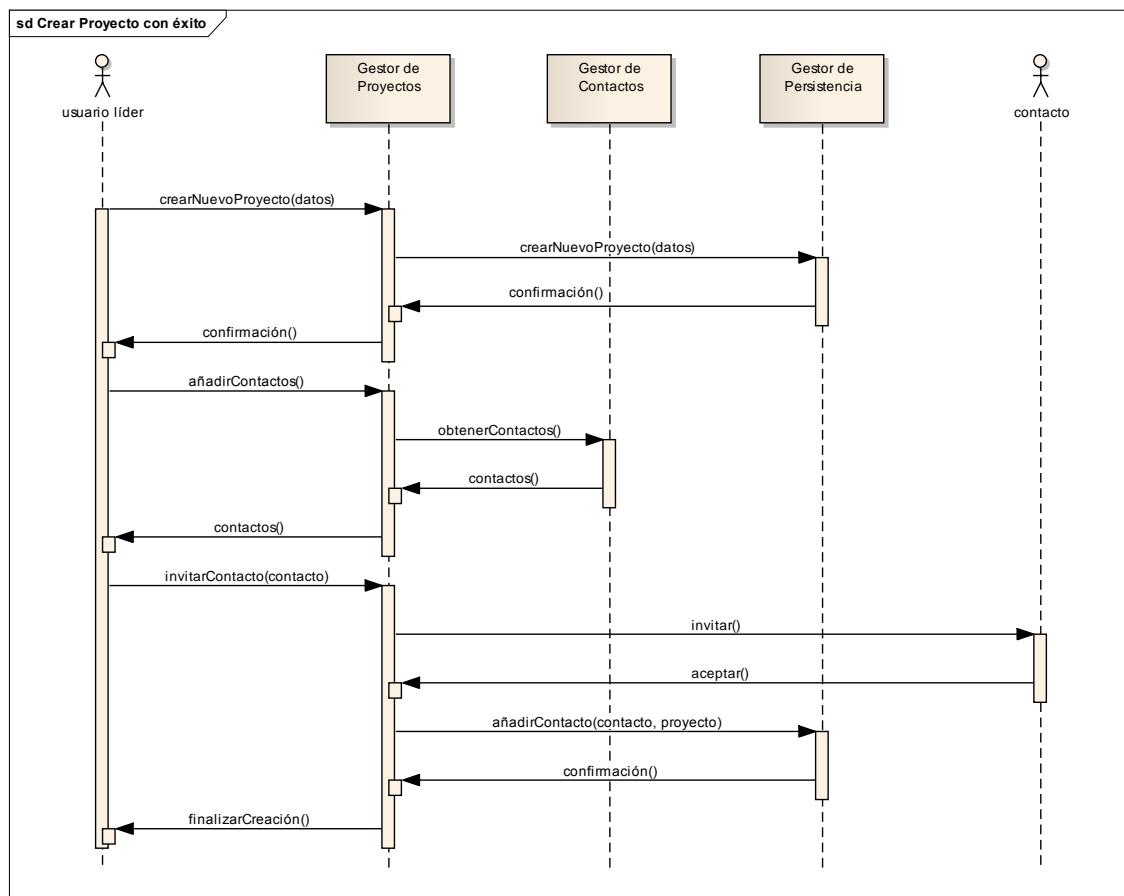


Figura 38: Crear proyecto con éxito

Caso de uso 3.2.1: Incluir/ Excluir contactos

Escenario 3.2.1.a : Incluir contactos con éxito	
Precondiciones	Los usuarios a incluir deben ser contactos del gestor del proyecto, por defecto el creador. Por supuesto deben estar en el sistema, y tener su cuenta activa.
Postcondiciones	Los contactos quedarán como participantes del proyecto.
Excepciones	Se intenta invitar a un participante, y este es eliminado del sistema, después de recibir la invitación. Se dará una respuesta negativa.
Iniciado por	Usuario gestor
Finalizado por	Usuario no gestor
Descripción	El usuario gestor va a la sección de administración del proyecto y administra los participantes de este. Selecciona incluir contactos, entonces escogerá de entre la lista de sus contactos, a los que añadir, para esto tendrá un límite. En este escenario los contactos aceptan y pasan a ser miembros del proyecto.
Corresponde al Requisito:	R3.2.2 Incluir contactos

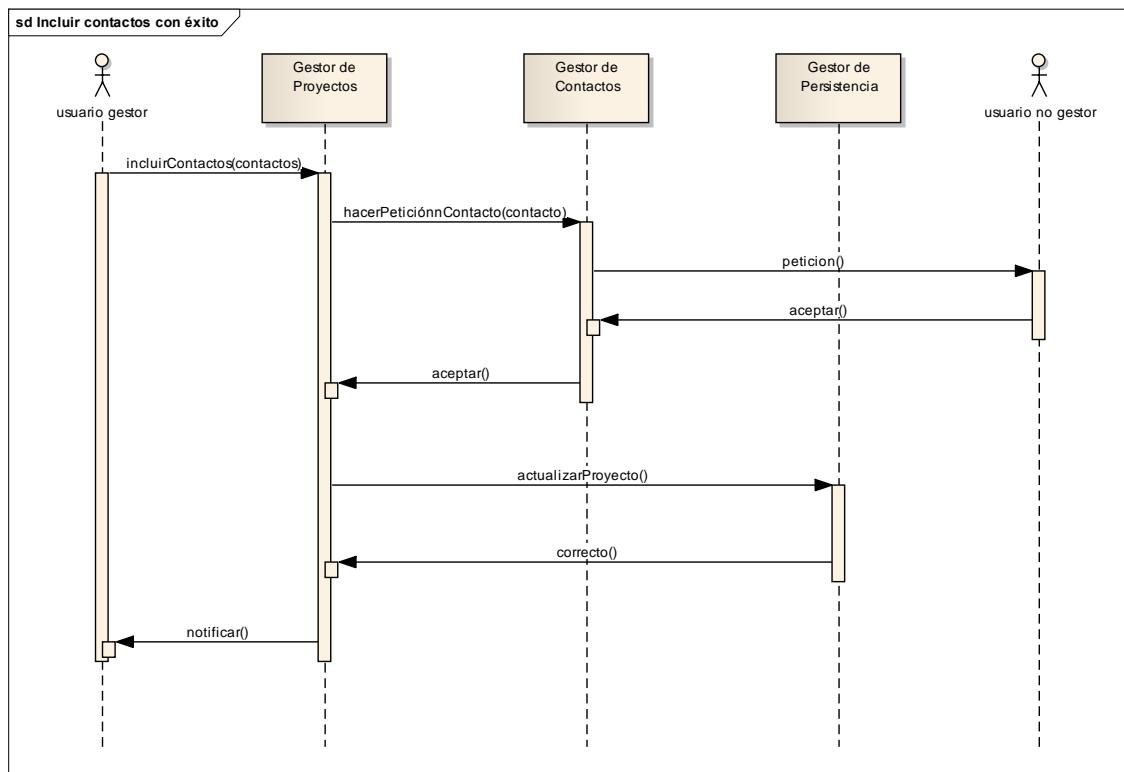


Figura 39: Incluir contactos con éxito

Escenario 3.2.1.b : Incluir contactos sin éxito	
Precondiciones	Los usuarios a incluir deben ser contactos del gestor del proyecto. Deben estar en el sistema, y tener su cuenta activa.
Postcondiciones	Los contactos no quedarán como participantes del proyecto y el gestor recibirá notificación de su rechazo. Con lo que se le desbloqueara el límite de contactos a añadir si estuviera bloqueado.
Excepciones	Se intenta invitar a un participante, y este es eliminado del sistema, después de recibir la invitación. Se dará una respuesta negativa.
Iniciado por	Usuario gestor
Finalizado por	Usuario no gestor
Descripción	El usuario gestor va a la sección de administración del proyecto y administra los participantes de este. Selecciona incluir contactos, entonces escogerá de entre la lista de sus contactos, a los que añadir, para esto tendrá un límite. En este escenario los contactos rechazan y el gestor recibe notificación de esto.
Corresponde al Requisito:	R3.2.3 Excluir contactos

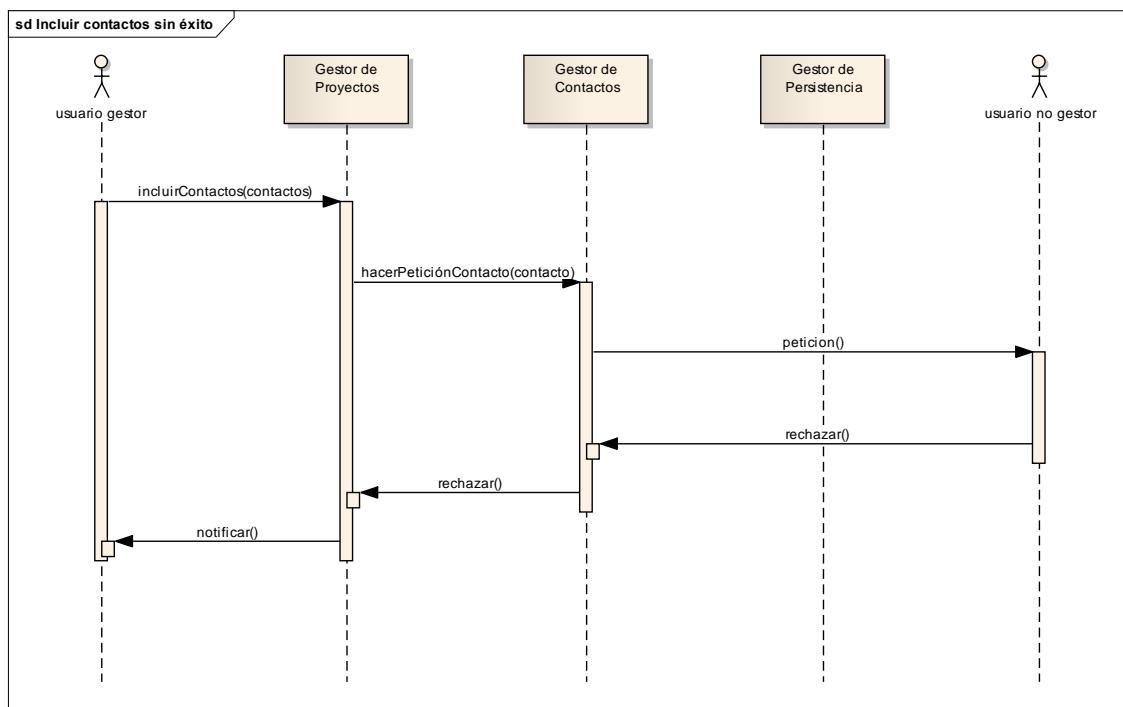


Figura 40: Incluir contactos sin éxito

Escenario 3.2.1.c : Excluir contactos con éxito	
Precondiciones	Los usuarios a excluir deben ser miembros del proyecto.
Postcondiciones	Los contactos quedarán excluidos del proyecto.
Excepciones	Se intenta excluir a un usuario al tiempo que este abandona el proyecto. El sistema debe avisar de que el usuario ya ha marchado.
Iniciado por	Usuario gestor
Finalizado por	Usuario gestor
Descripción	El usuario gestor va a la sección de administración del proyecto y administra los participantes de este. Selecciona excluir contactos, entonces escogerá de entre la lista de participantes a los que desea excluir. La exclusión será inmediata, esto desbloqueara el límite de participantes si estuviera bloqueado. Se notificará a los usuarios de su exclusión.
Corresponde al Requisito:	R3.2.2 Incluir contactos

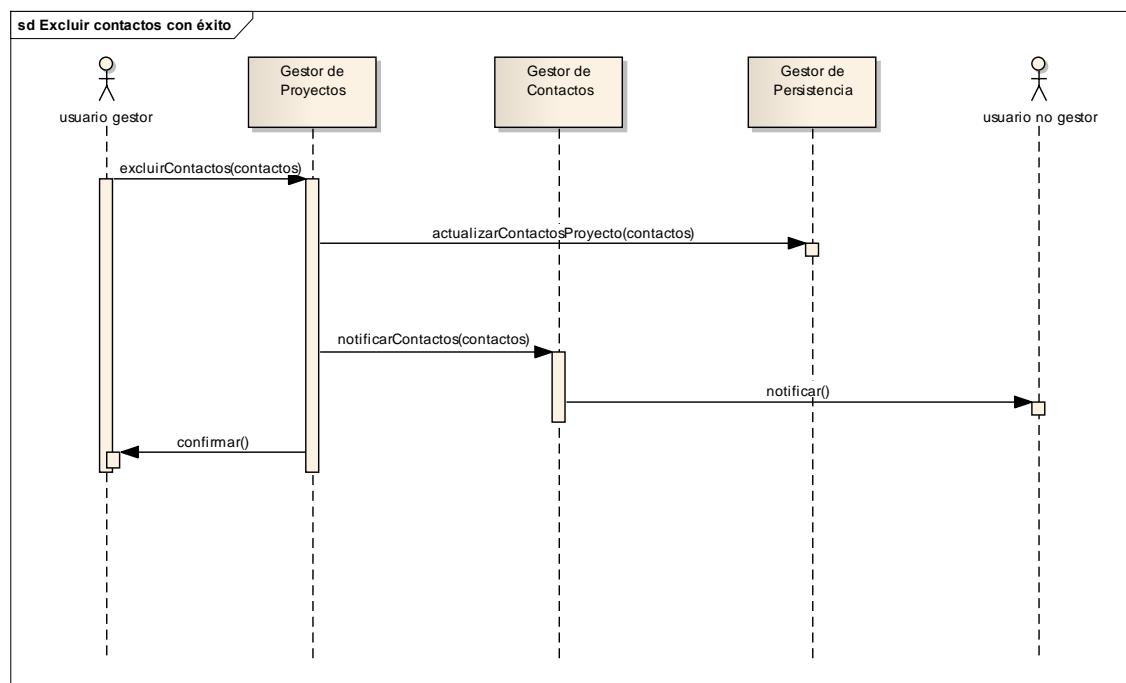


Figura 41: Excluir Contacts con éxito

Caso de uso 3.2.2: Cambiar Gestor

Escenario 3.2.2a : Cambiar Gestor con éxito	
Precondiciones	Tanto el gestor actual como el nuevo gestor deben ser miembros del proyecto.
Postcondiciones	El gestor antiguo no podrá acceder a la sección de administración del proyecto.
Excepciones	El participante elegido para ser el nuevo gestor, abandona el proyecto al tiempo que se le cede la gestión. En este caso se notificara y se impedirá el cambio.
Iniciado por	Usuario gestor
Finalizado por	Usuario gestor
Descripción	El usuario gestor decide ceder la gestión a otro participante. Va a la sección de administración y realiza esta operación.
Corresponde al Requisito:	R3.2.4 Ceder posición de gestor

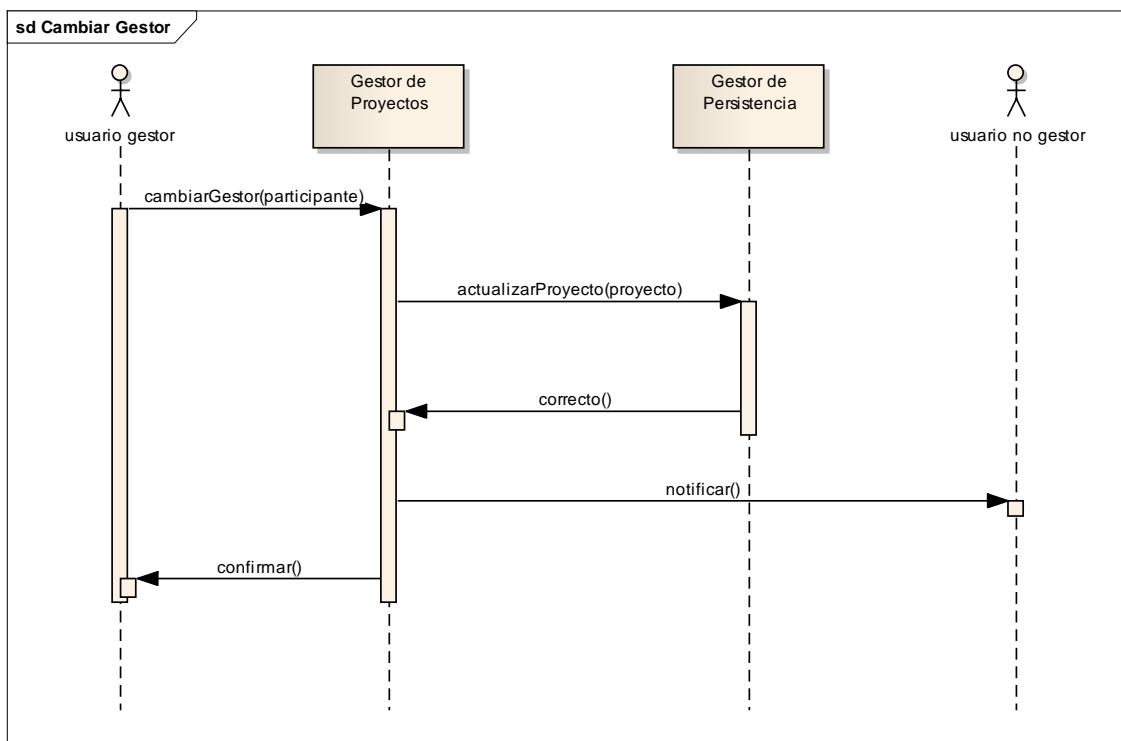


Figura 42: Cambiar gestor con éxito

Caso de uso 3.2.3: Publicar Proyecto

Escenario 3.2.3a : Publicar Proyecto con éxito	
Precondiciones	El publicador debe ser el gestor del proyecto.
Postcondiciones	El proyecto pasara al ámbito público, esto significa que todos lo podrán ver.
Excepciones	-
Iniciado por	Usuario gestor
Finalizado por	Usuario gestor
Descripción	<p>El usuario gestor desde la sección de administración, publica el proyecto, se le pedirá confirmación antes de hacer público el proyecto.</p> <p>Se notificará a todos los participantes, así como a sus contactos.</p>
Corresponde al Requisito:	R3.2.5 Publicar Proyecto

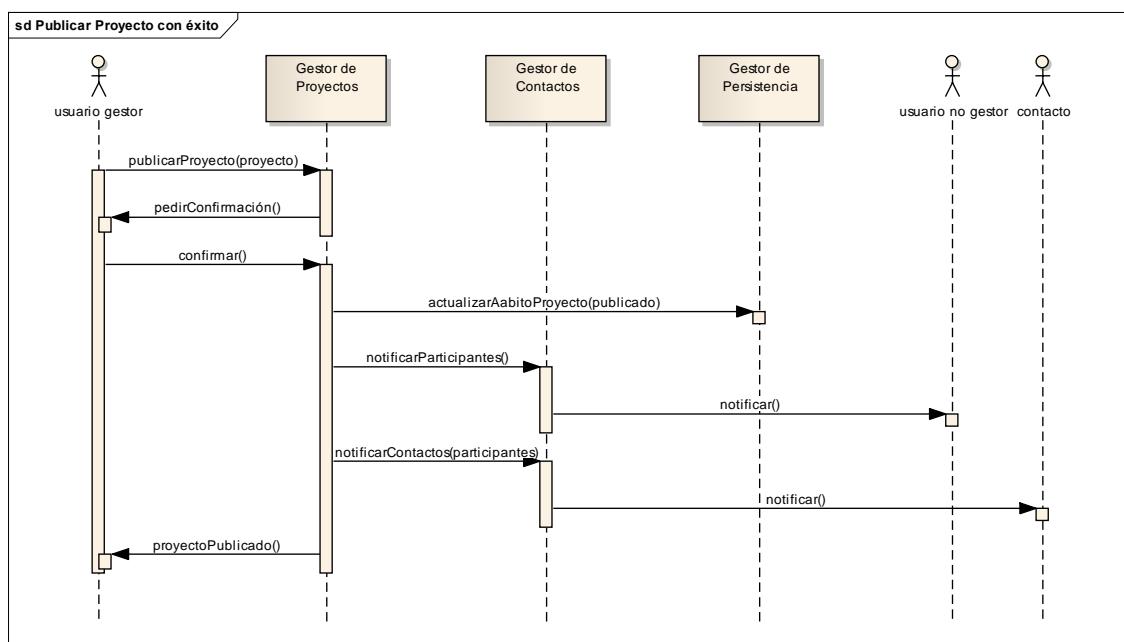


Figura 43: Publicar proyecto con éxito

Caso de uso 3.2.4: Eliminar Proyecto

Escenario 3.2.4a : Eliminar Proyecto con éxito	
Precondiciones	El borrador debe ser el usuario gestor del proyecto.
Postcondiciones	El proyecto desaparecerá para siempre.
Excepciones	El proyecto tiene varios participantes. En este caso el sistema notificara a los participantes, pero se eliminará igualmente el proyecto.
Iniciado por	Usuario gestor
Finalizado por	Usuario gestor

Descripción	El usuario gestor desde la sección se administración del proyecto selecciona la opción de eliminar el proyecto. El sistema le pedirá confirmación antes de hacer esto. Se notificará a todos los participantes del proyecto.
Corresponde al Requisito:	R3.2.6 Eliminar Proyecto

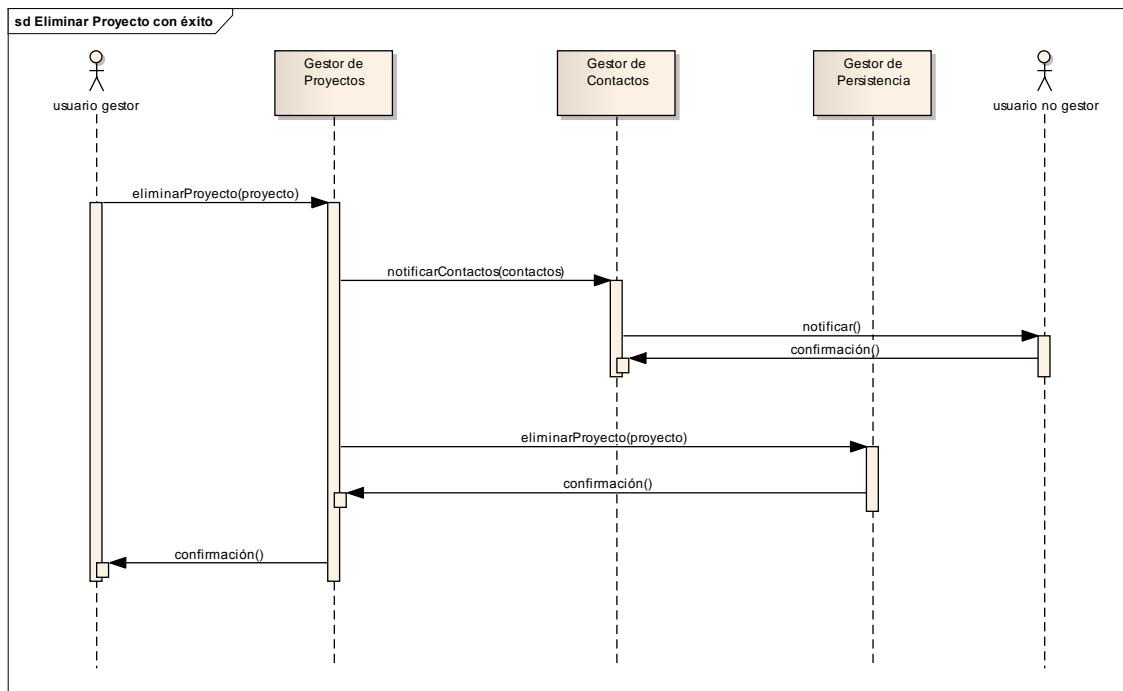


Figura 44: Eliminar proyecto con éxito

Caso de uso 3.3: Planificar Proyecto

Escenario 3.3.a : Planificar Proyecto con éxito	
Precondiciones	El planificador debe ser el gestor del proyecto. Los visualizadores de la planificación deben ser participantes del proyecto.
Postcondiciones	Una vez planificado el proyecto se actualizarán sus estados esperado y real.
Excepciones	-
Iniciado por	Usuario gestor
Finalizado por	Usuario gestor
Descripción	El usuario gestor va a la planificación del proyecto
Corresponde al Requisito:	R3.3 Planificar Estados del Proyecto

Caso de uso 3.4: Discutir en Proyecto

Escenario 3.4.1.a : Crear Discusión con éxito	
Precondiciones	Todos los participantes de una discusión deben ser miembros del proyecto.
Postcondiciones	Habrá una nueva discusión.
Excepciones	Ya existe una discusión con ese nombre en el sistema. En ese caso se le dará el error al usuario y se le pedirá otro nombre.
Iniciado por	Usuario gestor, Usuario no gestor
Finalizado por	Usuario gestor, Usuario no gestor
Descripción	Un usuario participante del proyecto decide crear una nueva discusión. Va a la sección de discusiones y crea una nueva discusión.
Corresponde al Requisito:	R3.4 Discutir en Proyecto

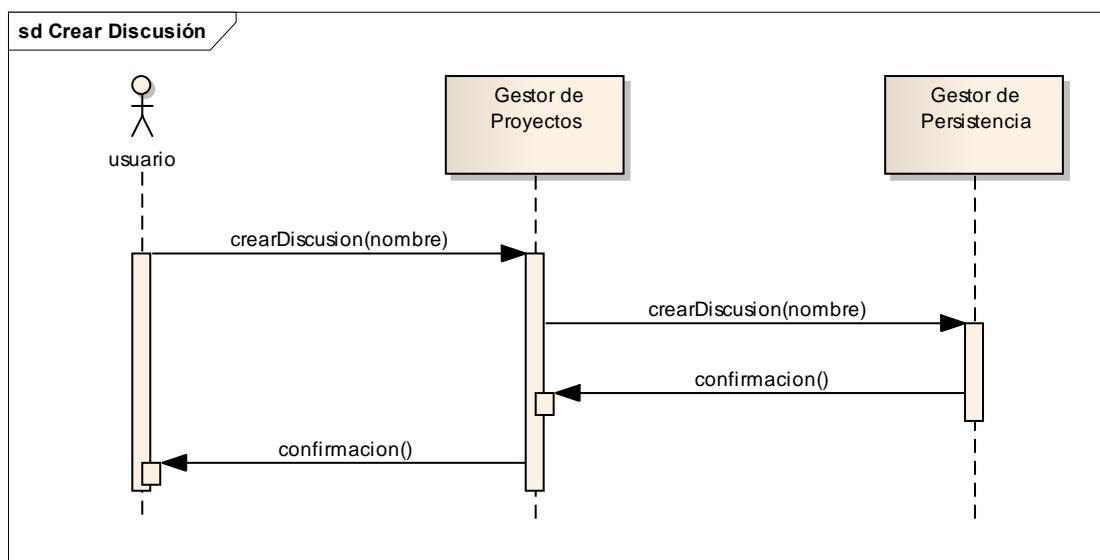


Figura 45: Crear discusión con éxito

Escenario 3.4.2.a : Votar en Discusión con éxito	
Precondiciones	Todos los participantes de una discusión deben ser miembros del proyecto. La discusión debe estar abierta y el usuario no haber votado.
Postcondiciones	Habrá un nuevo voto y una nueva alegación en la discusión.
Excepciones	El usuario en cuestión ya ha votado. En ese caso no se le dará la opción de votar.
Iniciado por	Usuario gestor, Usuario no gestor
Finalizado por	Usuario gestor, Usuario no gestor
Descripción	Un usuario entra en una discusión, selecciona una opción y deja una alegación.
Corresponde al Requisito:	R3.4 Discutir en Proyecto

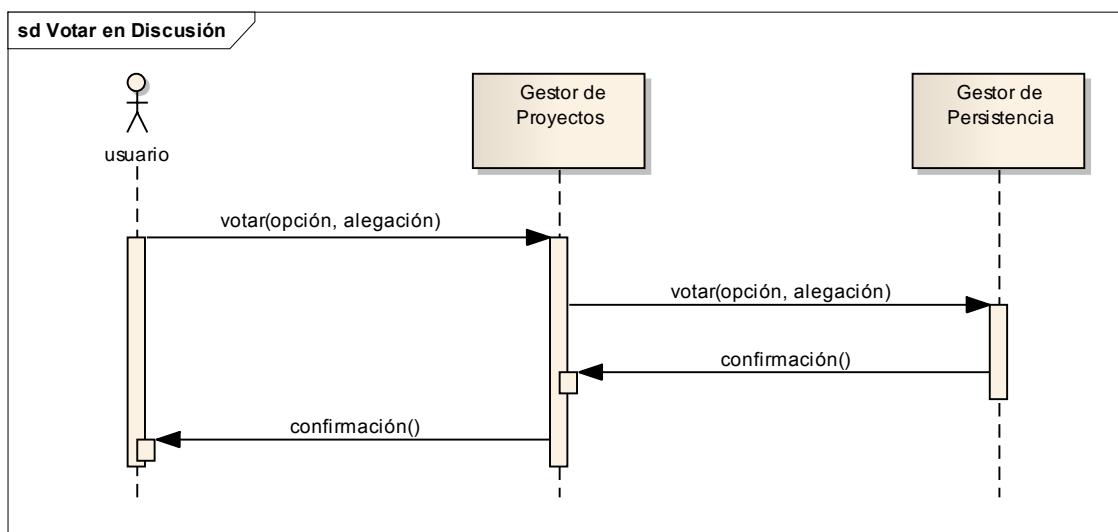


Figura 46: Votar en discusión con éxito

Caso de uso 3.5: Abandonar Proyecto

Escenario 3.5.a : Abandonar Proyecto con éxito	
Precondiciones	El usuario debe ser miembro del proyecto miembro del proyecto (participante o gestor).
Postcondiciones	El usuario no estará en el proyecto al finalizar.
Excepciones	Que el usuario haya sido excluido mientras decide marchar, el sistema le indicará que ha sido excluido.
Iniciado por	Usuario gestor, Usuario no gestor
Finalizado por	Usuario gestor, Usuario no gestor
Descripción	Un usuario decide abandonar un proyecto, se lo indica al sistema y lo abandona.
Corresponde al Requisito:	R3.5 Abandonar Proyecto

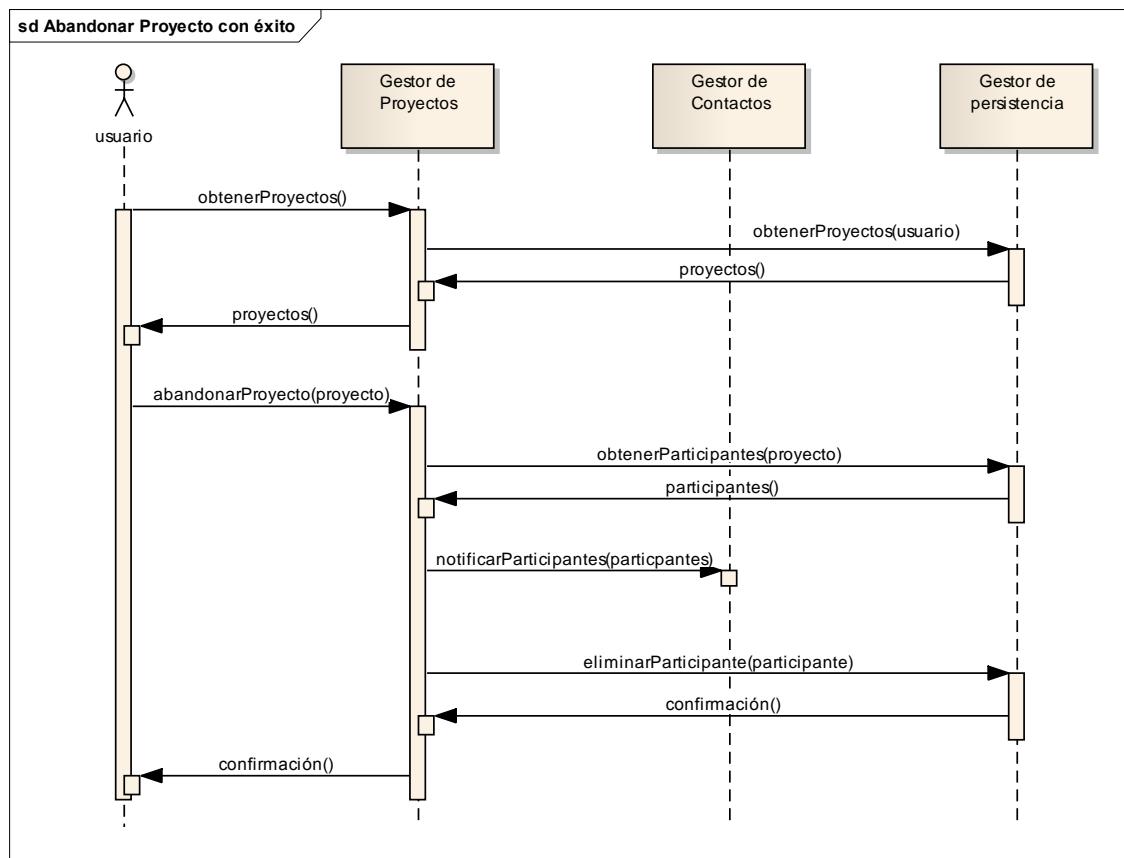


Figura 47: Abandonar proyecto con éxito

5.1.1.1.3 Gestionar Recursos

Caso de uso 4.1: Crear Recurso

Escenario 4.1.1.a : Crear nuevo recurso con éxito	
Precondiciones	El usuario creador debe ser un participante del proyecto.
Postcondiciones	Habrá un nuevo recurso en el proyecto.
Excepciones	-Existía un recurso con el mismo nombre en el mismo nivel del árbol. El sistema permitirá crear varios recursos con el mismo nombre.
Iniciado por	Usuario gestor, Usuario no gestor
Finalizado por	Usuario gestor, Usuario no gestor
Descripción	Un usuario se sitúa en la sección Árbol del proyecto, en la raíz del árbol o en algún recurso de este, y selecciona la opción de crear recurso, y dentro de esta nuevo recurso. Selecciona el tipo del recurso. Asigna nombre a dicho recurso, y ya estará creado.
Corresponde al Requisito:	R4.1.1 Crear nuevo recurso

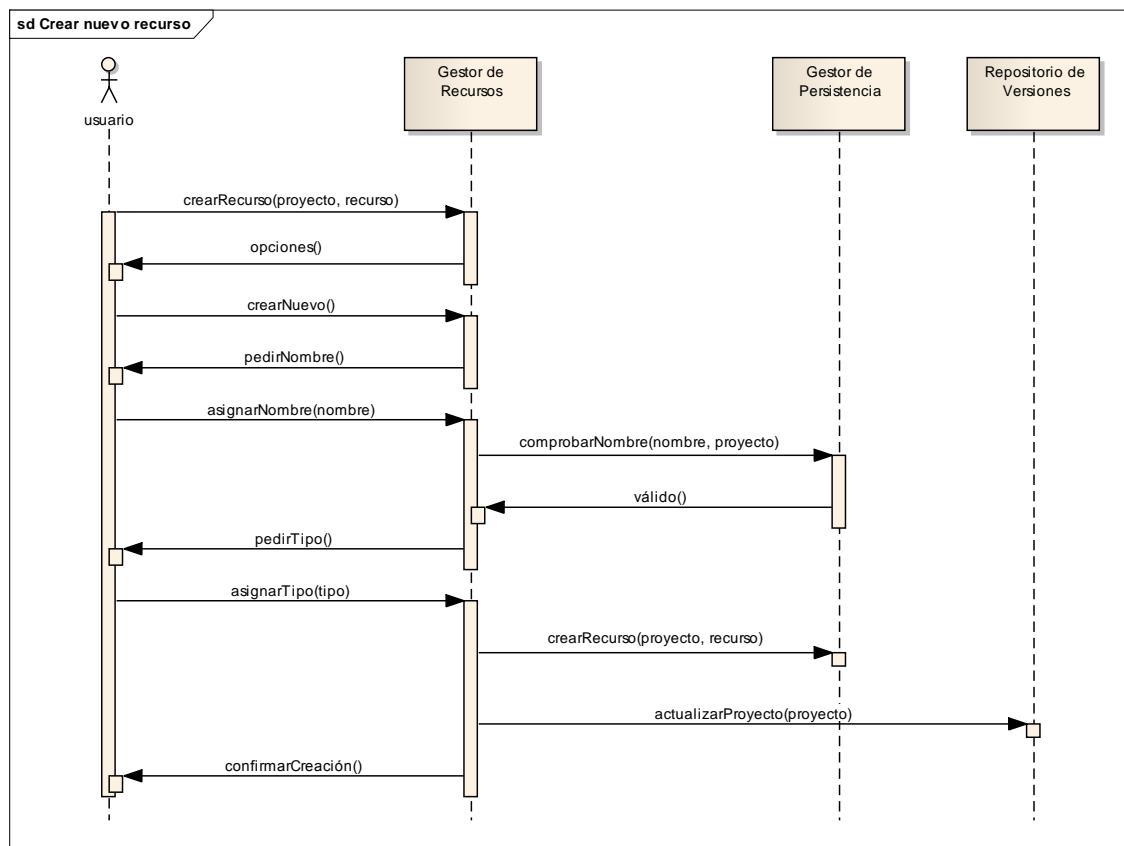


Figura 48: Crear nuevo recurso con éxito

Escenario 4.1.2.a :Crear e Importar desde Local con éxito	
Precondiciones	El usuario creador debe ser un participante del proyecto.
Postcondiciones	Habrá un nuevo recurso en el sistema.
Excepciones	-El recurso que se está importando tiene el mismo nombre que un recurso existente en el mismo nivel del árbol. El sistema permitirá importar varios recursos con el mismo nombre.
Iniciado por	Usuario gestor, Usuario no gestor
Finalizado por	Usuario gestor, Usuario no gestor
Descripción	Un usuario se sitúa en la sección Árbol de recursos del proyecto, en la raíz del árbol o en algún recurso de este, y selecciona la opción de importar desde local. Selecciona el tipo del recurso, nótese que esto condicionará lo que podrá importar. Selecciona el recurso que desea importar Si el nombre es válido, estará creado.
Corresponde al Requisito:	R4.1.2 Importar recurso Local

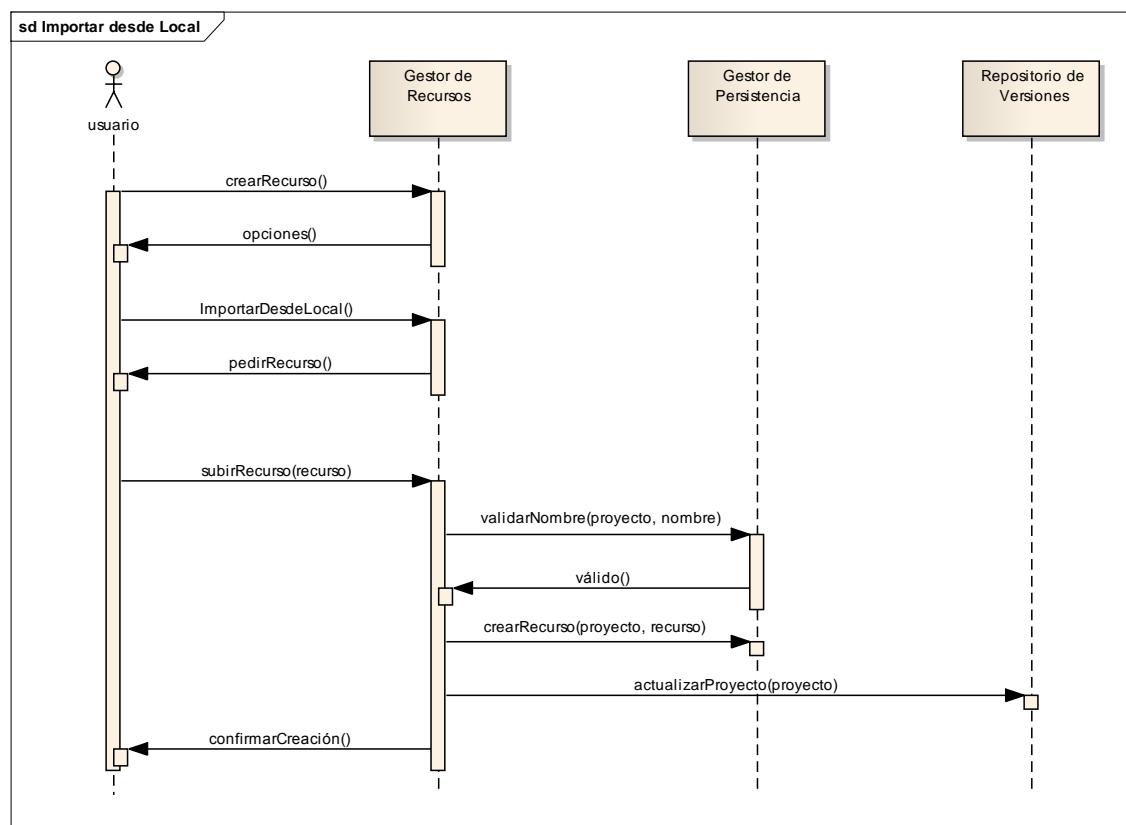


Figura 49: Importar desde local con éxito

Escenario 4.1.3.a : Crear e Importar Recurso de Google Docs con éxito	
Precondiciones	El usuario creador debe ser un participante del proyecto.
Postcondiciones	Habrá un nuevo recurso en el sistema.
Excepciones	-El recurso que se está importando tiene el mismo nombre que un recurso existente en el mismo nivel del árbol. El sistema permitirá importar desde Google Docs varios recursos con el mismo nombre.
Iniciado por	Usuario gestor, Usuario no gestor
Finalizado por	Usuario gestor, Usuario no gestor
Descripción	Un usuario se sitúa en la sección Árbol del proyecto, en la raíz del árbol o en algún recurso de este, y selecciona la opción de importar desde Google Docs. Selecciona el recurso que desea importar de entre los documentos que tiene en Google Docs.
Corresponde al Requisito:	R4.1.3 Importar recurso de Google Docs

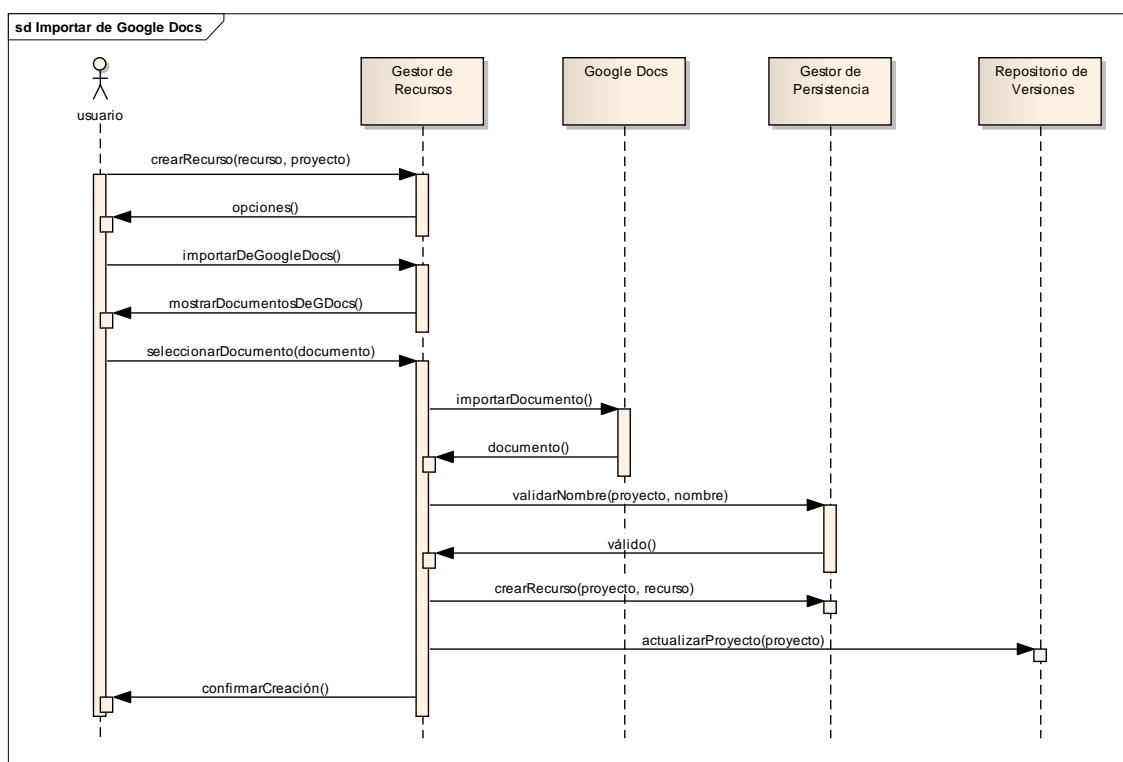


Figura 50: Importar desde Google Docs con éxito

Caso de uso 4.2: Modificar Recurso

Escenario 4.2.1.a : Editar recurso online con éxito	
Precondiciones	El usuario debe pertenecer al proyecto, el recurso debe existir previamente, y debe ser de tipo file.
Postcondiciones	El usuario editara el contenido del recurso online, y esta modificación persistirá.
Excepciones	Dos usuarios editan online el mismo recurso al mismo tiempo. El sistema lo soportara sin problemas, guardara primero los cambios de uno, y sobre estos aplicara la modificación del otro usuario. Si el fichero que se está modificando es eliminado, se notificará y se le dará la opción de crearlo de nuevo con el contenido modificado.
Iniciado por	Usuario Gestor, Usuario no Gestor
Finalizado por	Usuario Gestor, Usuario no Gestor
Descripción	Un usuario selecciona la opción de editar online un recurso. Entonces va a un editor, en el que modifica el contenido del documento. Guarda dicho contenido, y ya está modificado.
Corresponde al Requisito:	R4.2.1 Editar Contenido Online

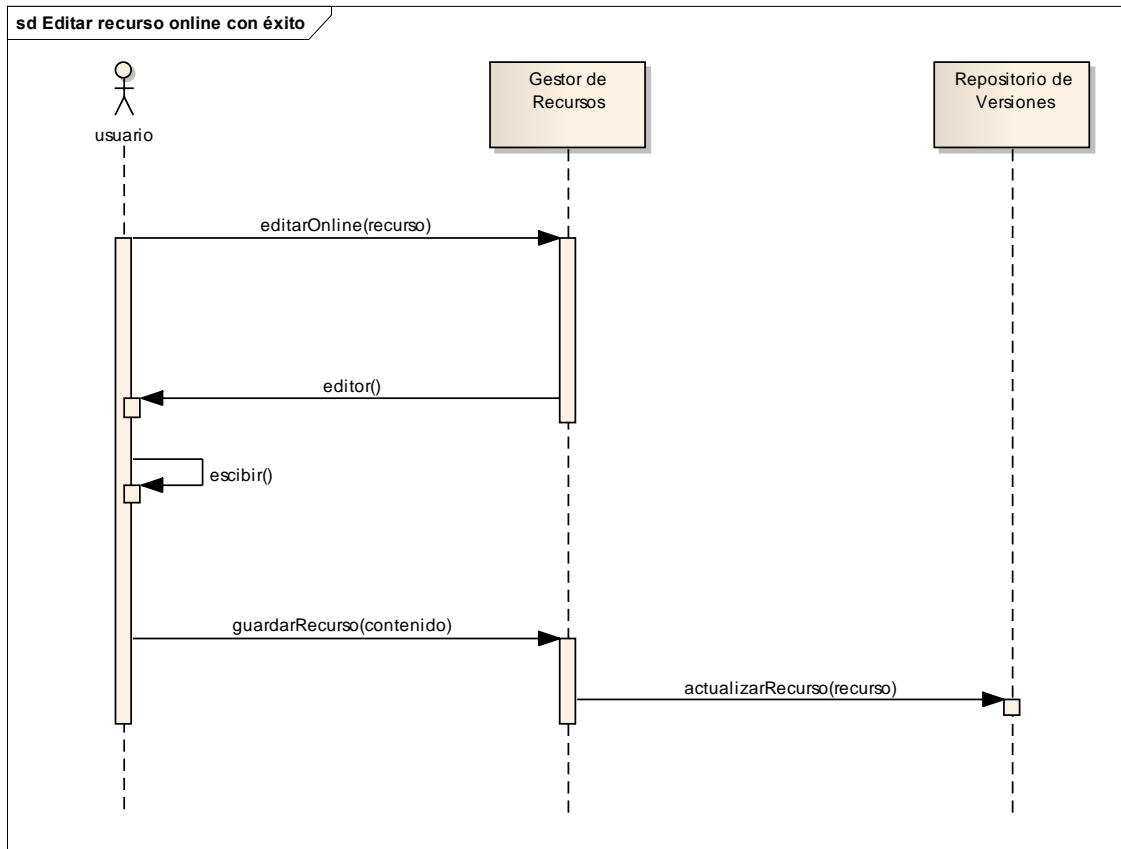


Figura 51: Editar recurso online con éxito

Escenario 4.2.2.a : Renombrar recurso con éxito	
Precondiciones	El usuario debe pertenecer al proyecto, el recurso debe existir previamente.
Postcondiciones	El recurso quedara renombrado.
Excepciones	<ul style="list-style-type: none"> -El usuario renombra un recurso que está editando otro miembro. En este caso no habrá problemas, el usuario editor podrá guardar el recurso sin problemas, y esto no afectará a la operación de renombrado. -EL usuario renombra un recurso y mientras este es eliminado, se le mostrará un mensaje informándole de que no se ha podido efectuar la operación de renombrado.
Iniciado por	Usuario Gestor, Usuario no Gestor
Finalizado por	Usuario Gestor, Usuario no Gestor
Descripción	<p>Un usuario selecciona la opción de renombrar un recurso.</p> <p>Introduce un nuevo nombre, y el recurso queda renombrado.</p>
Corresponde al Requisito:	R4.2.2 Renombrar Recurso

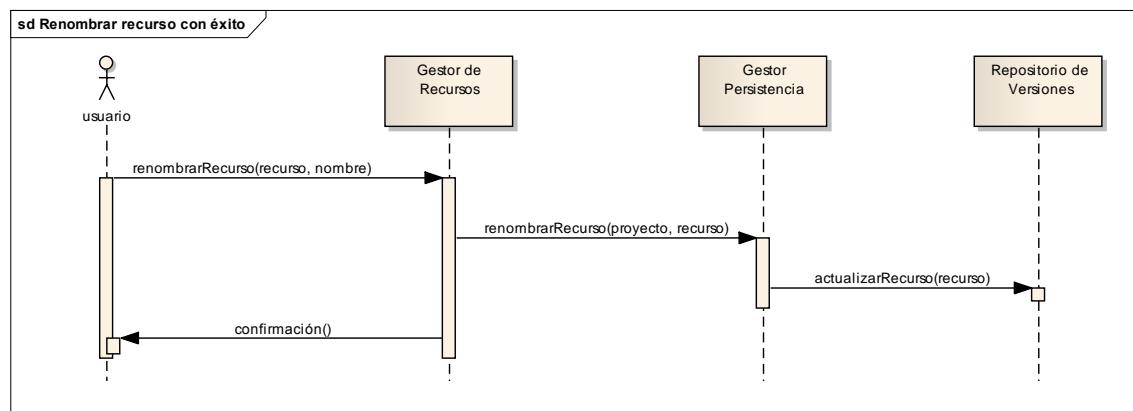


Figura 52: Renombrar recurso con éxito

Escenario 4.2.3.a : Mover recurso con éxito	
Precondiciones	El usuario debe pertenecer al proyecto, el recurso debe existir previamente.
Postcondiciones	El recurso estará en dentro de otro recurso contenedor, o dentro de la raíz.
Excepciones	-El recurso es eliminado mientras se mueve, en ese caso al completar el movimiento, se eliminará. -El recurso se mueve hacia el mismo recurso en el que ya está. En este caso la operación será transparente.
Iniciado por	Usuario Gestor, Usuario no Gestor
Finalizado por	Usuario Gestor, Usuario no Gestor
Descripción	Un usuario selecciona para un recurso, la opción de moverlo, a continuación se le pide que seleccione el nuevo recurso padre, lo selecciona y el recurso se mueve.
Corresponde al Requisito:	R4.2.3 Mover Recurso

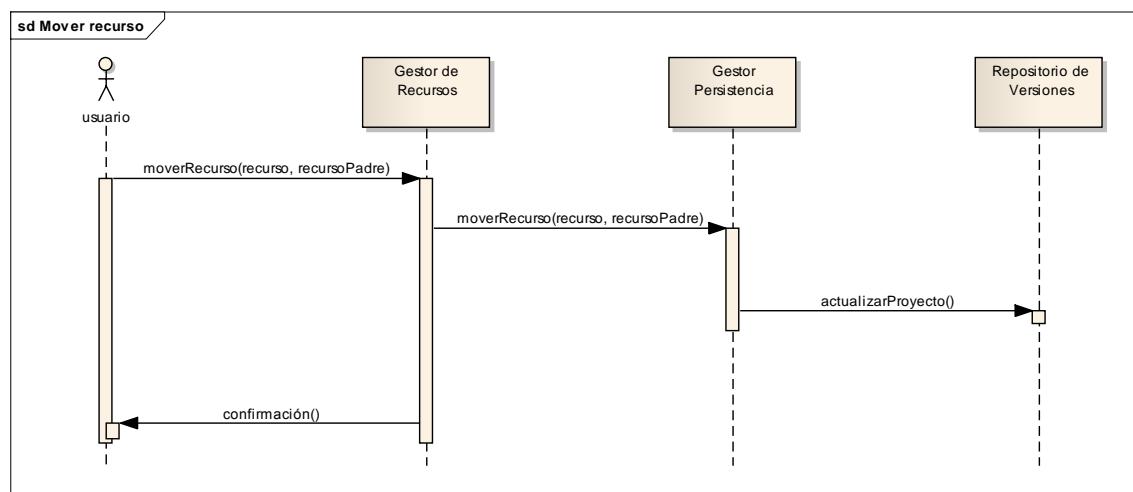


Figura 53: Mover recurso con éxito

Escenario 4.2.4.a : Descargar recurso con éxito	
Precondiciones	El usuario debe pertenecer al proyecto, el recurso debe existir previamente.
Postcondiciones	El usuario dispondrá en local del recurso descargado.
Excepciones	-
Iniciado por	Usuario Gestor, Usuario no Gestor
Finalizado por	Usuario Gestor, Usuario no Gestor
Descripción	El usuario selecciona para un recurso concreto la opción de descargarlo.
Corresponde al Requisito:	R4.2.4 Descargar Recurso

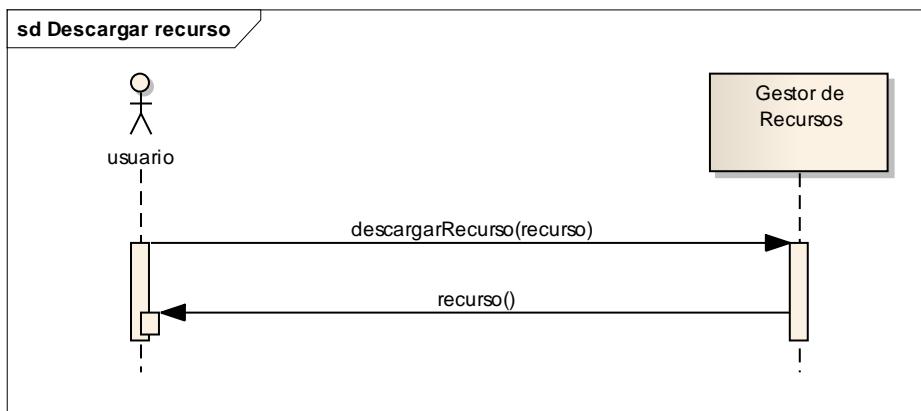


Figura 54: Descargar recurso con éxito

Escenario 4.2.5.a : Subir recurso con éxito	
Precondiciones	El usuario debe pertenecer al proyecto, el recurso debe existir previamente.
Postcondiciones	El recurso quedará modificado con el contenido subido por el usuario.
Excepciones	El usuario intenta subir contenido que no corresponde a ese recurso. El sistema no lo permitirá.
Iniciado por	Usuario Gestor, Usuario no Gestor
Finalizado por	Usuario Gestor, Usuario no Gestor
Descripción	El usuario selecciona un recurso existente, y para este la opción de subir recurso, y selecciona un contenido en su directorio local, y lo sube.
Corresponde al Requisito:	R4.2.5 Subir Recurso

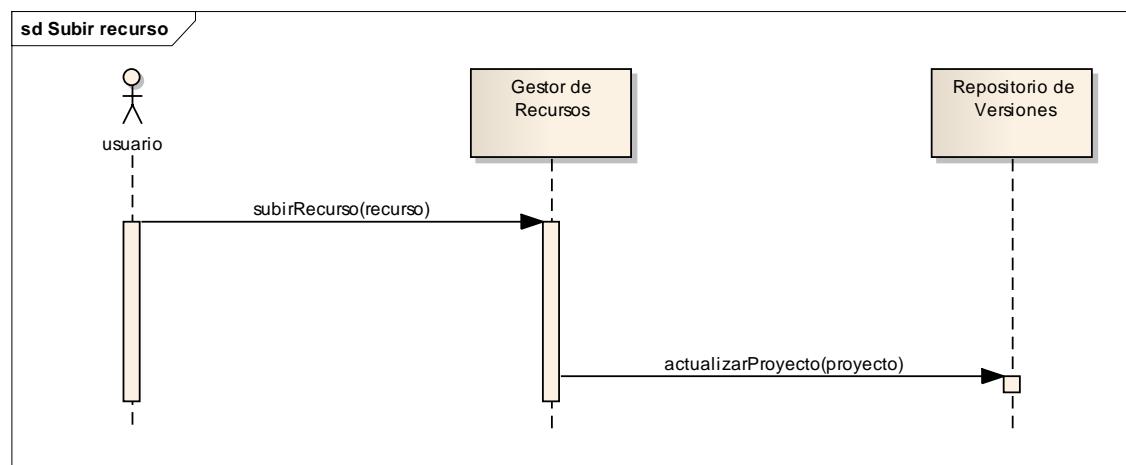


Figura 55: Subir recurso con éxito

Caso de uso 4.3: Eliminar Recurso

Escenario 4.3.a : Eliminar recurso con éxito	
Precondiciones	El usuario debe ser un participante del proyecto y el recurso debe existir previamente.
Postcondiciones	El recurso desaparecerá del proyecto.
Excepciones	Un usuario elimina un recurso que otro está modificando. El sistema notificará al usuario que realiza la modificación de que no se ha podido realizar la modificación.
Iniciado por	Usuario Gestor, Usuario no Gestor
Finalizado por	Usuario Gestor, Usuario no Gestor
Descripción	Un usuario selecciona la opción de borrar un recurso, y lo borra.
Corresponde al Requisito:	4.1 Eliminar Recurso

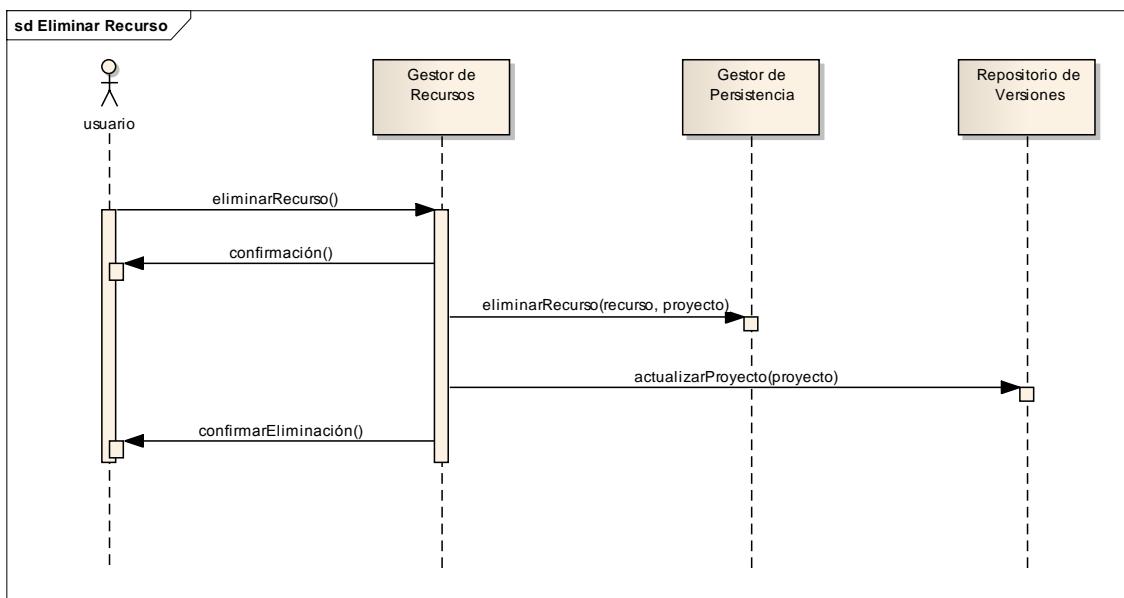


Figura 56: Eliminar recurso con éxito

5.3 Clases del Análisis

En este apartado haré un análisis de las clases que definen el dominio del sistema, abstracciones que modelan los problemas y las soluciones del mundo real que cubre este sistema.

Las clases mostradas en la Figura 57 son clases de análisis que podrán generar otro tipo de estructuras en el diseño del sistema.

5.3.1 Diagrama de Clases del Análisis

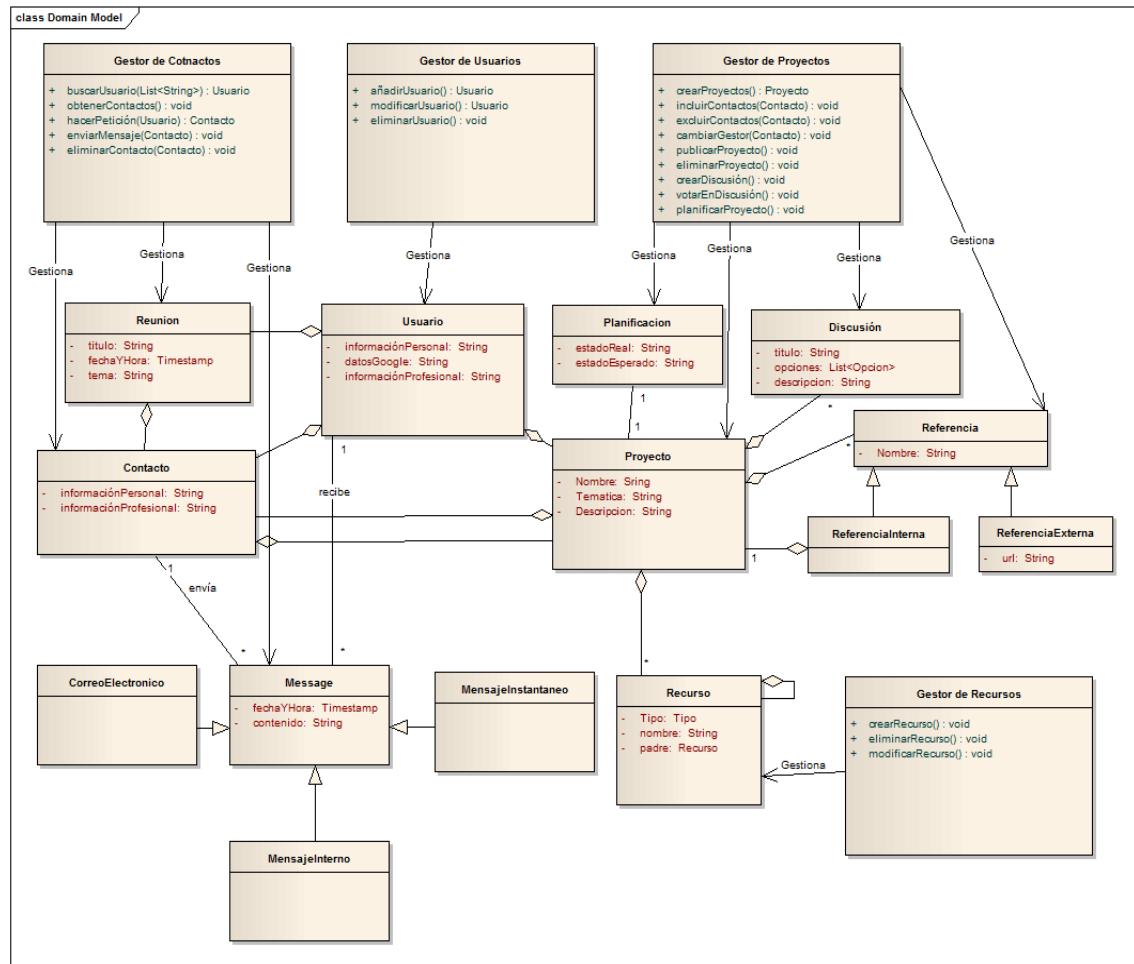


Figura 57: Diagrama de clases del Análisis

5.3.2 Descripción de las Clases del Análisis

Clase Usuario	
Descripción	Clase que modela a un usuario del sistema, este es una persona que accede a la aplicación a través de internet y realiza el proceso de registro, también conocido como añadir Usuario.
Atributos	<ul style="list-style-type: none"> • Información personal: Información personal del usuario, nombre apellidos, fecha de nacimiento y ciudad. • Información profesional: Estudios y empleos del usuario. • Cuenta de Google: Cuenta de Google aportada por el usuario. • Contactos: Listado de usuarios con los que está relacionado un usuario. • Proyectos: Listado de proyectos de los que un usuario es miembro.
Métodos	Esta clase no posee operaciones.

Clase Contacto	
Descripción	Modela a un usuario que está relacionado con el usuario tomado como referencia principal, para que esta relación exista el usuario principal debe añadir a otro usuario como contacto. Lo usamos para modelar la relación entre usuarios así como la perspectiva que tiene un usuario de sus contactos.
Atributos	<ul style="list-style-type: none"> • Información personal: Información personal del usuario, nombre apellidos, fecha de nacimiento y ciudad. • Información profesional: Estudios y empleos del usuario. • Contactos: Listado de usuarios con los que está relacionado un usuario. • Proyectos: Listado de proyectos de los que un usuario es miembro.
Métodos	Esta clase no posee operaciones.

Clase Reunión	
Descripción	Modela una reunión, esto es una reunión entre un usuario y algunos de sus contactos.
Atributos	<ul style="list-style-type: none"> • Título: Titulo o nombre de la reunión. • Fecha: Fecha en la que se celebrará la reunión. • Tema: Tema o asunto de la reunión.
Métodos	Esta clase no posee operaciones.

Clase Mensaje	
Descripción	Modela un mensaje que se envía entre un usuario y alguno de sus contactos. Los mensajes pueden ser instantáneos o no, los no instantáneos pueden ser enviados también por correo electrónico.
Atributos	<p>Tiene un contenido y una fecha y hora.</p> <ul style="list-style-type: none"> • Fecha: Fecha en la que se envía mensaje. • Contenido: Contenido del mensaje. • Emisor: Usuario que envía el mensaje. • Destinatario: Usuario al que está dirigido el mensaje.
Métodos	Esta clase no posee operaciones.

Clase Proyecto	
Descripción	Modela un proyecto realizado entre un usuario y algunos de sus contactos.
Atributos	<ul style="list-style-type: none"> • Nombre: Nombre o título del proyecto. • Temática: Temática o área de conocimientos del proyecto. • Descripción: Descripción del proyecto del proyecto. • Participantes: Usuarios que participan en el proyecto. • Recursos: Recursos que los usuarios desarrollan dentro del proyecto. • Referencias: Referencias del

	proyecto.
	<ul style="list-style-type: none"> • Planificación: Planificación del proyecto. • Discusiones: Discusiones del proyecto.
Métodos	Esta clase no posee operaciones.

Clase Planificación	
Descripción	Modela la planificación hecha sobre un proyecto, para tratar con la planificación el proyecto cuenta con hitos y tareas.
Atributos	<ul style="list-style-type: none"> • Estado esperado: El estado en el que se debería encontrar el proyecto. Es calculado por el sistema. • Estado real: Estado en el que realmente se encuentra el sistema. Es asignado por el gestor. • Hitos: Estados esperados asignados en fechas concretas, que serán usados para calcular el estado esperado. • Tareas: Tareas definidas por los usuarios para alcanzar los estados esperados o hitos.
Métodos	Esta clase no posee operaciones.

Clase Discusión	
Descripción	Modela una discusión, esto es un mecanismo para que los miembros de un proyecto expongan opiniones sobre un planteamiento, y escojan una entre varias opciones.
Atributos	<ul style="list-style-type: none"> • Título: título de la discusión. • Descripción: Descripción del tema sobre el que trata la discusión. Debe ser el contexto sobre el cual los usuarios forman una opinión del tema. Es aportada por el gestor. • Opciones: Opciones que se ofrecen para el tema expuesto. El objetivo de la discusión es que los usuarios escojan una de las opciones definidas en la discusión.

Métodos	Esta clase no posee métodos.
---------	------------------------------

Clase Referencia	
Descripción	Modela una referencia del proyecto, las referencias abstraen referencias a páginas web en internet. Un proyecto tendrá un listado de referencias.
Atributos	<ul style="list-style-type: none"> • Nombre: el nombre que se le da a la referencia. • URL: URL de la referencia en la web.
Métodos	Esta clase no posee operaciones.

Clase Gestor de Usuarios	
Descripción	Clase que se encarga de gestionar todo lo relativo a los usuarios. Sus responsabilidades son: <ul style="list-style-type: none"> • Crear nuevos usuarios • Eliminar usuarios existentes • Modificar información personal de los usuarios existentes. • Modificar información personal de los usuarios existentes. • Modificar la cuenta de los usuarios existentes. • Listar todos los usuarios existentes. • Obtener toda la información de un usuario.
Atributos	Esta clase no tiene atributos.
Métodos	<ul style="list-style-type: none"> • Añadir usuario: Añade un nuevo usuario al sistema. • Eliminar usuario: Eliminar un usuario existente. • Modificar usuario: Modifica un usuario. • Buscar usuarios: Obtiene los usuarios del sistema.

Clase Gestor de Contactos	
Descripción	Clase que se encarga de la gestión los contactos, así como de las interacciones entre estos y el usuario.
Atributos	Esta clase no tiene atributos.
Métodos	<p>Buscar Usuarios, Obtener Contactos, Hacer Petición, Enviar Mensaje, Eliminar Contacto.</p> <ul style="list-style-type: none"> • Obtener Contactos: Obtiene los contactos de un usuario. • Enviar petición: Envía una petición a un contacto. • Enviar Mensaje: Envía un mensaje a un contacto. • Eliminar contacto: Elimina a un contacto.

Clase Gestor de Proyectos	
Descripción	Clase que se encarga de la gestión de los proyectos.
Atributos	Esta clase no tiene atributos.
Métodos	<p>Crear Proyecto, Incluir Contacto, Excluir Contacto, Cambiar Gestor, Publicar Proyecto, Eliminar Proyecto, Crear Discusión, Votar en Discusión, Planificar Proyecto.</p> <ul style="list-style-type: none"> • Crear Proyecto: Crea un nuevo proyecto. • Incluir Contacto: Incluye un contacto en el proyecto. • Excluir Contacto: Excluye un contacto del proyecto. • Cambiar Gestor: Cambia el gestor del proyecto. • Publicar Proyecto: Publica el proyecto. • Eliminar Proyecto: Elimina el proyecto. • Crear discusión: Crea una nueva discusión. • Votar en discusión: Vota en una discusión.

Clase Gestor de Recursos	
Descripción	Esta clase se encarga de la gestión de los recursos de un proyecto.
Atributos	Esta clase no tiene atributos.
Métodos	<p>Crear Recurso, Modificar Recurso, eliminar Recurso.</p> <ul style="list-style-type: none">• Crear Recurso: Crea un nuevo recurso.• Modificar Recurso: Modifica un recurso.• Eliminar Recurso: Elimina un recurso.

5.4 Análisis de Interfaces de Usuario

En este apartado se expone un análisis de las interfaces de usuario que tendrá la aplicación de él sistema que finalmente verá el usuario, el objetivo es definir para cada funcionalidad del sistema un esbozo de la interfaz de usuario, que si bien no sea definitivo, permita preconcebir una idea de la misma.

5.4.1 Descripción de la Interfaz

Como se especificó en el apartado 5.2.1.5 “Requisitos No funcionales”, la aplicación será una aplicación web enriquecida. Esto permite tener una idea de las opciones con que se contaran para el diseño de dicha interfaz, y tomar provecho ello en esta fase.

La aplicación consta de dos secciones muy separadas y de gran entidad, la sección principal y la sección de un proyecto. Dentro de cada una de estas se pueden distinguir varias secciones, en la pantalla principal tenemos secciones que ofrecen funcionalidades generales mientras que en la pantalla de proyecto tendremos agrupadas, también en secciones las funcionalidades que nos ofrece la aplicación para el desarrollo de proyectos.

La interfaz de usuario estará formada por paneles de pestañas, en la sección principal habrá cuatro pestañas que delimitarán 4 secciones distintas. Una de estas secciones será la sección de proyectos, desde aquí se podrá ir a la sección de un proyecto que contará con otro panel de pestañas, este panel delimitará las secciones con que contará un proyecto.

A continuación revisamos mediante una serie de modelos y desde una perspectiva prematura como serán o como podrán ser las distintas secciones de la aplicación.

5.4.1.1 Sección Principal - Notificaciones

La aplicación tendrá una sección principal que albergará todo lo que el usuario necesita de la aplicación y está fuera del marco de un proyecto en concreto.

Esta sección tendrá varias sub secciones, una por cada área funcional de la aplicación, mensajes, contactos, etc...

La sección de noticias será la primera que el usuario vera al entrar, aquí tendrá un listado de notificaciones, que le informaran de eventos que han ocurrido en el resto de las secciones. El usuario podrá desde aquí manipular esas notificaciones.

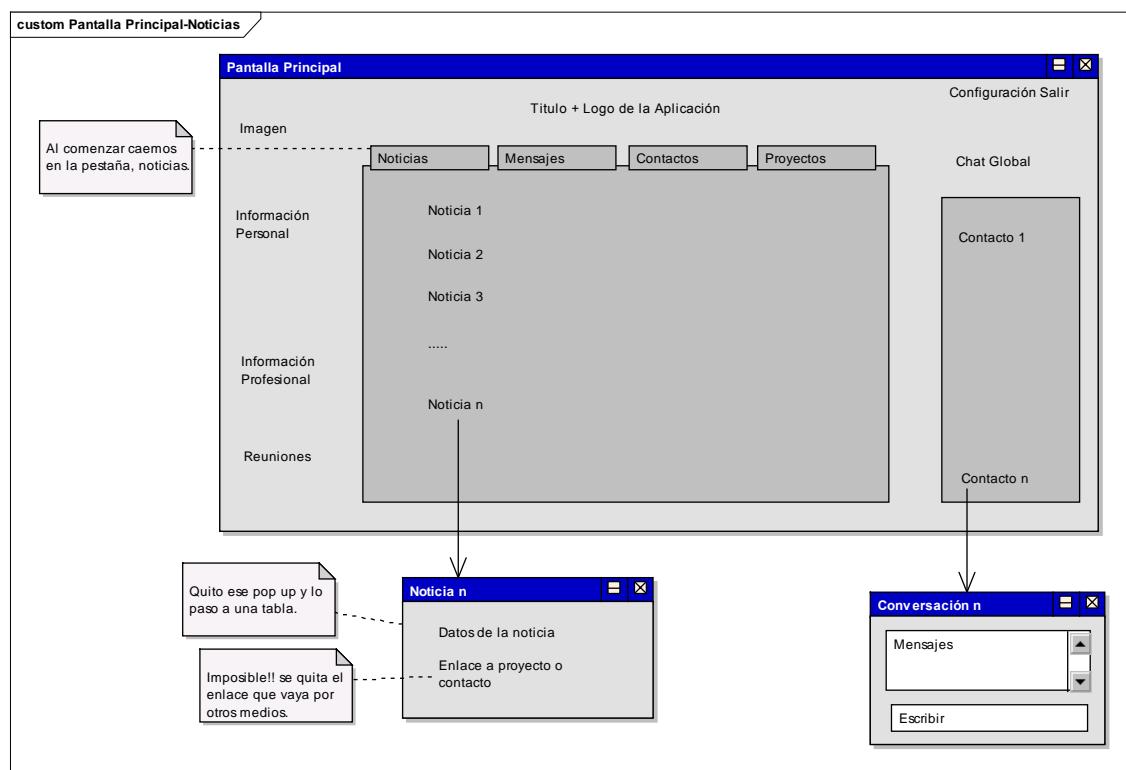


Figura 58: Sección principal - Noticias

5.4.1.2 Sección Principal - Mensajes

Sección dentro de la pantalla principal, que mostraran todos los mensajes que un usuario ha recibido, podrá contestarlos o enviar otros nuevos.

Al contestar un mensaje o enviar un nuevo mensaje se podrá seleccionar una opción para enviarlo por correo electrónico también.

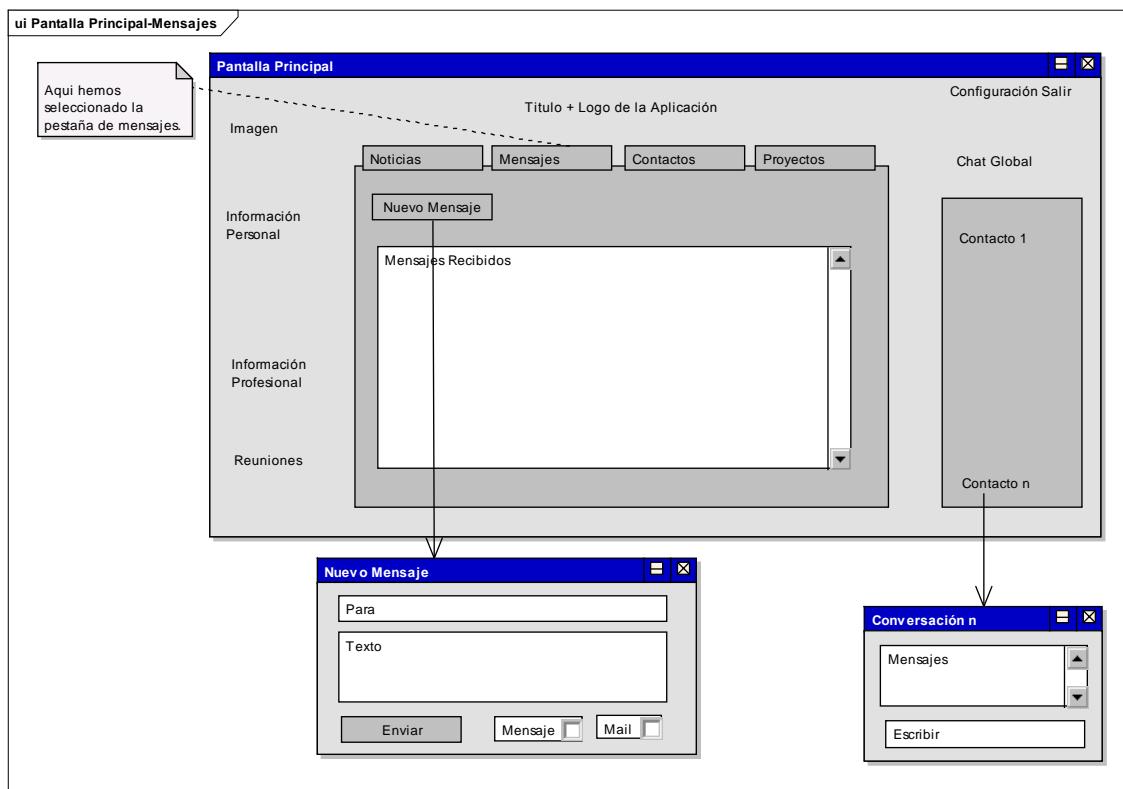


Figura 59: Sección principal - Mensajes

5.4.1.3 Sección Principal - Contactos

Sección dentro de la pantalla principal donde se encontrará todo lo relacionado con los contactos, desde aquí podrá ver la información de sus contactos o buscar usuarios.

Para cada contacto se tendrán que ofrecer las opciones necesarias para interactuar con contactos.

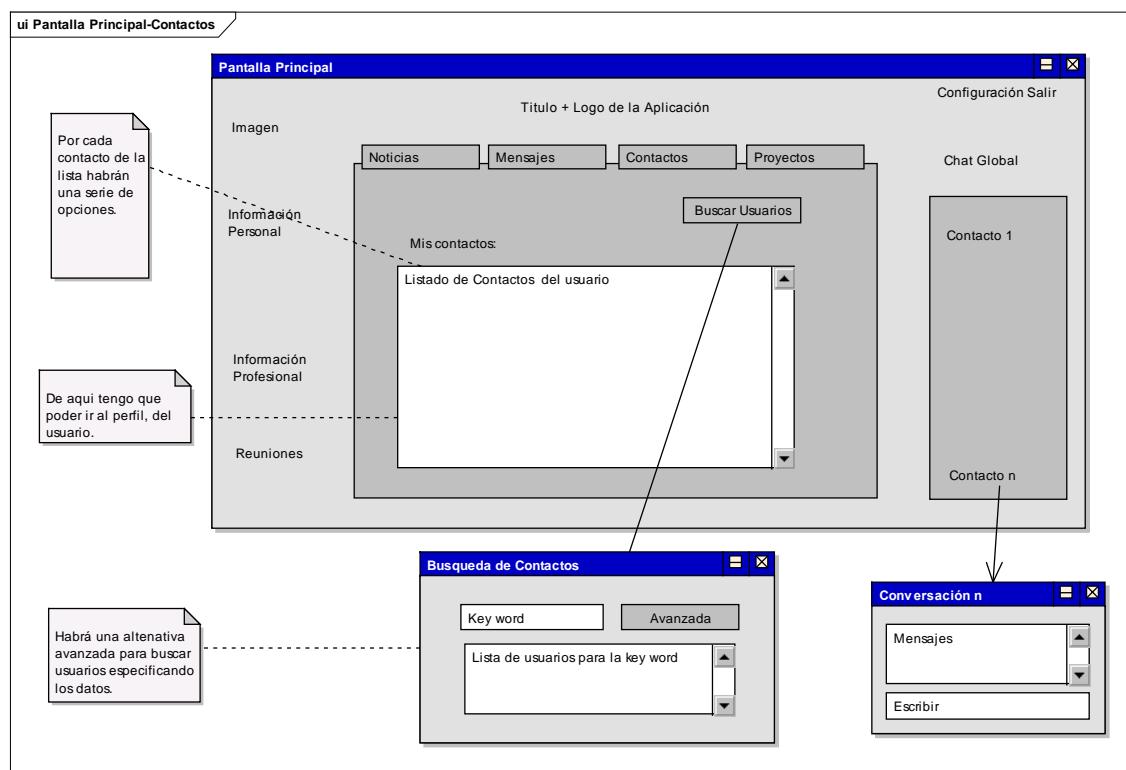


Figura 60: Sección principal- contactos

5.4.1.4 Sección Principal - Proyectos

Sección dentro de la pantalla principal donde se encontrará todo lo relativo a los proyectos, desde aquí un usuario tendrá un listado de sus proyectos.

Por cada proyecto se deben ofrecer las opciones necesarias para que el usuario interactúe con él, opciones comunes a todos los proyectos fuera del marco de un proyecto en concreto.

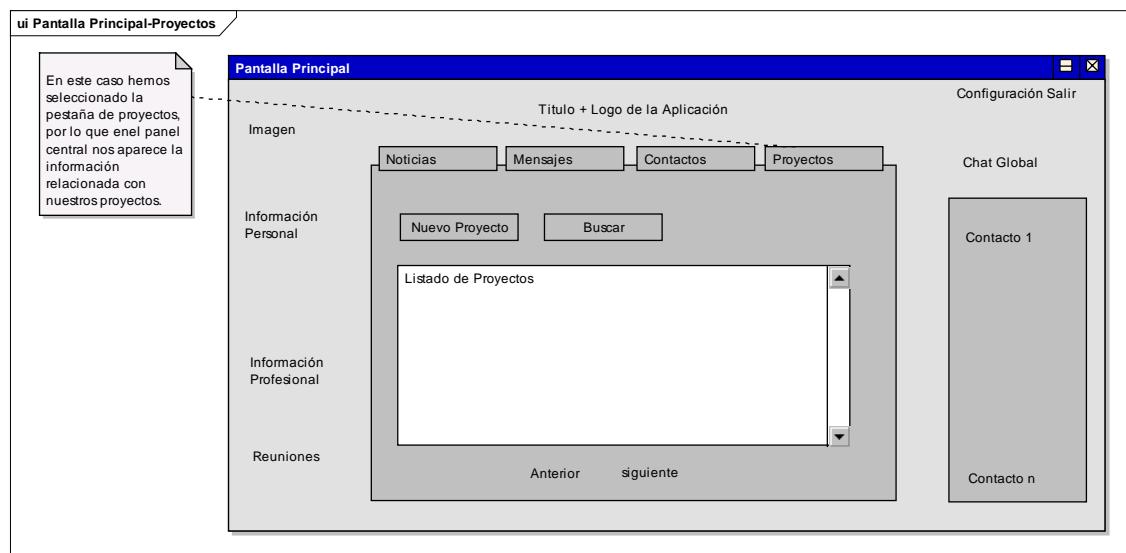


Figura 61: Sección principal - proyectos

5.4.1.5 Sección de Contacto

Sección donde un usuario podrá consultar la información de un contacto, se accederá desde la sección de contactos y en ella podrá ver información relativa a los proyectos y contactos de su contacto.

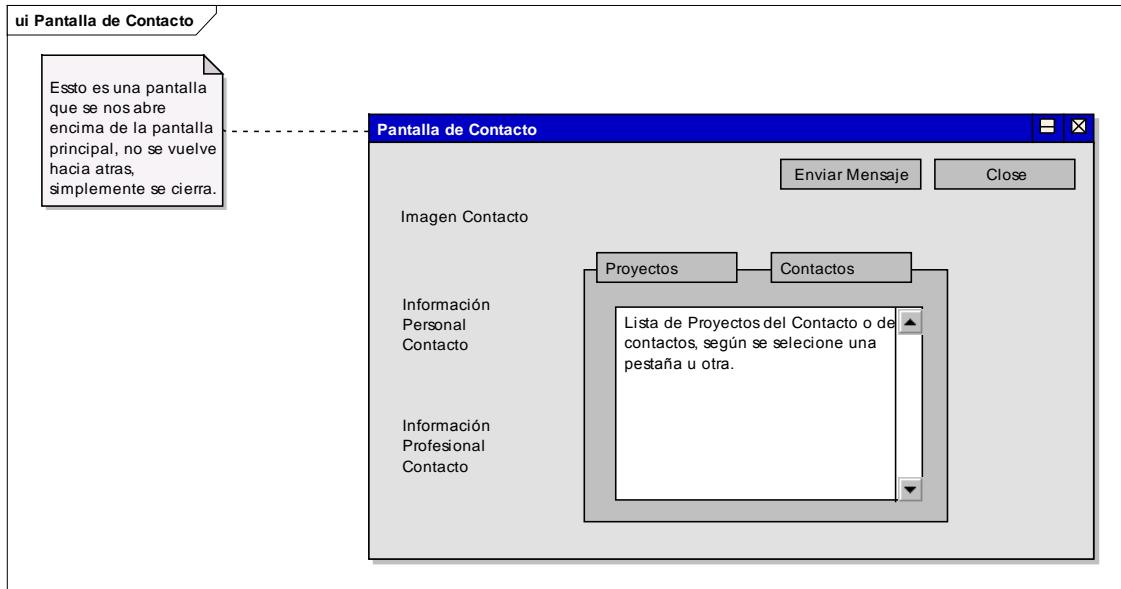


Figura 62: Sección de Contacto

5.4.1.6 Sección de Proyecto - Inicio

Para cada proyecto un usuario podrá acceder a una sección con todo lo relativo a ese proyecto, la sección del proyecto. Esta sección tendrá a su vez varias secciones, una sección por cada aspecto del proyecto. La Figura 63 muestra la primera de estas secciones.

En esta sección inicial el usuario podrá ver toda la información del proyecto, el título, la descripción, el área y todos los participantes.

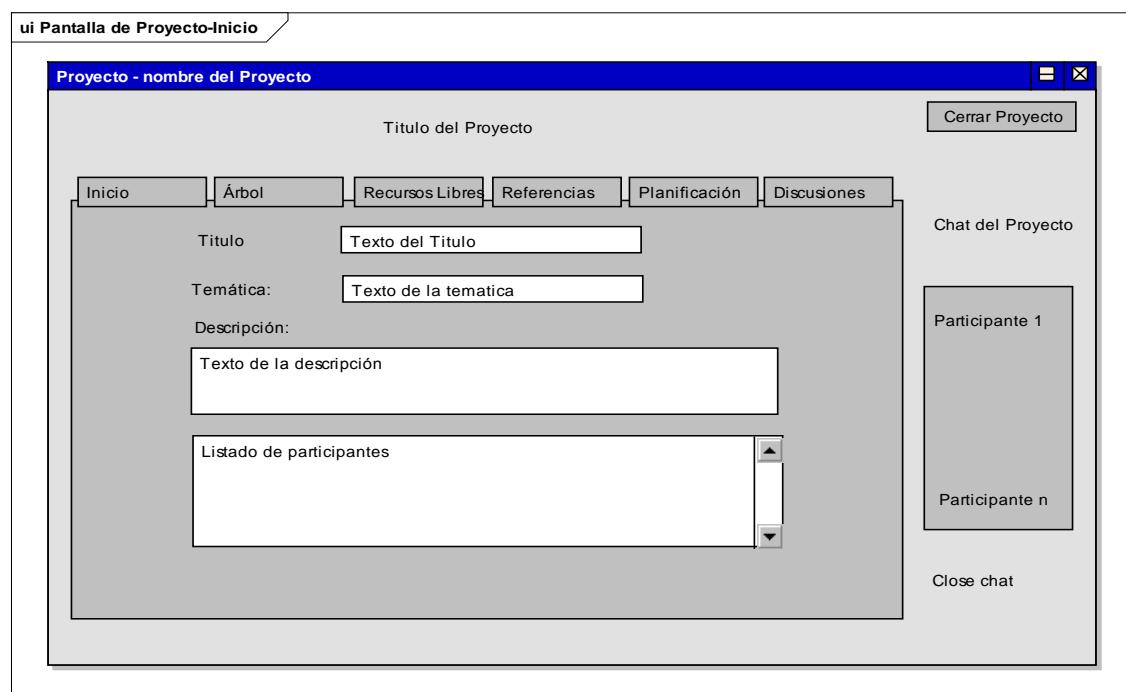


Figura 63: Sección del Proyecto - inicio

5.4.1.7 Sección de Proyecto - Árbol

Sección del proyecto que contendrá el cuerpo del proyecto, con todos los recursos del proyecto, desde esta sección se podrán por tanto crear y manipular dichos recursos.

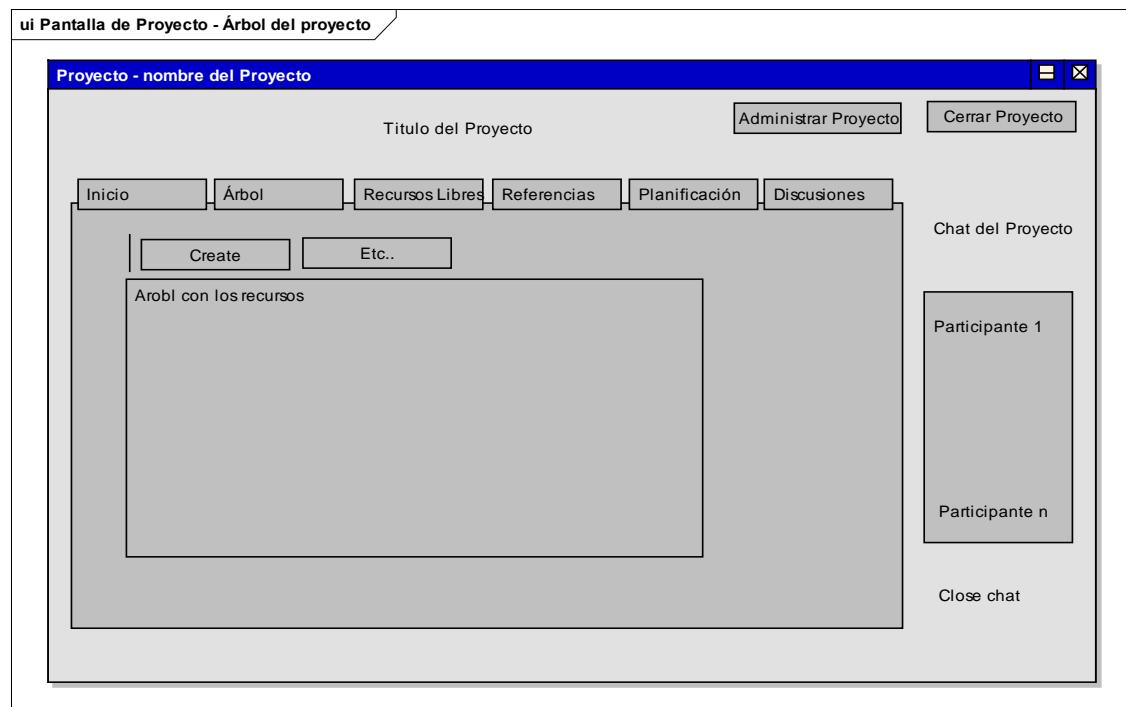


Figura 64: Sección del Proyecto - árbol

5.4.1.8 Sección de Proyecto - Recursos Libres

En esta sección del proyecto el usuario podrá ver, crear y manipular recursos libres del proyecto, estos recursos son borradores que no serán publicados.

Desde esta sección se podrán enviar recursos a la sección del árbol del proyecto y del árbol hacia esta.

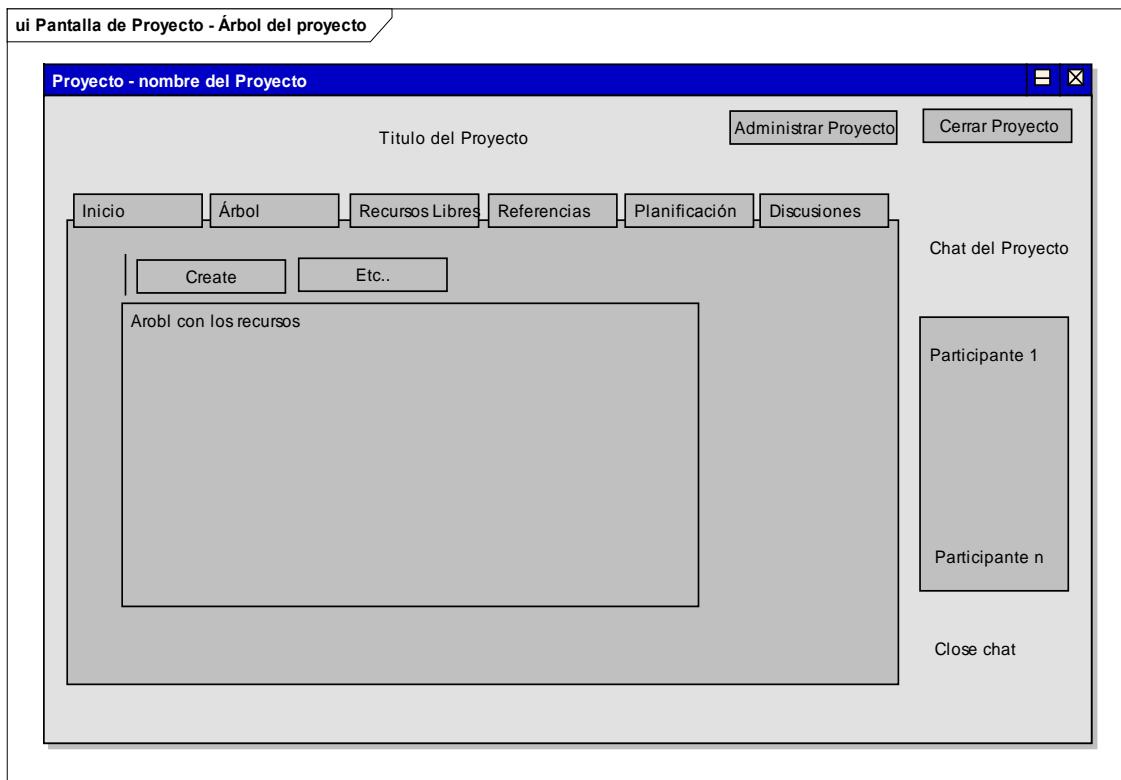


Figura 65: Sección del Proyecto – recursos libres

5.4.1.9 Sección de Proyecto - Referencias

Sección del proyecto que contendrá las referencias del proyecto, el usuario verá un listado de referencias que han sido definidas para el proyecto, podrá definir nuevas referencias y manipular las referencias existentes.

Además desde esta sección sería conveniente que el usuario pudiese acceder a las referencias en la web.

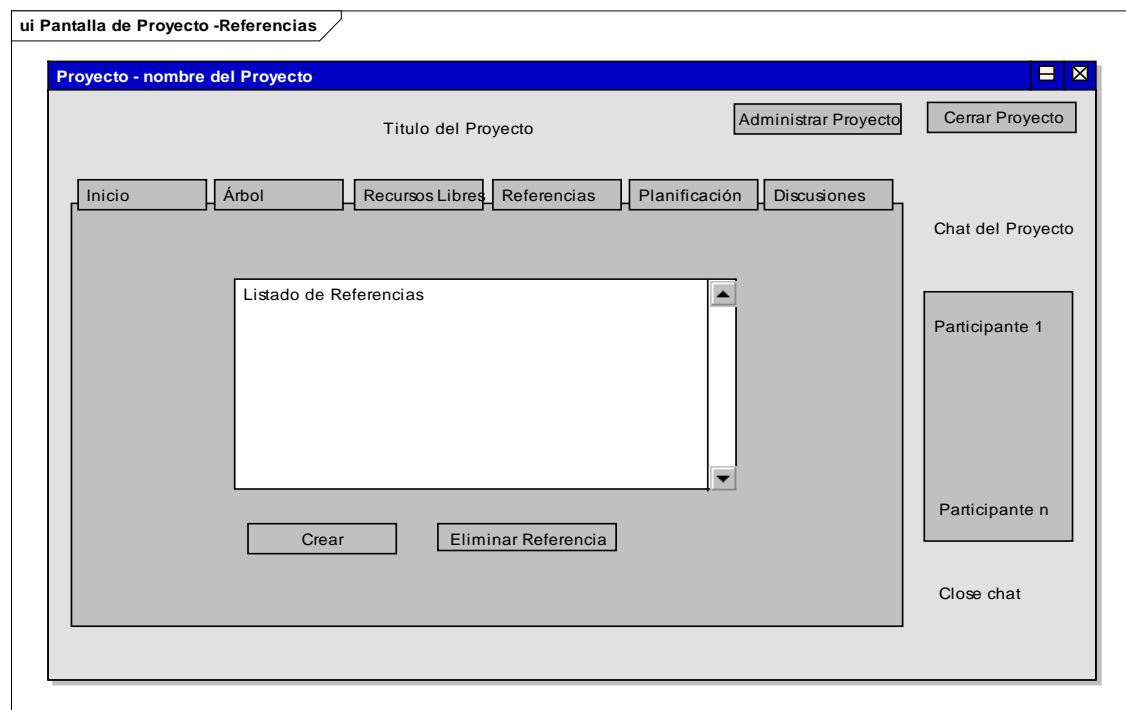


Figura 66: Sección del proyecto – referencias

5.4.1.10 Sección de Proyecto - Planificación

Sección donde el usuario tendrá los hitos del sistema, que serán el mecanismo para llevar una planificación del proyecto. El usuario podrá desde aquí crear nuevos hitos, manipular los hitos y definir el estado del proyecto.

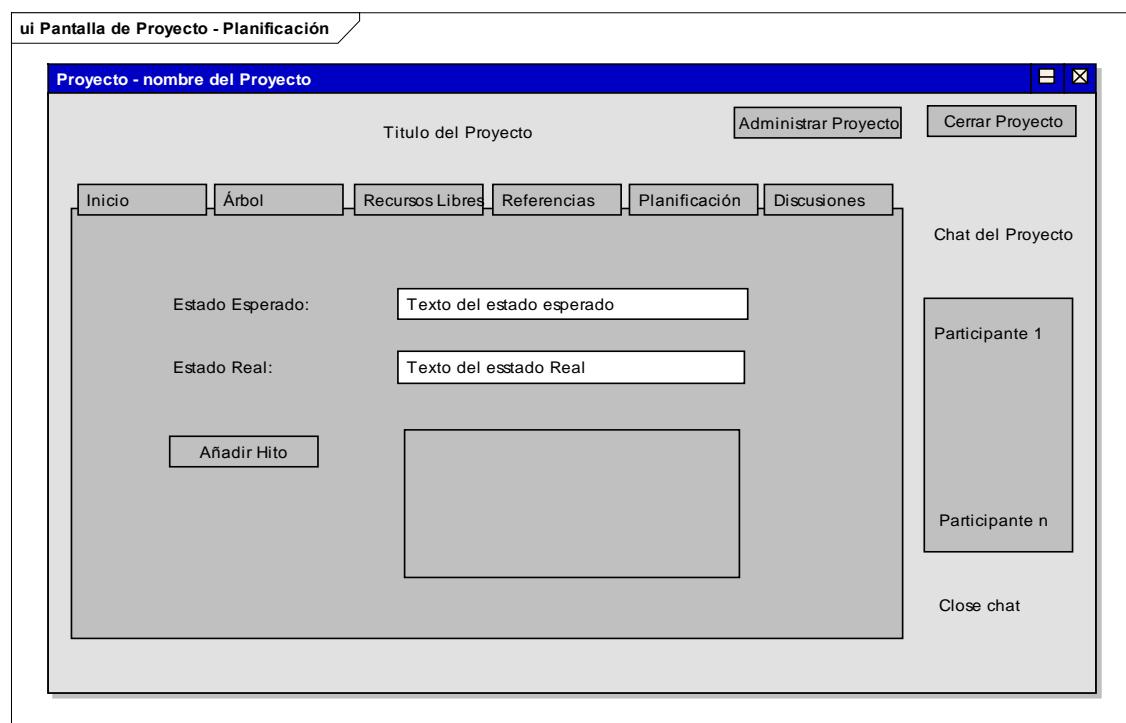


Figura 67: Sección del proyecto - planificación

5.4.1.11 Sección de Proyecto - Discusiones

El proyecto tendrá un apartado para discusiones, donde los usuarios podrán crear discusiones, además podrán por supuesto discutir en ellas.

Se podrá ver un listado de discusiones, sobre el que se podrá interactuar para discutir o realizar otras opciones sobre la discusión.

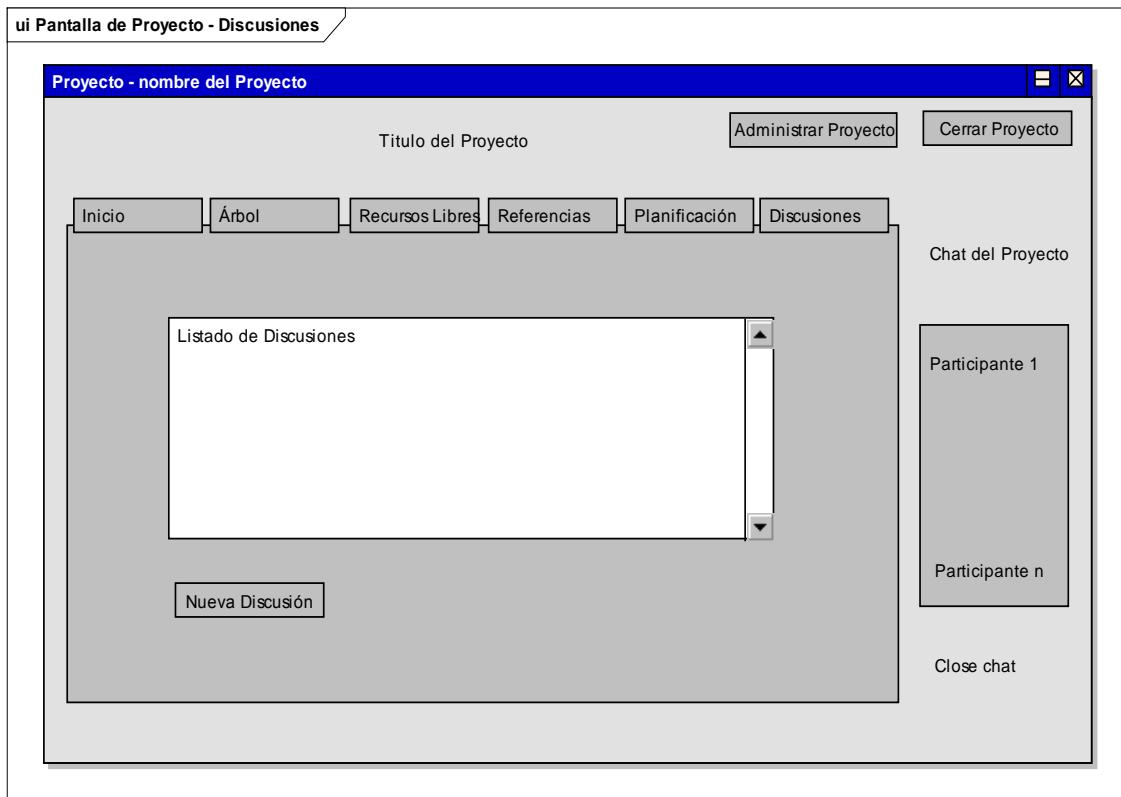


Figura 68: Sección de Proyecto - discusiones

5.4.1.12 Sección de Proyecto - Administración

Dentro de la sección del proyecto se deben proporcionar opciones relativas a la administración, presumiblemente en una sección de administración.

Además estas opciones deben estar restringidas al usuario gestor del proyecto, lo que deberá ser tenido en cuenta durante el diseño de esta interfaz.

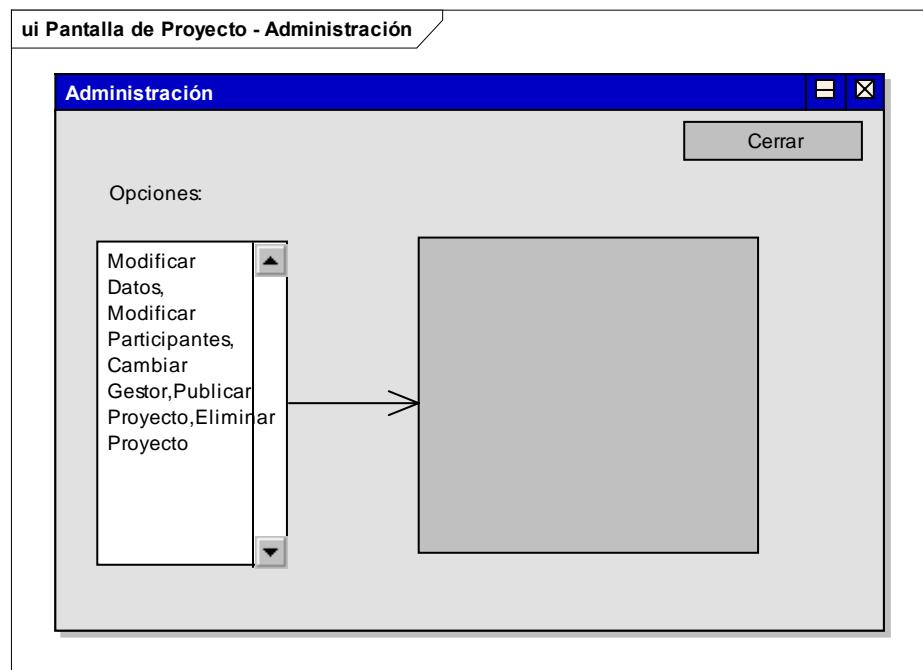


Figura 69: Sección del Proyecto - discusiones

5.4.2 Descripción del Comportamiento

Aunque las secciones de la interfaz están estructuradas en pestañas, el diagrama de la Figura 70 muestra mediante un diagrama de navegabilidad, como el usuario puede navegar por las distintas secciones que las pestañas definen.

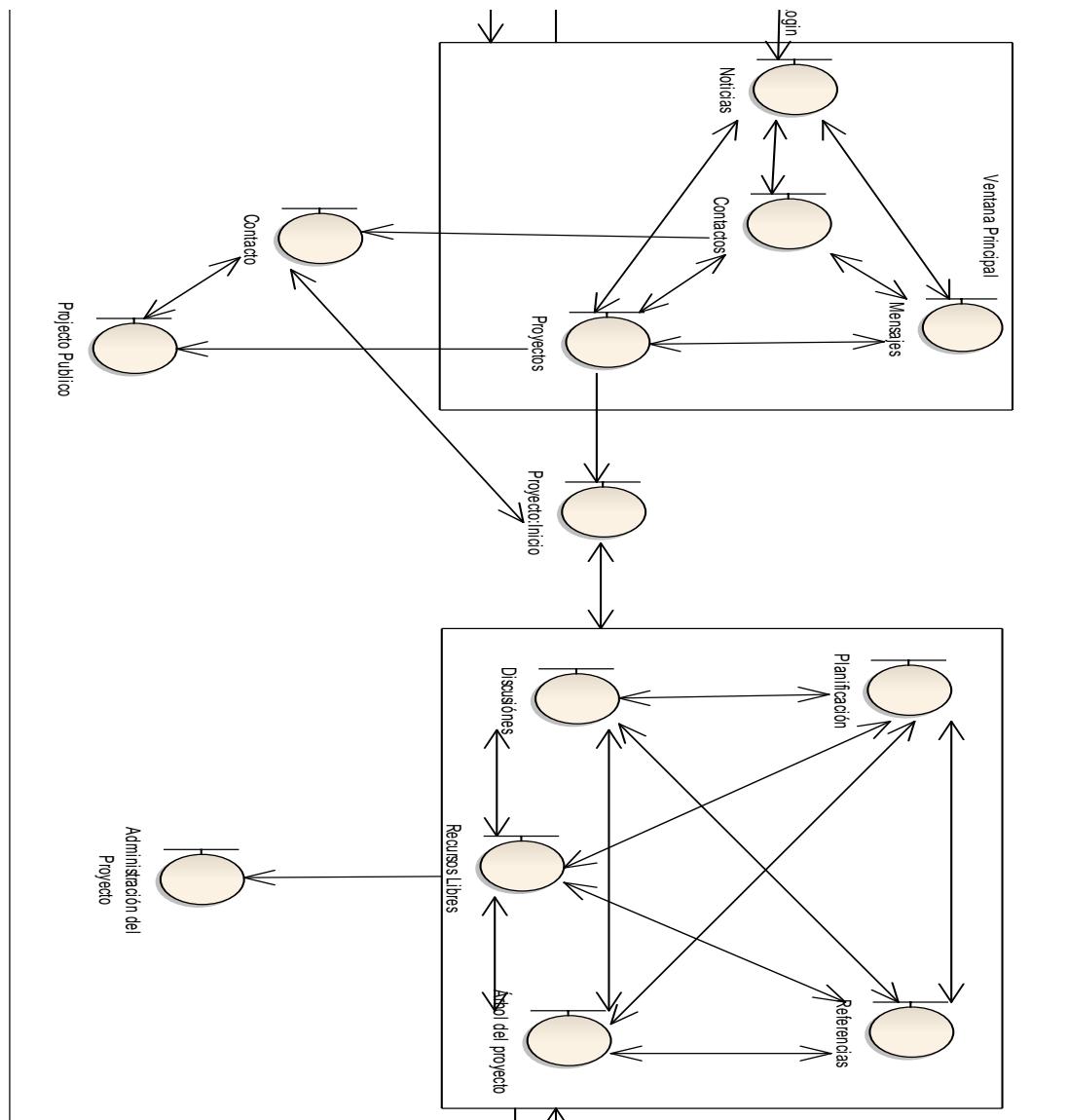


Figura 70: Diagrama de navegabilidad

5.5 Especificación del Plan de Pruebas

En esta sección se especifica el plan a seguir para el diseño y ejecución de las pruebas de la aplicación y sus funciones. Describe por tanto, el tipo de pruebas que se han de realizar y sobre qué componentes del sistema se aplicarán.

A continuación se especifica los tipos de pruebas que se realizarán, y los componentes que serán probados por cada una, también se describen detalles concretos de cada tipo de prueba.

5.5.1 Pruebas Unitarias

Una parte importante del sistema es la gestión de las entidades que se describen en el apartado 5.3 "Clases del Análisis" por tanto es necesario diseñar y realizar una prueba unitaria por cada clase del modelo del dominio. Debido a que estas clases se corresponden con entidades cuyos datos se almacenan en una base de datos relacional, será necesario que todas las pruebas unitarias incluyan al menos, un método que pruebe cada una de las operaciones CRUD (consultar, crear, actualizar y eliminar) por separado para comprobar que todas las clases del modelo estén correctamente construidas y correctamente relacionadas entre sí.

Se deberá utilizar alguna herramienta de automatización de test, como JUnit, para realizar este tipo de pruebas.

Dado que los componentes del sistema serán ejecutados en el seno de un contenedor de aplicaciones, se deberá utilizar alguna herramienta que permita ejecutar las pruebas en dicho contenedor como *Arquillian* o bien usar un framework de simulación para simular estas pruebas.

También sería conveniente utilizar alguna herramienta como DBunit para garantizar la estabilidad de los datos antes y después de las pruebas.

5.5.2 Pruebas de Integración

La implementación de los escenarios de cada caso de uso requiere de la comunicación de varias clases para cumplir con la funcionalidad especificada. Uno de los requisitos no funcionales es que la arquitectura del sistema siga el patrón n-tiers, esto hace fácil predecir un diseño en capas, por lo que la forma que tendrán las clases de comunicarse durante la ejecución de los escenarios será a través de las interfaces que le proporciona la capa inferior.

Será necesario por tanto realizar pruebas de integración que comprueben la correcta comunicación entre todas las capas para dar funcionalidad a los escenarios del sistema. De este modo habrá un test de integración por cada escenario de los casos de uso que se contemplan en el sistema. A continuación se listan los escenarios a probar y se comentan los puntos críticos. Estos escenarios deben ser detallados en el diseño especificando todos los casos de prueba, las entradas y las salidas esperadas.

- **Caso de uso 1.1 Añadir Usuario:** Se debe probar el caso de añadir un usuario al sistema, valorando diferentes alternativas.
- **Caso de Uso 1.2 Modificar Usuario:** Se debe probar la modificación de un usuario, valorando diferentes alternativas, tanto positivas como negativas. En este caso se debe cuidar mucho la prueba negativa.
- **Caso de Uso 1.3 Eliminar Usuario:** Se debe probar la funcionalidad de eliminar un usuario, valorando con mucho cuidado los diferentes estados en los que puede estar un usuario.
- **Caso de Uso 2.1 Buscar Usuario:** Se debe probar la funcionalidad de buscar a un usuario, presentando cuidado a los resultados de la búsqueda en función de los parámetros de entrada y la salida.
- **Caso de Uso 2.1 Añadir Usuario como Contacto:** Se debe probar la creación de una relación entre usuarios prestando especial cuidado al estado de ambos usuarios, si existen peticiones o no, o si estos pudiesen ser o no ya contactos.
- **Caso de Uso 2.1 Ver perfil de contacto:** Se debe probar la funcionalidad que obtiene la información de un contacto para mostrarla a un usuario.
- **Caso de Uso 2.1 Comunicarse con Contacto:** se deben probar las vías de comunicación entre los contactos.
- **Caso de Uso 2.1 Crear Proyecto:** Se debe probar la creación de un nuevo proyecto, prestando especial atención a la información y contactos que se aporten.
- **Caso de Uso 3.2.1 Incluir Excluir/Contactos:** Se debe probar la funcionalidad relativa a la gestión de los miembros del proyecto.
- **Caso de Uso 3.2.2 Cambiar gestor:** Se debe probar la funcionalidad de cambiar el gestor del proyecto.
- **Caso de Uso 3.2.3 Publicar Proyecto:** Se debe probar la funcionalidad de publicar un proyecto, prestando mucha atención al estado del proyecto.
- **Caso de Uso 3.2.1 Eliminar Proyecto:**
- **Caso de Uso 3.2.1 Planificar Proyecto:** Se debe probar la funcionalidad de eliminar un proyecto, valorando diferentes casos que tengan en cuenta si un proyecto tiene miembros, o si hay invitaciones a él.
- **Caso de Uso 3.4.1 Crear Discusión:** Se probara la funcionalidad de la creación de una discusión cotejando los valores aportados así como el estado de la discusión creada.
- **Caso de Uso 3.2.1 Votar en Discusión:** Se debe probar la funcionalidad de votar en una discusión.

- **Caso de Uso 3.5 Abandonar Proyecto:** Se debe probar la funcionalidad de abandonar un proyecto, prestando especial atención al nuevo gestor.
- **Caso de Uso 4.1 Crear Recurso:** Se debe probar la funcionalidad de la creación de un recurso.
- **Caso de Uso 4.2 Modificar Recurso:** Se debe probar la modificación de un recurso comprobando que este es modificado.
- **Caso de Uso 4.3 Eliminar Recurso:** Se debe comprobar la funcionalidad de eliminar recursos, comprobando con cuidado el nuevo estado del árbol del recurso.

5.5.3 Pruebas del Sistema

Las pruebas del sistema comprobarán los casos de uso detallados para las pruebas de integración sobre el sistema en funcionamiento.

Con esto se pretende probar que el sistema en su conjunto funciona de una forma adecuada sin comportamientos extraños fruto de agentes externos como los navegadores o la comunicación cliente servidor.

5.5.4 Pruebas de Usabilidad

El sistema desarrollado debe ser usable, de modo que un usuario con los conocimientos habituales del sobre el uso de ordenadores y aplicaciones web sea capaz de aprender a manejarse dentro de la aplicación en un corto periodo de tiempo. Para ello se han de realizar pruebas de usabilidad del sistema sobre usuarios reales con el perfil especificado en el apartado 5.2.1.5 “Requisitos No funcionales”, estas pruebas de usabilidad serán detalladas en el apartado 6.6 “Especificación Técnica del Plan de Pruebas”, y los resultados de su ejecución serán mostrados en el Capítulo 8 “Desarrollo de las Pruebas”.

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

Este apartado expone mediante una visión top down la arquitectura del sistema, partiendo desde la arquitectura de más alto nivel hasta llegar al límite con el diseño de detalle. Para esto se usan diagramas de componentes, de despliegue y de paquetes.

Empezaremos por describir los patrones arquitectónicos que sigue el sistema. Estos patrones condicionan la arquitectura de todo el sistema al tiempo que ofrecen una forma rápida tanto de definirla como de diseñarla.

Se expondrán, mediante una serie de modelos, los componentes que conforman el sistema así como su interacción con componentes externos que se han incorporado al sistema con el fin de reutilizar funcionalidades.

Una vez estudiados los componentes del sistema, veremos la arquitectura de despliegue del sistema, esta refleja cómo se distribuirán los componentes sobre unidades de despliegues.

Finalmente veremos cómo se organiza el código de los componentes en paquetes con el fin de obtener una perspectiva de la arquitectura de este.

6.1.1 Patrones Arquitectónicos

El sistema sigue 2 patrones arquitectónicos que aunque parezcan solaparse o ser lo mismo ni se solapan ni son lo mismo, sino que describen la arquitectura desde dos puntos de vista distintos y perfectamente compatibles, Arquitectura de Despliegue y Arquitectura de Software, dicho de otra forma y especificando más, el sistema sigue el patrón de despliegue en N niveles (más conocido como N- Tiers) mientras que la arquitectura del Software sigue el patrón Layers.

6.1.1.1 Patrón N Tiers

Este patrón contempla la separación en niveles físicos de los componentes del sistema. El número de niveles es variable, siendo habitual tener arquitecturas de 2 niveles (Figura 71), de tres niveles (Figura 72) y de hasta 4 niveles (Figura 73). Este sistema define 4 niveles como muestra la Figura 73.

Sobre este patrón de distribución en niveles, encaja a la perfección una arquitectura de software en capas.

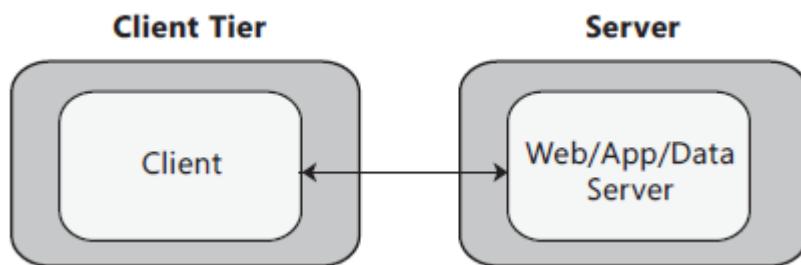


Figura 71: Patrón N Tiers – Dos niveles

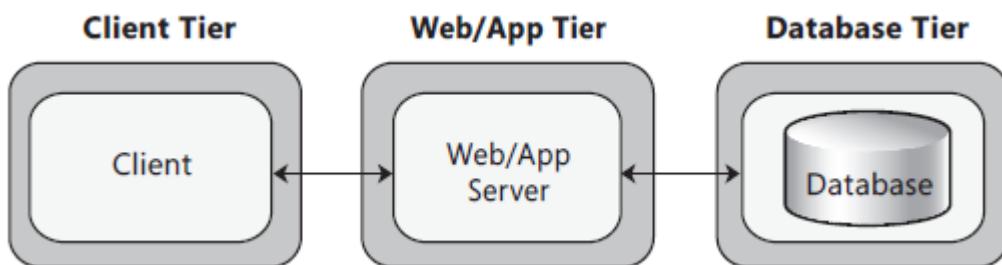


Figura 72: Patrón N Tiers – Tres niveles

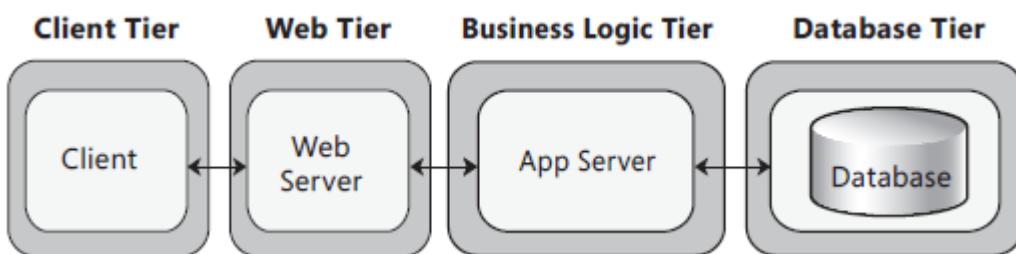


Figura 73: Patrón N Tiers – Cuatro niveles

6.1.1.2 Patrón Layers

El patrón *Layers* aporta precisamente una arquitectura de software estratificada, dicho de otra forma, una estructura en capas.

La estructura más típica de este patrón es la mostrada en la Figura 74, que define una capa de software para la presentación del sistema, otra para las reglas de negocio y una capa que se encargue de todo lo relativo a la persistencia. Estas capas deben estar separadas por fachadas y las capas inferiores nunca deben invocar a las superiores. Esta estructura encaja perfectamente sobre la arquitectura de despliegue en niveles.

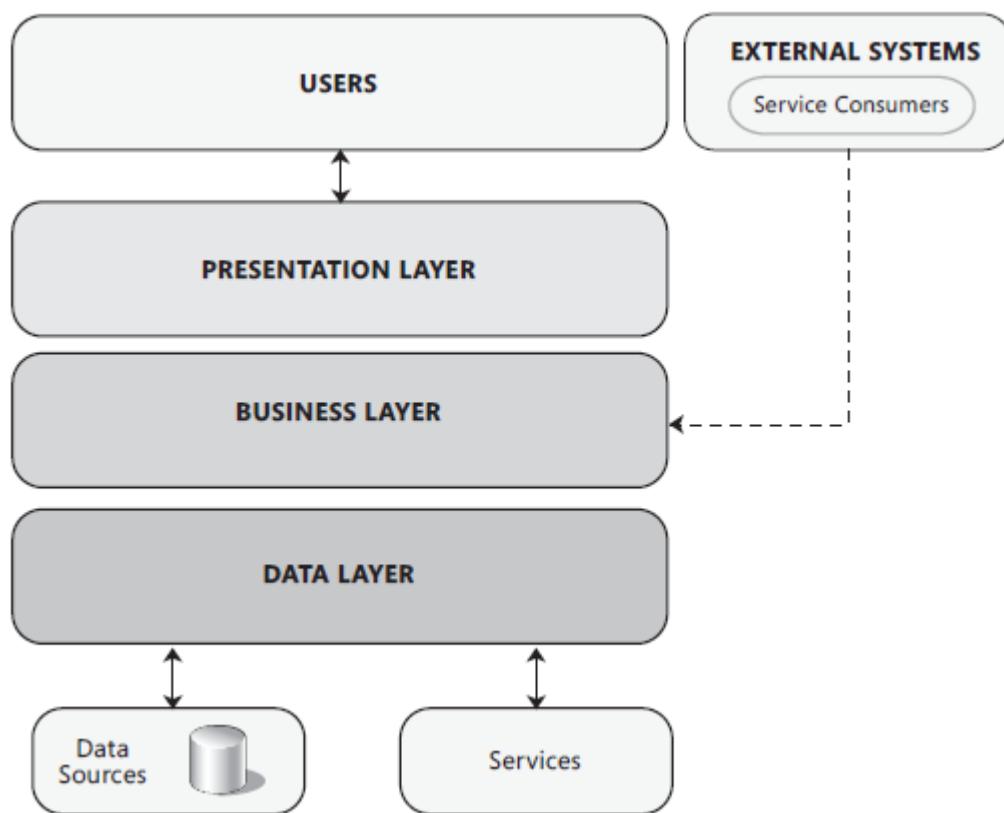


Figura 74: Patrón Layers

6.1.2 Diagramas de Componentes

Este apartado muestra los componentes del sistema así como sus dependencias con otros componentes.

Un primer diagrama (Figura 75) muestra los componentes del sistema sin dependencias, con el fin de que se identifiquen bien para la comprensión del resto de modelos.

En un segundo diagrama se muestran los componentes del sistema con todas sus dependencias.

Finalmente se ha añadido un último diagrama de componentes que muestra las relaciones entre las dependencias del sistema y sus especificaciones de la JCP. Este diagrama aunque no necesario, permite identificar de forma gráfica a los componentes definidos como dependencias del sistema.

6.1.2.1 Componentes Internos del Sistema

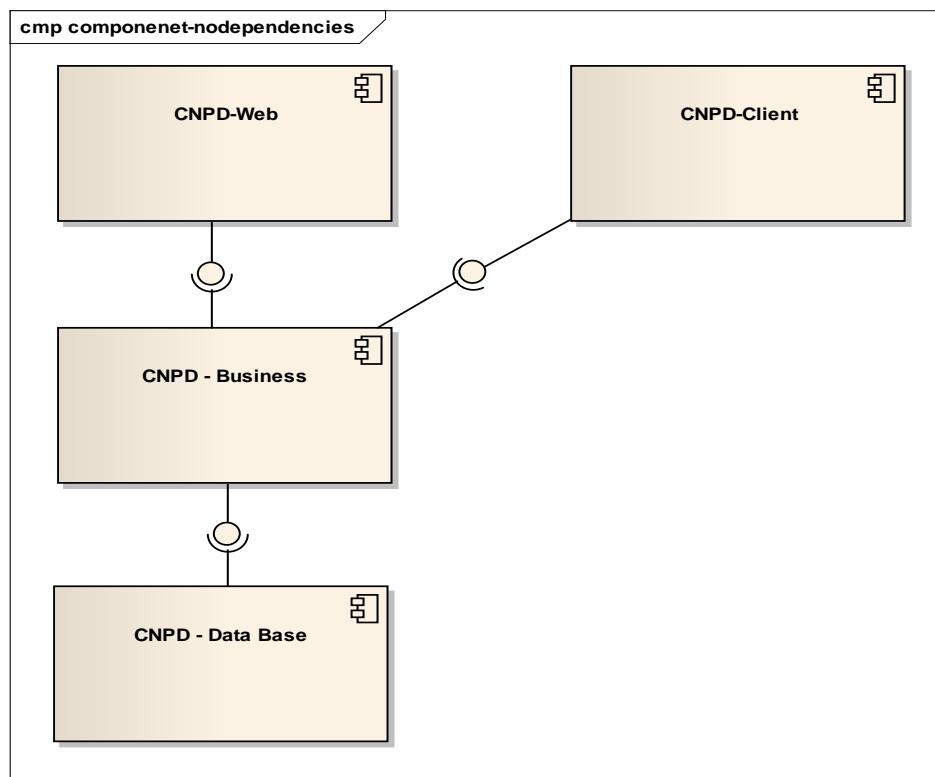


Figura 75: Diagrama de componentes internos

El diagrama de la Figura 75 muestra los componentes, desarrollados íntegramente para este sistema, se da una perspectiva de estos al margen de sus dependencias con el fin de que queden perfectamente identificados.

Las siglas *CNPD* que dan comienzo a los nombres de los cuatro componentes, significan en inglés *Collaborative Network for Project Development* que en castellano se traduce por *Red Colaborativa para el Desarrollo de Proyectos*, título de este proyecto. A continuación se detalla cada uno de estos componentes.

6.1.2.1.1 CNPD-Data Base

Este componente modela al conjunto de datos del sistema, que conforman dentro del servidor de bases de datos, la base de datos del sistema. Dicha base de datos cuenta con más de veinte tablas que modelan el sistema de acuerdo al modelo relacional de *CODD*.

Este componente es generado mediante un sistema de mapeo objeto relacional, que se encarga de mapear nuestros objetos y las relaciones definidas entre ellos a un sistema relacional, este mapeo se podría ver de una forma simplificada como el paso a tablas de un modelo entidad-relación.

6.1.2.1.2 CNPD-Business

Este componente contiene tanto las reglas de negocio como el acceso a datos, que encajan con las capas de persistencia y de negocio definidas por la arquitectura lógica del sistema.

Su nombre (Business) se debe que el componente representa el nivel de negocio o Business Tier según la arquitectura de despliegue definida para el sistema (N Tiers). Esto implica que además de encapsular la persistencia y las reglas de negocio, se adapta a las interfaces ofrecidas por un contendor de forma que cuente con la infraestructura adecuada para soportar invocaciones remotas concurrencia y transaccionalidad.

6.1.2.1.3 CNPD-WEB

Al igual que el componente anterior, este toma su nombre de la arquitectura de despliegue elegida para el sistema, este componente lleva consigo todo lo relativo a la Web-Tier de acuerdo al patrón de despliegue *N- Tiers*. Por otro lado este componente además lleva consigo todo lo relativo a la capa de presentación definida por el *Patrón Layers* usado para definir la arquitectura de la mayor parte del software del sistema.

Esta capa de presentación implementa una aplicación web enriquecida, desarrollada con tecnología GWT. Esta aplicación se divide forzosamente en 3 partes, código que GWT compilará a Java Script, código que se ejecutara en el servidor, y código compartido, que será usado en las comunicaciones entre cliente y servidor, por lo que será compilado también a Java Script.

Desde el punto de vista estructural es importante recalcar el hecho de que esta capa como capa de presentación, define 3 capas internas, una capa de vistas, una de lógica de presentación y un modelo propio especializado para tareas de presentación y de comunicación cliente servidor.

6.1.2.1.4 CNPD – Client

Este componente es un cliente de escritorio desarrollado sobre Java SE que ofrece una interfaz de usuario con fines administrativos. Este componente carece de lógica de negocio, delega todas las responsabilidades en el componente CNPD-Business.

6.1.2.2 Componentes del Sistema

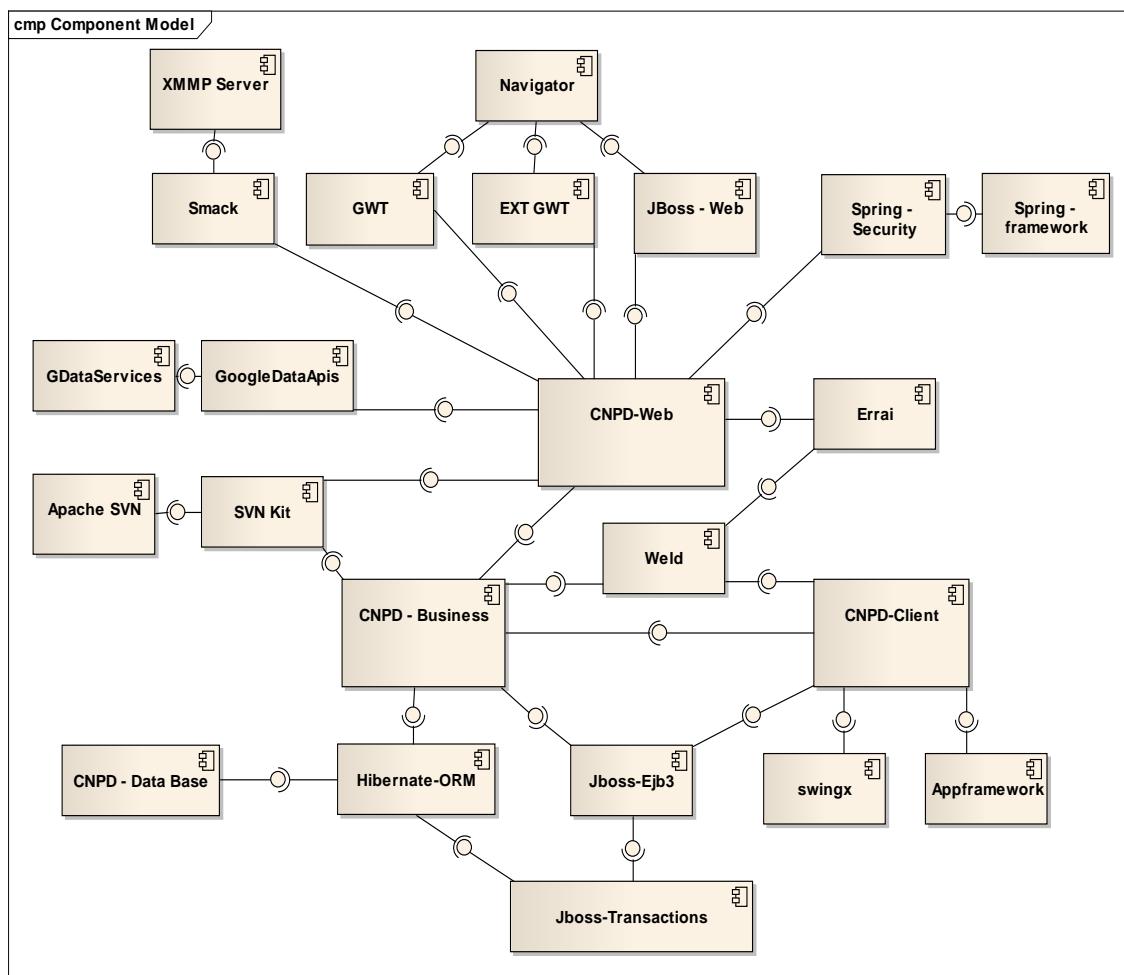


Figura 76: Diagrama de componentes

El diagrama de componentes de la Figura 76 muestra los componentes del sistema con todas sus dependencias, es a mi modo de ver uno de los principales exponentes de la arquitectura del sistema.

Este diagrama refleja las relaciones de los componentes desarrollados para este sistema con el resto de componentes que se han incorporado, así como las relaciones entre estos últimos.

Es muy importante comprender que gran parte de estos componentes vienen incluidos dentro del servidor de aplicaciones en el cual nuestros componentes son desplegados, pero no por ello dejan de tener suficiente autonomía y funcionalidad para ser considerados como tal.

Con el fin de facilitar la identificación de los componentes sobre los que está construido el sistema, se ha añadido el diagrama de la Figura 77. Este diagrama muestra algunos de los componentes del sistema relacionados con la especificación que implementan, en los casos que procede.

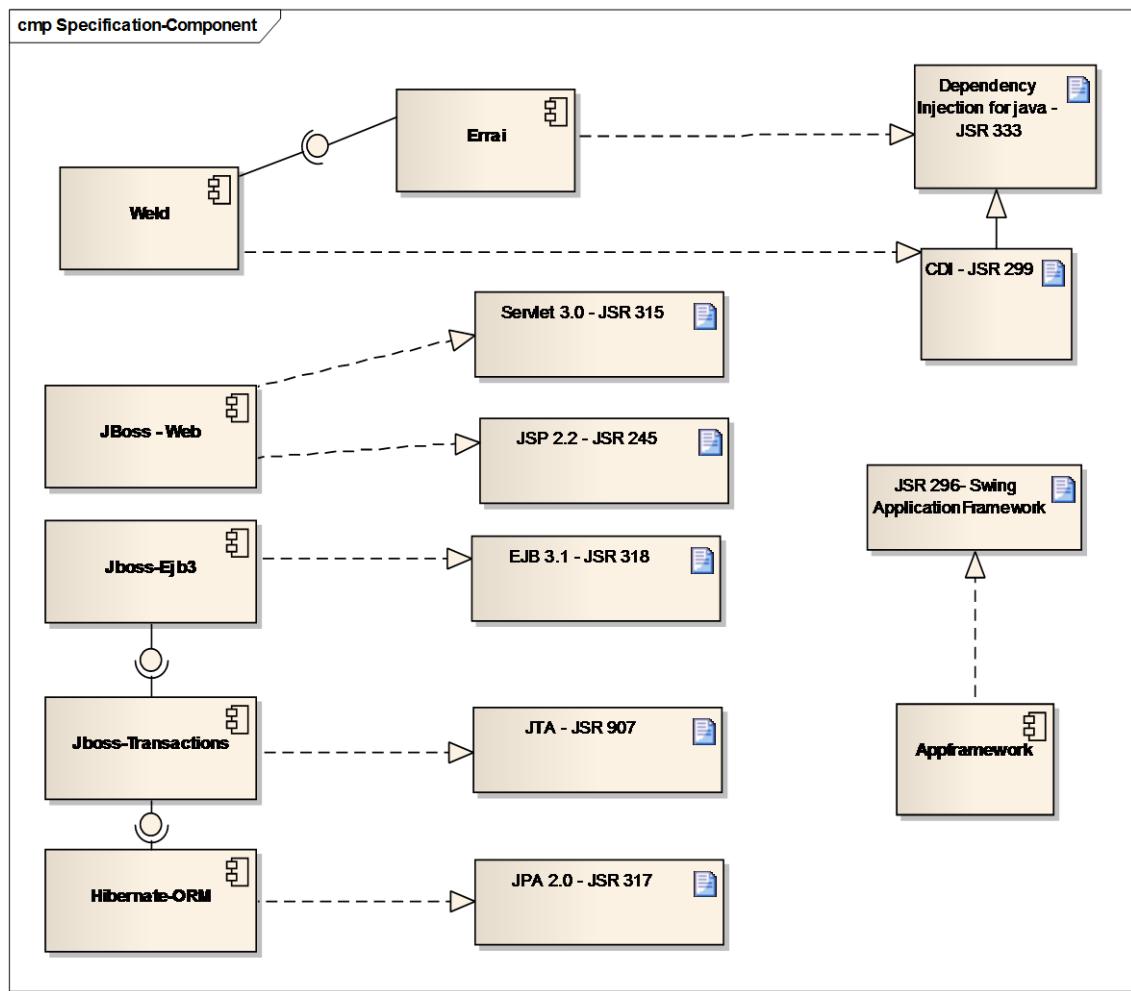


Figura 77: Especificaciones de los componentes del sistema

A continuación se describirá la funcionalidad que aportan los distintos componentes al sistema.

6.1.2.2.1 Hibernate ORM

Popular framework para el Mapeo Objeto relacional que implementa al completo la especificación JPA 2.0. El servidor de aplicaciones JBoss AS7 ofrece una distribución de Hibernate pre configurado para ser el proveedor de persistencia por defecto.

Además de implementar la especificación, Hibernate aporta varias extensiones, que en algunos casos son muy útiles con vistas a resolver algunos problemas que la especificación se ha dejado.

6.1.2.2.2 JBoss-EJB3

JBoss EJB3 es la implementación de la especificación EJB 3.1 (JSR 318) que incluye el Servidor de aplicaciones JBoss AS 7.

La funcionalidad que este componente provee al sistema es toda la aquella garantizada por la especificación.

6.1.2.2.3 JBoss-Transactions

Proyecto de JBoss que entre otras muchas cosas incluye la implementación de la especificación JTA 1.1 (JSR 907), que viene integrada en el servidor de aplicaciones JBoss AS7. Provee al sistema de todas las garantías del JSR que implementa.

6.1.2.2.4 Weld

Es la implementación de referencia del JSR 299 (Contextos e Inyección de dependencias), es la tecnología predilecta de la plataforma JEE 6 para la inyección de dependencias, que es como cabe suponer la funcionalidad que de este componente se consigue en el sistema.

Esta tecnología ha sido abordada en los aspectos teóricos del proyecto, por lo que no se entrara en detalles.

El servidor de aplicaciones nos provee mediante un módulo de JBoss AS7 de una distribución de Weld. Merece la pena destacar que en el sistema se hace uso de Weld tanto dentro del contenedor Web como desde el contenedor del cliente, en este caso sí que tenemos que añadir explícitamente el módulo, concretamente una distribución especial de Weld para JSE.

6.1.2.2.5 Apache SVN

Este componente representa una distribución de Apache Subversión instalada para la gestión de los ficheros del sistema.

6.1.2.2.6 SVN KIT

Este componente facilita al sistema el acceso a la distribución de Apache Subversion que gestiona los ficheros del sistema.

6.1.2.2.7 Google Data Services

Este componente modela una parte de los servicios de Google que son expuestos para ser accedidos vía HTTP, como pueden ser Google Docs, Google Contacts o Google Analytics.

6.1.2.2.8 Google Data APIs

Google distribuye una serie de APIs que encapsulan el acceso a sus servicios, permitiendo a los desarrolladores abstraerse del protocolo usado para el intercambio de información y de la invocación de los servicios.

6.1.2.2.9 XMPP Server

Software que juega el rol de servidor, de acuerdo al protocolo XMPP, en este caso se ha usado el servidor XMPP del servicio de Google Talk. En este sistema se usa como servidor para el servicio de mensajería instantánea.

6.1.2.2.10 Smack

Bibliotecas que ofrecen un API para comunicarse con servidores XMPP, se usa para comunicarse con el servidor de Google Talk y ofrecer la funcionalidad de mensajería instantánea.

6.1.2.2.11 JBoss Web

Implementación de las especificaciones Servlet y JSP basada en Apache Tomcat que conforma el contendor web con él cuenta el JBoss AS7. Este proyecto de JBoss que en tiempos pasados fue un servidor web autónomo, se ha dejado de distribuir como tal para pasar a ser exclusivamente un componente del AS7.

6.1.2.2.12 GWT

Este componente ofrece al sistema toda la infraestructura para el desarrollo de la aplicación Web enriquecida, así como un conjunto de componentes gráficos para la construcción de las vistas.

6.1.2.2.13 EXT GWT

Ofrece al sistema un conjunto de componentes gráficos de más alto nivel que los de GWT, al tiempo que son perfectamente compatibles. La mayor parte de las vistas de la aplicación está desarrollada con estos componentes.

6.1.2.2.14 Errai

Este componente es de gran importancia para el sistema, consiste entre otras cosas en un bus de mensajes entre cliente y servidor, que permite hacer un diseño más sofisticado de la aplicación. Sobre dicho bus Errai nos ofrece una implementación parcial del JSR 299 en cliente y un mecanismo simplificado de invocaciones al servidor.

Es usado exclusivamente por la aplicación web y se nutre especialmente de la inyección de dependencias, los eventos y las invocaciones al servidor mediante proxies.

6.1.2.2.15 Spring Security

Framework que simplifica enormemente las tareas de autenticación y autorización, permitiendo su configuración de forma declarativa y su extensión para obtener una autenticación más específica, reutilizando los mismos mecanismos de configuración.

6.1.2.2.16 Spring Framework

Framework para el desarrollo de aplicaciones Java, con especial soporte para la inyección de dependencias y Orientación a Aspectos, funcionalidades sobre las que se construyen muchas otras tales como acceso a datos, gestión de transacciones o desarrollo de aplicaciones web (tanto MVC clásicas como enriquecidas basadas en Flex) y frameworks como Spring Batch, Spring Web Flow , Spring Integration o Spring Security, en este proyecto se usa exclusivamente para la configuración y extensión de Spring Security.

6.1.2.2.17 Swingx

Conjunto de componentes visuales algo más avanzados que los provistos por Swing y construidos sobre estos. Es usado por la aplicación de administración. Para la construcción de la interfaz de usuario.

6.1.2.2.18 Appframework

Implementación del JSR 296 (Swing Application Framework) que ofrece un conjunto de funcionalidades típicas, que se hacen necesarias en el desarrollo de aplicaciones basadas en Swing.

6.1.3 Diagramas de Despliegue

Como se adelantó en apartados anteriores el sistema sigue un patrón de despliegue en n niveles. Con el fin de ilustrar de forma precisa las unidades de despliegue que componen la arquitectura, así como la distribución de los componentes sobre dichas unidades, se muestran diagramas de despliegue con distintos niveles de detalles.

6.1.3.1 Diagrama de Despliegue de Alto Nivel

Un primer diagrama (Figura 78) muestra los distintos niveles de la arquitectura desde una perspectiva más abstracta, ocultando los componentes del sistema y dejando al descubierto solamente las piezas de software de más alto nivel del sistema.

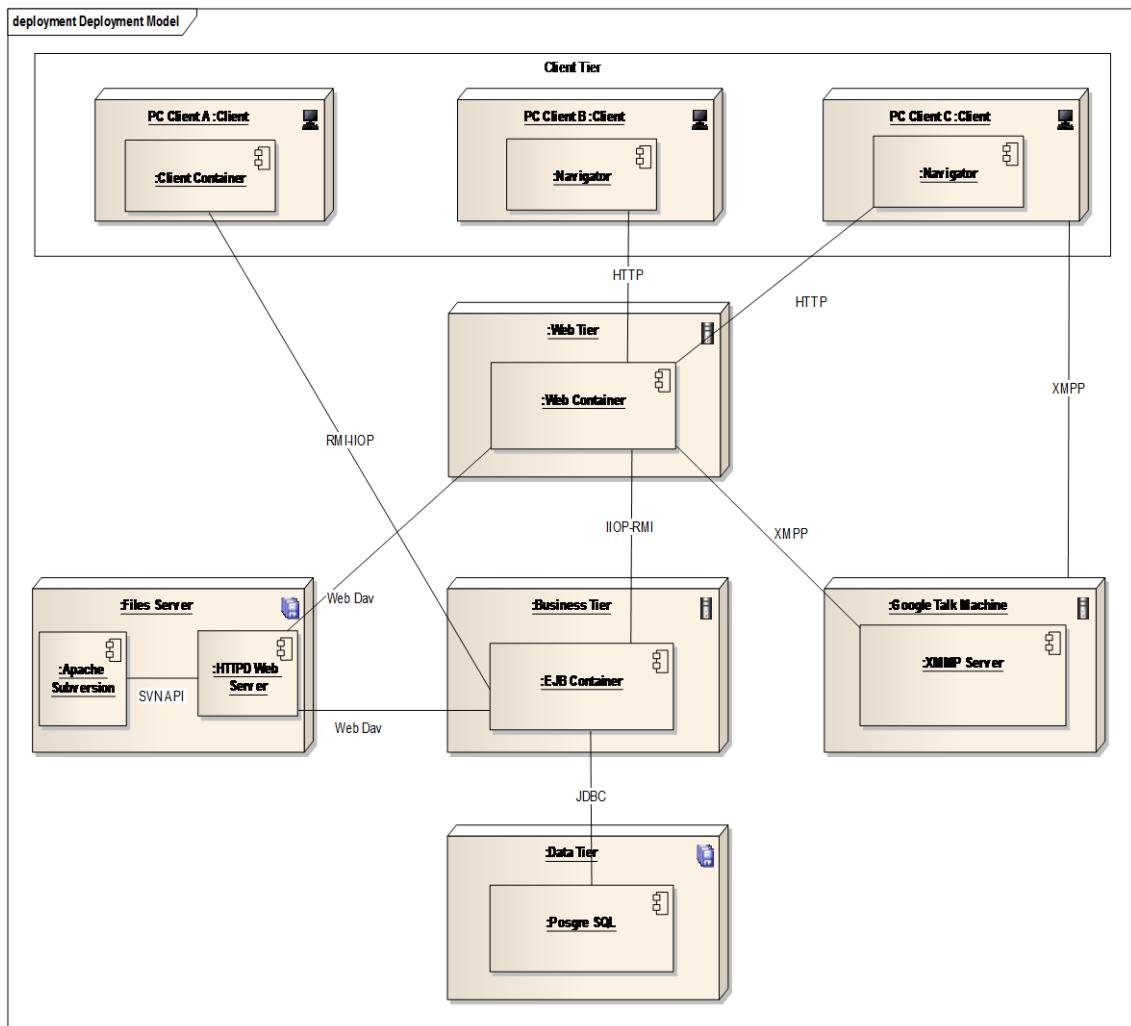


Figura 78: Diagrama Despliegue de Alto Nivel

A continuación se explican los distintos nodos y componentes de este diagrama, así como su ubicación dentro del patrón de despliegue N Tiers.

6.1.3.1.1 Data Tier

Modela el nivel de datos según el patrón N Tiers, en este nivel se pueden tener desde servidores de bases de datos hasta otros sistemas de integración empresarial más heterogéneos, en este caso solamente tenemos un SGBD, concretamente PostgreSQL.

6.1.3.1.2 PostgreSQL

Sistema Gestor de Base de Datos usado para persistir los datos del sistema, se ha explicado con más detalles en los aspectos teóricos.

6.1.3.1.3 Business Tier

Modela el nivel de negocio, donde descansan los objetos de negocio, bien servidos de transaccionalidad, concurrencia y acceso remoto por parte de un contenedor.

6.1.3.1.4 EJB Container

Contenedor de Enterprise Java Beans que alberga los objetos de negocio garantizándoles todos los servicios que vienen recogidos en el JSR 318.

6.1.3.1.5 Web Tier

Nivel donde descansaran tanto los servlets como los JSPs, be provistos por el contenedor Web.

6.1.3.1.6 Web Container

Contenedor de Servlet y JSP, que albergará la aplicación web del sistema ofreciéndoles todos los servicios que recoge la especificación Servlet.

6.1.3.1.7 Files Server

Servidor que cuyo fin será el de albergar los ficheros del sistema.

6.1.3.1.8 Apache Subversion

Instalación de apache Subversión.

6.1.3.1.9 HTTD Web Server

Instalación del conocido servidor Web Apache, que con la configuración adecuada sirve de punto de acceso a la instalación de apache Subversion.

6.1.3.1.10 Google Talk Machine

Servidor donde se hospeda el servicio de Google Talk.

6.1.3.1.11 XMMP Server

Servidor del protocolo XMMP, usado para ofrecer el servicio de Google Talk.

6.1.3.1.12 Client Tier

Nivel donde descansan las aplicaciones cliente y las páginas HTML dinámicas generadas en el servidor.

6.1.3.1.13 Client Container

Software necesario para consumir objetos de negocio desplegados en el nivel de negocio.

6.1.3.1.14 Navigator

Navegador del cliente encargado de comunicarse con la Web Tier y renderizar las páginas HTML generadas por el servidor.

6.1.3.2 Diagrama de Despliegue de Bajo Nivel

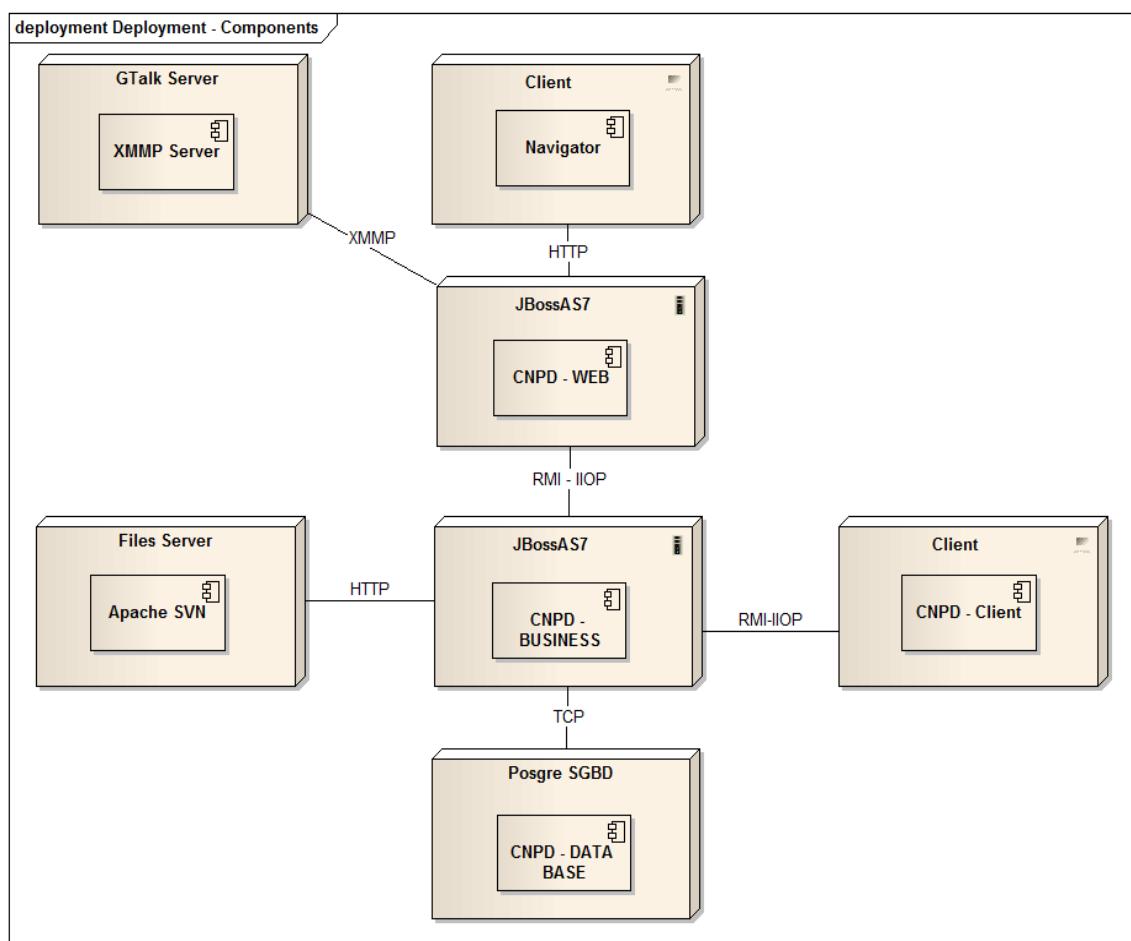


Figura 79: Diagrama de despliegue de Bajo Nivel

En un segundo diagrama de despliegue (Figura 79), haciendo una especie de zoom sobre el primero, se puede ver como se distribuyen los componentes desarrollados para el sistema sobre unidades de despliegues más concretas.

En este caso los nodos son los componentes del diagrama anterior (Figura 78) mientras que los componentes han sido extraídos del diagrama de componentes (Figura 75), de esta forma conseguimos relacionar el diagrama de componentes con el de despliegue.

6.1.4 Diagramas de Paquetes

Con esta sección se pretende mostrar la estructura del software mediante el estudio de sus paquetes, recordemos que como se vio en el apartado 6.1.1 se ha seguido un patrón arquitectónico en capas sobre distintos niveles físicos.

Debido al número de paquetes del sistema, se hace relativamente difícil comprenderlo mediante un solo diagrama, por lo que se han realizado diagramas de paquetes para distintos niveles de detalles.

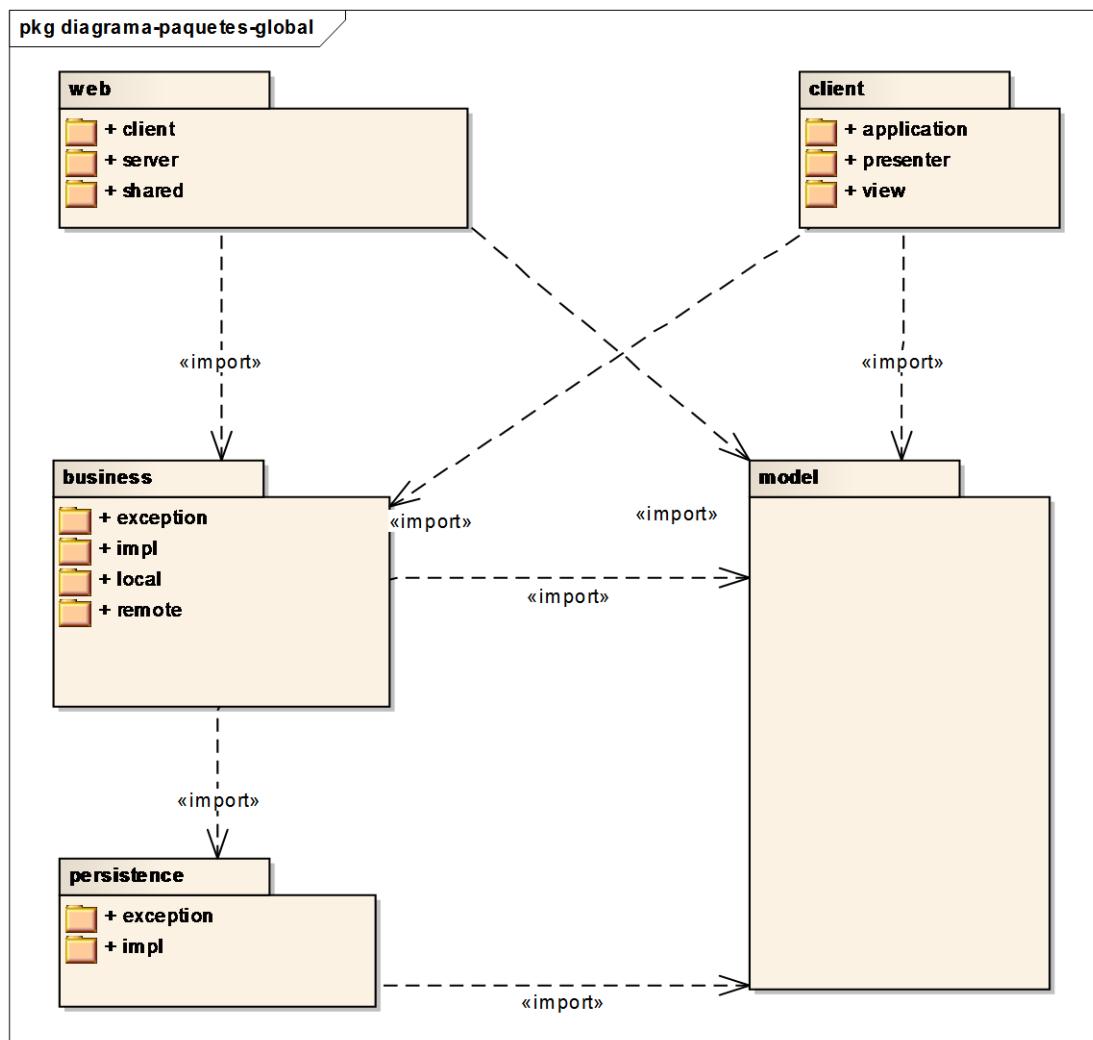


Figura 80: Diagrama de paquetes global

El diagrama de la Figura 80 muestra una visión global de los paquetes del sistema, así como un primer nivel de los paquetes dentro de cada uno de ellos.

A continuación se detallan los paquetes, haciendo énfasis en sus responsabilidades y añadiendo algún diagrama de más detalle donde proceda.

6.1.4.1 Paquete persistence

Este paquete encapsula todo lo relativo a la persistencia, define interfaces para el acceso a los datos de las distintas entidades del sistema, así como algunas excepciones para indicar problemas específicos de este sistema en la capa de persistencia.

Las clases de acceso a datos permanecen ocultas tras interfaces, cuya implementación se encuentra en el paquete *impl*, esta implementación está fuertemente ligada a la tecnología de acceso a datos que se ha usado, Java Persistence API 2.0.

Se corresponde con la capa de acceso a datos definida por el patrón *Layers*.

Este paquete tiene una dependencia con el paquete *model*, ya que define el acceso a los datos de las entidades que el paquete *model* modela.

6.1.4.2 Paquete model

Este paquete contiene las clases que modelan las entidades del sistema, define las clases que modelan el sistema. Las clases de este paquete viajan por todas las capas y niveles del sistema, por lo que son incluidas desde todos los paquetes, además están fuertemente ligadas a la tecnología de acceso a datos.

6.1.4.3 Paquete business

Este paquete incluye toda la lógica de negocio y expone fachadas para todo el nivel de negocios, dicho de otra forma, cualquiera de los otros dos niveles que quiera usar los objetos del nivel de negocio, tendrá que acceder a través de las fachadas expuestas por el paquete *business*.

Se corresponde con la capa de negocio definida por el patrón *Layers*, a lo que debe su nombre, y tiene dependencias con el paquete *persistence* y con el paquete *model*.

Está fuertemente ligado a la tecnología EJB ya que le debe a esta la transaccionalidad y el acceso remoto a sus fachadas, entre otras cosas.

Debido a que es un paquete más complejo que los anteriores, tiene una estructura interna algo más compleja (Figura 81).

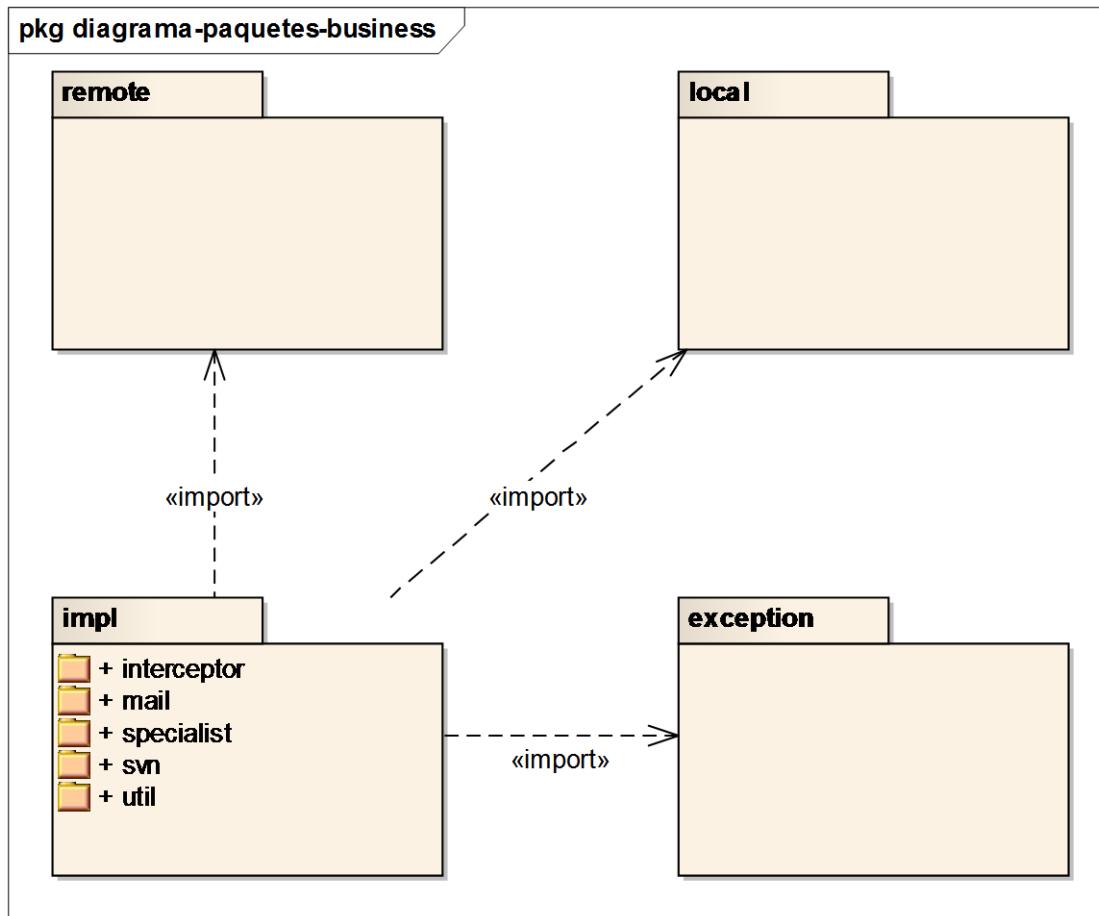


Figura 81: Diagrama paquetes business

6.1.4.3.1 Paquete Remote

Define las interfaces para acceder a las fachadas de negocio mediante invocaciones remotas.

6.1.4.3.2 Paquete Local

Define las interfaces para acceder a las fachadas de negocio de forma local.

6.1.4.3.3 Paquete impl

Contiene todas las clases que implementan las reglas de negocio, así como otras de infraestructura.

6.1.4.3.4 Paquete Exception

Este paquete define una jerarquía de excepciones propia de la capa de negocio.

6.1.4.4 Paquete web

Este paquete contiene todo lo relativo a la *web tier*, que en este caso coincide con la capa de presentación del sistema que consiste en una aplicación web enriquecida.

Es el paquete más complejo del sistema, tanto por el número de clases que alberga como por la complejidad de estas.

Una vista más detallada de este paquete puede verse en la Figura 82: Diagrama de paquetes web

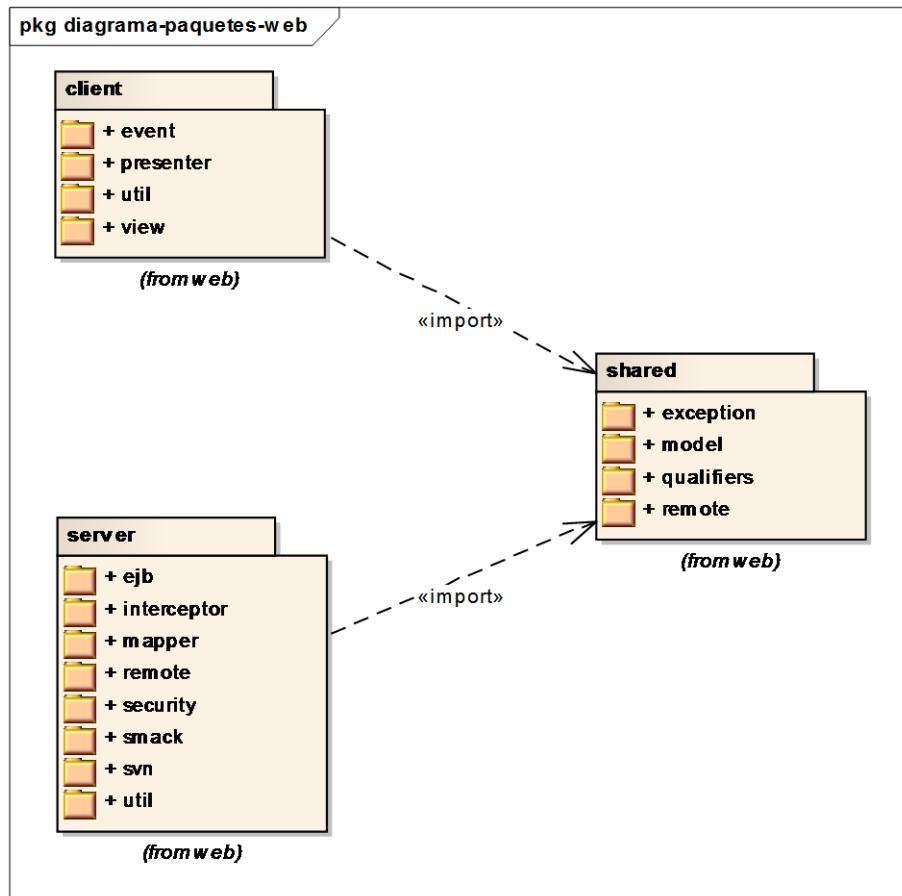


Figura 82: Diagrama de paquetes web

6.1.4.4.1 Paquete server

Contiene todas las clases que se ejecutarán en la parte servidor de la aplicación, define fachadas especializadas en presentación que atienden las peticiones del paquete cliente.

6.1.4.4.2 Paquete shared

Este paquete contiene todas aquellas clases que serán necesarias tanto en la parte cliente como en la parte del servidor.

6.1.4.4.3 Paquete client

Contiene todo el código que será compilado a Java Script para ser ejecutado en el cliente. Las clases de este paquete consisten en vistas y presentadores, clases que atienden peticiones de las vistas e invocan a la parte del servidor.

6.1.4.5 Paquete client

Este paquete contiene las clases de la aplicación de administración, no se debe confundir bajo ningún concepto con el paquete client de la aplicación web.

Las clases de este paquete son en su gran mayoría vistas y presentadores, ya que toda la lógica se encuentra en el paquete negocio.

Una perspectiva más detallada del contenido del paquete client puede verse en la Figura 83.

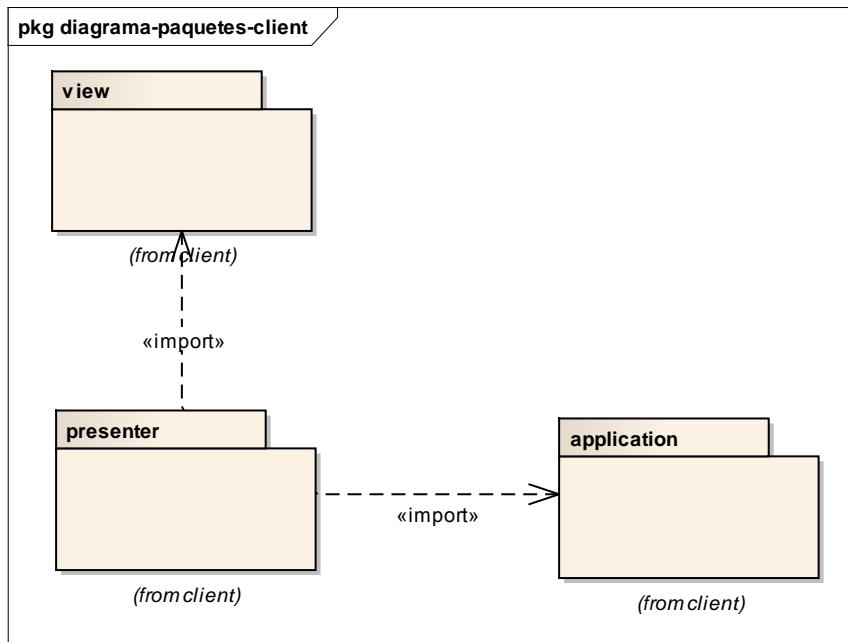


Figura 83: Diagrama de paquetes client

6.2 Diseño de Clases

Una vez sentadas las bases de la arquitectura estudiaremos el diseño de detalle de los distintos componentes de esta. Para ello se verán primero una serie de patrones de diseño que se aplican de manera repetida en el diseño de algunos componentes así como otras técnicas de diseño utilizadas, todo esto con el fin de comprender mejor las clases que se mostrarán para finalizar este apartado.

6.2.1 Patrones de Diseño

En este apartado detallamos los patrones de diseño aplicados, así como las ventajas que se ha obtenido de su aplicación al sistema, el objetivo de este apartado es crear un marco para la comprensión del diseño más que una exposición teórica sobre patrones de diseño.

6.2.1.1 Patrón DAO

Este patrón de diseño define una plantilla para el diseño de la capa de acceso a datos, esta plantilla consta de una interfaz que permita acceder a los datos eliminando cualquier relación con el sistema de persistencia subyacente.

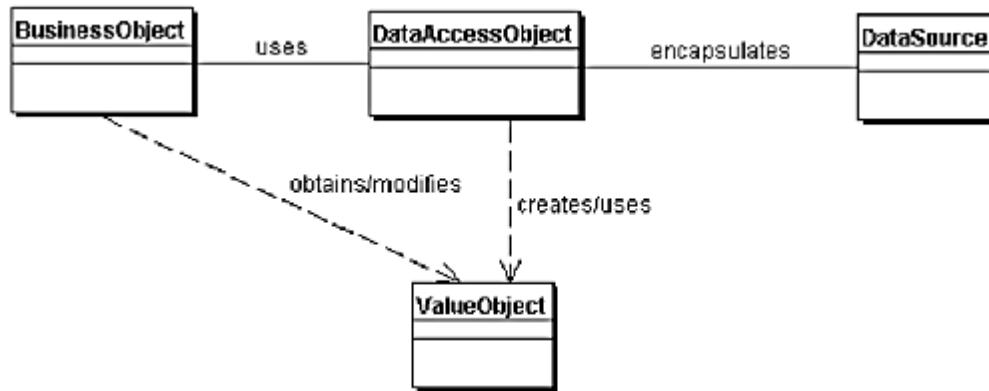


Figura 84: Patrón DAO

Para conseguir esto se propone que sea una factoría la encargada de crear los objetos de acceso a datos, de forma que los objetos de negocio pueden abstraerse totalmente del sistema de persistencia subyacente. En este sistema la factoría es el contendor y los DAO son obtenidos declarando un punto de inyección para la interfaz, sin tener contacto con la implementación. En el apartado 13.4.1 “Ejemplos Capa de Persistencia” pueden verse algunas clases que muestran en detalle cómo se ha aplicado el patrón.

6.2.1.2 Patrón Template Method

Consiste en definir parte de la funcionalidad de un algoritmo en la clase padre.

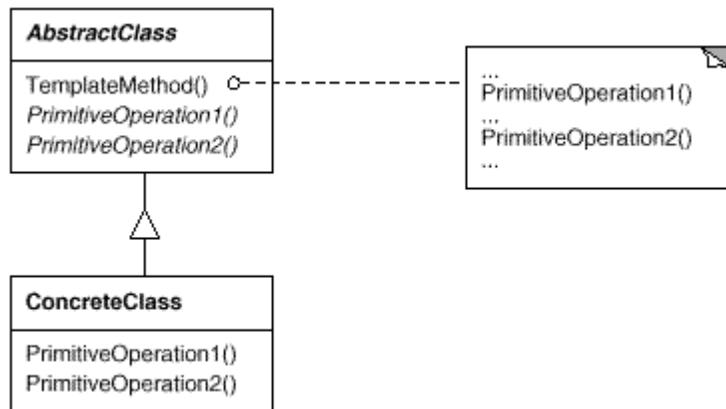


Figura 85: Patrón Template Method

Esto permite que todas las clases hijas puedan utilizar estas funcionalidades sin tener que re implementarlas. En el diseño de este sistema se ha aplicado este patrón para definir operaciones comunes a todos los DAO en una clase padre.

Este patrón está documentado en el libro de patrones de diseño de Eric Gamma [Gamma94].

6.2.1.3 Patrón Singleton

Este patrón consiste en crear una clase de la cual solo exista y pueda existir una instancia. Esto se consigue definiendo todos los constructores de la clase (típicamente uno) como privados, de forma que solo la propia clase pueda instanciarse y sea ella quien permite a los clientes el acceso a su única instancia.

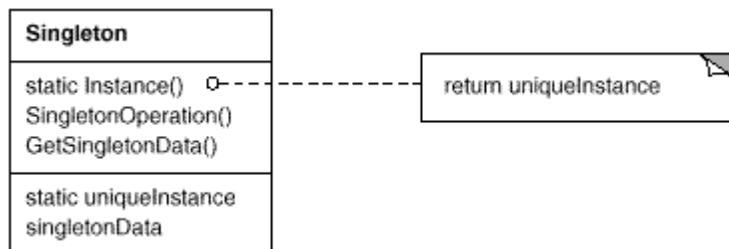


Figura 86: Patrón Singleton

Este patrón viene soportado por las especificación EJB y CDI, y por la implementación (parcial) de CDI para GWT, Errai. En este sistema se ha usado el soporte ofrecido para el patrón por CDI, este soporte consiste en garantizar que para una clase, todas las referencias definidas en los puntos de inyección apuntaran a la misma instancia.

Este patrón está documentado en el libro de patrones de diseño de Eric Gamma [Gamma94].

6.2.1.4 Patrón Factory Method

La motivación de este patrón es abstraer a la clase cliente de un objeto de la creación de este, extremadamente útil para ocultar la implementación de una interfaz.

La especificación CDI ofrece una implementación de este patrón, que consiste en anotar un método para que el contendor inyecte el objeto retornado por ese método en los puntos de inyección que resuelvan a él.

Este patrón fue documentado como un patrón creacional por Eric Gamma [Gamma94].

6.2.1.5 Patrón Facade

Este patrón consiste en encapsular uno o varios subsistemas detrás de una fachada, aislando al cliente de esta fachada de dichos subsistemas. Es un patrón sencillo y ampliamente conocido, fue documentado por Eric Gamma [Gamma94] y sus colegas y es especialmente útil para el diseño de aplicaciones en capas.

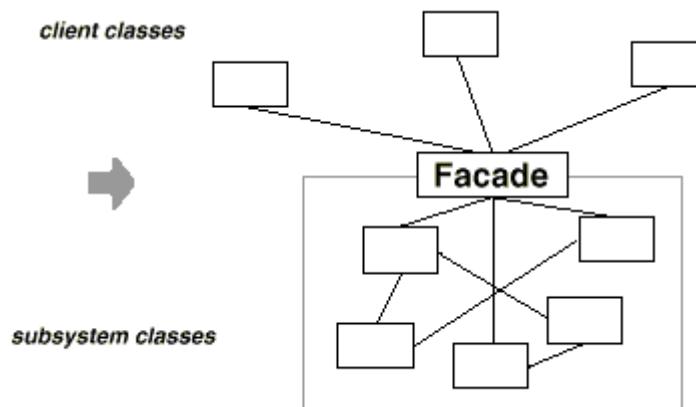


Figura 87: Patrón Facade

6.2.1.6 Patrón Session Facade

Este patrón es una extensión del Patrón Facade llevada al contexto de aplicaciones J2EE /JEE, consiste en definir un bean de sesión como fachada a los subsistemas del nivel de negocio, esto aporta orden a la arquitectura en n niveles y oculta los elementos del nivel de negocio tales como objetos de acceso a datos del nivel superior.

Esta contextualización del patrón Facade fue documentada en el libro “J2EE Core Patterns” de Sun Microsystems. La muestra un esquema de dicho patrón

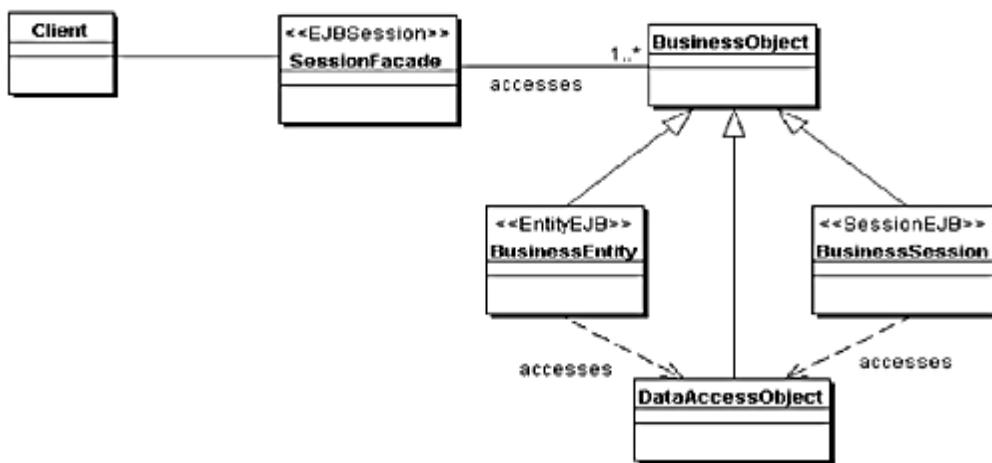


Figura 88: Patrón Session Facade

6.2.1.7 Patrón Adapter

El propósito de este patrón es convertir una interfaz en otra, permitiendo a un cliente que no puede ser modificado, trabajar contra una interfaz para la que no fue diseñado.

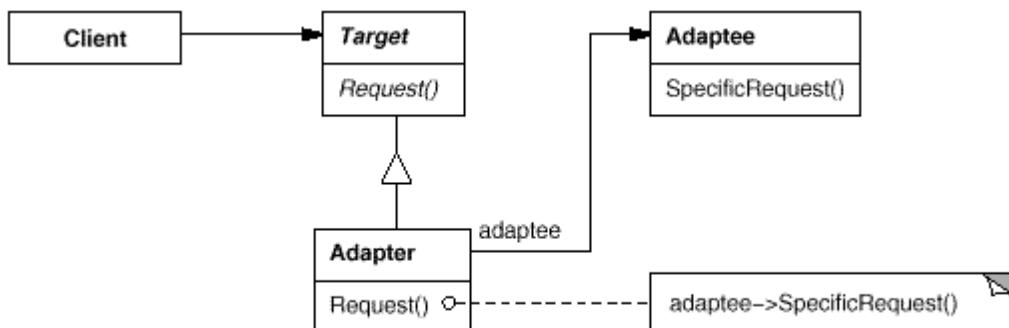


Figura 89: Patrón Adapter

En este sistema se usa para permitir a Spring Security utilizar un usuario de nuestro modelo de negocio. Para ello se construye un adaptador de la clase User a la interface UserDetails esperada por Spring Security. Es un patrón documentado en el libro de patrones de Eric Gamma [Gamma94] y clasificado como un patrón estructural.

6.2.1.8 Patrón Translator

Es un patrón algo menos conocido que el resto, está documentado en el libro "A functional Pattern System for Object-Oriented Design" [Küne98] que propone un sistema de patrones que faciliten realizar diseños con un enfoque más cercano al paradigma funcional usando el paradigma orientado a objetos.

Más concretamente este patrón permite convertir o traducir un objeto a otro, se basa en la idea de que existen dos objetos equivalentes en contextos distintos.

El apartado 13.4.6.3 “Ejemplo de Patrón Translator” muestra el uso hecho de este patrón en el sistema, que no ha sido más que el de definir una jerarquía de traductores que conviertan objetos pesados del modelos de negocio a objetos más ligeros y especializados del modelo de presentación.

6.2.1.9 Patrón Builder

Es un patrón creacional [Gamma94], que pretende separar y ocultar el proceso de construcción de un objeto. En este sistema se ha usado para encapsular procesos de construcción de elementos visuales como menus o rejillas, se ha variado algo la definición del patrón hecha pro Gama para usarlo más como un mecanismo de refactorización que como una técnica de Diseño, un ejemplo de la aplicación del patrón puede verse en el apartado 13.4.6.4 “Ejemplo de Patrón Builder”.

6.2.2 Inyección de Dependencias

Aunque es considerada por algunos un patrón de diseño, por otros entre los que me incluyo la inyección de dependencias es considerada más como un estilo de programación que un patrón de diseño. Este estilo de programación deja la instanciación de las clases en manos de un contenedor, contenedor que juega el rol de gran factoría que además de crear las instancias se las inyecta a otros objetos según lo hayamos definido. En este sistema se aplica la especificación CDI para realizar un diseño basado en inyección de dependencias, esta tecnología permite definir puntos de inyección mediante anotaciones, estos puntos son referencias que el contenedor asignará a instancias que él ha creado.

Dado que el sistema presenta una arquitectura en capas lo común será ver las agregaciones de unas capas a otras, definidas mediante atributos anotados para que el contenedor los inyecte.

En casi todos los ejemplos que se muestran en el apartado 13.4 “Código Fuente” puede verse el uso de la inyección de dependencias.

6.2.3 Programación Orientada a Aspectos

Otra técnica más que destacable que se ha usado para el diseño de algunas funcionalidades es la Programación Orientada a Aspectos. La Programación Orientada a Aspectos en adelante AOP, en lugar de descomponer la aplicación en jerarquías de objetos, la descompone en problemas a resolver o aspectos, de esta forma se pueden modularizar aspectos que afectan a grandes partes de la aplicación o módulo, también llamados aspectos de corte cruzado o aspectos transversales. En este sistema se ha utilizado AOP para modularizar la gestión de las excepciones.

JEE ofrece un mecanismo que permite aplicar un enfoque orientado a aspectos, son los *Interceptores*, estos nos permitirán encapsular aspectos de nuestro sistema en uno o más interceptores que se aplicarán a diversos puntos del código, que podremos definir mediante anotaciones.

A continuación se exponen los conceptos más básicos de orientación a aspectos y se relaciona con la aplicación de los interceptores a este fin.

- **Aspecto:** Es un problema que puede ser modularizado para que este módulo sea aplicado a varios objetos o piezas de código. El ejemplo más clásico es la gestión de las transacciones, en nuestro sistema hemos definido aspectos para el manejo de las excepciones.
- **Advice:** Es una acción que define el aspecto para ser ejecutada sobre diversos puntos de la aplicación. En nuestra aplicación se ha creado un interceptor que captura excepciones en el nivel de negocio, deja el log adecuado y trata la excepción. El apartado 13.4.7.1 Fichero “BusinessExceptionHandler.java” lo ilustra con exactitud.
- **Joinpoint:** Punto de ejecución de un programa sobre el que queremos aplicar determinadas acciones contempladas en el aspecto. En nuestra aplicación se aplica el interceptor “BusinessExceptionHandler” a todas las clases de la capa de negocio.

6.2.4 Diagramas de Clases

Debido al elevado número de clases que se han desarrollado, aprovecharemos la estructura de paquetes con el fin de facilitar la comprensión de los diagramas.

6.2.4.1 Diagramas de Clases de cnpd-business

6.2.4.1.1 Paquete model

El paquete model contiene las clases que **modelan el dominio del sistema**, además de modelar el sistema, estas clases tienen una importancia vital, ya que mediante la Java Persistence API definen las entidades que JPA usará para generar y tratar la base de datos.

Las clases principales del paquete, como *User* o *Project* son entidades, esto significa que el proveedor de persistencia generará una tabla para cada una de ellas.

Con el fin de simplificar las entidades se define una relación de composición entre algunas de ellas y otro tipo de clase que no son definidas como entidad, si no que se configuran de tal forma que todos sus atributos sean añadidos a la tabla de la entidad.

Además de las entidades, el motor de mapeo objeto-relacional generará tablas para representar las relaciones que se definen entre las entidades.

Para evitar el acceso no deseado a las colecciones, se han blindado los métodos accesorios de estas, de forma que devuelvan colecciones no modificables. Con el fin de modificar las colecciones se han añadido a las clases del modelo métodos para añadir y eliminar elementos de las colecciones.

En el diagrama de la Figura 90 no se han añadido los métodos accesorios, para facilitar su lectura.

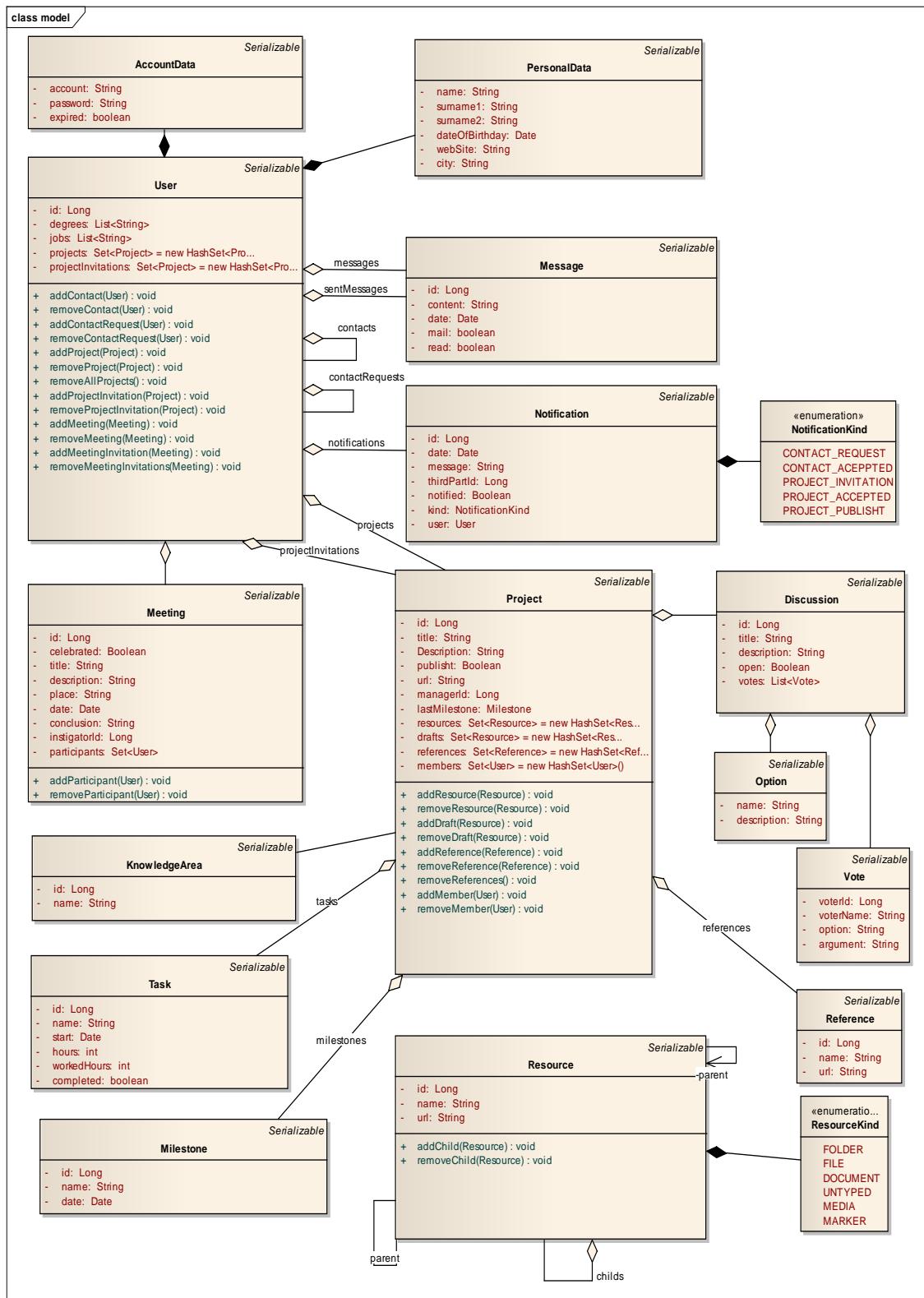


Figura 90: Diagrama de clases- Paquete model

6.2.4.1.2 Paquete persistence

El diagrama de la Figura 91 muestra la jerarquía de interfaces definidas dentro del paquete persistence con el fin de encapsular el acceso a los datos del sistema.

Se ha definido una interfaz para cada entidad según el patrón DAO explicado en el apartado 6.2.1 “Patrones de Diseño”.

Todas las interfaces heredan las operaciones básicas de un DAO genérico (*IDao*), definiendo el tipo de entidad para la que la interfaz concreta encapsulará el acceso, así como algunas operaciones adicionales.

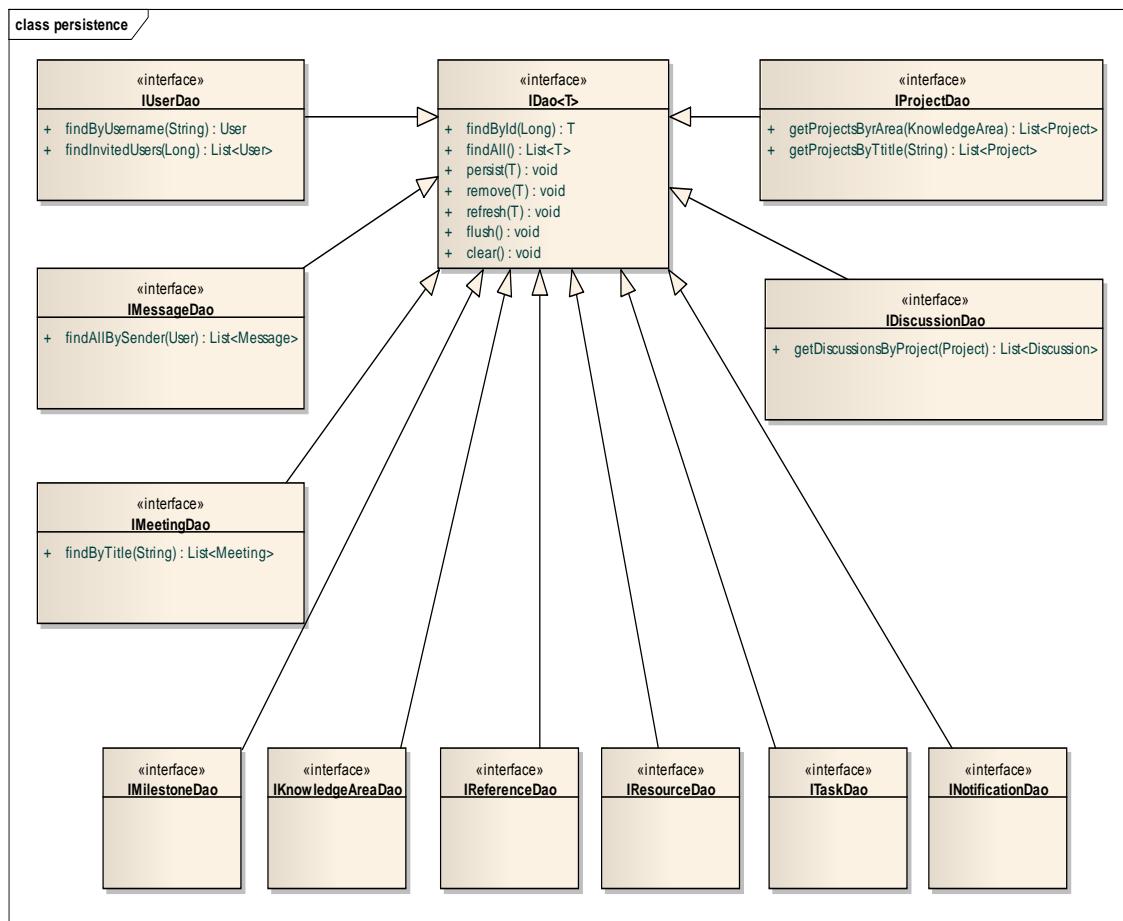


Figura 91: Diagrama de clases- Paquete persistence

La Figura 92 muestra la relación de las interfaces DAO con sus implementaciones, ubicadas en el paquete *persistence.impl*, existe una implementación para cada interfaz DAO, que obviamente implementa todas las operaciones definidas en el DAO genérico y en la interfaz propia.

Al tiempo que implementa una interfaz concreta, cada DAO extiende una clase abstracta que sirve de soporte para todos los DAOs, *JpaDaoSupport*.

JpaDaoSupport tiene una instancia de la interfaz javax.persistence.EntityManager que le ha sido inyectado por el contenedor, además define operaciones que no dependen del tipo de DAO y pueden ser reutilizadas por todos los DAOs.

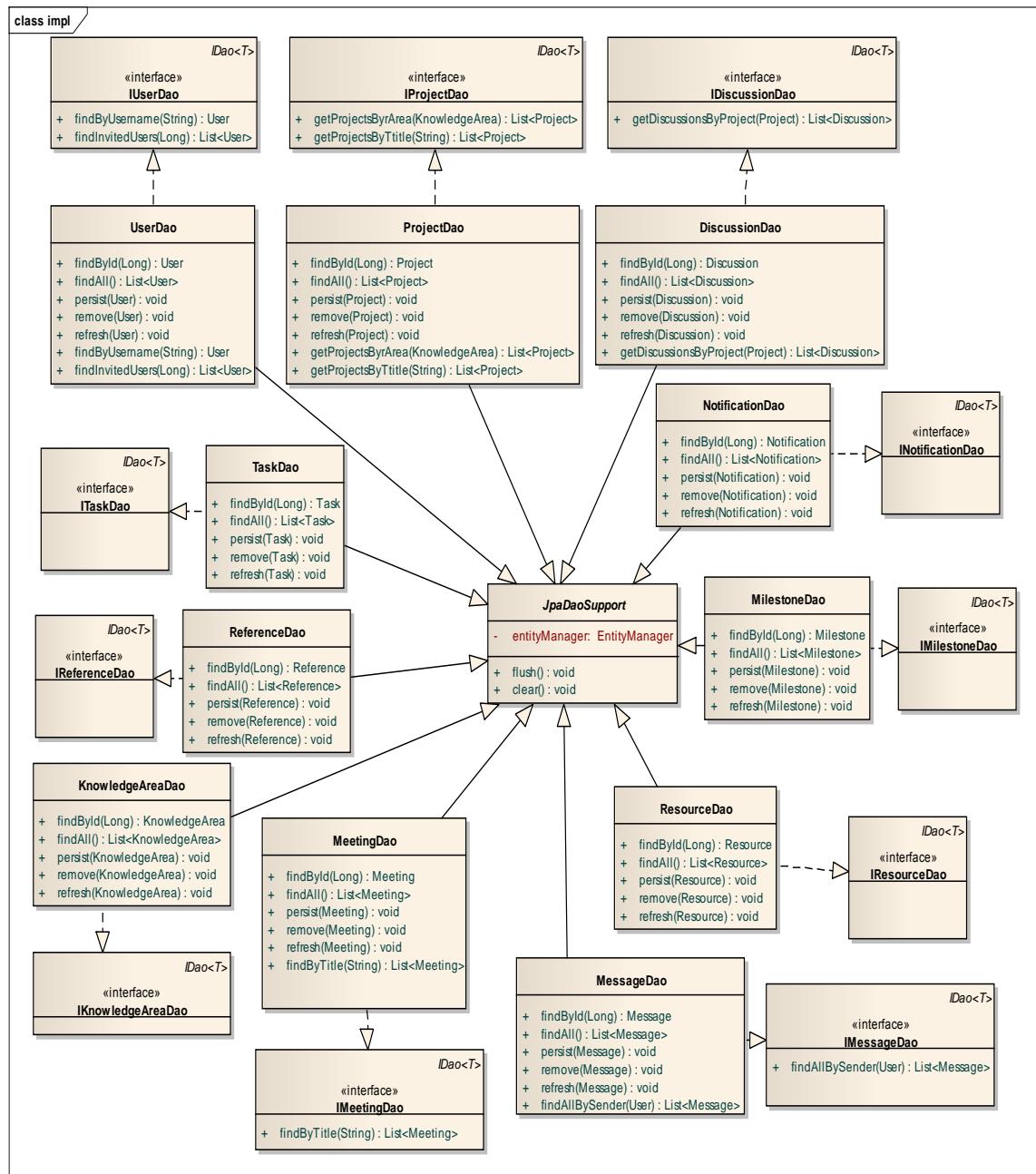


Figura 92: Diagrama de clases – Paquete persistence.impl

Además de los DAOs, se ha definido una jerarquía de excepciones propia de la capa de persistencia para encapsular los errores producidos en esta capa, siempre que no sean problemas de infraestructura, que ya están cubiertos por otras jerarquías de excepciones.

La Figura 93 muestra las excepciones definidas en esta capa, en este caso solo una para tratar el caso en el que pudiese haber nombres de usuario duplicado.

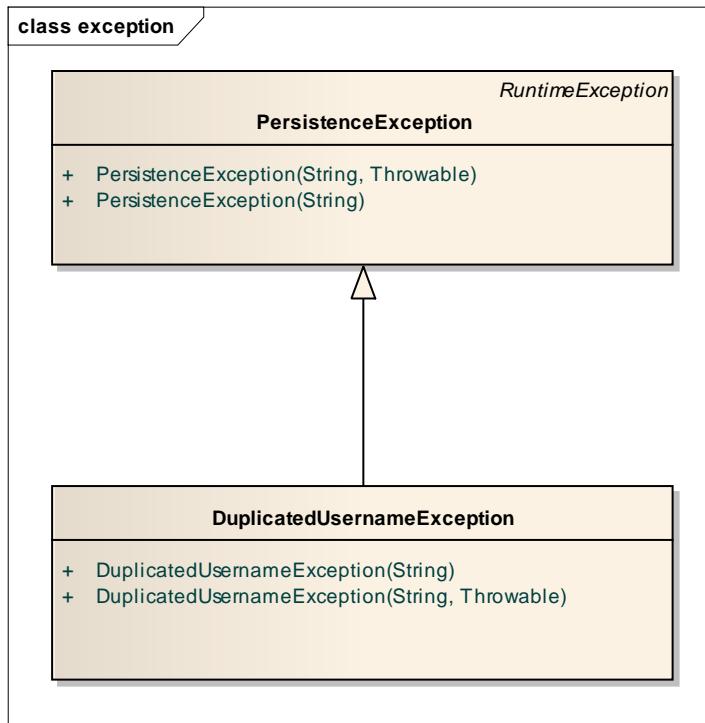


Figura 93: Diagrama de clases - persistence.exception

6.2.4.1.3 Paquete business

Este paquete contiene las clases de la capa de negocio, clases que se encargan de resolver todas las reglas de negocio del sistema.

Esta capa define cuatro fachadas para el resto del sistema. Estas fachadas utilizan la tecnología EJB para garantizar la transaccionalidad y el acceso remoto. Con el fin de ser desplegados local y remotamente, se definen dos interfaces una local y otra remota para cada fachada, estas no añaden ninguna operación, sino que permiten al contendor desplegar los beans con dos caras.

La Figura 94 muestra las fachadas y sus implementaciones, así como las interfaces remotas y locales.

- IUsersManager: Define todas las operaciones relativas a la gestión de los usuarios.
- IContactsManager: Define todas las operaciones relativas a las relaciones entre usuarios.
- IProjectsManager: Define todas las operaciones relativas a los proyectos.
- IResourcesManager: Define todas las operaciones relativas a los recursos del sistema.

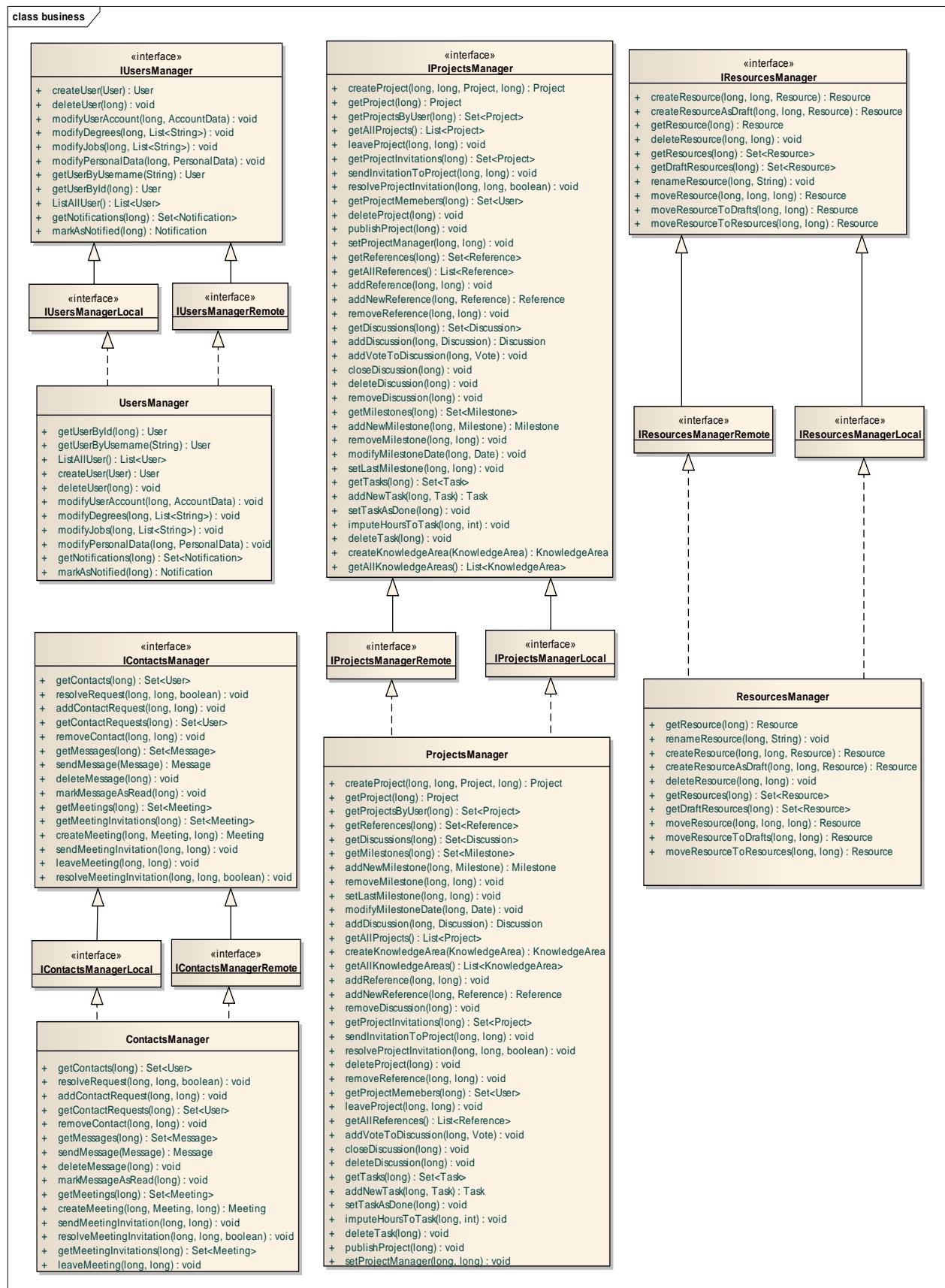


Figura 94: Diagrama de clases - Paquete business

La Figura 95 muestra la jerarquía de excepciones creada para tratar los errores producidos en la capa de negocio.

Todas las excepciones están definidas como hijas de la clase *BusinessException*, que además está anotada como *ApplicationException* para ser soportada por la tecnología EJB, y viajar a través de distintas maquinas, esta anotación es heredada por todas las clases de la jerarquía. A continuación se detallan las excepciones definidas para los distintos casos de error.

- La excepción *InvalidMemberException* es lanzada cuando se trata a un usuario como miembro de un proyecto, sin serlo, por ejemplo si se intenta establecer como gestor de un proyecto a un usuario que no es miembro de un proyecto.
- La excepción *InvalidInvitationException* es lanzada cuando se detecta una invitación inválida, por ejemplo si se intenta invitar a un usuario que ya es miembro de un proyecto, o que ya está invitado.
- La excepción *InfraestructureException* es lanzada cuando se detecta un error en la infraestructura de ficheros del sistema.
- La excepción *IllegalResourceMovementException* es lanzada cuando un recurso es movido de forma inválida, por ejemplo si se detectan inconsistencias con el padre antiguo y el nuevo parente.
- La excepción *InvalidMessageException* es lanzada cuando se detecta un mensaje inválido, por ejemplo si se envía un mensaje con igual emisor que destino.
- La excepción *InvalidReferenceException* es lanzada cuando se detecta un error en el manejo de las referencias, por ejemplo si se intenta añadir una referencia a un proyecto que ya la tiene.
- La excepción *InvalidRequestException* es lanzada cuando se detecta una petición inválida, por ejemplo si se intenta hacer una petición de un usuario a otro que ya tiene en su lista de contactos.
- La excepción *NonExistentalException* es lanzada cuando se aporta un identificador incorrecto para una operación.
- La excepción *IllegalResolutionException* es lanzada cuando se detecta una resolución inválida, por ejemplo si se intenta añadir un usuario como contacto sin que exista una invitación, será tratado como un error de este tipo.
- La excepción *ContactNotFoundException* es lanzada cuando un usuario que debía estar en la lista de contactos de otro y no lo está.

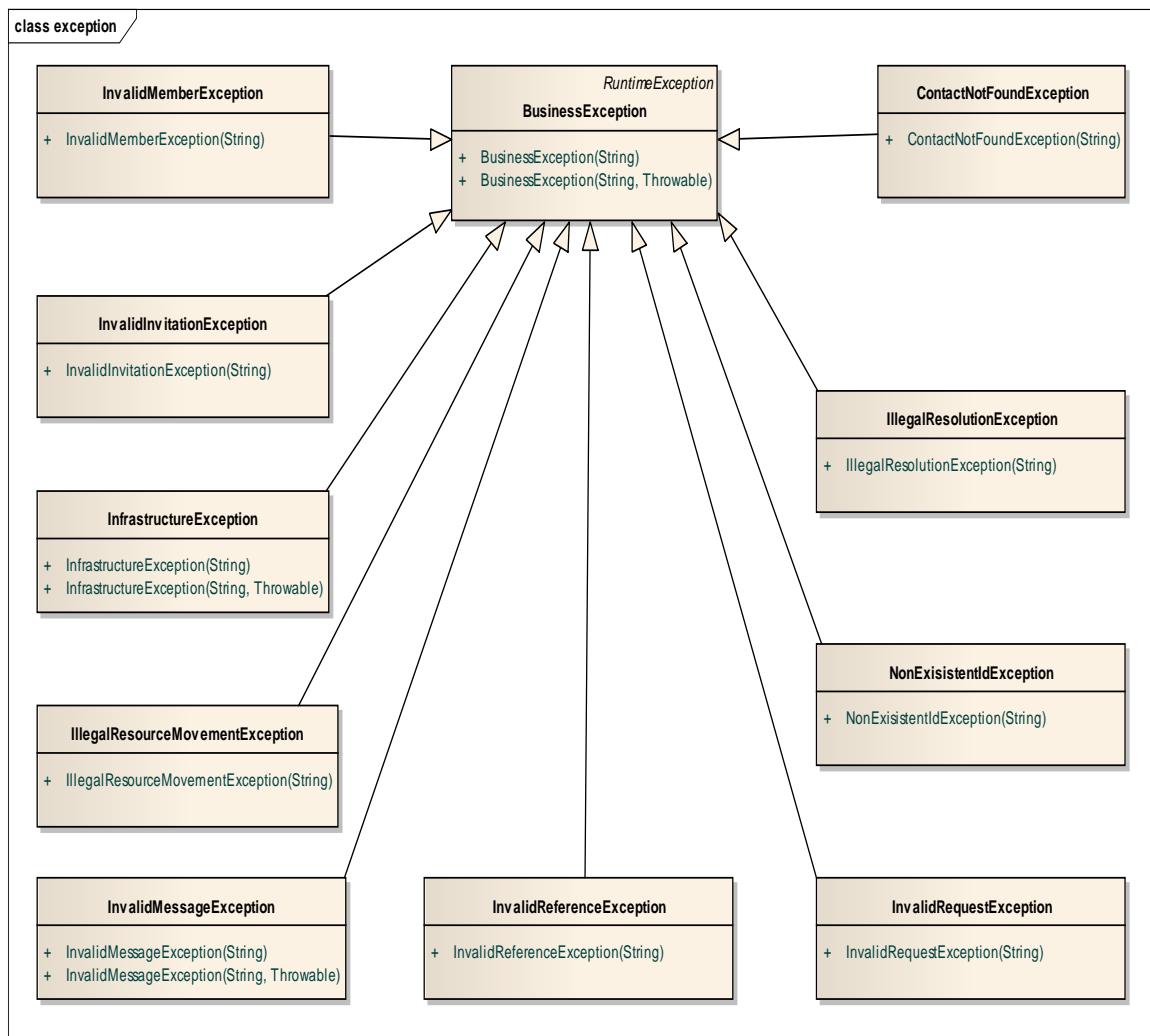


Figura 95: Diagrama de clases – Paquete business.exception

En los siguientes diagramas, se exponen divididas por funcionalidades, las distintas clases que implementan las reglas de negocio. Para hacer esta división se muestra un diagrama por cada fachada (Manager), que muestra las clases en que delega esta fachada para resolver las reglas de negocio así como sus relaciones.

La Figura 96 muestra las clases con que interacciona la clase *UsersManager* para resolver las reglas de negocio que esta clase resuelve.

- La clase *UsersSpecialist* se encarga de tratar con todo lo relativo a los usuarios, despreocupándose del resto de tareas como el envío o la recuperación de notificaciones.
- La clase *NotificationsSpecialist* se ocupa de todo lo relativo a las notificaciones.
- Finalmente la clase *MailHelper* permite enviar correos electrónicos.

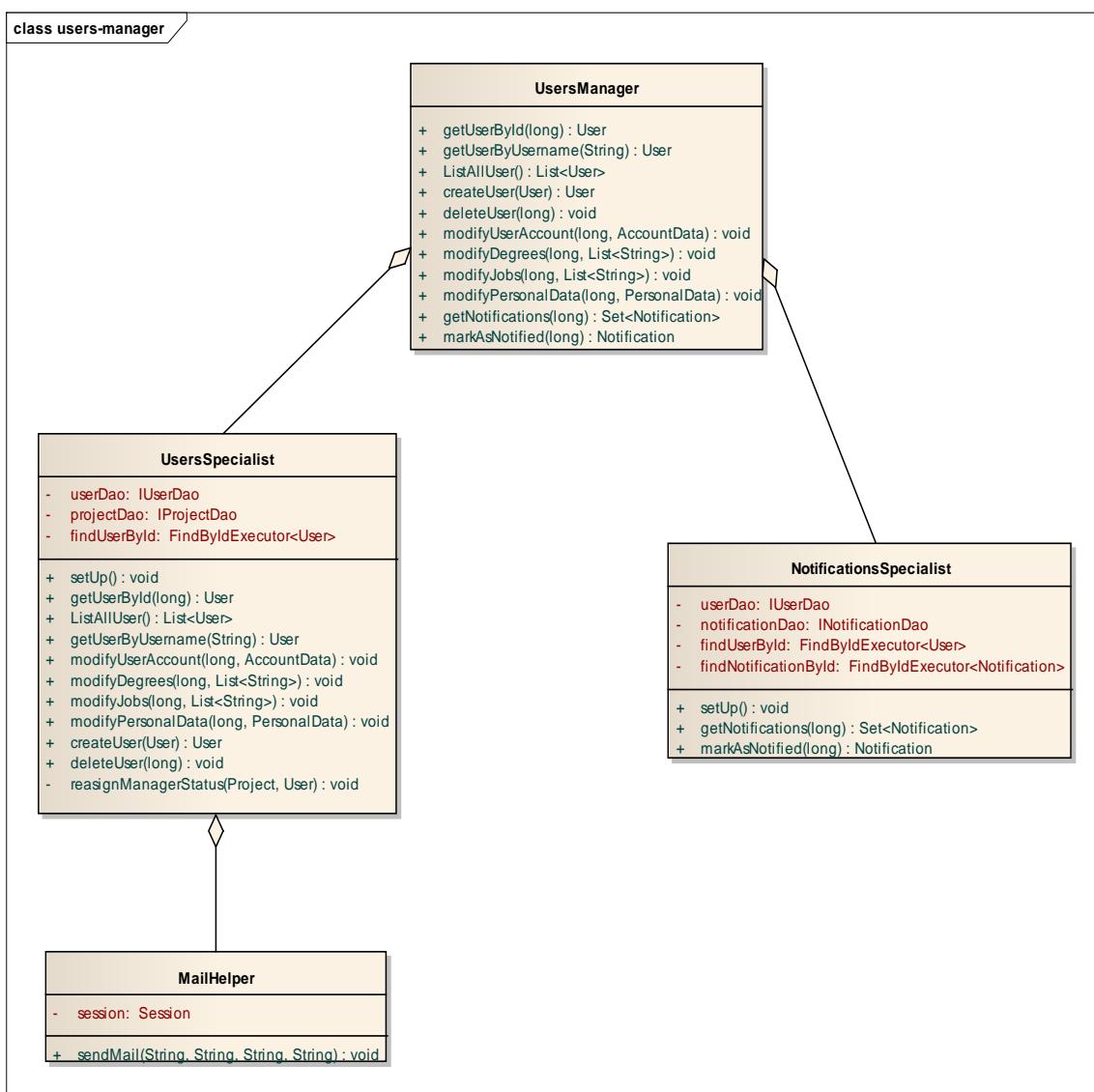


Figura 96: Diagrama de clases – `UsersManager`

La Figura 97 muestra la clase `ContactsManager` y las clases con que este interactúa.

- La clase `ContactsSpecialist` está especializada en operaciones básicas para el mantenimiento de la red de contactos.
- La clase `RequestsSpecialist` está especializada en el tratamiento de las peticiones.
- La clase `MeetingsSpecialist` está especializada en operaciones para la gestión de las reuniones.
- La clase `MeetingsSpecialist` está especializada en operaciones para el envío de mensajes.

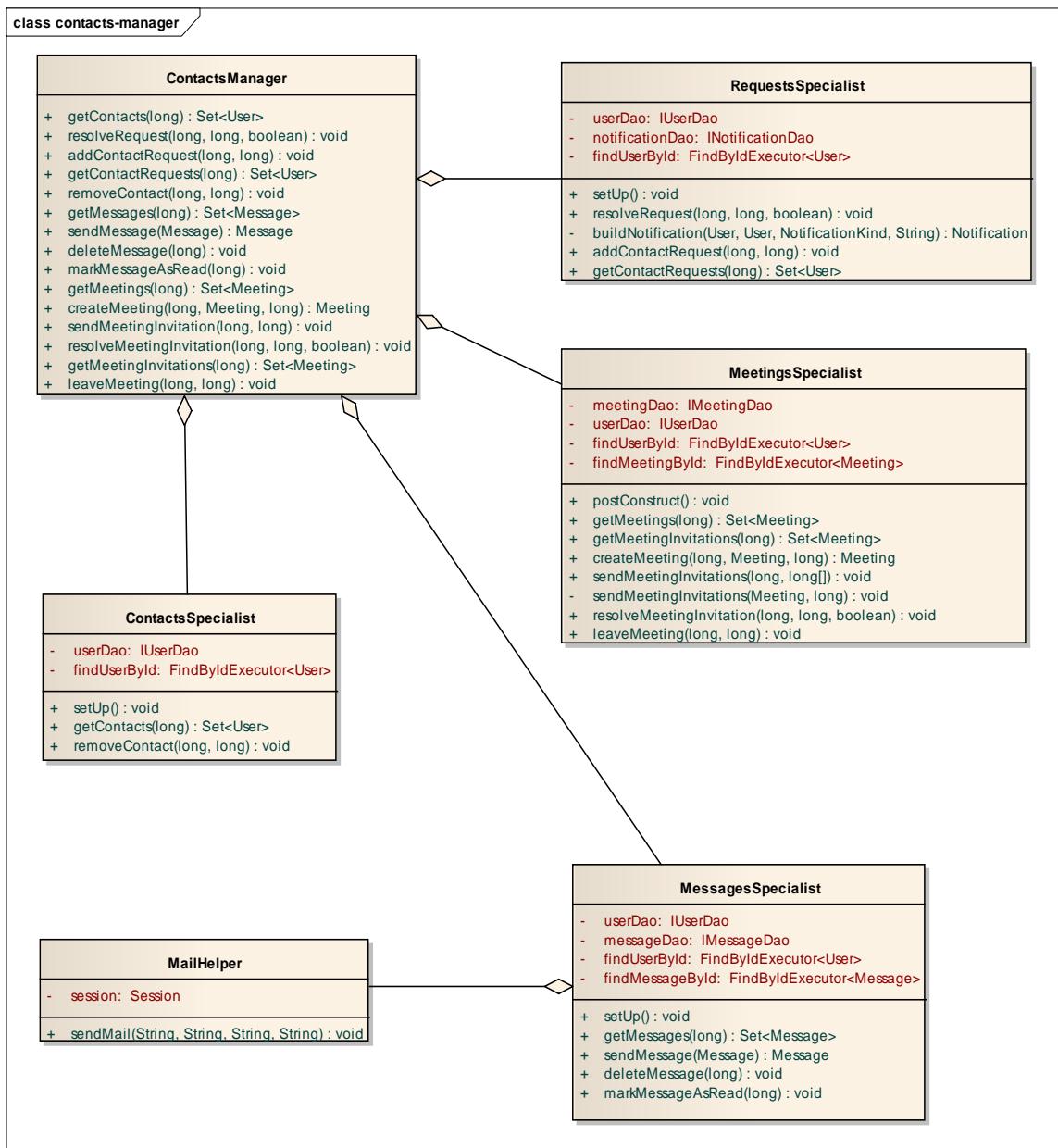


Figura 97: Diagrama de clases – `ContactsManager`

El siguiente diagrama (Figura 98) muestra las clases con que interacciona la clase `IProjectsManager`.

- La clase `ProjectsSpecialist` se encarga de las operaciones que tratan exclusivamente con los proyectos.
- La clase `ProjectInvitationsSpecialist` se encarga de tratar con las invitaciones a proyectos.
- La clase `DiscussionSpecialist` se encarga de tratar exclusivamente con discusiones.
- La clase `MilestonesSpecialist` contiene todas las operaciones para manipular hitos.

- La clase KnowledgeAreasSpecialist se encarga de todo lo relativo a las áreas de conocimiento.
- La clase TasksSpecialist se ocupa de todo lo relativo a las tareas.
- La clase SvnHelper encapsula operaciones para manipular los ficheros de un proyecto sobre Apache Subversion.

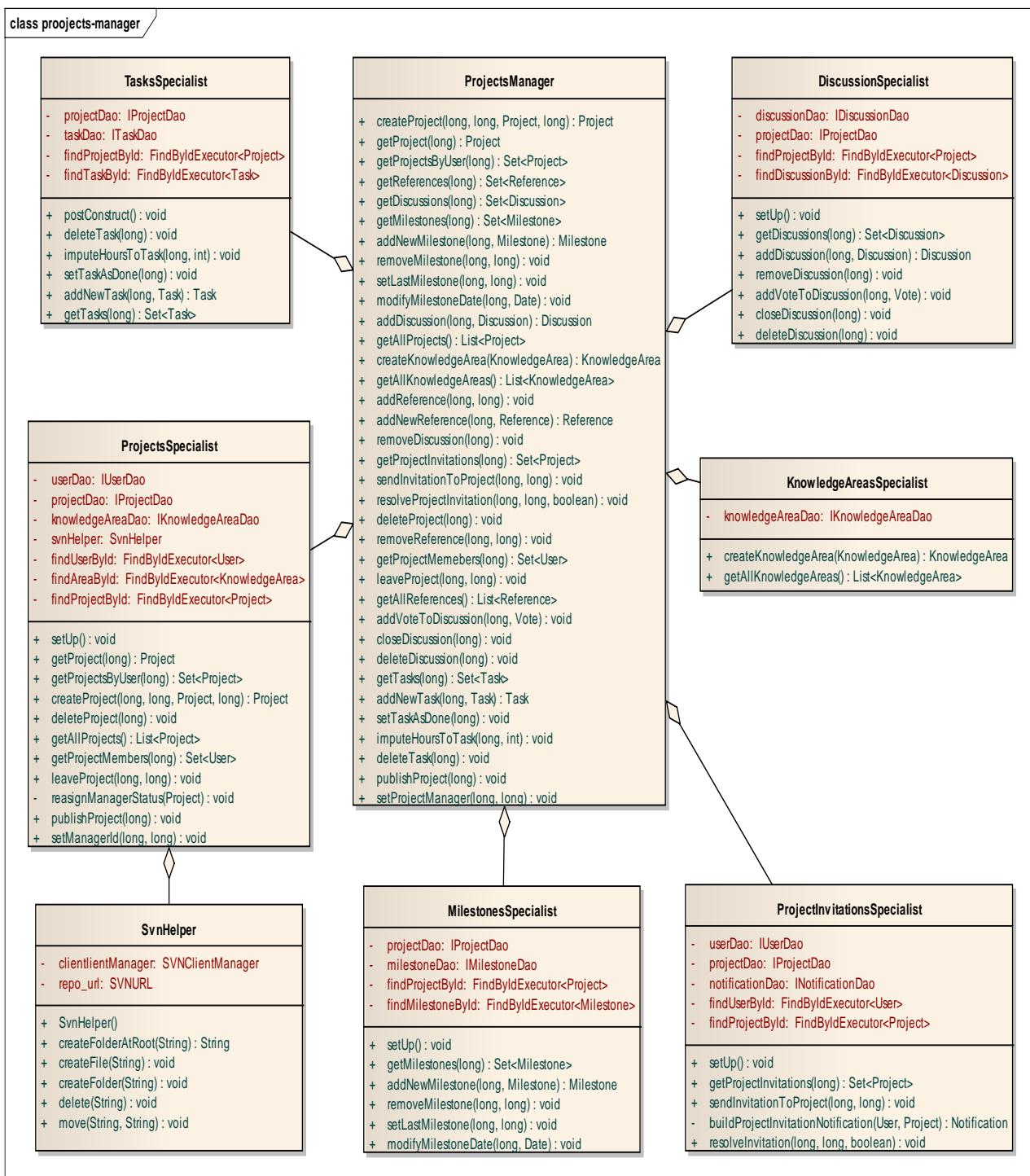


Figura 98: Diagrama de clases – ProjectsManager

Finalmente se muestran (Figura 99) las relaciones de la clase *ResourcesManager*.

- La clase *ResourcesSpecialist* se encarga de lo relativo a los recursos y colabora con la clase *SvnHelper*.

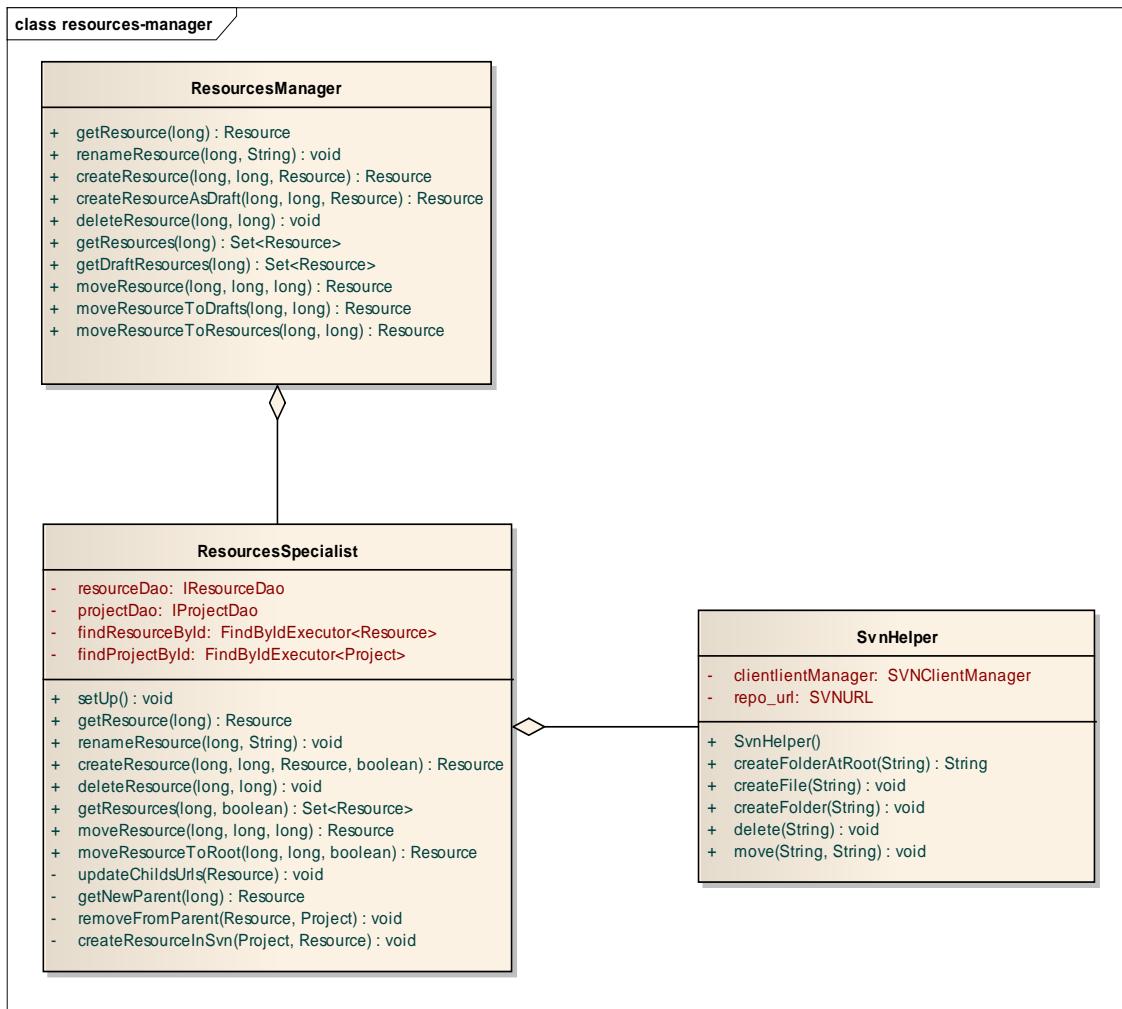


Figura 99: Diagrama de clases - *ResourcesManager*

6.2.4.2 Diagramas de Clases de *cnpd-web*

Como se detalló en el apartado 6.1.4 “Diagramas de Paquetes”, el paquete web, contiene la aplicación web del sistema, dividida en tres sub paquetes. Como ya se ha explicado, esta aplicación cuenta con clases ejecutadas en el servidor, clases ejecutadas en el cliente y clases comunes a cliente y servidor. Se detallan a continuación, agrupadas por paquetes, las clases de este módulo.

6.2.4.2.1 Paquete shared

Estas clases son utilizadas tanto desde el servidor como desde el cliente, incluyen las interfaces de comunicación entre cliente y servidor, excepciones, anotaciones y un modelo propio de la capa de presentación.

El diagrama de la Figura 100 muestra las clases del modelo de presentación, estas clases además de especializar el modelo de negocio para su uso en presentación, están anotadas para poder ser compartidas entre cliente y servidor.

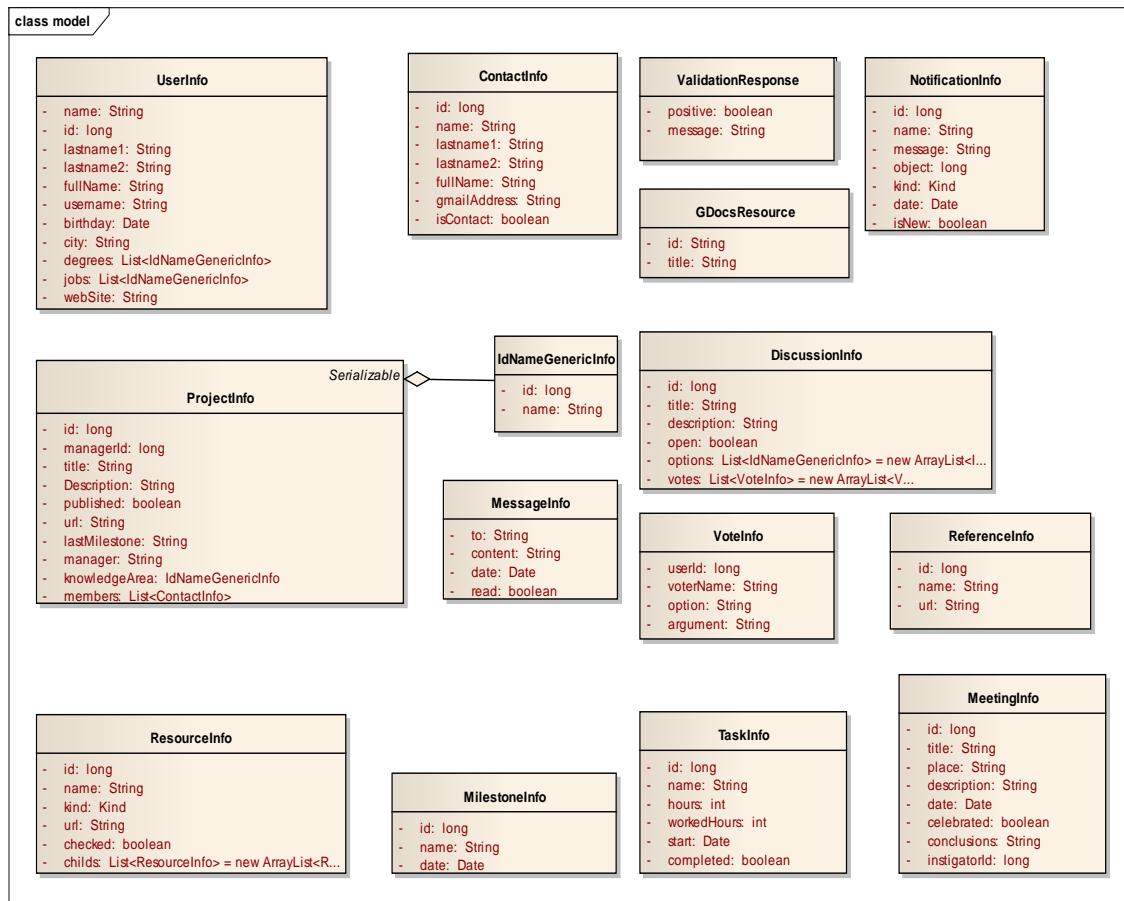


Figura 100: Diagrama de clases – Paquete shared.model

Además del modelo, también otro elemento en común a cliente y servidor, son las excepciones de la aplicación. El diagrama de la Figura 101 muestra las excepciones definidas para tratar los errores de la aplicación, estas excepciones también han sido anotadas de forma que puedan ser procesadas por el mecanismo de comunicación cliente servidor.

Se ha definido una excepción base para toda la capa de presentación, y dos jerarquías paralelas, una para excepciones lanzadas desde el lado del cliente y otra para las lanzadas el lado del servidor.

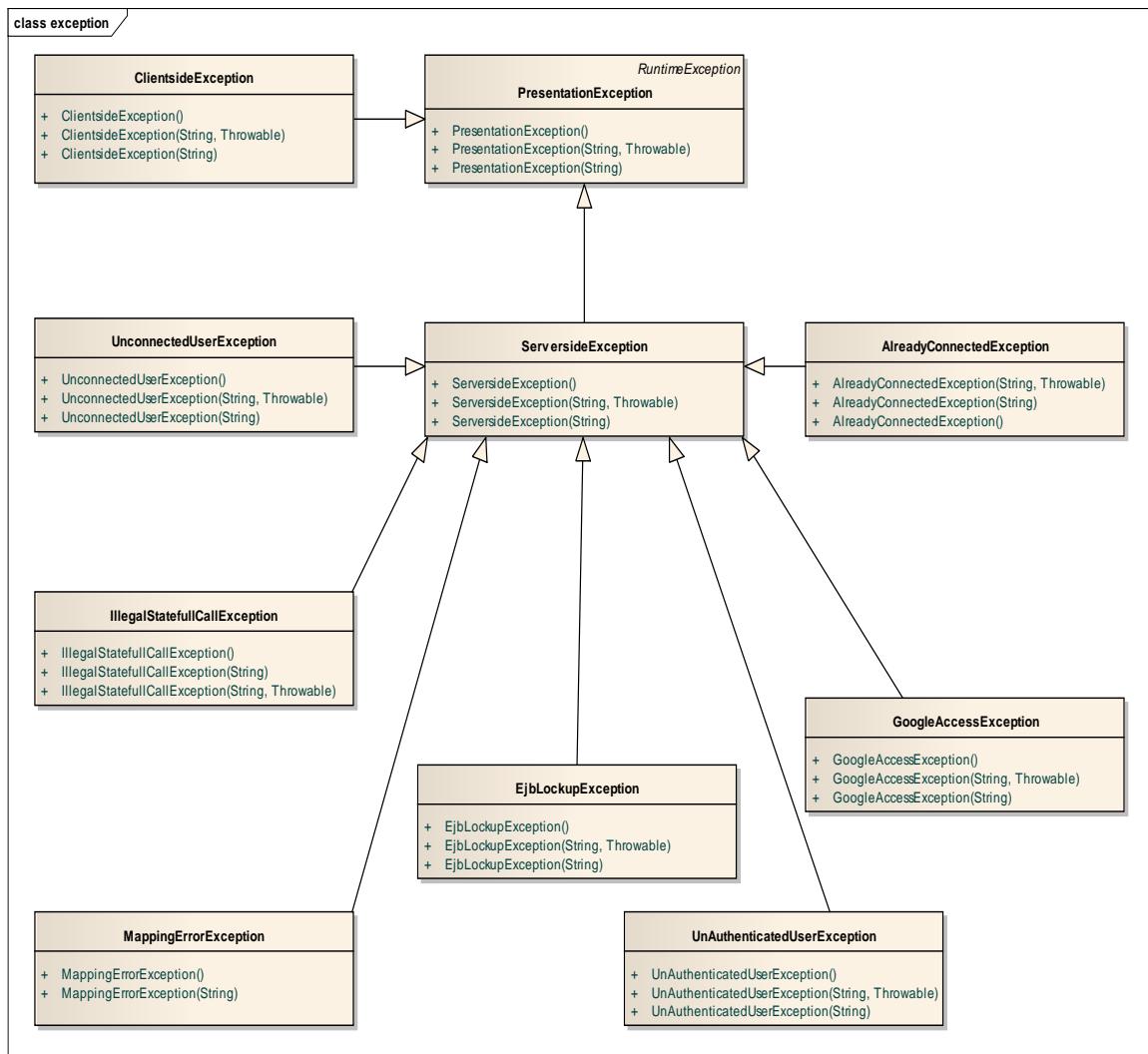


Figura 101: Diagrama de clases – Paquete `shared.exception`

Otro elemento en común, son los cualificadores, esto son anotaciones definidas para resolver los puntos de inyección a implementaciones concretas.

El diagrama de la Figura 102 muestra los cualificadores definidos para la aplicación, estas anotaciones son añadidas a las clases y a los puntos de inyección, y junto con el tipo de bean resolver la instancia contextual que será inyectada.

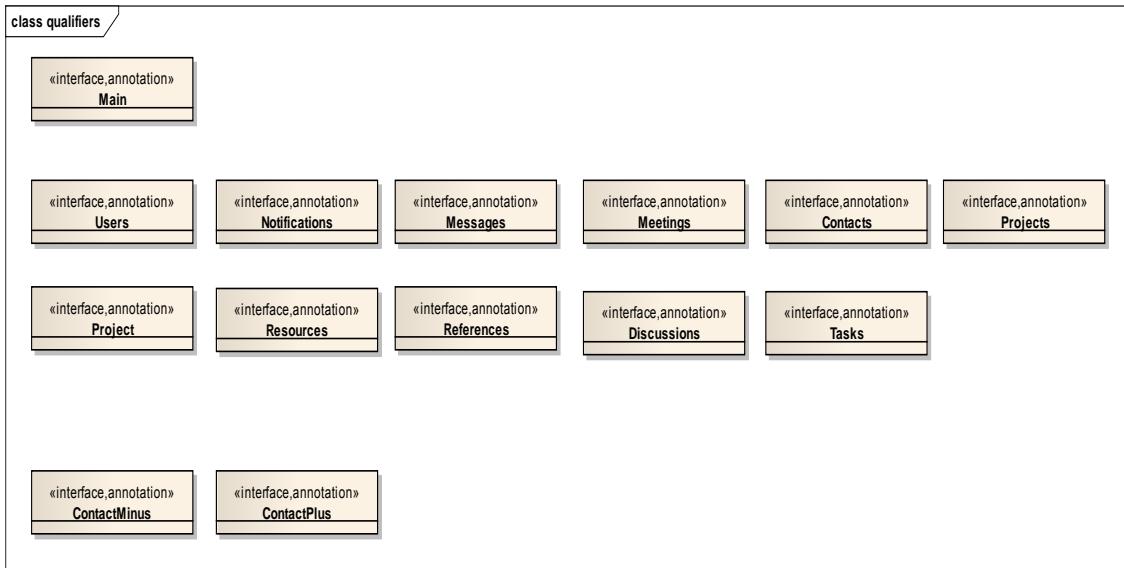


Figura 102: Diagrama de clases – Paquete shared.qualifiers

Finalmente este paquete define las interfaces que usarán las clases del lado del cliente para comunicarse con las clases del servidor.

Todas las interfaces definidas pueden verse en la Figura 103, las implementaciones de estas interfaces residen en el servidor por lo que se encuentran en el paquete server.

Tanto las interfaces como las implementaciones llevan anotaciones que permiten que sean procesadas por el mecanismo de comunicación ofrecido por el framework Errai.

EL fin del diagrama de la Figura 103 es mostrar todas las interfaces definidas en lado del servidor y sus operaciones, estas interfaces están más especializadas que las fachadas de negocio ya que están enmarcadas en la aplicación web, mientras que las fachadas de negocio están expuestas a todo el sistema.

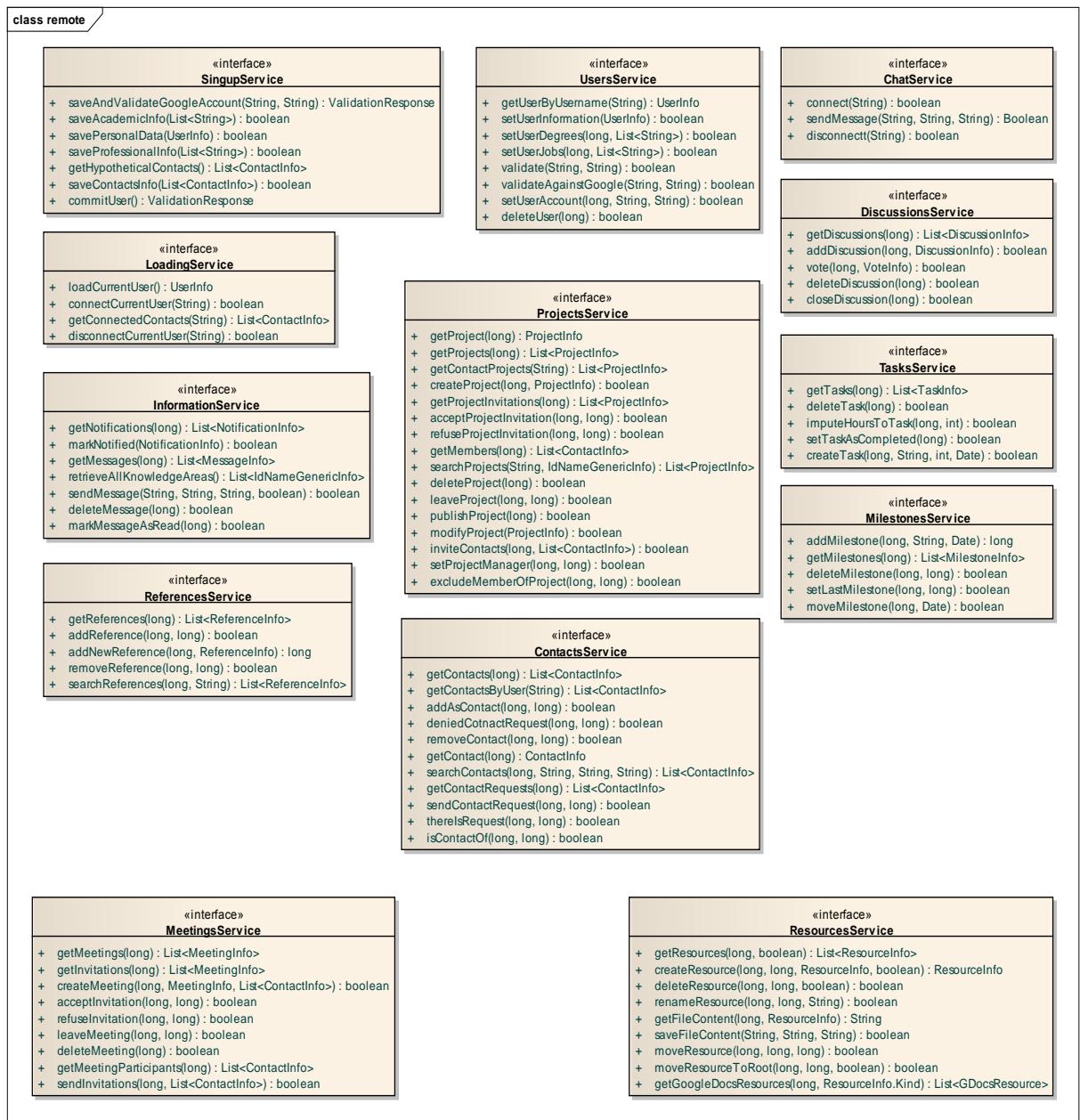


Figura 103: Diagrama de clases- Paquete shared.remote

6.2.4.2.2 Paquete server

Este paquete contiene todas las clases de la aplicación web que se ejecutarán en el lado del servidor.

El diagrama de la Figura 104 muestra la implementación de los servicios remotos usados por la las clases cliente, por cada interfaz definida existe una clase que la implementa.

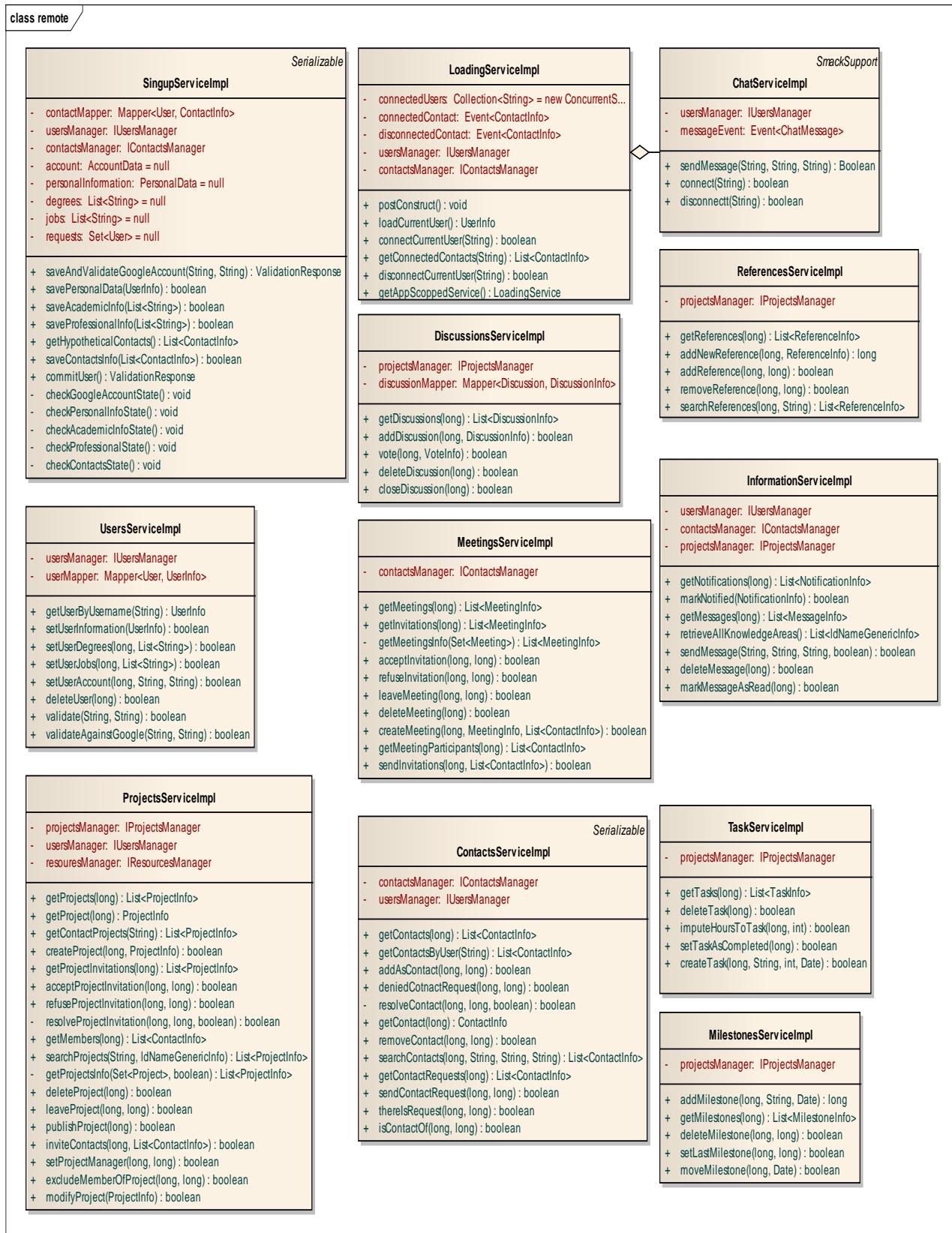


Figura 104: Diagrama de clases – Paquete server.remote

Estas clases se comunican con las fachadas de negocio, y usan una jerarquía de mapeadores para convertir las entidades del modelo de negocios, en objetos del modelo de presentación.

La Figura 105 muestra esta jerarquía, que cuelga de una interfaz genérica, dejando la definición del tipo de los objetos a mapear, a las implementaciones.

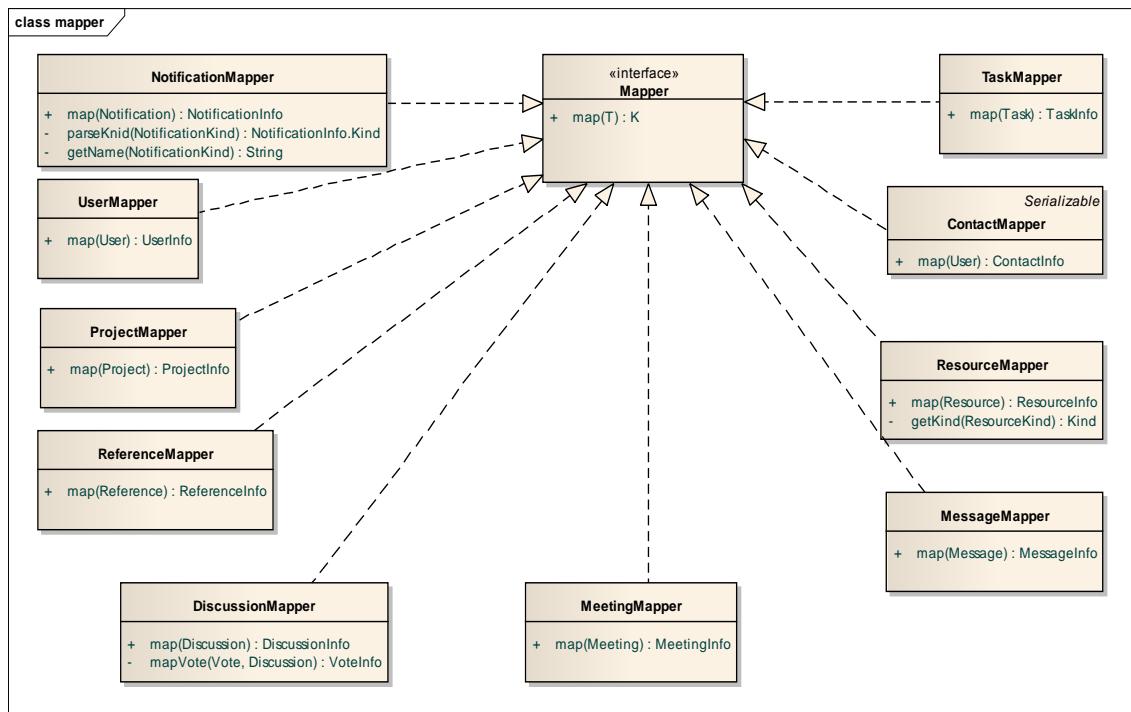


Figura 105: Diagrama de clases server.mapper

Otro sub paquete de clases del lado del servidor (Figura 106) define las clases relacionadas con el API Smack, estas clases se encargan de interactuar con el API para proporcionar la funcionalidad del Chat de la aplicación.

- La clase *ChatListener* contiene el código que el API ejecuta cuando se crea un *Chat*, por el usuario o por otro usuario, para ello implementa un interfaz definido por el API.
- La clase *IncomingMessageListener* contiene código que al API ejecuta cuando un usuario manda un mensaje o alguien se lo manda a él. Para ello implementa una interfaz definida por el API.
- La clase *SmackChanel* encapsula lo necesario para gestionar las comunicaciones de un usuario, cuando un usuario se conecta al chat se crea un canal que contiene una conexión, y un escuchador de mensajes.

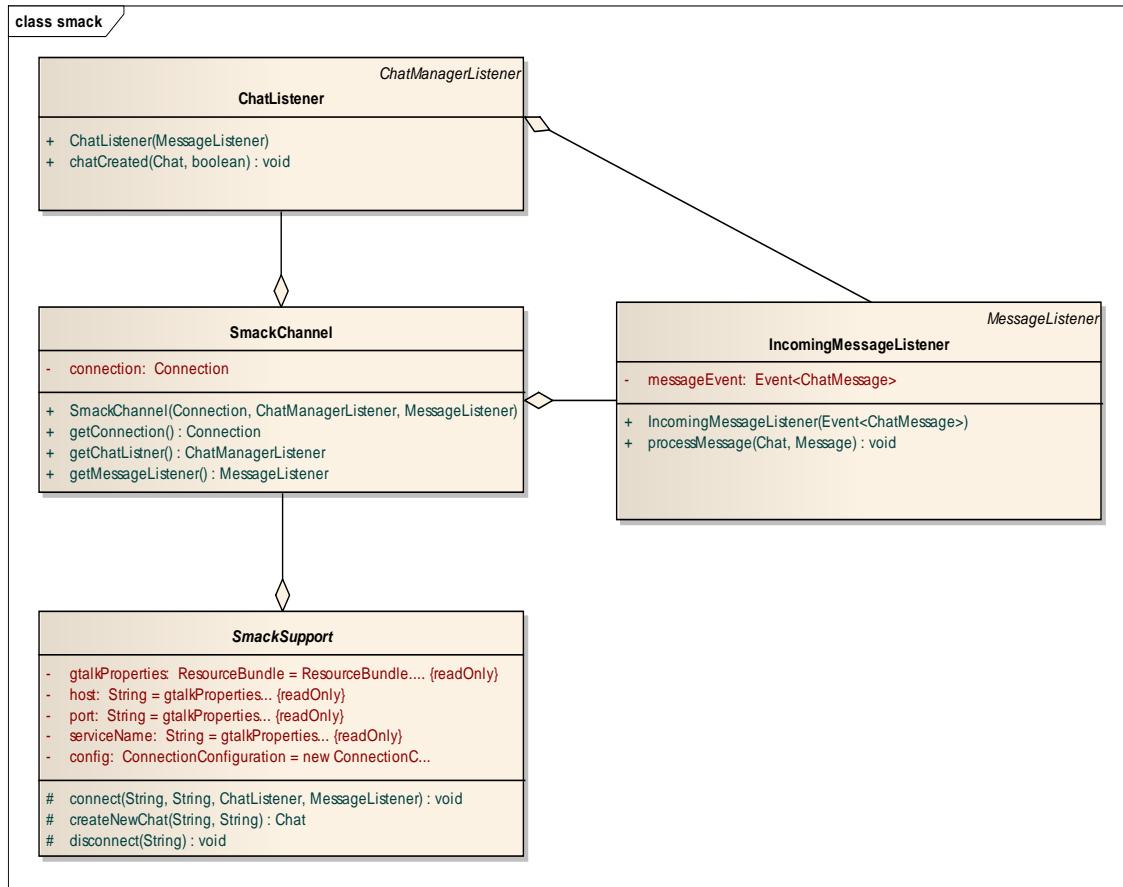


Figura 106: Diagrama de clases server.smack

La clases mostradas en la Figura 107, relacionadas con la seguridad de la aplicación, también radican en el servidor, aunque no están relacionadas directamente con la funcionalidad de la aplicación, estas clases se integran con el framework Spring Security para permitir la autenticación contra el nivel de negocio. Este proceso se detalla en el apartado 6.3.5 “Autenticación”.

- La clase `EjbUsersDetailsService` implementa la interfaz `UsersDetailsService` de Spring Security, esta implementación delega la carga del principal en la capa de negocio del sistema.
- La clase `UserDetailsAdaptor` adapta un objeto usuario a la interfaz `UserDetails` de Spring Security.

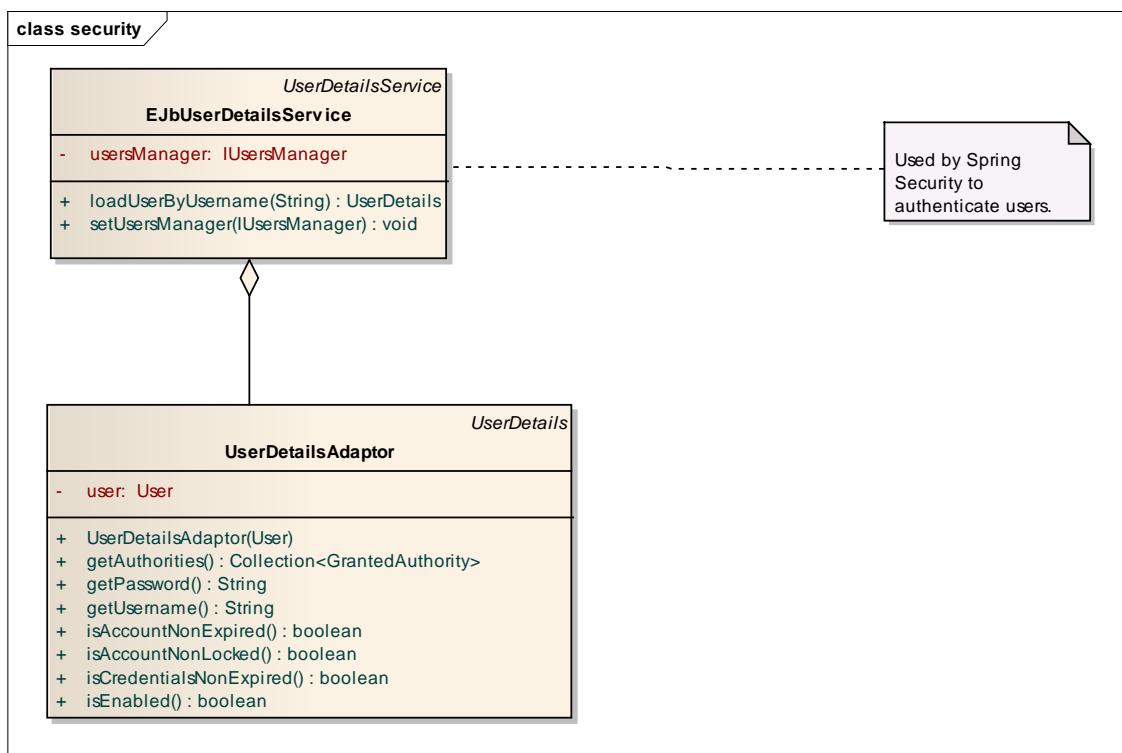


Figura 107: Diagramas de clases – Paquete server.security

Las clases del sub paquete útil implementan interfaces de la especificación Servlet para llevar a cabo tareas que no pueden ejecutarse de otra forma.

- La clase *SessionListener* se define para desconectar a los usuarios cuando la sesión es destruida.
- La clase *FileDownloadServlet* permite descargar ficheros por HTTP, haciendo una petición HTTP a este Servlet con el id del recurso como parámetro se descarga el recurso, comprimiéndolo en caso de que fuese necesario.
- La clase *FileUploadServlet* permite realizar lo contrario, sube un recurso, también mediante una petición HTTP con el parámetro del recurso que se desea modificar.

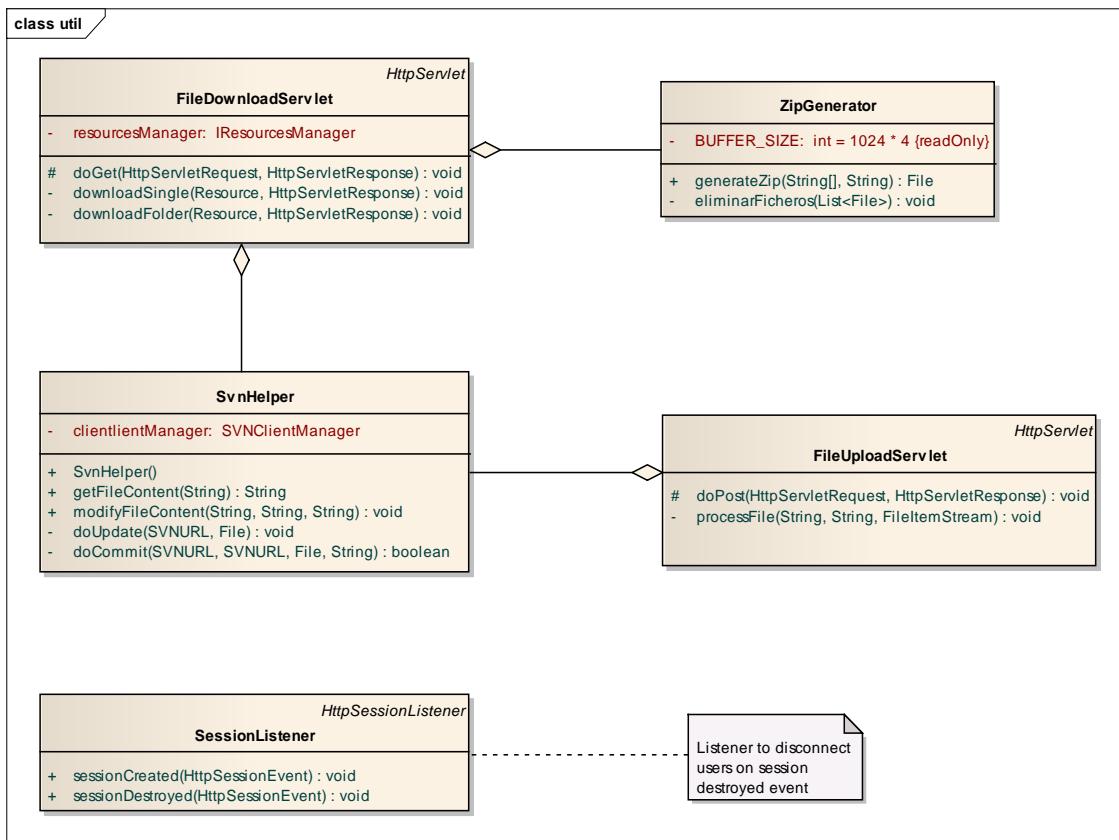


Figura 108: Diagrama de clases – Paquete server.util

6.2.4.2.3 Paquete client

Este paquete contiene las clases que serán ejecutadas como JavaScript dentro del navegador del cliente. Este paquete es relativamente complejo, contiene todas las clases que implementan el patrón modelo vista presentador así como varios eventos y manejadores de GWT que permiten tomar provecho del soporte de GWT para el historial del navegador.

La Figura 109 muestra las clases que funcionan como polea de arranque para la aplicación en el cliente, estas clases preparan los eventos para que funcione el historial del navegador con las secciones de la aplicación y ceden el control al presentador principal.

- La clase AppLoader es el punto de entrada de la aplicación, crea el bus de eventos de GWT y obtiene el contenedor de todos los componentes visuales, definiendo métodos para que el contendor de inversión de control inyecte estas instancias donde se definan dependencias para ellos.
- La clase HistoryToken permite asociar cadenas del historial, denominadas tokens, con presentador de la arquitectura MVP, para ello define una instancia de HistoryToken por cada sección que será manejada por el historial.
- La clase AppController implementa una interfaz de GWT que define el método ejecutado cuando cambia el valor del historial, cuando esto ocurre obtiene

programáticamente una instancia del presentador adecuado y la ejecuta. El apartado 6.3.6.2 “Cambio del historial” detalla esta secuencia.

- La clase *HistoryEventRegister* registra eventos con sus respectivos manejadores en el bus de eventos de GWT, estos manejadores actualizan el historial programáticamente cuando se ejecutan los presentadores, esto permite volver atrás entre ejecuciones de presentadores (cambios de sección).

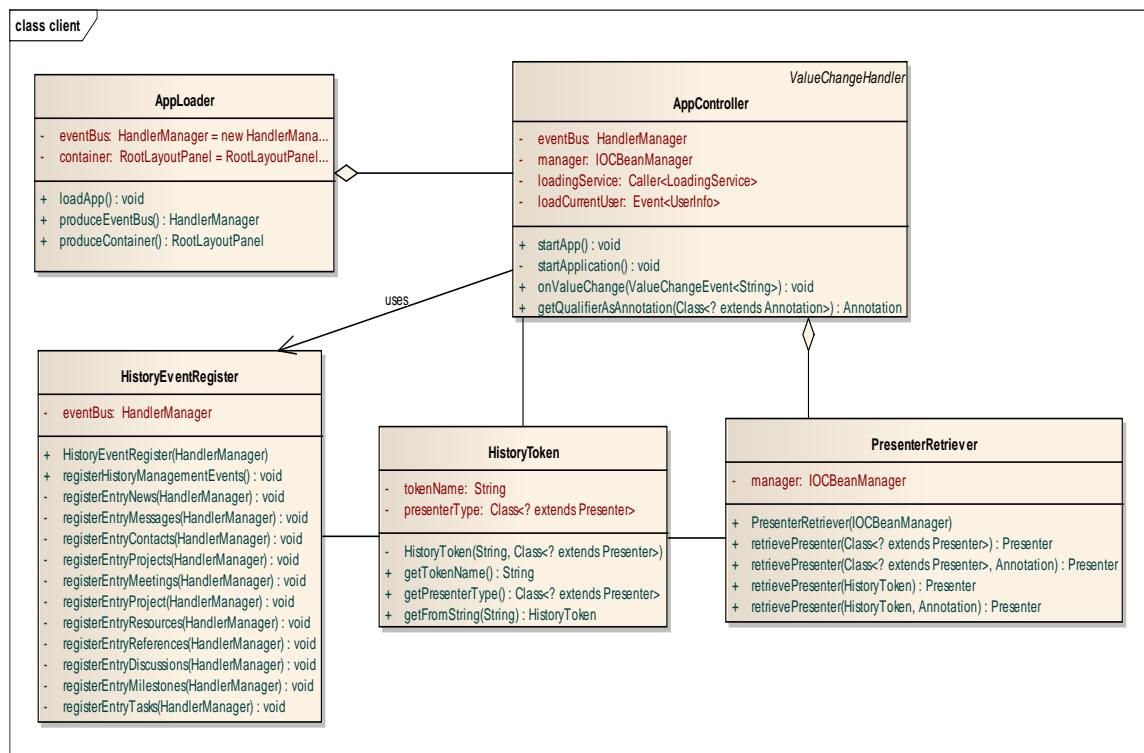


Figura 109: Diagrama de clases – Paquete client

La Figura 110 muestra los eventos y los manejadores definidos para añadir a la aplicación la funcionalidad del historial.

Los eventos definen “lo que sucederá” mientras que los manejadores definen una interfaz para todas las acciones que queremos ejecutar cuando ese evento tenga lugar. Así definimos un evento para la entrada en la sección de notificaciones, y una interfaz que define la operación a ejecutar cuando entremos a esta sección, la clase *HistoryEventRegister* añade implementaciones para estos manejadores que establecen el token adecuado en el historial del navegador. La última pieza de es el presentador de la vista que contiene todas las secciones, este es una especie de presentador maestro que se encarga de lanzar el evento adecuado cuando el usuario entra en una sección. Los nombres de las clases son suficientemente descriptivos del evento que representan, por lo que no se darán más detalles.

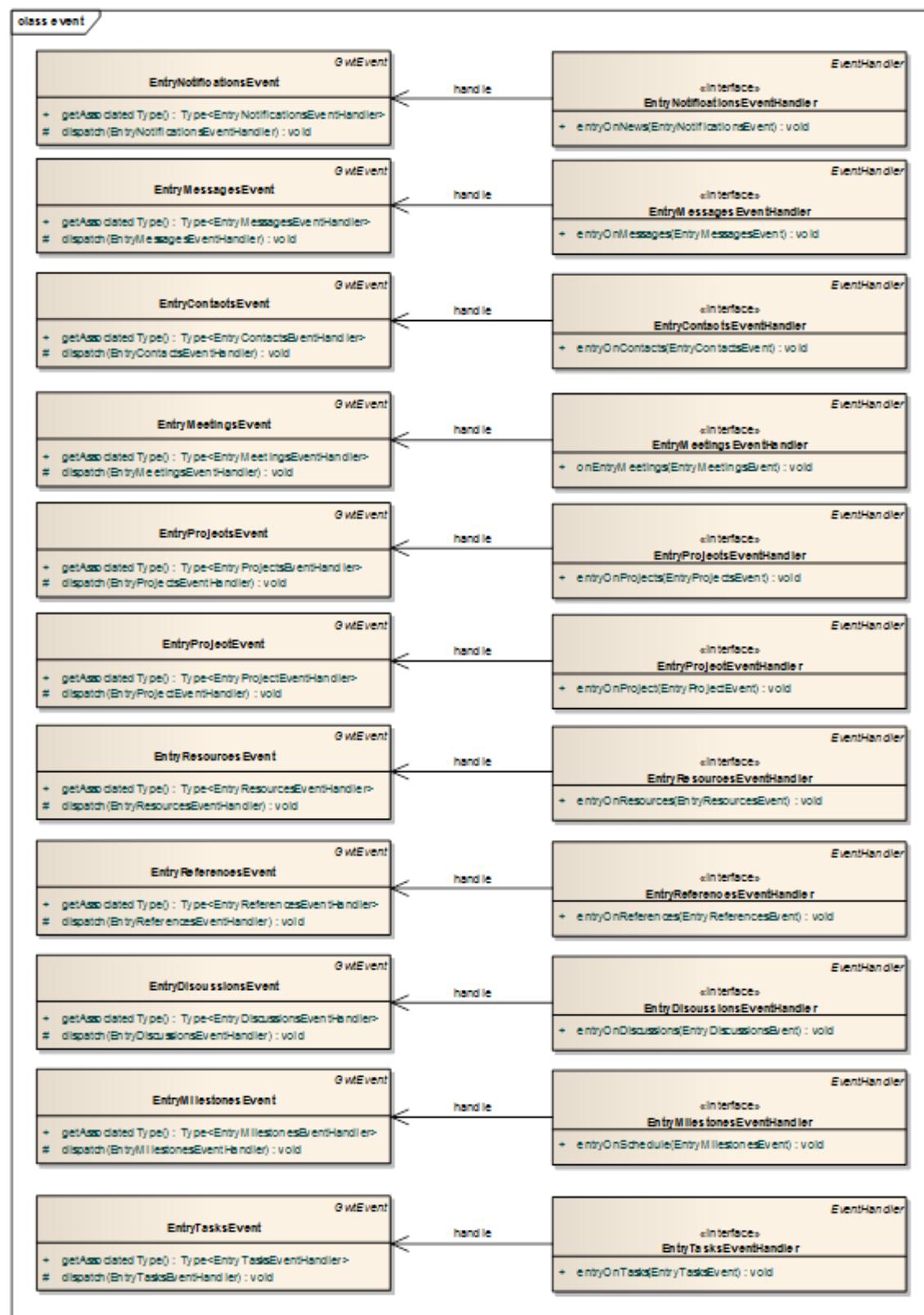


Figura 110: Diagrama de clases – Paquete client.event

Como se dijo previamente, entre las clases que irán al cliente, están todas aquellas que implementan el patrón MVP. Dado que la interfaz de usuario está estructurada en secciones el diseño define un presentador y una vista por cada sección, en la mayoría de los casos estas

secciones estarán compuestas por más secciones, que también tendrán sus presentadores y sus vistas. Todos los presentadores implementan una interfaz común que define un método para presentar la vista ejecutando la lógica necesaria.

La Figura 111 muestra el presentador principal de la aplicación, así como los presentadores de las secciones de primer nivel.

Recordemos que las vistas son construidas de acuerdo al patrón composite view, por lo que una sección, aunque en este formada de otras sub secciones sigue siendo una vista.

- La clase *MainPresenter* lanza eventos de GWT para establecer el valor adecuado en el historial cuando el usuario cambia de sección. Además agrega otros presentadores que coordina de forma directa sin intervención del historial.
- La clase *InfoPresenter* se trata con la lógica de la sección de información.
- La clase *NotificationsPresenter* se encarga de la lógica de la sección de notificaciones, carga las notificaciones del usuario desde el servidor y prepara la vista para mostrarlas. Además se encarga de responder a las interacciones de un usuario con las notificaciones.
- La clase *MessagesPresenter* se encarga de la sección de mensajes, carga los mensajes del usuario y responde a las interacciones de los usuarios con estos.
- La clase *ContactsPresenter* se encarga de la sección de contactos.
- La clase *MeetingsPresenter* trata con la lógica de la sección de reuniones.
- La clase *ProjectsPresenter* se encarga de la sección de contactos.
- La clase *ConversationsPresenter* se encarga de gestionar las conversaciones de un usuario, mantiene un listado de conversaciones y gestiona la selección del usuario para delegar la gestión de una conversación concreta en otro presenter.

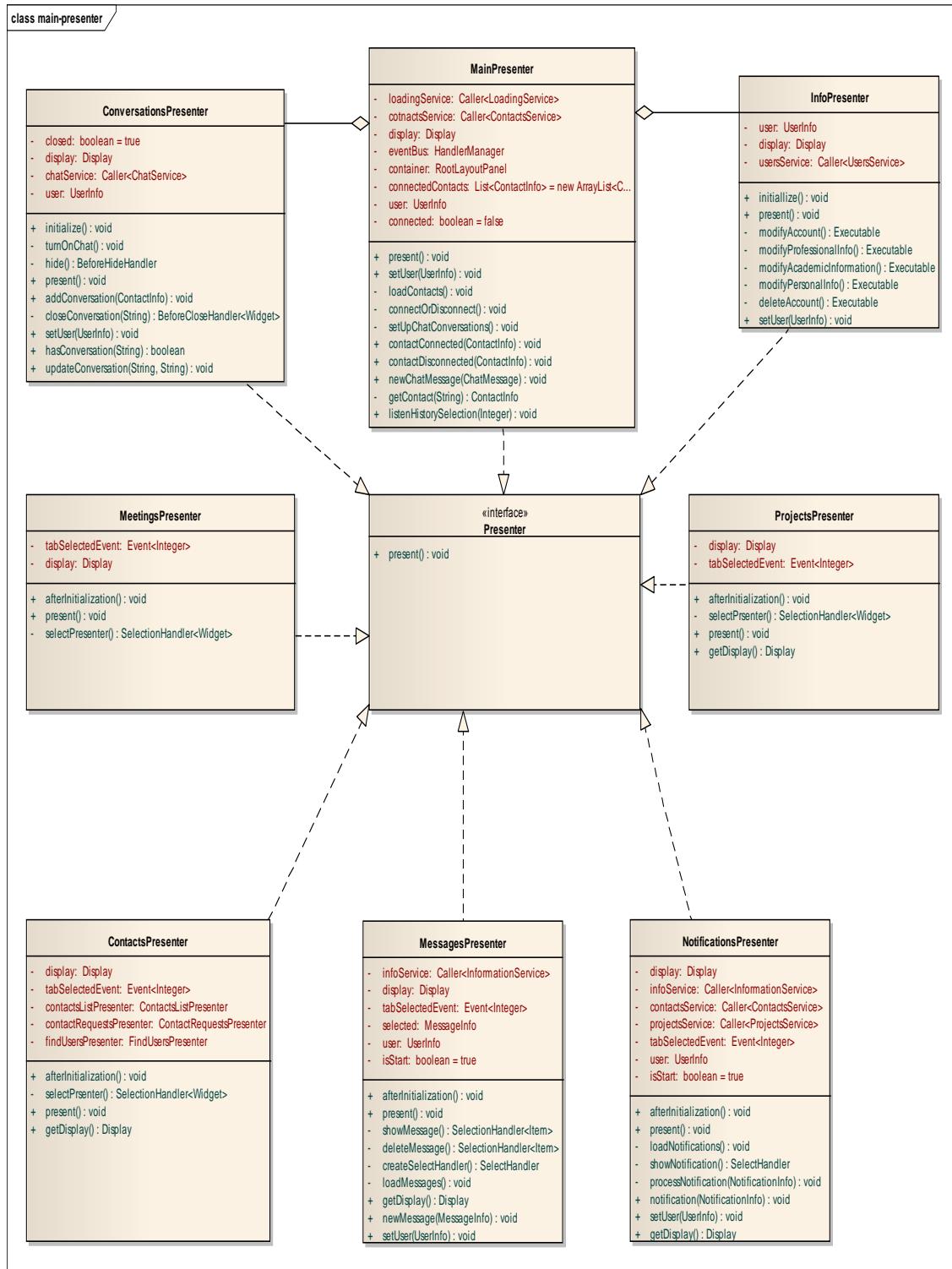


Figura 111: Diagrama de clases – MainPresenter

Además de las distintas secciones, el presentador principal se encarga de coordinar a otros dos presentadores comunes a todas las secciones, ConversationsPresenter e InfoPresenter.

El diagrama de la Figura 112 muestra la clase *ConversationsPresenter* y sus relaciones.

- La clase *ConversationsPresenter* trata con vista en la que hay varias conversaciones, y prepara al *ConversationPresenter* para mostrar la conversación adecuada.
- La clase *ConversationPresenter* trata con una vista en la que el usuario recibe y escribe texto de sus conversaciones con los demás usuarios.

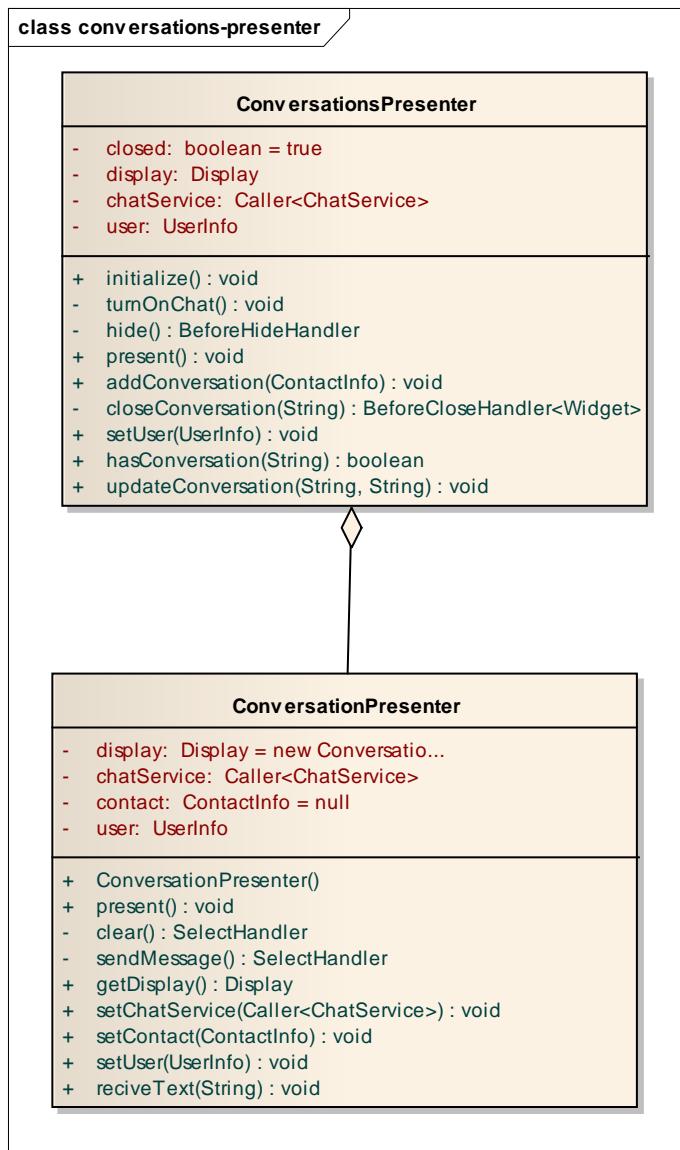


Figura 112: Diagrama de clases - *ConversationsPresenter*

Otra sección que es común a todas las de la sección principal es, la de información, el presentador encargado de esa sección es el *InfoPresenter* (Figura 113).

La clase *InfoPresenter* se encarga de mostrar la información del usuario así como de coordinar otros presentadores para modificar esta.

- La clase *ModifyPersonalInfoPresenter* se encarga de la modificación de la información personal.

- La clase *ModifyAcademicInfoPresenter* se encarga de la vista de modificación de la información académica.
- La clase *ModifyProfessionalInfoPresenter* se encarga de la vista de modificación de la información profesional.
- Finalmente la clase *ModifyAccountInfoPresenter* trata con la vista que permite al usuario la modificación de su cuenta, usuario y contraseña.

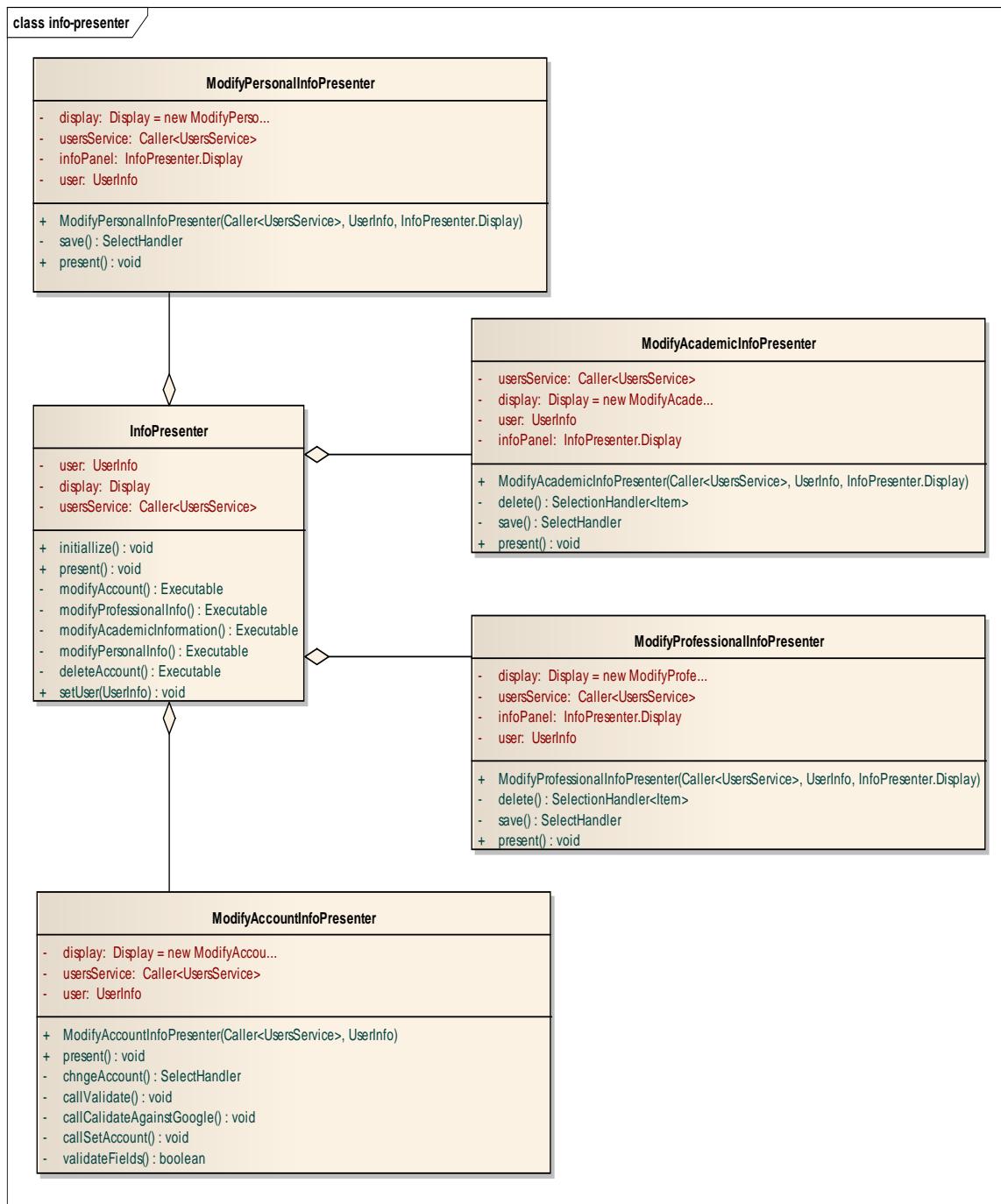


Figura 113: Diagrama de clases - InfoPresenter

La primera sección es la de notificaciones, el presentador de esta sección es el NotificationsPresenter.

- La clase *NotificationPresenter* se encarga de tratar con la vista que muestra una única notificación, para cargar la información en la vista interactúa con otras clases.
- La clase *GetContactCallback* implementa la interfaz *RemoteCallback* de Errai, y contiene el código ejecutado cuando finaliza la llamada remota al método *getContact* de la interfaz *ContactsService*.
- La clase *GetProjectCallback* es análoga pero para el obtener un proyecto.

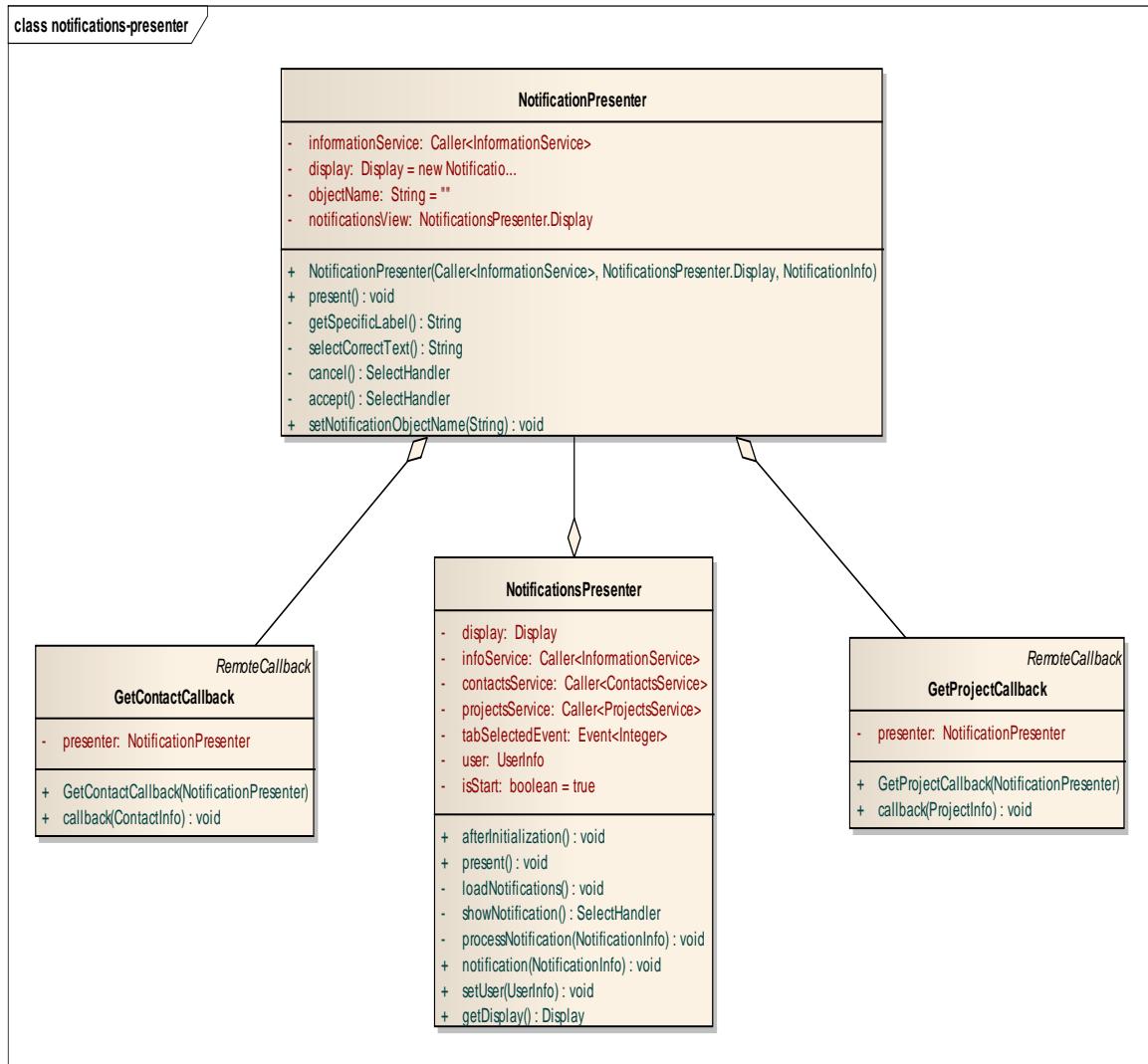


Figura 114: Diagrama de clases – NotificationsPresenter

La Figura 115 muestra las relaciones de la clase *MessagesPresenter*, que se encarga de la sección de mensajes.

- La clase *MessagePresenter* es responsable de la vista de un mensaje, además de recuperar y preparar el mensaje, gestiona la respuesta.

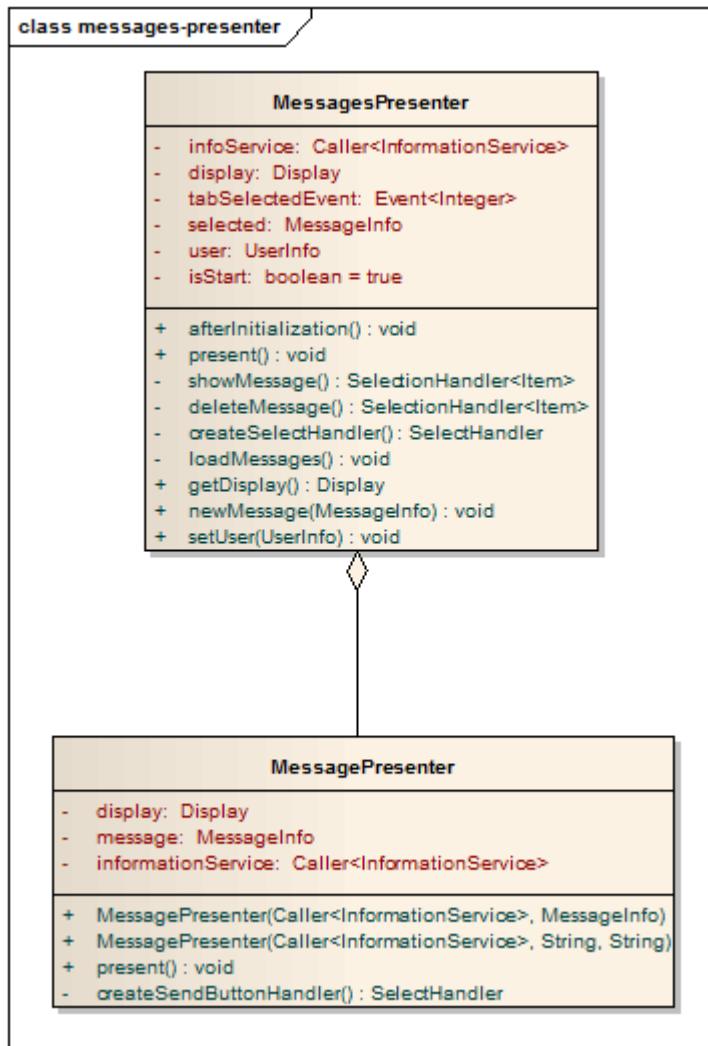


Figura 115: Diagrama de clases – MessagesPresenter

Una sección algo más compleja es la de contactos, esta contiene 3 sub secciones, por ello el *ContactsPresenter* además de ocuparse de la su vista, la sección de contactos, se ocupa de coordinar los otros tres presentadores, que se encargan a su vez de las tres subsecciones.

- La clase *ContactsListPresenter* se encarga de la subsección que contiene el listado de contactos, obtiene los contactos, se los provee a la vista y se ocupa de las acciones que los usuarios realizan sobre los contactos.
- La clase *ContactRequestsPresenter* se encarga la sección de peticiones, es análoga a la del listado de contactos, solo que esta sección contiene un listado de peticiones, con otras opciones lógicamente.
- La clase *FindUsersPresenter* se encarga de la sección de búsqueda de usuarios, obtiene parámetros de búsqueda de la vista, realiza la búsqueda y le proporciona los resultados a la vista.

- La clase *ContactPresenter* se encarga de tratar con la vista de un contacto, esta vista muestra a un usuario información de uno de sus contactos, para ello obtiene los contactos del contacto, los proyectos publicados del contacto, y se los proporciona a la vista.
- Finalmente *ContactProjectPresenter* trata con la vista que muestra a un usuario información de un proyecto de un contacto.

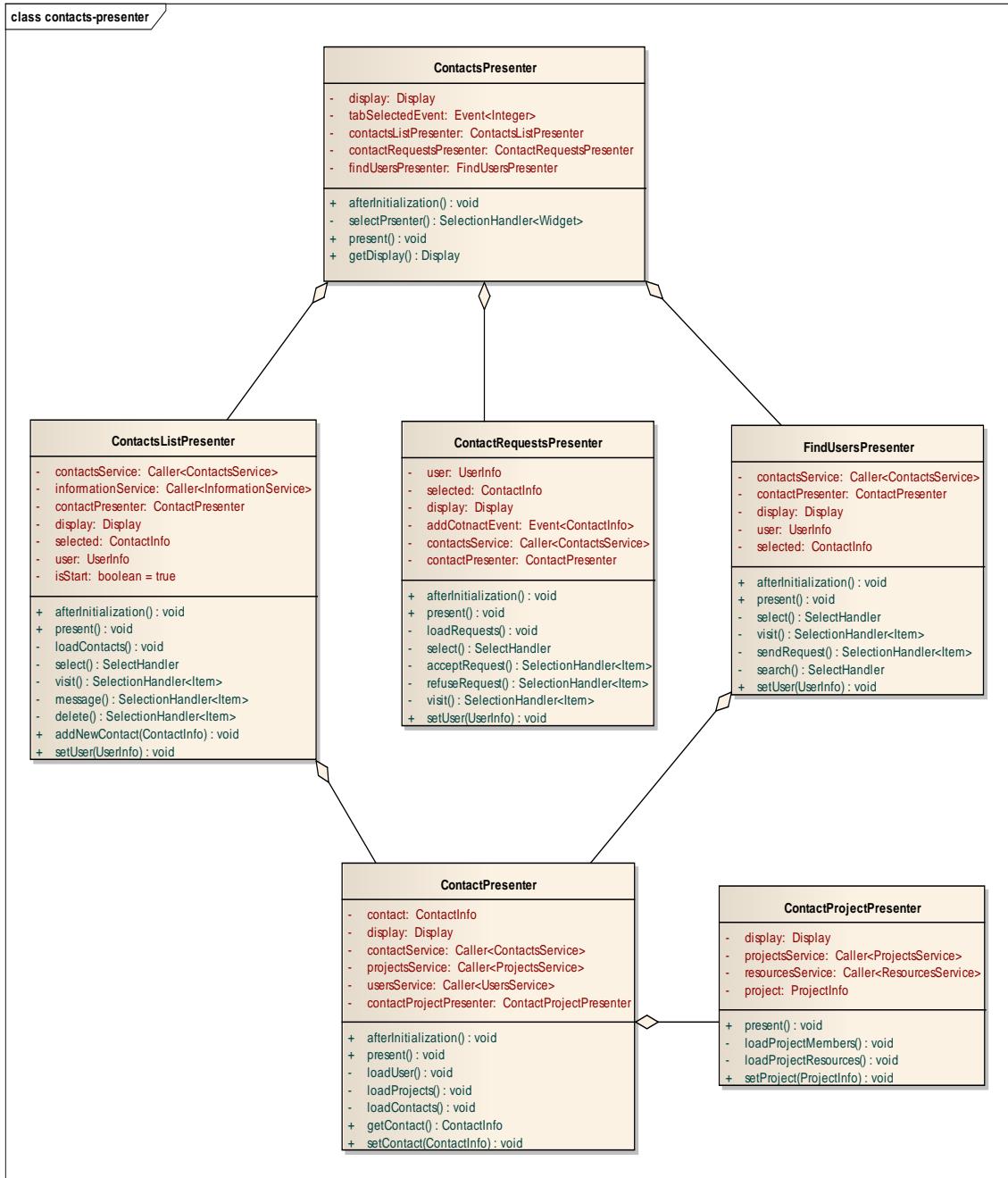


Figura 116: Diagrama de clases – *ContactsPresenter*

La sección de reuniones presenta la misma estructura que la de contactos, un presentador para la sección y otros presentadores para las sub secciones.

- La clase *MeetingsListPresenter* es la encargada de “presentar” la subsección que contiene el listado de reuniones, carga el listado desde el servidor y trata las interacciones del usuario con este.
- La clase *MeetingInvitationsPresenter* trata con el listado de invitaciones a reunión, obtiene las invitaciones y ejecuta las acciones disponibles para cada invitación.
- La clase *NewMeetingPresenter* se encarga de la vista de creación de una reunión, obtiene los parámetros de la nueva reunión e invoca al servidor para crearla.
- La clase *MeetingDetailsPresenter* se encarga de la vista donde se muestran los detalles de una reunión.

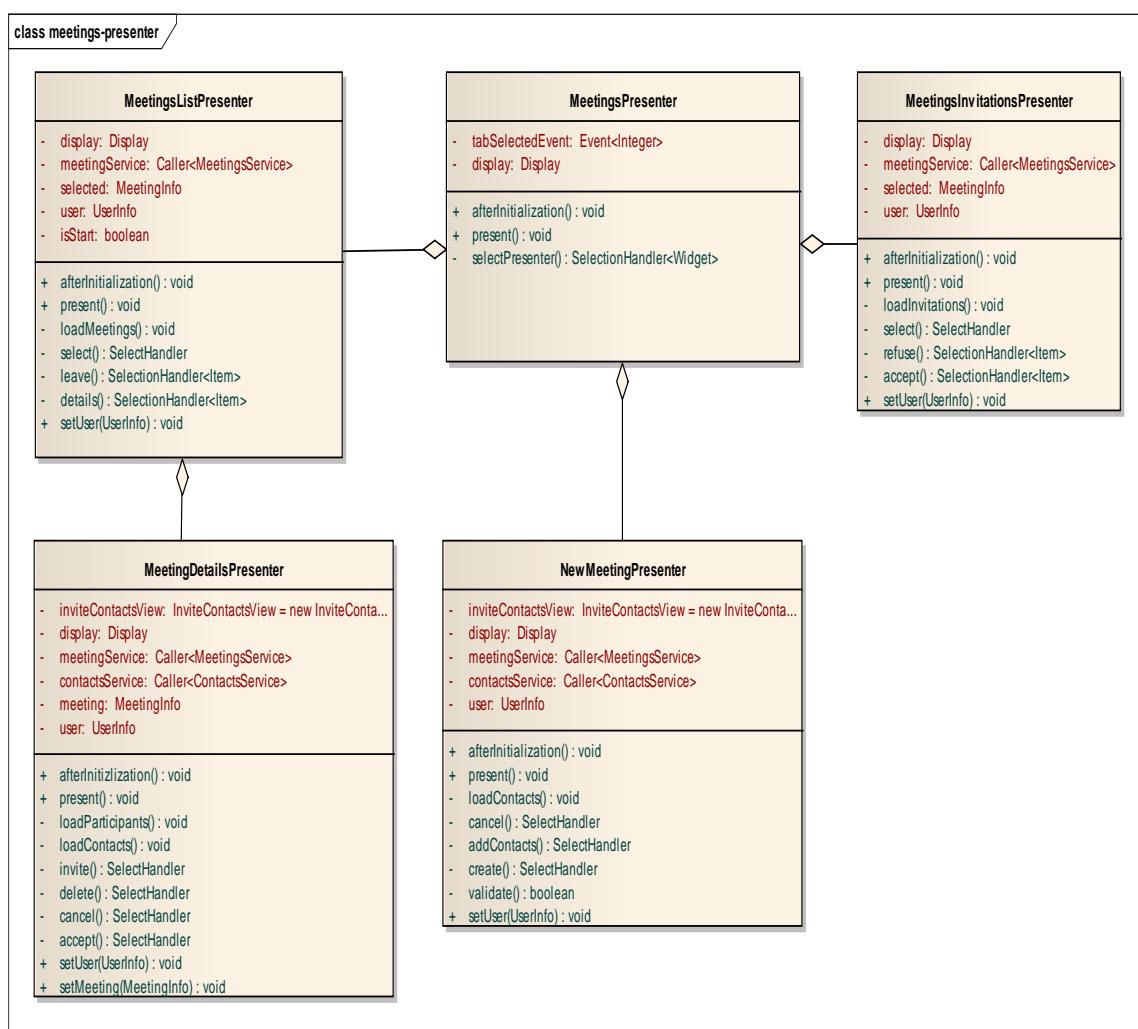


Figura 117: Diagrama de clases – MeetingsPresenter

La última sección dentro de la vista principal es la sección de proyectos, esta sección tiene también varias subsecciones, para listar proyectos e invitaciones a proyectos, y para crear nuevos proyectos y buscar proyectos. Desde esta sección podremos ir a la otra gran sección de la aplicación, la sección del proyecto.

- La clase ProjectsListPresenter se encarga de la subsección del listado de proyectos, entre las opciones que gestiona está la de abrir el proyecto.
- La clase ProjectInvitationsPresenter hace lo propio para las invitaciones a proyecto, obtiene las invitaciones, y trata las opciones de aceptar y rechazar estas.
- La clase FindProjectsPresenter trata con la vista de búsqueda de proyectos, recoge los parámetros introducidos en la vista y ejecuta la búsqueda, actualizando los resultados a la vista.
- La clase NewProjectPresenter, está a cargo de la sección de creación de nuevos proyectos.
- Finalmente la clase ProjectPresenter que es la encargada de toda la sección del proyecto, es muy importante ya que juega el mismo papel que la clase MainPresenter, para dar soporte al historial sobre las pestañas de la sección del proyecto.

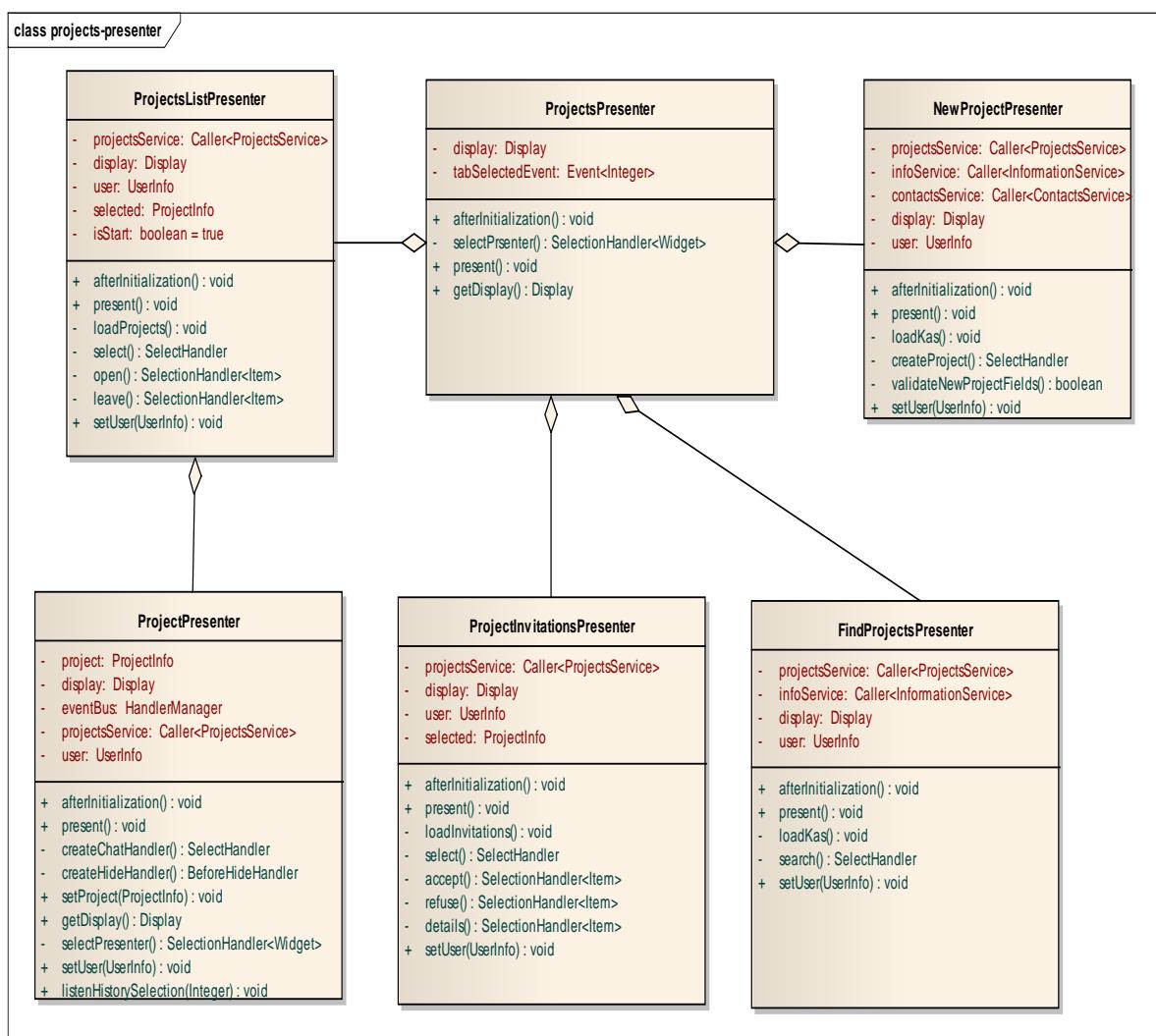


Figura 118: Diagrama de clases – ProjectsPresenter

La Figura 119 muestra todos los presentadores de primer nivel de la sección del proyecto, la sección del proyecto está compuesta de seis secciones, como se especificó en el apartado 5.4 más atrás “Análisis de Interfaces de Usuario”, se ha definido un presentador para cada sección.

- La clase *ProjectPresenter* además de ocuparse de una vista, se encarga de estar atenta a la sección seleccionada y lanzar el evento de GWT adecuado para la gestión del historial. Cada sección está asociada con un token del historial por lo que se puede retroceder o volver hacia adelante pro las secciones del proyecto.
- La clase *ManagementPresenter* se encarga de la vista de administración del proyecto, además de coordinar los presentadores de las diferentes sub secciones.
- La clase *ResourcesPresenter* se encarga de la vista de recursos de la vista del proyecto, además de coordinar los presentadores de las diferentes sub secciones.
- La clase *ReferencesPresenter* se encarga de la vista de referencias del proyecto, además de coordinar los presentadores de las diferentes sub secciones.
- La clase *DiscussionsPresenter* se encarga de la vista de discusiones del proyecto, además de coordinar los presentadores de las diferentes sub secciones.
- La clase *MilestonesPresenter* se encarga de la vista de hitos del proyecto, además de coordinar los presentadores de las diferentes sub secciones.
- La clase *TasksPresenter* se encarga de las vista de tareas, además de coordinar los presentadores de las diferentes sub secciones.

En posteriores diagramas se detalla cada uno de estos y su relación con los presentadores de las diferentes subsecciones.



Figura 119: Diagrama de clases – ProjectPresenter

La sección de administración es común al resto, que son excluyentes entre sí.

- La clase *ModifyProjectPresenter* se encarga de una sub vista de esta sección, destinada a la modificación de la información del proyecto.
- La clase *InvitationsPresenter* trata con una vista para invitar contactos al proyecto.
- La clase *MemebersPresenter* se encarga de una vista con opciones de administración sobre los miembros.

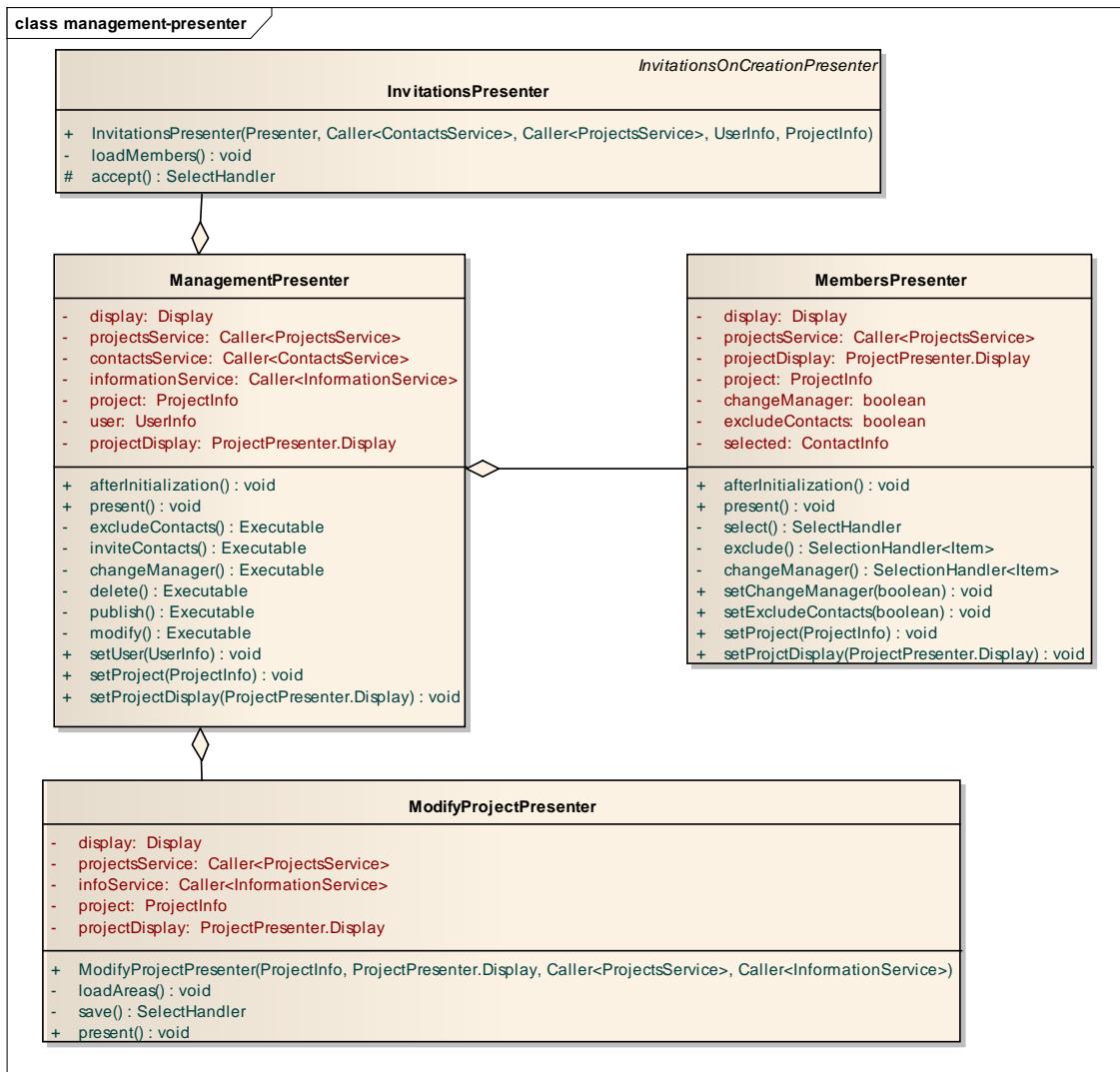


Figura 120: Diagrama de clases – ManagementPresenter

La sección de recursos no cuenta con subsecciones, por lo que no hay más presentadores que el de primer nivel. En su lugar cuenta con una varias clases que implementan diferentes funcionalidades sobre los recursos, en la Figura 121 se pueden ver detalladas estas clases, que se describen brevemente a continuación.

- La clase *CreateResourceHandler* se encarga de la creación de un recurso, los pasos a seguir para la creación de un recurso se detallan en el apartado 6.4 “Diagramas de Actividad y de Estado”.
- La clase *MoveResourceHandler* se encarga del movimiento de un recurso, para ello se debe modificar tanto la vista como el recurso en negocio.
- La clase *ShowResourceHandler* se encarga de mostrar un recurso, debe obtener el contenido del recurso y mostrarlo.
- La clase *DeleteReosurceHandler* se ocupa de eliminar un recurso.
- La clase *MoveLockerHandler* se encarga de evitar que un recurso sea movido de forma inválida, por ejemplo que la raíz del árbol sea movida.
- La clase *EditResourceHandler* se encarga de obtener el contenido de un recurso y prepararlo para su edición.
- La clase *UploadResourceHandler* e encarga de invocar mostrar la vista de subir un recurso y preparar la invocación al Servlet.
- La clase *RenameResourceHandler* se encarga de mostrar un mensaje con una entrada para un nuevo nombre, validarla y cambiarlo.
- La clase *DownloadResourceHandler* se encarga de invocar a un Servlet para que descargue un recurso dado su identificador.
- La clase *ImportResourceHandler* se encarga de obtener los recursos de Google Docs y mostrarlos, así como de importar el seleccionado.
- La clase *HandlerConfigurer* define métodos para añadir instancias de cada uno de los manejadores anteriores, a elementos que sean capaces de ejecutarlo cuando tenga lugar un evento determinado.

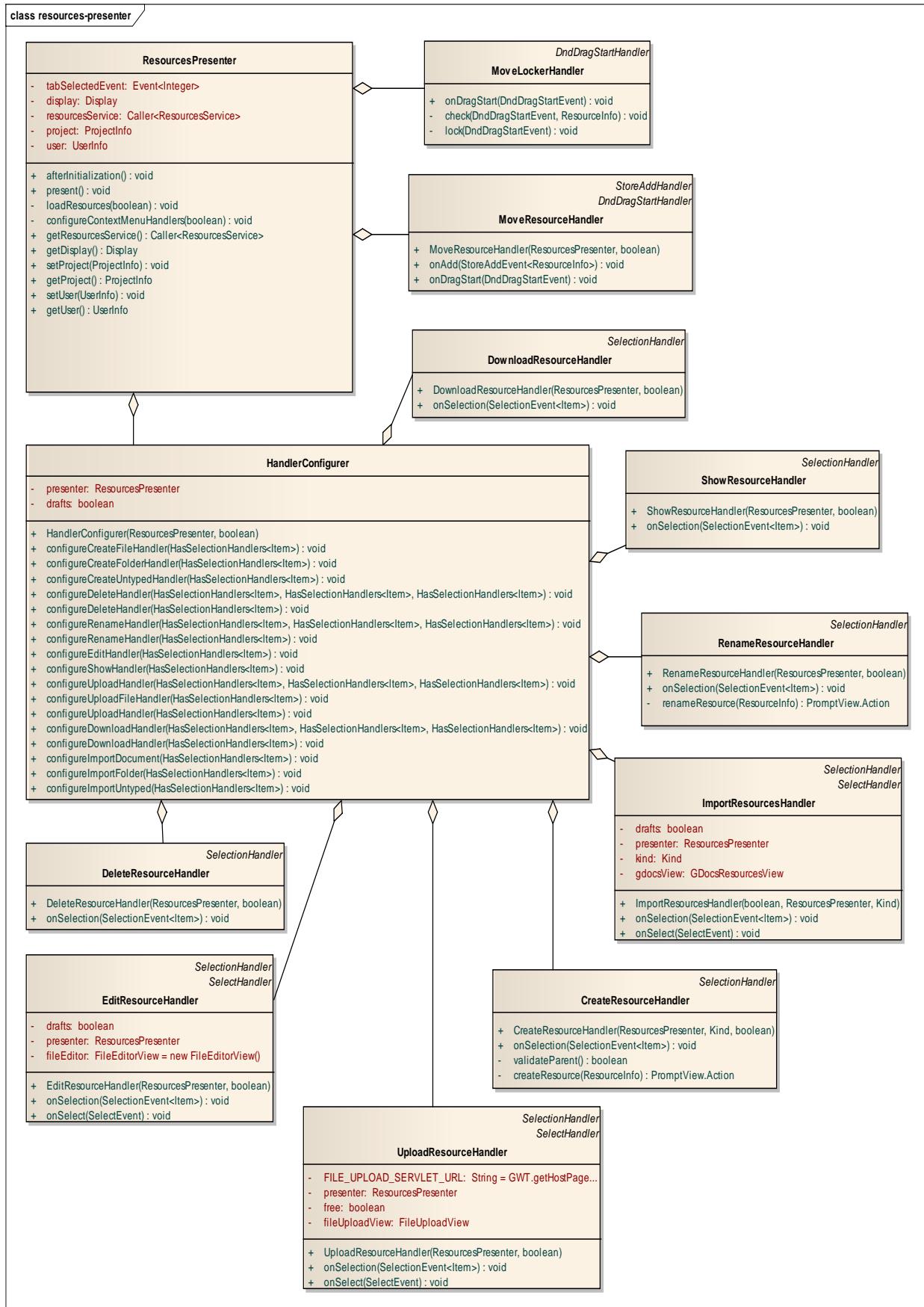


Figura 121: Diagrama de clases – ResourcesPresenter

La sección de referencias retoma el esquema de presentador con sub presentadores.

- La clase *ReferencesListPresenter* se encarga de la sub sección del listado de referencias, obtiene las referencias se las establece a la vista y trata las acciones del usuario sobre el listado de referencias.
- La clase *NewReferencePresenter* trata la vista de creación de referencias, obtiene los valores de la vista y crea las referencias.

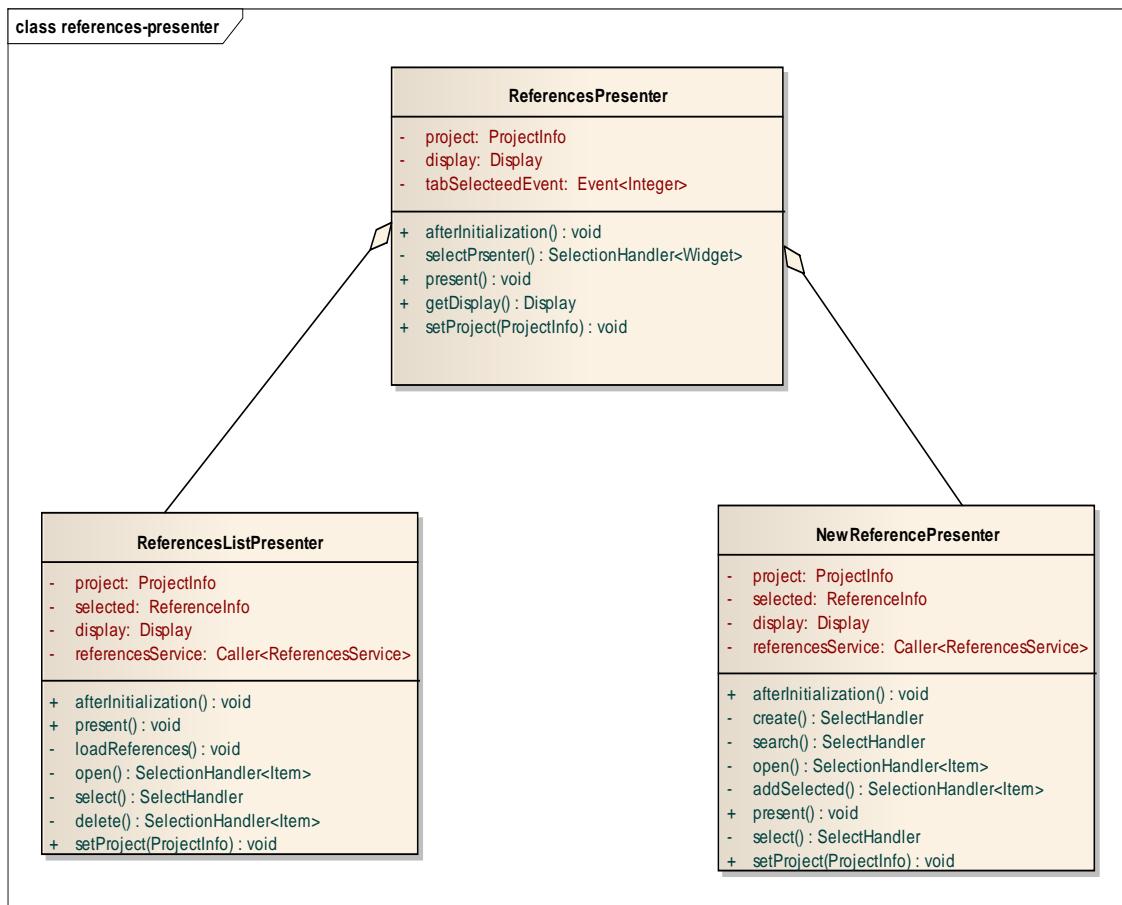


Figura 122: Diagrama de clases – *ReferencesPresenter*

La sección de discusiones sigue el mismo esquema, en el que un presentador se encarga de coordinar al resto de presentadores, encargados de las subsecciones.

- La clase *NewDiscussionPresenter* se encarga de la creación de una nueva discusión.
- La clase *DiscussionListPresenter* se encarga de la vista que contiene un listado de presentadores.
- La clase *DiscussionDetailsPresenter* se encarga de la vista donde se muestran los detalles de una discusión.
- La clase *DiscussionVotePresenter* trata con la vista que permite votar a los usuarios.

- La clase *DiscussionResultsPresenter* se ocupa de la vista de los resultados de la votación.

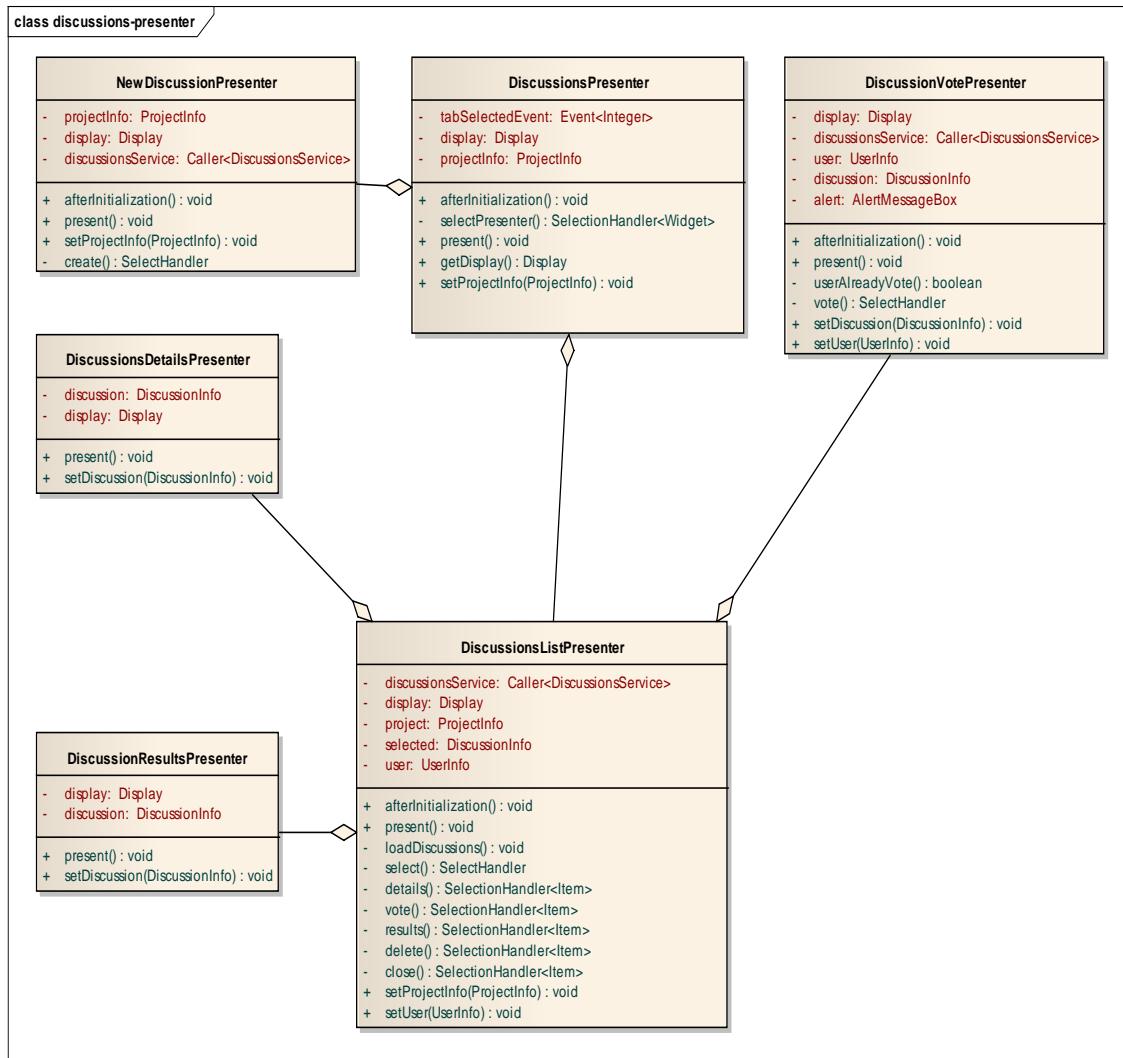
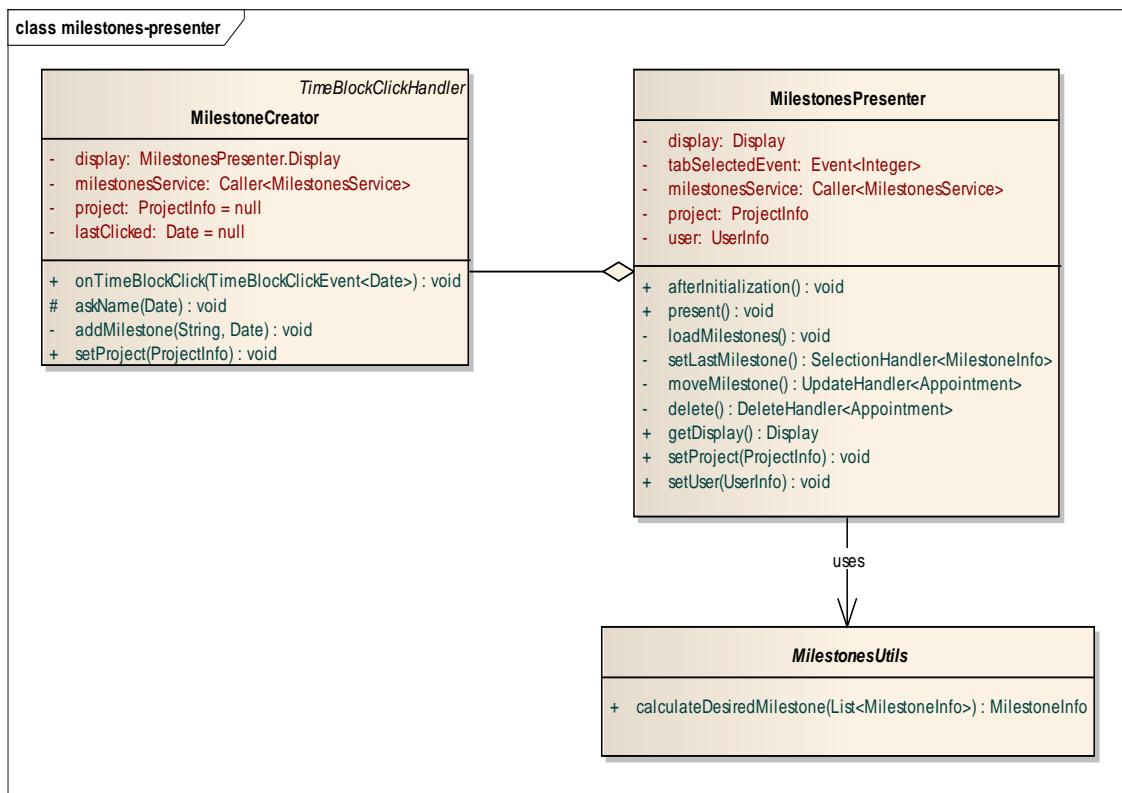


Figura 123: Diagrama de clases – *DiscussionsPresenter*

La Figura 124 muestra el presentador de la vista de hitos, y las clases con que se relaciona.

- La clase *MilestoneCreator* se ocupa de crear hitos, implementa una interfaz específica del componente usado para representar los hitos, un calendario, esta clase crea hitos al hacer doble click en las fechas del calendario.
- La clase *MilestoneUtils* permite realizar algunos cálculos con los hitos, concretamente calcula el hito que debería haberse cumplido para la fecha.

Figura 124: Diagrama de clases – *MilestonesPresenter*

La última sección de un proyecto es la de tareas, los presentadores de esta sección son los mostrados en la Figura 125, y son gestionados por la clase *TasksPresenter*.

- La clase *TasksListPesenter* es la encargada de recuperar las tareas y proporcionárselas a la vista, además debe responder a las acciones que los usuarios lleven a cabo sobre el listado.
- La clase *NewTaskPresenter* se ocupa de crear nuevas tareas con los datos obtenidos de la vista que maneja.
- Finalmente la clase *TaskSummaryPresenter* trata con una vista que muestra un resumen de las tareas.

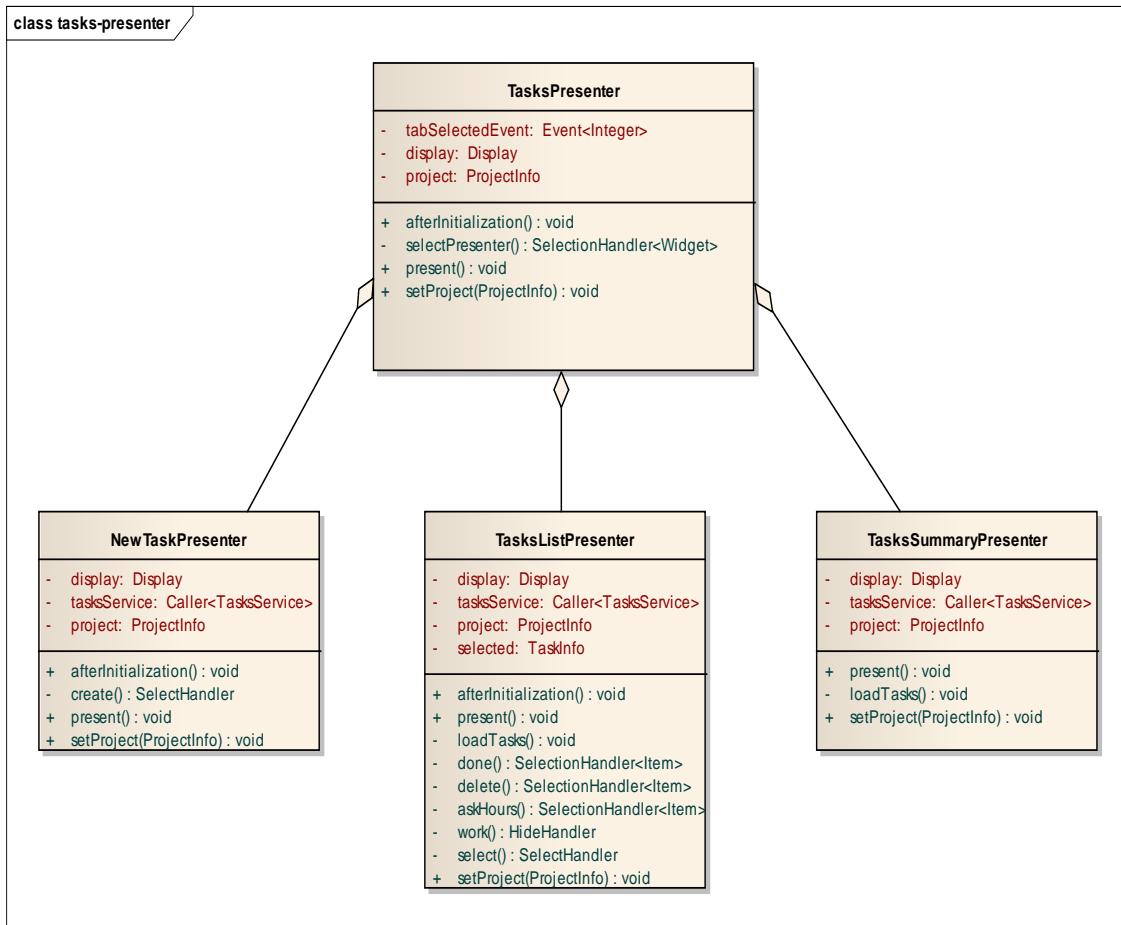


Figura 125: Diagrama de clases – TasksPresenter

6.2.4.3 Diagramas de Clases de cnpd-client

Este paquete se encuentra en el nivel del cliente, y sus clases implementan una aplicación de escritorio, toda la lógica la obtienen mediante invocaciones remotas a la capa de negocio por lo que su diseño es muy simple.

La aplicación sigue el patrón modelo vista presentador donde los presentadores se encargan de invocar a la capa de negocio para responder a las peticiones de las vistas.

- La clase *Launcher* se encarga de inicializar el contenedor de inyección de dependencias y enviar un evento a la clase *AppLoader* que cargará la aplicación.
- La clase *AppLoader* creará la ventana principal de la aplicación (clase *BaseFrame*) e invocará al presentador principal (*MainPresenter*).
- La clase *MainPresenter* se encargará de ceder el control al presentador adecuado según la sección de la aplicación en la que se esté.
- La clase *MainView* muestra una vista que permite seleccionar entre cuatro opciones.

- Las clases *ListUsersPresenter* y *ListUsersView* definen la vista y el correspondiente presentador para una sección en la que se pueden listar los contactos, ver sus detalles y eliminarlos.
- Las clases *AreasPresenter* y *AreasView* definen la vista y el correspondiente presentador para una sección en la que se pueden listar las áreas de conocimiento, así como también eliminarlas.
- Las clases *NewUserPresenter* y *NewUserView* definen la vista y el correspondiente presentador para una sección en la que se pueden crear nuevos usuarios.
- Las clases *NewAreaPresenter* y *NewAreaView* definen la vista y el correspondiente presentador para una sección en la que se pueden crear nuevas áreas de conocimiento.
- Finalmente la clase *UserDetailsView* es un dialogo que muestra los detalles de un usuario y la clase *UserFields* encapsula todos los campos de un usuario, para reutilizarlos en varias vistas.

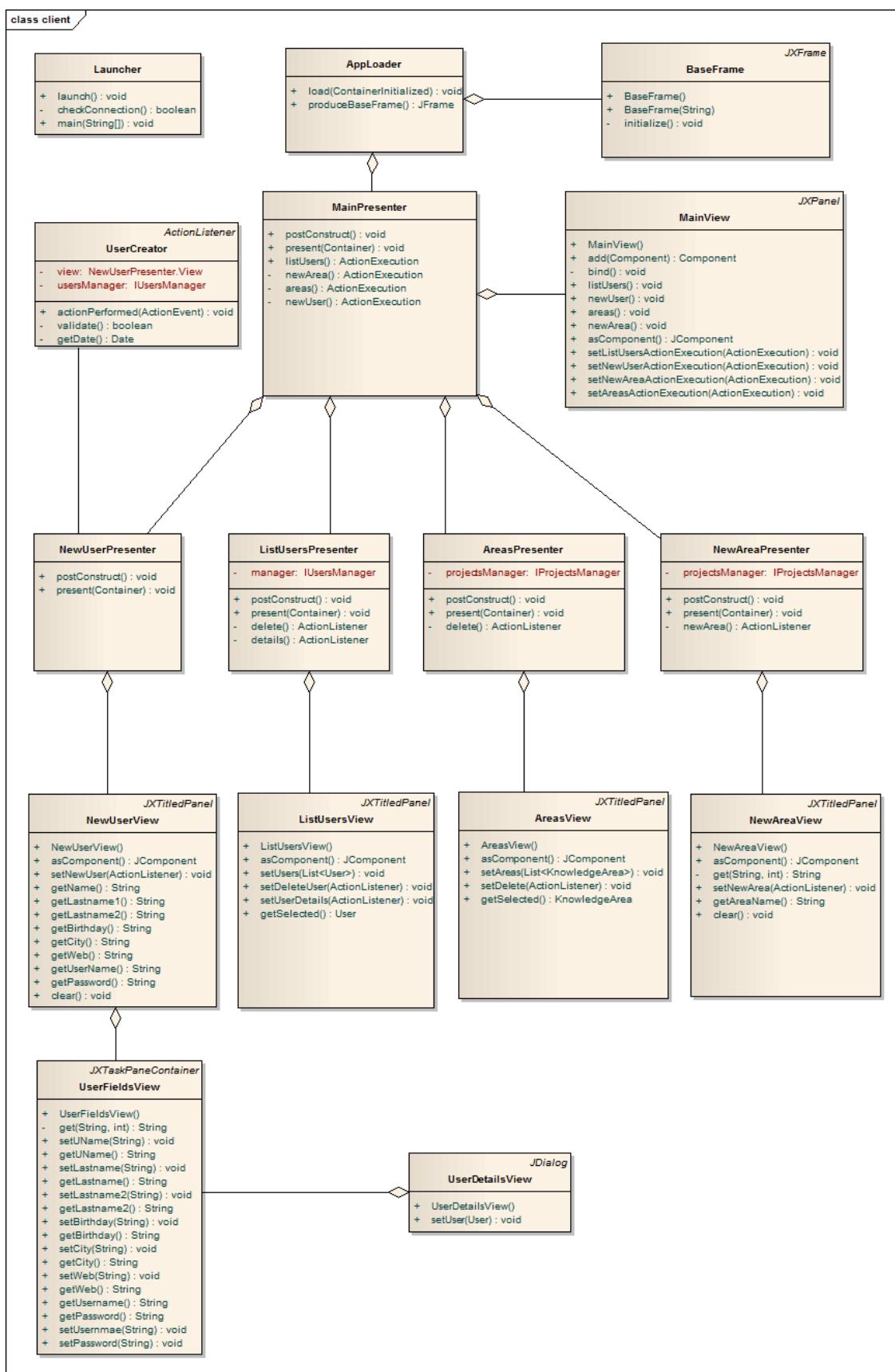


Figura 126: Diagrama de clases - Paquete client

6.3 Diagramas de Secuencia

Para comprender mejor algunas secuencias del sistema, se han realizado algunos diagramas de secuencia. Estas secuencias incluyen procesos implicados en la resolución de los casos de uso y otros necesarios para implementar aspectos más técnicos como la autenticación.

6.3.1 Gestión de Usuarios

En este apartado se detallan los procesos más complejos, involucrados en la gestión de los usuarios, obviando los que carecen de complejidad.

6.3.1.1 Eliminar usuario

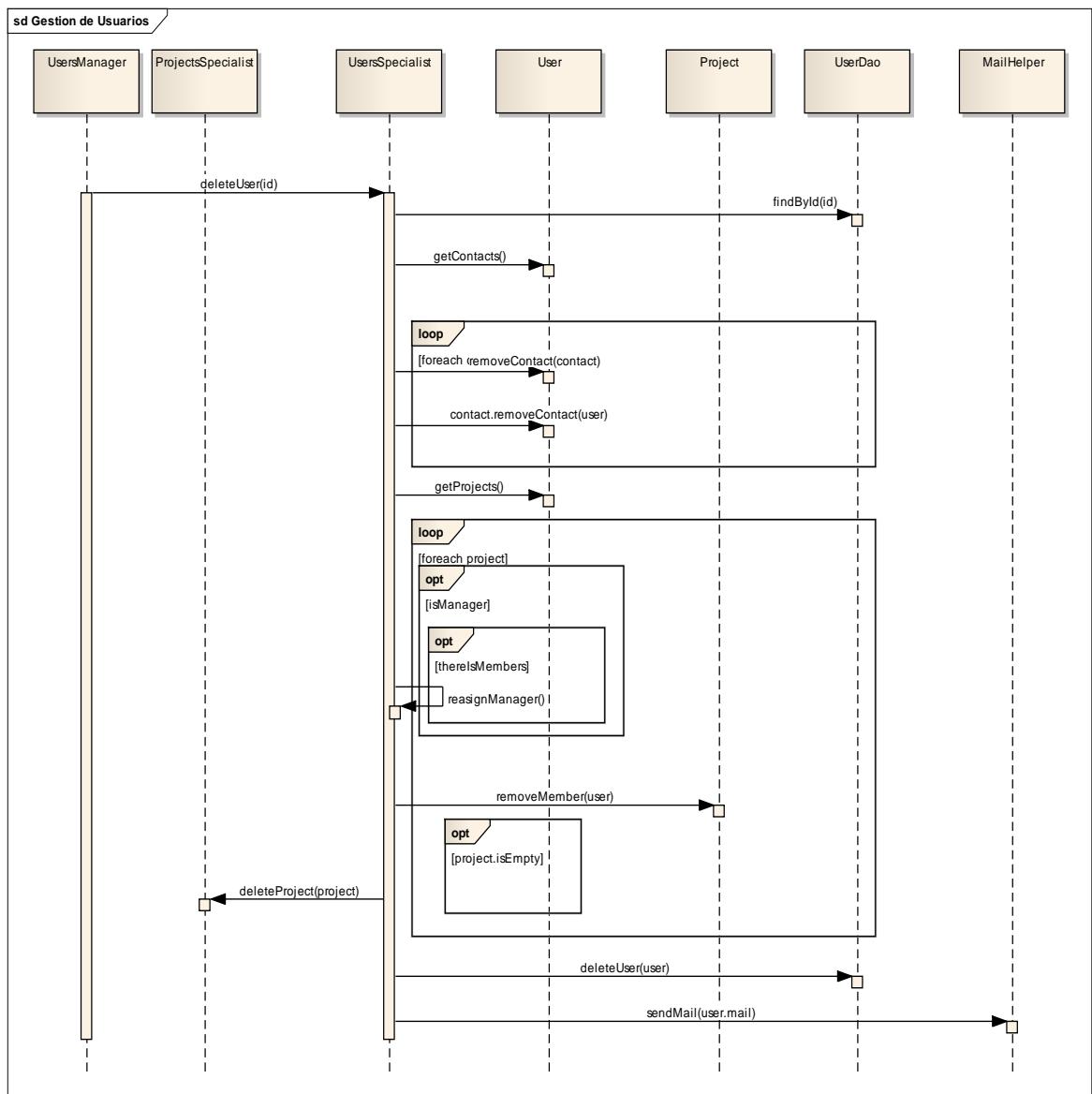


Figura 127: Diagrama de secuencia – Eliminar usuario

1. Se obtiene el usuario que se quiere eliminar de la base de datos.
2. Se elimina el usuario a eliminar de las listas de contactos de los usuarios que lo tengan.
3. Comprobar sus proyectos se comprueban todos sus proyectos, y si fuese el gestor de alguno se reasigna un gestor, en cualquier caso se elimina como miembro del proyecto y si el proyecto queda sin miembros se elimina también.
4. Se elimina el usuario del sistema.
5. Se envía un mail al usuario confirmando la eliminación.

6.3.2 Gestión de Contactos

6.3.2.1 Añadir Contacto

El proceso de añadir un contacto, tiene dos pasos, el envío de una petición y la resolución de esa petición.

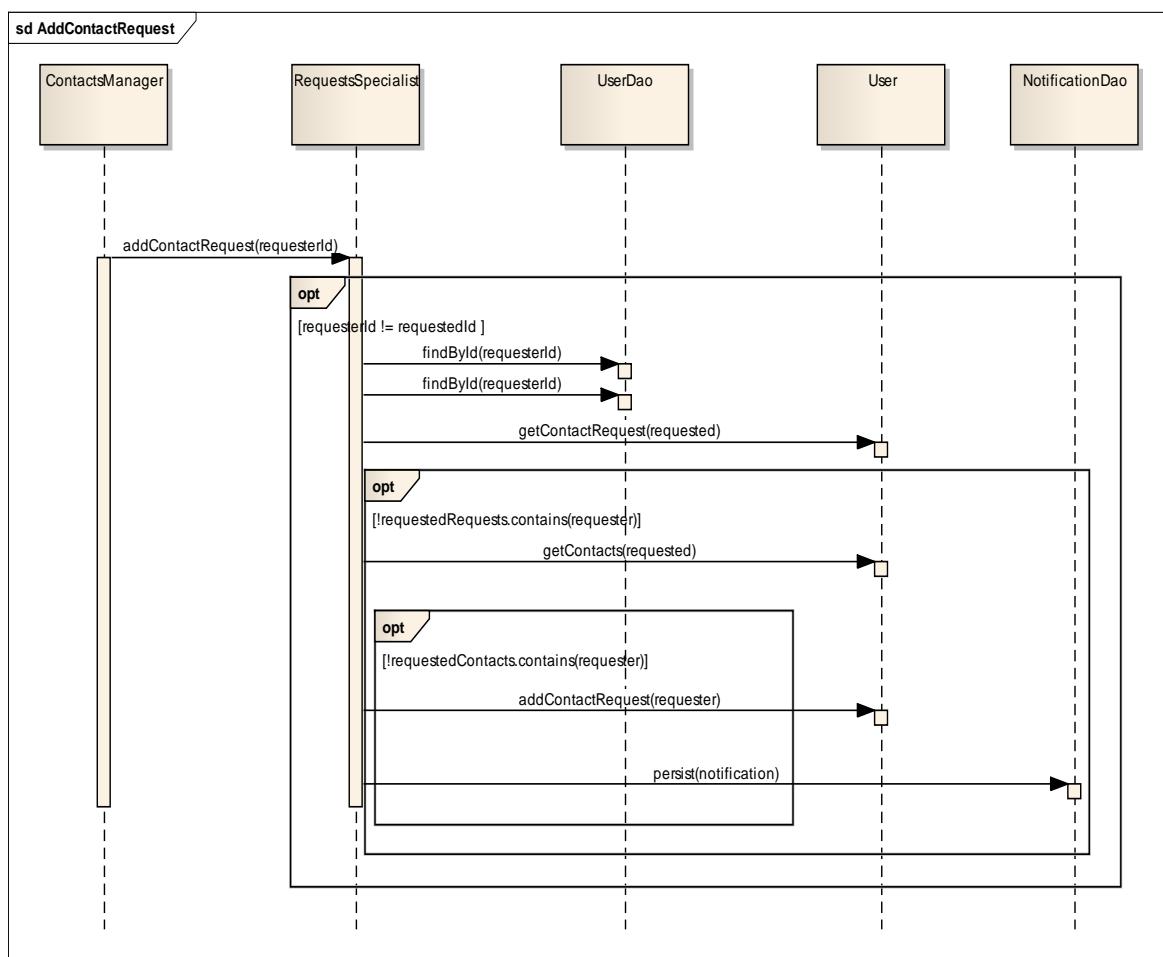


Figura 128: Diagrama de Secuencia – Añadir Petición

1. Se comprueba que no la petición no es reflectiva.
2. Se recuperan ambos contactos de la base de datos.
3. Se recuperan las peticiones del contacto solicitado y se comprueba que no tiene ya una del solicitante.
4. Se recuperan los contactos y se comprueba que la petición no es de un contacto.
5. Se añade la petición a la lista de peticiones del usuario solicitado.
6. Se crea una notificación para el usuario solicitado.

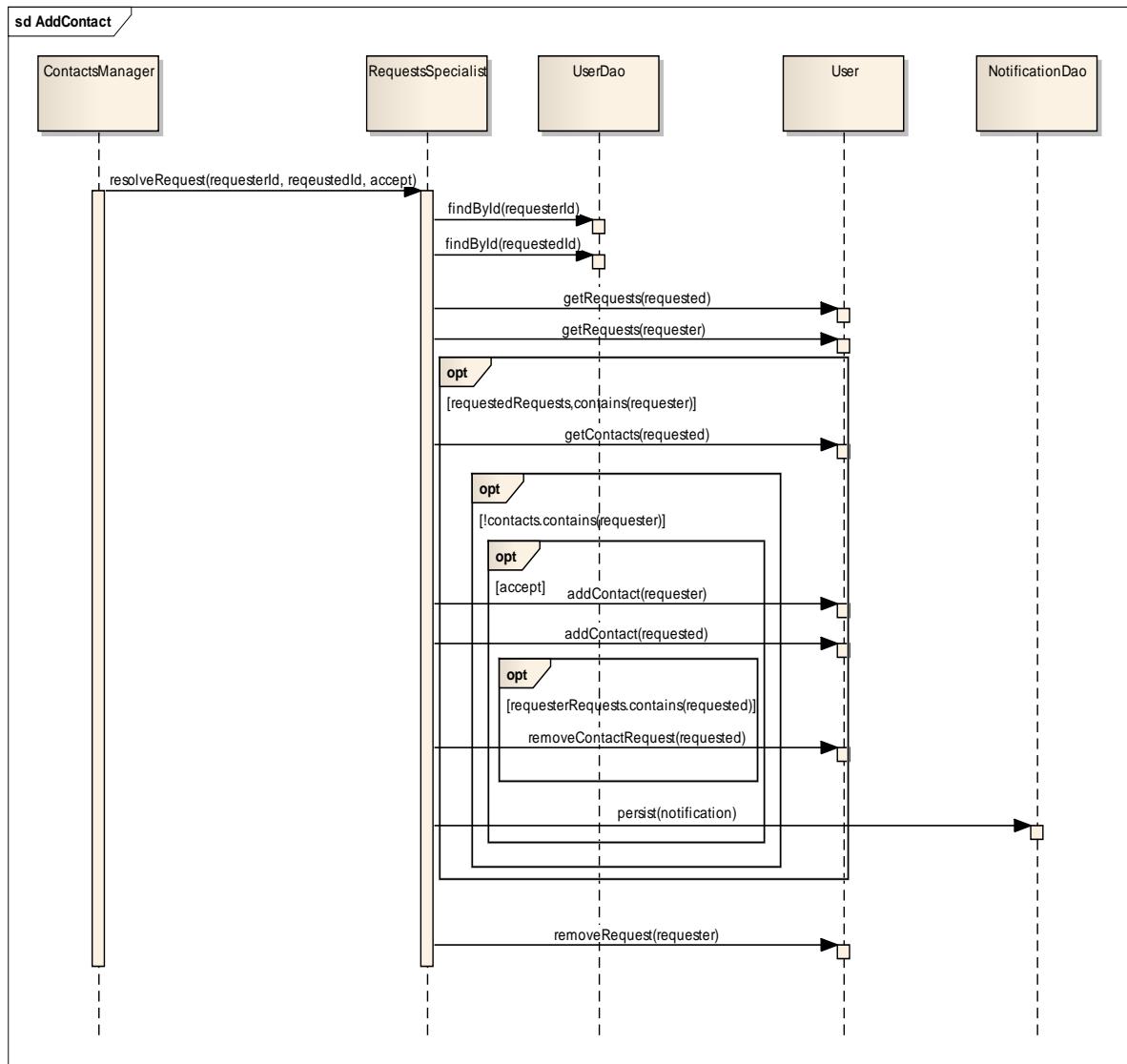


Figura 129: Diagrama de Secuencia – Añadir Contacto

El proceso de añadir un contacto y sus casos se detallan mediante un diagrama de actividad en el apartado 6.4.1 “Añadir Contacto”, solamente para el supuesto de que ya se ha realizado una petición, que se ha considerado el más crítico y complicado.

6.3.3 Gestión de Proyectos

Algunos proceso complejos, relacionados con la gestión de proyectos, se detallan en los siguientes diagramas.

6.3.3.1 Eliminar Proyecto

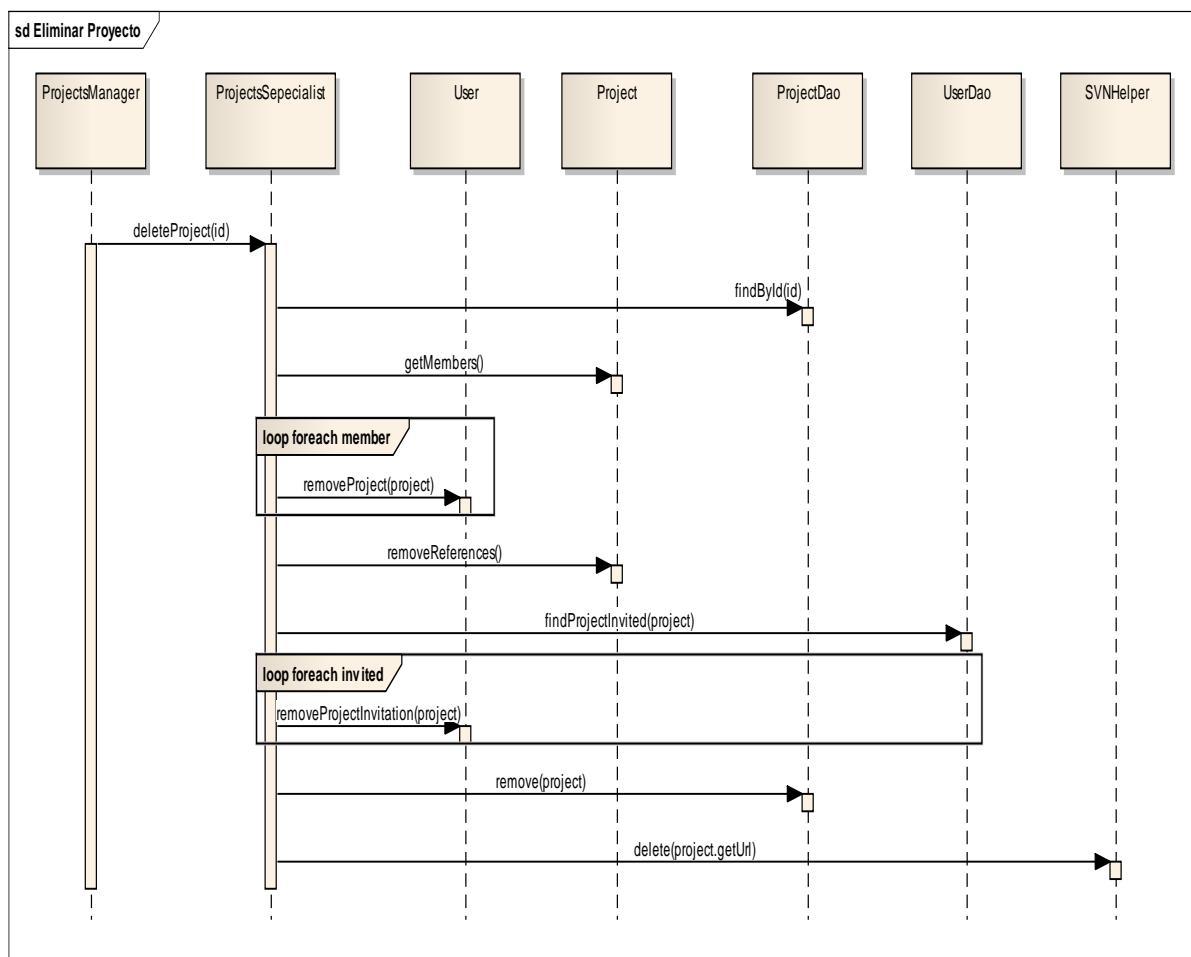


Figura 130: Diagrama de secuencia – Eliminar Proyecto

1. Se obtiene el proyecto a eliminar de la base de datos.
2. Se obtienen los miembros del proyecto y se elimina el proyecto de su lista.
3. Se obtienen los usuarios invitados al proyecto y se elimina el proyecto de su lista de invitaciones.
4. Se elimina el proyecto.
5. Se eliminan los ficheros del proyecto.

6.3.4 Gestión de Recursos

6.3.4.1 Crear Recurso

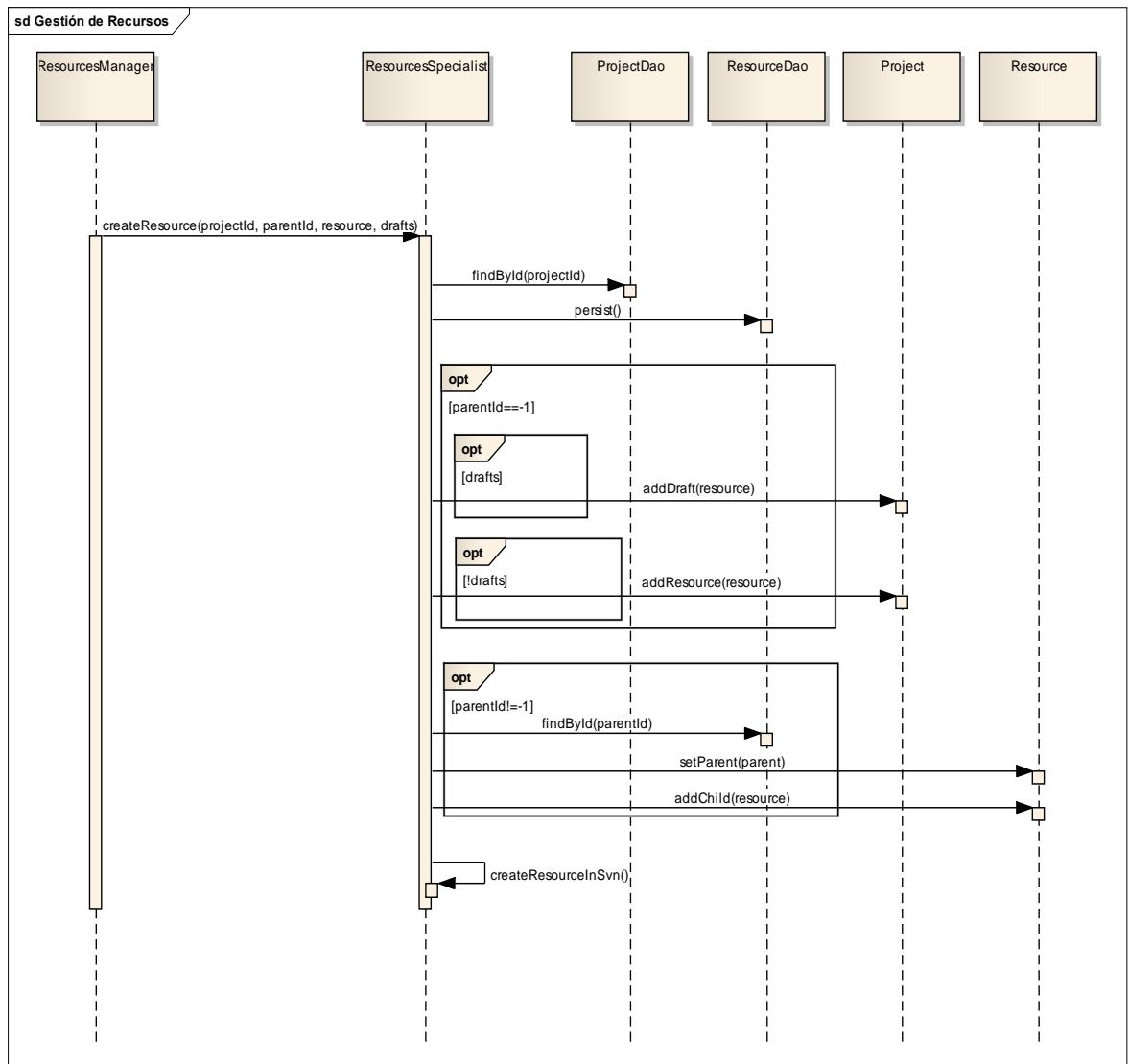


Figura 131: Diagrama de Secuencia – Crear Recurso

1. Obtener el proyecto sobre el que se está creando el recurso.
2. Crear el recurso en base de datos.
3. Asignar el recurso a su padre, si el id del padre es menos uno se asignará al proyecto, como borrador o recurso. Si tiene un id de padre valido se recupera el padre y se añade el parent al recurso y el recurso al parent.
4. Crear el recurso en el servidor de ficheros.

6.3.5 Autenticación

La autenticación del sistema tiene lugar de una forma un tanto particular ya que los usuarios están en un SGBD al que solo se accede a través de una fachada de negocio remota. El siguiente diagrama muestra los pasos que tiene lugar para llevar a cabo este proceso.

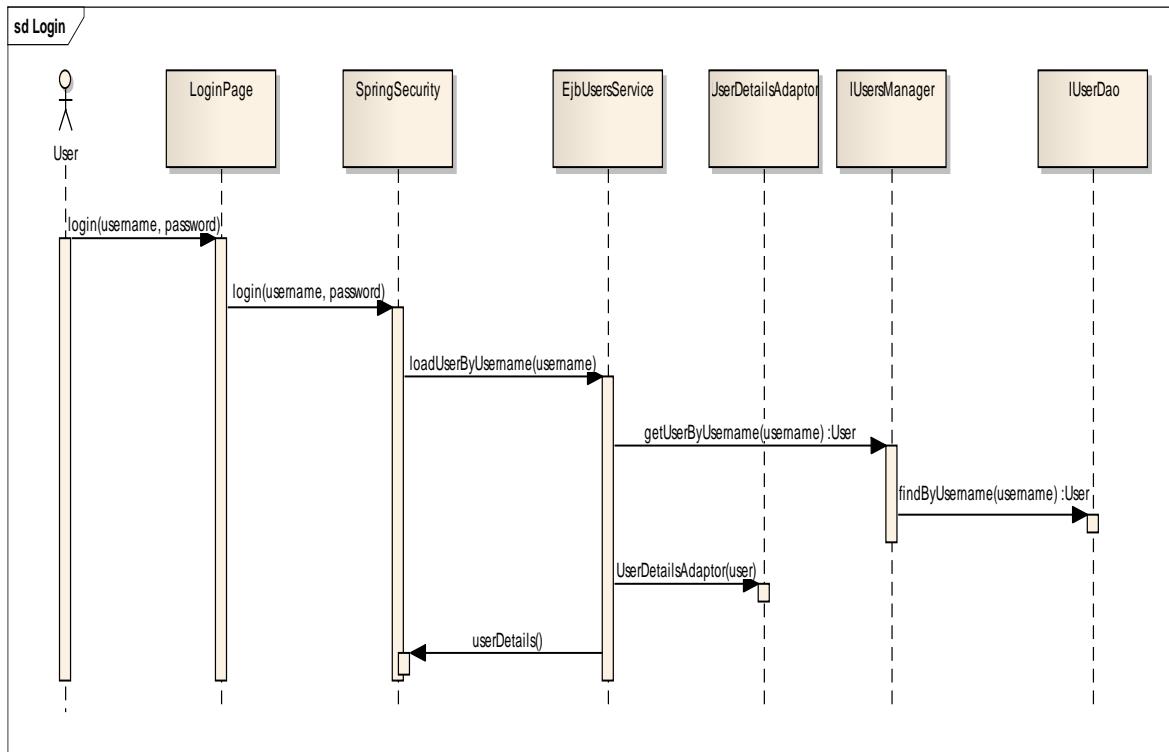


Figura 132: Diagrama de secuencia – Autenticación

- El usuario introduce los datos en un formulario web, que los envía al framework Spring Security.
- El framework ejecuta su mecanismo de autenticación que finalmente usará la clase *EjbUserService* para intentar obtener un principal a partir de los datos aportados.
- La clase *EjbUserService* invoca a la capa de negocio de forma remota, con el fin de obtener un usuario para los datos aportados.
- La capa de negocio consulta la capa de acceso a datos que devolverá un usuario desde el sistema de bases de datos o null si no hay.
- Finalmente la clase *EjbUserService* adapta el usuario obtenido a la interfaz *UserDetails*, usada por el framework para realizar, con la información introducida por el usuario, la obtención de negocio, la autenticación.

6.3.6 Historial

La aplicación web, utiliza el soporte de GWT para el historial del navegador para relacionar tokens de navegador con secciones de la aplicación. Para ello tienen lugar una serie de flujos de acciones que se detallan mediante los siguientes diagramas de secuencia.

6.3.6.1 Evento GWT para Cambiar el historial

El siguiente diagrama de secuencia muestra los pasos mediante los cuales un presentador envía un evento mediante el bus de eventos de GWT cuando la interfaz de usuario que presenta, cambia de sección. En este ejemplo el usuario ha entrado en la sección de notificaciones.

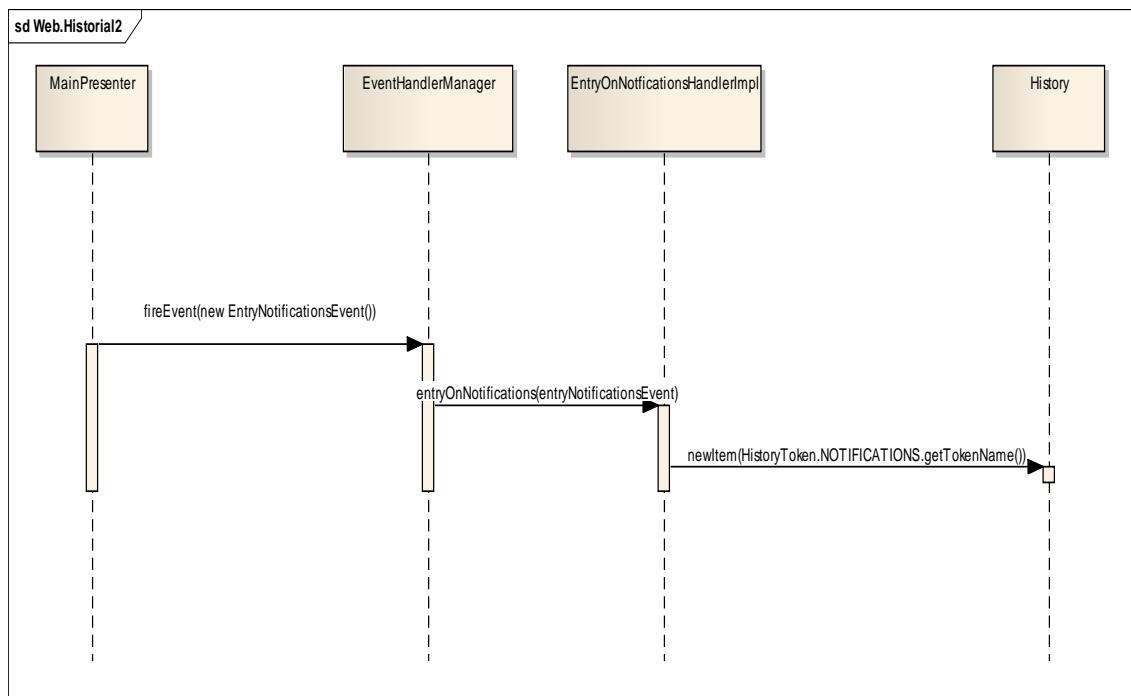


Figura 133: Diagrama de secuencia – Envío de evento GWT para cambiar el historial

1. El Presentador principal detecta que se ha accedido a la sección de notificaciones y lanza el evento definido para ello.
2. El bus de eventos (*EventHandlerManager*, clase de GWT para gestionar manejadores de muchos tipos de eventos), ejecuta un manejador que se le ha registrado para ese evento.
3. El manejador ejecutado pone en el historial el token adecuado para la sección a la que el usuario ha accedido.

6.3.6.2 Cambio del historial

El siguiente diagrama detalla el proceso que tiene lugar cuando el token del historial del navegador cambia. En esta aplicación los tokens del historial están asociados a secciones de la aplicación, y cambiarán bien porque el usuario cambio de sección (Figura 133), o bien porque utilice los botones del navegador. En cualquiera de los dos casos los pasos son los mostrados en la Figura 134

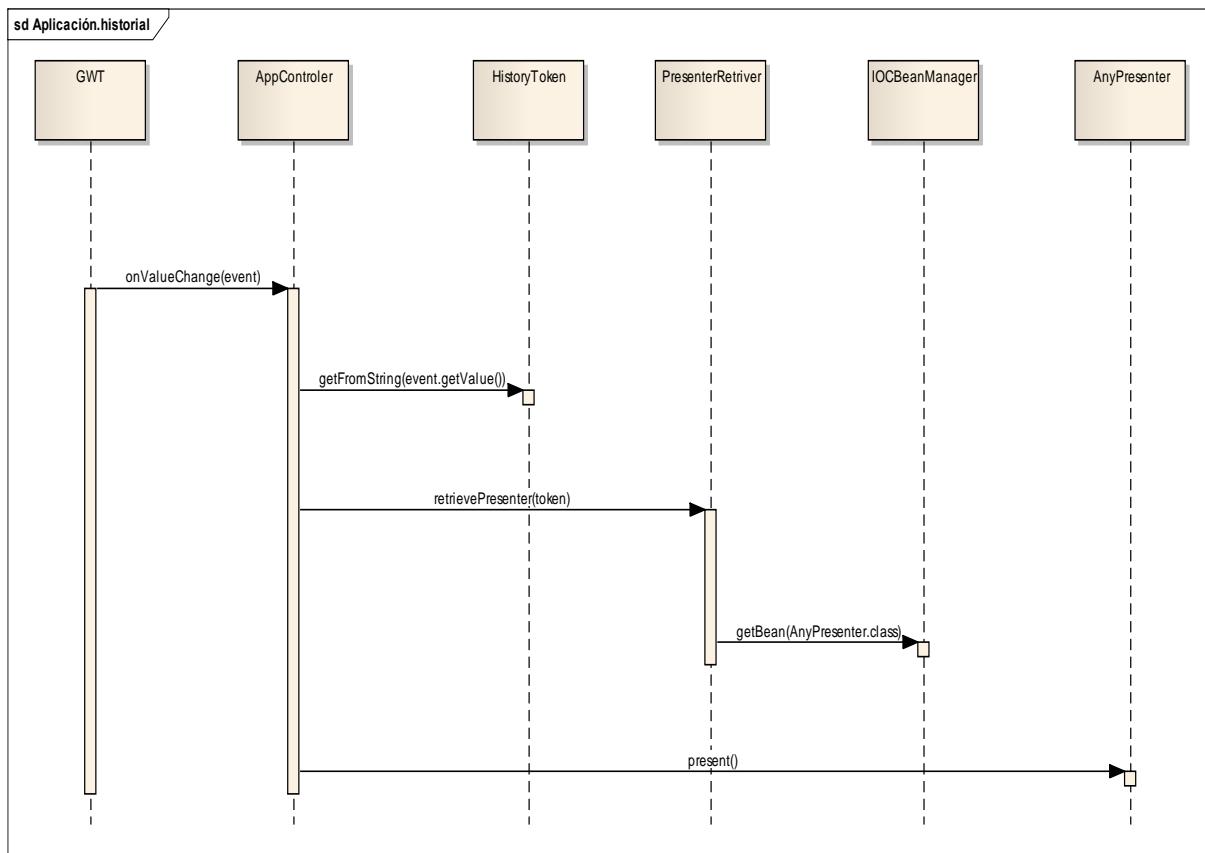


Figura 134: Diagrama de secuencia - Cambio de valor del historial

1. El framework GWT ejecuta el método `onValueChange` de la clase `AppController`, pasándole como parámetro el evento, que contiene la cadena que se ha puesto en el historial.
2. La clase `AppController` obtiene una instancia de la clase `HistoryToken` para la cadena del historial, esta instancia tiene asociado también un objeto class, correspondiente al presentador que se debe ejecutar para esa cadena de historial.
3. Con la instancia de `HistoryToken` el `AppController` recupera la instancia del presentador adecuado, pidiéndosela a la clase `PresenterRetriever`, que se la pide a su vez una instancia del contenedor de Inyección de dependencias.
4. Finalmente el `AppController` ejecuta el presentador que ha obtenido.

6.4 Diagramas de Actividad y de Estado

6.4.1 Añadir Contacto

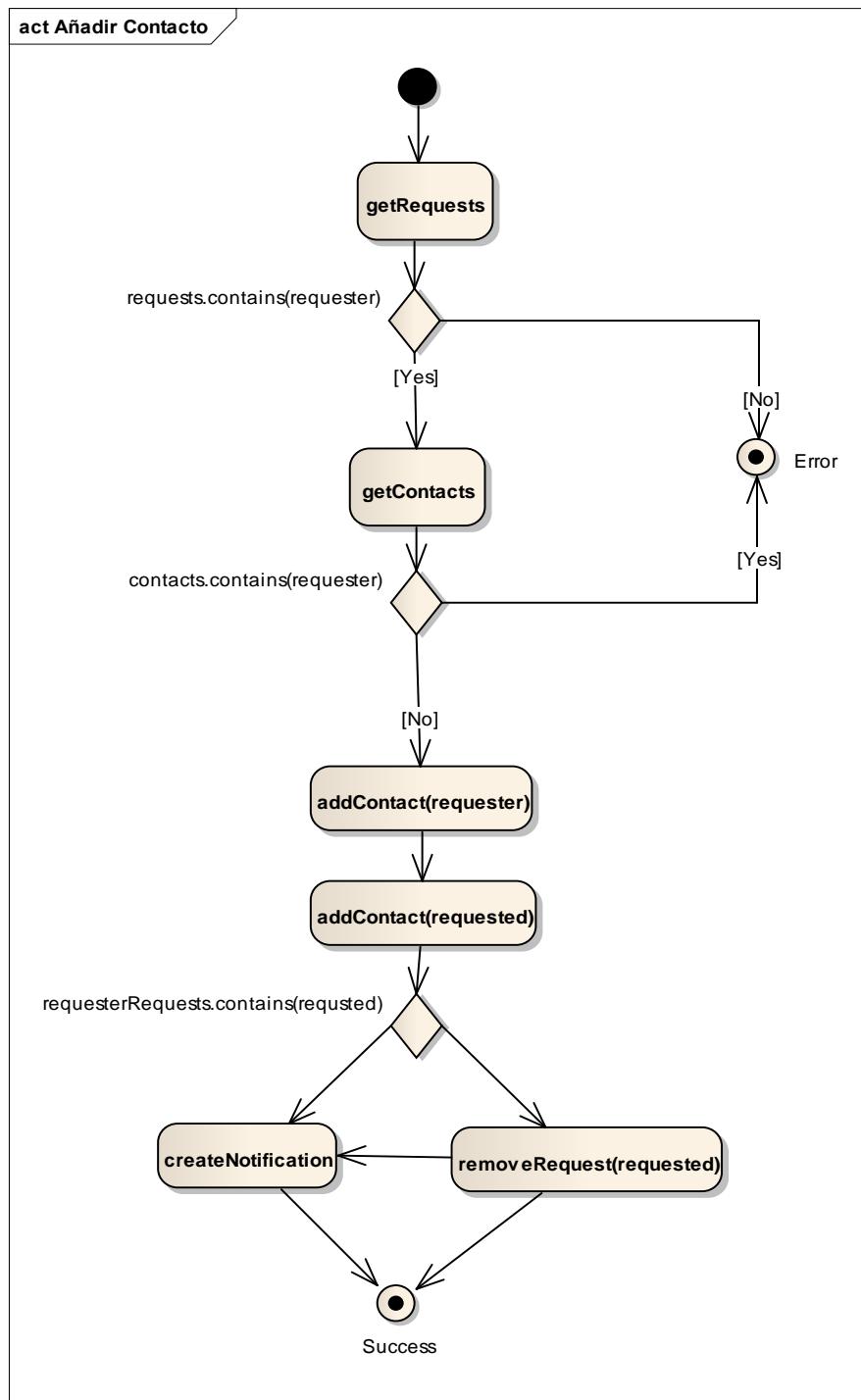


Figura 135: Diagrama de Actividad - Añadir Contacto

Para añadir a un contacto previamente se tiene que haber pasado por el proceso de realizar una petición, al menos por alguna de ambas partes. Los pasos del proceso son los siguientes.

1. Se obtienen las peticiones del contacto solicitado, si no tiene una petición del usuario solicitante se va a un estado de error.
2. Se obtienen los contactos del contacto solicitado, si ya tiene como contacto al usuario solicitante se va a un estado de error.
3. Se añaden ambos usuarios a las listas de contactos de ambos usuarios
4. Se obtienen las peticiones del usuario solicitante, y si tenía una del solicitado, se borra.
5. Se crea una notificación para el contacto añadido.

6.4.2 Crear Recurso

Se detalla a continuación los pasos y casos a tener en cuenta cuando creamos un recurso en la capa de presentación, para preparar toda la información requerida por la capa de negocio. La capa de presentación tiene que lidiar con todas las posibles opciones que tendrá el usuario para finalmente invocar a la capa de negocios con la información adecuada.

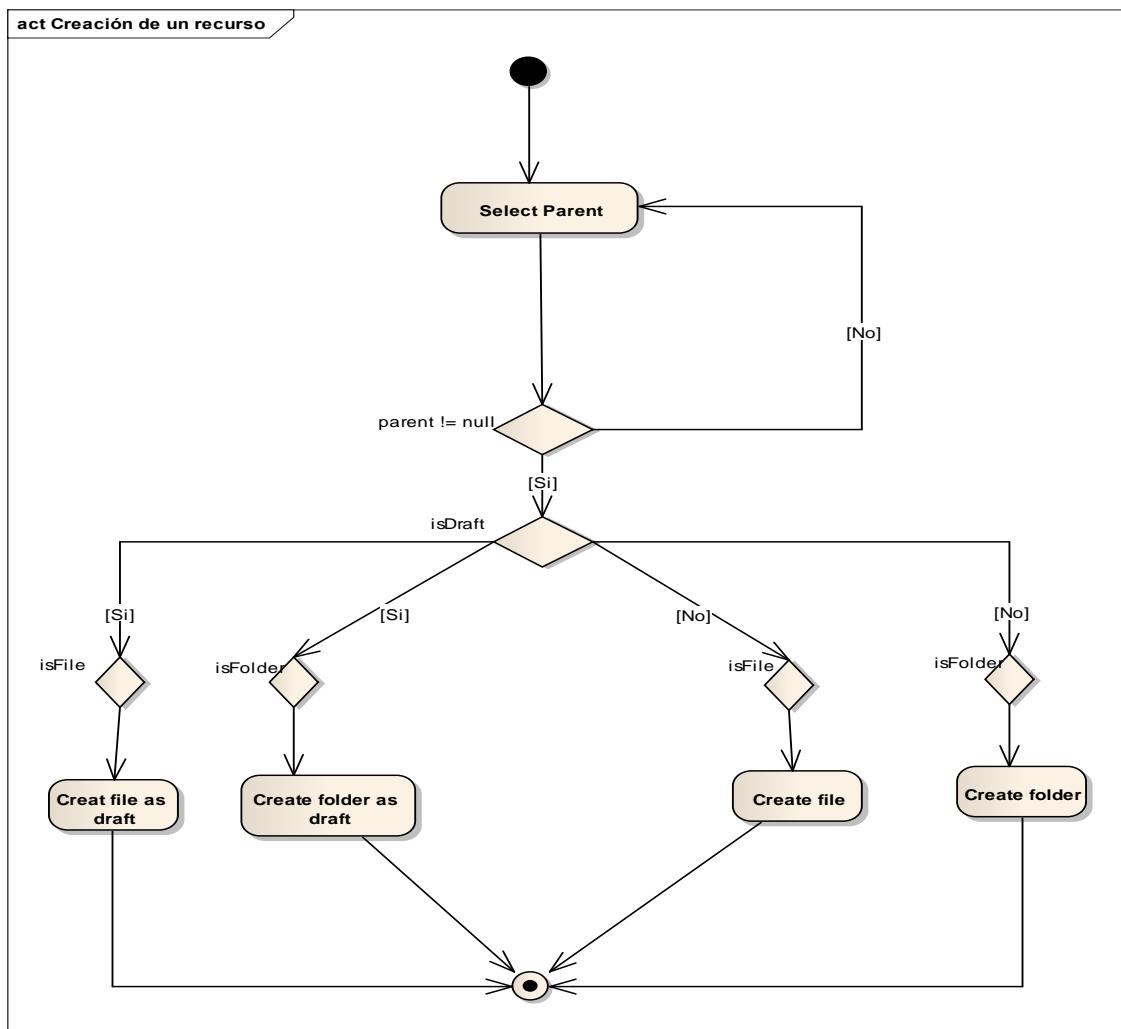


Figura 136: Diagrama de Actividad – Crear Recurso en Presentación

1. Seleccionar padre: Para crear un nuevo recurso, se deberá indicar un parent, aunque este sea la raíz.
2. Crear recurso: Actividad de la creación de un recurso, si se ha seleccionado un parent, se lleva a cabo con éxito, si no se debe seleccionar uno.
3. Se debe comprobar si el recurso es un borrador y que tipo de recurso es, antes de crearlo.

Por otro lado en la capa de negocio tienen lugar otra serie de pasos más abstractos.

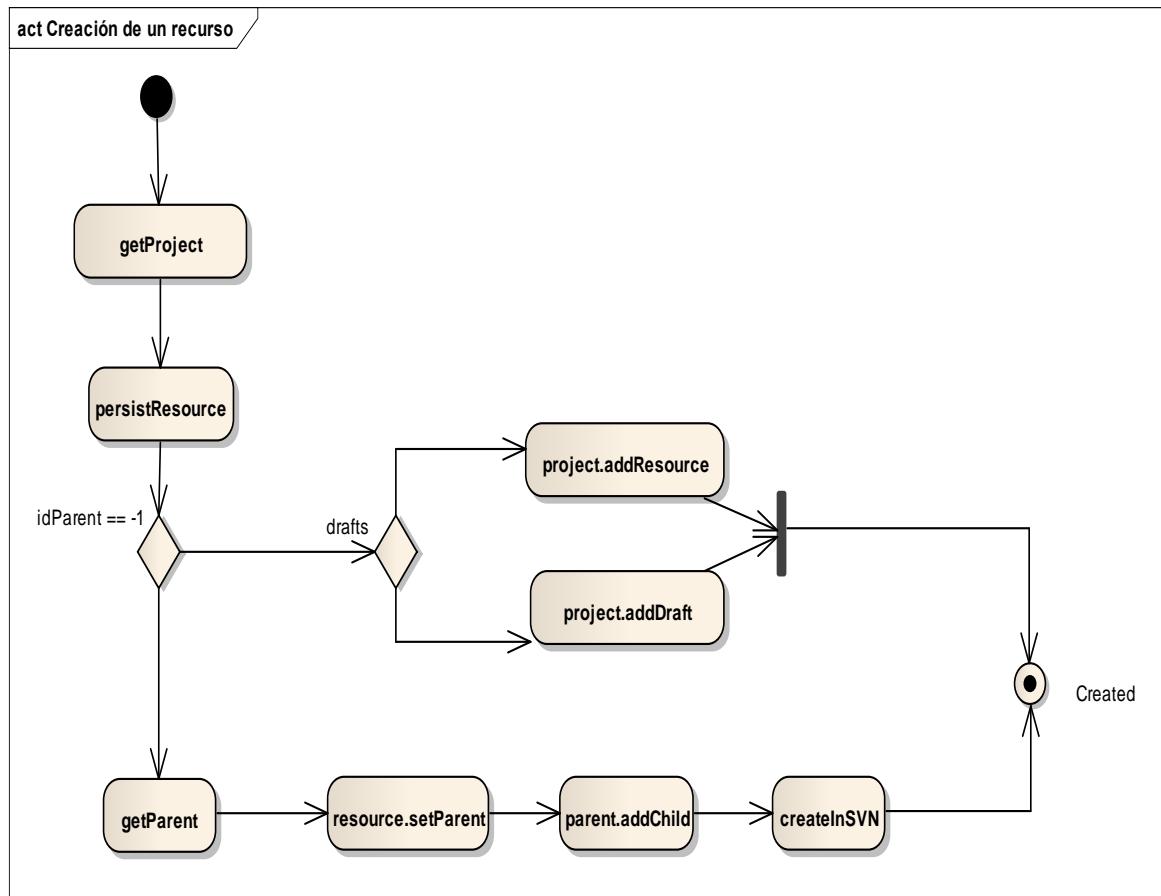


Figura 137: Diagrama de Actividad Crear Recurso en Negocio

1. Obtenemos el proyecto sobre el que creamos el recurso.
2. Creamos el recurso en la base de datos.
3. Si al recurso no se le asignó parent.
 - a. Se añade la raíz del proyecto como borrador o recurso.
4. Si al recurso se le asignó parent.
 - a. Se obtiene el parent del recurso de la base de datos.

- b. Se establece el padre al recurso y se añade el recurso al padre
- c. Se crea el recurso en el repositorio de ficheros.

6.4.3 Ciclo de Vida de un Proyecto

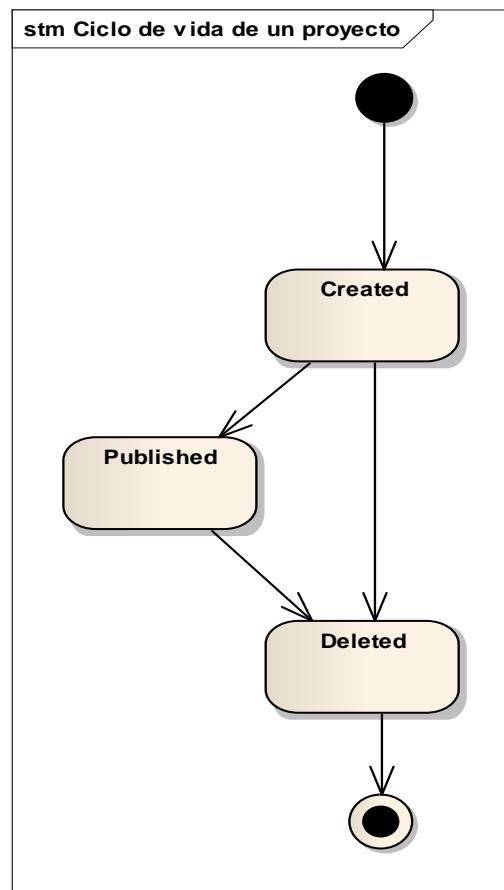


Figura 138: Diagrama de estados - Ciclo de vida de un proyecto

1. Creado: El proyecto acaba de ser creado.
2. Publicado: El proyecto ha sido publicado, esto lo hace accesible a buscadores.
3. Eliminado: El proyecto ha sido eliminado así como todos sus recursos.

6.4.4 Ciclo de Vida de una Tarea

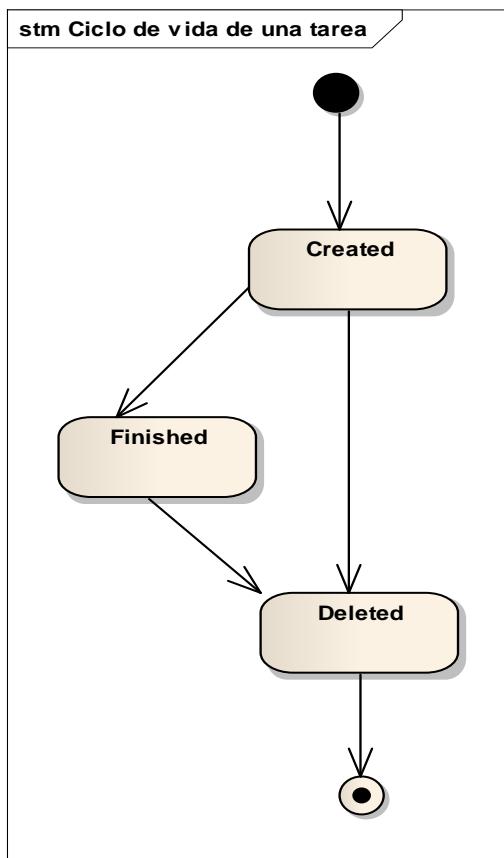


Figura 139: Diagrama de estados – Ciclo de vida de una tarea

1. Creada: La tarea acaba de ser creada.
2. Finalizada: La tarea ha sido marcada como completada.
3. Eliminada: La tarea ha sido eliminada, esto puede hacerse tanto si está creada como si esta completada.

6.4.5 Ciclo de Vida de una Discusión

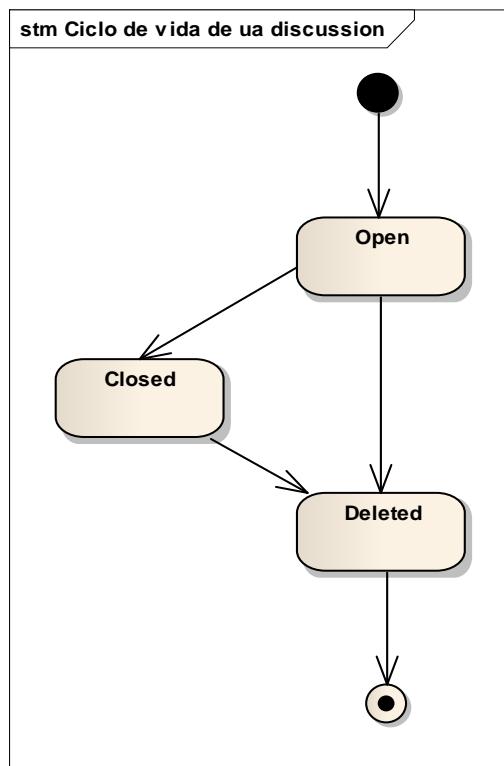


Figura 140: Diagrama de estado – Ciclo de vida de una discusión

1. Abierta: Cuando se crea una discusión, esta se encuentra abierta, esto significa que se puede discutir.
2. Cerrada: El gestor de un proyecto puede cerrar las reuniones para que todos puedan ver el resultado y que no se pueda votar más, una discusión cerrada será eliminada del listado.
3. Eliminada: Las discusiones se podrán eliminar del listado de discusiones, tanto si están cerradas como si están abiertas.

6.4.6 Ciclo de Vida de una Reunión

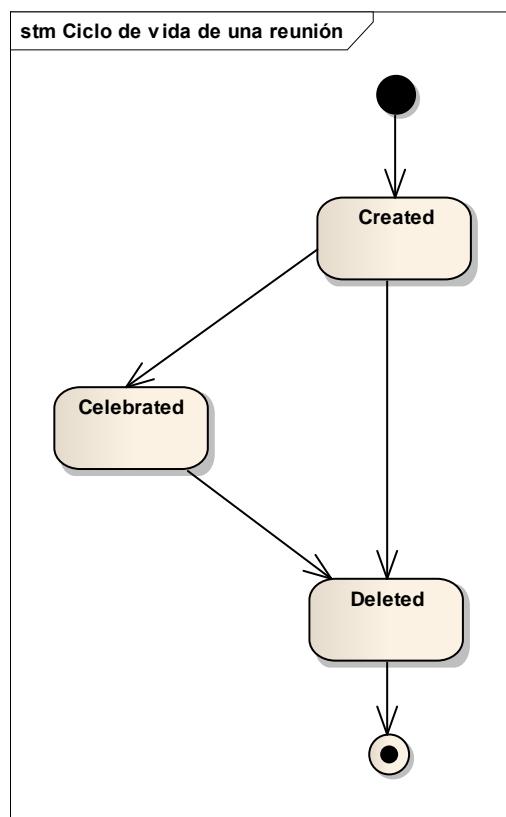


Figura 141: Diagrama de estados – ciclo de vida de una reunión

1. Creada: La reunión acaba de ser creada.
2. Celebrada: La reunión ha sido clausurada.
3. Eliminada: La reunión ha sido eliminada, se pueden eliminar una reunión creada y una reunión celebrada.

6.5 Diseño de la Interfaz

En este apartado revisamos todas las secciones de la interfaz de usuario con detalle, partiendo del trabajo realizado en el apartado 5.4 “Análisis de Interfaces de Usuario” donde se especificó la interfaz en términos abstractos.

El apartado 6.2.4.2 “Diagramas de Clases de cnpd-web”, donde se detalla el diseño de las clases presentadoras de cada vista, también está relacionado con este, ya que la arquitectura de los presentadores determina la estructura de secciones de la aplicación.

Cabe destacar que el diseño de las interfaces de usuario se ha hecho mediante el desarrollo de prototipos, que permitiesen valorar los resultados de aplicar y agrupar las opciones ofrecidas por la tecnología.

6.5.1 Sección Principal

La sección principal de la aplicación está compuesta por una cabecera, un pie, una zona central y dos paneles laterales. La zona central contiene las diferentes secciones de la aplicación, el panel izquierdo contiene la información del usuario, el panel derecho contiene el chat de la aplicación y el pie y la cabecera son netamente decorativos.

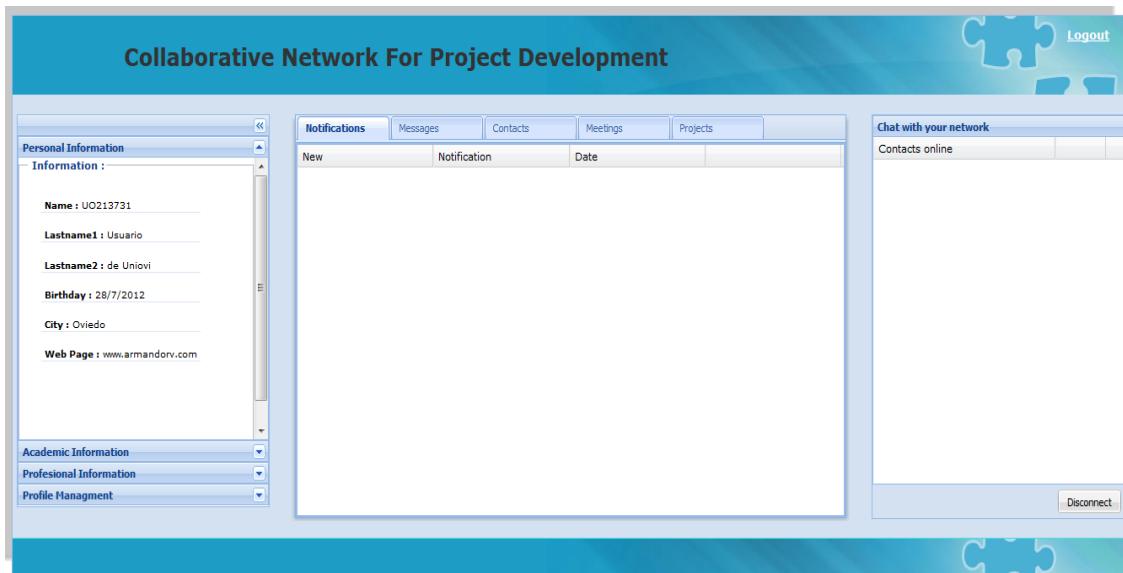


Figura 142: Diseño de Sección Principal

6.5.2 Sección de Información

El panel de la izquierda contiene la información del usuario dividida por tipos, información personal, información académica, información profesional y finalmente un árbol de opciones para modificar esa información.



Figura 143: Diseño de Sección de Información

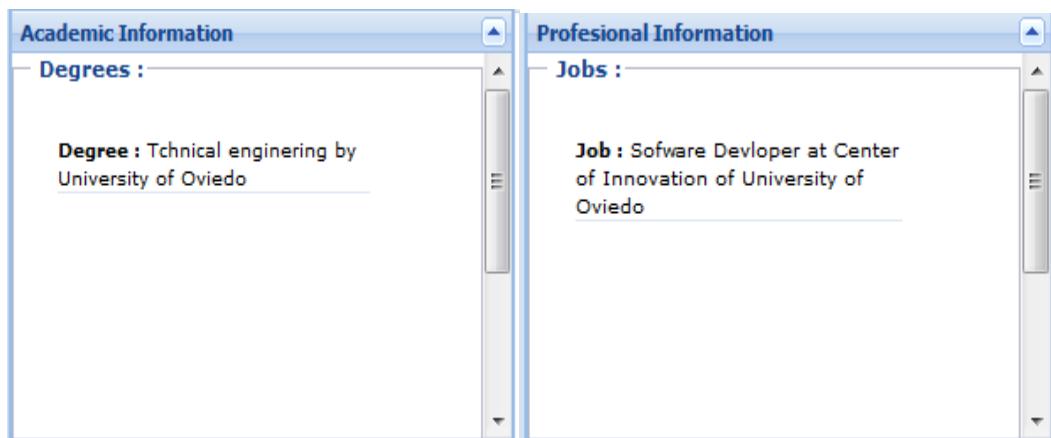


Figura 144: Diseño de Sección de Información – Estudios y Trabajos

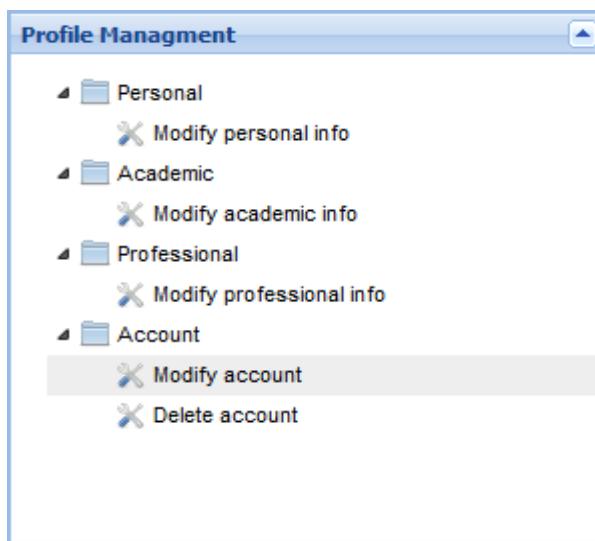


Figura 145: Diseño de la sección de Información – Gestión del perfil

La sección de modificación del perfil, muestra un árbol de opciones sobre las que el usuario podrá hacer doble click. Según la opción se muestran diálogos distintos para la modificación de la opción seleccionada.

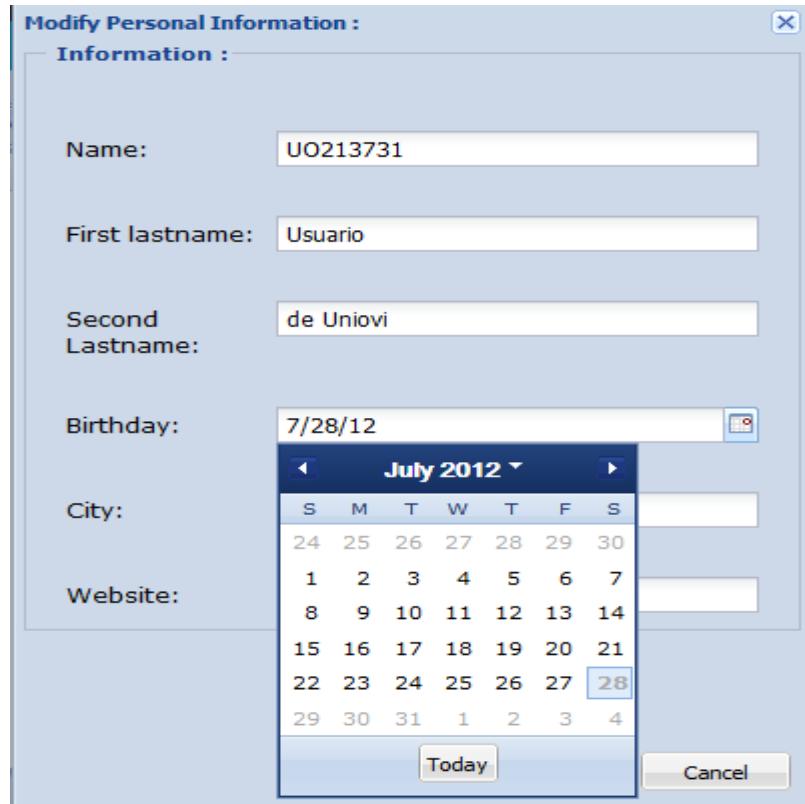


Figura 146: Diseño de Sección de Información – Modificación de Datos

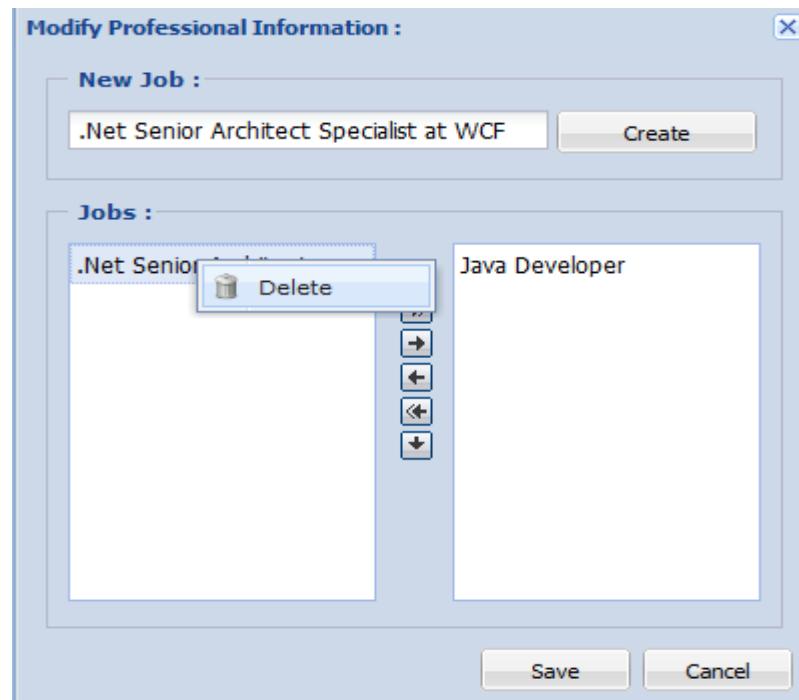


Figura 147: Diseño de Sección de Información – Modificación de Trabajos

6.5.3 Sección de Chat

El panel derecho de la sección principal contiene el chat de la aplicación, aquí estarán los contactos conectados, y se irán añadiendo de forma automática los contactos que se conecten. Esta lista permite filtrar los contactos por nombre y seleccionarlos para hablar.

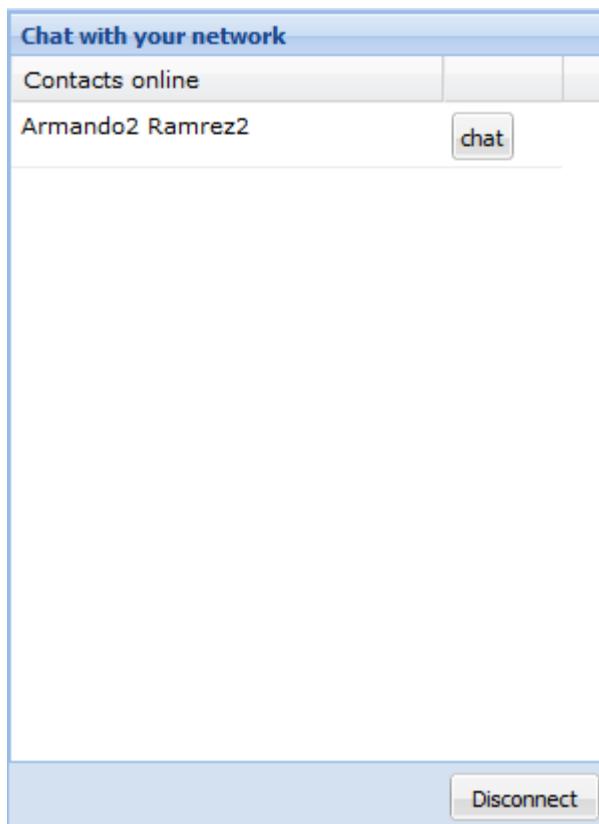


Figura 148: Diseño de Sección de Chat

Seleccionando la opción de hablar con un contacto, se nos abrirá la ventana de conversaciones, esta ventana tendrá una pestaña por cada conversación abierta, se podrán cerrar y añadir seleccionando más contactos en la lista.

Cuando un contacto entabla una conversación con nosotros, se nos abrirá esta ventana si no lo está ya. En caso de que la ventana este ya abierta se añadirá una nueva pestaña a ella con la conversación entrante.

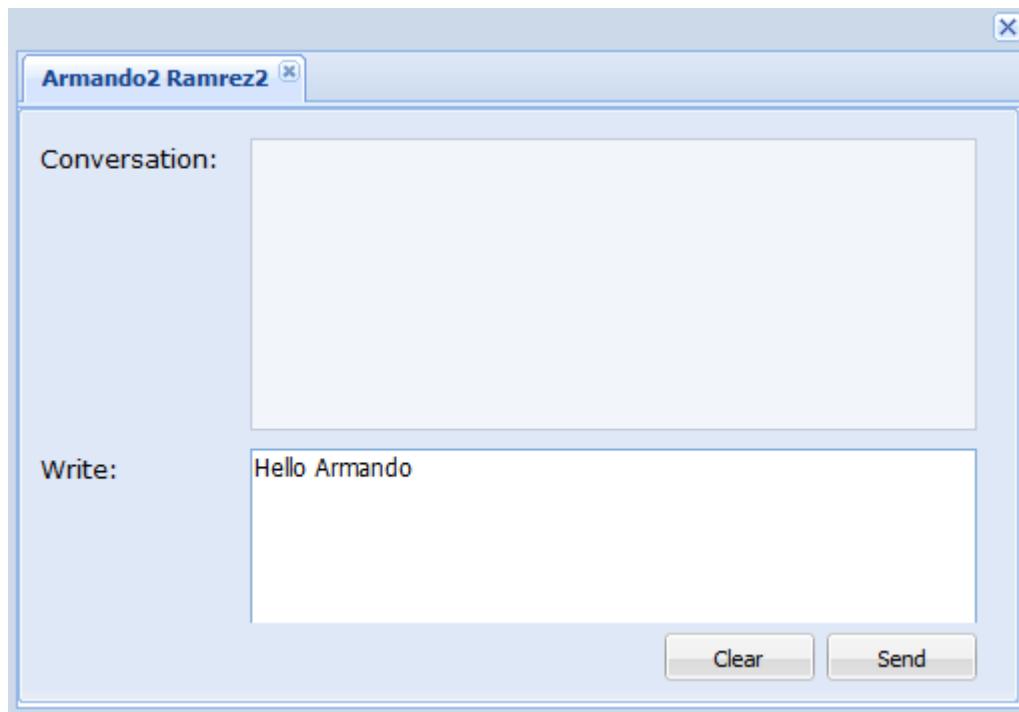


Figura 149: Diseño de Sección de Conversaciones

6.5.4 Sección de Notificaciones

Para la sección de notificaciones se ha creado una tabla que muestra las notificaciones que tiene el usuario. Esta tabla muestra información genérica de las notificaciones, como fecha o el tipo de notificación, además marca con un ícono las notificaciones nuevas.

New	Notification	Date	
	New Contact	8/19/12 1:09 PM	
	Contact Request	8/19/12 1:53 PM	
	Contact Request	8/18/12 11:15 PM	
	Project Invitation	7/28/12 4:25 PM	
	Contact Request	8/19/12 10:28 AM	
	Contact Request	8/18/12 2:50 PM	
	Contact Request	7/28/12 7:23 PM	
	Contact Request	8/18/12 2:50 PM	
	Project Invitation	7/28/12 4:25 PM	
	Contact Request	7/28/12 7:24 PM	

Figura 150: Diseño de Sección de Notificaciones

El usuario podrá filtrar las notificaciones y reordenar las columnas de la tabla, la tabla cuenta con una columna con acciones, en este caso solamente ver los detalles de la notificación.

La captura de pantalla muestra una tabla de notificaciones. La columna 'Date' tiene un menú suspenso abierto que incluye 'Sort Ascending', 'Sort Descending', 'Columns' y 'Filters'. El menú 'Filters' está resaltado y muestra una opción 'Contact' que también está resaltada.

New	Notification	Date
	New Contact	
	Contact Request	
	Contact Request	
	Contact Request	8/19/12 10:28 AM
	Contact Request	8/18/12 2:50 PM

Figura 151: Diseño de Sección de Notificaciones – Filtros

En la siguiente captura de uno de los prototipos, se ha movido la columna de la izquierda a la derecha y se está moviendo la de la derecha hacia el centro.

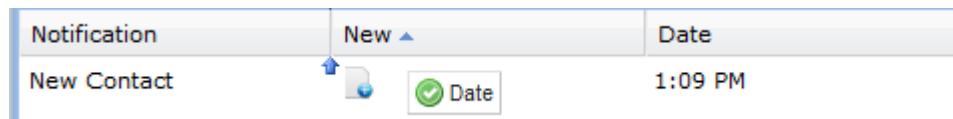


Figura 152: Diseño de Sección de Notificaciones - Movimientos

La opción de ver los detalles de la notificación mostrará un dialogo como el siguiente:

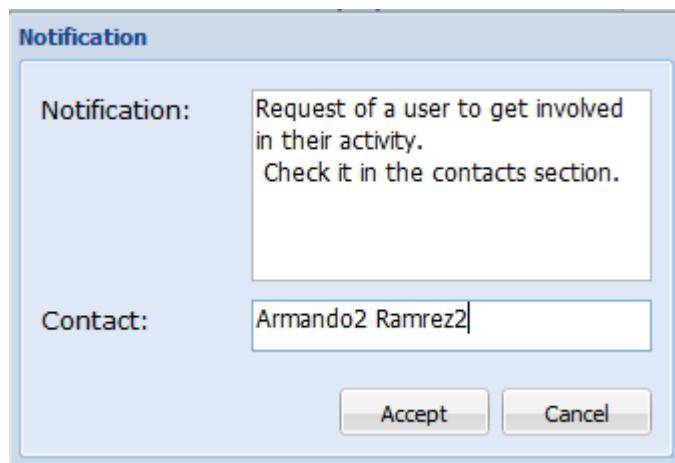


Figura 153: Diseño de Sección de Notificaciones – Detalles

El diálogo mostrará distinto contenido para los distintos tipos de notificaciones. Al aceptar la notificación dejará de ser nueva.

6.5.5 Sección de Mensajes

La segunda pestaña mostrará una tabla con los mensajes que le han enviado al usuario.

Read	Date	Sender	Actions
	8/25/12 1:38 PM	armmandoo2@gmail.com	Select

Figura 154: Diseño de Sección de Mensajes

Sobre este listado el usuario podrá aplicar filtros y seleccionar opciones, la opción de ver un mensaje muestra un dialogo como el siguiente, que permite además de verlo responderlo si se desea.

Para la redacción de los mensajes se ofrece un editor HTML que permite dar formato a los mensajes.

The screenshot shows a messaging application window titled "Messaging on your network". At the top, it displays a message from "armmandoo2@gmail.com wrote :". The message content is "Hi , what about your PFC, is it finished ?". Below this, there is a "Write your response :" section with a rich text editor toolbar and a text input field. At the bottom right of the window are buttons for "Close", "Send", and "Mail :".

Figura 155: Diseño de Sección de Mensajes - Responder mensaje

6.5.6 Sección de Contactos

La sección de contactos cuenta con tres subsecciones, una sección con el listado de contactos, otra sección con el listado de peticiones de contactos y finalmente una sección de búsqueda.

Notifications	Messages	Contacts	Meetings	Projects
Contacts list	Contact requests	Find users		

Figura 156: Diseño de la sección de Contactos

La secciones de listado de contactos y listado de peticiones mostrarán una tabla con información a de los contactos, así como un menú de opciones, que obviamente será distinto en ambos casos.

Name	Lastname1	Lastname2	Actions
Armando2	Ramrez2	Vila2	Select ▾
Armando	Ramírez	Vila	visit delete message

Figura 157: Diseño Sección de Contactos – Listado de Contactos

La sección de búsqueda de contacto mostrará un listado con los contactos que contengan alguno de los parámetros introducidos.

Name:	Armando2								
Lastname:									
Lastname 2:									
Search									
<table border="1"> <thead> <tr> <th>Name</th> <th>Lastname1</th> <th>Lastname2</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>Armando2</td> <td>Ramrez2</td> <td>Vila2</td> <td>Select ▾</td> </tr> </tbody> </table>		Name	Lastname1	Lastname2	Actions	Armando2	Ramrez2	Vila2	Select ▾
Name	Lastname1	Lastname2	Actions						
Armando2	Ramrez2	Vila2	Select ▾						
visit request									

Figura 158: Diseño Sección de Contactos – Búsqueda de Contactos

6.5.7 Sección de Reuniones

La sección de reuniones cuenta al igual que la sección de contactos con varias subsecciones, una sección con el listado de reuniones del usuario, otra sección con el listado de invitaciones a reunión y una sección para crear reuniones.

Notifications	Messages	Contacts	Meetings	Projects
Meetings list	Meeting invitations	New Meeting		

Figura 159: Diseño de Sección de Reuniones

El listado de reuniones muestra una tabla con información genérica de la reunión, así como opciones para acceder a ellas o dejarlas.

Celebrated	Title	Date	Actions
	Created Meeting	August 19, 2012	Select
	DASD	August 19, 2012	Details Leave

Figura 160: Diseño Sección de Reuniones – Listado de Reuniones

6.5.8 Sección de Proyectos

La sección de proyectos agrupa en su interior cuatro subsecciones, una sección que muestra en una tabla todos los proyectos de los que le usuario es miembro, otra que muestra una tabla con todas las invitaciones a proyectos, una sección para buscar proyectos en el sistema y una sección para crear proyectos.

Notifications	Messages	Contacts	Meetings	Projects
Projects	Invitations	Search for Projects	New Projects	

Figura 161: Diseño de Sección de Proyectos

La sección del listado de proyectos mostrará una tabla con información genérica de los proyectos y un menú de opciones para cada uno.

Title	Knowledge Area	Actions
Web tier created	Sofware Development.	Actions
Good life	Sofware Development.	Open Leave
Project	Sofware Development.	
Simple Proiect	Sofware Development.	Actions

Figura 162: Diseño de Sección de Proyectos – Listado de Proyectos

La sección de invitaciones mostrará exactamente lo mismo para los proyectos a los que hayan invitado al usuario, cambiando por supuesto las opciones, en este caso serán aceptar y rechazar.

La sección de búsqueda mostrará un listado similar para los resultados de la búsqueda, cuyos parámetros serán elegidos en esta sección.

The screenshot shows a search interface for projects. It has two input fields: 'Title:' containing 'Web' and 'Knowledge area:' containing 'Software Development.'. To the right of these fields is a dropdown arrow. In the bottom right corner of the form is a blue 'Search' button.

Figura 163: Diseño de Sección de Proyectos – Búsqueda de Proyectos

Finalmente la subsección de creación de proyectos ofrecerá un sencillo formulario para la creación de nuevo proyecto.

The screenshot shows a creation form for a new project. It includes three input fields: 'Knowledge area:' with 'Software Development.', 'Title:' with 'Coll', and 'Description:' which is currently empty. In the bottom right corner is a blue 'Create' button.

Figura 164: Diseño Sección de Proyectos – Creación de Proyecto

6.5.8.1 Sección de Proyecto

La sección de proyecto muestra la información de un proyecto y define las secciones que agrupan las distintas áreas funcionales del proyecto, para ello se ha usado un panel de pestañas, una pestaña por cada sección, que a su vez contienen subsecciones. Además esta sección cuenta con dos paneles laterales, uno con el chat del proyecto (panel derecho) y otro para la administración del proyecto.

La siguiente captura muestra la sección inicial donde se muestra la información relativa a un proyecto, así como las pestañas de todas las subsecciones con las cuenta el proyecto.

The screenshot shows a user interface for managing a project. At the top, there is a navigation bar with tabs: Project, Resources, References, Discussions, Milestones, and Tasks. The 'Project' tab is selected.

Info :

- Title : Good life
- Knowledge Area : Software Development.
- Description : sdsdsds
- Manager : UO213731
- Published : false

Members :

- Member : UO213731

Figura 165: Diseño Sección de Proyecto

6.5.8.2 Gestión del Proyecto

EL panel de gestión del proyecto alberga un árbol con todas las opciones de la administración del proyecto. Las opciones del árbol se ejecutan haciendo doble click sobre ellas y dependiendo de la opción la respuesta será distinta.

Las opciones más complejas en términos de la interfaz de usuario son las relativas a los miembros, ya que implican tratar con un listado de miembros, la siguiente captura de uno de los prototipos muestra el árbol de opciones y el resultado de seleccionar la opción de invitar contactos.

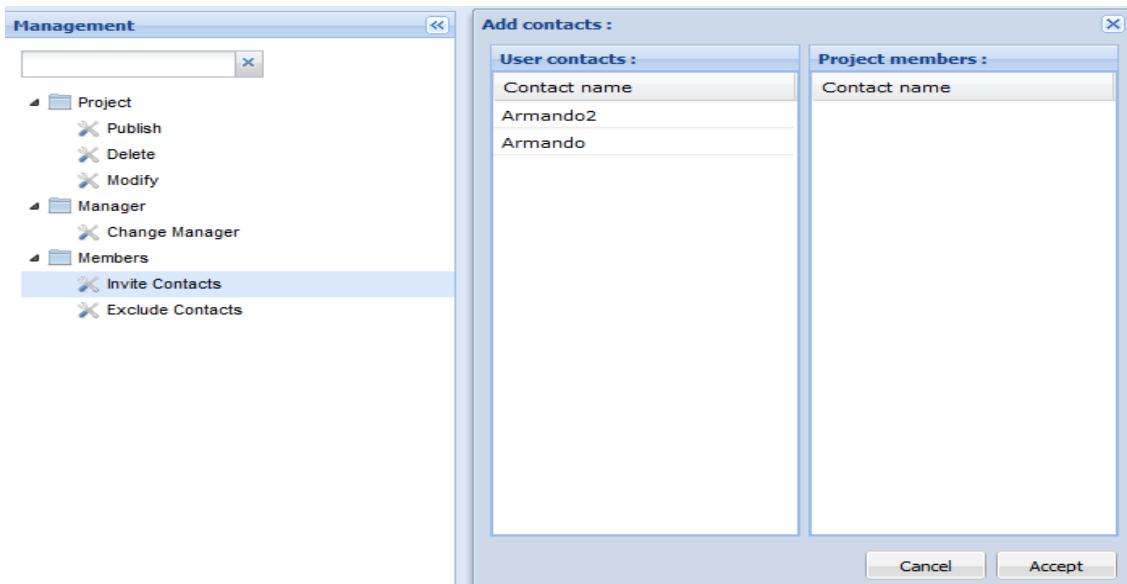


Figura 166: Diseño Sección de Proyecto - Administración

6.5.8.3 Sección de Recursos

La sección de recursos está compuesta por dos árboles, un árbol con todos los recursos del proyecto, y otro árbol con todos los borradores del proyecto.

Los árboles permiten arrastrar y soltar recursos tanto dentro del propio árbol, como de un árbol a otro.

Además cuentan con menus contextuales que ofrecen opciones sobre los recursos, como editar, descargar o borrar. Las opciones cambian según el tipo de recurso.

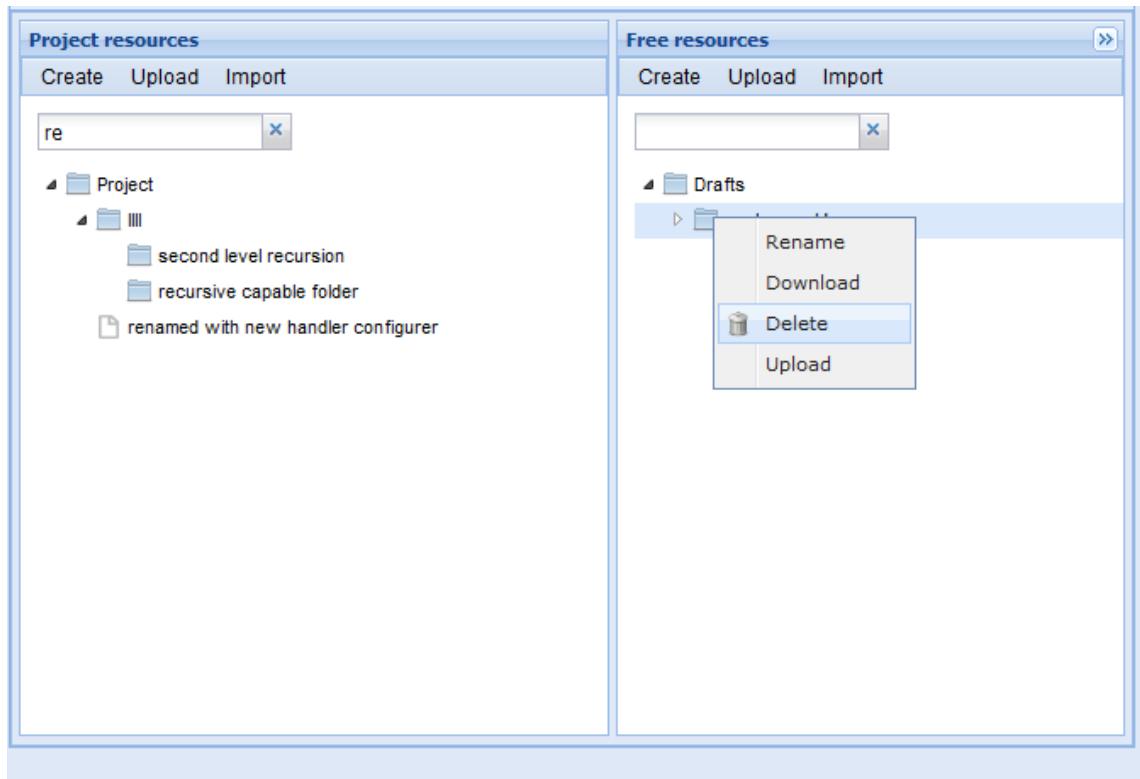


Figura 167: Diseño de Sección de Recursos

6.5.8.4 Sección de Referencias

La sección de referencias tendrá dos subsecciones, una donde se listarán las referencias y otra para añadir nuevas referencias al proyecto.

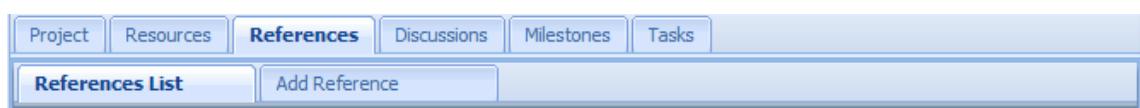


Figura 168: Diseño Sección de Referencias

La sección del listado mostrará el nombre, la URL y un menu con opciones.

Name	URL	Actions
Testing	http://localhost:8080	Select
fdfdf	dfdffd	Open
asdasdas	dasdasdasd	Remove

Figura 169: Diseño Sección de Referencias - Listado

Las nuevas referencias podrán ser creadas desde cero, o añadidas mediante un buscador de referencias del sistema.

The screenshot shows two main sections: 'Create Reference' and 'Find References'. In the 'Create Reference' section, there are fields for 'Name' and 'URL' with a 'Create' button. In the 'Find References' section, there is a 'Key words' input field containing 'Test' and a 'Search' button. Below these is a table listing references, showing one entry: 'Testing' with URL '<http://localhost:8080>'. Action buttons for 'Select', 'Open', and 'Add' are available for this entry.

Figura 170: Diseño de Sección de Referencias – Añadir Referencias

6.5.8.5 Sección de Discusiones

La sección de discusiones cuenta con dos sub secciones, una para listar las discusiones, y otra para crear nuevas discusiones.

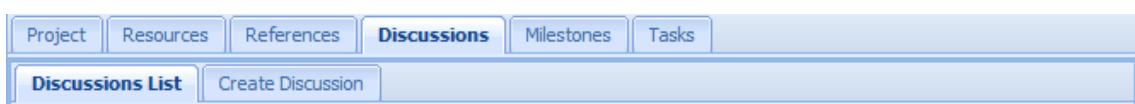


Figura 171: Diseño Sección de Discusiones

La sección del listado de discusiones muestra información genérica de las discusiones y ofrece un menú desplegable de opciones, como ver detalles, votar o ver resultados.

Open	Title	Actions
	Tecnology	<input type="button" value="Select ▾"/>
	dfdf	<input type="button" value="Select ▾"/>

Figura 172: Diseño Sección de Discusiones – Listado

La subsección de creación de discusiones consta de todos los campos necesarios para crear la discusión, entre los que destaca el campo para añadir las opciones, que permite añadir y quitar opciones.

Para cada una de las opciones se abrirá un dialogo mostrando lo necesario para la opción seleccionada.

Details

Title:

Description:

Options

Add options:

Options:

Figura 173: Diseño Sección de Discusiones – Nueva Discusión

6.5.8.6 Sección de Hitos

La sección de hitos no está compuesta por otras secciones, si no que posee una única sección que se compone de un calendario y dos campos para los estados del proyecto.

Los hitos pueden ser movidos sobre el calendario, cambiando así su fecha, además se pueden eliminar con el botón suprimir y se pueden crear nuevos hitos haciendo doble clic en las fechas vacías. Todos los hitos estarán accesibles desde el combo box para ser establecidos como último hito.

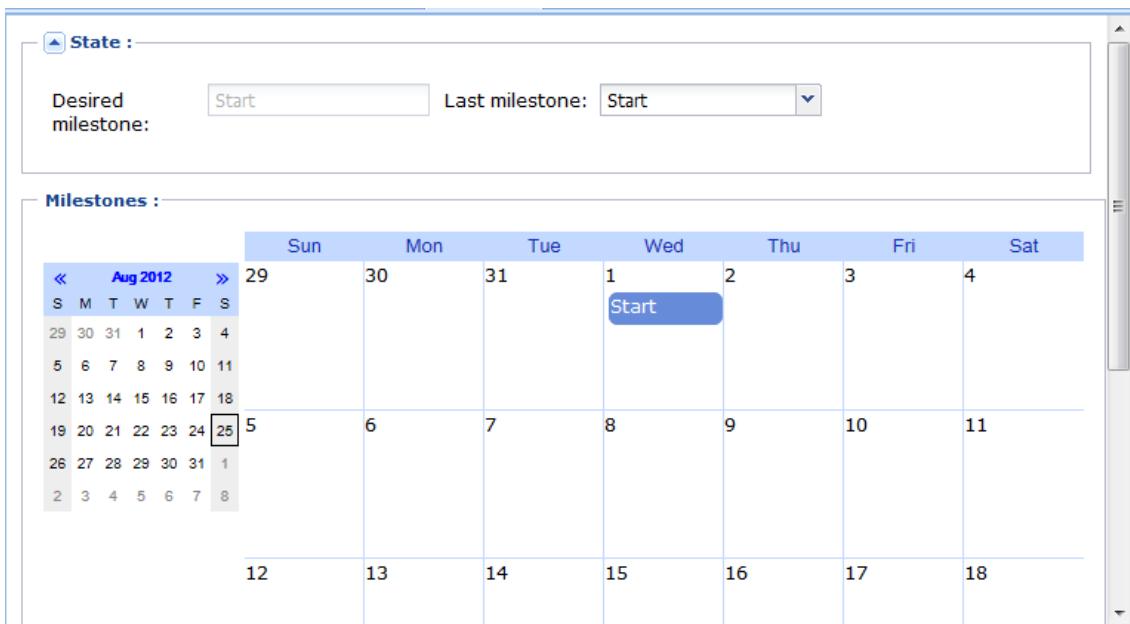


Figura 174: Diseño Sección de Hitos

6.5.8.7 Sección de Tareas

Finalmente tendremos la sección de tareas donde los usuarios podrán crear y consultar las tareas definidas para llevar a cabo el proyecto.

La sección de tareas cuenta con tres subsecciones, una para listar las tareas, otra para crear nuevas tareas y una última sección que muestra un resumen del porcentaje completado de todas las tareas.

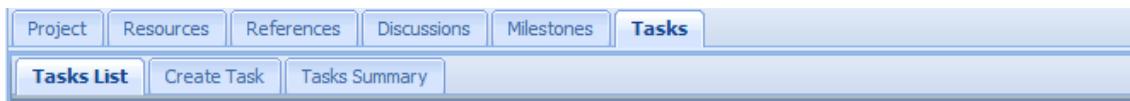


Figura 175: Diseño Sección de Tareas

El listado de tareas muestra la información de las tareas en forma tabular y un menu desplegable con opciones para cada tarea.

Compl...	Name	Beguining	Total Hours	Worked Hours	Actions
	task C	12:00 AM	33	5	Actions ▾
	More and more	12:00 AM	3	0	Actions ▾
	ne3 w	12:00 AM	3	0	Actions ▾

Figura 176: Diseño Sección de Tareas – Listado de Tareas

La subsección de creación muestra un simple formulario donde introducir la información de una nueva tarea.

Name:

Beguining:

Size:

Figura 177: Diseño Sección de Tareas – Nueva Tarea

El resumen de tareas muestra un gráfico con los porcentajes completados de cada tarea, la siguiente captura muestra un ejemplo de este gráfico.

El gráfico se redimensiona al tiempo que lo hace la ventana del proyecto.

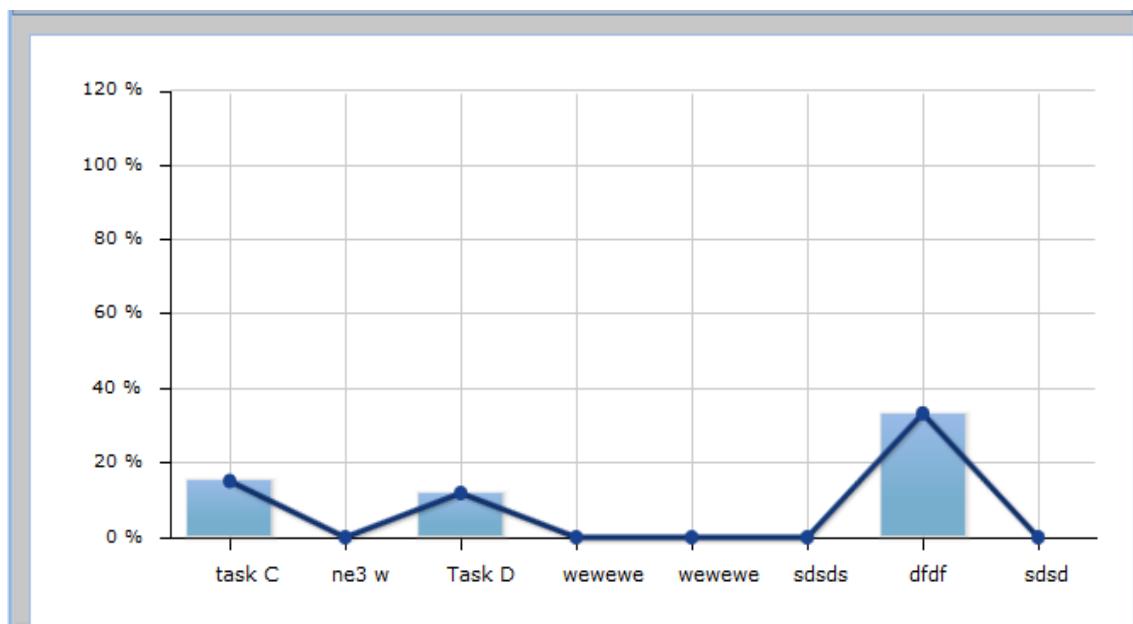


Figura 178: Diseño Sección de Tareas – Resumen de Tareas

6.6 Especificación Técnica del Plan de Pruebas

Este apartado especifica los detalles de las pruebas propuestas en el apartado 5.5 “Especificación del Plan de Pruebas”, especifica los detalles de las pruebas así como el entorno hardware y software para la realización de las pruebas.

Las pruebas han sido realizadas sobre un ordenador personal Intel de 64 bits, con un procesador i5 y 4GB de memoria RAM.

Para el desarrollo de las pruebas se aplica de forma extensa la herramienta JUnit, con el fin de automatizarlas. Además dado que el software se desarrolla para ser ejecutado dentro de un contendor, se ha aplicado una herramienta que integrándose con JUnit nos permite ejecutar dichas pruebas sobre el contendor, *Arquillian*. Esta herramienta crea un artefacto, lo despliega en el contendor y ejecuta todas las pruebas que se le han indicado. Se usa tanto para las pruebas unitarias como para las de integración.

6.6.1 Pruebas Unitarias

- Se realizará una clase de prueba unitaria por cada clase de la capa de persistencia, probando todos los métodos de los Daos, esto prueba también las relaciones establecidas entre las clases del dominio.
- Se realizará una clase de prueba por cada clase de infraestructura de la capa de negocio, tales como *SvnHelper*, *MailHelper* o *BusinessExceptionHandler*.
- Se realizará una clase de pruebas para cada una de las clases que implementan la lógica de negocio del sistema, probando todos sus métodos y demostrando así que cumplen con las reglas de negocio aplicadas.
- Se probarán todos los mapeadores de la capa de presentación, desarrollando una clase de pruebas para cada uno, y comprobando que realizan la traducción con éxito.
- Se probarán las clases de infraestructura de la capa de presentación, tales como *SvnHelper* o *EjbUserDetails*.

6.6.2 Pruebas de Integración y del Sistema

En este apartado se especifican las pruebas definidas en el apartado 5.5 “Especificación del Plan de Pruebas” como pruebas de integración y de sistemas, definiendo para cada aspecto contemplado allí, los casos de prueba pertinentes así como sus entradas y resultados esperados.

Caso de uso 1.1: Añadir Usuario	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Añadir a un usuario no	El sistema posee un usuario más.

existente en la aplicación	
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Añadir a un usuario que ya existe	
Añadir usuario no existente con información incorrecta.	El sistema no posee un usuario más y se genera un error indicando que el usuario ya existe en la aplicación.
Caso de Prueba: CP1.1.3	
Entrada	Resultado Esperado
Añadir usuario no existente con información incorrecta.	El sistema no posee un usuario más y se genera un error indicando que el usuario ya existe en la aplicación.

Caso de Uso 1.2: Modificar Usuario	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Modificar a un usuario ya existente en la aplicación con datos correctos.	
Modificar a un usuario ya existente en la aplicación con datos incorrectos.	El usuario es modificado y no se generan errores.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Modificar a un usuario ya existente en la aplicación con datos incorrectos.	El usuario no es modificado y se genera un error indicando que el usuario ya existe en la aplicación.

Caso de Uso 1.3: Eliminar Usuario	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Eliminar a un usuario del sistema, sin proyectos, y sin contactos.	
Eliminar a un usuario del sistema, sin proyectos, y con contactos.	El sistema posee un usuario menos y no se generan errores.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Eliminar a un usuario del sistema, sin proyectos, y con contactos.	El sistema posee un usuario menos y no se generan errores.
Caso de Prueba: CP1.1.3	
Entrada	Resultado Esperado
Eliminar a un usuario del sistema, sin proyectos, y con contactos.	El sistema posee un usuario menos y no se generan errores.

Caso de Uso 2.1: Buscar Usuario	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Obtener todos los usuarios dado un parámetro.	
Obtener todos los usuarios dado un parámetro.	El sistema devuelve los usuarios que contienen el parámetro entre sus datos personales.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Obtener todos los usuarios	El sistema devuelve los usuarios que contienen los dos

dados dos parámetros.	parámetros entre sus datos personales.
-----------------------	--

Caso de Uso 2.1: Añadir Usuario como Contacto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Añadir a un usuario que no se tiene en la lista de contactos, y al que se le ha hecho una petición.	El sistema añade el usuario seleccionado a la lista de contactos del usuario.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Añadir a un usuario que se tiene en la lista de contactos.	El sistema no debe permitirlo y se debe generar un error.
Caso de Prueba: CP1.1.3	
Entrada	Resultado Esperado
Añadir a un usuario que no se tiene en la lista de contactos, y al que no se le ha hecho una petición.	El sistema no debe permitirlo y se debe generar un error.

Caso de Uso 2.2: Ver perfil de Contacto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Ver contacto que se tiene en la lista de contactos.	El sistema devolverá el contacto, con su lista de contactos y de proyectos publicados.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Ver contacto que no se tiene en la lista de contactos.	El sistema no debe permitirlo y se debe generar un error.

Caso de Uso 2.3: Comunicarse con Contacto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Se envía un mensaje a un usuario que se tiene como contacto.	El usuario destinatario tendrá un nuevo mensaje.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Se envía un mensaje a un usuario que no se tiene como contacto.	El sistema no lo permitirá y generará un error en caso de que esto pase.

Caso de Uso 3.1: Crear Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Crear un nuevo proyecto sin	Habrá un nuevo proyecto en el sistema, solo tendrá un

miembros.	participante, será privado y el gestor será su único participante y creador.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Crear un nuevo proyecto con miembros.	Habrá un nuevo proyecto en el sistema, solo tendrá un participante, será privado y el gestor será su único participante y creador. Además se habrán enviado invitaciones a los contactos seleccionados.

Caso de Uso 3.2.1: Incluir /Excluir Contactos	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Excluir contactos de un proyecto.	El proyecto tendrá automáticamente un contacto menos.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Incluir contactos en un proyecto.	El usuario incluido tendrá una invitación a unirse al proyecto.

Caso de Uso 3.2.2: Cambiar Gestor	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Asignar el rol de gestor a otro participante.	El gestor del proyecto será el participante seleccionado.

Caso de Uso 3.2.3: Publicar Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
El usuario gestor publica el proyecto.	El proyecto es público.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Un usuario no gestor publica el proyecto.	El sistema no lo permite.

Caso de Uso 3.2.1: Eliminar Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
El usuario gestor elimina el proyecto.	El sistema y los miembros tienen un proyecto menos.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Un usuario no gestor elimina el proyecto.	El sistema no lo permite.

Caso de Uso 3.2.1: Planificar Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
El usuario gestor establece el estado del proyecto.	El sistema tiene un nuevo estado.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
El usuario no gestor establece el estado del proyecto.	El sistema no lo permite.

Caso de Uso 3.4.1: Crear Discusión	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Crear una discusión con opciones.	El proyecto tendrá una nueva discusión.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Crear una discusión sin opciones.	El proyecto no tendrá una nueva discusión.

Caso de Uso 3.2.1: Votar en Discusión	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Un usuario que no ha votado añade un voto a una discusión.	La discusión tendrá un nuevo voto.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Un usuario que ha votado añade un voto a una discusión.	El sistema no debe permitirlo.

Caso de Uso 3.5: Abandonar Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Un miembro que no es gestor abandona el proyecto.	El proyecto debe tener un miembro menos.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
El gestor abandona el proyecto.	El proyecto debe tener un miembro menos y un nuevo gestor.

Caso de Uso 4.1: Crear Recurso	
Caso de Prueba: CP1.1.1	

Entrada	Resultado Esperado
Crear recurso en la raíz.	El proyecto tendrá un recurso más.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Crear recurso dentro de otro recurso.	Uno de los recursos del proyecto tendrá un recurso más.

Caso de Uso 4.2: Modificar Recurso	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Modificar un recurso vacío del sistema.	El recurso será modificado.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Modificar un recurso que contiene otros recursos.	El recurso será modificado y sus hijos permanecerán inalterados.

Caso de Uso 4.3: Eliminar Recurso	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Eliminar un recurso vacío del sistema.	El recurso será eliminado.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Eliminar un recurso que contiene otros recursos.	El recurso será eliminado, así como todos sus hijos.

6.6.3 Pruebas de Usabilidad

6.6.3.1 Actividades de las Pruebas de Usabilidad

Las pruebas de usabilidad se realizarán empíricamente con un número de entre 3 y 5 futuros usuarios potenciales. Esto persigue determinar el nivel de usabilidad del sistema y las posibles mejoras. En este apartado se diseñan cuestionarios y se definen tareas que los usuarios deben llevar a cabo para probar la aplicación. Una vez realizadas las pruebas, se cumplimentarán los cuestionarios y se documentarán los problemas y dificultades encontrados en la realización de las tareas.

6.6.3.1.1 Preguntas de Carácter General

Los usuarios del sistema deben cumplir los requisitos no funcionales de usuario, que especifican que los usuarios del sistema estarán familiarizados con el uso del ordenador y de la web. Las siguientes preguntas ayudan a definir mejor a los usuarios dentro de este perfil.

¿Qué tipo de actividades realiza con el ordenador?
<ol style="list-style-type: none"> 1. Es parte de mi trabajo o profesión. 2. Lo uso básicamente para ocio. 3. Solo empleo aplicaciones estilo Office. 4. Únicamente leo el correo y navego ocasionalmente.
¿Ha usado alguna vez una aplicación web con contactos (Redes Sociales)?
<ol style="list-style-type: none"> 1. Sí (indicar nombre del sitio web). 2. No, pero conozco algún sitio web. 3. No, nunca.
¿Ha usado alguna vez una aplicación con chat?
<ol style="list-style-type: none"> 1. Sí (indicar nombre del sitio web). 2. No, pero conozco algún sitio web. 3. No, nunca.
¿Ha usado alguna vez una aplicación para gestionar tareas o recursos?
<ol style="list-style-type: none"> 1. Sí (indicar nombre del sitio web). 2. No, pero conozco algún sitio web. 3. No, nunca
¿Qué busca usted principalmente en una aplicación web (en general)?
<ol style="list-style-type: none"> 1. Que sea fácil de usar. 2. Que sea intuitiva. 3. Que sea rápida. 4. Que tenga todas las funcionalidades relacionadas con la temática del sitio web necesarias. 5. En aquellas en las que solicitan datos personales así como login y password que sea segura y mis datos estén protegidos.

6.6.3.1.2 Actividades Guiadas

Cada usuario deberá realizar las siguientes actividades dentro de la aplicación:

- Registrarse en la aplicación
- Loguearse y acceder a la aplicación.
- Buscar usuarios conocidos
- Enviarles una invitación
- Crear un nuevo proyecto
- Invitar un contacto al proyecto
- Crear un recurso
- Eliminar un recurso
- Modificar un recurso
- Crear una referencia
- Buscar una referencia
- Eliminar una referencia
- Crear una discusión
- Votar en una discusión
- Eliminar una discusión
- Cerrar una discusión
- Crear un hito
- Modificar un hito
- Eliminar un hito
- Cambiar el estado actual del proyecto
- Crear una tarea
- Imputar horas a una tarea
- Dar por finalizada una tarea
- Eliminar una tarea
- Ver resumen de tareas
- Crear una reunión
- Invitar contactos a la reunión

- Clausurar una reunión
- Chatear con un contacto

6.6.3.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

Basándose en la experiencia de las tareas realizadas, cada usuario evaluado deberá responder a las siguientes preguntas relacionadas con su experiencia de uso con la aplicación.

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación?				
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Funciona cada tarea como Vd. Espera?				
¿El tiempo de respuesta de la aplicación es muy grande?				
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
Diseño de la Interfaz	Si	No	A veces	
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				
Observaciones				
Cualquier comentario del usuario				

6.6.3.1.4 Cuestionario para el Responsable de las Pruebas

El responsable de realizar las pruebas deberá llenar el siguiente cuestionario por cada usuario evaluado.

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	
Tiempo en realizar cada tarea	
Errores leves cometidos	
Errores graves cometidos	
Tarea de mayor dificultad	
Tarea de menor dificultad	

<i>Impresiones generales de los usuarios</i>	
<i>Cuestiones a mejorar</i>	

Capítulo 7. Implementación del Sistema

7.1 Lenguajes de Programación

El lenguaje de programación utilizado ha sido Java en su versión 1.6, Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 1990. El lenguaje toma mucha de su sintaxis de C y C++ aunque supone muchos avances con respecto a estos, la memoria se reserva y se libera de forma automática por lo que se puede prescindir de elementos de tan bajo nivel como los punteros. Actualmente la especificación del lenguaje Java se encuentra enmarcada en la Java Community Process , concretamente por el JSR 901.

Las aplicaciones Java son compiladas a un código intermedio llamado bytecode que es ejecutado por una máquina virtual, que se ejecuta sobre la máquina real. La máquina virtual de Java es definida por una especificación (Actualmente por el JSR 924), existen diversas implementaciones de esta especificación para diversos lenguajes como C, Python, Ruby o PHP.

La JVM lee ficheros .class que contienen código máquina de la JVM, y ejecuta las operaciones sobre la máquina real, las operaciones contenidas en estos ficheros .class así como todo el formato del fichero son definidos por la especificación de la JVM. En última instancia se traducirían instrucciones máquina de la JVM a instrucciones máquina de la máquina real.

Las implementaciones de la JVM que provee Oracle, Client Hotspot y Server Hotspot realizan JIT (Just in time compilation) para la generación del código máquina, esto junto a otras optimizaciones dotan a la máquina virtual de Java de un alto rendimiento.

El lenguaje Java, la Java Virtual Machine junto a un conjunto de APIs conforman la plataforma Java, que en su versión estándar actualmente se encuentra en su séptima edición, aunque para este proyecto se ha desarrollado sobre JSE 6, cuyos contenidos vienen definidos en el JSR 270.

Parte del sistema ha sido desarrollado usando la tecnología GWT para el desarrollo de aplicaciones enriquecidas para internet, a grandes rasgos esta tecnología toma parte de nuestra aplicación desarrollada en Java, y compila todo el código Java a JavaScript, el JavaScript obtenido como resultado se incrusta en una página HTML que será enviada al cliente. Esto significa que parte del código del sistema desarrollado en Java nunca se ejecutará dentro de una máquina virtual (en producción), si no que será JavaScript ejecutándose en el ordenador del cliente, algo muy jugoso a la hora de valorar algunos de los costes computacionales de nuestra aplicación.

7.2 Convenios adoptados

En este apartado se describen algunos convenios que se han adoptado para el desarrollo del sistema, con el fin de ilustrarlos así como de que sean seguidos en futuras ampliaciones.

7.2.1 Convenios de Codificación

El convenio de codificación aplicado está basado en el convenio de codificación de Java, aunque con algunos retoques basados en las normas de codificación seguidas por JBoss.

- Se usarán nombres completos de clases para los imports evitando el uso de *.
- Los corchetes estarán siempre en una nueva línea, con el fin de visualizar mejor los bloques de código.
- Se separarán las secciones de código añadiendo espacios, comentarios o incluso líneas de comentarios cuando sea necesario.

Estos convenios se han tomado con el fin de facilitar la lectura del código, para ello se ha hecho uso de una plantilla para modificar el formateador de código del Eclipse.

7.2.2 Convenios de Nombrado

Otros convenios a tener en cuenta son los de nombrado, en el desarrollo del código de este sistema se han definido algunos convenios para evitar incoherencias.

- Todas las clases que implementen vistas del Patrón Modelo Vista Presentador tendrán el sufijo View.
- Todas las clases que implementen presentadores del Patrón Modelo Vista Presentador tendrán el sufijo Presenter.
- Todas las interfaces que definían servicios remotos entre cliente y servidor de la aplicación web basada en GWT tendrán el sufijo Service y todas sus implementaciones el sufijo Impl.
- Todas las clases del patrón DAO tendrán el sufijo Dao.
- Todas las clases que implementen el Patrón Translator tendrán el sufijo Mapper.
- Todas las clases del modelo de presentación que extienden al modelo del dominio tendrán el sufijo Info.
- Las interfaces remotas y locales de los EJBs tendrán los sufijos Remote y Local respectivamente.

7.2.3 Convenios de Log

Se han definido estos convenios con el fin de que los mensajes de log que deje el sistema sean coherentes y de fácil seguimiento. Además de garantizar una forma homogénea de obtener los Loggers.

7.2.3.1 Obtención de Loggers

La forma de obtener un Logger será siempre la siguiente:

```
private static Logger log = Logger.getLogger(BusinessExceptionHandler.class);
```

Esto es importante para disminuir el número de referencias a Loggers y que todas las clases tengan un aspecto similar.

7.2.3.2 Niveles de Log

1. FATAL: Se usará para indicar que un servicio crítico ha fallado.
2. ERROR: Se usará para indicar la imposibilidad de atender una petición aunque el servicio deberá continuar.
3. WARN: Se usará para indicar errores no críticos.
4. INFO: Se usará para indicar fases del ciclo de vida de los servicios.
5. DEBUG: Se usará para indicar información extra de las fases del ciclo de vida de los servicios.
6. TRACE: Se usarán para indicar información de la actividad del servicio.

Estos convenios definen como se deben usar los distintos niveles de Log pero no obligan al uso de estos.

7.2.3.3 Configuración del Log

Toda la configuración se realizará mediante XML y se configurarán dos salidas, una a consola y a un fichero.

Cada módulo del sistema tendrá su propio fichero de Log al que se enviarán todos los mensajes con Nivel INFO o superior.

Todos los módulos enviarán a la consola y solo los mensajes de nivel ERROR o superior.

7.3 Herramientas y Programas Usados para el Desarrollo

En este apartado se describen las distintas herramientas usadas así como variantes, versiones y aditamentos de estas.

7.3.1 Eclipse Indigo

Para el desarrollo de las aplicaciones del sistema se ha usado el entorno de desarrollo integrado Eclipse, concretamente la versión denominada Indigo (3.7). Más específicamente se ha usado una versión del IDE preparada para el desarrollo sobre la plataforma JEE. Además se ha enriquecido el IDE con algunos plugins adicionales.

7.3.1.1 *Maven Integration for Eclipse*

Plugin que facilita el desarrollo con Maven desde el eclipse, permite ejecutar los Maven desde el eclipse, resuelve las dependencias de Maven incorporándolas al proyecto como una biblioteca y permite la creación de proyectos de Maven, que presentan la estructura típica de directorios de Maven y son los que soportan las características mencionadas anteriormente.

7.3.1.2 *JBoss Tools*

Conjunto de plugin distribuidos por JBoss, que ofrecen diversas funcionalidades para el desarrollo sobre JEE en particular y con tecnologías JBoss en particular.

Entre las funcionalidades ofrecidas destacan los configuradores de Maven, las herramientas para trabajar con CDI y un adaptador para manipular instancias del JBoss AS7 desde el eclipse.

7.3.1.3 *Google Plugin*

Plugin que dota a Eclipse de varias funcionalidades relacionadas con el desarrollo de aplicaciones basadas en tecnología GWT.

El plugin ofrece la posibilidad de ejecutar la aplicación en modo de desarrollo desde el Eclipse, para ello arranca un servidor Jetty en el puerto 8888.

Además ofrece ayudas para crear nuevos proyectos así como elementos propios de la tecnología.

7.3.1.4 *Metrics*

Con el fin de mantener una idea de las dimensiones de las aplicaciones del sistema y de la calidad de estas se instaló este plugin al Eclipse.

El plugin genera un listado de métricas de varios tipos para los proyectos seleccionados en el Eclipse. Algunas de las métricas calculadas por el plugin son número de clases, número de métodos, líneas de código, líneas de método, número de interfaces, número de atributos estáticos o complejidad ciclomática, entre otras muchas.

7.3.2 Apache Maven

Herramienta altamente implantada para la gestión de dependencias, hasta tal punto que es imposible o muy difícil incorporar ciertos componentes si no se desarrolla con esta herramienta.

Además de resolver dependencias, Maven es un framework de plugins, es capaz de ejecutar plugin que le sean adecuadamente configurados. Por defecto cuenta con plugins para compilar empaquetar, testear y limpiar el código fuente, entre otros.

Para el desarrollo de las aplicaciones de este sistema, el uso de Maven ha sido un pilar, al punto que la primera tarea de cada aplicación ha sido la de configurar el grueso de dependencias y plugins necesarios para la aplicación completa.

Para dar un idea de las ventajas que ha aportado Maven al desarrollo, mediante una orden sola orden se puede compilar , empaquetar , desplegar un artefacto temporal sobre el contendor, ejecutar tests dentro del contendor , replegar el artefacto de testing, instalar el artefacto real en el repositorio local de Maven y desplegarlo en el contendor para su uso.

Otro ejemplo de la utilidad de Maven en el proyecto fue la de generar para el mismo proyecto dos artefactos distintos, el artefacto de la capa e negocio, y otro artefacto con las interfaces para las invocaciones remotas, todo ello añadiendo una simple opción a la orden.

A continuación se enumeran los distintos plugin de Maven que se han añadido o que han sido configurados de alguna forma particular.

7.3.2.1 *Maven-jar-plugin*

Se ha configurado el plugin encargado de construir el Java Archive RAR durante la fase de empaquetado, para que permita construirlo de varias formas en función de opciones que se pasan al ejecutar Maven.

7.3.2.2 *JBoss-as-maven-plugin*

Se añadió un plugin que nos permite desplegar y replegar artefactos en el JBoss AS7 durante las fases de Maven. Gracias a este plugin se puede por ejemplo desplegar un artefacto cada vez que se ejecute la fase de empaquetado.

7.3.2.3 *Maven-javadoc-plugin*

Se ha configurado este plugin para generar el javadoc desde Maven utilizando un doclet más avanzado (APIviz) que permite añadir diagramas de clases UML al javadoc.

7.3.2.4 *GWT-maven-plugin*

Se ha añadido un plugin para trabajar con GWT desde Maven, este plugin nos permite realizar la compilación a Java script durante la fase de empaquetado y ejecutar el modo de desarrollo desde Maven entre otras cosas.

7.3.2.5 *Maven-war-plugin*

Se ha configurado el plugin que construye el Web Archive RAR para que excluya ciertos ficheros que no es necesario que sean incluidos.

7.3.2.6 *Maven-clean-plugin*

Se ha configurado el plugin que realiza la tarea de limpiar el directorio objetivo con el fin de que elimine además otros artefactos generados por GWT.

7.3.3 API VIZ

APIviz es una implementación del API doclet que genera la documentación de las clases Java añadiéndoles algunos diagramas de clases y de paquetes, se integra perfectamente con Maven y aporta en mi opinión un extra de claridad a la documentación del código Java. La versión usada ha sido la 1.3.1 General Availability, integrada con el maven-javadoc-plugin.

Para que los gráficos sean generados es preciso contar con la herramienta Graphviz en el path del sistema.

7.3.4 Graphviz

Es una herramienta de código libre para la representación de gráficos, la herramienta toma descripciones de gráficos en un lenguaje de texto y genera gráficos en distintos formatos.

En este proyecto es preciso tenerla instalada para la generación del JavaDoc, la versión usada ha sido la 2.28.

7.3.5 PgAdmin

El Sistema Gestor de Bases de Datos usado ha sido PostgreSQL, este cuenta con una herramienta muy útil para la consulta y manipulación de los datos, PgAdmin. La versión usada de PgAdmin ha sido la 9.1, la misma que el SGBD.

El PgAdmin nos ofrece una interfaz gráfica con la que crear y consultar bases de datos, podemos manejar varias bases de datos, en este proyecto se ha creado una base de datos para los tests y otra para la ejecución de la aplicación. Para cada base de datos, esta aplicación cliente, nos permite crear, consultar y modificar todas sus tablas, tanto visualmente como mediante SQL.

PgAdmin facilita la modificación de los ficheros de configuración del SGBD mediante paneles gráficos.

7.3.6 Enterprise Architect

Potente herramienta para el modelado de diagramas UML, se ha utilizado para el desarrollo de los diagramas durante todas las fases del proyecto. La versión utilizada ha sido la 7.5.

7.3.7 Notepad ++

Editor gratuito de código fuente, que soporta varios lenguajes de programación y de marcado. Puede ser ejecutado en Microsoft Windows y se ha utilizado su versión 5.9 para la edición y manipulación de diversos ficheros, ficheros de código Java, ficheros de XML, HTML, de configuración o de log.

7.4 Creación del Sistema

Este apartado describe todos los aspectos destacables de la fase de implementación del sistema.

7.4.1 Problemas Encontrados

Durante la fase de desarrollo se encontraron numerosos problemas, en su gran mayoría relacionados con los mecanismos de desarrollo propios de la tecnología GWT y con la configuración de la infraestructura del sistema.

Los problemas descritos a continuación se mantuvieron ocultos durante toda la fase investigación y aprendizaje y se manifestaron cuando el sistema empezó a crecer y ya eran muy difíciles de eludir.

7.4.1.1 *Modo de Desarrollo de GWT*

El proceso de compilación del código Java a código Java Script se torna bastante lento e incluso desesperante cuando la aplicación se hace grande, para dar una idea las últimas compilaciones se adueñaron de entre 10 y 15 minutos. Para palear esto, o para hacer viable el desarrollo la tecnología cuenta con un modo de desarrollo, en el cual no es necesario compilar el código si no que el navegador mediante un plugin ejecuta el código directamente en la máquina virtual.

El problema manifestado fue que al manejar cierta cantidad de clases este plugin empezaba a hacer aguas y sucumbía toda la ejecución, además de tardar cerca de un minuto para realizar la tan esperada ejecución en la que se habían cambiado algunos iconos.

Después de varios días lidiando con este problema se consiguió dar solución a la crisis añadiendo una línea en el pom del proyecto, puede verse en la Figura 179, marcada en rojo.

Esta línea aumentaba la memoria de la instancia de la máquina virtual dentro de la cual se ejecutaba el código Java que se estaba ejecutando en lugar del Java script sin compilar.

Hubiese sido imposible acabar la aplicación Web sin solucionar este problema en un punto en el cambiar de tecnología suponía tirar una gran cantidad de trabajo a la basura con una baja garantía de éxito debido al tiempo perdido.

```

<!-- ***** GWT PLUGIN ***** -->
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>gwt-maven-plugin</artifactId>
  <configuration>
    <logLevel>TRACE</logLevel>
    <extraJvmArgs>-XX:MaxPermSize=512M -Xmx1024M</extraJvmArgs>
    <!-- -Xms256m (increment vm heap) -->
    <soyc>false</soyc>
    <hostedWebapp>src/main/webapp/</hostedWebapp>
    <!-- <server>org.jboss.errai.cdi.server.gwt.JettyLauncher</server> -->

    <!-- Jboss as7 host mode-->
    <runTarget>http://${web.tier.ip}:${web.tier.port}/cnpd-web/App.html</runTarget>
    <noServer>true</noServer>

    <skip>${gwt.compile.skip}</skip>

  </configuration>
  <executions>
    <execution>
      <id>gwt-clean</id>
      <phase>clean</phase>
      <goals>
        <goal>clean</goal>
      </goals>
    </execution>
    <execution>
      <id>gwt-compile</id>
      <goals>
        <goal>resources</goal>
        <goal>compile</goal>
      </goals>
    </execution>
  </executions>
</plugin>

<!-- ***** War Plugin for custom war building ***** -->

```

Figura 179: Solución Problema Plugin GWT

7.4.1.2 Permutaciones ++

El compilador de GWT, con el fin de generar un código Java Script que sea compatible con múltiples navegadores, compila una permutación por cada navegador o agente de usuario como los llama la tecnología, esto es, para 6 navegadores y 2 idiomas se compilan 12 permutaciones. Esta fue una de las razones por las que no se internacionalizó la aplicación. ¿Dónde está el problema?

Con el fin de que la aplicación esté internacionalizada, la alternativa consiste en compilar una permutación por cada idioma y cada navegador, esto es, para 6 navegadores y 2 idiomas se compilan 12 permutaciones. Esta fue una de las razones por las que no se internacionalizó la aplicación. ¿Dónde está el problema?

En GWT el código se compila por módulos, un módulo es un componente que lleva un fichero de configuración propio de la tecnología y puede ser compilado y referenciado por otros módulos. Un dato importante es que todos los módulos incluidos son compilados juntos que el módulo desarrollado, pues bien, uno de los módulos incluidos tenía definidos diez idiomas distintos, esto provocó que en un momento dado, después de estar trabajando en modo de

desarrollo bastante tiempo, al realizar una compilación esta no se realizaba con éxito por oscuras razones.

Esto suponía un problema grave ya que sin compilar todo el código desarrollado era absolutamente inservible.

Después de bucear en un fichero de log de más de 10000 líneas se descubrió que el error era que la maquina instancia de la máquina virtual dentro de la cual se ejecutaba la compilación, quedaba sin memoria durante la compilación.

Obviamente la primera idea fue aumentar la memoria, pero no fue suficiente, aunque si fue suficiente para darme cuenta de que se estaban intentando compilar demasiadas permutaciones, por lo que empecé a investigar y descubrí la causa del problema. Una vez aquí fue fácil, basta sobrescribir esa definición.

7.4.2 Métricas

Con el fin de describir las dimensiones del sistema se han obtenido algunas métricas de tamaño usando al plugin *Metrics* para Eclipse.

Se trata de métricas muy discutibles, pero que sirven en cualquier caso para comparar los módulos entre si y para realizar un juicio subjetivo del tamaño de las aplicaciones, teniendo en cuenta también el diseño y la calidad general del código.

De todas las métricas obtenidas con el plugin se han seleccionado solo las de tamaño que muestran un valor absoluto, excluyendo todas las clases de testing.

- **Número de atributos:** Número de atributos con que cuenta el módulo, incluye a los estáticos.
- **Número de atributos estáticos:** Número de atributos estáticos con que cuenta el módulo.
- **Número de métodos:** Número de métodos con que cuenta el módulo, incluye a los métodos estáticos.
- **Número de métodos estáticos:** Número de métodos estáticos con que cuenta el módulo.
- **Número de clases:** Número de clases, tanto abstractas como no, que incluye el módulo.
- **Número de interfaces:** Número de interfaces que incluye el módulo.
- **Número de paquetes:** Número de paquetes con que cuenta el módulo, un paquete y un sub paquete son considerados paquetes distintos.

- **Número de líneas de código:** Número de líneas de código sin contar ni los comentarios del código ni las sentencias import, aunque sí cuenta los espacios en blanco y por supuesto las anotaciones. También se cuentan las líneas de código de las interfaces.
- **Número de líneas de código de método:** Número de líneas en el seno de un método, no cuenta los espacios en blanco dentro del método, podría verse como el número de líneas que realmente contienen instrucciones. No se cuenta las líneas de código de las interfaces.

7.4.2.1 Métricas (*cnpd-business*)

Métrica	Valor
Número de atributos	185
Número de atributos estáticos	45
Número de métodos	610
Número de métodos estáticos	11
Número de clases	92
Número de interfaces	24
Número de paquetes	14
Número de líneas de código	4623
Número de líneas de código de método	1409

7.4.2.2 Métricas (*cnpd-web*)

Métrica	Valor
Número de atributos	995
Número de atributos estáticos	212
Número de métodos	2162
Número de métodos estáticos	17
Número de clases	442
Número de interfaces	165
Número de paquetes	69
Número de líneas de código	20136
Número de líneas de código de método	7015

7.4.2.3 Métricas (*cnpd-client*)

Métrica	Valor
Número de atributos	80
Número de atributos estáticos	22
Número de métodos	158
Número de métodos estáticos	1
Número de clases	33
Número de interfaces	8
Número de paquetes	5

Número de líneas de código	1367
Número de líneas de código de método	467

7.4.3 Descripción Detallada de las Clases

Dado que el número de clases de la aplicación es muy elevado (lo que supondría que este apartado tuviera una extensión bastante desproporcionada) y que existen herramientas electrónicas que permiten un mejor uso y consulta de esta información, no se ha incluido la descripción detallada de las clases en este apartado, sino que se ha decidido generar esta información utilizando la herramienta Javadoc y albergarla en el CD adjunto.

No obstante, y a modo de ejemplo de lo que se puede encontrar en el CD se incluye una captura del Javadoc correspondiente a una de las clases de la capa de negocio (Figura 180).

com.armandorv.cnpd.business.exception

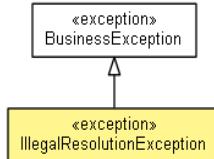
Class IllegalResolutionException

```
java.lang.Object
  ↘ java.lang.Throwable
    ↘ java.lang.Exception
      ↘ java.lang.RuntimeException
        ↘ com.armandorv.cnpd.business.exception.BusinessException
          ↘ com.armandorv.cnpd.business.exception.IllegalResolutionException
```

All Implemented Interfaces:

java.io.Serializable

```
public class IllegalResolutionException
extends BusinessException
```



Exception thrown when a illegal resolution has occurred. A illegal resolution could be :

- Try to resolve a contact request for a user who hasn't a request of that contact.
- Try to resolve a project invitation for a user who hasn't a invitation for that project.
- Try to resolve a meeting invitation for a user who hasn't a invitation for that meeting.

A projectId could match with a legal project but not with a legal project invitation.

Author:

armandorv

See Also:

[Serialized Form](#)

Constructor Summary

[IllegalResolutionException\(java.lang.String message\)](#)

Figura 180: Ejemplo Javadoc

Capítulo 8. Desarrollo de las Pruebas

Para la ejecución de las pruebas de forma automática, se ha usado la herramienta *JUnit 4*, con la que es posible ejecutar las pruebas tanto desde el IDE *Eclipse* como desde *Maven*.

Además de permitir la ejecución automática de pruebas dirigidas por anotaciones, JUnit es altamente extensible, y permite mediante fácil configuración la ejecución de pruebas especiales, como pruebas en un contendor o enmarcadas en el contendor de Inyección de Dependencias de Spring. En este sistema se han incorporado dos frameworks que extienden JUnit, Arquillian y *Spring Test*.

Arquillian ha sido usado extensamente, casi todas las pruebas se han desarrollado sobre él, y la funcionalidad que aporta es nada menos que la de desplegar un artefacto, construido a la carta, en un contendor, una vez allí ejecuta las pruebas, repliega el artefacto y nos muestra el resultado, en Eclipse o en Maven.

Spring Test se ha usado para realizar las pruebas de las clases relacionadas con la autenticación, este módulo de Spring Framework nos permite obtener acceso al contexto de Spring durante una prueba, entre otras cosas.

8.1 Pruebas Unitarias

En este apartado se muestran los resultados de las pruebas unitarias, especificadas en el apartado 6.6 “Especificación Técnica del Plan de Pruebas”.

Para la ejecución de estas pruebas se ha creado una base de datos paralela, con el fin de poder ejecutarlas en cualquier momento sin que interfieran con los datos que haya en la base de datos principal. Además se desarrollaron algunas clases de utilidad para encapsular el proceso de construcción de artefactos de despliegue así como la creación de objetos de prueba y su inserción en la base de datos de pruebas.

8.1.1 Pruebas Unitarias Capa de Persistencia

Se desarrolló una clase de pruebas para cada DAO, estas clases prueban las operaciones CRUD que ofrecen los distintos DAOs sobre las entidades del sistema.

Antes de ejecutar estas pruebas es recomendable limpiar la base de datos y ejecutar la clase *com.armandorv.cnpd.test.BBDDPersistencePopulator*, provista junto al resto de utilidades de prueba. Esta clase crea varias entidades en la base de datos, sin ningún tipo de lógica, con el fin de contar con datos que permitan probar las operaciones CRUD sin fracasar por la integridad referencial, de la que no se deben ocupar las clases de esta capa, si no las del siguiente nivel.

Además todas las clases de prueba de la capa de persistencia deben heredar de la clase `com.armandorv.cnpd.test.TransactionalTest` que hereda la clase base para todos los tests que usen Arquillian, `com.armandorv.cnpd.test.ArquillianSupport`, y le añade la funcionalidad de abrir una transacción antes de cada método y cerrarla después, esto es necesario ya que los DAOs están pensados para ser ejecutados dentro de una transacción de JTA.

Todas las clases de prueba de la capa de persistencia han sido agrupadas en una `TestSuite` llamada `PersistenceTests` que ejecuta todas las clases de Test en una sola ejecución. Las siguientes capturas muestra el resultado de ejecutar todos los Tests.

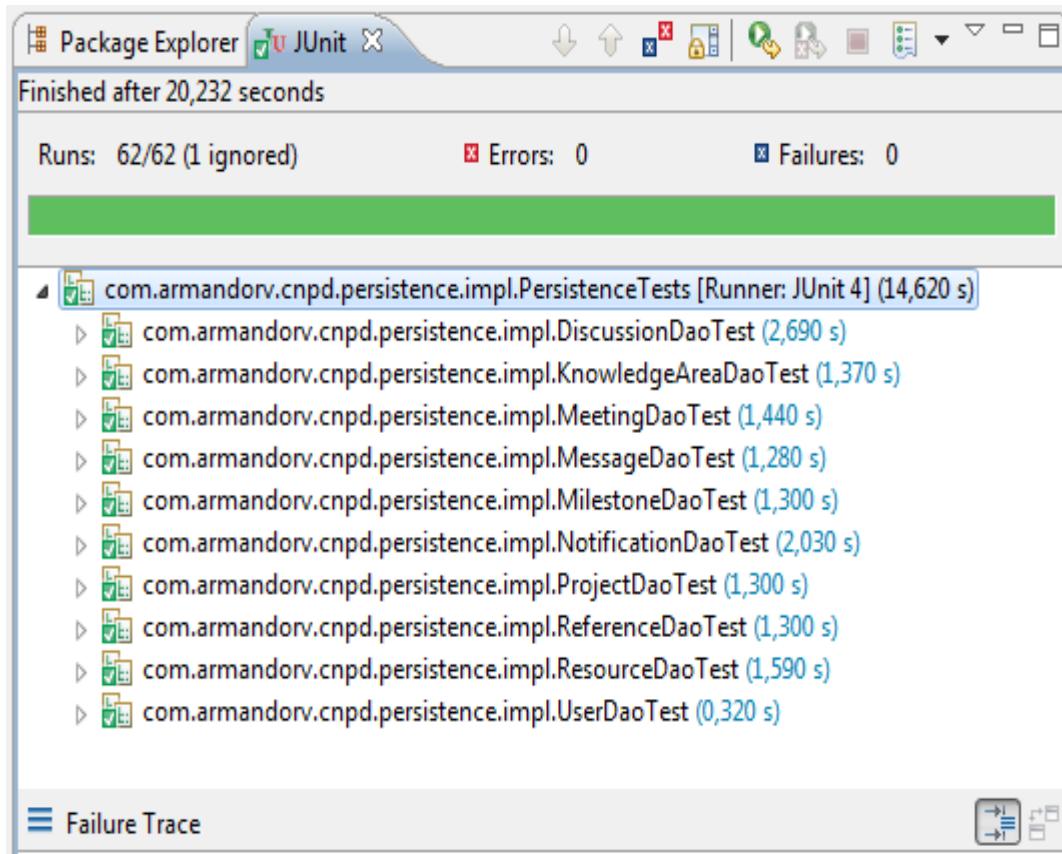


Figura 181: Desarrollo Pruebas Unitarias – Persistencia

La Figura 181 muestra los resultados de cada una de las clases de Test, que a su vez contienen un método de pruebas por cada método del DAO al que pretenden probar. La Figura 182 muestra los resultados por método de algunas de las clases probadas, que como se puede ver se repiten en cada DAO, por lo que no se muestran los de todas las clases.

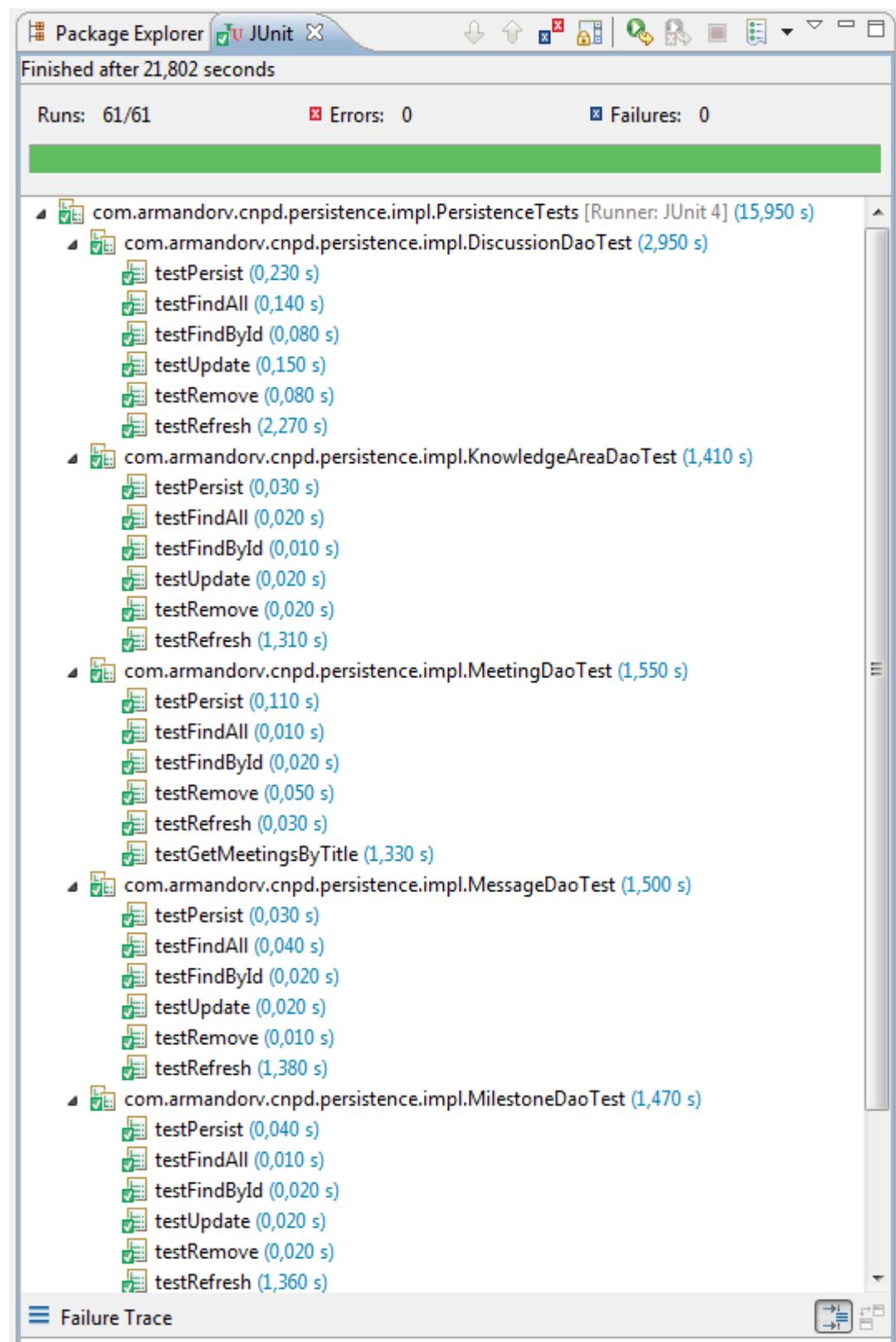


Figura 182: Desarrollo Pruebas Unitarias – Persistencia II

8.1.2 Pruebas Unitarias Capa de Negocio

En la capa de negocio se han realizado clases de prueba para las distintas clases de infraestructura y finalmente una clase de prueba por cada una de las cuatro grandes fachadas del sistema, UsersManager, ContactsManager, ProjectsManager y ResourcesManager.

8.1.2.1 Pruebas de Clases de Utilidad y Soporte

Existen varias clases que desarrollan tareas de infraestructura en la capa de negocio, como pueden ser enviar correos electrónicos o crear ficheros en el repositorio de ficheros, estas clases no requieren interacción con la base de datos para ser probadas.

Algunas de estas clases tienen la gran particularidad de que pueden ser probadas fuera del contenedor, aunque en estos casos se han realizado las dos pruebas, dentro y fuera del contenedor. La siguiente captura muestra el resultado de ejecutar estas pruebas.

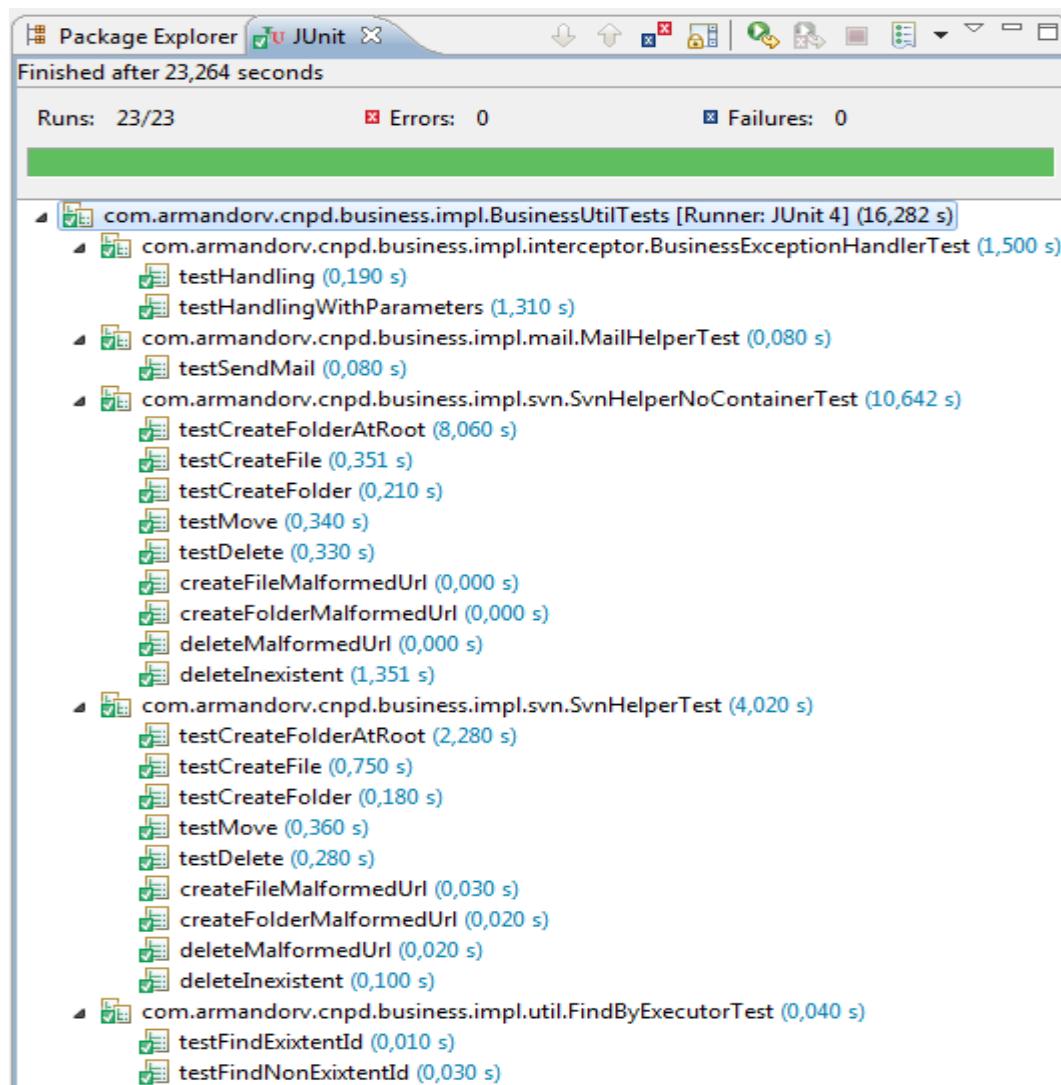


Figura 183: Desarrollo de Pruebas Unitarias – Capa de Negocio I

8.1.2.2 Pruebas de las Fachadas de Negocio

Las operaciones de estas son operaciones que integran todos los DAOs, y mucha lógica, propio de una prueba de integración, dado que el sistema es extremadamente, vertical estas pruebas siguen teniendo un nivel de integración bajo en comparación con las pruebas de integración consideradas en el apartado 6.6 “Especificación Técnica del Plan de Pruebas” y además no están dirigidas por casos de usos si no por clases, por lo que en el marco de un estudio de las pruebas de todo el sistema, se han contemplado como pruebas unitarias , aunque no prueben las unidades de forma aislada. Esto no quiere decir que no estén contempladas, al igual que todo el sistema por las pruebas de integración y de sistemas.

Se ha diseñado una clase de pruebas por cada fachada, estas pruebas serán ejecutadas por Arquillian por lo que deben extender de la clase `com.armandorv.cnpd.test.ArquillianSupport` y no necesitan abrir y cerrar transacciones ya que las clases probadas son beans de sesión sin estado, que hacen por defecto a todos sus métodos transaccionales. Sí que es conveniente limpiar la base de datos y ejecutar la clase `com.armandorv.cnpd.test.BBDBBusinessPopulator`, con el fin de tener datos correctos para las pruebas, además se debe limpiar el repositorio de ficheros de prueba, para eliminar ficheros de pruebas anteriores.

Se agrupado los tests de estas fachadas en una Suite de forma que se puedan ejecutar todos juntos, las siguientes capturas muestran los resultados de ejecutar esta Suite, comenzando por una vista global y profundizando en los métodos de cada clase.

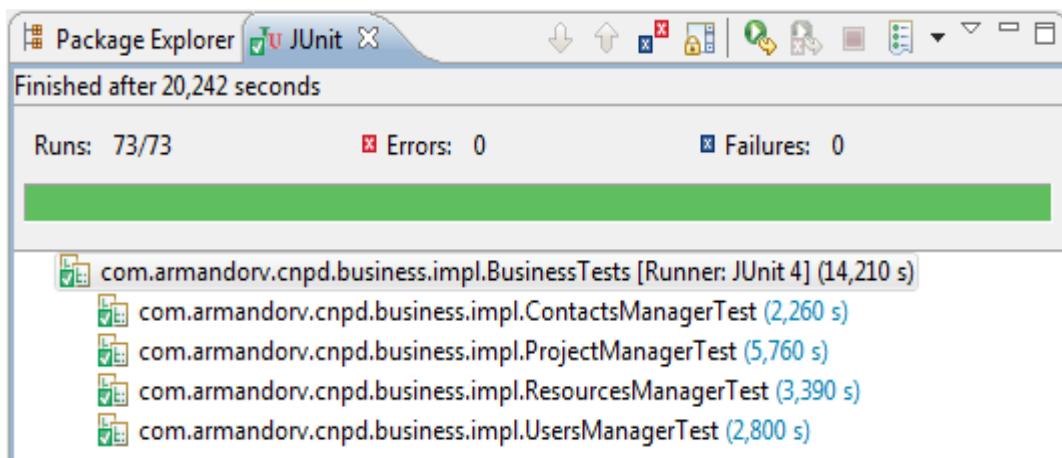


Figura 184: Desarrollo de Pruebas Unitarias – Capa de Negocio II

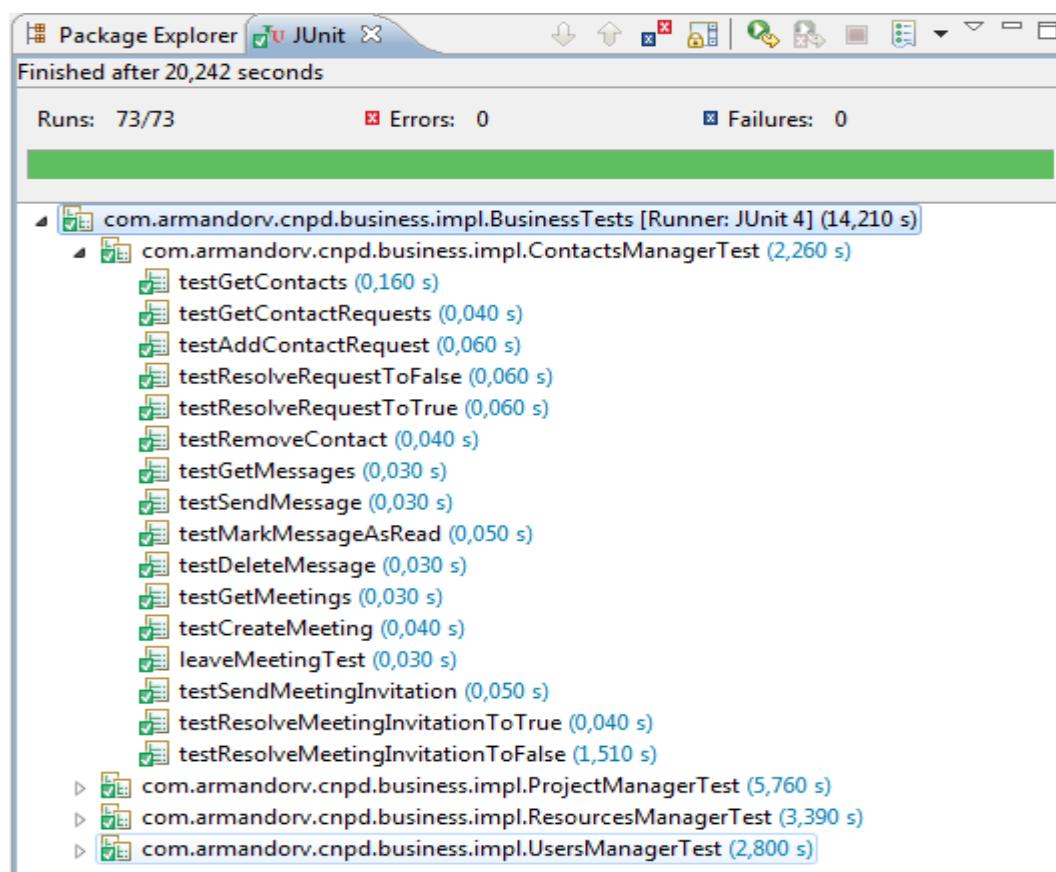


Figura 185: Desarrollo de Pruebas Unitarias – Capa de Negocio II

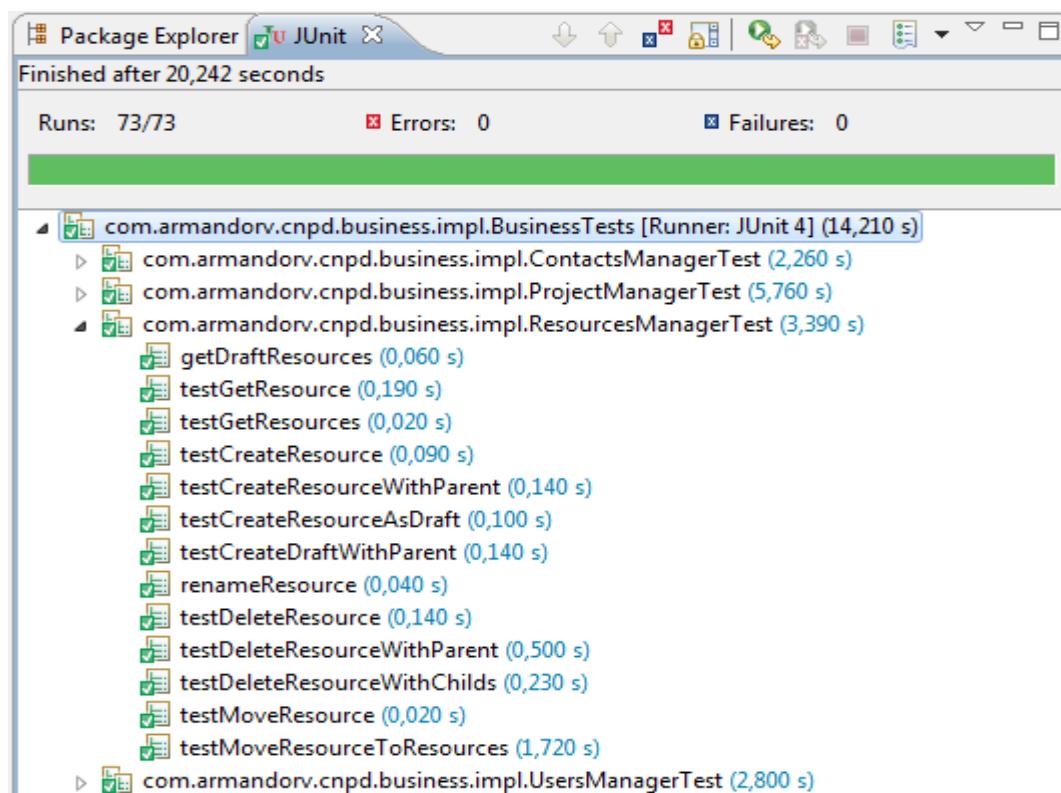


Figura 186: Desarrollo de Pruebas Unitarias – Capa de Negocio II

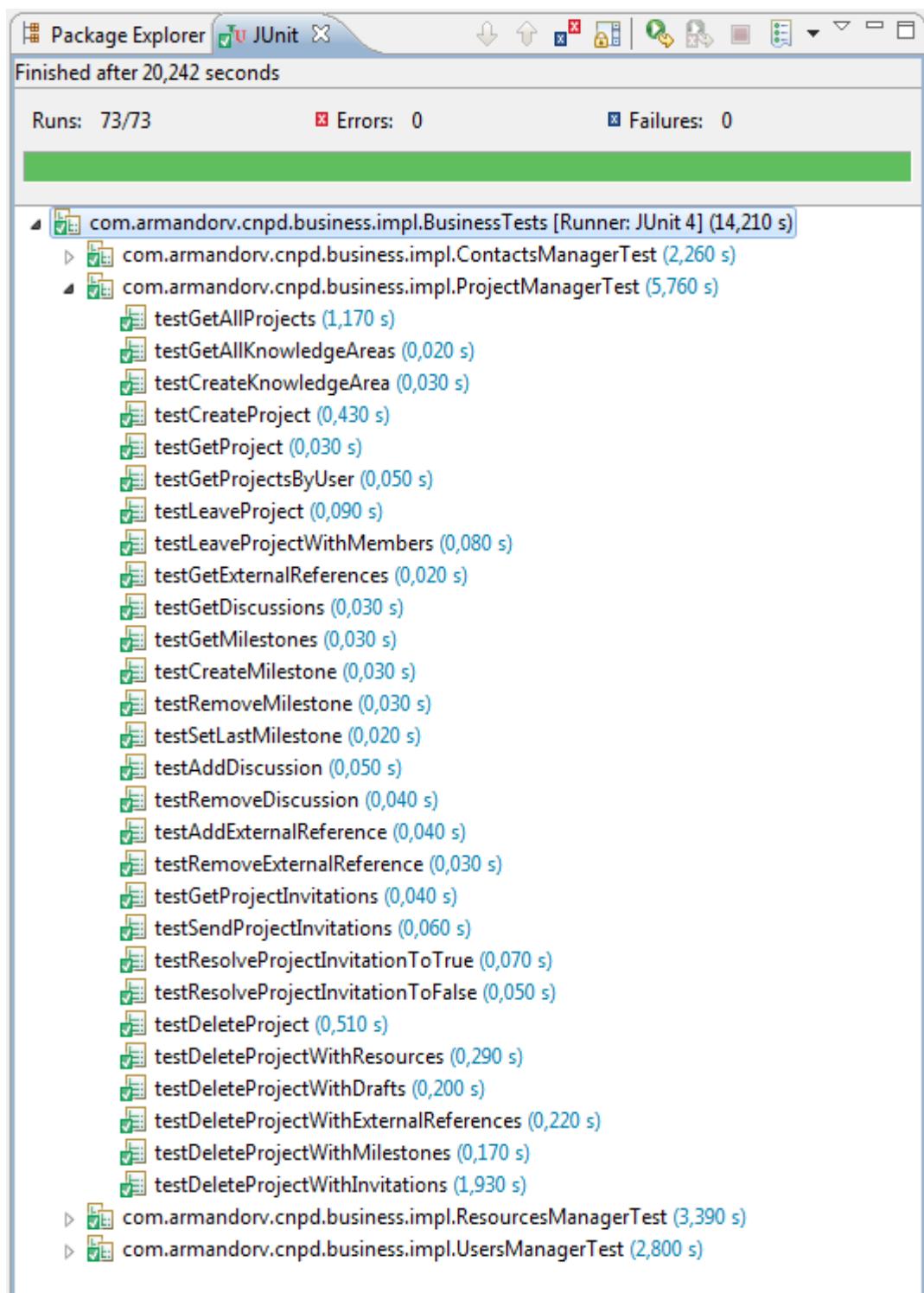


Figura 187: Desarrollo de Pruebas Unitarias – Capa de Negocio III

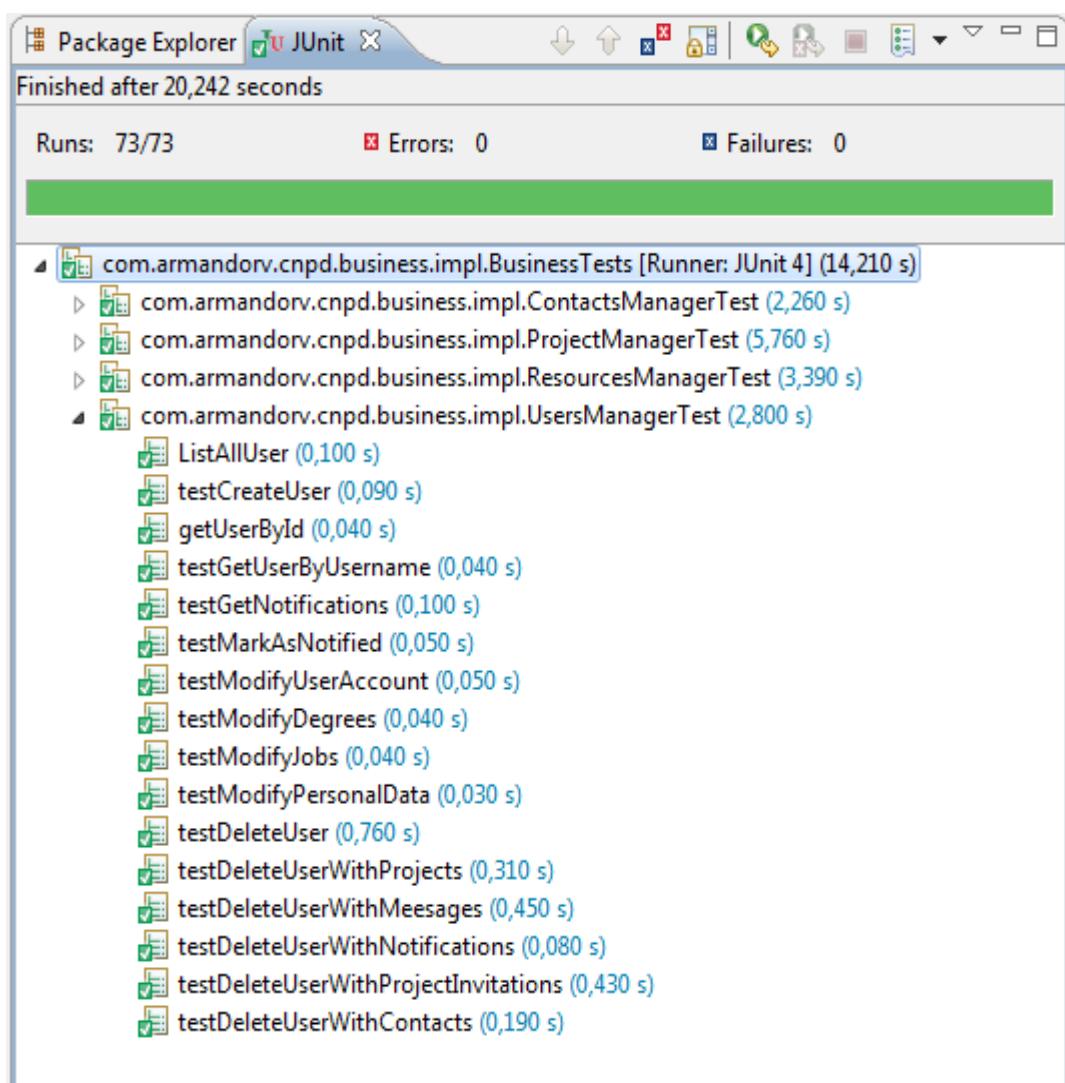


Figura 188: Desarrollo de Pruebas Unitarias – Capa de Negocio IV

8.1.3 Pruebas Unitarias Capa de Presentación

En esta capa se han desarrollado pruebas de unidad para las distintas clases de utilidad e infraestructura desarrolladas así como todas las clases de la jerarquía de mapeadores definida para la conversión de entidades a objetos de presentación.

Para automatizar la repetición de estas pruebas se crearon dos Suites de Test, una con las pruebas de todas las clases de infraestructura, y otra con las pruebas de los mapeadores (Figura 189 y Figura 190).

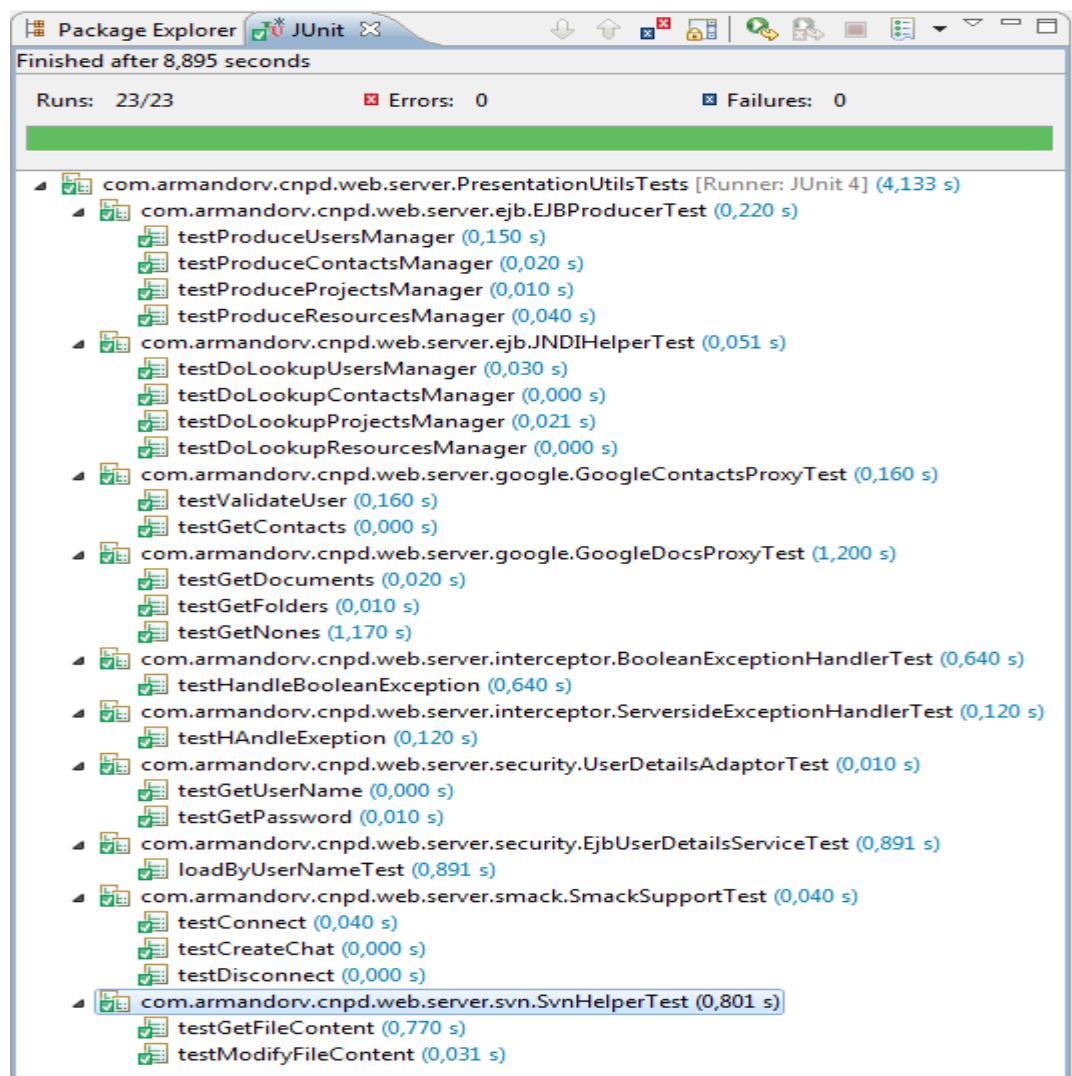


Figura 189: Desarrollo Pruebas Capa de Presentación I

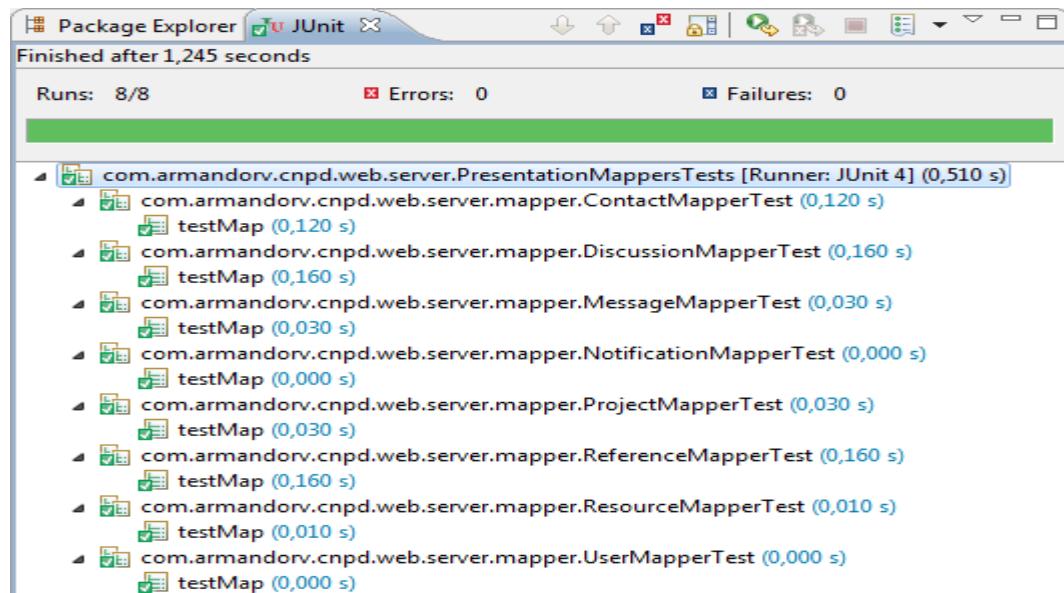


Figura 190: Desarrollo Pruebas Capa de Presentación II

8.2 Pruebas de Integración y del Sistema

En este apartado se muestran los resultados de las pruebas de integración y del sistema definidos en el apartado 6.6.2 “Pruebas de Integración y del Sistema”. Estas pruebas se han ido realizando a medida que la implementación del proyecto evolucionaba, repitiéndose y modificándose con el fin de adaptarse a la implementación del sistema. En esta sección se muestran los resultados obtenidos de aplicar las pruebas definidas previamente.

Para las pruebas de integración, se ha usado la herramienta JUnit complementada con Arquillian, para ejecutarlas directamente sobre el contenedor los métodos de más alto nivel del sistema, que son los ofrecidos remotamente a la parte cliente de la aplicación GWT.

Las pruebas de sistema han sido ejecutadas de forma manual sobre las distintas versiones que se han ido obteniendo del sistema.

Las tablas que se muestran a continuación, muestran los resultados de aplicar a cada caso de prueba definido, tanto las pruebas de integración como las de sistema. Además cabe destacar que estas pruebas se han ejecutado varias veces atendiendo a la evolución del sistema.

Caso de uso 1.1: Añadir Usuario	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Añadir a un usuario no existente en la aplicación	<p>El sistema posee un usuario más.</p> <p>Resultado Obtenido 1</p> <p>El sistema en algunos casos no permitía la importación de los contactos, debido a comportamientos extraños de la interfaz de usuario.</p> <p>Resultado Obtenido 2</p> <p>Debido a un error de programación el sistema creaba el usuario sin la ciudad, a pesar de que este la introducía adecuadamente durante el registro.</p> <p>Resultado Obtenido 3</p> <p>Prueba Correcta.</p>
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Añadir a un usuario que ya existe.	<p>El sistema no posee un usuario más y se genera un error indicando que el usuario ya existe en la aplicación.</p> <p>Resultado Obtenido 1</p> <p>Prueba Correcta</p>
Caso de Prueba: CP1.1.3	
Entrada	Resultado Esperado
Añadir usuario no existente con información incorrecta.	<p>El sistema no posee un usuario más y se genera un error indicando que el usuario ya existe en la aplicación.</p> <p>Resultado Obtenido</p> <p>Prueba Correcta</p>

Caso de Uso 1.2: Modificar Usuario
Caso de Prueba: CP1.1.1

Entrada	Resultado Esperado
Modificar a un usuario ya existente en la aplicación con datos correctos.	El usuario es modificado y no se generan errores.
	Resultado Obtenido 1
	Debido a un error de programación el sistema creaba el usuario sin la ciudad, a pesar de que este la introducía adecuadamente.
	Resultado Obtenido 2
	Prueba Correcta
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Modificar a un usuario ya existente en la aplicación con datos incorrectos.	El usuario no es modificado y se genera un error indicando que el usuario ya existe en la aplicación.
	Resultado Obtenido 1
	Prueba Correcta.

Caso de Uso 1.3: Eliminar Usuario	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Eliminar a un usuario del sistema, sin proyectos, y sin contactos.	El sistema posee un usuario menos y no se generan errores.
	Resultado Obtenido 1
	Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Eliminar a un usuario del sistema, sin proyectos, y con contactos.	El sistema posee un usuario menos y no se generan errores.
	Resultado Obtenido 1
	Prueba Correcta.
Caso de Prueba : CP1.1.3	
Entrada	Resultado Esperado
Eliminar a un usuario del sistema, sin proyectos, y con contactos.	El sistema posee un usuario menos y no se generan errores.
	Resultado Obtenido 1
	Prueba Correcta.

Caso de Uso 2.1: Buscar Usuario	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Obtener todos los usuarios dado un parámetro.	El sistema devuelve los usuarios que contienen el parámetro entre sus datos personales.
	Resultado Obtenido 1
	Prueba Incorrecta, el sistema devuelve en ocasiones el usuario buscador, y esto no es correcto.
	Resultado Obtenido 2
	Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Obtener todos los usuarios dados dos parámetro.	El sistema devuelve los usuarios que contienen los dos parámetros entre sus datos personales.
	Resultado Obtenido 1
	Prueba Incorrecta, el sistema devuelve en ocasiones el

	usuario buscador, y esto no es correcto.
	Resultado Obtenido 1
	Prueba Correcta.

Caso de Uso 2.1: Añadir Usuario como Contacto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Añadir a un usuario que no se tiene en la lista de contactos, y al que se le ha hecho una petición.	El sistema añade el usuario seleccionado a la lista de contactos del usuario. Resultado Obtenido 1 Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Añadir a un usuario que se tiene en la lista de contactos.	El sistema no debe permitirlo y se debe generar un error. Resultado Obtenido 1 Prueba Correcta.
Caso de Prueba: CP1.1.3	
Entrada	Resultado Esperado
Añadir a un usuario que no se tiene en la lista de contactos, y al que no se le ha hecho una petición.	El sistema no debe permitirlo y se debe generar un error. Resultado Obtenido 1 Prueba Correcta.

Caso de Uso 2.2: Ver perfil de Contacto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Ver contacto que se tiene en la lista de contactos.	El sistema devolverá el contacto, con su lista de contactos y de proyectos publicados. Resultado Obtenido 1 Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Ver contacto que no se tiene en la lista de contactos.	El sistema no debe permitirlo y se debe generar un error. Resultado Obtenido 1 Prueba Correcta.

Caso de Uso 2.3: Comunicarse con Contacto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Se envía un mensaje a un usuario que se tiene como contacto.	El usuario destinatario tendrá un nuevo mensaje. Resultado Obtenido 1 Prueba Incorrecta, al responder a un mensaje enviado, este se enviaba al usuario emisor. Resultado Obtenido 2 Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado

Se envía un mensaje a un usuario que no se tiene como contacto.	El sistema no lo permitirá y generará un error en caso de que esto pase.
	Resultado Obtenido 1
	Prueba Correcta.

Caso de Uso 3.1: Crear Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Crear un nuevo proyecto sin miembros.	Habrá un nuevo proyecto en el sistema, solo tendrá un participante, será privado y el gestor será su único participante y creador. Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Crear un nuevo proyecto con miembros.	Habrá un nuevo proyecto en el sistema, solo tendrá un participante, será privado y el gestor será su único participante y creador. Además se habrán enviado invitaciones a los contactos seleccionados. Resultado Obtenido 1 Entre los miembros a invitar se permitía la invitación del creador del proyecto, esto no es correcto. Resultado Obtenido 2 Prueba Correcta.

Caso de Uso 3.2.1: Incluir /Excluir Contactos	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Excluir contactos de un proyecto.	El proyecto tendrá automáticamente un contacto menos. Resultado Obtenido 1 Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Incluir contactos en un proyecto.	El usuario incluido tendrá una invitación a para unirse al proyecto. Resultado Obtenido 1 Entre los miembros a invitar se permitía la invitación del creador del proyecto, esto no es correcto. Resultado Obtenido 2 Prueba Correcta.

Caso de Uso 3.2.2: Cambiar Gestor	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Asignar el rol de gestor a otro participante.	El gestor del proyecto será el participante seleccionado. Resultado Obtenido 1 Prueba Correcta.

Caso de Uso 3.2.3: Publicar Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
El usuario gestor publica el proyecto.	<p>El proyecto es público.</p> <p>Resultado Obtenido 1</p> <p>El proyecto era accesible desde el listado de contactos pero no desde el buscador, esto es incorrecto.</p> <p>Resultado Obtenido 2</p> <p>Prueba Correcta.</p>
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Un usuario no gestor publica el proyecto.	<p>El sistema no lo permite.</p> <p>Resultado Obtenido 1</p> <p>Prueba Correcta.</p>

Caso de Uso 3.2.1: Eliminar Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
El usuario gestor elimina el proyecto.	<p>El sistema y los miembros tienen un proyecto menos.</p> <p>Resultado Obtenido 1</p> <p>Al eliminar proyectos con invitaciones pendientes de resolver se genera un error, esto es incorrecto.</p> <p>Resultado Obtenido 2</p> <p>Prueba Correcta.</p>
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Un usuario no gestor elimina el proyecto.	<p>El sistema no lo permite.</p> <p>Resultado Obtenido 1</p> <p>Prueba Correcta.</p>

Caso de Uso 3.2.1: Planificar Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
El usuario gestor establece el estado del proyecto.	<p>El sistema tiene un nuevo estado.</p> <p>Resultado Obtenido 1</p> <p>Prueba Correcta.</p>
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
El usuario no gestor establece el estado del proyecto.	<p>El sistema no lo permite.</p> <p>Resultado Obtenido 1</p> <p>Prueba Correcta.</p>

Caso de Uso 3.4.1: Crear Discusión	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Crear una discusión con opciones.	<p>El proyecto tendrá una nueva discusión.</p> <p>Resultado Obtenido 1</p>

	El proyecto creado no contiene las opciones definidas por el usuario.
	Resultado Obtenido 2
	Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Crear una discusión sin opciones.	El proyecto no tendrá una nueva discusión.
	Resultado Obtenido 1
	Prueba Incorrecta, el proyecto aunque no de forma explícita, permitía la creación de discusiones sin opciones.
	Resultado Obtenido 2
	Prueba Correcta.

Caso de Uso 3.2.1: Votar en Discusión	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Un usuario que no ha votado añade un voto a una discusión.	La discusión tendrá un nuevo voto.
	Resultado Obtenido 1
	Prueba Incorrecta, los votos no se añadían de forma adecuada.
	Resultado Obtenido 2
	Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Un usuario que ha votado añade un voto a una discusión.	El sistema no debe permitirlo.
	Resultado Obtenido 1
	Prueba Correcta.

Caso de Uso 3.5: Abandonar Proyecto	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Un miembro que no es gestor abandona el proyecto.	
	El proyecto debe tener un miembro menos.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
El gestor abandona el proyecto.	El proyecto debe tener un miembro menos y un nuevo gestor.
	Resultado Obtenido 1
	Prueba Correcta.

Caso de Uso 4.1: Crear Recurso	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Crear recurso en la raíz.	El proyecto tendrá un recurso más.
	Resultado Obtenido 1

	Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Crear recurso dentro de otro recurso.	Uno de los recursos del proyecto tendrá un recurso más. Resultado Obtenido 1 El nuevo recurso se creaba en la raíz a pesar de haberse seleccionado un recurso padre. Resultado Obtenido 2 Prueba Correcta.

Caso de Uso 4.2: Modificar Recurso	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Modificar un recurso vacío del sistema.	El recurso será modificado. Resultado Obtenido 1 Al mover los recursos, se producían errores en el repositorio de ficheros, que dada la transaccionalidad de las operaciones de negocio, impedían el movimiento. Resultado Obtenido 2 Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Modificar un recurso que contiene otros recursos.	El recurso será modificado y sus hijos permanecerán inalterados. Resultado Obtenido 1 Al mover los recursos, se producían errores en el repositorio de ficheros, que dada la transaccionalidad de las operaciones de negocio, impedían el movimiento. Resultado Obtenido 2 Prueba Correcta.

Caso de Uso 4.3: Eliminar Recurso	
Caso de Prueba: CP1.1.1	
Entrada	Resultado Esperado
Eliminar un recurso vacío del sistema.	El recurso será eliminado. Resultado Obtenido 1 Prueba Correcta.
Caso de Prueba: CP1.1.2	
Entrada	Resultado Esperado
Eliminar un recurso que contiene otros recursos.	El recurso será eliminado, así como todos sus hijos. Resultado Obtenido 1 Prueba Correcta.

8.3 Pruebas de Usabilidad

A continuación se muestran los resultados de aplicar las pruebas de usabilidad especificadas en la fase de diseño. En las pruebas han participado 4 personas que forman parte del dominio de posibles futuros usuarios de la aplicación, que como se especifica en los requisitos no funcionales del sistema son usuarios familiarizados con el uso de aplicaciones web.

Las pruebas con los usuarios consistieron en: responder un test previo para establecer el perfil de cada usuario; realizar las tareas establecidas en el diseño de estas pruebas; una vez realizadas dichas tareas los usuarios cumplimentaron un cuestionario sobre la aplicación y dieron su opinión sobre posibles mejoras en la interfaz.

8.3.1 Resultados Preguntas de Carácter General

El objetivo de este cuestionario es establecer el perfil del usuario que va a realizar las pruebas de usabilidad, las respuestas del Usuario 1 se representan en la tabla por U1 y así sucesivamente.

¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión.--> U1,U2
2. Lo uso básicamente para ocio.--> U3,U4
3. Solo empleo aplicaciones estilo Office.
4. Únicamente leo el correo y navego ocasionalmente.
¿Ha usado alguna vez una aplicación web con contactos (Redes Sociales)?
1. Sí (indicar nombre del sitio web).
2. No, pero conozco algún sitio web.
3. No, nunca.
¿Ha usado alguna vez una aplicación con chat?
1. Sí (indicar nombre del sitio web).--> U1,U2,U3,U4
2. No, pero conozco algún sitio web.
3. No, nunca.
¿Ha usado alguna vez una aplicación para gestionar tareas o recursos?
1. Sí --> U1,U2
2. No, pero conozco algún sitio web.
3. No, nunca
¿Qué busca usted principalmente en una aplicación web (en general)?
1. Que sea fácil de usar.--> U1,U3,U4
2. Que sea intuitiva.--> U2
3. Que sea rápida.
4. Que tenga todas las funcionalidades relacionadas con la temática del sitio web necesarias.
5. En aquellas en las que solicitan datos personales así como login y password que sea segura y mis datos estén protegidos.

8.3.2 Preguntas Cortas sobre la Aplicación y Observaciones

Una vez hechas las tareas definidas en el apartado 6.6 “Especificación Técnica del Plan de Pruebas”, los usuarios han cumplimentado el cuestionario establecido.

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?	U1,U2,U3	U4		
¿Existe ayuda para las funciones en caso de que tenga dudas?	U1, U2, U3, U4.			
¿Le resulta sencillo el uso de la aplicación?	U1, U2, U3	U4.		
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Funciona cada tarea como Vd. Espera?	U1, U2	U3,U4		
¿El tiempo de respuesta de la aplicación es muy grande?			U1, U2	U3,U4
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es		U1,U2,U3,U4		
Los iconos e imágenes usados son		U1,U2,U3,U4		
Los colores empleados son	U4	U1,U2,U3		
Diseño de la Interfaz	Si	No	A veces	
¿Le resulta fácil de usar?	U1,U2,U3,U4			
¿El diseño de las pantallas es claro y atractivo?	U1,U2,U3,U4			
¿Cree que el programa está bien estructurado?	U1,U2,U3,U4			
Observaciones				
El usuario U4 ha recomendado que desde las notificaciones se pueda ir a los objetos de estas notificaciones, y que se permitan editar documentos.				

8.3.3 Cuestionario para el Responsable de las Pruebas

El responsable de realizar las pruebas deberá llenar el siguiente cuestionario por cada usuario evaluado.

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	Se comienza trabajar de forma rápida, en general los usuarios se ponen bien en marcha, se distraen un ligeramente explorando las secciones, pero rápidamente van a sus tareas.
Tiempo en realizar cada tarea	El tiempo de las tareas es el esperado, a modo general.
Errores leves cometidos	Ninguno

<i>Errores graves cometidos</i>	Ninguno
<i>Tarea de mayor dificultad</i>	Crear un proyecto
<i>Tarea de menor dificultad</i>	Crear una referencia
<i>Impresiones generales de los usuarios</i>	Los usuarios se encuentran a gusto.
<i>Cuestiones a mejorar</i>	Edición de Documentos.

8.3.4 Comprobación Heurísticos de Usabilidad

Además de las pruebas de usabilidad cuyos resultados se muestran en los apartados anteriores, se ha pasado la guía de heurísticos de usabilidad desarrollada por Yusef Hassan Montero que se muestra a continuación. [Hassan08]

Criterios	¿Cumplido?
<u>Generales</u>	
¿Cuáles son los objetivos del sitio web? ¿Son concretos y bien definidos?	SI
¿Los contenidos y servicios que ofrece se corresponden con esos objetivos?	SI
¿Tiene una URL correcta, clara y fácil de recordar? ¿Y las URL de sus páginas internas? ¿Son claras y permanentes?	SI
¿Muestra de forma precisa y completa qué contenidos o servicios ofrece realmente el sitio web? El diseño de la página de inicio debe ser diferente al resto de páginas y cumplir la función de 'escaparate' del sitio.	SI
¿La estructura general del sitio web está orientada al usuario? Los sitios web deben estructurarse pensando en el usuario, sus objetivos y necesidades. La estructura interna de la empresa u organización, cómo funciona o se organiza no interesan al usuario.	SI
¿El <i>look & feel</i> general se corresponde con los objetivos, características, contenidos y servicios del sitio web? Ciertas combinaciones de colores ofrecen imágenes más o menos formales, serias o profesionales.	SI
¿Es coherente el diseño general del sitio web? Se debe mantener una coherencia y uniformidad en las estructuras y colores de todas las páginas. Esto sirve para que el usuario no se desoriente en su navegación.	SI
¿Es reconocible el diseño general del sitio web? Cuanto más se parezca el sitio web al resto de sitios web, más fácil será de usar.	SI
¿El sitio web se actualiza periódicamente? ¿Indica cuándo se actualiza? Las fechas que se muestren en la página deben corresponderse con actualizaciones, noticias, eventos...no con la fecha del sistema del usuario.	NO SE ACTUALIZA.
<u>Identidad e Información</u>	
¿Se muestra claramente la identidad de la empresa-sitio a través de todas las páginas?	NO
El Logotipo, ¿es significativo, identificable y suficientemente visible?	NO HAY

El eslogan o <i>tagline</i> , ¿expresa realmente qué es la empresa y qué servicios ofrece?	NO HAY
¿Se ofrece algún enlace con información sobre la empresa, sitio web, 'webmaster',...?	NO
¿Se proporciona mecanismos para ponerse en contacto con la empresa? (email, teléfono, dirección postal, fax...)	NO
¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?	NO
En artículos, noticias, informes... ¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?	SI
<u>Lenguaje y Redacción</u>	
¿El sitio web habla el mismo lenguaje que sus usuarios? Se debe evitar usar un lenguaje corporativista. Así mismo, hay que prestarle especial atención al idioma, y ofrecer versiones del sitio en diferentes idiomas cuando sea necesario.	SI
¿Emplea un lenguaje claro y conciso?	SI
¿Es amigable, familiar y cercano? Es decir, lo contrario a utilizar un lenguaje constantemente imperativo, mensajes crípticos, o tratar con "desprecio" al usuario.	SI
¿1 párrafo = 1 idea? Cada párrafo es un objeto informativo. Trasmita ideas, mensajes... Se deben evitar párrafos vacíos o varios mensajes en un mismo párrafo.	SI
<u>Rotulado</u>	
Los rótulos, ¿son significativos? Ejemplo: evitar rótulos del tipo "haga clic aquí".	SI
¿Usa rótulos estándar? Siempre que exista un "estándar" comúnmente aceptado para el caso concreto, como "Mapa del Sitio" o "Acerca de...".	SI
¿Usa un único sistema de organización, bien definido y claro? No se deben mezclar diferentes. Los sistemas de organización son: alfabético, geográfico, cronológico, temático, orientado a tareas, orientado al público y orientado a metáforas.	SI
¿Utiliza un sistema de rotulado controlado y preciso? Por ejemplo, si un enlace tiene el rótulo "Quiénes somos", no puede dirigir a una página cuyo encabezamiento sea "Acerca de"	SI
El título de las páginas, ¿Es correcto? ¿Ha sido planificado? Relacionado con la 'findability' del sitio web.	SI
<u>Estructura y Navegación</u>	
La estructura de organización y navegación, ¿Es la más adecuada? Hay varios tipos de estructuras: jerárquicas, hipertextual, facetada,...	SI
En el caso de estructura jerárquica, ¿Mantiene un equilibrio entre Profundidad y Anchura?	SI
En el caso de ser puramente hipertextual, ¿Están todos los clúster de nodos comunicados? Aquí se mide la distancia entre nodos.	SI

¿Los enlaces son fácilmente reconocibles como tales? ¿Su caracterización indica su estado (visitados, activos,...)? Los enlaces no sólo deben reconocerse como tales, sino que su caracterización debe indicar su estado, y ser reconocidos como una unidad	SI
En menús de navegación, ¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística? No se deben superar los 7±2 elementos, ni los 2 o, como mucho, 3 términos por elemento.	SI
¿Es predecible la respuesta del sistema antes de hacer clic sobre el enlace? Relacionado con el nivel de significación del rótulo del enlace, aunque también con: el uso de globos de texto, información contextual, la barra de estado del navegador,...	SI
¿Se ha controlado que no haya enlaces que no llevan a ningún sitio? Enlaces que no llevan a ningún sitio: Los enlaces rotos, y los que enlazan con la misma página que se está visualizando (por ejemplo enlaces a la "home" desde la misma página de inicio)	SI
¿Existen elementos de navegación que orienten al usuario acerca de dónde está y cómo deshacer su navegación? ...como <i>breadcrumbs</i> , enlaces a la página de inicio,...recuerde que el logo también es recomendable que enlace con la página de inicio.	SI
Las imágenes enlace, ¿se reconocen como clicables? ¿Incluyen un atributo 'title' describiendo la página de destino? En este sentido, también hay que cuidar que no haya imágenes que parezcan enlaces y en realidad no lo sean.	SI
¿Se ha evitado la redundancia de enlaces?	SI
¿Se ha controlado que no haya páginas "huérfanas"? Páginas huérfanas: que aún siendo enlazadas desde otras páginas, éstas no enlacen con ninguna.	SI
<u>Layout de la Página</u>	
¿Se aprovechan las zonas de alta jerarquía informativa de la página para contenidos de mayor relevancia? (como por ejemplo la zona central)	SI
¿Se ha evitado la sobrecarga informativa? Esto se consigue haciendo un uso correcto de colores, efectos tipográficos y agrupaciones para discriminar información. Los grupos diferentes de objetos informativos de una página deben ser 7±2.	SI
¿Es una interfaz limpia, sin ruido visual?	SI
¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?	SI
¿Se hace un uso correcto del espacio visual de la página? Es decir, que no se desaproveche demasiado espacio con elementos de decoración, o grandes zonas en "blanco", y que no se adjudique demasiado espacio a elementos de menor importancia.	SI
¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página? (La jerarquía visual se utiliza para orientar al usuario)	SI

¿Se ha controlado la longitud de página? Se debe evitar en la medida de lo posible el <i>scrolling</i> . Si la página es muy extensa, se debe fraccionar.	SI
<u>Búsqueda (si es necesario, por la extensión del sitio, incorporar un buscador interno)</u>	
¿Se encuentra fácilmente accesible? Es decir: directamente desde la home, y a ser posible desde todas las páginas del sitio, y colocado en la zona superior de la página.	NO hay
¿Es fácilmente reconocible como tal?	NO hay
¿Permite la búsqueda avanzada? (siempre y cuando, por las características del sitio web, fuera de utilidad que la ofreciera)	NO hay
¿Muestra los resultados de la búsqueda de forma comprensible para el usuario?	NO hay
¿La caja de texto es lo suficientemente ancha?	NO hay
¿Asiste al usuario en caso de no poder ofrecer resultados para una consultada dada?	NO hay
<u>Elementos Multimedia</u>	
¿Las fotografías están bien recortadas? ¿Son comprensibles? ¿Se ha cuidado su resolución?	SI
¿Las metáforas visuales son reconocibles y comprensibles por cualquier usuario? (prestar especial atención a usuarios de otros países y culturas)	SI
¿El uso de imágenes o animaciones proporciona algún tipo de valor añadido?	SI
¿Se ha evitado el uso de animaciones cíclicas?	SI
<u>Ayuda</u>	
Si posee una sección de Ayuda, ¿Es verdaderamente necesaria? Siempre que se pueda prescindir de ella simplificando los elementos de navegación e interacción, debe omitirse esta sección.	NO hay
En enlace a la sección de Ayuda, ¿Está colocado en una zona visible y "estándar"? La zona de la página más normal para incluir el enlace a la sección de Ayuda, es la superior derecha.	NO hay
¿Se ofrece ayuda contextual en tareas complejas? (transferencias bancarias, formularios de registro...)	NO hay
Si posee FAQs, ¿Es correcta tanto la elección como la redacción de las preguntas? ¿Y las respuestas?	NO hay
<u>Accesibilidad (debería cubrirse con los test de Accesibilidad posteriores)</u>	
¿El tamaño de fuente se ha definido de forma relativa, o por lo menos, la fuente es lo suficientemente grande como para no dificultar la legibilidad del texto?	SI
¿El tipo de fuente, efectos tipográficos, ancho de línea y alineación empleadas facilitan la lectura?	SI
¿Existe un alto contraste entre el color de fuente y el fondo?	SI
¿Incluyen las imágenes atributos 'alt' que describan su contenido?	SI
¿Es compatible el sitio web con los diferentes navegadores? ¿Se visualiza correctamente con diferentes resoluciones de pantalla? Se debe prestar atención a: <i>JScript, CSS, tablas, fuentes...</i>	SI

¿Puede el usuario disfrutar de todos los contenidos del sitio web sin necesidad de tener que descargar e instalar <i>plugins</i> adicionales?	SI
¿Se ha controlado el peso de la página? Se deben optimizar las imágenes, controlar el tamaño del código <i>JScript</i> ...	No aplica
¿Se puede imprimir la página sin problemas? Leer en pantalla es molesto, por lo que muchos usuarios preferirán imprimir las páginas para leerlas. Se debe asegurar que se puede imprimir la página (no salen partes cortadas), y que el resultado es legible.	No aplica
<u>Control y Retroalimentación</u>	
¿Tiene el usuario todo el control sobre el interfaz? Se debe evitar el uso de ventanas pop-up, ventanas que se abren a pantalla completa, banners intrusivos...	SI
¿Se informa constantemente al usuario acerca de lo que está pasando? Si el usuario tiene que esperar hasta que se termine una operación, se debe mostrar un mensaje indicándoselo y que debe esperar, con el tiempo de espera estimado o una barra de progreso.	SI
¿Se informa al usuario de lo que ha pasado? Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente.	SI
Cuando se produce un error, ¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema? Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error.	SI
¿Posee el usuario libertad para actuar? NO restringir la libertad del usuario: Uso de animaciones que no pueden ser "saltadas", páginas en las que desaparecen los botones de navegación, no impida al usuario poder usar el botón derecho de su ratón...	SI
¿Se ha controlado el tiempo de respuesta? Esto tiene que ver con el peso de cada página (accesibilidad) y tiene relación con el tiempo que tarda el servidor en finalizar una tarea y responder. El tiempo máximo que esperará un usuario son 10 segundos	SI
<u>Aclaraciones</u>	
¿Se ha evaluado adecuadamente la orientación del usuario? (Donde estoy, como volver, que he visitado, que va a pasar)	SI
¿Se ha usado correctamente la publicidad?	NO hay

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

9.1.1 Instalación de PostgreSQL

Explicaremos a continuación, el proceso de instalación de PostgreSQL, que es el SGBD usado por el sistema.

El primer paso es hacerse con el instalador de PostgreSQL 9.1, que es la versión que se ha usado en el sistema. Este instalador puede obtenerse del CD del proyecto (ver apartado 13.2 “Contenido Entregado en el CD-ROM”), o bien desde la URL <http://postgresql.org/download/>.

9.1.1.1 *Instalación*

Para instalar el SGBD ejecutamos el instalador, que nos mostrará una vista como la siguiente:



Figura 191: Instalación PostgreSQL

Pulsamos siguiente y vemos una pantalla donde podremos seleccionar el directorio de instalación, podemos dejar el que nos asigna por defecto.

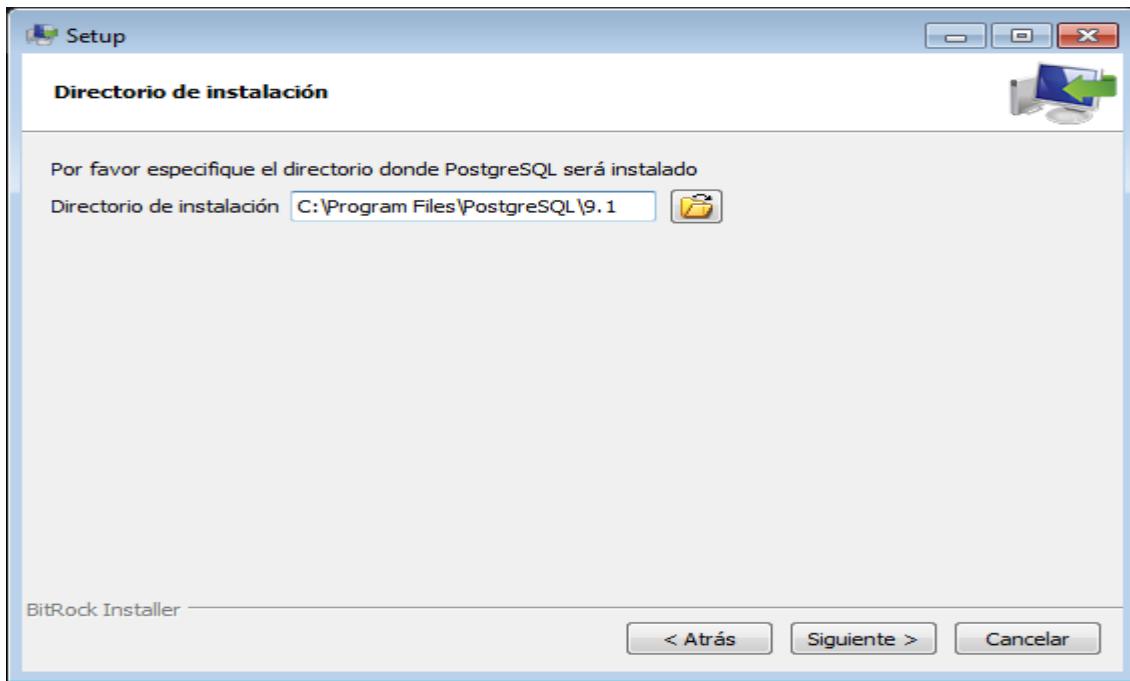


Figura 192: Instalación PostgreSQL – Directorio de instalación

A continuación nos pide que introduzcamos el directorio de datos, este puede ser conveniente cambiarlo, aunque se puede dejar también por defecto.

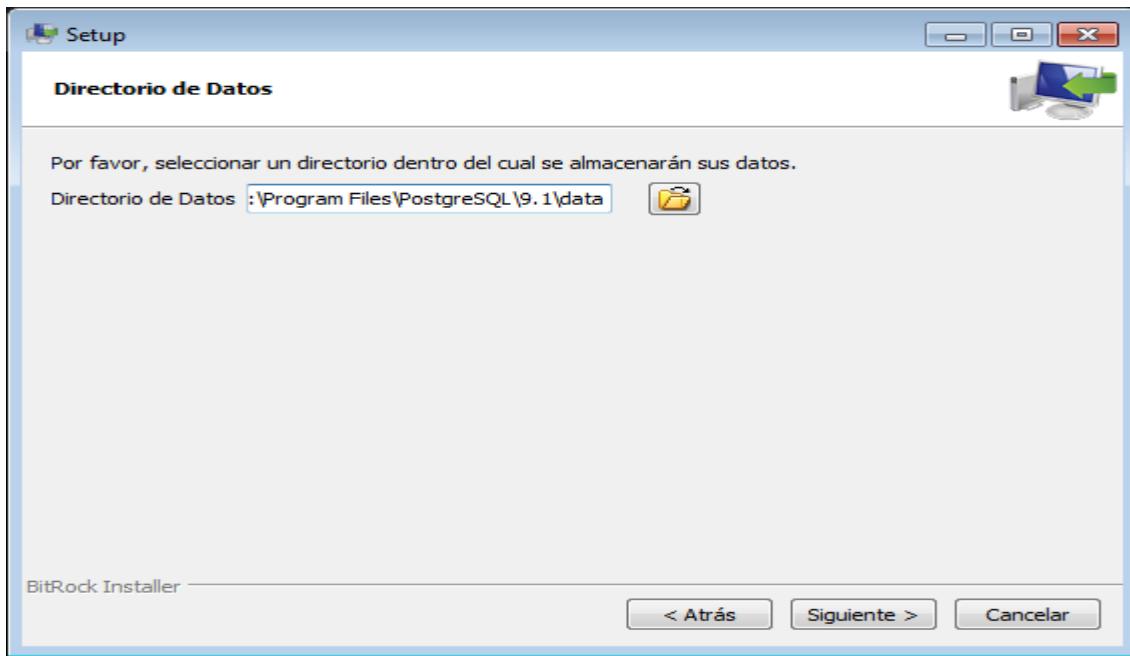


Figura 193 : Instalación PostgreSQL – Directorio de datos

Una vez completados estos pasos nos pedirá la contraseña de súper usuario, la introducimos y continuamos.

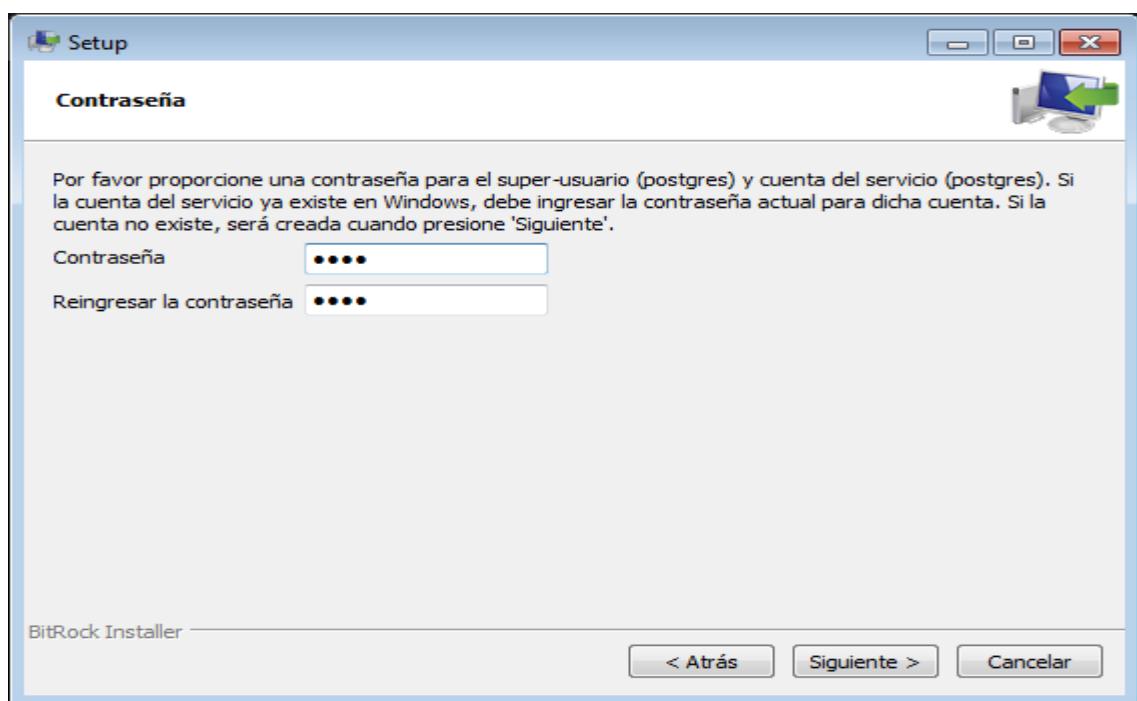


Figura 194: Instalación de PostgreSQL – Contraseña

Una vez aportada la contraseña debemos definir el puerto usado por el servicio del SGBD.

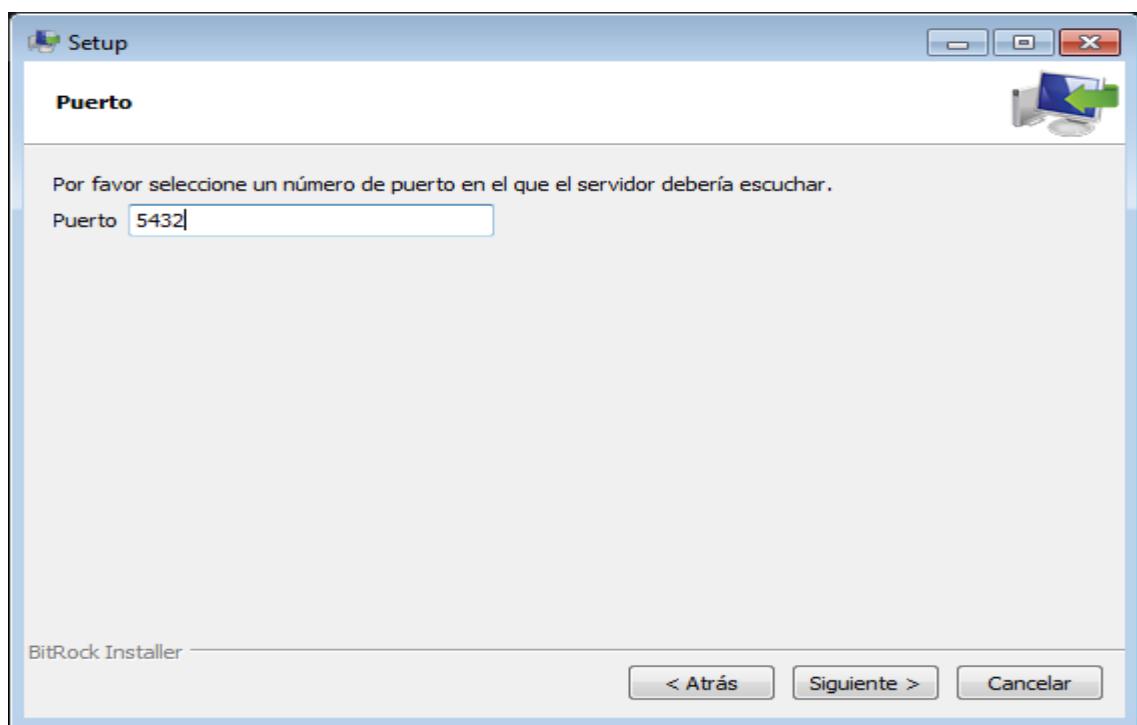


Figura 195: Instalación de PostgreSQL- Puerto

El siguiente paso al puerto es seleccionar la configuración regional.

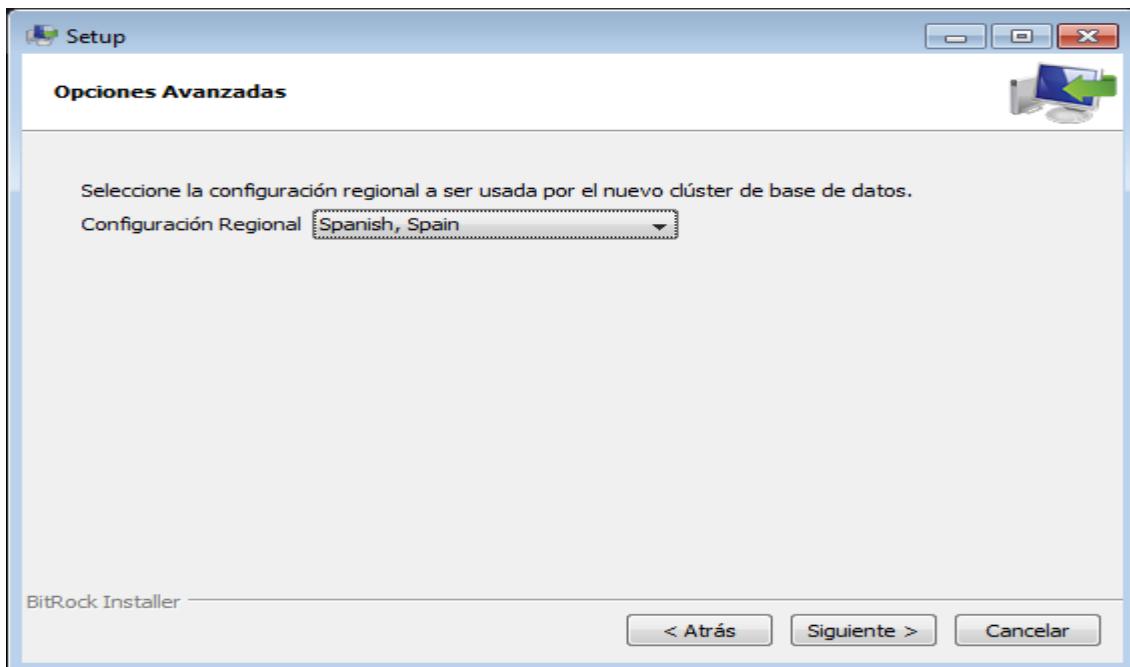


Figura 196: Instalación de PostgreSQL -Configuración

Se nos pedirá confirmación antes de instalar, pulsamos siguiente y procedemos.



Figura 197: Instalación de PostgreSQL - Confirmación

El proceso de instalación durará unos pocos minutos, se debería mostrar el progreso así como las diferentes bibliotecas dinámicas que se están instalando.

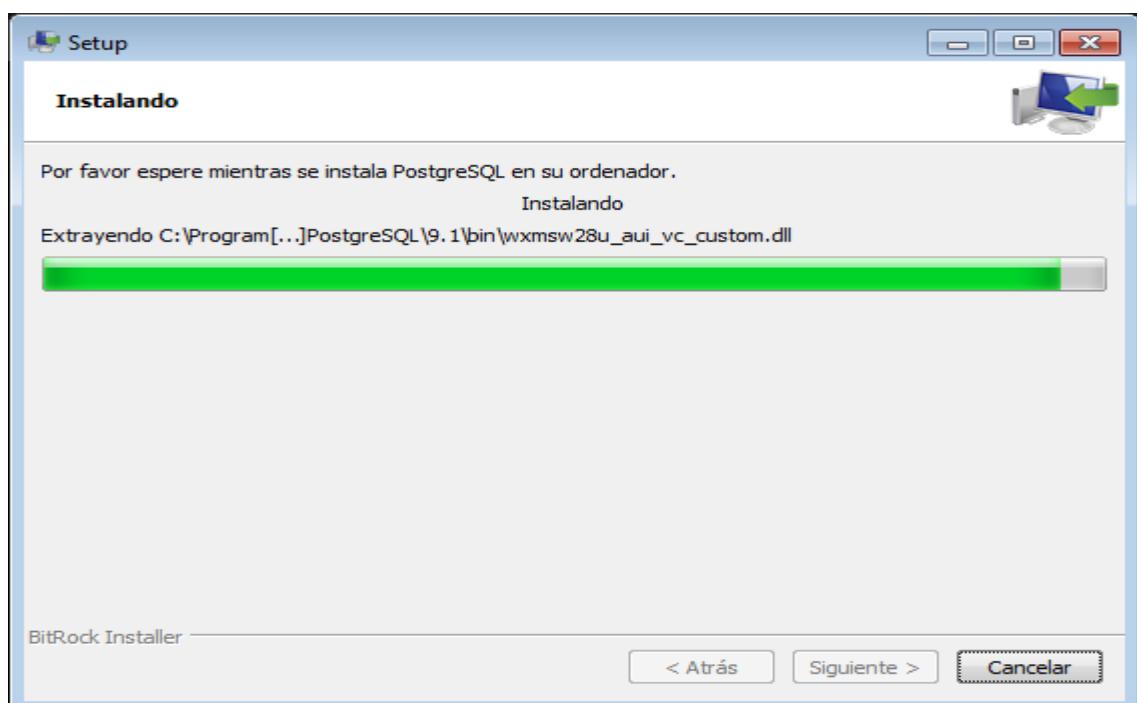


Figura 198: Instalación de PostgreSQL - Proceso

Una vez completado este proceso nos ofrecerá instalar aditamentos, que no necesitamos por el momento. Pulsamos finalizar y el SGBD está instalado.



Figura 199: Instalación de PosgreSQL - Fin

9.1.1.2 Configuración

Una vez instalado el SGBD, necesitamos realizar algunas configuraciones para que pueda funcionar.

En primer lugar necesitamos crear la base de datos del sistema dentro del SGBD, para ello ejecutamos el programa pgAdmin3 ubicado en el subdirectorio bin, dentro del directorio de instalación.

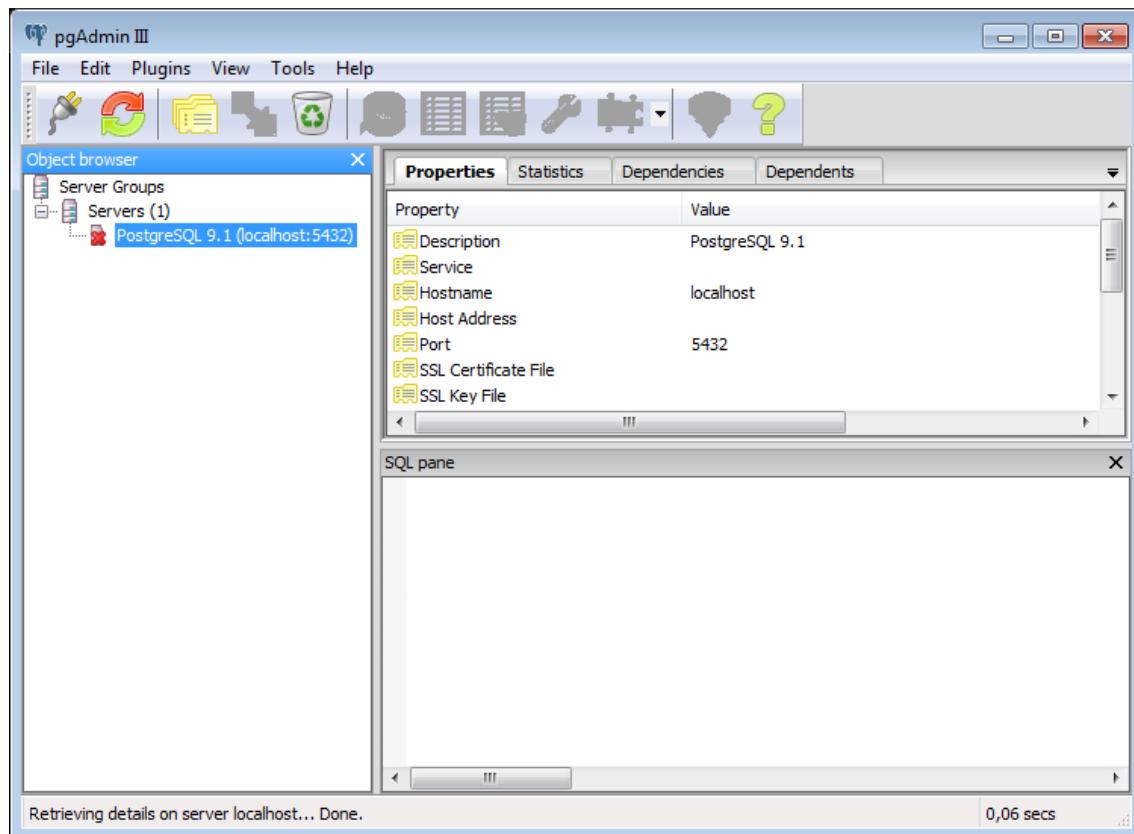


Figura 200: Configuración de PostgreSQL - PgAdmin3

Para acceder al servidor que se ha creado por defecto, el PgAdmin nos pedirá una contraseña, debemos introducir la que aportamos durante la instalación.

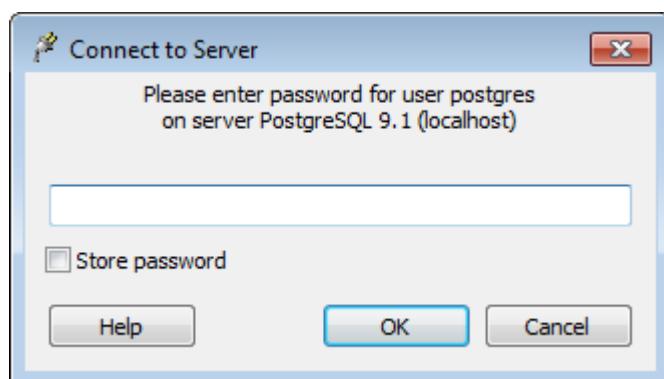


Figura 201: Configuración de PostgreSQL – Contraseña

Una vez que accedamos al servidor, veremos que existe una base de datos por defecto, es la base de datos de mantenimiento. Junto a ella crearemos la nuestra, pero antes debemos crear el usuario para esa base de datos, ya que no es apropiado usar el super usuario ni el usuario de mantenimiento. Para ello en el ícono Login Roles pulsamos botón derecho y seleccionamos nuevo Login Role.

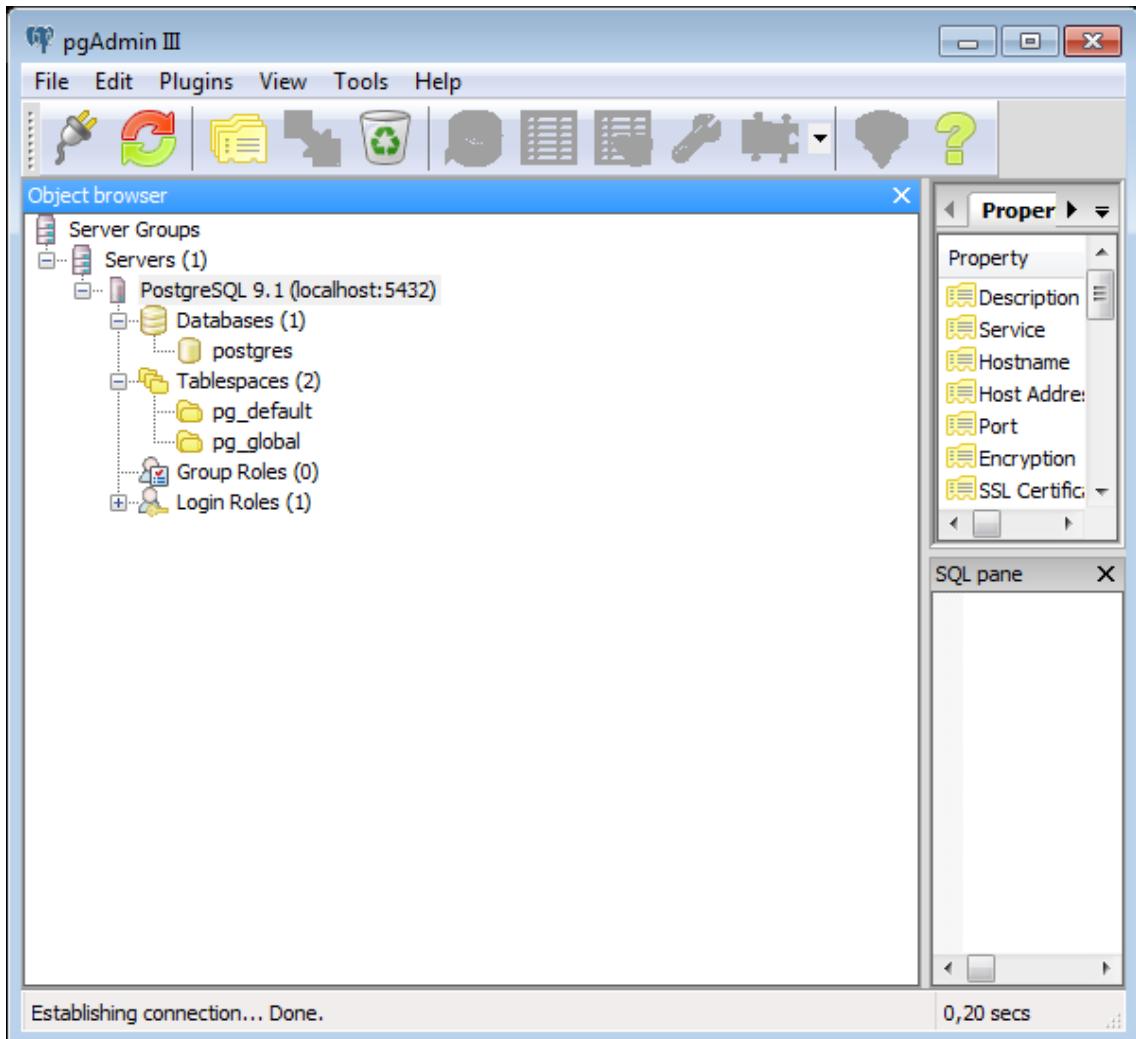


Figura 202: Configuración de PostgreSQL – PgAdmin iniciado

El nuevo rol se debe llamar cnpd y debe tener contraseña cnpd, de lo contrario se debe modificar las configuraciones de los clientes. Para establecer la contraseña debemos movernos a la pestaña de definition.

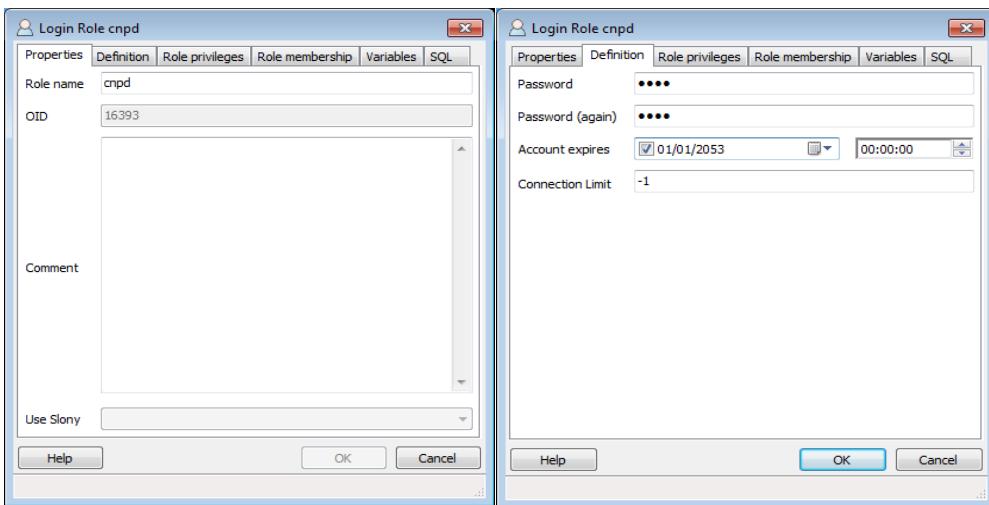


Figura 203: Configuración de PostgreSQL – Usuario y contraseña

Una vez tengamos el usuario, nos colocamos en el nodo del árbol llamado Databases, y en el menú contextual, seleccionamos New DataBase, podemos prescindir del resto de la configuración.

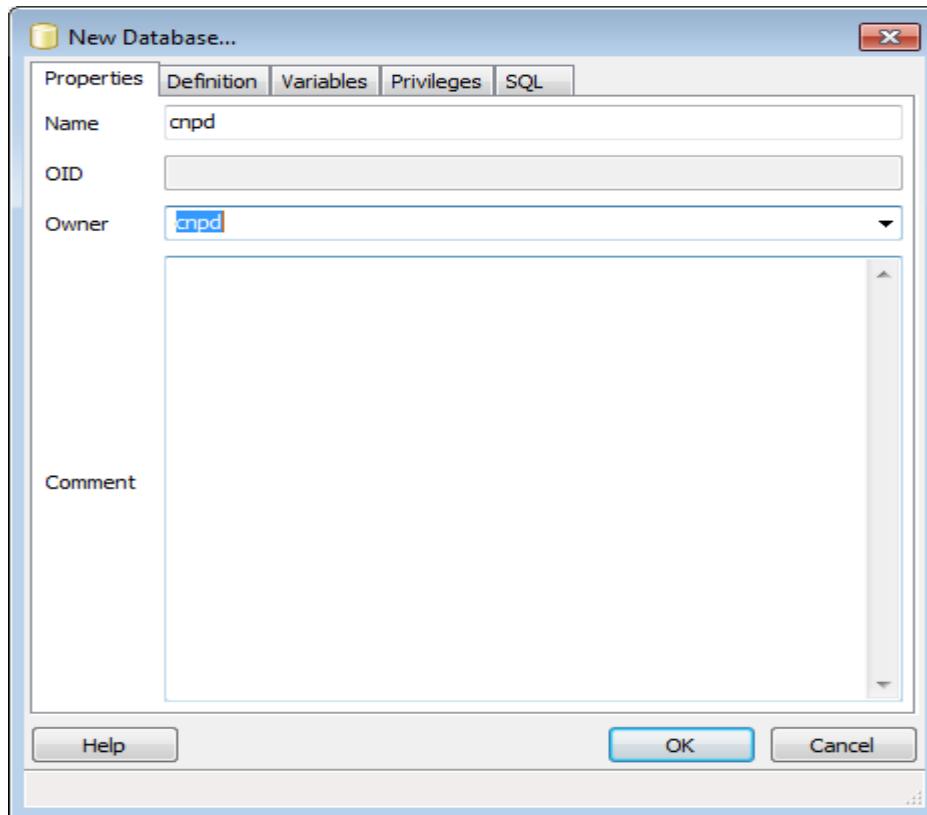


Figura 204: Configuración de PosgreSQL – Creación de la base de datos

En este punto, ya tenemos nuestra base de datos, esto es todo lo que necesitamos ya que el esquema será creado por Hibernate.

Nos queda un último detalle y es configurar el SGBD para que admita conexiones remotas.

El SGBD tiene toda su configuración en un fichero llamado postgresql.conf, que se encuentra ubicado en el directorio de datos, por defecto el directorio data dentro del directorio de instalación, la configuración de este fichero es correcta para la versión usada, aunque si en otras versiones. Otro fichero es el pg_hba.conf, en el que debemos tocar algunos detalles.

Para configurar el fichero desde PgAdmin seleccionamos el servidor en el árbol, seleccionamos la opción del menu superior Tools, allí Server Configuration y finalmente pg_hba.conf.

Por defecto la configuración del fichero es la siguiente:

Type	Database	User	IP-Address	Method	Option
<input checked="" type="checkbox"/>	host	all	all	127.0.0.1/32	md5
<input checked="" type="checkbox"/>	host	all	::1/128		md5
<input type="checkbox"/>	host	replication	postgres	127.0.0.1/32	md5
<input type="checkbox"/>	host	replication	postgres	::1/128	md5
<input type="checkbox"/>					

Configuration read from localhost

Figura 205: Configuración de PostgreSQL – Conexiones remotas 1

Nosotros necesitamos que quede de la siguiente forma:

Type	Database	User	IP-Address	Method	Option
<input checked="" type="checkbox"/>	host	all	all	0.0.0.0/0	trust
<input checked="" type="checkbox"/>	host	all	::1/128		md5
<input type="checkbox"/>	host	replication	postgres	127.0.0.1/32	md5
<input type="checkbox"/>	host	replication	postgres	::1/128	md5
<input type="checkbox"/>					

Configuration read from localhost

Nótese que se ha eliminado la encriptación, para simplificar la configuración de los clientes, aunque es más que conveniente que se use en producción. La modificación de la IP es absolutamente necesaria, de lo contrario solo podrán acceder a nuestro SGBD desde localhost.

Debemos asegurarnos de que el sistema permite las conexiones entrantes, en el caso de Windows debemos configurar el firewall para que permita conexiones de la red local.

9.1.2 Instalación de visual SVN Server

Visual SVN Server es una distribución de Apache Subversion que facilita la instalación y la configuración de Subversion. La versión usada ha sido 2.5, y puede ser descargada desde <http://visualsvn.com>. Descargamos el instalador y lo ejecutamos.

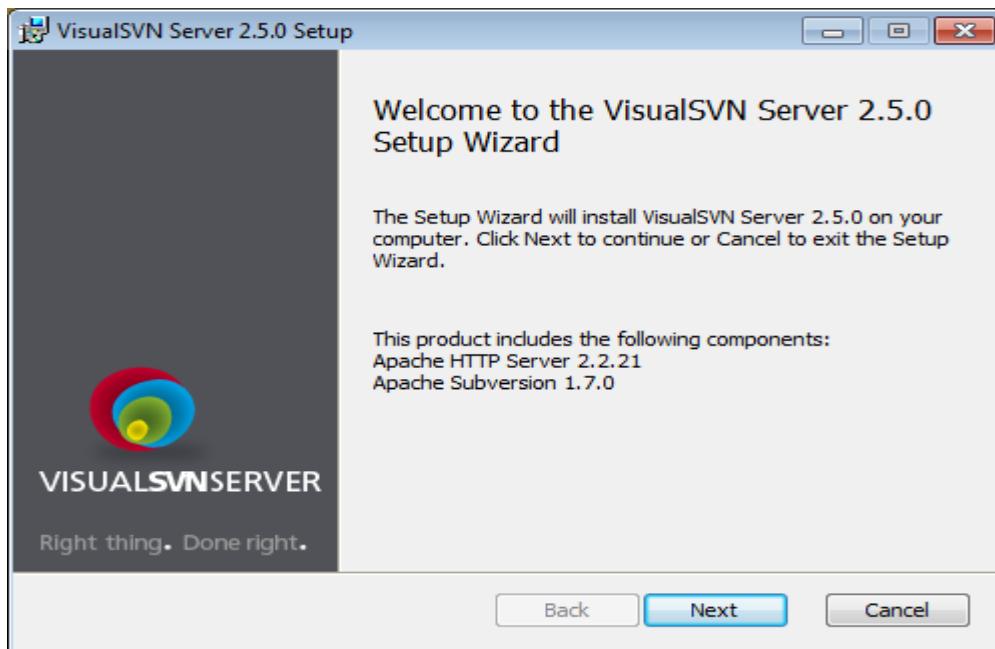


Figura 206: Instalación de Visual SVN Server

Después de aceptar la licencia, tendremos que seleccionar el tipo de instalación.

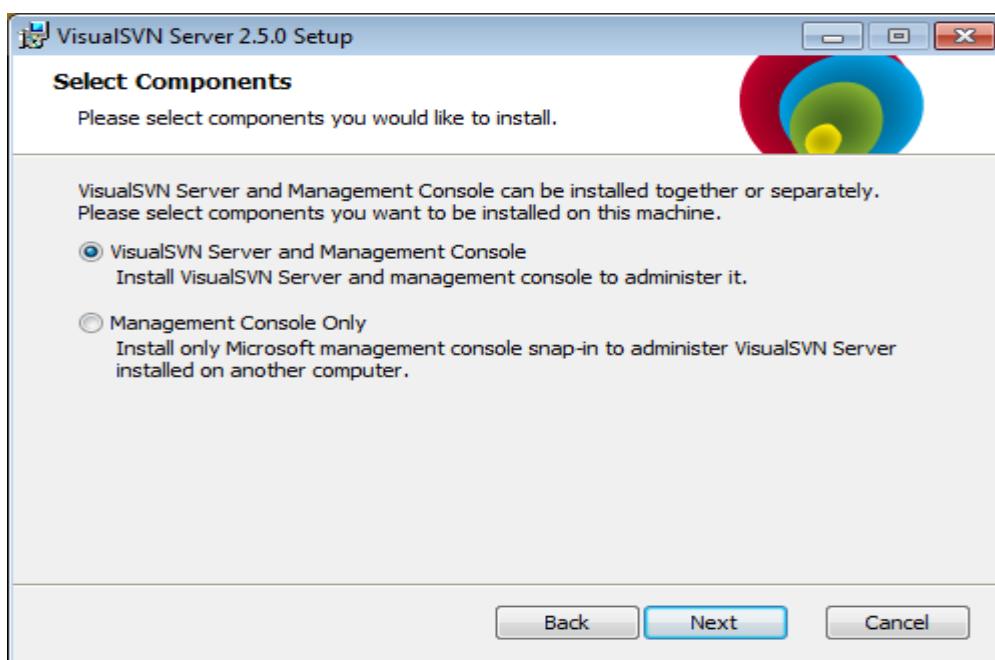


Figura 207: Instalación Visual SVN Server – Tipo de instalación

Establecemos a continuación algunos parámetros de funcionamiento.

- Directorio de instalación, seleccionamos el directorio donde se instalarán los componentes necesarios (SVN y HTTPD).
- Repositories, directorio donde se crearán los repositorios, en este directorio se guardarán los ficheros del sistema.
- ServerPort, seleccionamos el puerto y si se accederá mediante protocolo seguro.
- Por último seleccionamos el tipo de autenticación.

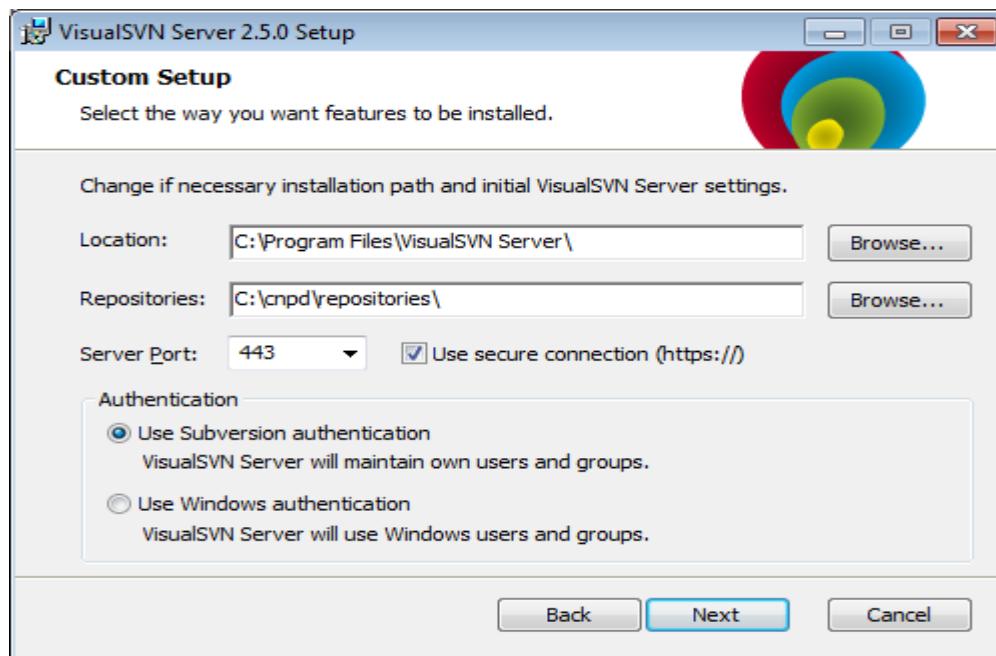


Figura 208: Instalación de Visual SVN Server – Configuración

Continuamos y finalizamos la instalación.

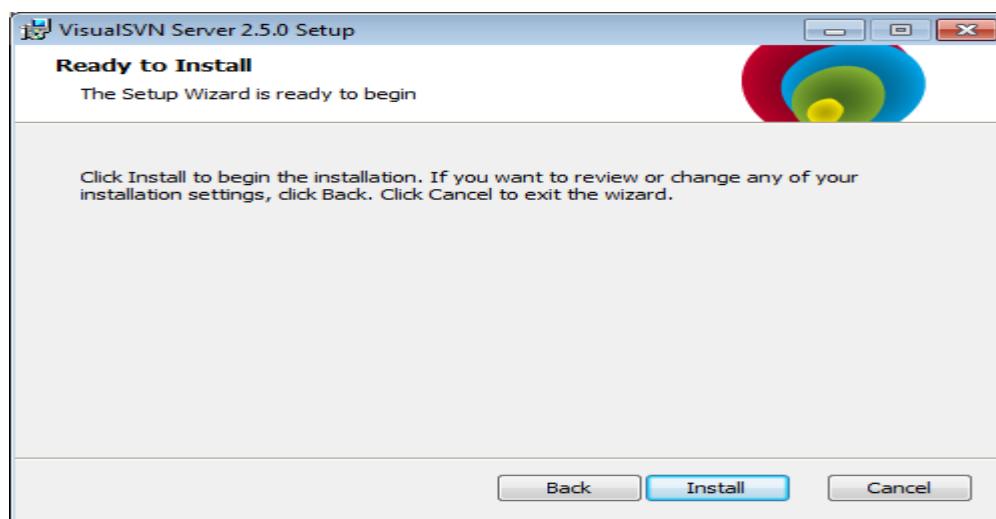


Figura 209: Instalación de Visual SVN Server – Fin

9.1.3 Instalación de JBoss AS7

Para este sistema se ha usado la versión 7.1.1 de JBoss AS, dependiendo de las funcionalidades requeridas, se han configurado de forma distinta las dos instancias. La distribución del AS7 1.1.1 puede ser descargada desde <http://www.jboss.org/jbossas/>. Además es necesario tener una JDK 1.6 instalada en el sistema.

9.1.3.1 Instalación de JBoss AS7 – Nivel de Negocios

Descargamos una instancia del servidor y la descomprimimos en un directorio del nivel de negocios. A continuación procedemos a configurar el servidor. El JBoss AS7 viene con 4 ficheros de configuración, uno para cada perfil de la plataforma con dos variantes para cada perfil, nosotros usaremos el fichero standalone-full.xml.

9.1.3.1.1 Configurar interfaces de red

Para que el servidor este accesible desde el exterior es necesario configurar la sección de interfaces del fichero de configuración de la siguiente forma.

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:169.254.17.118}" />
    </interface>
    <interface name="public">
        <inet-address value="${jboss.bind.address:169.254.17.118}" />
    </interface>
    <interface name="unsecure">
        <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}" />
    </interface>
</interfaces>
```

Figura 210: Instalación AS7 – Interfaces de Red

Esta configuración usa como interfaz de red la dirección configurada si no se da un parámetro con el nombre adecuado, en este caso se ha configurado para esta accesible desde la dirección 169.254.17.118, salvo que se pase un parámetro con el nombre que va delante de los dos puntos. La dirección debe ser la de la maquina donde se ejecutará el servidor.

9.1.3.1.2 Configurar módulos

Para que el sistema funcione correctamente han sido añadidos algunos módulos al JBoss AS7. En el CD se han incluido dichos módulos.

En el nivel de negocio es necesario añadir los módulos siguientes:

- org.antlr.runtime
- org.postgres

- org.tmatesoft.sqljet
- org.tmatesoft.svn

Estos módulos se encuentran en un directorio del CD tal y como deberían ir dentro del directorio modules del servidor, para añadirlos se deben fusionar las dos carpetas modules, esto si ya hay una carpeta org, tomamos lo que hay dentro de org, y lo copiamos a ella, si lo que hubiese dentro de org, fuese apache, y también estuviese la carpeta apache en el destino, repetimos.

9.1.3.1.3 Configurar Datasources

Es necesario configurar un origen de datos, para la base de datos del sistema. Para ello primero se debe definir el driver, usando el módulo añadido previamente, la sección de configuración de drivers luce de la siguiente forma:

```
<drivers>
  <driver name="postgres" module="org.postgres">
    <xa-datasource-class>org.postgresql.xa.PGXADatasource</xa-datasource-class>
  </driver>
  <driver name="h2" module="com.h2database.h2">
    <xa-datasource-class>org.h2.jdbc.JdbcDataSource</xa-datasource-class>
  </driver>
</drivers>
```

Figura 211: Instalación AS7 – Driver PostgreSQL

El driver que viene configurado por defecto para Hypersonic se puede y se recomienda eliminar.

Una vez configurado el driver debemos añadir el origen de datos, para ello se debe añadir el siguiente fragmento. La dirección debe ser la del servidor de datos y se debe prestar atención que el nombre de la base de datos, el usuario y la contraseña, coincidan con los creados en el SGBD.

```
<subsystem xmlns="urn:jboss:domain:datasources:1.0">
  <datasources>
    <datasource jta="true" jndi-name="java:jboss/datasources/CNPD_DS" pool-name="CNPD_DS_Pool"
      enabled="true" use-java-context="true" use-ccm="true">
      <connection-url>jdbc:postgresql://localhost:5432/cnpd</connection-url>
      <driver>postgres</driver>
      <security>
        <user-name>cnpd</user-name>
        <password>cnpd</password>
      </security>
    </datasource>
  </datasources>
</subsystem>
```

Figura 212: Instalación AS7 – Datasource

9.1.3.1.4 Configurar Java Mail

Por último se debe configurar el sub sistema de Mail para poder obtener sesiones del contenedor en nuestra aplicación. Para ello se debe añadir una sesión al subsistema de Mail, la siguiente captura muestra la sesión añadida (la segunda). La sesión que viene por defecto puede ser eliminada.

```
<subsystem xmlns="urn:jboss:domain:mail:1.0">
    <mail-session jndi-name="java:jboss/mail/Default">
        <smtp-server outbound-socket-binding-ref="mail-smtp"/>
    </mail-session>
    <mail-session jndi-name="java:jboss/mail/gmail">
        <smtp-server ssl="true" outbound-socket-binding-ref="gmail-smtp">
            <login name="cnpd.manager@gmail.com" password="cnpd.cnpd"/>
        </smtp-server>
    </mail-session>
```

Figura 213: Instalación AS7 – Java Mail

Además es necesario configurar la conexión de salida para el servidor SMTP de Gmail, esto se hace en la sección de *socket-binding-group*, al final del fichero.

```
<outbound-socket-binding name="gmail-smtp">
    <remote-destination host="smtp.gmail.com" port="465"/>
</outbound-socket-binding>
</socket-binding-group>
```

Figura 214: Instalación AS7 – Conexión Gmail

9.1.3.1.5 Reduciendo subsistemas

Finalmente conviene eliminar algunos subsistemas que no serán utilizados, así como las extensiones que los implementan, esto en el AS7 equivale a eliminar la funcionalidad por completo, será como si nunca la hubiese tenido. Concretamente eliminaremos los subsistemas de servicios webs, mensajería, y escáner de despliegues, que no son usados por el sistema, además se eliminarán las extensiones (módulos del as7 que dan soporte algún subsistema) que los soportan.

9.1.3.1.6 Realems de Seguridad

Para finalizar se deben establecer Realems que permitan realizar los despliegues y que el contenedor del nivel web se comunique con el de negocio. Para ello usamos la herramienta add-user que se encuentra en el directorio bin del servidor. Para evitar errores se han configurado un usuario (cnpd, secret) en los dos servidores para los dos Realems.

9.1.3.2 Instalación de JBoss AS7 – Nivel Web

Descargamos una instancia del servidor y la descomprimimos en un directorio del nivel de negocios. Este nivel también requiere la configuración de las interfaces de red y conviene hacer una limpieza de subsistemas inútiles, además tiene algunas particularidades.

9.1.3.2.1 Configurar conexiones remotas

Con el fin de que el servidor pueda establecer conexiones remotas con el servidor del nivel de negocio, debemos añadir la siguiente configuración al subsistema remoting.

```
<subsystem xmlns="urn:jboss:domain:remoting:1.1">
    <connector name="remoting-connector" socket-binding="remoting" security-realm="ApplicationRealm"/>
    <outbound-connections>
        <remote-outbound-connection name="remote-ejb-connection" outbound-socket-binding-ref="remote-ejb"
            username="cnpd" security-realm="ejb-security-realm">
            <properties>
                <property name="SASL_POLICY_NOANONYMOUS" value="false"/>
                <property name="SSL_ENABLED" value="false"/>
            </properties>
        </remote-outbound-connection>
    </outbound-connections>
</subsystem>
```

Figura 215: Instalación AS7 – Subsistema remoting

Es muy importante destacar que el usuario cnpd (u otro), debe estar creado en el ApplicationRealem.

Además hay que añadir la conexión de salida en la sección de *socket-binding-group*.

```
<outbound-socket-binding name="remote-ejb">
    <remote-destination host="localhost" port="4547"/>
</outbound-socket-binding>
```

Figura 216: Instalación AS7 – Conexión con negocio

Obviamente la dirección de esa conexión debe ser la del nivel de negocio. Esta configuración es referenciada desde la aplicación web, mediante el fichero jboss-ejb-client.xml. Finalmente debemos añadir un Realem especial para la autenticación entre un servidor y otro, la siguiente imagen muestra el Realem añadido, con contraseña secret y encriptado MD5.

El nombre de usuario definido en remoting y la contraseña que se defina en este Realem nuevo, deben estar en el ApplicationRealem del servidor de negocio.

```
<security-realms>
    <security-realm name="ManagementRealm">
        <authentication>
            <properties path="mgmt-users.properties" relative-to="jboss.server.config.dir"/>
        </authentication>
    </security-realm>
    <security-realm name="ApplicationRealm">
        <authentication>
            <properties path="application-users.properties" relative-to="jboss.server.config.dir"/>
        </authentication>
    </security-realm>
    <security-realm name="ejb-security-realm">
        <server-identities>
            <secret value="c2VjcmV0"/>
        </server-identities>
    </security-realm>
</security-realms>
```

Figura 217: Instalación AS7 – Seguridad Conexión con Negocio

9.1.3.2.2 Configurar módulos

En el nivel web es necesario añadir los mismos módulos que se añadieron en el nivel de negocio, aunque estos podrían ir en el artefacto de despliegue, se ha decidido que estén como módulos del AS7. Además se ha configurado otro módulo para ejecutar las pruebas unitarias en el contenedor.

9.1.4 Instalación de las aplicaciones

Para instalar las aplicaciones se usará la aplicación web provista por el servidor para ello, en ambos casos. La aplicación nos pedirá autenticarnos mediante HTTP Basic, el usuario y contraseña deben estar en el ManagementRealm, en este caso se ha añadido (cnpd, secret).

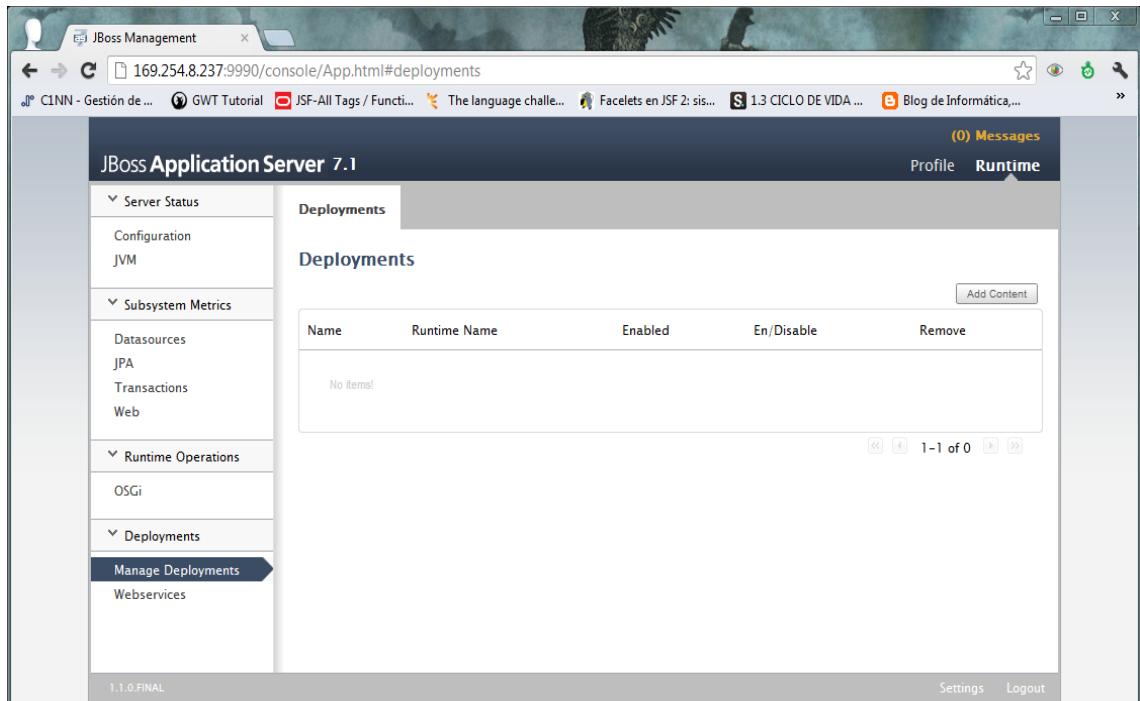


Figura 218: Aplicación de gestión de JBossAS7

- **Instalación de Negocio:** Para instalar el nivel de negocio, se debe desplegar el fichero cnpd-business.jar del CD, accediendo a la aplicación de gestión del nivel de negocio.
- **Instalación de Web:** Para instalar el nivel web se debe desplegar el fichero cnpd-web.jar aportado en el CD, accediendo a la aplicación de gestión del AS7 instalado en el nivel web.
- **Instalación de Cliente:** Basta con ejecutar el fichero cnpd-client.jar aportado en el CD, desde un ordenador que se encuentre en la misma red local que la capa de negocio.

9.2 Manual de Usuario

Este manual es la guía de cómo usar la aplicación, contiene todo lo que debería saber un usuario para obtener un buen provecho de la aplicación.

Al acceder a la aplicación se le dará a usuario la posibilidad de registrarse o acceder al sistema.

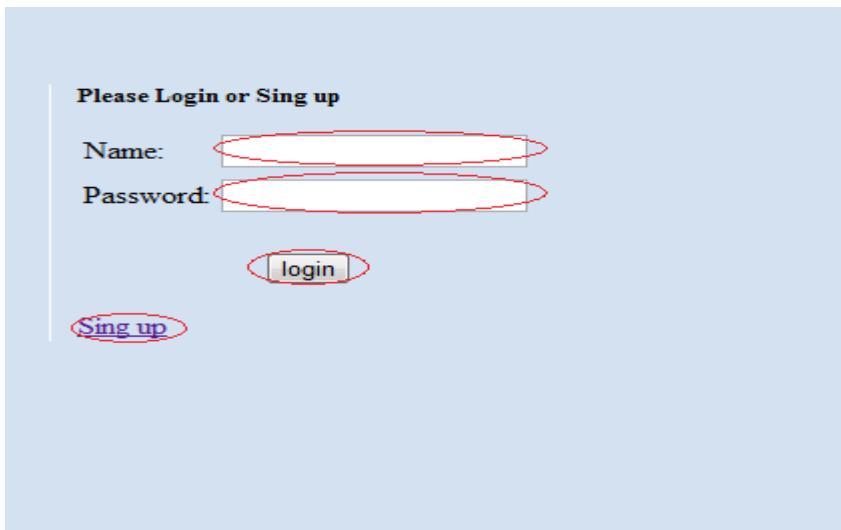


Figura 219: Manual de Usuario- Pantalla Inicial

9.2.1 Registrarse en la Aplicación

Para acceder a la aplicación posteriormente el usuario deberá realizar un proceso de registro, que consiste en llenar una serie de formularios aportando distintos tipos de información.

El primero de estos formularios le pediría al usuario una cuenta de Google (Figura 220).

A screenshot of a registration form titled "Google Information". It contains three input fields: "Google account" with the value "juan@gmail.com", "Google password" with a masked value, and "Repeat google Password" with a masked value. At the bottom right are "Cancel" and "Continue" buttons.

Figura 220: Manual de Usuario – Registrarse en la Aplicación I

Este formulario comprobará que el usuario introducido sea una cuenta de Google valida y que no hay un usuario dado ya de alta en el sistema con ella. Si el usuario lo rellena de forma correcta, podrá pasar al siguiente formulario. Se debe completar un formulario para datos personales, otros dos para datos académicos y profesionales y finalmente uno para enviar peticiones a los contactos de Google Contacts.

9.2.2 Rasgos Generales

Una vez dentro de la aplicación, el usuario se encontrará con tres zonas, un panel izquierdo que agrupa todo lo relativo a su perfil, un panel derecho que contiene el chat de la aplicación y un panel central que muestra en función de la pestaña seleccionada una sección u otra de la aplicación.

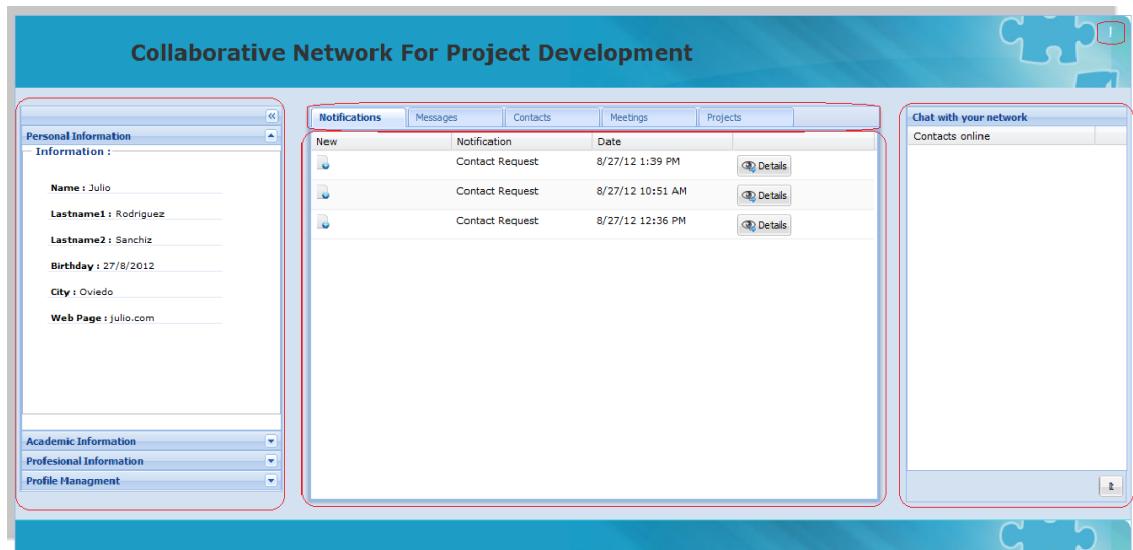


Figura 221: Manual de Usuario – Rasgos Generales

9.2.3 Perfil del Usuario

El panel de información contiene además de toda la información, un apartado que permite modificarla. Para ello se le muestra un árbol con las opciones de modificación del perfil agrupadas por categorías (Figura 222). Para ejecutar estas opciones el usuario debe hacer doble click sobre ellas, lo que le mostrará un dialogo para aportar la nueva información.

La modificación de la información está sometida a las mismas restricciones que el proceso de registro, por lo que no se podrá aportar una cuenta de Google inválida y no se podrá dejar ningún campo en sin aportar.

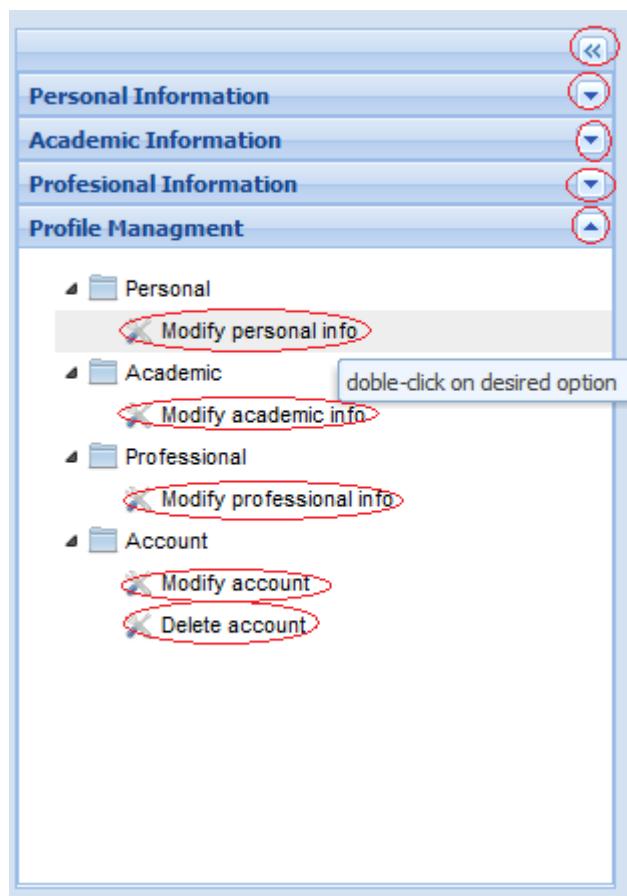


Figura 222: Manual de Usuario - Perfil del Usuario

9.2.4 Chat de la Aplicación

El panel derecho de la aplicación contiene un listado con los contactos conectados al sistema, ofreciendo la posibilidad de iniciar conversaciones con todos ellos. Para las conversaciones se abrirá una ventana que tendrá una pestaña por cada conversación, estas pestañas se podrán cerrar, y se podrán añadir nuevas pulsando en el botón chat de nuevos usuarios.

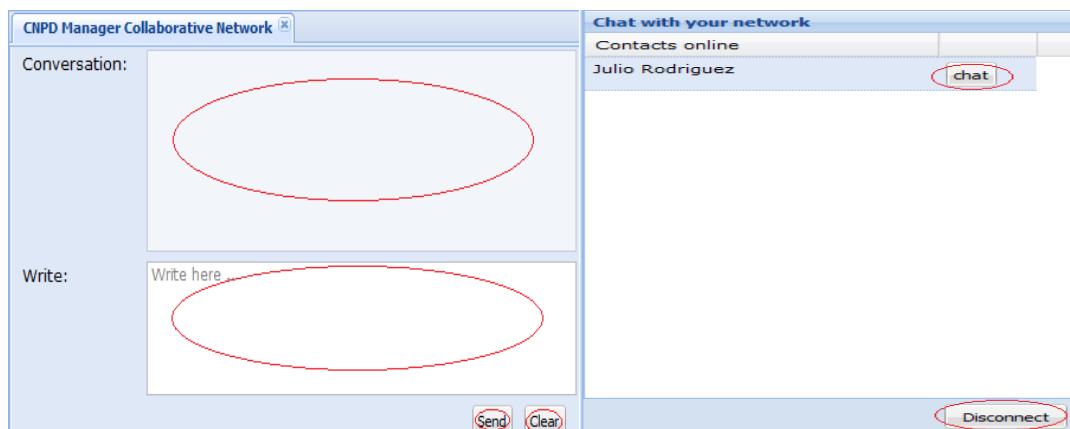


Figura 223: Manual de Usuario – Chat de la aplicación

9.2.5 Consultar Notificaciones

Sobre la zona central de la aplicación, la primera pestaña nos mostrará un listado de notificaciones. Pulsando en el botón de detalles de una notificación nos muestra un dialogo con los detalles, que podemos aceptar, marcando así la notificación como notificada o cancelar, dejando esta tarea para más adelante.

En cualquier caso tanto las notificaciones como los detalles de las notificaciones son meramente informáticos, no hay acciones que realizar sobre una notificación.

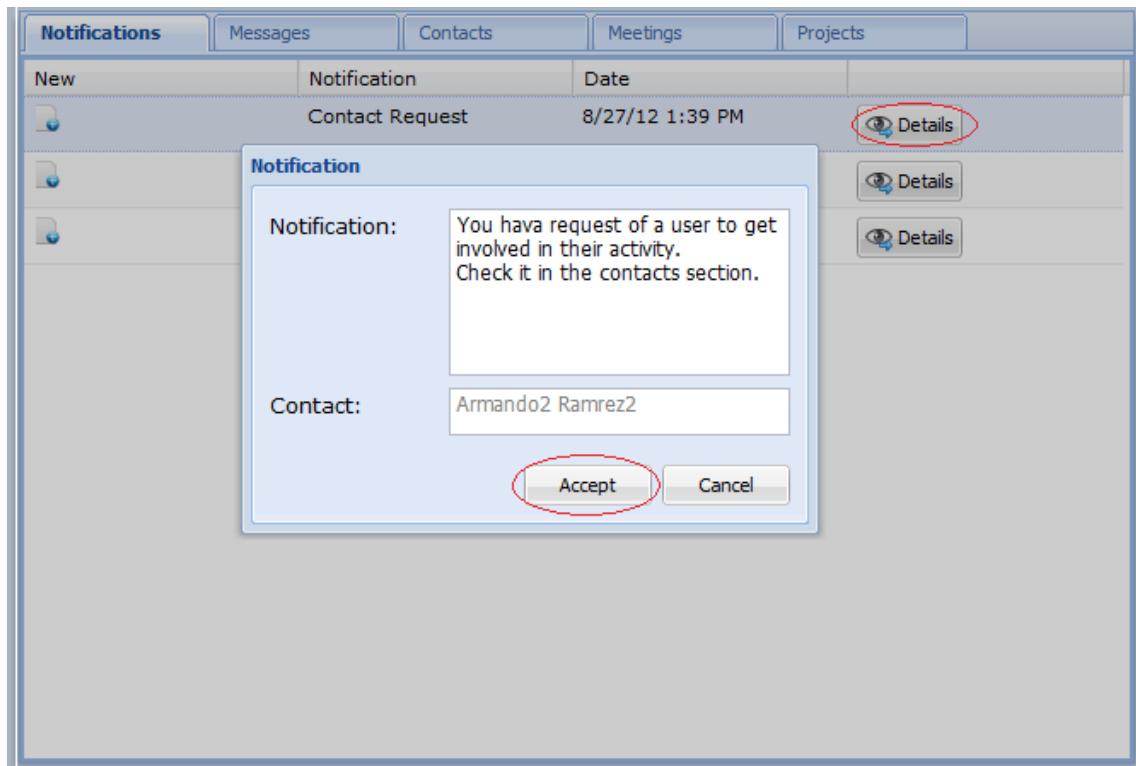


Figura 224: Manual de Usuario – Consultar Notificaciones

9.2.6 Consultar Mensajes

La segunda pestaña de la zona central mostrará al usuario un listado de los mensajes que le han enviado (Figura 225), indicando si estos han sido leídos, la fecha en que fueron enviados, y ofreciendo un conjunto de opciones.

Para ver las opciones el usuario debe pulsar el botón Select, que le permite seleccionar la acción que a este le interesa.

Las acciones son eliminar el mensaje, esto hará que desaparezca del listado ipso facto y verlo.

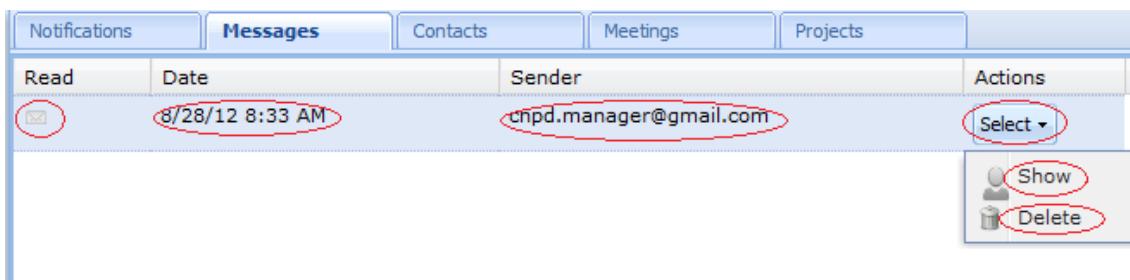


Figura 225: Manual de Usuario – Consultar Mensajes

Seleccionando la opción de ver mensaje, el usuario podrá por supuesto ver el mensaje, y además contestarlo (Figura 226).

El usuario podrá copiar el texto del mensaje recibido e incluso modificar su formato, pero estos cambios no persistirán.

El usuario podrá escribir la respuesta dándole formato y este formato, sí le llegará al usuario destinatario del mensaje.

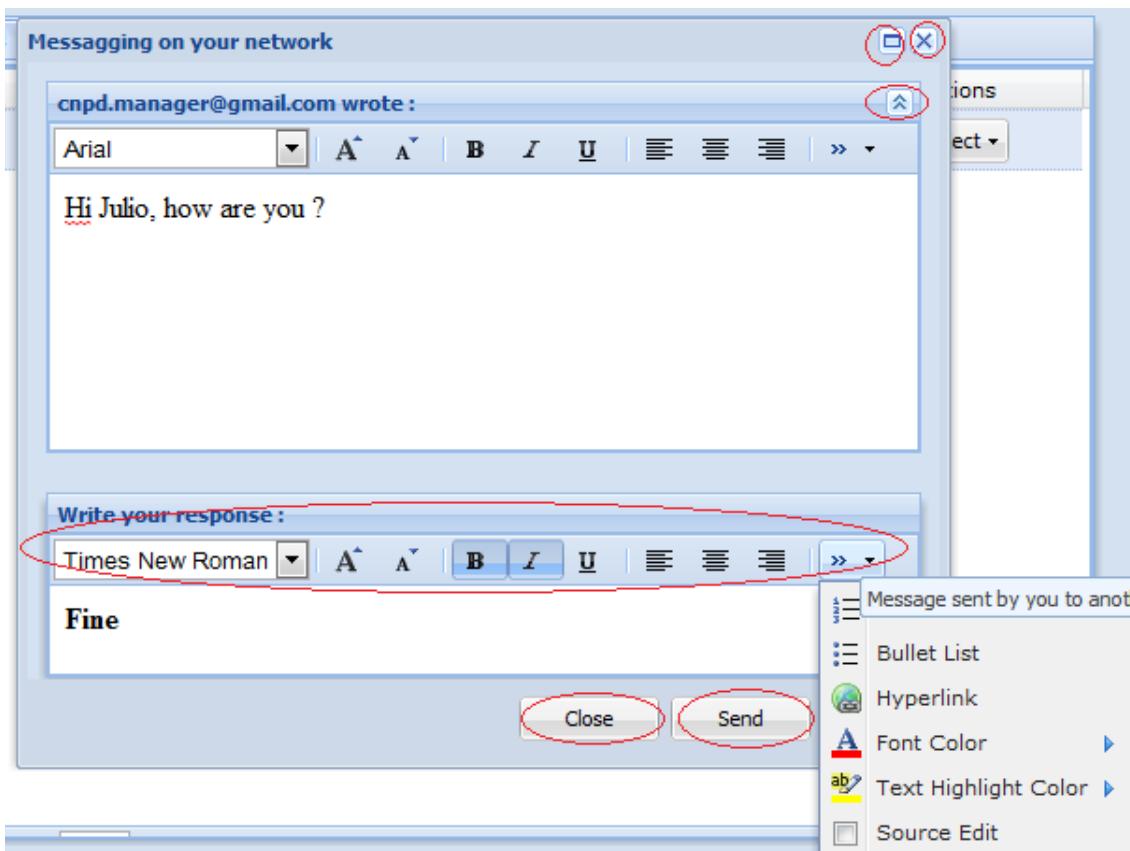


Figura 226: Consultar Mensajes – Responder Mensajes

9.2.7 Contactos

La tercera pestaña de la zona central, contienen todo lo relativo a los contactos del usuario, es la sección a la que tiene que acudir el usuario para realizar cualquier acción relativa a los contactos. En esta sección se pueden consultar los contactos, las peticiones de contacto y buscar contactos.

	Name	Lastname1	Lastname2	Actions
	Julio	Rodriguez	Sanchiz	Select ▾

Figura 227: Manual de Usuario –Contactos

9.2.7.1 Consultar Contactos

La primera de estas pestañas contiene un listado con todos los contactos del usuario (Figura 228), este listado permite ejecutar un conjunto de acciones, como visitar un contacto, borrarlo del listado o enviarle un mensaje.

	Name	Lastname1	Lastname2	Actions
	Julio	Rodriguez	Sanchiz	Select ▾

Figura 228: Manual de Usuario – Consultar Contactos

9.2.7.2 Consultar Peticiones

En otra subsección se pueden consultar las peticiones, que nos han hecho el resto de los contactos. Aquí tendremos dos opciones sobre cada petición, aceptarla o rechazarla.

Al aceptar una petición tendremos al contacto ipso facto en el listado de contactos, y desaparecerá del listado de peticiones.

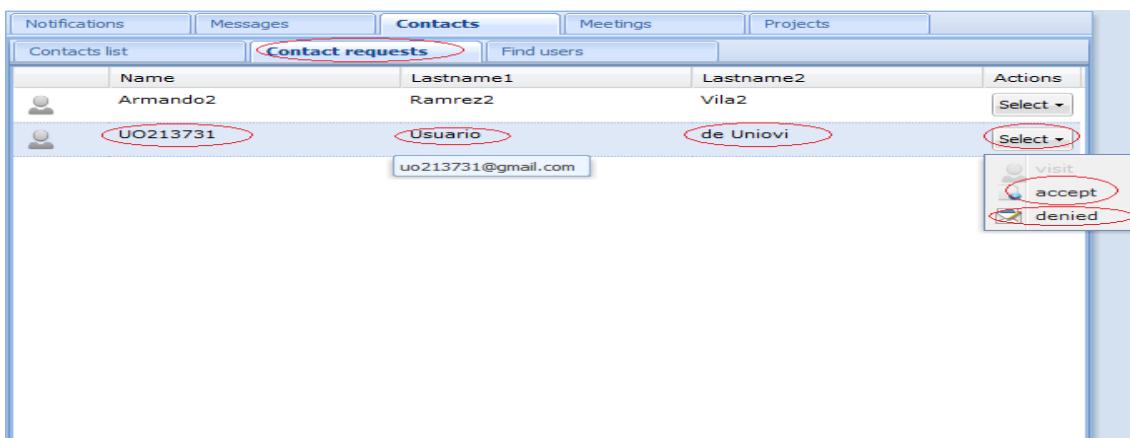


Figura 229: Manual de Usuario – Consultar Peticiones

9.2.7.3 Buscar Usuarios

Por último en tendremos una subsección para buscar usuarios, introduciendo su nombre, primer apellido y segundo apellido. Podremos dejar cualquier campo en blanco, y este no será usado para la búsqueda. Si introducimos Armando2, obtendremos todos los contactos que tengan ese nombre y cualquier pareja de apellidos, si añadimos un apellido acotaremos la búsqueda a los que tengan ese nombre y ese apellido. El listado de resultados nos permitirá visitar al usuario si es un contacto o hacerle una petición si no lo es.

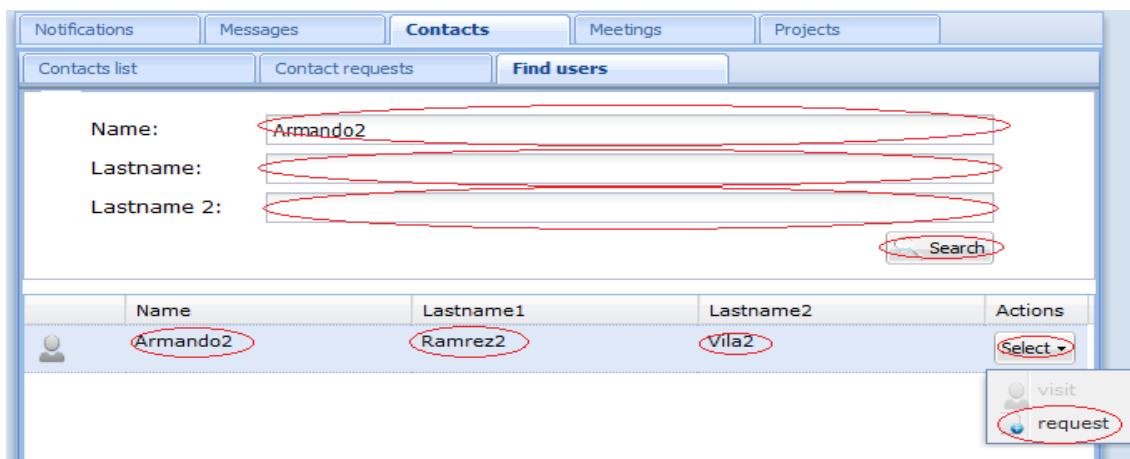


Figura 230: Manual de Usuario – Buscar Usuarios

9.2.8 Reuniones

La cuarta pestaña de la zona central es la sección de reuniones, esta sección contiene todo lo relativo a las reuniones, un listado con las reuniones, un listado con invitaciones a reuniones y una sub sección para la creación de reuniones.

9.2.8.1 Consultar Reuniones

El listado de reuniones muestra información de la reunión y permite ver sus detalles o abandonarla.

Celebrated	Title	Date	Actions
	Project Startup	March 15, 2013	Select

Figura 231: Manual de Usuario – Consultar Reuniones

Seleccionando la opción de ver los detalles iremos a una vista de detalles de la reunión (Figura 232), allí tendremos la información de la reunión y los participantes, así como opciones de eliminar reunión o invitar más participantes, en el caso de que el usuario sea el creador.

Los usuarios no creadores no tendrán las opciones marcadas en rojo en la Figura 232, de forma que solo podrán ver la información de la reunión.

Figura 232: Manual de Usuario – Detalles de Reunión

9.2.8.2 Crear Nueva Reunión

Finalmente podremos crear una nueva reunión completando un formulario e invitando posteriormente a los contactos que veamos oportunos (Figura 234).

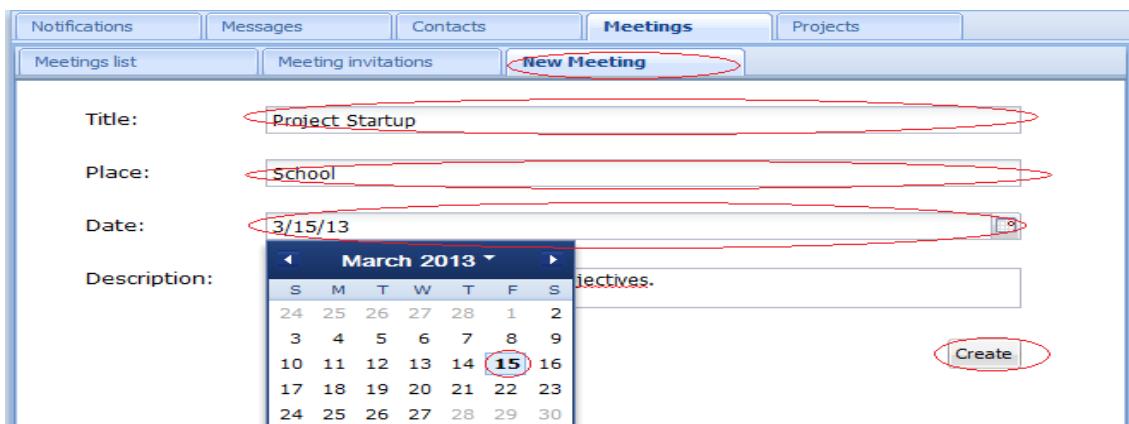


Figura 233: Manual de Usuario – Crear Nueva Reunión

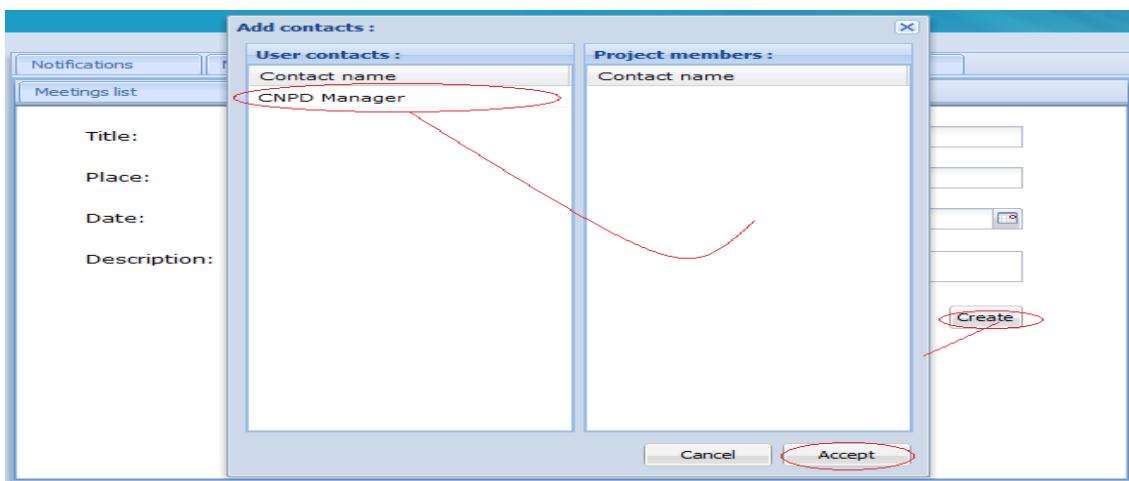


Figura 234: Manual de Usuario – Crear Nueva Reunión 2

9.2.9 Proyectos

La última pestaña de la zona central es la sección de proyectos, para todo lo relativo a sus proyectos el usuario deberá acudir a esta pestaña, donde podrá listarlos, ver las invitaciones a proyectos, buscar proyectos y crear nuevos proyectos.

9.2.9.1 Consultar proyectos

En esta sección el usuario verá un listado de sus proyectos, análogo al resto de listados, tendrá opciones relativas a los proyectos, en este caso abrir y abandonar.

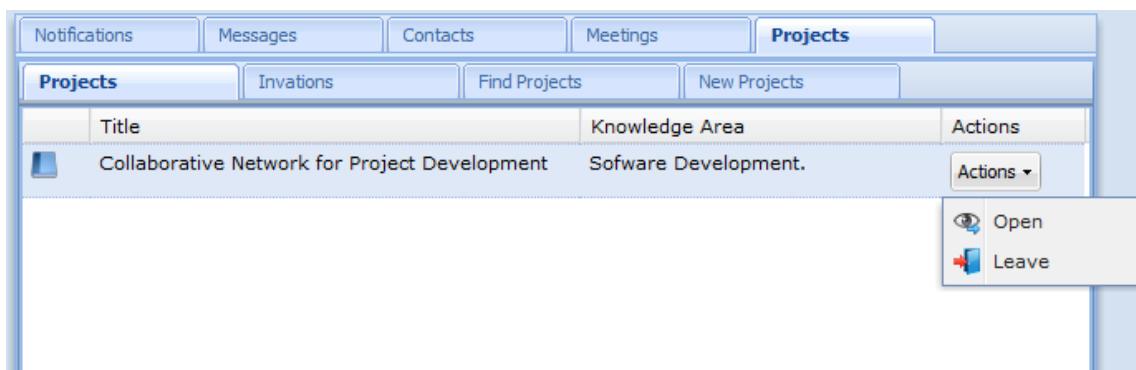


Figura 235: Manual de Usuario – Consulta de Proyectos

9.2.9.2 Crear Proyecto

Para crear un proyecto el usuario debe completar el formulario y pulsar el botón crear, entonces se le dará la opción de invitar contactos, lo cual no es obligatorio, pulsando en aceptar creará el proyecto, pulsando en cancelar volverá al formulario de creación.

The screenshot shows a 'New Projects' creation form. It includes fields for Knowledge area (highlighted with a red oval), Title (filled with 'Collaborative Network for Project Development'), and Description (filled with 'A network to develop projects with another people.' and a note 'Description of the new project'). A 'Create' button is at the bottom right, also highlighted with a red oval.

Figura 236: Manual de Usuario Crear Proyecto

9.2.10 Proyecto

Seleccionando la opción de abrir proyecto el usuario verá la sección del proyecto (Figura 236), esta sección es una ventana independiente, que tiene tres zonas principales, un panel izquierdo para la administración del proyecto, un panel derecho con el chat del proyecto y una zona central con pestañas para cada una de las secciones del proyecto.

El panel de administración y gestión del proyecto, cuenta con opciones en forma de árbol, análogo al panel de gestión del perfil, pero con opciones sobre el proyecto, desde aquí se podrá publicar, eliminar y modificar un proyecto.

El panel de chat funciona exactamente igual que el chat de la aplicación pero solo muestra los participantes del proyecto que estén conectados.

Figura 237: Manual de Usuario - Proyecto

9.2.10.1 Recursos

La segunda pestaña de la sección de un proyecto es la sección de recursos, aquí estarán todos los recursos y borradores del proyecto, agrupados en dos árboles, el árbol izquierdo contiene los recursos que serán publicados con el proyecto, mientras que el derecho contiene recursos de menos importancia que no se quieren eliminar.

Se pueden arrastrar los recursos dentro de su mismo árbol y entre los dos árboles, se puede minimizar el árbol derecho y se pueden filtrar los recursos de ambos árboles.

Figura 238: Manual de Usuario - Recursos

9.2.10.2 Referencias

A continuación de los recursos esta la pestaña de referencias,

The screenshot shows a software interface for managing references. At the top, there's a navigation bar with tabs: 'References List' (which is highlighted and circled in red), 'Add Reference', 'Discussions', 'Milestones', and 'Tasks'. Below the navigation bar is a table titled 'References List' with three columns: 'Name', 'URL', and 'Actions'. A single row is present in the table, representing a reference to 'Wikipedia' with the URL 'www.wikipedia.com'. To the right of the table, there are three buttons: 'Select' (with a dropdown arrow), 'Open' (represented by a blue folder icon), and 'Remove' (represented by a trash bin icon). All three buttons are circled in red.

Figura 239: Manual de Usuario - Referencias

9.2.10.3 Discusiones

La tercera pestaña contiene las discusiones,

The screenshot shows the 'References' tab selected in the navigation bar. Below it is a 'Create Reference' form with fields for 'Name' (containing 'La nueva España') and 'URL' (containing 'www.lne.es'). There is also a 'Create' button. Below the form is a 'Find References' section with a 'Key words:' input field and a 'Search' button. A link 'Create a new refer' is visible above the search button. At the bottom, there is a table titled 'References List' with columns 'Name', 'URL', and 'Actions', which is currently empty.

9.2.10.4 Hitos

La pestaña de hitos contiene un calendario en el que los usuarios pueden definir hitos en determinadas fechas. Además el gestor, podrá desde esta sección establecer el estado del proyecto o último hito superado. El estado esperado o hito esperado se calcula automáticamente en función de la fecha del proyecto.

Los hitos pueden ser movidos sobre el calendario, pueden ser eliminados y pueden ser creados, para crear hitos el usuario debe hacer doble clic en la fecha deseada, y para eliminarlos seleccionar el hito y presionar la tecla suprimir.

Milestones :

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
Start Project						
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8
12	13	14	15	16	17	18

State :

Desired milestone: Project Deliver. Last milestone: Start Project

Figura 240: Manual de Usuario – Hitos

9.2.10.5 Tareas

La última sección, también relacionada con la planificación, es la sección de tareas donde el usuario podrá crear y consultar tareas, además de tener acceso a un resumen del porcentaje de tareas completadas.

9.2.10.6 Consultar Tareas

Compl...	Name	Beguining	Total Hours	Worked Hours	Actions
	Develop a protot...	12:00 AM	27	0	Actions ▾
	Test the prototype	12:00 AM	27	0	work Done Delete

Figura 241: Manual de Usuario - Tareas

9.2.10.7 Crear Tarea

Para crear una tarea el usuario debe ir a la pestaña de crear tarea, y allí completar los campos de la tarea y pulsar el botón crear, entonces la tarea aparecerá automáticamente en el listado de tareas y en el resumen.

The screenshot shows a user interface for creating a task. At the top, there are three tabs: 'Tasks List', 'Create Task' (which is active and highlighted in blue), and 'Tasks Summary'. Below the tabs, there are three input fields: 'Name' containing 'Develop a prototype.', 'Beginning' containing '4/17/13' with a small calendar icon to its right, and 'Duration' with a slider bar set to approximately 10 units. In the bottom right corner of the form area, there is a blue rectangular button labeled 'Create'.

Figura 242: Manual de Usuario – Tareas I

9.2.10.8 Resumen de Tareas

Finalmente cada vez que el usuario acceda a la pestaña del resumen se dibujará una gráfica que resume los porcentajes completados para cada una de las tareas.

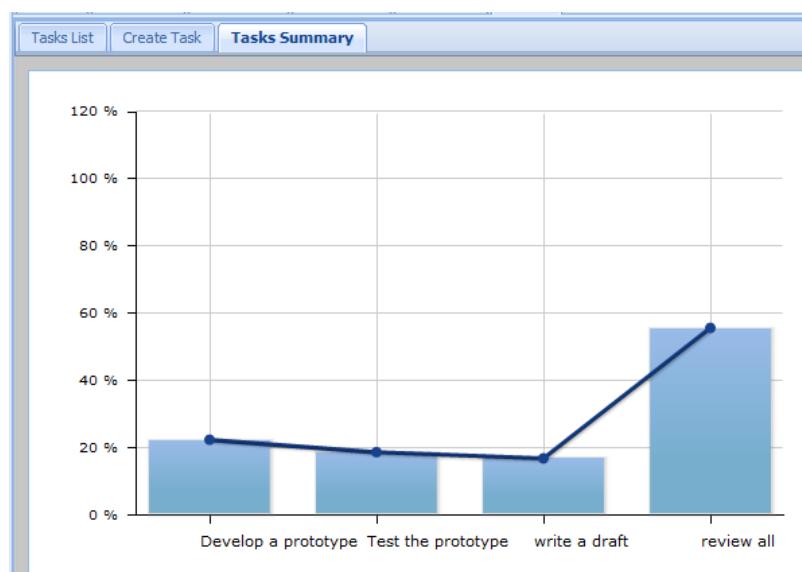


Figura 243: Manual de Usuario – Resumen de Tareas

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Una vez finalizado el proyecto final de carrera “Red colaborativa para el desarrollo de proyectos online”, podemos extraer las siguientes conclusiones.

En primer lugar me siento altamente satisfecho, los objetivos académicos fijados previo a la construcción del sistema, han sido cumplidos con creces, se ha desarrollado un sistema distribuido con una amplia gama de tecnologías bien aceptadas por la comunidad, que responde a una solución elaborada mediante la aplicación de técnicas de ingeniería del Software.

El resultado obtenido cumple con las especificaciones de la solución construida, solución que responde a un problema que si bien se define como pretexto para el desarrollo de un proyecto que completase mi perfil profesional, es en mi opinión una realidad, y este proyecto le ha ofrecido una solución moderna y creativa, aunque algo escasa de recursos y tiempo.

Entre las capacidades técnicas que he adquirido me gustaría destacar algunas.

- La construcción de una arquitectura distribuida.
- La aplicación de varias tecnologías y especificaciones como JPA 2.0, CDI 1.0, EJB 3.1, GWT, GXT o Errai.
- El diseño con patrones, inyección de dependencias y orientación aspectos.
- El uso de APIs más particulares con el fin de resolver problemas también más particulares, como las de Google, SVN KIT o Smack.
- La construcción de una aplicación enriquecida para internet.

Además de conocimientos teóricos y prácticos, he adquirido una serie de habilidades relacionadas con la toma de decisiones a todos los niveles, he visto como en ocasiones es mejor evitar un muro que intentar derribarlo, he sufrido malas decisiones, a corto y a largo plazo, y estas me han ayudado a tomar otras mejores, a modo general puedo definir esta retribución del proyecto como una dosis de seguridad y madurez, a falta de un océano de ella.

10.2 Ampliaciones

El sistema desarrollado destaca sobre todas las cosas por su flexibilidad a modificaciones y ampliaciones. En este apartado se describen algunas ampliaciones que serían viables e interesantes de incluir en un futuro. Todas y cada una de ellas disparaban el tiempo y la complejidad del desarrollo de este proyecto, lo cual es la principal razón de que no se encuentren incluidas.

10.2.1 Control de Versiones de los Recursos

Esta ampliación supone añadir a todos los recursos una opción para ver su historial, y permitir recuperar una versión anterior.

Existen dos posibles caminos, el primero y menos viable es tener recursos que modelen las versiones antiguas, y el segundo y más pragmático es utilizar el soporte que da para ello Apache Subversion.

El desarrollo de esta opción suponía un elevado coste durante todas las fases del proyecto por lo que fue desecharla.

10.2.2 Editor de Documentos Online

Esta ampliación pretende convertir los recursos de tipo documento en recursos editables, mediante la construcción de un editor especializado, una posible solución sería la de utilizar un editor HTML con conversores de HTML a PDF, DOCX u otros formatos. Estos conversores disparaban la complejidad del proyecto, y en consecuencia el tiempo.

10.2.3 Applet para Subir Carpetas

Esta ampliación pretende ofrecer al usuario la posibilidad de subir carpetas de recursos. Para ello se precisa la construcción de un Applet que permita seleccionar el o los directorios y los envíe al servidor.

10.2.4 Exportar Recursos a Google Docs.

Ofrecer la opción de exportar recursos hacia Google Docs, esto podría ser interesante y es mucho más fácil de añadir que las ampliaciones propuestas.

No se ha implementado porque no fue contemplado en el análisis inicial y el tiempo para implantarla de una forma prudente se estimó demasiado alto.

10.2.5 Diagrama de Gant

Añadir un diagrama de Gant que permita en función de las tareas definidas, sus fechas y duraciones, calcular y dibujar un diagrama de Gant.

No se ha implementado porque no fue contemplado en el análisis inicial y el tiempo para implantarla de una forma prudente no compensaba la funcionalidad aportada.

10.2.6 Relacionar Recursos con Tareas

Permitir asignar recursos previamente definidos a las tareas, tanto en el momento de su definición como posteriormente.

No se ha implementado porque no fue contemplado en el análisis inicial e influye en varias partes del sistema por lo que el riesgo de añadirla era demasiado alto.

10.2.7 Histórico de Acciones de Usuario

Permitir consultar todas o algún número de las acciones de un usuario sobre un proyecto.

No se ha implementado porque no fue contemplado en el análisis inicial y supone modificar el diseño del sistema desde su dominio. Esto no era viable en absoluto una vez diseñado el sistema.

Capítulo 11. Presupuesto

En esta sección se especifica de forma detallada el presupuesto general del proyecto que ya se ha resumido en el apartado 4.2 “Resumen del Presupuesto”.

Cabe decir que aunque el apartado 2.2 “Objetivos del Proyecto” se deja muy claro que no existe un fin comercial detrás de este proyecto, dado que es un proyecto de ingeniería el presupuesto se han hecho pensando en ejecutarlo e implantarlo, definiendo perfiles y sueldos capaces de conseguirlo, de ahí el elevado presupuesto.

Ítem	Subítem	Cantidad/ Horas	Concepto	Coste unitario/ Sueldo hora	Coste Total	Coste ítem
001			<i>Desarrollo del Sistema</i>			37.976,00 €
	01	128	Formación	22,87 €	2.928,00 €	
	02	24	Planificación	24,00 €	576,00 €	
	03	280	Análisis	24,00 €	6.720,00 €	
	04	232	Diseño	44,17 €	10.248,00 €	
	05	664	Implementación y Pruebas	26,36 €	17.504,00 €	
002			<i>Software</i>			2.650,00 €
	01	4	Windows 7 Home Premium	0,00 €	0,00 €	
		4	Red Hat Enterprise Linux 6.1	1.260,00 €	1.260,00 €	
	02	1	JBoss AS 7.1	0,00 €	0,00 €	
	04	1	Posgreet SQL 9.1	0,00 €	0,00 €	
			Visual SVN Server	0,00 €	0,00 €	
	03	1	Eclipse 3.6 Indigo	0,00 €	0,00 €	
	04		Maven 3.0.3	0,00 €	0,00 €	
	05	1	Sencha GXT 3 + Standard Support	755,00 €	755,00 €	
003			<i>Hardware</i>			11.632,00 €
	01	4	Servidor Dell PowerEdge R520	2.179,00 €	8.716,00 €	
	02	4	Ordenador Dell Performance Vostro with Dual Monitor	729,00 €	2.916,00 €	
004			<i>Otros Gastos</i>			2.200,00 €
	01	1	Gastos de Oficina (material, alquiler, luz, etc.)	1.200,00 €	1.200,00 €	

	02	2	Gastos de transportes y comidas.	1.000,00 €	1.000,00 €	
				Subtotal	54.458,72 €	
				IVA (18 %)	9.802,44 €	
				TOTAL	64.260,44 €	

- **Desarrollo del Sistema:** Este ítem representa todo el esfuerzo humano requerido para el desarrollo del sistema, que comprende tareas como investigación, análisis, diseño o desarrollo. Todas las tareas de este ítem se encuentran detalladas en la planificación del proyecto.
- **Software:** Software usado tanto para el desarrollo como para la implantación, incluye licencias de frameworks y sistemas operativos. Se ha detallado todo el software usado aunque en algunos casos este no suponga gasto, sí que representa el posible gasto y el ahorro. Más concretamente se ha adquirido una licencia para un framework de componentes visuales y 4 licencias para Red Hat Enterprise Linux.
- **Hardware:** Hardware necesario tanto para el desarrollo como para la puesta en producción. Se han adquirido 4 estaciones de trabajo y 4 servidores.
- **Otros Gastos:** Gastos derivados del trasiego de personal durante los 8 meses que dura el proyecto.

Capítulo 12. Referencias Bibliográficas

12.1 Libros y Artículos

[Alur03] Alur, Deepak; Crupi, Jhon; Malks, Dan."Core J2EE Patterns: Best Practices and Design Strategies (2nd Edition)".Prentice Hall. ISBN 978-0-321-41309-3.

[Cockburn01] Cockburn, Alistar. "Writing Effective Use Cases".Addison-Wesley. 2001. ISBN 0-201-70225-8.

[Gamma94] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John; "Design Patterns: Elements of Reusable Object-Oriented Software". Addison Wesley. 1994. ISBN-10:0201633612, ISBN-13: 978-0201633610.

[Keith09] Mike, Keith; Merrick Schincariol. "Pro JPA 2: Mastering the Java Persistence API". Apress. 2009. ISBN 978-1-4302-1957-6.

[Kühne98] Thomas Kühne. "A functional Pattern System for Object-Oriented Design".

[Mak08] Mak, Gary. "Spring Recipes: A Problem-Solution Approach". Press. 2008. ISBN 978-1-59059-979-2.

[Marchioni11] Marchioni, Francesco. "JBoss AS 7 Configuration, Deployment, and Administration". Packt.2011.ISBN 978-1-84951-678-5.

[Microsoft09] Microsoft Patterns & Practices Team. "Microsoft Application Architecture Guide 2nd Edition". Microsoft Corporation.2009.ISBN 9780735627109.

[Mularien10] Peter, Mularien. "Spring Security 3: Secure your web applications against malicious intruders with this easy to follow practical guide".Packt. 2010. ISBN 978-1-59059-979-2.

[Smeets08] Bram, Smeets; Uri, Boness; Roald, Bankras. "Beginning Google Web Toolkit: From Novice to Professional".Apress.2008. ISBN 978-1-4302-1032-0.

12.2 Referencias en Internet

[Arquillian12] JBoss Community. "Arquillian Guides".

http://arquillian.org/guides/getting_started.

[CDI09] JCP. "JSR 299 Contexts and Dependency Injection Specification".

<http://jcp.org/aboutJava/communityprocess/final/jsr299/index.html.2009>.

[Errai 11] JBoss Community. "Errai Project". <http://www.jboss.org/errai.2011>.

[Errai 12] Heiko Braun, Mike Brock, Christian Sadilek, Jonathan Fuerth. "Errai Developer Blog".

<http://www.jboss.org/errai.2011>.

[MVP11] Google. "Model View Presenter Architecture".<https://developers.google.com/web-toolkit/articles/mvp-architecture.2011>

[GWT11] Google. "Guía para desarrolladores de GWT".<https://developers.google.com/web-toolkit/doc/latest/DevGuide.2011>

[Hassan08] Hassan Montero, Y. "Guía de Evaluación Heurística de Sitios Web".

<http://www.nosolousabilidad.com/articulos/heuristica.htm.2008>.

[Hostettler12] Steve Hostettler. "A JEE6 Security Interceptor for Tomcat 7".

<http://www.hostettler.net/blog/2012/05/02/a-jee6-security-interceptor-for-tomcat-7.2012>.

[JBoss-MODULES12] JBoss Community. "JBoss Modules

Project".<https://docs.jboss.org/author/display/MODULES/Introduction>.

[JEE09]JCP. "JSR 00316 Java™ Platform, Enterprise Edition 6".

<http://jcp.org/aboutJava/communityprocess/final/jsr316/index.html>.

[JPA09] JCP." JSR 00317 JavaTM Persistence 2.0".

<http://jcp.org/aboutJava/communityprocess/final/jsr317/index.html>.

[JVM12] ORACLE. "JavaTM Virtual Machine Specification".

<http://docs.oracle.com/javase/specs/jvms/se7/html/index.html>.

[JEE12] ORACLE. "The Java EE 6 Tutorial". <http://docs.oracle.com/javaee/6/tutorial/doc>.

[Paredes12] Adrián Paredes. "Spring V.S. Java EE".

<http://elblogdelfrasco.blogspot.com.es/2012/07/spring-vs-java-ee.html>.

[PSQL11] Wikipedia. "PosgreSQL".

<http://www.postgresql.org/docs/9.1/interactive/index.html>.

[TAFZAL09] Shaikh Tafzal. "Google Talk & Fun with XMPP".

<http://tafzal.blogspot.com.es/2009/09/google-talk-gtalk-google-chatting-xmpp.html>.

[Vogella11] Lars Vogel."GWT Tutorial". <http://www.vogella.com/articles/GWT/article.html>.

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

- **API:** “Application Programming Interface”. Es el conjunto de métodos que ofrece un componente software para ser utilizado desde otro componente.
- **CDI:** “Contexts and Dependency Injection”. Especificación para la inyección de dependencias en Java.
- **EJB:** “Enterprise Java Beans”. Arquitectura de componentes de servidor para la plataforma Java.
- **GWT:** “Google Web Toolkit”. Es un framework creado por Google para el desarrollo de aplicaciones Web con interfaces enriquecidas
- **Hibernate:** Es un framework para el “Mapeo objeto-relacional (ORM)” para la plataforma Java, aunque también dispone de una versión para .Net.
- **HTML:** “eXtensible HyperText Markup Language”. Lenguaje extensible de marcado, es usado para representar mediante etiquetas el contenido de una página web.
- **HTTP:** “HyperText Transfer Protocol”. Protocolo de transferencia de hipertexto. Es el protocolo empleado en la comunicación entre navegadores y servidores Web.
- **JAR:** “Java Archive File”. Fichero que envuelve un conjunto de clases Java y permite ejecutarlas.
- **Java:** Lenguaje de programación orientado a objetos desarrollado por Sun Microsystems en los años 90.
- **JBossAS7:** “JBoss Application Server 7”. Es la séptima edición del servidor de aplicaciones de JBoss.
- **JCP:** Es un mecanismo normalizado para el desarrollo de especificaciones técnicas de para la tecnología Java. Para ello las partes interesadas deben ser miembros de la JCP, algunos miembros actualmente son Google Inc., Oracle o Red Hat JBoss. Hay más de trescientas especificaciones enmarcadas en la JCP.
- **JDBC:** “Java Database Connectivity”. Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del SGBD usado.
- **JEE:** “Java Platform Enterprise Edition”. Es una plataforma de programación, parte de la plataforma Java, para el desarrollo de aplicaciones empresariales que se distribuyen en Internet.
- **JPA:** “Java Persistence API”. API para el tratamiento de datos relacionales mediante el lenguaje Java.

- **JSR:** “Java Specification Request”. Es una petición de un miembro de la JCP para la creación de una especificación o norma de alguna tecnología Java, en el marco de la JCP.
- **JUnit:** Framework para la automatización de las pruebas unitarias sobre aplicaciones o componentes software desarrollados bajo la plataforma Java.
- **Middleware:** Software que esta entre dos elementos hardware, un elemento hardware y otro software o dos elementos software.
- **MVP:** Model View Presenter”. Patrón de diseño derivado del Model View Controller, que pero incluye diferencias importantes, como separar la vista del modelo, e introducir el objeto Presenter en lugar del Controller, que a diferencia de éste, sí contiene lógica de presentación.
- **ORM:** “Object Relational Mapping”. Técnica de programación para convertir datos de un sistema orientado a objetos a datos de una base de datos relacional
- **Patrón Arquitectónico:** Patrones de diseño que ofrecen solución a problemas de arquitectura de software.
- **Patrón de Diseño:** Plantilla de diseño reutilizable, que resuelve un problema conocido en un contexto conocido, y esta solución está documentada.
- **PostgreSQL:** Sistema de gestión de bases de datos libre y de código abierto.
- **SGBD:** “Sistema de Gestión de Bases de Datos”. Software utilizado para la gestión de bases de datos.
- **SQL:** Es un lenguaje de definición y de manipulación de datos, que se espera estén en una base de datos relacional.
- **UML:** “Unified Modeling Language”. Se trata del lenguaje de modelado de software más utilizado en la actualidad.
- **Usabilidad:** Es la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto.
- **WAR:** “Web Archive File”. Fichero para empaquetar aplicaciones web java, contiene las clases Java y un descriptor de despliegue, todo con una estructura bien definida.
- **Widget:** Componente visual, en contextos de desarrollo de software, componente reutilizable de interfaz de usuario.
- **XML:** “eXtensible Markup Language”. Lenguaje de marcado extensible. Es un metalenguaje que permite definir la gramática de un lenguaje específico.
- **XMPP:** “extensible Messaging Presence Protocol”. Es un protocolo abierto basado en XML para la comunicación entre middleware de mensajería instantánea.

13.2 Contenido Entregado en el CD-ROM

13.2.1 Contenidos

Adjunta a esta documentación se entrega un CD-ROM, cuyo contenido se detalla a continuación.

Directorio	Contenido
<i>./Directorio raíz del CD</i>	Contiene todos los ficheros del proyecto.
<i>./cnpd</i>	Contiene toda la estructura de directorios del sistema para desarrollo.
<i>./cnpd/cnpd-business/</i>	Contiene toda la estructura de directorios del proyecto cnpd-business para desarrollo.
<i>./cnpd/cnpd-web/</i>	Contiene toda la estructura de directorios del proyecto cnpd-web para desarrollo.
<i>./cnpd/cnpd-client/</i>	Contiene toda la estructura de directorios del proyecto cnpd-client para desarrollo.
<i>./instalacion</i>	Ficheros utilizados para la instalación del proyecto, los artefactos de despliegue y ejecutables.
<i>./documentacion</i>	Contiene toda la documentación asociada al proyecto, en formato PDF y Word. Además se incluye el fichero de Enterprise Architect con todos los modelos del sistema y el fichero de Microsoft Office Project con la planificación del proyecto.
<i>./herramientas</i>	Contiene las herramientas utilizadas para el desarrollo y puesta en marcha del proyecto.
<i>./herramientas/desarrollo</i>	Herramientas utilizadas para el desarrollo. El entorno de desarrollo (Eclipse Indigo) con todos los plugins instalados y ficheros de configuración usados. Apache Maven 3.03 y todos sus ficheros de configuración. Contiene el instalador de Graphviz.
<i>./herramientas/explotacion</i>	Contiene todas las herramientas necesarias para desplegar el sistema. El instalador del SGBD PostgreSQL 9.1 y el fichero de configuración modificado. El servidor de aplicaciones JBoss AS7 por defecto, y todos los ficheros de configuración para configurarlo. Contiene una JDK 1.6 portable.

13.2.2 Estructura de Directorios de “cnpd”

Dentro del directorio cnpd, hay tres directorios, correspondientes a los tres proyectos desarrollados. Aunque se han desarrollado por separado, todos comparten la misma estructura, aunque en algunos casos no se usan todos los directorios.

Directorio	Contenido
./ Directorio raíz desarrollo	Contiene todos los ficheros del proyecto.
./src	Contiene todo el código fuente y los recursos del proyecto.
./src/main/	Contiene todo el código fuente y los recursos del proyecto final, no tests.
./src/main/java	Contiene todo el código fuente Java.
./src/main/resources	Contiene todos los recursos del proyecto en su mayoría, ficheros de configuración.
./src/main/resources/META-INF	Contiene ficheros de configuración especiales, en la aplicación web se encuentran en WEB-INF.
./src/main/webapp	Contiene la estructura del WAR a generar, en el caso que procede,
./src/test/java	Contiene todos los ficheros de código fuente Java para las pruebas.
./src/test/resources	Contiene todos los ficheros de recursos para las pruebas.
./src/Jetty/	Redefine parte de la estructura del war para el despliegue sobre un servidor Jetty incrustado.
./src/jboss7/	Redefine parte de la estructura del war para el despliegue sobre el JBoss AS7.
./target	Contiene todos los ficheros generados por Maven tras construir el proyecto.
./doc	Contiene notas de implementación y Java doc.
./doc/api-docs	Contiene todo el Java Doc del proyecto.

13.2.3 Código Ejecutable e Instalación

Para poner en marcha el sistema es necesario llevar a cabo una serie de pasos que se resumen a continuación.

1. Instalar PostgreSQL 9.1, configurarlo para que acepte conexiones remotas y crear una base de datos llamada cnpd.
2. Instalar Visual SVN Server 2.5 y crear un repositorio llamado cnpd con usuario y contraseña (cnpd, cnpd).
3. Instalar una JDK 1.6 tanto en el nivel de negocios como en el nivel web.
4. Instalar y configurar JBoss AS7 para arrancar la instancia del nivel de negocios.

5. Instalar y configurar JBoss AS7 para arrancar la instancia del nivel de web.
6. Acceder a la URL <http://{jboss-as7-business-tier}:9990/console/App.html> y desplegar el fichero cnpd-business.jar.
7. Acceder a la URL <http://{jboss-as7-web-tier}:9990/console/App.html> y desplegar el fichero cnpd.war.
8. Acceder a la URL <http://{jboss-as7-web-tier}/cnpd/App.html>.
9. Ejecutar el fichero cnpd-client.jar.

13.2.4 Ficheros de Configuración

En este apartado se listan todos los ficheros de configuración de la aplicación, dando una breve descripción de ellos (para más información ver Manual de Instalación).

Es muy importante tener presente que algunos de estos ficheros están presentes en los tres proyectos y con distintas configuraciones. También se usan distintas configuraciones para el código de pruebas.

- persistence.xml: Fichero de configuración definido por la especificación de JPA, en él se define todo lo relativo al API, origen de datos, dialecto, si queremos que nos actualice el esquema de la bb.dd o el dialecto de la bb.dd. Debe estar en el directorio META-INF.
- beans.xml: Fichero de configuración de beans definido por la especificación de CDI, en él se definen aspectos relativos a la inyección de dependencias, TIENE QUE ESTAR aunque sea vacío. Debe estar en el directorio META-INF.
- jboss-deployment-structure.xml: Fichero para especificar aspectos relativos al despliegue del artefacto sobre el JBossAS7, se definen aspectos como módulos del JBoss AS7 que se deben cargar. Debe estar en META-INF o en WEB-INF si el artefacto es un war.
- log4j.xml: Fichero de configuración de Apache Log4j, aquí definimos todo lo relativo a los mensajes de log de las aplicaciones, destinos, niveles, etc...
- mail.properties: Fichero de configuración para el envío de correos, es propio del sistema y está pensado para cosas como deshabilitar el envío de correos de forma global.
- svn.properties: Fichero de configuración relativo al repositorio de ficheros, contiene parámetros de configuración como la dirección del repositorio y el usuario y la contraseña del mismo.
- strings.properties: Fichero con cadenas de la aplicación como por ejemplo correos electrónicos genéricos.

- ejb-names.properties: Fichero con los nombres de los Enterprise Java Beans de la capa de negocio, son los nombres usados para obtener acceso a la capa remota. Su objetivo principal es poder cambiar el despliegue distribuido a un despliegue local si fuese interesante.
- jboss-ejb-client.properties: Fichero de configuración para clientes de EJBs desplegados en JBoss, es solo para clientes independientes, no para aplicaciones web desplegadas en otro contenedor.
- ErraiApp.properties: Fichero de configuración para el framework Errai, Puede estar vacío y no se debe borrar incluso, es usado como marca para cargar los módulos de Errai.
- ErraiService.properties: Fichero de configuración para aspectos más técnicos del framework Errai, por ejemplo implementaciones para factorías abstractas.
- gtalk.properties: Fichero propio del sistema, contiene los parámetros necesarios para conectar con Google Talk, dirección puertos.
- spring-security.xml: Fichero de configuración de Spring que contiene la configuración relativa al framework Spring Security.

13.3 Índice Alfabético

A

AJAX, 41
 Análisis, 24, 25, 28, 55, 93, 129, 130, 136, 206, 232, 311
 AOP, 174, 351
 API doclet, 264
 Applet, 308
Arquillian, 147
 Arquitectura, 28, 32, 40, 151, 315
 Autenticación, 222

C

Casos de Uso, 74, 76
 CDI, 11
 Conclusiones, 307
 Contact, 11
 Contacto, 7, 63, 65, 74, 81, 82, 83, 102, 103, 104, 105, 106, 107, 108, 109, 130, 134, 140

D

DBunit, 147
 Diagrama de Gant, 309
 Diseño, 24, 25, 28, 151, 169, 173, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 256, 311, 316, 347

E

Eclipse, 41, 44, 260, 262, 263, 268, 311, 317
 EJB, 5, 7, 11, 24, 29, 31, 32, 157, 162, 165, 307, 332
 Errai, 5, 9, 43, 159, 170, 188, 201, 307, 314, 320

F

Filter, 36, 40

G

Google Docs, 58, 71, 72, 89, 90, 122, 158, 308
 Google Talk, 24, 45, 159, 162, 163, 314, 320
 Google Web Toolkit, 5, 9, 24, 41, 60, 313, 315
 GWT, 5, 7, 11, 24, 41, 42, 43, 60, 155, 159, 259, 262, 264, 266, 267, 307, 337, 338
 GXT, 5, 7, 11, 307

H

HTML, 163, 259, 265, 308

I

Implementación, 24, 25, 159, 160, 259, 311
 Interceptores, 351
 Inyección de Dependencias, 173

J

Java Community Process, 259
JBoss AS7, 37, 157, 158, 159, 262, 263, 300, 303, 318, 319
 JEE, 5, 7, 11, 24, 28, 31, 32, 37, 158, 262, 351
 JUnit, 147

L

Listener, 36

M

Maven, 262, 263, 264
 Mensaje, 7, 45, 131, 134
 Message, 11, 348, 349
 Microsoft Windows, 265

P

Patrón Adapter, 172, 347
 Patrón Builder, 173, 350
 Patrón Facade, 171
 Patrón Factory Method, 171
 Patrón Layers, 152, 153, 155
 Patrón Modelo Vista Presentador, 260
 Patrón MVP, 41
 Patrón N Tiers, 151, 152
 Patrón Session Facade, 171, 172
 Patrón Singleton, 170, 348
 Patrón Translator, 173, 260, 351
 Planificación, 51, 131, 132, 143, 311
 PostgreSQL, 49, 289, 290, 291, 292, 293, 294, 295, 296, 318
 Presupuesto, 54, 311
 Programación Orientada a Aspectos, 173
 Project, 11

Proyecto, 1, 7, 23, 24, 27, 28, 51, 59, 66, 67, 69, 84, 85, 86, 87, 88, 110, 115, 116, 117, 118, 119, 131,

134, 140, 141, 142, 143, 144, 158, 306

Pruebas, 147, 149, 248, 254, 256, 271, 280, 287, 311

R

Recurso, 7, 71, 72, 89, 90, 91, 92, 120, 122, 123, 124, 125, 126, 127, 135

Requisitos, 61, 63, 66, 71, 73, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 136, 149

Resource, 11, 327, 329

SGBD, 49, 162, 222, 265, 289, 291, 293, 294, 296, 297, 301

Smack, 45, 46, 159, 191, 307

Spring Security, 39, 40, 159, 160, 172, 192, 222, 313, 320

Swing, 42, 44, 160

Swingx, 44, 160

S

Servlet, 31, 36, 43, 159, 162

T

Tarea, 7, 69

Task, 11, 327, 328

X

XML, 35, 45, 261, 265

13.4 Código Fuente

El código de la aplicación alcanzo un número clases y de líneas excesivamente elevadas, ver apartado (7.4.2 Métricas), por ello en esta sección solamente se incluyen algunas clases como exponentes de las partes más significativas del sistema, así como implementaciones de algunos patrones útiles para la comprensión de otras partes de la documentación.

13.4.1 Ejemplos Capa de Persistencia

Algunos ficheros de código como ejemplo de las clases de la capa de persistencia, las cuales siguen todas el patrón DAO.

13.4.1.1 Fichero “*IDao.java*”

```
package com.armandorv.cnpd.persistence;

import java.util.List;

/**
 * Generic interface for all entities
 *
 * @author armandorv
 */
public interface IDao<T>
{

    /**
     * Obtain entities by their primary key
     */
    T findById(Long id);

    /**
     * Get all entities
     */
    List<T> findAll();

    /**
     * Persist a new entity
     */
    void persist(T entity);

    /**
     * Remove a entity
     */
    void remove(T entity);

    /**
     * Reload the value of an entity from the data base overwriting the changes
     * over the entity.
     */
    public void refresh(T entity);

    /**
     * Take entity manager cache to database.
     */
    public void flush();

    /**
     * Clear the persistence context.
     */
    public void clear();
}
```

13.4.1.2 Fichero “IUserDao.java”

```

package com.armandorv.cnfd.persistence;

import java.util.List;

import com.armandorv.cnfd.model.User;

/**
 * Extends the basic {@link IDao} class to add some useful methods for users.
 *
 * @author armandorv
 *
 */
public interface IUserDao extends IDao<User>
{
    /**
     * Get a user by their username.
     *
     * @param username
     *          a user name of {@link AccountData} class.
     * @return the user who match with given username or null;
     */
    public User findByUsername(String username);

    /**
     * @return all users who has a invitation to a given project.
     */
    public List<User> findProjectInvitedUsers(Long projectId);

    /**
     * @return all users who has a invitation to a given meeting.
     */
    public List<User> findMeetingInvitedUsers(Long meetingId);
}

```

13.4.1.3 Fichero “UserDao.java”

```

package com.armandorv.cnfd.persistence.impl;

import java.util.List;

import javax.persistence.TypedQuery;

import org.apache.log4j.Logger;

import com.armandorv.cnfd.model.User;
import com.armandorv.cnfd.persistence.IUserDao;
import com.armandorv.cnfd.persistence.exception.DuplicatedUsernameException;

/**
 * Class which encapsulate data access for users, it is necessary to be into a
 * managed context
 *
 * @author armandorv
 *
 */
public class UserDao extends JpaDaoSupport implements IUserDao
{
    private static Logger log = Logger.getLogger(UserDao.class);

    @Override
    public User findById(Long id)
    {
        return em.find(User.class, id);
    }

    @Override
    public List<User> findAll()
    {
        return em.createNamedQuery("User.findAll", User.class).getResultList();
    }
}

```

```
}

@Override
public void persist(User entity)
{
    em.persist(entity);
}

@Override
public void remove(User entity)
{
    em.remove(entity);
}

@Override
public void refresh(User entity)
{
    em.refresh(entity);
}

@Override
public User findByUsername(String username)
{
    TypedQuery<User> query = em.createNamedQuery("User.findByUsername",
        User.class);

    query.setParameter("account", username);

    List<User> result = query.getResultList();

    if (result.size() > 1)
    {
        log.error("Duplicated username :" + username + ", there is:"
            + result.size() + "users.");

        throw new DuplicatedUsernameException("Username :" + username
            + " is duplicated");
    }

    if (result.size() == 0)
        return null;

    return result.get(0);
}

@Override
public List<User> findProjectInvitedUsers(Long projectId)
{
    TypedQuery<User> query = em.createNamedQuery("User.findProjectInvited",
        User.class);

    query.setParameter("projectId", projectId);

    return query.getResultList();
}

@Override
public List<User> findMeetingInvitedUsers(Long meetingId)
{
    TypedQuery<User> query = em.createNamedQuery("User.findMeetingInvited",
        User.class);

    query.setParameter("meetingId", meetingId);

    return query.getResultList();
}

}
```

13.4.2 Ejemplos Entidades

Ejemplos de código de algunas entidades anotadas como entidades de JPA con sus respectivos mapeos.

13.4.2.1 Fichero “Project.java”

```
package com.armandorv.cnpd.model;

import java.io.Serializable;
import java.util.Collections;
import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;

/**
 * A object which model a project of business domain.
 *
 * @author armandorv
 *
 */
@NamedQueries(
{
    @NamedQuery(name = "Project.findAll", query = "select p from Project p"),
    @NamedQuery(name = "Project.findByArea", query = "select p from Project p where
p.knowledgeArea = :knowledgeArea"),
    @NamedQuery(name = "Project.findByTitle", query = "select p from Project p where
p.title = :title" })
@Table(name = "project")
@Entity
public class Project implements Serializable
{
    private static final long serialVersionUID = -6743202992210799015L;

    @Id
    @GeneratedValue
    private Long id;

    private String title;

    private String Description;

    private Boolean publisht;

    private String url;

    @Column(nullable = false)
    private Long managerId;

    @OneToOne
    private Milestone lastMilestone;

    @ManyToOne
    private KnowledgeArea knowledgeArea;

    @OneToMany(cascade = CascadeType.REMOVE, mappedBy = "project")
    private Set<Milestone> milestones = new HashSet<Milestone>();
}
```

```

@OneToMany(cascade = CascadeType.REMOVE)
@JoinTable(name = "project_resources")
private Set<Resource> resources = new HashSet<Resource>();

@OneToMany(cascade = CascadeType.REMOVE)
@JoinTable(name = "project_drafts")
private Set<Resource> drafts = new HashSet<Resource>();;

@ManyToMany(cascade = CascadeType.REMOVE)
@JoinTable(name = "project_references")
private Set<Reference> references = new HashSet<Reference>();

@OneToMany(cascade = CascadeType.REMOVE, mappedBy = "project")
private Set<Discussion> discussions = new HashSet<Discussion>();

@ManyToMany(mappedBy = "projects", cascade = CascadeType.REFRESH)
private Set<User> members = new HashSet<User>();

@OneToMany(cascade = CascadeType.REMOVE, mappedBy = "project")
private Set<Task> tasks = new HashSet<Task>();

/* ***** Add and Remove ***** */

public void addResource(Resource resource)
{
    this.resources.add(resource);
}

public void removeResource(Resource resource)
{
    this.resources.remove(resource);
}

public void addDraft(Resource draft)
{
    this.drafts.add(draft);
}

public void removeDraft(Resource draft)
{
    this.drafts.remove(draft);
}

public void addReference(Reference reference)
{
    this.references.add(reference);
}

public void removeReference(Reference reference)
{
    this.references.remove(reference);
}

public void removeReferences()
{
    this.references.clear();
}

public void addMember(User user)
{
    this.members.add(user);
}

public void removeMember(User user)
{
    this.members.remove(user);
}

/* ***** Getters and setters ***** */

public Set<User> getMembers()
{
    return Collections.unmodifiableSet(members);
}

```

```
public void setMembers(Set<User> members)
{
    this.members = members;
}

public void setDiscussions(Set<Discussion> discussions)
{
    this.discussions = discussions;
}

public Set<Discussion> getDiscussions()
{
    return Collections.unmodifiableSet(discussions);
}

public Set<Task> getTasks()
{
    return Collections.unmodifiableSet(tasks);
}

public void setTasks(Set<Task> tasks)
{
    this.tasks = tasks;
}

public void setReferences(Set<Reference> references)
{
    this.references = references;
}

public Set<Reference> getReferences()
{
    return Collections.unmodifiableSet(references);
}

public void setDescription(String description)
{
    Description = description;
}

public String getDescription()
{
    return Description;
}

public void setTitle(String title)
{
    this.title = title;
}

public String getTitle()
{
    return title;
}

public Boolean isPublisht()
{
    return publisht;
}

public void setPublisht(Boolean publisht)
{
    this.publisht = publisht;
}

protected void setId(Long id)
{
    this.id = id;
}

public Long getId()
{
    return id;
}

public String getUrl()
```

```

    {
        return url;
    }

    public void setUrl(String url)
    {
        this.url = url;
    }

    public Set<Resource> getResources()
    {
        return Collections.unmodifiableSet(resources);
    }

    public void setResources(Set<Resource> resources)
    {
        this.resources = resources;
    }

    public Long getManagerId()
    {
        return managerId;
    }

    public void setManagerId(Long managerId)
    {
        this.managerId = managerId;
    }

    public Set<Milestone> getMilestones()
    {
        return Collections.unmodifiableSet(milestones);
    }

    public void setMilestones(Set<Milestone> milestones)
    {
        this.milestones = milestones;
    }

    public Milestone getLastMilestone()
    {
        return lastMilestone;
    }

    public void setLastMilestone(Milestone lastMilestone)
    {
        this.lastMilestone = lastMilestone;
    }

    public Set<Resource> getDrafts()
    {
        return Collections.unmodifiableSet(drafts);
    }

    public void setDrafts(Set<Resource> drafts)
    {
        this.drafts = drafts;
    }

    public KnowledgeArea getKnowledgeArea()
    {
        return knowledgeArea;
    }

    public void setKnowledgeArea(KnowledgeArea knowledgeArea)
    {
        this.knowledgeArea = knowledgeArea;
    }

    /* ***** Hash code, equals and toString.***** */

    @Override
    public String toString()
    {
        return "Project [id=" + id + ", title=" + title + ", Description=" +
               Description + ", externalReferences=" + references
    }
}

```

```

        + ", discussions=" + discussions + ", members=" + members + "]";
    }

@Override
public int hashCode()
{
    final int prime = 31;
    int result = 1;
    result = prime * result + ((id == null) ? 0 : id.hashCode());
    return result;
}

@Override
public boolean equals(Object obj)
{
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Project other = (Project) obj;
    if (id == null)
    {
        if (other.id != null)
            return false;
    }
    else if (!id.equals(other.id))
        return false;
    return true;
}
}

```

13.4.2.2 Fichero “Notification.java”

```

package com.armandorv.cnpd.model;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Entity;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "notifiaction")
@NamedQueries(
{
    @NamedQuery(name = "Notification.findAll", query = "select n from Notification
n"),
    @NamedQuery(name = "Notification.findByUser", query = "select n from Notification
n where n.user=:user"))
public class Notification implements Serializable
{

    private static final long serialVersionUID = 1934510367303781619L;

    @Id
    @GeneratedValue
    private Long id;

    private Date date;

    private String message;

    private Long thirdPartId;
}

```

```
private Boolean notified;

@Enumerated(EnumType.STRING)
private NotificationKind kind;

@ManyToOne
private User user;

public Long getId()
{
    return id;
}

protected void setId(Long id)
{
    this.id = id;
}

public Date getDate()
{
    return date;
}

public void setDate(Date date)
{
    this.date = date;
}

public String getMessage()
{
    return message;
}

public void setMessage(String message)
{
    this.message = message;
}

public Boolean isNotified()
{
    return notified;
}

public void setNotified(Boolean notified)
{
    this.notified = notified;
}

public Long getThirdPartId()
{
    return thirdPartId;
}

public void setThirdPartId(Long thirdPartId)
{
    this.thirdPartId = thirdPartId;
}

public User getUser()
{
    return user;
}

public void setUser(User user)
{
    this.user = user;
}

public NotificationKind getKind()
{
    return kind;
}

public void setKind(NotificationKind kind)
{
```

```

        this.kind = kind;
    }

@Override
public String toString()
{
    return "Notification [id=" + id + ", time=" + date + ", message="
           + message + ", notified=" + notified + ", user=" + user + "]";
}

@Override
public int hashCode()
{
    final int prime = 31;
    int result = 1;
    result = prime * result + ((id == null) ? 0 : id.hashCode());
    return result;
}

@Override
public boolean equals(Object obj)
{
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Notification other = (Notification) obj;
    if (id == null)
    {
        if (other.id != null)
            return false;
    }
    else if (!id.equals(other.id))
        return false;
    return true;
}
}

```

13.4.3 Ejemplos Capa de Negocio

Ejemplos de la capa de negocio la cual ofrece interfaces transaccionales y accesibles remotamente gracias a la tecnología EJB.

Algunas de las clases de la capa de negocio, sirven como ejemplo de la aplicación del patrón Fachada.

13.4.3.1 Fichero “*IProjectsManager.java*”

```

package com.armandorv.cnpd.business;

import java.util.Date;
import java.util.List;
import java.util.Set;

import com.armandorv.cnpd.model.Discussion;
import com.armandorv.cnpd.model.KnowledgeArea;
import com.armandorv.cnpd.model.Milestone;
import com.armandorv.cnpd.model.Project;
import com.armandorv.cnpd.model.Reference;
import com.armandorv.cnpd.model.Task;
import com.armandorv.cnpd.model.User;
import com.armandorv.cnpd.model.Vote;

```

```

/**
 * Session Facade for deal with projects and all projects related concerns.
 *
 * @author armandorv
 *
 */
public interface IProjectsManager
{
    /**
     * Create a new project.
     *
     * @param userId
     *          id of user who are creating the project.
     * @param idKnowledgeArea
     *          id of selected knowledge area for the project.
     * @param project
     *          project info.
     * @param idMembers
     *          variable array of hypothetical members identifiers.
     * @return the new project full filled.
     */
    Project createProject(long userId, long idKnowledgeArea, Project project,
        long... idMembers);

    /**
     * Retrieve a project for a given identifier.
     *
     * @param id
     *          identifier of desired project.
     * @return the project which their identifier match with given identifier.
     */
    Project getProject(long id);

    /**
     * Get all projects of a user.
     *
     * @param user
     *          user to get their projects.
     * @return all projects of given user.
     */
    Set<Project> getProjectsByUser(long userId);

    /**
     * @return All projects in the system.
     */
    List<Project> getAllProjects();

    /**
     * A user leave a project, if user is the manager other manager will be
     * reassigned and if there is no more users, project will disappear
     *
     * @param userId
     *          identifier of user who are leaving the project.
     * @param projectId
     *          identifier of project.
     */
    void leaveProject(long userId, long projectId);

    /**
     * Get all invitations to project that a user has.
     *
     * @param userId
     *          identifier of user.
     * @return project invitations.
     */
    Set<Project> getProjectInvitations(long userId);

    /**
     * Send a invitation for a project to a user.
     *
     * @param userId
     *          identifier of user to send the invitation.
     * @param projectId
     *          identifier of the project.
     */
}

```

```
void sendInvitationToProject(long userId, long projectId);

/**
 * Resolve a invitations accepting or refusing it.
 *
 * @param userId
 *         id of invited user.
 * @param projectId
 *         id of project.
 * @param accept
 *         if user accept is true.
 */
void resolveProjectInvitation(long userId, long projectId, boolean accept);

/**
 * Method to get all members of a project.
 *
 * @param projectId
 * @return
 */
Set<User> getProjectMembers(long projectId);

/**
 * Remove a project of system.
 *
 * @param projectId
 *         identifier of the project to remove.
 */
void deleteProject(long projectId);

/**
 * Set the project scope to public.
 *
 * @param projectId
 */
void publishProject(long projectId);

/**
 * Modify a project given their identifier.
 *
 * @param projectId identifier of project to be modified.
 * @param knowledgeArea a correct Knowledge Area identifier.
 * @param title the new title, is supposed to be a valid String.
 * @param description the new description, is supposed to be a valid String.
 */
void modifyProject(long projectId, long knowledgeArea, String title, String
description);

/**
 * Set a user as manager of a project.
 *
 * @param projectId identifier of project added.
 * @param newManagerId identifier of the new manager.
 */
void setProjectManager(long projectId, long newManagerId);

/**
 * Method to obtain all references of a project.
 *
 * @param projectId
 *         identifier of a project.
 * @return All external references of given project.
 */
Set<Reference> getReferences(long projectId);

/**
 * @return All references into the system.
 */
List<Reference> getAllReferences();

/**
 * Add a existent reference to project.
 * @param referenceId identifier of reference.
 */
void addReference(long projectId, long referenceId);
```

```

    /**
     * Create a reference and add it to the project.
     *
     * @param reference new reference.
     *
     * @return the new reference.
     */
    Reference addNewReference(long projectId, Reference reference);

    /**
     * Remove a reference from the given project.
     *
     * @param projectId
     *         identifier of project.
     *
     * @param referenceId
     *         identifier of reference to remove.
     */
    void removeReference(long projectId, long referenceId);

    /**
     * Method to obtain all discussion of a given project.
     *
     * @param projectId
     *         identifier of project.
     * @return a list with all discussions of given project.
     */
    Set<Discussion> getDiscussions(long projectId);

    /**
     * Create a new discussion and add it to given identifier project.
     *
     * @param projectId identifier of project.
     *
     * @param discussion information of new discussion.
     *
     * @return the new discussion full filled.
     */
    Discussion addDiscussion(long projectId, Discussion discussion);

    /**
     * Add a new vote to an open discussion.
     *
     * @param discussionId identifier of discussion to add the vote.
     *
     * @param vote new vote to add.
     */
    void addVoteToDiscussion(long discussionId, Vote vote);

    /**
     * Close set a discussion.open to false.
     *
     * @param discussionId identifier of discussion to close.
     */
    void closeDiscussion(long discussionId);

    /**
     * Delete a discussion.
     *
     * @param discussionId identifier of the discussion.
     */
    void deleteDiscussion(long discussionId);

    /**
     * Remove a discussion from system, and therefore from their project.
     *
     * @param idDiscussion identifier of discussion to remove.
     */
    void removeDiscussion(long idDiscussion);

    /**
     * Method to obtain all milestones of project.
     *
     * @param projectId identifier of project.
     *
     * @return milestone of given identifier project.
     */

```

```
/*
Set<Milestone> getMilestones(long projectId);

/**
 * Create a new milestone.
 *
 * @param milestone new milestone.
 *
 * @return the new milestone full filled.
 */
Milestone addNewMilestone(long projectId, Milestone milestone);

/**
 * Remove a milestone of a project.
 *
 * @param projectId identifier of project that own the milestone.
 *
 * @param milestoneId identifier of milestone to remove.
 */
void removeMilestone(long projectId, long milestoneId);

/**
 * Change the date of a milestone.
 *
 * @param milestoneId identifier of milestone to change.
 *
 * @param newDate new date.
 */
void modifyMilestoneDate(long milestoneId, Date newDate);

/**
 * Set the last reached milestone.
 *
 * @param projectId project.
 *
 * @param milestone milestone information.
 */
void setLastMilestone(long projectId, long milestoneId);

/**
 *
 * @param projectId
 * @return
 */
Set<Task> getTasks(long projectId);

/**
 *
 * @param projectId
 * @param task
 * @return
 */
Task addNewTask(long projectId, Task task);

/**
 *
 * @param taskId
 */
void setTaskAsDone(long taskId);

/**
 *
 * @param taskId
 * @param hours
 */
void imputeHoursToTask(long taskId, int hours);

/**
 *
 * @param projectId
 * @param taskId
 */
void deleteTask(long taskId);

/**
 * Define a new knowledge area.

```

```

/*
 * @param knowledgeArea new knowledge area.
 *
 * @return the new knowledge area full filled.
 */
KnowledgeArea createKnowledgeArea(KnowledgeArea knowledgeArea);

/**
 * @return all available knowledge areas.
 */
List<KnowledgeArea> getAllKnowledgeAreas();

}

```

13.4.4 Ejemplos Capa de Presentación

Ejemplos de la capa de presentación, está se compone de vistas, presentadores, lógica de aplicación y un modelo de presentación.

13.4.4.1 Fichero “ProjectsListView.java”

```

package com.armandorv.cnpd.web.client.view.projects;

import java.util.List;

import com.armandorv.cnpd.web.client.presenter.projects.ProjectsListPresenter;
import com.armandorv.cnpd.web.client.view.util.ProgressView;
import com.armandorv.cnpd.web.client.view.util.builder.ColumnsBuilder;
import com.armandorv.cnpd.web.client.view.util.builder.FiltersBuilder;
import com.armandorv.cnpd.web.client.view.util.builder.MenuBuilder;
import com.armandorv.cnpd.web.client.view.util.properties.ProjectInfoProperties;
import com.armandorv.cnpd.web.shared.model.ProjectInfo;
import com.armandorv.cnpd.web.theme.client.icons(IconsBundle);
import com.google.gwt.core.client.GWT;
import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.uibinder.client.UiBinder;
import com.google.gwt.uibinder.client.UiFactory;
import com.google.gwt.uibinder.client.UiField;
import com.google.gwt.user.client.ui.Widget;
import com.sencha.gxt.cell.core.client.TextButtonCell;
import com.sencha.gxt.data.shared.ListStore;
import com.sencha.gxt.widget.core.client.Composite;
import com.sencha.gxt.widget.core.client.event.SelectEvent.HasSelectHandlers;
import com.sencha.gxt.widget.core.client.grid.ColumnModel;
import com.sencha.gxt.widget.core.client.grid.Grid;
import com.sencha.gxt.widget.core.client.grid.GridView;
import com.sencha.gxt.widget.core.client.menu.Item;
import com.sencha.gxt.widget.core.client.menu.MenuItem;
import com.sencha.gxt.widget.core.client.toolbar.PagingToolBar;

/**
 * Partial view of projects section, this widget is supposed to be inside a tab of
 * projects view section.
 *
 * @author armandorv
 *
 */
public class ProjectsListView extends Composite implements ProjectsListPresenter.Display
{
    public interface ProjectsListViewUiBinder extends UiBinder<Widget, ProjectsListView>
    {

        private static ProjectsListViewUiBinder uiBinder =
GWT.create(ProjectsListViewUiBinder.class);

        private static ProjectInfoProperties props = GWT.create(ProjectInfoProperties.class);

        private static IconsBundle icons = GWT.create(IconsBundle.class);
    }
}

```

```

private TextButtonCell actions = new TextButtonCell();

private MenuItem open = new MenuItem("Open");

private MenuItem leave = new MenuItem("Leave");

@UiField
ColumnModel<ProjectInfo> columnModel;

@UiField
ListStore<ProjectInfo> store;

@UiField
Grid<ProjectInfo> grid;

@UiField
GridView<ProjectInfo> view;

public ProjectsListView()
{
    super.initWidget(uiBinder.createAndBindUi(this));

    new FiltersBuilder<ProjectInfo>()
        .build(props.title())
        .build(props.area())
        .get(grid);
}

@UiFactory
ColumnModel<ProjectInfo> buildColumnModel()
{
    actions.setText("Actions");

    actions.setMenu(new MenuBuilder()
        .build(open, icons.show16())
        .build(leave, icons.exit16())
        .get());

    return new ColumnModel<ProjectInfo>(new ColumnsBuilder<ProjectInfo>()
        .build("Title", props.title(), 300, false)
        .build("Knowledge Area", props.area(), 250, true)
        .build("Actions", props.title(), 100, false, actions)
        .get());
}

@UiFactory
ListStore<ProjectInfo> buildStore()
{
    store = new ListStore<ProjectInfo>(props.id());
    return store;
}

@Override
public ProjectInfo getSelected(int row)
{
    return store.get(row);
}

@Override
public void addProject(ProjectInfo project)
{
    store.add(project);
}

@Override
public void setProjects(List<ProjectInfo> projects)
{
    store.clear();
    store.addAll(projects);
}

@Override
public HasSelectHandlers getActions()
{
    return this.actions;
}

```

```

    }

    @Override
    public void showProgress()
    {
        new ProgressDialog("Loading projects", "Loading ...", 3000).start();
    }

    @Override
    public void refresh()
    {
        view.refresh(false);
    }

    @Override
    public HasSelectionHandlers<Item> getOpen()
    {
        return open;
    }

    @Override
    public HasSelectionHandlers<Item> getLeave()
    {
        return leave;
    }

    @Override
    public void remove(ProjectInfo selected)
    {
        store.remove(selected);
    }
}

```

13.4.4.2 Fichero “ProjectsListPresenter.java”

```

package com.armandorv.cnfd.web.client.presenter.projects;

import java.util.List;

import javax.enterprise.event.Observes;
import javax.inject.Inject;
import javax.inject.Singleton;

import org.jboss.errai.bus.client.api.RemoteCallback;
import org.jboss.errai.ioc.client.api.AfterInitialization;
import org.jboss.errai.ioc.client.api.Caller;

import com.armandorv.cnfd.web.client.presenter.Presenter;
import com.armandorv.cnfd.web.client.presenter.projects.project.ProjectPresenter;
import com.armandorv.cnfd.web.client.presenter.util.BooleanMessenger;
import com.armandorv.cnfd.web.shared.exception.ClientsideException;
import com.armandorv.cnfd.web.shared.model.ProjectInfo;
import com.armandorv.cnfd.web.shared.model.UserInfo;
import com.armandorv.cnfd.web.shared.remote.ProjectsService;
import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.event.logical.shared.SelectionEvent;
import com.google.gwt.event.logical.shared.SelectionHandler;
import com.google.gwt.user.client.ui.Widget;
import com.sencha.gxt.widget.core.client.event.SelectEvent;
import com.sencha.gxt.widget.core.client.event.SelectEvent.HasSelectHandlers;
import com.sencha.gxt.widget.core.client.event.SelectEvent.SelectHandler;
import com.sencha.gxt.widget.core.client.menu.Item;

@Singleton
public class ProjectsListPresenter implements Presenter
{
    public interface Display
    {
        ProjectInfo getSelected(int row);

        Widget asWidget();

        void addProject(ProjectInfo project);
    }
}

```

```

void setProjects(List<ProjectInfo> projects);

void refresh();

HasSelectHandlers getActions();

HasSelectionHandlers<Item> getOpen();

HasSelectionHandlers<Item> getLeave();

void showProgress();

void remove(ProjectInfo selected);
}

@Inject
private Caller<ProjectsService> projectsService;

@Inject
private ProjectPresenter projectPresenter;

@Inject
private Display display;

private UserInfo user;

private ProjectInfo selected;

private boolean isStart = true;

@AfterInitialization
public void afterInitialization()
{
    display.getActions().addSelectHandler(select());
    display.getOpen().addSelectionHandler(open());
    display.getLeave().addSelectionHandler(leave());
}

@Override
public void present()
{
    if (user == null)
        throw new ClientSideException("A user must be set previously.");

    this.loadProjects();

    if (isStart)
    {
        display.showProgress();
        isStart = false;
    }
}

private void loadProjects()
{
    this.projectsService.call(new RemoteCallback<List<ProjectInfo>>()
    {
        public void callback(List<ProjectInfo> response)
        {
            display.setProjects(response);
            display.refresh();
        }
    }).getProjects(user.getId());
}

private SelectHandler select()
{
    return new SelectHandler()
    {
        @Override
        public void onSelect(SelectEvent event)
        {
            selected = display.getSelected(event.getContext().getIndex());
        }
    };
}

```

```

    };

    private SelectionHandler<Item> open()
    {
        return new SelectionHandler<Item>()
        {
            @Override
            public void onSelection(SelectionEvent<Item> event)
            {
                projectPresenter.setProject(selected);
                projectPresenter.present();
            }
        };
    }

    private SelectionHandler<Item> leave()
    {
        return new SelectionHandler<Item>()
        {
            @Override
            public void onSelection(SelectionEvent<Item> event)
            {
                projectsService.call(new RemoteCallback<Boolean>()
                {
                    @Override
                    public void callback(Boolean response)
                    {
                        BooleanMessenger.getMessenger(
                            "Project leaft.", "Error leaving project.")
                            .message(response);

                        if(response){
                            display.remove(selected);
                            display.refresh();
                        }
                    }
                }).leaveProject(user.getId(), selected.getId());
            }
        };
    }

    public void setUser(@Observes UserInfo user)
    {
        this.user = user;
    }
}

```

13.4.4.3 Fichero “ChatService.java”

```

package com.armandorv.cnpd.web.shared.remote;

import org.jboss.errai.bus.server.annotations.Remote;

/**
 * Service that hold chat communications related concerns.
 *
 * @author armandorv
 */
@Remote
public interface ChatService
{
    /**
     * Connect a user to the chat communications system.
     *
     * @return true if all OK.
     */
    boolean connect(String username);

    /**
     * Send a message to a contact.
     */
}

```

```
/*
 * @param gmailAddress
 *          gmail address of addressee contact.
 * @param sentText
 *          text of the message.
 * @return true.
 */
Boolean sendMessage(String username, String gmailAddressTo, String sentText);

/**
 * Disconnect a user from the chat communications system.
 *
 * @return true if all OK.
 */
boolean disconnectt(String username);

}
```

13.4.4.4 Fichero “*ChatMessage.java*”

```
package com.armandorv.cnpd.web.shared.model;

import org.jboss.errai.common.client.api.annotations.Portable;

/*
 * Model a chat message.
 */
@Portable
public class ChatMessage
{
    private String senderMail;

    private String text;

    public ChatMessage()
    {
    }

    public ChatMessage(String senderMail, String text)
    {
        super();
        this.senderMail = senderMail;
        this.text = text;
    }

    public String getSenderMail()
    {
        return senderMail;
    }

    public void setSenderMail(String senderMail)
    {
        this.senderMail = senderMail;
    }

    public String getText()
    {
        return text;
    }

    public void setText(String text)
    {
        this.text = text;
    }

    @Override
    public String toString()
    {
        return "ChatMessage [senderMail=" + senderMail + ", text=" + text + "]";
    }
}
```

```

@Override
public int hashCode()
{
    final int prime = 31;
    int result = 1;
    result = prime * result + ((senderMail == null) ? 0 : senderMail.hashCode());
    result = prime * result + ((text == null) ? 0 : text.hashCode());
    return result;
}

@Override
public boolean equals(Object obj)
{
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    ChatMessage other = (ChatMessage) obj;
    if (senderMail == null)
    {
        if (other.senderMail != null)
            return false;
    }
    else if (!senderMail.equals(other.senderMail))
        return false;
    if (text == null)
    {
        if (other.text != null)
            return false;
    }
    else if (!text.equals(other.text))
        return false;
    return true;
}
}

```

13.4.5 Ejemplos Cliente de Escritorio

13.4.5.1 Fichero “MainView.java”

```

package com.armandorv.cnpd.client.view;

import java.awt.BorderLayout;
import java.awt.Component;

import javax.inject.Singleton;
import javax.swing.JComponent;
import javax.swing.JScrollPane;

import org.jdesktop.application.Action;
import org.jdesktop.application.Application;
import org.jdesktop.application.ApplicationActionMap;
import org.jdesktop.application.ApplicationContext;
import org.jdesktop.swingx.JXDialog;
import org.jdesktop.swingx.JXPanel;
import org.jdesktop.swingx.JXTaskPane;
import org.jdesktop.swingx.JXTaskPaneContainer;

import com.armandorv.cnpd.client.presenter.ActionEventExecution;
import com.armandorv.cnpd.client.presenter.MainPresenter;

/**
 * View that show a task panel for users an another for project, and change the central
content
 * dependes on selected task.
 */

```

```

* @author armandorv
*
*/
@Singleton
public class MainView extends JXPanel implements MainPresenter.View
{
    private static final long serialVersionUID = -2194252038604033610L;

    private JXTTaskPaneContainer taskPaneContainer;

    private JXTTaskPane usersTaskPane;

    private JXTTaskPane areasTaskPane;

    private ActionExecution listUsers;

    private ActionExecution newUser;

    private ActionExecution newArea;

    private ActionExecution areas;

    public MainView()
    {
        super(new BorderLayout());

        taskPaneContainer = new JXTTaskPaneContainer();

        usersTaskPane = new JXTTaskPane();
        usersTaskPane.setTitle("Users");

        areasTaskPane = new JXTTaskPane();
        areasTaskPane.setTitle("Knowledge areas");

        taskPaneContainer.add(usersTaskPane);
        taskPaneContainer.add(areasTaskPane);

        this.bind();

        areasTaskPane.setCollapsed(true);

        super.add(new JScrollPane(taskPaneContainer));
    }

    public Component add(Component componenet)
    {
        if (taskPaneContainer.getComponentCount() == 3)
        {
            if (!(componenet instanceof JXDialog))
                taskPaneContainer.remove(2);
        }
        taskPaneContainer.add(componenet);
        taskPaneContainer.updateUI();

        return this;
    }

    private void bind()
    {
        ApplicationContext context = Application.getInstance().getContext();
        context.getResourceMap(getClass()).injectComponents(this);

        ApplicationActionMap map = context.getActionMap(this);

        usersTaskPane.add(map.get("Users List"));
        usersTaskPane.add(map.get("New User"));

        areasTaskPane.add(map.get("Areas List"));
        areasTaskPane.add(map.get("New area"));
    }

    @Action(name="Users List")
    public void listUsers()
    {
        listUsers.execute();
    }
}

```

```

    }

    @Action(name="New User")
    public void newUser()
    {
        newUser.execute();
    }

    @Action(name="Areas List")
    public void areas()
    {
        areas.execute();
    }

    @Action(name="New area")
    public void newArea()
    {
        newArea.execute();
    }

    @Override
    public JComponent asComponent()
    {
        return this;
    }

    @Override
    public void setListUsersActionExecution(ActionEventExecution listUsers)
    {
        this.listUsers = listUsers;
    }

    @Override
    public void setNewUserActionExecution(ActionEventExecution newUser)
    {
        this.newUser = newUser;
    }

    @Override
    public void setNewAreaActionExecution(ActionEventExecution newArea)
    {
        this.newArea = newArea;
    }

    @Override
    public void setAreasActionExecution(ActionEventExecution areas)
    {
        this.areas = areas;
    }
}

```

13.4.5.2 Fichero “MainPresenter.java”

```

package com.armandorv.cnpd.client.presenter;

import java.awt.Container;

import javax.annotation.PostConstruct;
import javax.inject.Inject;
import javax.inject.Singleton;
import javax.swing.JComponent;

/**
 * Presenter for global view of the application.
 *
 * @author armandorv
 */
@Singleton
public class MainPresenter implements Presenter
{
    public interface View

```

```
{  
    JComponent asComponent();  
  
    void setListUsersActionExecution(ActionEvent execution);  
  
    void setNewUserActionExecution(ActionEvent execution);  
  
    void setNewAreaActionExecution(ActionEvent execution);  
}  
  
@Inject  
private ListUsersPresenter listUsersPresenter;  
  
@Inject  
private NewUserPresenter newUserPresenter;  
  
@Inject  
private NewAreaPresenter newAreaPresenter;  
  
@Inject  
private AreasPresenter areasPresenter;  
  
@Inject  
private View view;  
  
@PostConstruct  
public void postConstruct()  
{  
    view.setListUsersActionExecution(listUsers());  
  
    view.setNewUserActionExecution(newUser());  
  
    view.setAreasActionExecution(areas());  
  
    view.setNewAreaActionExecution(newArea());  
}  
  
@Override  
public void present(Container container)  
{  
    container.add(view.asComponent());  
    listUsersPresenter.present(view.asComponent());  
}  
  
public ActionExecution listUsers()  
{  
    return new ActionExecution()  
    {  
        @Override  
        public void execute()  
        {  
            listUsersPresenter.present(view.asComponent());  
        }  
    };  
}  
  
private ActionExecution newArea()  
{  
    return new ActionExecution()  
    {  
        @Override  
        public void execute()  
        {  
            newAreaPresenter.present(view.asComponent());  
        }  
    };  
}  
  
private ActionExecution areas()  
{  
    return new ActionExecution()  
    {  
        @Override  
        public void execute()  
    };  
}
```

```

        {
            areasPresenter.present(view.asComponent());
        }
    }

    private ActionExecution newUser()
    {
        return new ActionExecution()
        {
            @Override
            public void execute()
            {
                newUserPresenter.present(view.asComponent());
            }
        };
    }
}

```

13.4.6 Ejemplos Patrones de Diseño

En este apartado se incluyen algunos ejemplos que aunque no son los más firmes representantes de ninguna parte de la arquitectura, si son buenos exponentes de las implementaciones de algunos patrones de diseño.

Algunos patrones como DAO, Fachada o Modelo Vista presentador se ven bien representados en los ejemplos anteriores.

13.4.6.1 *Ejemplo de Patrón Adapter*

```

package com.armandorv.cnpd.web.server.security;

import java.util.Collection;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.UserDetails;

import com.armandorv.cnpd.model.User;

/**
 * Custom implementation of UserDetails which adapt an user of my business
 * domain to the UserDetails interface.
 *
 * @author armandorv
 *
 */
public class UserDetailsAdaptor implements UserDetails
{
    private static final long serialVersionUID = -4388701527691863214L;

    private static final String ROLE_USER = "ROLE_USER";

    private User user;

    public UserDetailsAdaptor(User user)
    {
        this.user = user;
    }

    @Override
    public Collection<GrantedAuthority> getAuthorities()
    {
        return AuthorityUtils.createAuthorityList(ROLE_USER);
    }
}

```

```

@Override
public String getPassword()
{
    return user.getGoogleAccount().getPassword();
}

@Override
public String getUsername()
{
    return user.getGoogleAccount().getAccount();
}

@Override
public boolean isAccountNonExpired()
{
    return user != null;
}

@Override
public boolean isAccountNonLocked()
{
    return user != null;
}

@Override
public boolean isCredentialsNonExpired()
{
    return user != null;
}

@Override
public boolean isEnabled()
{
    return user != null;
}
}

```

13.4.6.2 Ejemplo de Patrón Singleton

```

package com.armandorv.cnpd.web.client.presenter.util;

import com.sencha.gxt.widget.core.client.info.Info;

/**
 * Class to show messages depends on boolean states.
 *
 * Messages are Info.display(Label, text)
 *
 * @author armandorv
 *
 */
public class BooleanMessenger
{
    private static BooleanMessenger messenger = new BooleanMessenger();

    /**
     * Label of Info.display when value is true.
     */
    private String trueLabel = "Success";

    /**
     * Message of Info.display when value is true.
     */
    private String trueMessage;
}

```

```

    /**
     * Label of Info.display when value is false.
     */
    private String falseLabel = "Fail";

    /**
     * Message of Info.display when value is false.
     */
    private String falseMessage;

    private BooleanMessenger()
    {
    }

    public static BooleanMessenger getMessenger(String trueMessage, String falseMessage)
    {
        messenger.trueMessage = trueMessage != null ? trueMessage : "";
        messenger.falseMessage = falseMessage != null ? falseMessage : "";

        return messenger;
    }

    public static BooleanMessenger getMessenger(String trueLabel, String trueMessage,
String falseLabel, String falseMessage)
    {
        messenger.trueLabel = trueLabel != null ? trueLabel : "";
        messenger.trueMessage = trueMessage != null ? trueMessage : "";
        messenger.falseLabel = falseLabel != null ? falseLabel : "";
        messenger.falseMessage = falseMessage != null ? falseMessage : "";

        return messenger;
    }

    public void message(Boolean response)
    {
        if (response)
        {
            Info.display(trueLabel, trueMessage);
        }
        else
        {
            Info.display(falseLabel, falseMessage);
        }
    }
}

```

13.4.6.3 Ejemplo de Patrón Translator

```

package com.armandorv.cnpd.web.server.mapper;

import javax.enterprise.context.Dependent;

import com.armandorv.cnpd.model.Message;
import com.armandorv.cnpd.web.shared.exception.MappingErrorException;
import com.armandorv.cnpd.web.shared.model.MessageInfo;
import com.armandorv.cnpd.web.shared.qualifiers.Messages;

@Messages
@Dependent
public class MessageMapper implements Mapper<Message, MessageInfo>
{
    @Override
    public MessageInfo map(Message object)
    {

        if (object.getId() == null)
            throw new MappingErrorException("Message must has a identifier.");

        MessageInfo info = new MessageInfo();

        info.setContent(object.getContent());
        info.setDate(object.getDate());
        info.setId(object.getId());
    }
}

```

```

        info.setRead(object.isRead());
        info.setSender(object.getSender().getGoogleAccount().getAccount());
        info.setTo(object.getAddressee().getGoogleAccount().getAccount());

        return info;
    }

}

```

13.4.6.4 Ejemplo de Patrón Builder

```

package com.armandorv.cnpd.web.client.view.util.builder;

import com.google.gwt.resources.client.ImageResource;
import com.sencha.gxt.widget.core.client.menu.Menu;
import com.sencha.gxt.widget.core.client.menu.MenuItem;

/**
 * Class to build GXT Menu instances, offer methods to execute an step by step building
process.
 *
 * @author armandorv
 *
 */
public class MenuBuilder
{
    private Menu menu = new Menu();

    /**
     * Build a menu item and add it to menu.
     *
     * @param item item to add.
     * @param icon icon of the item.
     * @param enabled if the item must be enabled.
     *
     * @return the builder instance to continue the process.
     */
    public MenuBuilder build(MenuItem item, ImageResource icon, boolean enabled)
    {
        item.setEnabled(enabled);

        if (icon != null)
            item.setIcon(icon);

        menu.add(item);

        return this;
    }

    /**
     * Build a menu item and add it to menu.
     *
     * @param item item to add.
     * @param icon icon of the item.
     *
     * @return the builder instance to continue the process.
     */
    public MenuBuilder build(MenuItem item, ImageResource icon)
    {
        return build(item, icon, true);
    }

    public MenuBuilder build(MenuItem item)
    {
        return build(item, null);
    }

    /**
     * @return Menu built during the building process.
     */
    public Menu get()
    {
        return menu;
    }
}

```

```

    }
}
```

13.4.7 Ejemplos Interceptores

Por último se muestran algunos ejemplos de cómo se han usado los mecanismos que ofrece JEE 6 para la *Programación Orientada a Aspectos*

13.4.7.1 Fichero “BusinessExceptionHandler.java”

```

package com.armandorv.cnpd.business.impl.interceptor;

import javax.interceptor.AroundInvoke;
import javax.interceptor.Interceptor;
import javax.interceptor.InvocationContext;

import org.apache.log4j.Logger;

import com.armandorv.cnpd.business.exception.BusinessException;
import com.armandorv.cnpd.persistence.exception.PersistenceException;

/**
 * AOP approach of Exception handling based on method intercepting, a subset of
 * Aspect oriented programming that is covered by JEE6 specifications set.
 *
 * AOP Definitions : The interceptor operation is the advice, all interceptors
 * annotated with HandleBusinessException compound the aspect of Business
 * Exceptions Handling, and code points annotated with HandleBusinessException
 * are join points. JEE6 hasn't yet a mechanism to define point cuts with regex
 * or similar.
 *
 * @author armandorv
 *
 */
@HandleBusinessException
@Interceptor
public class BusinessExceptionHandler
{
    private static Logger log = Logger.getLogger(BusinessExceptionHandler.class);

    @AroundInvoke
    public Object handleException(InvocationContext invocationContext)
        throws Exception
    {
        try
        {
            return invocationContext.proceed();
        }
        catch (BusinessException e)
        {
            throw e;
        }
        catch (PersistenceException e)
        {
            log.error(errorMessage(invocationContext, e));
            throw new BusinessException(errorMessage(invocationContext, e), e);
        }
        catch (Exception e)
        {
            log.error(errorMessage(invocationContext, e));
            throw new BusinessException(errorMessage(invocationContext, e), e);
        }
    }

    private String errorMessage(InvocationContext invocationContext, Exception e)
    {
        String message = "\n -- Error :" + e.getClass().getSimpleName() + " \n"
        + " -- Executing method :"
    }
}
```

```

        + invocationContext.getMethod().getName() + " at "
        + invocationContext.getMethod().getDeclaringClass().getName()
        + " \n";
    }

    message += " -- Parameters :";

    for (Object parm : invocationContext.getParameters())
        message += "Parameter [Tpye:" + parm.getClass().getSimpleName()
            + " Value :" + parm + "]";

    message += "\n -- Error message :" + e.getMessage();

    return message;
}

}

```

13.4.7.2 Fichero “HandleBusinessException.java”

```

package com.armandorv.cnfd.business.impl.interceptor;

import java.lang.annotation.ElementType;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

import javax.interceptor.InterceptorBinding;

@Inherited
@InterceptorBinding
@Target(
{ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
public @interface HandleBusinessException {

}

```

13.4.7.3 Fichero “DiscussionsSpecialist.java”

```

package com.armandorv.cnfd.business.impl.specialist;

import java.util.Set;

import javax.annotation.PostConstruct;
import javax.inject.Inject;

import com.armandorv.cnfd.business.impl.interceptor.HandleBusinessException;
import com.armandorv.cnfd.business.impl.util.FindByIdExecutor;
import com.armandorv.cnfd.model.Discussion;
import com.armandorv.cnfd.model.Project;
import com.armandorv.cnfd.model.Vote;
import com.armandorv.cnfd.persistence.IDiscussionDao;
import com.armandorv.cnfd.persistence.IProjectDao;

/**
 * Specialist on deal with discussions, is used by projects manager and must be
 * within persistent context and be injected with @Inject.
 *
 * @author armandorv
 */
@HandleBusinessException
public class DiscussionsSpecialist
{
    @Inject
    private IDiscussionDao discussionDao;

    @Inject
    private IProjectDao projectDao;
}

```

```
private FindByIdExecutor<Project> findProjectById;
private FindByIdExecutor<Discussion> findDiscussionById;

@PostConstruct
public void setUp()
{
    findProjectById = new FindByIdExecutor<Project>(projectDao);
    findDiscussionById = new FindByIdExecutor<Discussion>(discussionDao);
}

public Set<Discussion> getDiscussions(long projectId)
{
    Project project = findProjectById.findById(projectId);

    Set<Discussion> discussions = project.getDiscussions();
    discussions.size();

    return discussions;
}

public Discussion addDiscussion(long projectId, Discussion discussion)
{
    Project project = findProjectById.findById(projectId);

    discussionDao.persist(discussion);
    discussion.setProject(project);

    return discussion;
}

public void removeDiscussion(long discussionId)
{
    Discussion discussion = findDiscussionById.findById(discussionId);
    discussionDao.remove(discussion);
}

public void addVoteToDiscussion(long discussionId, Vote vote)
{
    Discussion discussion = findDiscussionById.findById(discussionId);
    discussion.getVotes().add(vote);
}

public void closeDiscussion(long discussionId)
{
    Discussion discussion = findDiscussionById.findById(discussionId);
    discussion.setOpen(false);
}

public void deleteDiscussion(long discussionId)
{
    Discussion discussion = findDiscussionById.findById(discussionId);
    discussionDao.remove(discussion);
}

}
```