

**UNIVERSIDAD DE ALCALÁ**



**Escuela Técnica Superior de Ingeniería  
Informática**  
**Ingeniería Informática**

**Trabajo Fin de Carrera**

**INTEGRACIÓN DEL ENTORNO DE DESARROLLO  
PARA UN PROYECTO  
DE SOFTWARE LIBRE**

**Roberto Carlos Zapatera Pilo**  
**2009**



UNIVERSIDAD DE ALCALÁ  
Escuela Técnica Superior de Ingeniería Informática  
INGENIERÍA INFORMÁTICA

---

# Trabajo Fin de Carrera

INTEGRACIÓN DEL ENTORNO DE DESARROLLO PARA UN  
PROYECTO  
DE SOFTWARE LIBRE

**Autor:** Roberto Carlos Zapatera Pilo

**Director:** José Javier Martínez Herráiz

---

Tribunal:

Presidente:

Vocal 1º:

Vocal 2º:

Calificación: \_\_\_\_\_

Alcalá de Henares a      de      de



*A Lucia por estar siempre ahí.  
A Pedro, por sus correcciones, ayuda  
y dedicación.*



# ÍNDICE RESUMIDO

---

1	INTRODUCCIÓN .....	1
2	RESUMEN.....	5
3	OBJETIVOS DEL PROYECTO .....	9
4	RESULTADOS .....	13
5	ORGANIZACIÓN DE LA MEMORIA .....	17
6	INSTALACIÓN DEL SISTEMA KYOSEI-POLIS .....	21
7	DESARROLLO APLICACIONES .....	183
8	CONCLUSIÓN.....	293
9	FUTURAS LÍNEAS DE TRABAJO.....	297
10	PRESUPUESTO .....	301
11	BIBLIOGRAFÍA .....	309
12	GLOSARIO DE TÉRMINOS.....	313

---



# ÍNDICE DETALLADO

---

<b>1</b>	<b>INTRODUCCIÓN .....</b>	<b>1</b>
<b>2</b>	<b>RESUMEN.....</b>	<b>5</b>
<b>3</b>	<b>OBJETIVOS DEL PROYECTO .....</b>	<b>9</b>
<b>4</b>	<b>RESULTADOS .....</b>	<b>13</b>
<b>5</b>	<b>ORGANIZACIÓN DE LA MEMORIA .....</b>	<b>17</b>
<b>6</b>	<b>INSTALACIÓN DEL SISTEMA KYOSEI-POLIS .....</b>	<b>21</b>
6.1.	INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR CKYOSEI .....	24
6.1.1.	<i>Preparación del entorno.....</i>	24
6.1.2.	<i>Instalación del Servidor de BBDD Mysql.....</i>	25
6.1.2.1.	Introducción .....	25
6.1.2.2.	Requerimientos.....	25
6.1.2.3.	Proceso de instalación.....	25
6.1.3.	<i>Opcional: Creación de la BBDD de Spring-Security.....</i>	32
6.1.4.	<i>Instalación de Java Development Kit.....</i>	36
6.1.4.1.	Introducción .....	36
6.1.4.2.	Prerrequisitos.....	37
6.1.4.3.	Proceso de instalación.....	37
6.1.4.4.	Configuración de variables de entorno.....	37
6.1.5.	<i>Instalación de TOMCAT.....</i>	39
6.1.5.1.	Introducción .....	39
6.1.5.2.	Requerimientos.....	39
6.1.6.	<i>Proceso de instalación .....</i>	39
6.1.7.	<i>Instalación de Apache Web Server.....</i>	43
6.1.7.1.	Introducción .....	43
6.1.7.2.	Requerimientos.....	43
6.1.7.3.	Proceso de Instalación.....	43
6.1.8.	<i>Configuración del conector Apache – Tomcat.....</i>	47
6.1.8.1.	Introducción .....	47
6.1.8.2.	Requerimientos.....	47
6.1.9.	<i>Proceso de Configuración.....</i>	47
6.1.9.1.	OPCIONAL: Prueba de configuración Apache - Tomcat.....	49
6.1.10.	<i>Instalación de PHP para Apache.....</i>	59
6.1.10.1.	Introducción .....	59
6.1.10.2.	Requerimientos.....	59
6.1.10.3.	Proceso de Configuración .....	59
6.1.10.4.	OPCIONAL: Prueba de Configuración PHP .....	61
6.1.11.	<i>Configuración seguridad para Apache.....</i>	63
6.1.11.1.	Introducción .....	63
6.1.11.2.	Secure Sockets Layer (SSL) .....	63
6.1.11.3.	Proceso de instalación.....	66
6.1.11.4.	Instalación de OpenSSL.....	73
6.1.11.5.	Instalación de Active Perl.....	76
6.1.12.	<i>Creación Entidades Certificadoras y certificados firmados .....</i>	77
6.1.12.1.	Introducción .....	77

6.1.12.2.	Implementación de una Autoridad Certificadora.....	77
6.1.12.3.	Creación de Certificados .....	82
6.1.12.4.	Creación de un Certificado de cliente.....	85
6.1.12.5.	Añadir el certificado al navegador .....	90
6.1.12.6.	OPCIONAL: Aplicativo de prueba SSL.....	92
6.1.13.	<i>Instalación de Subversion e integración con Apache 2.2.x.</i> .....	97
6.1.13.1.	Introducción: Sistemas de control de versiones.....	97
6.1.13.2.	Requerimientos.....	97
6.1.13.3.	Proceso de instalación .....	97
6.1.13.4.	Creación de repositorios .....	99
6.1.13.5.	Integración SVN-Apache 2.2.x.....	100
6.1.14.	<i>Instalación de Maven</i> .....	105
6.1.14.1.	Introducción .....	105
6.1.14.2.	Requerimientos.....	105
6.1.14.3.	Instalación .....	105
6.1.14.4.	Inicialización repositorio local de Maven.....	105
6.1.14.5.	Creación de un repositorio Maven interno (Integración con Apache).....	106
6.1.14.6.	Acceso al repositorio interno .....	110
6.1.15.	<i>Instalación de servidor de correo James Server</i> .....	111
6.1.15.1.	Introducción .....	111
6.1.15.2.	Requerimientos.....	111
6.1.15.3.	Proceso de instalación .....	111
6.1.15.4.	Proceso de configuración.....	112
6.1.15.5.	Prueba del servidor SMTP.....	113
6.2.	INSTALACIÓN Y CONFIGURACIÓN DEL ENTORNO DE DESARROLLO.....	117
6.2.1.	<i>Preparación del entorno</i> .....	117
6.2.2.	<i>Instalación de Java Development Kit</i> .....	118
6.2.3.	<i>Instalación de Maven</i> .....	119
6.2.3.1.	Instalación de Librerías en el repositorio.....	119
6.2.3.2.	Creación de <i>releases</i> o versiones.....	122
6.2.3.3.	Ejemplo Práctico.....	126
6.2.4.	<i>Instalación de Ant</i> .....	133
6.2.4.1.	Introducción .....	133
6.2.4.2.	Requerimientos.....	133
6.2.4.3.	Instalación .....	133
6.2.5.	<i>Instalación de Eclipse</i> .....	135
6.2.5.1.	Introducción .....	135
6.2.5.2.	Requerimientos.....	135
6.2.5.3.	Proceso de instalación .....	135
6.2.5.4.	OPCIONAL: Prueba de eclipse .....	136
6.2.6.	<i>Instalación de plugins de Eclipse</i> .....	143
6.2.6.1.	Instalación de Subclipse plugin en Eclipse 3.4.x.....	143
6.2.6.2.	Instalación de SQL Explorer plugin en Eclipse 3.4.x .....	146
6.2.6.3.	Instalación de Maven plugin en Eclipse 3.4.x .....	150
6.2.6.4.	OPCIONAL: Configuración y prueba del plugin .....	151
6.2.6.5.	Configuración de Ant en Eclipse.....	152
6.3.	INSTALACIÓN Y CONFIGURACIÓN DE FRAMEWORKS .....	156
6.3.1.	<i>Instalación de Hibernate</i> .....	157
6.3.1.1.	Requerimientos.....	157
6.3.1.2.	Proceso de instalación .....	157
6.3.1.3.	OPCIONAL: Proceso de inclusión de Hibernate en aplicaciones.....	157
6.3.1.4.	Instalación de Hibernate tools en Eclipse 3.4.x .....	159
6.3.1.5.	OPCIONAL: Configuración y prueba de Hibernate .....	160

6.3.2.	<i>Instalación de Spring</i> .....	168
6.3.2.1.	Requerimientos.....	168
6.3.2.2.	Proceso de instalación.....	168
6.3.2.3.	OPCIONAL: Proceso de inclusión de Spring en aplicaciones.....	168
6.3.3.	<i>Sistemas de Trazas</i> .....	170
6.3.3.1.	Introducción .....	170
6.3.3.2.	Log4j .....	170
6.3.3.3.	java.util.logging.....	179
6.3.3.4.	Jakarta-Commons-Logging (JCL) .....	181
<b>7</b>	<b>DESARROLLO APLICACIONES .....</b>	<b>183</b>
7.1.	TECNOLOGÍAS EMPLEADAS EN EL DESARROLLO.....	186
7.2.	METODOLOGÍA EMPLEADA.....	189
7.2.1.	<i>Principios del Desarrollo Ágil</i> .....	189
7.2.2.	<i>Características de un método ágil</i> .....	190
7.2.3.	<i>¿Qué es eXtreme Programming?</i> .....	190
7.2.4.	<i>¿Por qué eXtreme Programming?</i> .....	190
7.2.5.	<i>Cuando usar la Metodología XP</i> .....	191
7.2.6.	<i>Trabajando con XP</i> .....	192
7.2.6.1.	Fases del la Metodología XP .....	192
7.3.	LA APLICACIÓN ALPHA.....	195
7.3.1.	<i>Historias de usuario</i> .....	198
7.3.2.	<i>Aplicando XP y AMDD al arquetipo</i> .....	205
7.3.2.1.	Arquitectura de las futuras aplicaciones diseñadas a partir del arquetipo. ....	205
7.3.2.2.	Estructura del arquetipo. ....	211
7.3.2.3.	Modelo de dominio.....	213
7.3.2.4.	Prototipos de interfaces de usuario .....	215
7.3.2.5.	Storyboard.....	227
7.3.2.6.	Enfoque y diseño del Artefacto .....	228
7.3.2.7.	Arquitectura .....	228
7.3.2.8.	CRC Cards .....	233
7.3.2.9.	Mapa de flujo Aplicación Inicial generada con el Arquetipo.....	238
7.3.2.10.	Diagramas de Clase UML.....	238
7.3.2.11.	UML Package Diagram.....	246
7.3.2.12.	Creación de Test y Aceptación .....	248
7.3.2.13.	Creación de Arquetipos en Maven .....	249
7.3.2.14.	Creación de un Arquetipo para el Sistema Kyosei-Polis .....	249
7.3.3.	<i>Aplicando XP y AMDD a la librería de utilidades</i> .....	266
7.3.3.1.	Arquitectura de la librería.....	266
7.3.3.2.	CRC Cards .....	268
7.3.3.3.	Diagramas de Clase UML .....	274
7.3.4.	<i>Aplicando XP y AMDD a la aplicación alpha</i> .....	280
7.3.4.1.	Arquitectura de la aplicación .....	280
7.3.4.2.	Modelo de dominio.....	281
7.3.4.3.	Prototipos de interfaces de usuario .....	282
7.3.4.4.	Storyboard.....	291
7.3.4.5.	Plan de Entregas .....	292
<b>8</b>	<b>CONCLUSIÓN .....</b>	<b>293</b>
<b>9</b>	<b>FUTURAS LÍNEAS DE TRABAJO.....</b>	<b>297</b>

<b>10 PRESUPUESTO .....</b>	<b>301</b>
10.1.    INTRODUCCIÓN.....	304
10.2.    PRESUPUESTO DE EJECUCIÓN MATERIAL .....	305
10.3.    GASTOS GENERALES Y BENEFICIO INDUSTRIAL.....	307
10.3.1. <i>Presupuesto de Ejecución por Contrata</i> .....	307
10.3.2. <i>Honorarios</i> .....	307
10.4.    IMPORTE TOTAL DEL PRESUPUESTO.....	308
<b>11 BIBLIOGRAFÍA .....</b>	<b>309</b>
<b>12 GLOSARIO DE TÉRMINOS.....</b>	<b>313</b>

# ÍNDICE DE FIGURAS Y TABLAS

---

<b>1 INTRODUCCIÓN .....</b>	<b>1</b>
<b>2 RESUMEN.....</b>	<b>5</b>
<b>3 OBJETIVOS DEL PROYECTO .....</b>	<b>9</b>
<b>4 RESULTADOS .....</b>	<b>13</b>
<b>5 ORGANIZACIÓN DE LA MEMORIA .....</b>	<b>17</b>
<b>6 INSTALACIÓN DEL SISTEMA KYOSEI-POLIS .....</b>	<b>21</b>
6.1.      INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR CKYOSEI .....	24
6.1.1. <i>Preparación del entorno.....</i>	24
6.1.2. <i>Instalación del Servidor de BBDD Mysql.....</i>	25
6.1.2.1.    Introducción .....	25
6.1.2.2.    Requerimientos.....	25
6.1.2.3.    Proceso de instalación.....	25
Figura 1 Instalación MySQL Server.....	25
Figura 2 Selección de componentes .....	26
Figura 3 Confirmación Instalación MYSQL .....	26
Figura 4 Configuración del Servidor.....	27
Figura 5 Selección de Configuración .....	27
Figura 6 Configuración como servicio Windows.....	28
Figura 7 Configuración usuario administrador .....	28
Figura 8 Confirmación instalación .....	29
Figura 9 Aviso seguridad de usuario root de MySQL .....	29
Figura 10 Error de conexión TCP de MySQL.....	30
Figura 11 Apertura de puerto de MySQL en el firewall de Windows .....	30
Figura 12 Pantalla de finalización de la instalación de MySQL .....	31
6.1.3. <i>Opcional: Creación de la BBDD de Spring-Security.....</i>	32
Figura 13 Modelo lógico de autenticación de Spring security .....	32
Tabla 1 Instrucciones MySQL.....	33
Tabla 2 Script de creación de BBDD Spring Security .....	35
6.1.4. <i>Instalación de Java Development Kit.....</i>	36
6.1.4.1.    Introducción .....	36
Figura 14 Ciclo vida generación código Java .....	36
6.1.4.2.    Prerrequisitos.....	37
6.1.4.3.    Proceso de instalación.....	37
Figura 15 Instalación JSRDK .....	37
6.1.4.4.    Configuración de variables de entorno.....	37
6.1.5. <i>Instalación de TOMCAT.....</i>	39
6.1.5.1.    Introducción .....	39
6.1.5.2.    Requerimientos.....	39
6.1.6. <i>Proceso de instalación .....</i>	39
Figura 16 Pantalla instalación Tomcat .....	39
Figura 17 Aceptar licencia .....	40
Figura 18 Selección de componentes .....	40
Figura 19 Directorio de instalación .....	41
Figura 20 Selección de puerto y usuario administrador .....	41
Figura 21 Selección de JRE para Tomcat .....	42
6.1.7. <i>Instalación de Apache Web Server.....</i>	43
6.1.7.1.    Introducción .....	43
6.1.7.2.    Requerimientos.....	43
6.1.7.3.    Proceso de Instalación.....	43
Figura 22 Pantalla Instalación Apache.....	43

Figura 23 Pantalla de configuración Apache .....	44
Figura 24 Tipo de instalación de Apache.....	45
Figura 25 Selección de componentes de Apache .....	45
<b>6.1.8.     <i>Configuración del conector Apache – Tomcat</i>.....</b>	<b>47</b>
6.1.8.1.     Introducción .....	47
6.1.8.2.     Requerimientos.....	47
<b>6.1.9.     <i>Proceso de Configuración</i>.....</b>	<b>47</b>
Tabla 3 workers.properties.....	48
Tabla 4 server.xml .....	48
Tabla 5 inclusión en httpd.conf de mod_jk.conf .....	49
Tabla 6 Mapeo de URL en el archivo mod_jk.conf.....	49
6.1.9.1.     OPCIONAL: Prueba de configuración Apache - Tomcat .....	49
Tabla 7 Fichero host de Windows.....	50
Tabla 8 Permisos para el directorio de publicación www del fichero mod_jk.conf.....	51
Tabla 9 Creación de un Virutal Host en mod_jk.conf.....	52
Tabla 10 Fichero completo mod_jk.conf .....	54
Tabla 11 Restringir permisos sobre directorio raíz de Apache en el httpd.conf .....	54
Tabla 12 JSP de prueba de configuración Apache-Tomcat .....	55
Figura 26 Denegación de servicio directorio raíz apache .....	55
Figura 27 Denegación de servicio directorio raíz apache .....	55
Tabla 13 Html de bienvenida que reenvia la información a la JSP.....	56
Figura 28 página de interconexión Apache Tomcat.....	56
Tabla 14 Información del Log alpha.com.localhost-access.log .....	57
Figura 29 Llamada directa a Tomcat.....	57
6.1.10. <i>Instalación de PHP para Apache</i> .....	59
6.1.10.1.     Introducción .....	59
6.1.10.2.     Requerimientos.....	59
6.1.10.3.     Proceso de Configuración.....	59
Figura 30 Configuración de PHP en httpd.conf .....	60
6.1.10.4.     OPCIONAL: Prueba de Configuración PHP .....	61
Figura 31 Respuesta Servidor Apache a index.php.....	62
6.1.11. <i>Configuración seguridad para Apache</i> .....	63
6.1.11.1.     Introducción .....	63
6.1.11.2.     Secure Sockets Layer (SSL) .....	63
Figura 32 SL sub-protocols in the TCP/IP model (Imagen SecurityFocus) .....	64
Figura 33 Establecimiento de Conexión SSL paso a paso (Imagen securityfocus) .....	65
6.1.11.3.     Proceso de instalación .....	66
Figura 34 Aviso Conexión segura .....	66
Tabla 15 CKyosei_SSL_script.bat .....	68
Tabla 16 httpd-ssl.conf.....	69
Tabla 17 Directivas SSL mod_jk.conf .....	70
Tabla 18 Fichero resultante mod_jk.conf .....	71
Tabla 19 Directivas SSL mod_jk.conf .....	72
6.1.11.4.     Instalación de OpenSSL .....	73
Figura 35 Pantalla de selección de directorio para OpenSSL.....	74
Figura 36 Selección de tareas adicionales de OpenSSL .....	75
6.1.11.5.     Instalación de Active Perl.....	76
Figura 37 Pantalla instalación ActivePerl.....	76
6.1.12. <i>Creación Entidades Certificadoras y certificados firmados</i> .....	77
6.1.12.1.     Introducción .....	77
6.1.12.2.     Implementación de una Autoridad Certificadora.....	77
Tabla 20 Configuración por defecto OpenSSL.conf .....	79
Figura 38 Instalación del certificado de la entidad certificadora en Windows. ....	80
Figura 39 Importación de entidad certificadora .....	81
Figura 40 Instalación del certificado de la entidad certificadora en windows. ....	81
Figura 41 Información del certificado reconocido por Windows .....	82
6.1.12.3.     Creación de Certificados .....	82
Figura 42 Fichero de configuración SSL de Apache .....	85
6.1.12.4.     Creación de un Certificado de cliente.....	85
Tabla 21 Configuración openssl.conf con los nuevos OIDs.....	88
6.1.12.5.     Añadir el certificado al navegador .....	90
Figura 43 Ver certificados del navegador .....	90
Figura 44 Importación de certificados .....	90
Figura 45 En firefox pide contraseña maestra almacen de certificados .....	91
Figura 46 Contraseña de cifrado del fichero de certificados .....	91
Figura 47 Certificado importado en el navegador.....	91

Figura 48 Información del certificado OIDs Creados.....	92
<b>6.1.12.6. OPCIONAL: Aplicativo de prueba SSL.....</b>	<b>92</b>
Tabla 22 Fichero httpd-ssl.conf .....	93
Figura 49 Diagrama de dependencias del paquete org.ckyosei.demossll.ssl .....	94
Figura 50 Diagrama Clases paquete org.ckyosei.demossll.web.....	94
Figura 51 Diagrama de Secuencia del método de negocio de Director .....	95
Figura 52 Diagrama de Secuencia del método del Controlador .....	95
Figura 53 Petición de identificación mediante certificado Cliente del navegador.....	96
Figura 54 Información del certificado autenticado .....	96
<b>6.1.13. Instalación de Subversion e integración con Apache 2.2.x.....</b>	<b>97</b>
6.1.13.1. Introducción: Sistemas de control de versiones .....	97
6.1.13.2. Requerimientos.....	97
6.1.13.3. Proceso de instalación.....	97
Figura 55 TortoiseSVN en el explorador Windows.....	98
Figura 56 Pantalla instalación TortoiseSVN.....	98
6.1.13.4. Creación de repositorios.....	99
Figura 57 Creación de repositos medianta TortoiseSVN.....	99
6.1.13.5. Integración SVN-Apache 2.2.x.....	100
Tabla 23 Fichero configuración permisos de usuarios Subversion svn-acl .....	102
Tabla 24 Configuración de acceso a los Repositorios en Apache .....	103
Figura 58 Acceso a un repositorio.....	104
Figura 59 Pantalla de bienvenida del repositorio .....	104
<b>6.1.14. Instalación de Maven .....</b>	<b>105</b>
6.1.14.1. Introducción .....	105
6.1.14.2. Requerimientos.....	105
6.1.14.3. Instalación.....	105
6.1.14.4. Inicialización repositorio local de Maven.....	105
Figura 60 Repositorio local de Maven .....	106
6.1.14.5. Creación de un repositorio Maven interno (Integración con Apache) .....	106
Tabla 25 Fichero configuración Apache inclusión librerías WebDav .....	107
Tabla 26 Inclusión fichero configuración (extra Apache) WebD .....	108
Tabla 27 Fichero httpd-dav.conf.....	109
6.1.14.6. Acceso al repositorio interno .....	110
Figura 61 Acceso a repositorio interno .....	110
Figura 62 Repositorio interno .....	110
<b>6.1.15. Instalación de servidor de correo James Server .....</b>	<b>111</b>
6.1.15.1. Introducción .....	111
6.1.15.2. Requerimientos.....	111
6.1.15.3. Proceso de instalación.....	111
Figura 63 Directorio de instalación de Apache James Server .....	111
6.1.15.4. Proceso de configuración .....	112
Tabla 28 Fichero configuración Apache James config.xml.....	113
Tabla 29 Fichero configuración DNS resolv.conf.....	113
6.1.15.5. Prueba del servidor SMTP .....	113
<b>6.2. INSTALACIÓN Y CONFIGURACIÓN DEL ENTORNO DE DESARROLLO .....</b>	<b>117</b>
6.2.1. Preparación del entorno.....	117
6.2.2. Instalación de Java Development Kit.....	118
6.2.3. Instalación de Maven .....	119
6.2.3.1. Instalación de Librerías en el repositorio.....	119
Tabla 30 Pom.xml para la importación de librerías de 3os.....	120
Tabla 31 Fichero de configuración setting.xml donde configuramos los servidores.....	121
Tabla 32 Fragmento del fichero setting.xml de Maven donde se configura el repositorio local .....	122
6.2.3.2. Creación de releases o versiones .....	122
Tabla 32 Fragmento de un Fichero Pom.xml donde se configura las distribuciones.....	123
Tabla 33 Fragmento de un fichero Pomxml donde se configura el SCM .....	124
Tabla 34 Fragmento de un fichero Pomxml donde se configura los plugins.....	125
6.2.3.3. Ejemplo Práctico .....	126
Tabla 35 Pom.xml totalmente configurado.....	129
Figura 64 Repositorio de versiones con el artefacto generado .....	131
Figura 65 Repositorio de Documentación .....	132
Figura 66 Página Web de documentación generada. ....	132
6.2.4. Instalación de Ant .....	133
6.2.4.1. Introducción .....	133

6.2.4.2.	Requerimientos.....	133
6.2.4.3.	Instalación .....	133
6.2.5.	<i>Instalación de Eclipse</i> .....	135
6.2.5.1.	Introducción .....	135
6.2.5.2.	Requerimientos.....	135
6.2.5.3.	Proceso de instalación .....	135
	Figura 67 Selección Workspace Eclipse .....	136
6.2.5.4.	OPCIONAL: Prueba de eclipse .....	136
	Figura 68 Bienvenida Eclipse .....	136
	Figura 69 Vistas de Eclipse .....	137
	Figura 70 WorkSpace de Eclipse .....	137
	Figura 71 Selección Proyecto .....	138
	Figura 72 Creación proyecto Java .....	139
	Figura 73 Selección de fuentes.....	140
	Figura 74 Creación de una Clase.....	141
	Figura 75 Área de código de Eclipse .....	141
	Figura 76 Ejecución de la aplicación .....	142
6.2.6.	<i>Instalación de plugins de Eclipse</i> .....	143
6.2.6.1.	Instalación de Subclipse plugin en Eclipse 3.4.x.....	143
	Figura 77 Actualización Plugins Eclipse .....	143
	Figura 78 Selección plugin Subversion.....	144
	Figura 79 Perspectiva de Eclipse para Subversion.....	144
	Figura 80 Creación de un nuevo acceso a repositorio.....	145
	Figura 81 URL acceso a repositorio.....	145
	Figura 82 Explorador de Repositorio .....	146
6.2.6.2.	Instalación de SQL Explorer plugin en Eclipse 3.4.x .....	146
	Figura 83 Selección proyecto SQLExplorer en eclipse .....	147
	Figura 84 Selección perspectiva SQLExplorer en eclipse.....	148
	Figura 85 Selección de driver .....	148
	Figura 86 Configuración de conexión.....	149
	Figura 87 Establecimiento de conexión .....	150
6.2.6.3.	Instalación de Maven plugin en Eclipse 3.4.x .....	150
	Figura 88 Selección de JRE de eclipse .....	151
6.2.6.4.	OPCIONAL: Configuración y prueba del plugin.....	151
6.2.6.5.	Configuración de Ant en Eclipse.....	152
	Tabla 36 Fichero build.xml.....	154
	Figura 89 Ejecución de Ant desde eclipse .....	155
6.3.	INSTALACIÓN Y CONFIGURACIÓN DE FRAMEWORKS .....	156
6.3.1.	<i>Instalación de Hibernate</i> .....	157
6.3.1.1.	Requerimientos.....	157
6.3.1.2.	Proceso de instalación .....	157
6.3.1.3.	OPCIONAL: Proceso de inclusión de Hibernate en aplicaciones.....	157
	Tabla 37 Fichero Manifest .....	158
6.3.1.4.	Instalación de Hibernate tools en Eclipse 3.4.x .....	159
	Figura 90 Wizard de hibernate .....	160
6.3.1.5.	OPCIONAL: Configuración y prueba de Hibernate .....	160
	Figura 91 Selección de directorio para la Configuración de Hibernate .....	161
	Figura 92 Configuración de la base de datos.....	161
	Tabla 38 Fichero Configuración hibernate .....	162
	Figura 93 Con sola de hibernate .....	163
	Figura 94 Vista de la consola de Hibernate de la base de datos .....	163
	Tabla 39 Fichero Revenge.xml hibernate .....	167
6.3.2.	<i>Instalación de Spring</i> .....	168
6.3.2.1.	Requerimientos.....	168
6.3.2.2.	Proceso de instalación .....	168
6.3.2.3.	OPCIONAL: Proceso de inclusión de Spring en aplicaciones .....	168
6.3.3.	<i>Sistemas de Trazas</i> .....	170
6.3.3.1.	Introducción .....	170
6.3.3.2.	Log4j.....	170
	Tabla 40 Clase de Test log4j.....	172
	Tabla 41 log4j.properties .....	173
	Tabla 42 Fichero log4j.properties.....	174
	Tabla 43 Fichero log4j.xml.....	174
	Tabla 44 Ejemplo Log 1.....	176

6.3.3.3.	Tabla 45 Log4j ejemplo 2 .....	178
6.3.3.3.	java.util.logging.....	179
	Tabla 46 Programa básico de Log .....	180
6.3.3.4.	Jakarta-Commons-Logging (JCL) .....	181
	Tabla 47 common-logging.properties .....	182
<b>7</b>	<b>DESARROLLO APLICACIONES .....</b>	<b>183</b>
7.1.	TECNOLOGÍAS EMPLEADAS EN EL DESARROLLO.....	186
7.2.	METODOLOGÍA EMPLEADA. ....	189
7.2.1.	<i>Principios del Desarrollo Ágil.</i> .....	189
7.2.2.	<i>Características de un método ágil.</i> .....	190
7.2.3.	<i>¿Qué es eXtreme Programming?</i> .....	190
7.2.4.	<i>¿Por qué eXtreme Programming?.....</i>	190
7.2.5.	<i>Cuando usar la Metodología XP.</i> .....	191
	Figura 95 Ciclo de vida de XP .....	191
7.2.6.	<i>Trabajando con XP .....</i>	192
	Figura 96 Fases de la metodología XP .....	192
	7.2.6.1. <i>Fases del la Metodología XP .....</i>	192
	Figura 97 Fase de XP en detalle .....	192
7.3.	LA APLICACIÓN ALPHA.....	195
	Figura 98 Problemática Ckyosei.....	195
7.3.1.	<i>Historias de usuario.....</i>	198
	Figura 99 Esquema reunión Skype.....	199
	Figura 100 Diagrama generación historias de usuario .....	201
7.3.2.	<i>Aplicando XP y AMDD al arquetipo.....</i>	205
7.3.2.1.	Arquitectura de las futuras aplicaciones diseñadas a partir del arquetipo. .....	205
	Figura 101 APLICACIONES MEDIANTE J2EE.....	205
	Figura 102 Esquema funcioneamiento J2EE .....	206
	Figura 103 Patrones de Diseño Blueprint .....	207
	Figura 104 MVC DE SPRING .....	208
	Figura 105 Modelo de capas de Aplicaciones con Spring .....	209
	Figura 106 Modelo de Capas Spring - Hibernate .....	210
7.3.2.2.	Estructura del arquetipo .....	211
	Figura 107 Estructura del arquetipo Ckyosei.....	211
7.3.2.3.	Modelo de dominio.....	213
	Figura 108 Modelo de dominio de Menús .....	213
	Figura 109 Estructura de menús horizontales usados para navegación global.....	214
	Figura 110 Estructura de menús verticales usados para navegación local .....	214
	Figura 111 Modelo lógico de autenticación de Spring security .....	215
	Figura 112 Modelo de dominio de Autenticación .....	215
7.3.2.4.	Prototipos de interfaces de usuario .....	215
	Figura 113 Prototipo de la pantalla principal de las futuras aplicaciones .....	216
	Figura 114 Pantalla de Identificación y Registro .....	217
	Figura 115 Estructura de capas del CSS - Contenedor Global .....	219
	Figura 116 Estructura de capas del CSS - Contenedor Global .....	219
	Figura 117 Estructura de capas del CSS - Cabeceras, Navegación Global y rastro de migas..	220
	Figura 118 Estructura de capas del CSS - Contenido .....	220
	Figura 119 Estructura de capas del CSS - Contenido en detalle .....	221
	Figura 120 Esquema estructura global de capas del aplicativo .....	222
	Figura 121 Estructura Modelo Vista Controlador .....	222
	Figura 122 Esquema estructura global de capas del aplicativo Usando Sitemesh .....	223
	Figura 123 Funcionamiento en detalle de Sitemesh.....	224
7.3.2.5.	Storyboard.....	227
	Figura 124 Storyboard.....	227
7.3.2.6.	Enfoque y diseño del Artefacto .....	228
	Figura 125 Artefactos y iteraciones.....	228
7.3.2.7.	Arquitectura .....	228
	Figura 126 MVC DE SPRING .....	229
7.3.2.8.	CRC Cards .....	233
7.3.2.9.	Mapa de flujo Aplicación Inicial generada con el Arquetipo.....	238
7.3.2.10.	Diagramas de Clase UML .....	238
	Figura 127: Generación de clases mediante al Api Jaxb.....	239
	Figura 128 Diagrama de dependencias de la clase ConfigMenuTag .....	239

Figura 129 Diagrama de dependencia de MenuTag.....	240
Figura 130 Diagrama dependencias MenuConfig.....	240
Figura 131 Diagrama asociación clase Menus .....	241
Figura 132 Diagrama asociación clase Menus .....	241
Figura 133 Diagrama asociación de los modelos .....	242
Figura 134 Diagrama de dependencias controladores.....	245
Figura 135 Listener de configuración .....	245
<b>7.3.2.11. UML Package Diagram.....</b>	<b>246</b>
Figura 136 Diagrama de paquetes UML.....	246
Figura 137 Diagrama de paquetes UML para aplicación generada con el GroupId org.ckyosei.....	247
<b>7.3.2.12. Creación de Test y Aceptación .....</b>	<b>248</b>
<b>7.3.2.13. Creación de Arquetipos en Maven .....</b>	<b>249</b>
Figura 138 Arquetipos en Maven .....	249
<b>7.3.2.14. Creación de un Arquetipo para el Sistema Kyosei-Polis.....</b>	<b>249</b>
Figura 139 Habilitar administrador de dependencias en eclipse .....	250
Figura 140 Creación el proyecto en eclipse para el arquetipo.....	251
Figura 141 Selección de Maven en eclipse.....	251
Figura 142 Datos del fichero Pom.xml, del arquetipo .....	252
Figura 143 Figura 143 Esquema funcionamiento repositorios de Maven .....	252
Figura 144 : Ejemplo instalación de un artefacto en el repositorio en este caso Junit.....	253
Figura 145 Arquetipo de Ckyosei una vez implantado en el repositorio local.....	253
Figura 146 Creación del arquetipo como Proyecto Maven .....	254
Figura 147 : Estructura global del arquetipo CkyoseiArchetype .....	259
Figura 148 Arquetipo instalado en el control de versiones.....	260
Figura 149 Aplicativo generado a partir del arquetipo CkyoseiArchetype .....	263
Figura 150 Cargador clases Tomcat.....	264
Figura 151 Página principal de la aplicación generada con el arquetipo.....	265
<b>7.3.3. Aplicando XP y AMDD a la librería de utilidades.....</b>	<b>266</b>
<b>7.3.3.1. Arquitectura de la librería.....</b>	<b>266</b>
Figura 152 Patrón diseño Dao .....	267
Figura 153 Patrón de diseño Proxy .....	268
<b>7.3.3.2. CRC Cards .....</b>	<b>268</b>
<b>7.3.3.3. Diagramas de Clase UML .....</b>	<b>274</b>
Figura 154 Iterfaces genéricos DAOs .....	274
Figura 155 Interfaces finder para usar Spring AOP con los DAO .....	274
Figura 156 Diagrama de dependencias genericdao .....	275
Figura 157 Diagrama de asociación de genericdao .....	276
Figura 158 Diagrama de dependencias de BaseHibernateDAOImpl .....	276
Figura 159 Diagrama de Dependencias paquete exception .....	277
Figura 160 Interfaces de la lógica de negocio .....	278
Figura 161 Diagrama de dependencia de la implementación de genericbservice .....	278
Figura 162 Diagrama de dependencias de implementación de service .....	279
<b>7.3.4. Aplicando XP y AMDD a la aplicación alpha .....</b>	<b>280</b>
<b>7.3.4.1. Arquitectura de la aplicación .....</b>	<b>280</b>
<b>7.3.4.2. Modelo de dominio .....</b>	<b>281</b>
Figura 163 Modelo de dominio del aplicativo alpha.....	281
<b>7.3.4.3. Prototipos de interfaces de usuario.....</b>	<b>282</b>
Figura 164 Pantalla principal del aplicativo alpha .....	282
Figura 165 Pantalla de validación de usuarios aplicativo alpha.....	283
Figura 166 Pantalla de registro de usuarios aplicativo alpha.....	283
Figura 167 Pantalla de menú del Rol Usuario .....	284
Figura 168 Pantalla de menú del Rol Administrador .....	284
Figura 169 Pantalla de datos básicos de ambos roles.....	285
Figura 170 Pantalla de Listado de Foros de ambos Roles .....	286
Figura 171 Pantalla de Mis Foros de ambos Roles .....	287
Figura 172 Pantalla mis documentos ambos roles .....	288
Figura 173 Pantalla de subida de documentos ambos roles.....	288
Figura 174 Pantalla de permisos sobre documentos ambos roles .....	289
Figura 175 Listado de foros suscrito .....	290
Figura 176 Selección de usuarios para autorizar .....	290
Figura 177 Pantalla de creación de Foros para el Rol Administrador .....	291
<b>7.3.4.4. Storyboard .....</b>	<b>291</b>
Figura 178 StoryBoard parte pública aplicación alpha .....	291
Figura 179 StoryBoard parte privada aplicación alpha .....	292
<b>7.3.4.5. Plan de Entregas .....</b>	<b>292</b>

<b>9 FUTURAS LÍNEAS DE TRABAJO.....</b>	<b>297</b>
<b>10 PRESUPUESTO .....</b>	<b>301</b>
10.1. INTRODUCCIÓN .....	304
10.2. PRESUPUESTO DE EJECUCIÓN MATERIAL .....	305
10.3. GASTOS GENERALES Y BENEFICIO INDUSTRIAL.....	307
10.3.1. <i>Presupuesto de Ejecución por Contrata.....</i>	307
10.3.2. <i>Honorarios.....</i>	307
10.4. IMPORTE TOTAL DEL PRESUPUESTO.....	308
<b>11 BIBLIOGRAFÍA .....</b>	<b>309</b>
<b>12 GLOSARIO DE TÉRMINOS.....</b>	<b>313</b>

---



# **1 INTRODUCCIÓN**

---





El avance de Internet y la rápida aparición de nuevas tecnologías asociadas a la Web están facilitando que los sistemas de información se universalicen, permitiendo su acceso a cualquier usuario potencial conectado a Internet. Las características novedosas que plantea el desarrollo de aplicaciones en este contexto hacen que tenga sentido introducir el término de “Aplicación Web”. Con este término nos referimos a una nueva familia de aplicaciones informáticas especialmente modeladas y diseñadas para ser ejecutadas en la Web. Análogamente, empieza a utilizarse el término “Ingeniería Web” para referirse al conjunto de métodos, técnicas y herramientas que deben ser utilizadas para abordar el desarrollo de tales aplicaciones.

Las Aplicaciones Web presentan particularidades distintivas que van mucho más allá de ser utilizadas exclusivamente desde un navegador, aunque sin duda esta característica condiciona notablemente la interfaz de usuario.

En este contexto, se hace imprescindible disponer de métodos de desarrollo que proporcionen soluciones al problema de crear Aplicaciones Web fiables en tiempos relativamente cortos, y que permitan la reutilización de los desarrollos para problemas de dominios similares. Esto garantiza una posición de fuerza en cualquier sector que desee utilizar aplicaciones Web tanto en el entorno de negocios (e-Business), como en el desarrollo de las comunicaciones sociales.

Paralelamente con esta expansión del uso de Internet a nivel mundial, en todos los ámbitos de la actividad humana, está también surgiendo la necesidad de aplicar las nuevas tecnologías en los procesos políticos (Miller y Webb, 2007). En el contexto de una crisis de legitimidad de nuestros sistemas democráticos, que se expresa en un aumento de la abstención en los procesos electorales y unos niveles muy bajos de confianza en las instituciones y actores democráticos, se hace cada vez más necesaria la creación de herramientas de participación basadas en Internet que permitan que tanto los ciudadanos y ciudadanas como las administraciones públicas puedan beneficiarse con el uso continuado de espacios alternativos para el diálogo y la movilización. Es la llamada Participación Electrónica o e-Participación.

Actualmente se están imponiendo las arquitecturas basadas en marcos de desarrollo o frameworks. Los frameworks son la piedra angular de la moderna Ingeniería del software (Martínez et al., 1996). El desarrollo de frameworks está ganando rápida-mente aceptación debido a su capacidad para promover la reutilización del código del diseño y el código fuente. Los frameworks son los generadores de aplicaciones que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados.

En este marco las diferentes compañías fabricantes de software están involucradas en desarrollar aplicaciones y herramientas que permiten crear software de forma rápida, abarcando todo el ciclo de vida de los sistemas. En esta misma dirección trabajan personas y fundaciones dedicados a crear aplicaciones de código abierto (Open Source). La filosofía del Open Source centra su atención en la premisa de que al compartir el código, el programa resultante tiende a ser de calidad superior al software propietario.

Bajo la filosofía Open Source es como nace Spring Framework (Jonson, 2002, Hoeller, 2004, Risberg 2004), como un marco de desarrollo de código abierto basado en metodologías ágiles (Martin, 2003), que diera un salto cualitativo respecto a los otros entornos existentes y que fuera capaz de proporcionar un desarrollo dinámico de aplicaciones, que se caracterizase por su alta calidad.

Spring Framework buscaba asimismo proporcionar, gracias a su arquitectura modular, una integración completa con otros frameworks como Hibernate (Bauer, 2004) o Structs (Siggelkow, 2005). De esta forma, podrían utilizarse con independencia algunos de los módulos del framework, como su componente de modelo-vista-controlador, la lógica de negocios o la capa de persistencia por separado, o alternativamente podrían utilizarse todos conjuntamente, para dar un soporte completo a una aplicación. El presente Proyecto de Fin de Carrera trataría de integrar diversos frameworks para el desarrollo del Proyecto de Software Libre "Kyosei-Polis", cuyo objetivo es crear un Sistema para la e-Participación Municipal.

Realizaremos un análisis detallado y exhaustivo de varios frameworks de código abierto, como Spring e Hibernate, así como de los estándares y protocolos sobre los que se sustentaría el sistema. Elaboraremos un documento que, por un lado, permitirá a los futuros contribuidores/desarrolladores del Proyecto Kyosei-Polis configurar su sistema de desarrollo, y que por otro, servirá como una "base didáctica" sobre estas tecnologías, que con apenas cambios podrían ser aplicadas en multitud de ámbitos y proyectos distintos de Kyosei-Polis.

## **2 RESUMEN**

---





En el contexto anteriormente expuesto en la introducción al TFC, se hace imprescindible disponer de métodos de desarrollo que proporcionen soluciones al problema de crear Aplicaciones Web fiables en tiempos relativamente cortos, y que permitan la reutilización de los desarrollos para problemas de dominios similares. Esto garantiza una posición de fuerza en cualquier sector que desee utilizar aplicaciones Web tanto en el entorno de negocios (e-Business), como en el desarrollo de las comunicaciones sociales.

Paralelamente con esta expansión del uso de Internet a nivel mundial, en todos los ámbitos de la actividad humana, está también surgiendo la necesidad de aplicar las nuevas tecnologías en los procesos políticos. En el contexto de una crisis de legitimidad de nuestros sistemas democráticos, que se expresa en un aumento de la abstención en los procesos electorales y unos niveles muy bajos de confianza en las instituciones y actores democráticos, se hace cada vez más necesaria la creación de herramientas de participación basadas en Internet que permitan que tanto los ciudadanos y ciudadanas como las administraciones públicas puedan beneficiarse con el uso continuado de espacios alternativos para el diálogo y la movilización. Es la llamada Participación Electrónica o e-Participación. En este punto es donde entra en juego la Asociación Ciudades Kyosei. Esta asociación no es solamente una Entidad sin Ánimo de Lucro. Somos más bien una Entidad con ansia de Desarrollo Social, y están comprometidos con el fortalecimiento de la participación ciudadana democrática en el ámbito municipal y regional.

Este proyecto surge como continuación del TFC de Marta Prieto Martín, Análisis, diseño, creación de prototipos y evaluación para el desarrollo de una plataforma de participación ciudadana, por la Universidad de Alcalá (España), en colaboración con el Proyecto e-Participa en el que se daba un marco teórico a un sistema de participación ciudadana. En este proyecto, pretendemos ir un paso más adelante y llevar a la práctica el sistema. Lo que pretendemos es establecer la plataforma tecnológica sobre la que se sustentará el sistema, así como la arquitectura de desarrollo de la misma



### **3 OBJETIVOS DEL PROYECTO**

---





### **3. Objetivos del proyecto**

---

El propósito principal del trabajo es el establecimiento de la plataforma tecnológica sobre la que se instalará el sistema de participación ciudadana, así como la implantación de la arquitectura de desarrollo.

Para conseguirlo, se plantean los siguientes objetivos específicos:

1. Suministrar un completo manual de instalación y configuración de todo el entorno tecnológico basado en Software libre. Plataforma de desarrollo, Servidores de Aplicaciones, Servidores Web, plataforma de seguridad con ejemplos de funcionamiento en los casos necesarios. Sistemas de control de versiones y de documentación.
2. Establecimiento de una metodología de trabajo del proyecto, que permitirá el rápido desarrollo de los diferentes módulos del sistema, como es la metodología extrema (XP).
3. Implantación de Maven como herramienta de software para la gestión y construcción de proyectos Java.
4. Spring Framework como marco de desarrollo de aplicaciones que permite la cobertura total al ciclo de vida del Software. Así como la fácil integración con otros marcos de desarrollo como Hibernate para persistencia de datos dentro de la capa de integración o JSF para la capa de presentación, etc.
5. Se creará un arquetipo de Maven que permita la generación del armazón de una aplicación Web que incorporé y configure todos los frameworks de desarrollo que vamos a usar inicialmente como punto de partida para el sistema.
6. Creación de una aplicación alpha, que ponga el punto de partida al sistema y que sirva para probar tanto la plataforma tecnológica como la eficiencia de la metodología y arquitectura elegida para el sistema de participación ciudadana.



## **4 RESULTADOS**

---





## 4. Resultados

---

El principal resultado del trabajo será un informe (memoria) con los siguientes contenidos:

Memoria de instalación y configuración para el sistema Kyosei-Polis de la plataforma tecnológica que incluirá:

1. Instalación de la BBDD Mysql. (Incluye Ejemplo creación BBDD.)
2. Instalación del servidor de aplicaciones Tomcat.
3. Instalación del servidor Web Apache.
4. Instalación y Configuración del plugin de conexión Apache-Tomcat.
5. Instalación de OpenSSL así como configuración SSL de Apache.
6. Instalación de PHP para Apache. (Incluye Ejemplo.)
7. Prueba de todo el sistema de servidores.
8. Instalación del entorno de desarrollo que incluye:
  9. Eclipse y plugins necesarios. Incluye Ejemplo creación de aplicaciones
  10. Instalación de Java Development Kit.
  11. Instalación de framework de desarrollo Spring, Hibernate y ejemplos de funcionamiento.
  12. Instalación de sistemas de trazas o Logs.
  13. Instalación del sistema de pruebas.

Una primera versión "alfa" del sistema Kyosei-Polis, que englobe los módulos principales de los que consta Spring, así como la integración con otros marcos de trabajo. Proporcionaremos los siguientes productos:

1. Documentación de Ingeniería del Software del prototipo de ejemplo.
2. Código fuente generado.
3. Aplicativo compilada y documentación para el paso a explotación de la misma.
4. Arquetipo Maven que permitan crear la estructura de un aplicativo Web de una manera rápida, con el objetivo que personas no versadas en la utilización de estos frameworks puedan comenzar rápidamente su utilización.
5. Librería de utilidades para la capa media de las aplicaciones Web, que incluya acceso a datos y lógica de negocios, etc.



## **5 ORGANIZACIÓN DE LA MEMORIA**

---





## 5. Organización de la memoria

---

La memoria del proyecto se va a organizar en 2 bloques principalmente.

En el primer bloque se dedicará a la Instalación / Configuración de las tecnologías sobre las que implantaremos el sistema.

Este bloque a su vez va a estar formado por 3 partes principales que son:

- Ü La instalación de los servidores sobre los que se desplegarán las aplicaciones (Configuración servidor Ckyosei). Esta primera parte irá destinada al administrador del entorno de Ckyosei, aunque puede ser interesante que conozcan los futuros programadores el entorno tecnológico sobre el que trabajarán en el futuro.
- Ü La instalación del entorno de desarrollado. (Configuración equipos de desarrollo de Ckyosei). Cada Programador tendrá que instalar y configurar su máquina local con al menos el entorno de desarrollo indicado en este documento.
- Ü Instalación y uso de los frameworks de desarrollado. (Configuración equipos de desarrollo de Ckyosei)

En el segundo bloque se introducirá la documentación de la aplicación alpha así como los diferentes desarrollos creados para la misma. Veremos las diferentes partes que compondrán este bloque más adelante.

En las instalaciones del entorno tecnológico, aparecen varios puntos opcionales, como:

- Ü Ejemplos de creación base de datos,
- Ü Interconexión Apache-Tomcat,
- Ü Creación aplicaciones con Maven,
- Ü Etcétera.

Sería recomendable realizar estos pasos opcionales, ya que nos permitirán desplegar más fácilmente la aplicación alpha en la segunda parte de la memoria.



## **6 INSTALACIÓN DEL SISTEMA KYOSEI-POLIS**

---





A continuación se muestran los diferentes formatos que pueden encontrarse en la guía de instalación de la plataforma tecnológica.

Explicación de formatos de texto: Texto descriptivo:

Texto que aparece en algún lugar

Texto a ser insertado

**Texto que debe ser sustituido por la propia configuración;**

**Nombres de archivos, directorios o clases;**

Texto introducido en una ventana de comandos

## **6.1. Instalación y Configuración del Servidor ckyosei**

En esta primera parte, se va a ver la configuración de los servidores donde se instalarán las aplicaciones desarrolladas para el sistema, así como los repositorios de versiones. Etc.

### **6.1.1. Preparación del entorno**

Para mantener todos los programas relacionados con el Entorno de Desarrollo de Kyosei-Polis, conviene que crees un directorio llamado **CKyosei**, y dentro de él los directorios **SDK** (para el Java SDK), **Herramientas**, **WorkSpace** (para el código de la aplicación) y **Repositorios** (para los repositorios Subversion).

Algunas de las descargas del sistema poseen un tamaño considerable. En caso de que encuentres problemas para la descarga de los instalables, te recomendamos utilizar alguna aplicación de gestión de descargas gratuita.



### 6.1.2. Instalación del Servidor de BBDD Mysql

#### 6.1.2.1. Introducción

MySQL es un sistema de gestión de base de datos relacional, multi-hilo y multiusuario. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo licencia GNU GPL, pero las empresas que quieran incorporarlo en productos privativos pueden comprar una licencia que les permita ese uso. Está desarrollado en su mayor parte en ANSI C.

MySQL es una base de datos cliente/servidor: hay un servidor (o *daemon*) que se ejecuta en segundo plano (modo background) a la escucha de las peticiones del programa cliente. En MySQL, el servidor es `mysqld` y el cliente `mysql`.

#### 6.1.2.2. Requerimientos

- Paquete de instalación de MySQL, `mysql-5.1.X-rc-win32.zip`  
<http://dev.mysql.com/downloads/mysql/5.1.html#win32>

#### 6.1.2.3. Proceso de instalación

1. Descomprimir paquete de instalación de MySQL y ejecutar `Setup.exe`.
2. En el menú de instalaciones seleccionar la opción `Custom`.



Figura 1 Instalación MySQL Server

3. La siguiente pantalla muestra los módulos que podemos instalar, dejaremos la selección por defecto, cambiando el directorio de instalación a C:\CKyosei\Herramientas\ MySQLServer5.1, pulsando **Change**....

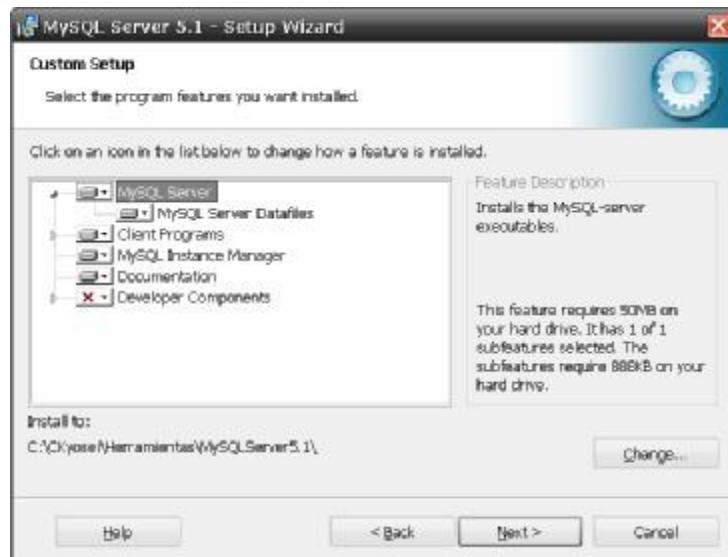


Figura 2 Selección de componentes

4. Por último pulsaremos la opción **Install**, que instalará el programa en el directorio indicado.

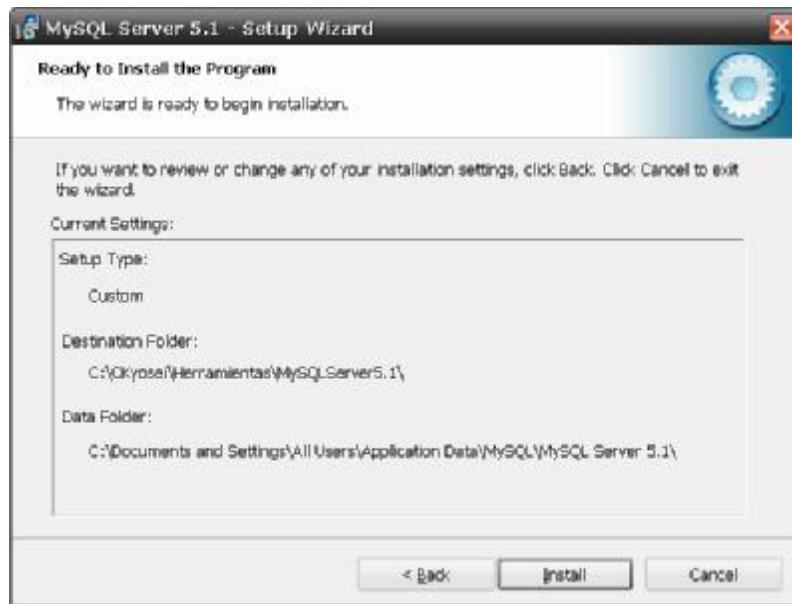


Figura 3 Confirmación Instalación MYSQL



- Una vez realizada la instalación, seleccionaremos **Configure the MySQL Server now**, que nos permitirá configurar el servidor de base de datos.



Figura 4 Configuración del Servidor

- A continuación seleccionamos **Standard Configuration** y pulsamos **Next**.

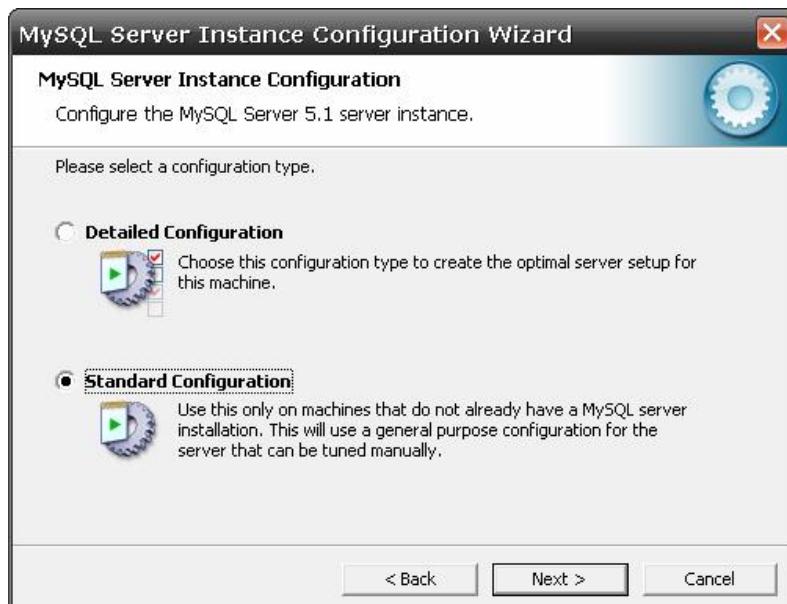


Figura 5 Selección de Configuración

- La siguiente pantalla nos solicita la instalación de MySQL como servicio, así como la inclusión del directorio con los ejecutables MySQL en el path de Windows. Seleccionaremos ambos y pulsamos **Next**.



Figura 6 Configuración como servicio Windows

8. El siguiente paso de la instalación de MySQL solicita un usuario de administración de la base de datos, una vez introducido pulsamos **Next**.



Figura 7 Configuración usuario administrador

9. Por último, la instalación muestra un resumen de los pasos que va a ejecutar para finalizar el proceso. Pulsamos **Execute**.



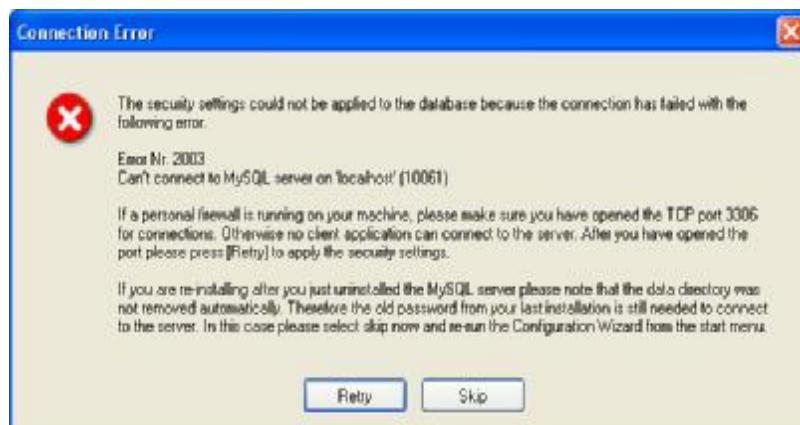
**Figura 8 Confirmación instalación**

*En algunos casos, cuando se establece una clave para el usuario root, se ha reportado que el asistente de configuración genera un error de conexión como el que se muestra a continuación. Pese a ello, la configuración de la seguridad se establece correctamente. Pulsando Skip se completará el asistente:*



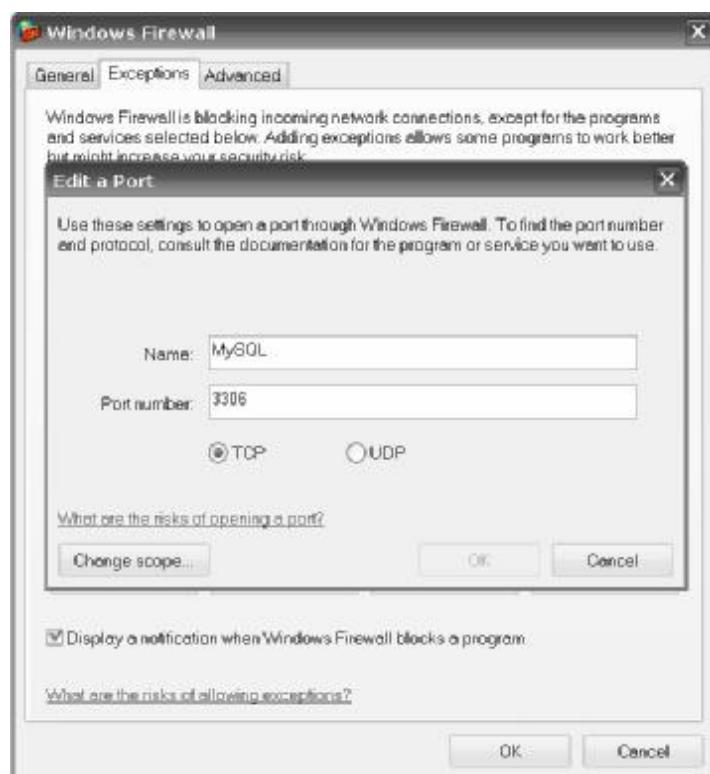
**Figura 9 Aviso seguridad de usuario root de MySQL**

*NOTA 2: La instalación estándar de MySQL utiliza para el servidor el puerto 3306. Este puerto debe admitir conexiones TCP, por lo que si se utiliza un Firewall como con Windows XP SP2, podríamos recibir un error del tipo:*



**Figura 10 Error de conexión TCP de MySQL**

- 9.1. Para abrir el acceso al puerto tendremos que ir a **Inicio > Panel de Control > Firewall > Excepciones > Agregar Puerto.**



**Figura 11 Apertura de puerto de MySQL en el firewall de Windows**

10. La pantalla final indica si todas las actividades de configuración se han realizado correctamente. Pulsamos **Finish** para concluir el proceso.



Figura 12 Pantalla de finalización de la instalación de MySQL

### 6.1.3. Opcional: Creación de la BBDD de Spring-Security

- Una vez instalado MySQL, el servicio que arranca el servidor de MySQL se establecerá en modo automático, es decir el servidor de base de datos arrancará con el inicio de Windows. Vamos a crear la base de datos de Spring-security que formará parte del futuro aplicativo alpha.

La base de datos de datos de Spring-security está formada principalmente por 6 entidades.

La gestión de seguridad puede ser dirigida a grupos (Entidades en Rosa) o a usuarios (Entidades en amarillo).

Si la gestión de seguridad es orientada a usuarios, como se observa en la figura anterior. Un usuario puede tener N Roles.

Así mismo si está orientada a Grupos. Un usuario puede pertenecer a N grupos y un Grupo puede tener N Roles.

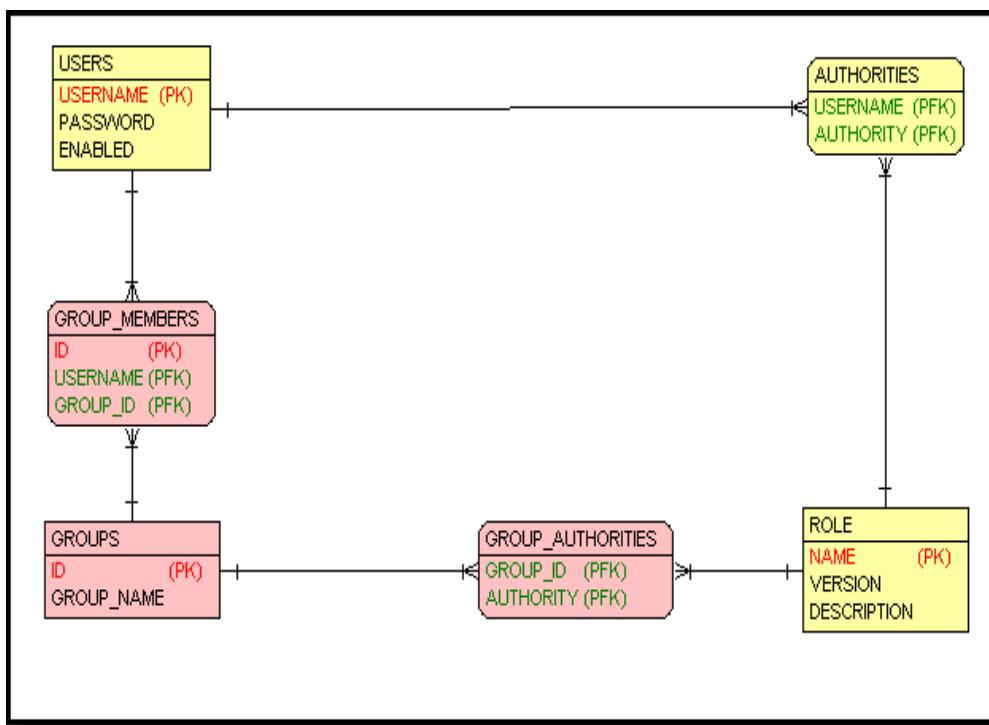


Figura 13 Modelo lógico de autenticación de Spring security

Tipos de tablas en MySQL:

- **ISAM**. Es el formato de almacenaje más antiguo, y posiblemente pronto desaparecerá. Presentaba limitaciones (los ficheros no eran transportables entre máquinas con distinta arquitectura, no podía manejar ficheros de tablas superiores a 4 gigas). Si aún tienes tablas tipo ISAM, cámbialas a MYISAM.
- **MYISAM**. Es el tipo de tabla por defecto en MySQL desde la versión 3.23. Optimizada para sistemas operativos de 64 bits, permite ficheros de mayor tamaño que ISAM. Además los



datos se almacenan en un formato independiente, con lo que se pueden copiar tablas entre distintas plataformas. Posibilidad de indexar campos BLOB y TEXT.

- ü **HEAP.** Crea tablas en memoria. Son temporales y desaparecen cuando el servidor se cierra; a diferencia de una tabla TEMPORARY, que solo puede ser accedida por el usuario que la crea, una tabla HEAP puede ser utilizada por diversos usuarios.
  - ü **BDB.** Base de datos Berkeley TST. Sólo en MySQL MAX
  - ü **INNODB.** TST, ACID con posibilidad de *commit*, *rollback*, recuperación de errores y bloqueo a nivel de fila.
  - ü **MERGE.** Más que un tipo de tabla es la posibilidad de dividir tablas MYISAM de gran tamaño (sólo útil si son verdaderamente de GRAN tamaño) y hacer consultas sobre todas ellas con mayor rapidez. Las tablas deben ser MYISAM e idénticas en su estructura.
2. Para realizar la conexión con el servidor a través del cliente tendremos que validarnos en el sistema. Abrimos la línea de comandos y nos identificaremos como root frente a *MySQL*. Si no tiene contraseña el usuario de administración que hemos creado durante la instalación

```
>mysql -u root
```

En otro caso, usaremos la opción **-p** que nos solicitará la contraseña.

```
>mysql -u root -p
```

Una vez validados en el sistema podemos ejecutar comandos tales como:

INSTRUCCIÓN	DESCRIPCIÓN
<code>show databases;</code>	Muestra el conjunto de bases de datos presentes en el servidor
<code>use nombre_de_la_bd;</code>	Determina la base de datos sobre la que vamos a trabajar
<code>create database nombre_de_la_bd;</code>	Crea una nueva bd con el nombre especificado
<code>drop database nombre_de_la_bd;</code>	Elimina la base de datos del nombre especificado
<code>show tables;</code>	Muestra las tablas presentes en la base de datos actual
<code>describe nombre_de_la_tabla;</code>	Describe los campos que componen la tabla
<code>drop table nombre_de_la_tabla;</code>	Borra la tabla de la base de datos
<code>load data local infile 'archivo.txt' into table nombre_de_la_tabla;</code>	Crea los registros de la tabla a partir de un fichero de texto en el que separamos por tabulaciones todos los campos de un mismo registro
<code>quit</code>	Salir de MySQL

Tabla 1 Instrucciones MySQL

3. Comenzamos la creación de la base de datos. Crear un script en un fichero llamado security.sql con el siguiente contenido.



```
Create table ROLE (
    NAME Varchar(20) NOT NULL,
    VERSION Int NOT NULL,
    DESCRIPTION Varchar(255) NOT NULL,
    Primary Key (NAME)) ENGINE = MyISAM;

Create table AUTHORITIES (
    USERNAME Varchar(50) NOT NULL,
    AUTHORITY Varchar(20) NOT NULL,
    Primary Key (USERNAME,AUTHORITY)) ENGINE = MyISAM;

Create table USERS (
    USERNAME Varchar(50) NOT NULL,
    PASSWORD Varchar(50) NOT NULL,
    ENABLED Tinyint NOT NULL,
    Primary Key (USERNAME)) ENGINE = MyISAM;

Create table GROUP_MEMBERS (
    ID Decimal(18,0) NOT NULL,
    USERNAME Varchar(50) NOT NULL,
    GROUP_ID Decimal(18,0) NOT NULL,
    Primary Key (ID,USERNAME,GROUP_ID)) ENGINE = MyISAM;

Create table GROUPS (
    ID Decimal(18,0) NOT NULL,
    GROUP_NAME Varchar(50) NOT NULL,
    Primary Key (ID)) ENGINE = MyISAM;
```



```
Create table GROUP_AUTHORITIES (
    GROUP_ID Decimal(18,0) NOT NULL,
    AUTHORITY Varchar(20) NOT NULL,
    Primary Key (GROUP_ID,AUTHORITY)) ENGINE = MyISAM;

Alter table AUTHORITIES add Foreign Key (AUTHORITY)
    references ROLE (NAME) on delete restrict on update restrict;

Alter table GROUP_AUTHORITIES add Foreign Key (AUTHORITY)
    references ROLE (NAME) on delete restrict on update restrict;

Alter table AUTHORITIES add Foreign Key (USERNAME) references
USERS (USERNAME) on delete restrict on update restrict;

Alter table GROUP_MEMBERS add Foreign Key (USERNAME) references
USERS (USERNAME) on delete restrict on update restrict;

Alter table GROUP_AUTHORITIES add Foreign Key (GROUP_ID)
    references GROUPS (ID) on delete restrict on update restrict;

Alter table GROUP_MEMBERS add Foreign Key (GROUP_ID) references
GROUPS (ID) on delete restrict on update restrict;
```



Tabla 2 Script de creación de BBDD Spring Security

## 6.1.4. Instalación de Java Development Kit

### 6.1.4.1. Introducción

Java es un lenguaje de programación creado por *Sun Microsystems* con el objetivo de que fuera funcional y portable en distintos tipos de procesadores y sistemas operativos. Su sintaxis se basa en lenguajes previos como fueron C y C++, aunque aporta un modelo de objetos mucho más simple, eliminando algunas características problemática de estos dos lenguajes, como son la aritmética de punteros y la gestión programática de la memoria.

Java es un lenguaje compilado e interpretado, lo que nos garantiza una portabilidad entre diferentes sistemas. El código fuente Java es en primera instancia compilado, generándose el *bytecode* del programa. Estos *bytecodes* puede llevarse sin modificación sobre cualquier máquina y ser ejecutados, ya que el código final se genera sobre la Java Virtual Machine, que se encarga así de interpretar el código pre-compilado y convertirlo al código particular de la plataforma dónde se ejecute.

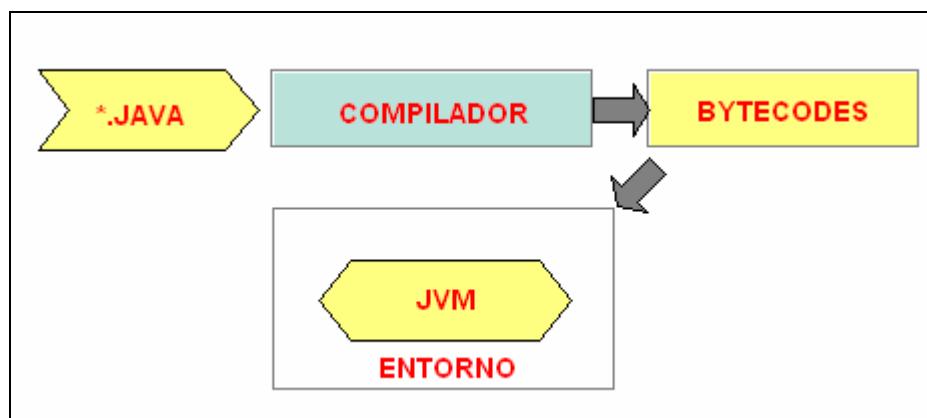


Figura 14 Ciclo vida generación código Java

La interpretación del gráfico anterior significa que mientras el ordenador tenga una JVM (Java Virtual Machine) el mismo programa escrito en Java puede ejecutarse en *Windows*, *Solaris*, *Mac*, *Linux*, etc.

La plataforma Java está compuesta de 2 componentes:

- La máquina virtual de Java (JVM)
- El Java API (Application Programming Interface)

El Java API incluye un conjunto de componentes que nos proporciona el lenguaje de alto nivel Java para el desarrollo de aplicaciones basadas en este lenguaje. En otras palabras podríamos decir que el API



Java es una capa intermedia entre el código fuente y la Java Virtual Machine y que ambos aislan el código fuente del HW de la plataforma.

### 6.1.4.2. Prerequisitos

- **Java SE Development Kit 6** (update 10 en el momento de redacción de este manual), **jdk-6uX-  
Windows-i586-p.exe**:

<http://java.sun.com/javase/downloads/index.jsp>

### 6.1.4.3. Proceso de instalación

El proceso de instalación no reviste de dificultad. Simplemente han de seguirse los pasos indicados por el programa instalador, estableciendo en la primera pantalla el directorio **C:\CKyosei\SDK\jdk1.6.0\_X\**, para la instalación

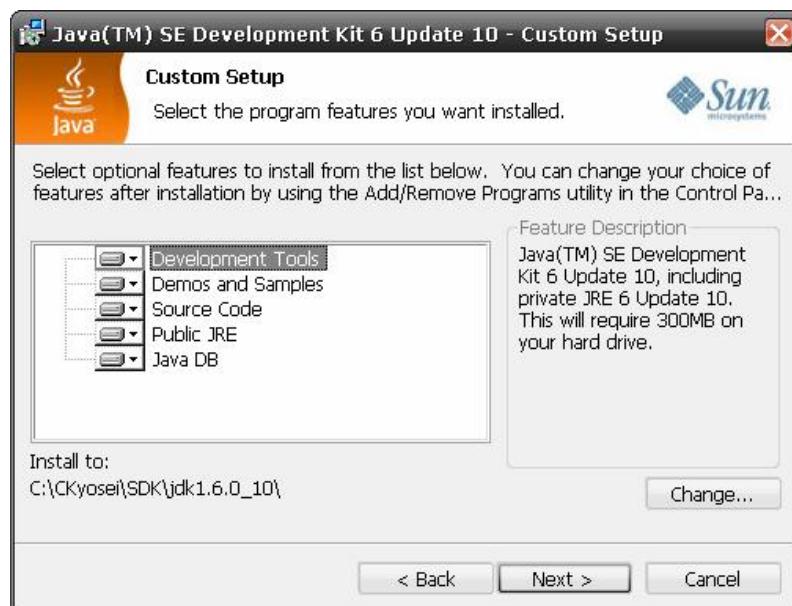


Figura 15 Instalación JSDK

### 6.1.4.4. Configuración de variables de entorno

Una vez instalado el **Java Development Kit**, tendremos que configurar las variables de entorno para la correcta ejecución de java desde cualquier punto del sistema de archivos, así como para la correcta localización de las clases. Debemos definir dos variables de entorno llamadas **PATH** y **CLASSPATH**.

1. La variable **CLASSPATH** indica dónde están las clases externas necesitadas para compilar o ejecutar un programa. **CLASSPATH** debe apuntar al directorio raíz donde comienzan las clases que queremos compilar. Por ello, añadimos a la variable la ruta:

- Finalmente, debemos definir la variable `JAVA_HOME`, que indica donde tenemos instalado el JDK. Esta variable la usarán múltiples programas basados en Java, como Tomcat (Servidor de aplicaciones), Eclipse (IDE programación java), etc. El valor de `JAVA_HOME` es `C:\CKyosei\SDK\jdk1.6.0_10`.



## 6.1.5. Instalación de TOMCAT

### 6.1.5.1. Introducción

*Tomcat* (también llamado *Jakarta Tomcat* o *Apache Tomcat*) funciona como un contenedor de servlets desarrollado bajo el proyecto *Jakarta* en la *Apache Software Foundation*. *Tomcat* implementa las especificaciones de los servlets y de Java Server Pages (JSP) de Sun Microsystems.

### 6.1.5.2. Requerimientos

- *Apache Tomcat*, versión estable de la serie 6.0.x (*6.0.18 en el momento de redacción de este manual*), [apache-tomcat-6.0.x.exe](http://www.apache.org/dist/tomcat/tomcat-6/):  
<http://www.apache.org/dist/tomcat/tomcat-6/>
- Conector de MySQL, controlador JDBC para utilizar java como cliente de la base de datos, [mysql-connector-java-5.1.x-rc.zip](http://dev.mysql.com/downloads/connector/j/5.1.html)  
<http://dev.mysql.com/downloads/connector/j/5.1.html>
- Java Development Kit (JDK) debe estar instalado (requiere mínimo la versión JSE 5.0)
- La variable JAVA\_HOME correctamente establecida en el sistema.

## 6.1.6. Proceso de instalación

1. Ejecutar [apache-tomcat-6.0.x.exe](http://tomcat.apache.org).
2. En la pantalla que aparece pulsar **Next**.



Figura 16 Pantalla instalación Tomcat

3. La siguiente pantalla muestra la licencia de instalación del producto. Pulsamos **I Agree**.

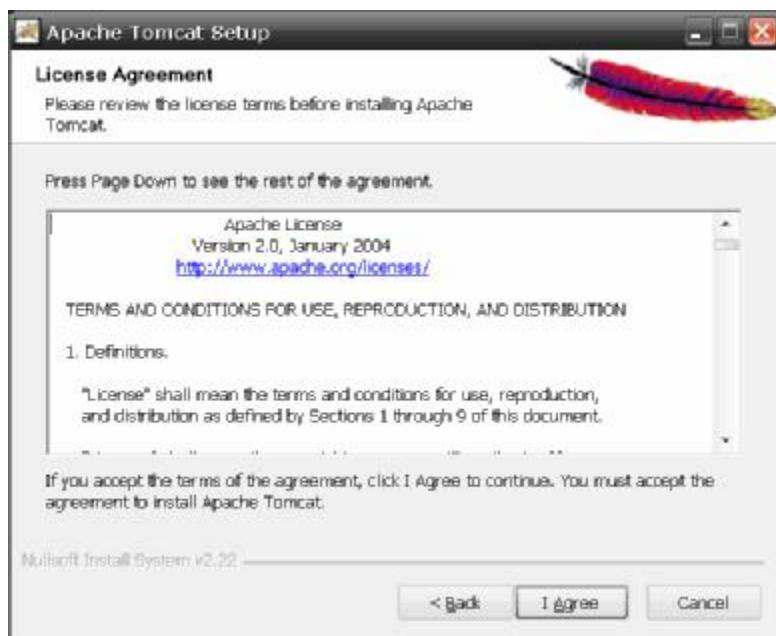


Figura 17 Aceptar licencia

4. La siguiente pantalla muestra los módulos que podemos instalar, dejaremos la selección por defecto, y pulsaremos **Next**.

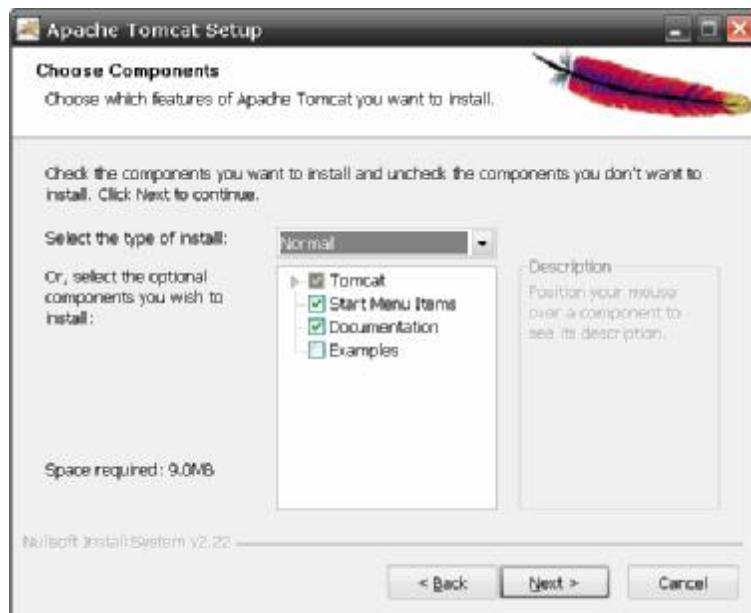


Figura 18 Selección de componentes



- En la siguiente pantalla seleccionamos el directorio C:\CKyosei\Herramientas, en el que se instalará el programa, y pulsamos **Next**.

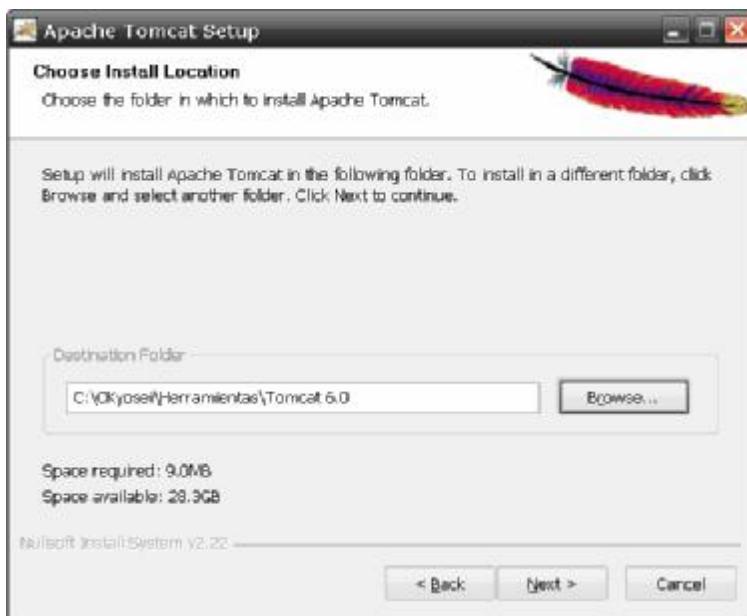


Figura 19 Directorio de instalación

- Por último seleccionamos el puerto que asignado al servidor Web y el usuario que tendrá permisos de administración sobre la consola de Tomcat y pulsamos **Next**.

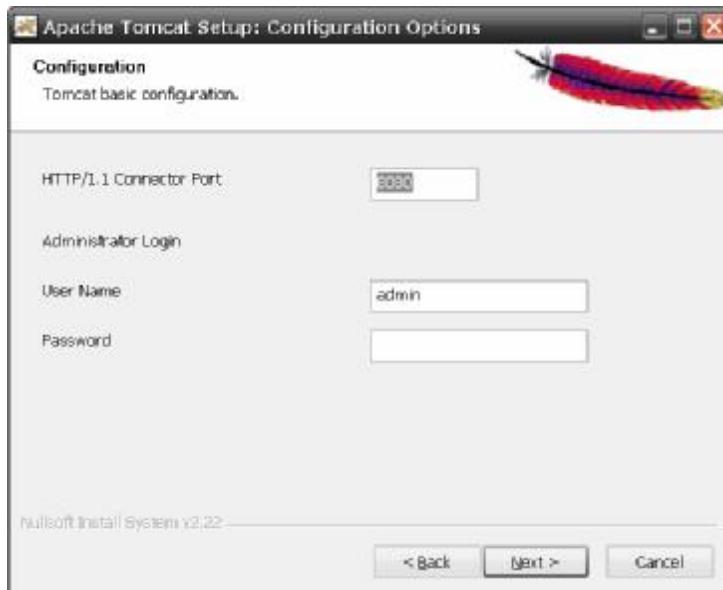
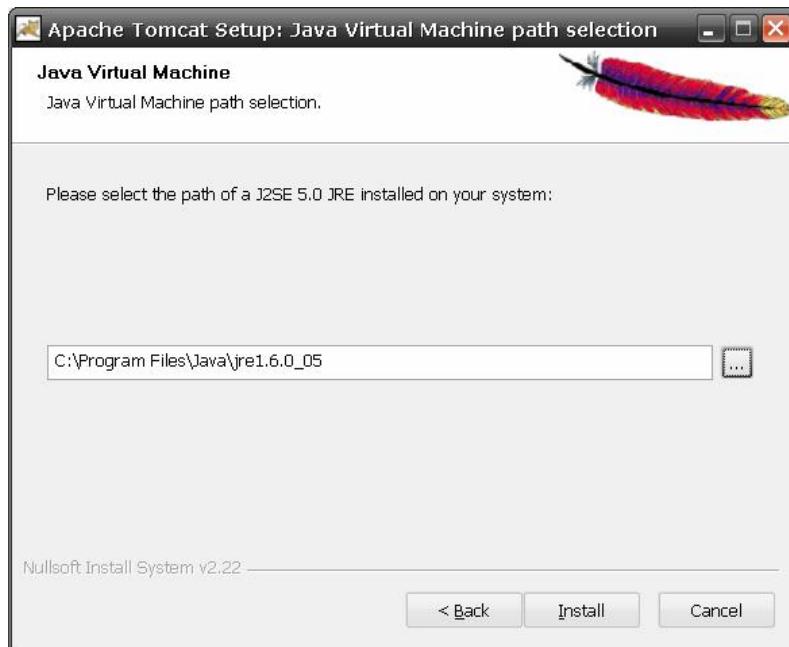


Figura 20 Selección de puerto y usuario administrador

- Como último paso seleccionamos la máquina virtual o el jdk que tengamos instalados.



**Figura 21 Selección de Jre para Tomcat**

8. La instalación creará un servicio denominado *Tomcat*, que por defecto arranca al iniciar el sistema.
9. Una vez finalizada la instalación creamos las variables del sistema **TOMCAT\_HOME** y **CATALINA\_HOME**, asignándoles el valor del directorio de instalación de *Tomcat*.

**C:\CKyosei\Herramientas\Tomcat 6.0**

10. Descargar el driver conector para la base de datos. Extraer del fichero .zip el archivo **mysql-connector-java-5.1.7-bin.jar**, en el directorio de librerías de Tomcat: **\$TOMCAT\_HOME/lib**.



### 6.1.7. Instalación de Apache Web Server

#### 6.1.7.1. Introducción

El servidor HTTP Apache es un software (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

#### 6.1.7.2. Requerimientos

- Apache Web Server, versión estable de la serie 2.2.x (2.2.10 en el momento de redacción de este manual), apache\_2.2.x-win32-x86-openssl-0.9.8i.msi:

<http://httpd.apache.org/download.cgi>

#### 6.1.7.3. Proceso de Instalación

- Ejecutar apache\_2.2.x -win32-x86-openssl-0.9.8i.msi.
- En la pantalla que aparece pulsar **Next**.



Figura 22 Pantalla Instalación Apache

3. La siguiente pantalla muestra la licencia de instalación del producto pulsamos I Accept y después **Next** y nuevamente **Next**.

4. La siguiente pantalla nos muestra los datos de configuración del servidor web:



**Figura 23 Pantalla de configuración Apache**

En **Network Domain**, pondremos la dirección IP de donde va a correr el servidor Web o el nombre del dominio, esto permitirá que el servidor sea accesible globalmente. Si, como en este caso, es una instalación local, se indica **localhost** o **127.0.0.1**. La dirección IP sólo será válida en caso de tener una dirección IP estática. Si no se posee dirección IP estática el servidor Web no será visible desde el exterior salvo que utilicemos redireccionadores de IP como [www.DynDNS.org](http://www.DynDNS.org).

En **Server Name** pondremos el nombre del Servidor Web y en **Administrator's Email Address** la dirección del administrador del sitio Web.

**For all users, on port 80, as a service:** instala Apache como un servicio de Windows, es decir que Apache se ejecuta al iniciar el ordenador; eligiendo esta opción el servidor se pone a la escucha en el puerto 80. Una vez llenados todos los campos pulsamos **Next**.

5. Seleccionamos el tipo de instalación **Custom** y pulsamos **Next**.

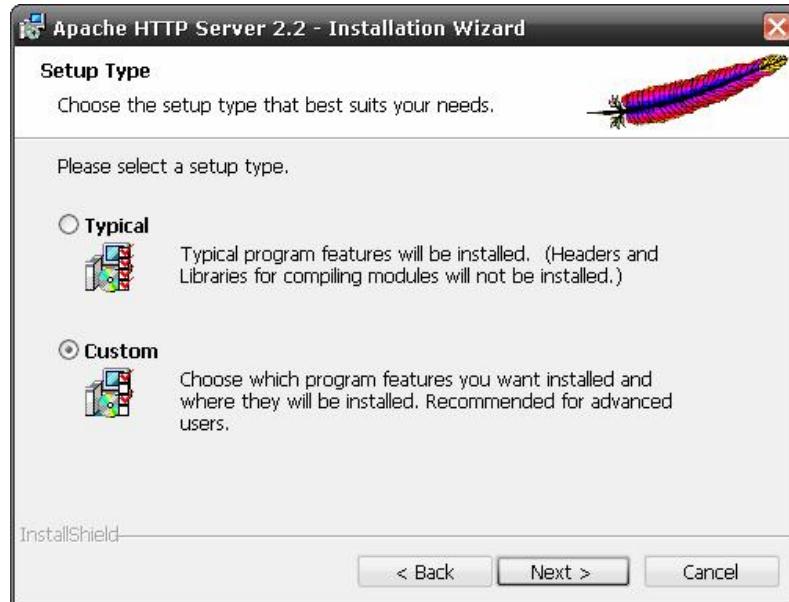


Figura 24 Tipo de instalación de Apache

6. A continuación indicamos el directorio de instalación, **C:\CKyosei\Herramientas\Apache2.2\**, y pulsamos **Next**.



Figura 25 Selección de componentes de Apache

7. Aceptamos la instalación del Servidor Web. En caso de que el puerto 80 de tu computadora esté siendo utilizado por otro servicio, el programa de instalación no será capaz de crear o de arrancar el

Servicio de Apache. Deberás modificar el archivo **C:\CKyosei\Herramientas\Apache2.2\conf\httpd.conf**, especificando en la línea **Listen 80** un puerto distinto (por ejemplo, el 8080). En caso de que el Servicio de Apache no hubiera sido creado, puede hacerse desde la línea de comandos, situándose en el directorio **C:\CKyosei\Herramientas\Apache2.2\bin** y ejecutando el siguiente comando:

```
httpd -k install
```

8. Establecer una variable de entorno **APACHE\_HOME** cuyo valor sea el directorio donde haya descomprimido **Apache**. Añadimos también el subdirectorio **bin** de **Apache** a la variable **PATH** de **Windows**, agregándole **; %APACHE\_HOME%\bin**, para que se encuentren automáticamente sus ejecutables.



## 6.1.8. Configuración del conector Apache – Tomcat

### 6.1.8.1. Introducción

Hasta este punto de la guía, el servidor web *Apache* y el contenedor de servlets *Tomcat* funcionan de manera independiente. Vamos a establecer ahora un vínculo entre ambos, para que trabajen coordinadamente. La motivación principal para hacerlo es que Tomcat no es tan rápido como Apache al servir contenido estático. El servidor web *Apache* es mucho más configurable que *Tomcat*, y permite fácilmente integrar otras tecnologías como *Perl*, *PHP* y *CGI*. Por este motivo en lugar de usar sólo un servidor vamos a hacer que cooperen, de forma que cuando se realice una petición http dinámica sea *Tomcat* el que la responda y en otro caso sea *Apache*.

### 6.1.8.2. Requerimientos

- Tomcat-connectors, mod\_jk-1.2.27-httpd-2.2.10.so:

<http://www.apache.org/dist/tomcat/tomcat-connectors/jk/binaries/win32/jk-1.2.27/>

## 6.1.9. Proceso de Configuración

1. Una vez descargado el módulo de conexión **mod\_jk-1.2.27-httpd-2.2.10.so**, lo copiamos con el nombre **mod\_jk.so** en el subdirectorio **modules** del directorio en que está instalado *Apache*; en nuestro caso: **C:\CKyosei\Herramientas\Apache2.2\modules**.
2. Crear en **\$TOMCAT\_HOME\conf** un subdirectorio **jk** y dentro de él un fichero de nombre **workers.properties**. Con el siguiente contenido.

“

```
# BEGIN workers.properties
# Definition for Ajp13 worker
worker.list=ajp13
worker.ajp13.port=8009
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
# END workers.properties
```

”

**Tabla 3 workers.properties**

3. Abrimos el fichero **\$TOMCAT\_HOME/conf/server.xml** y añadimos entre los tags **Engine** lo siguiente (ajustado al directorio de instalación de Apache):

“

```
<Engine ...>
<Listener className="org.apache.jk.config.ApacheConfig"
modJk="C:/CKyosei/Herramientas/Apache2.2/modules/mod_jk.so" />
.....
</Engine ...>
```

”

**Tabla 4 server.xml**

4. Rearrancar **Tomcat**. Al iniciarse **Tomcat** se generará un fichero dentro del directorio **conf**: **\$TOMCAT\_HOME\conf\auto\mod\_jk.conf**
5. Copiar el archivo anterior al directorio **jk** creado en el punto 2. Así tendremos toda la configuración de **Apache** junta en el mismo directorio.
6. En el fichero **httpd.conf** de **Apache**, que está en **\$APACHE\_HOME/conf**, debemos incluir una referencia a este fichero. Para ello, añadimos al final del fichero la siguiente línea, ajustada al



directorio de instalación de *Tomcat* (las comillas son importantes, pues son necesarias si algún directorio de la ruta contiene espacios en blanco):

"C:/CKyosei/Herramientas/Tomcat 6.0/conf/jk/mod_jk.conf"
---

Tabla 5 inclusión en httpd.conf de mod\_jk.conf

7. Por último configuraremos el propio archivo \$TOMCAT\_HOME/conf/jk/mod\_jk.conf. Este archivo indica a *Apache* cómo y cuándo llamar a *Tomcat*. Por defecto se llama a *Tomcat* en las aplicaciones Web que tenga *Tomcat*.

Son las líneas con *JkMount* dentro del directorio VirtualHost. Puesto que queremos que todas las *JSP*'s que se pidan las sirva *Tomcat* añadiremos:

JkMount /*.jsp ajp13
----------------------

Tabla 6 Mapeo de URL en el archivo mod\_jk.conf

### 6.1.9.1. OPCIONAL: Prueba de configuración Apache - Tomcat

Para poder probar la comunicación entre ambos servidores vamos a crear una simple aplicación Web que contenga un *JSP*, que realice una llamada a una imagen.

La imagen va a estar alojada en el servidor Apache.

Si realizáramos una invocación llamando a la *JSP* del servidor *Tomcat*, la imagen no se mostraría indicándonos el navegador que no la encuentra. En cambio si hacemos la llamada a través del servidor Apache, la imagen se verá correctamente debido a que esta es servida por el servidor Apache y la *JSP* es resulta por el *Tomcat*.

Para ello vamos a configurar *Tomcat* y Apache para que trabajen conjuntamente.

Esta prueba será idéntica a la que se realizará para la configuración del aplicativo alpha.

## Pasos a realizar:

1. Verificamos que *Tomcat* está funcionando correctamente. Al realizar esta llamada veremos la página de inicio de *Tomcat*.

`http://localhost:8080/`

2. Creación de un directorio Virtual en *Apache*

- 2.1. Dentro de la ruta `$APACHE_HOME/conf/extra/` podremos encontrar un fichero de ejemplo de configuración de virtualHost, denominado `httpd-vhosts.conf`. Este fichero nos servirá también para configurar el virtualHost que utilizaremos en la aplicación alpha.

Para ello conviene conocer que algunos de las directivas que aparecen en el mismo.

**ServerName** Nombre de host y número de puerto que el servidor usa para identificarse. La directiva ServerName especifica el nombre de host y el puerto que usa el servidor para identificarse.

**DocumentRoot** Directorio principal que contiene la estructura de directorios visible desde la Web. Esta directiva especifica el directorio desde el cuál httpd servirá los ficheros.

**Directory index**: Engloba a un grupo de directivas que se aplicarán solamente al directorio del sistema de ficheros especificado y a sus subdirectorios. Aquí es donde se indica el nombre de la página principal del sitio.

**ServerAdmin** Dirección de email que el servidor incluye en los mensajes de error que se envían al cliente.

**ErrorLog** Ubicación del fichero en el que se almacenan los mensajes de error

**CustomLog** Ubicación de donde esta el archivo en el cual se registran los accesos al sitio.

Vamos a crear a partir de una única dirección IP un Host virtual llamado `alpha.com.localhost` para resolver estos dominios virtuales se debe configurar el DNS para que contenga el nombre y la dirección IP (En este caso única para todos los Virtual Host).

Si no tenemos DNS y queremos probar la configuración en una máquina local, lo más fácil es configurar la Ip en el archivo de Hosts en la ruta `C:\WINDOWS\system32\drivers\etc`

	<code>127.0.0.1</code>	<code>alpha.com.localhost</code>	
--	------------------------	----------------------------------	--

Tabla 7 Fichero host de Windows.



*Si se desea que el virtualHost sea accesible desde Internet primeramente tendríamos que registrar este dominio en el organismo regulador de los mismos <http://www.nic.es>. Y configurar el DNS en el ISP que tengamos contratado.*

- 2.2. Creamos un directorio para alojar los contenidos del dominio **alpha.com.localhost** En este ejemplo vamos a crearlo dentro del directorio Home de Apache **\$APACHE\_HOME/** y lo llamaremos **www**. Dentro de este directorio situaremos tantos directorios, como dominios se deseen tener en el servidor.

En este caso **\$APACHE\_HOME\www\alpha.com.localhost**

- 2.3. En el interior creamos un directorio con el nombre del aplicativo web de Tomcat (Contexto de la aplicación) y en su interior la carpeta de imágenes img. Situando en esta última el logotipo de la Asociación Ciudades Kyosei.

El resultado sería:

**\$APACHE\_HOME\www\alpha.com.localhost\interconTomcatApache\img\logo.jpg**

Una vez visto un ejemplo de configuración del virtualHost procedemos a configurarle para ello abriremos el archivo **\$TOMCAT\_HOME/conf/jk/mod\_jk.conf**. (Este archivo es incluido en el fichero **http.conf** de Apache)

Primeramente en este archivo añadiremos permisos de acceso al directorio **www** creado anteriormente , pero evitaremos que se puedan listar los directorios. También añadiremos la directiva que permite la resolución por nombre del virtualHost. Para ello añadiremos las líneas siguientes al archivo.

```
<Directory "www/">
    Options -Indexes -Includes -FollowSymLinks
    Multiviews
    AllowOverride None
    Order Deny,Allow
    Allow from all
</Directory>
NameVirtualHost 127.0.0.1:80
```

Tabla 8 Permisos para el directorio de publicación www del fichero mod\_jk.conf

Por último añadiremos el virtualHost con la directiva **JKmount** que establecerá la relación entre la aplicación web desplegada en el servidor Tomcat con el dominio registrado en apache, indicando al

Servidor apache que todas las JSPs se redirigidas al servidor Tomcat a través del conector que se configuró en el archivo **workers.properties**.

```
“
<VirtualHost 127.0.0.1:80>
    ServerAdmin admin@ckyosei.org
    DocumentRoot "www/alpha.com.localhost"
    ServerName alpha.com.localhost
    ErrorLog "logs/alpha.com.localhost-error.log"
    CustomLog "logs/alpha.com.localhost-access.log"
    common

    JkMount /interconApacheTomcat/*.jsp ajp13
</VirtualHost>
”
```

Tabla 9 Creación de un Virutal Host en mod\_jk.conf

A continuación se muestra el archivo **\$TOMCAT\_HOME/conf/jk/mod\_jk.conf** completo:

```
“
<IfModule !mod_jk.c>
    LoadModule jk_module
    "$APACHE_HOME/modules/mod_jk.so"

</IfModule>

JkWorkersFile
"C:/CKyosei/Herramientas/Tomcat6.0/conf/jk/workers.properties"

JkLogFile
"C:/CKyosei/Herramientas/Tomcat6.0/logs/mod_jk.log"

JkLogLevel emerg

# Should mod_jk send SSL information to Tomcat
(default is On)

#JkExtractSSL On
”
```



```
# What is the indicator for SSL (default is HTTPS)
#JkHTTPSIndicator HTTPS

# What is the indicator for SSL session (default is
SSL_SESSION_ID)

#JkSESSIONIndicator SSL_SESSION_ID

# What is the indicator for client SSL cipher suit
(default is SSL_CIPHER)

#JkCIPHERIndicator SSL_CIPHER

#What is the indicator for the client SSL
certificated (default is #SSL_CLIENT_CERT)

#JkCERTSIndicator SSL_CLIENT_CERT

# Use name-based virtual hosting.

NameVirtualHost 127.0.0.1:80

#Directorio raíz de contenidos de los virtualHost

<Directory "www/">

    Options -Indexes -Includes -FollowSymLinks
    Multiviews

        AllowOverride None

        Order Deny,Allow

        Allow from all

</Directory>

#virtual para el domino alpha.com.localhost

<VirtualHost 127.0.0.1:80>

    ServerAdmin webmaster@ alpha.com.localhost

    DocumentRoot "www/alpha.com.localhost"

    ServerName alpha.com.localhost

    ErrorLog "logs/alpha.com.localhost-error.log"

    CustomLog "logs/alpha.com.localhost-access.log"
    common

    JkMount /interconApacheTomcat/*.jsp ajp13

</VirtualHost>
```



**Tabla 10 Fichero completo mod\_jk.conf**

3. Con la directiva *Directory* definimos opciones que se aplican al directorio indicado y sus subdirectorios. Lo habitual es configurar unos permisos por defecto muy restrictivos para el directorio raíz “/” y darle permisos a los diferentes subdirectorios.

La configuración del directorio raíz se encuentra por defecto en apache en el fichero de configuración C:\CKyosei\Herramientas\Apache2.2\conf\httpd.conf, cambiaremos la configuración por defecto por lo siguiente.

```
<Directory />
    Options SymLinksIfOwnerMatch
    AllowOverride None
</Directory>
```



**Tabla 11 Restringir permisos sobre directorio raíz de Apache en el httpd.conf**

La opción *SymLinksIfOwnerMatch* sólo permite que puedan seguirse los enlaces simbólicos si el propietario del link y el del archivo apuntado es el mismo.

La opción *FollowSymLinks* permite que puedan seguirse los enlaces simbólicos. Por seguridad, esta opción la desactivamos.

Con la opción *Indexes*, si se solicita un directorio y no existe la página especificada con *DirectoryIndex* (index.html, index.htm, etc.), Apache mostrará el contenido del directorio.

Es preferible desactivar esta opción con -Indexes para no arriesgarnos a que el usuario pueda acceder a archivos del directorio, Apache mostrará el error *Forbidden - You don't have permission to access / on this server*.

4. Creación aplicación web y Despliegue en Tomcat.

Creamos una aplicación Web con cualquier entorno de desarrollo.

Crearemos una sencilla JSP que haga una invocación a una imagen.



“

```
<html>
<body>
<h2>Esta JSP es resuelta por el Servidor Tomcat</h2>

</body>
</html>
```

”

Tabla 12 JSP de prueba de configuración Apache-Tomcat

### 5. Prueba de conexión.

#### 5.1. Arrancamos los servidores Tomcat y Apache.

Si intentáramos invocar a la raíz del servidor apache directamente obtendremos un mensaje de denegación de permisos. Este mensaje es correcto debido a la configuración que hemos realizado anteriormente.

<http://alpha.com.localhost/>

**Forbidden**

You don't have permission to access / on this server.

Figura 26 Denegación de servicio directorio raíz apache

Al igual ocurre si invocamos directamente a la aplicación Web.

<http://alpha.com.localhost/interconApacheTomcat/>

**Forbidden**

You don't have permission to access / on this server.

Figura 27 Denegación de servicio directorio raíz apache

Si colocamos una página índice **index.html** en el directorio  
**\$APACHE\_HOME\www\alpha.com.localhost\interconTomcatApache\** que reenvié a la página  
**index.jsp** de la aplicación **interconTomcatApache**.

```
“  
<html>  
  <head>  
    <META HTTP-EQUIV="Refresh"  
          CONTENT="0; URL=index.jsp">  
  </head>  
</html>
```

Tabla 13 Html de bienvenida que reenvia la información a la JSP.

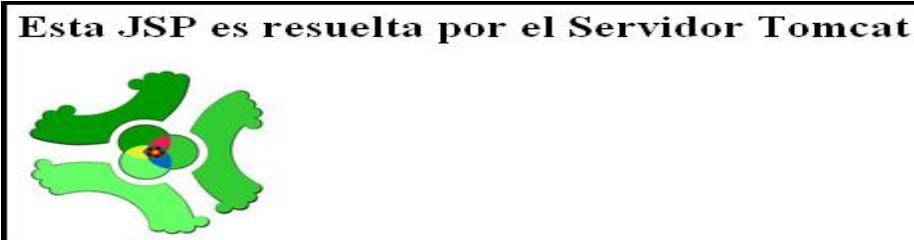


Figura 28 página de interconexión Apache Tomcat.

Si observamos la figura anterior, se puede leer el texto de la página JSP, a continuación vemos la imagen del logo de la asociación Ckyosei. Si vemos los accesos en el fichero de log configurado en el virtualHost **alpha.com.localhost-access.log** se puede observar como es Apache quien sirve la imagen.



“

```
127.0.0.1 - - [09/Jun/2009:14:23:00 +0200] "GET
/interconApacheTomcat/index.html HTTP/1.1" 200 128
127.0.0.1 - - [09/Jun/2009:14:23:00 +0200] "GET
/interconApacheTomcat/index.jsp HTTP/1.1" 200 108
127.0.0.1 - - [09/Jun/2009:14:23:00 +0200] "GET
/interconApacheTomcat/img/logo.jpg HTTP/1.1" 200 11205
```

”

Tabla 14 Información del Log alpha.com.localhost-access.log

Si invocamos directamente al servidor *Tomcat* comprobaremos que la imagen no es mostrada puesto que hemos colocado el contenido estático del aplicativo en él.

Como se observa en la URL siguiente, en lugar de usar el puerto 80 (puerto por defecto de los servidores Web) de Apache llamamos al puerto de *Tomcat* (8080 por defecto).

[http://alpha.com.localhost\[:puerto\]/interconApacheTomcat/index.jsp](http://alpha.com.localhost[:puerto]/interconApacheTomcat/index.jsp)

Esta JSP es resuelta por el Servidor Tomcat



Figura 29 Llamada directa a Tomcat





## **6.1.10. Instalación de PHP para Apache**

### **6.1.10.1. Introducción**

**PHP (PHP Hypertext Pre-processor)** es un lenguaje de programación usado normalmente para la creación de contenido para sitios web. Se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores o creación de contenido dinámico para sitios web.

### **6.1.10.2. Requerimientos**

- **PHP para Windows**, versión estable de la serie 5.2.x (5.2.6 en el momento de redacción de este manual) **php-5.2.x-Win32.zip**:  
<http://www.php.net/downloads.php>

### **6.1.10.3. Proceso de Configuración**

1. Una vez descargado el módulo de php **php-5.2.x-Win32.zip**, lo descomprimimos en:

**C:\CKyosei\Herramientas\php-5.2.6-Win32**

2. Establecer la variable de entorno **PHP\_HOME** al directorio donde haya descomprimido PHP. La añadimos también a la variable **PATH** de **Windows** para que encuentre los ejecutables.
3. Creamos una copia del fichero **php.ini.recommended** a uno nuevo que se llame **php.ini**.
4. En el fichero **httpd.conf** de **Apache**, que está en **\$APACHE\_HOME/conf**, debemos incluir una referencia a este fichero. Para realizar esta operación añadimos al final del fichero las siguientes líneas:

“

```
LoadModule php5_module "C:/CKyosei/Herramientas/php-5.2.6-Win32/php5apache2_2.dll"  
AddType application/x-httpd-php .php  
  
# configure the path to php.ini  
phpinidir "C:/CKyosei/Herramientas/php-5.2.6-Win32"
```

”

**Figura 30 Configuración de PHP en httpd.conf**

5. En este mismo fichero **httpd.conf** de *Apache* buscamos las líneas `DirectoryIndex` y le añadimos al final `index.php`. Con esta operación le indicamos a *Apache* que busque por defecto dentro de los directorios además de `index.html` la página `index.php`.

“

```
#añadiremos la página de indice index.php  
<IfModule dir_module>  
    DirectoryIndex index.html index.php  
</IfModule>
```

”

6. Reiniciamos el servidor web *Apache*



#### **6.1.10.4. OPCIONAL: Prueba de Configuración PHP**

Para probar la correcta configuración vamos a usar el virtualHost creado anteriormente en la configuración de Apache-Tomcat.

Para la realización de esta prueba vamos a crear una página PHP **index.php**, y la vamos a colocar en el directorio de contenidos **\$APACHE\_HOME\www\alpha.com.localhost\**

```
<?php phpinfo(); ?>
```

Si invocamos al ejemplo completo obtendremos la página con la configuración de php del servidor Apache.

<http://localhost/Portal/index.php>

## PHP Version 5.2.9-1



<b>System</b>	Windows NT RZP001 5.1 build 2600
<b>Build Date</b>	Mar 5 2009 20:01:54
<b>Configure Command</b>	cscript/nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\ sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\ sdk,shared" "--enable-htscanner=shared" "--without-pi3web"
<b>Server API</b>	Apache 2.0 Handler
<b>Virtual Directory Support</b>	enabled
<b>Configuration File (php.ini) Path</b>	C:\WINDOWS
<b>Loaded Configuration File</b>	
<b>Scan this dir for additional .ini files</b>	(none)
<b>additional .ini files parsed</b>	(none)
<b>PHP API</b>	20041225
<b>PHP Extension</b>	20060613
<b>Zend Extension</b>	220060519
<b>Debug Build</b>	no
<b>Thread Safety</b>	enabled
<b>Zend Memory Manager</b>	enabled
<b>IPv6 Support</b>	enabled
<b>Registered PHP Streams</b>	php, file, data, http, ftp, compress.zlib
<b>Registered Stream Socket Transports</b>	tcp, udp
<b>Registered Stream Filters</b>	convert.iconv.* , string.rot13, string.toupper, string.tolower, string.strip_tags, convert.* , consumed, zlib.*

Figura 31 Respuesta Servidor Apache a index..php



## 6.1.11. Configuración seguridad para Apache

### 6.1.11.1. Introducción

La seguridad en un servidor apache puede ser establecida de varias formas desde una autenticación básica *Http*, en la que se solicitará clave, usuario para poder acceder a las páginas solicitadas a técnicas de encriptación mediante claves públicas – privadas *PKI*.

La tecnología *PKI* permite a los usuarios identificarse frente a otros usuarios y usar la información de los certificados de identidad (por ejemplo, las claves públicas de otros usuarios) para cifrar y descifrar mensajes, firmar digitalmente información, garantizar el no repudio de un envío, etc.

En el proyecto vamos a emplear esta última, para ello vamos a introducir previamente a la configuración unos conceptos fundamentales.

### 6.1.11.2. Secure Sockets Layer (SSL)

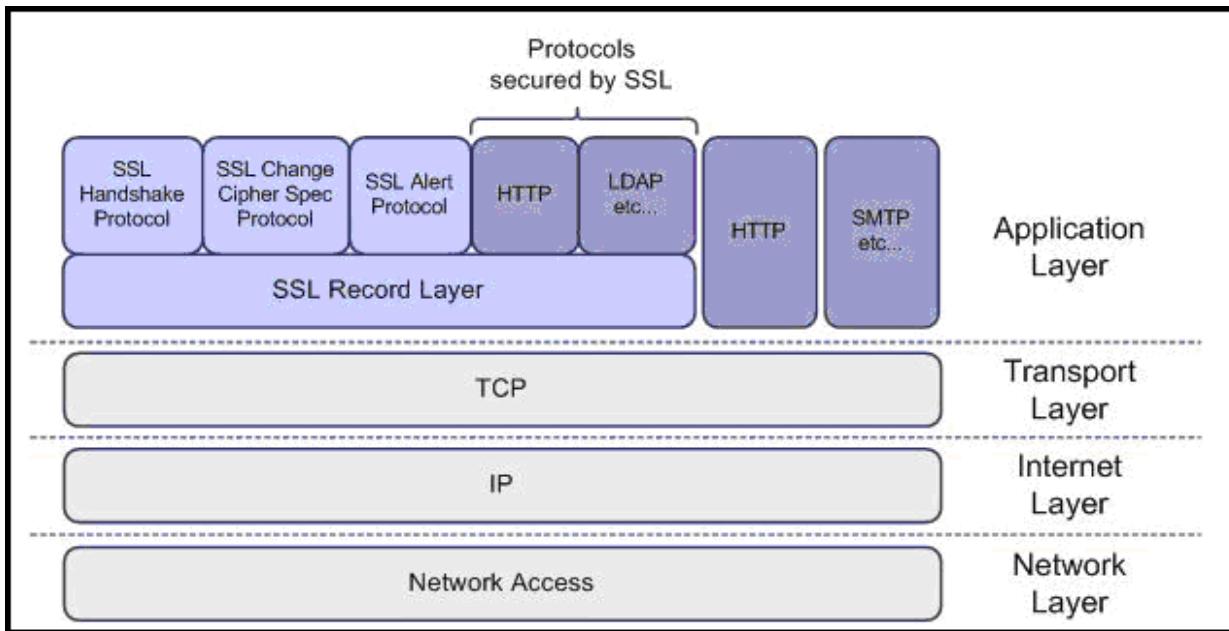
*Secure Sockets Layer (SSL)* es el protocolo más ampliamente conocido que ofrece la confidencialidad y fiabilidad de la comunicación cliente-servidor a través de Internet. *SSL* en sí es conceptualmente bastante simple:

Se negocia la criptografía de claves y algoritmos entre los dos interlocutores de una comunicación y se establece un túnel cifrado a través del cual otros protocolos (como *HTTP*) pueden ser transportados.

Opcionalmente, también se puede autenticar con SSL ambos lados de la comunicación mediante el uso de certificados.

SSL es un protocolo de capa y consta de cuatro subprotocolos:

- ü *SSL Handshake Protocol*
- ü *SSL Change Cipher Spec Protocol*
- ü *SSL Alert Protocol*
- ü *SSL Record Layer*



**Figura 32 SL sub-protocols in the TCP/IP model (Imagen SecurityFocus)**

### **El Protocolo SSL Handshake**

Durante el protocolo **SSL Handshake** el cliente y el servidor intercambian una serie de mensajes para negociar la seguridad. Este protocolo sigue las siguientes seis fases (de manera muy resumida):

- ü La fase Hola, usada para ponerse de acuerdo sobre el conjunto de algoritmos para la autenticación.
- ü La fase de intercambio de claves, en la que intercambia información sobre las claves, de modo que al final ambas partes comparten una clave maestra.
- ü La fase de producción de clave de sesión, que será la usada para cifrar los datos intercambiados.
- ü La fase de verificación del servidor, presente sólo cuando se usa RSA como algoritmo de intercambio de claves, y sirve para que el cliente autentique al servidor.
- ü La fase de autenticación del cliente, en la que el servidor solicita al cliente un certificado X.509 (si es necesaria la autenticación de cliente).
- ü Por último, la fase de fin, que indica que ya se puede comenzar la sesión segura.

### **El Protocolo SSL Record**

El Protocolo SSL Record especifica la forma de encapsular los datos transmitidos y recibidos. Los datos del protocolo se componen de tres componentes:



- ü MAC-DATA, el código de autenticación del mensaje.
- ü ACTUAL-DATA, los datos de aplicación a transmitir.
- ü PADDING-DATA, los datos requeridos para llenar el mensaje cuando se usa cifrado en bloque.

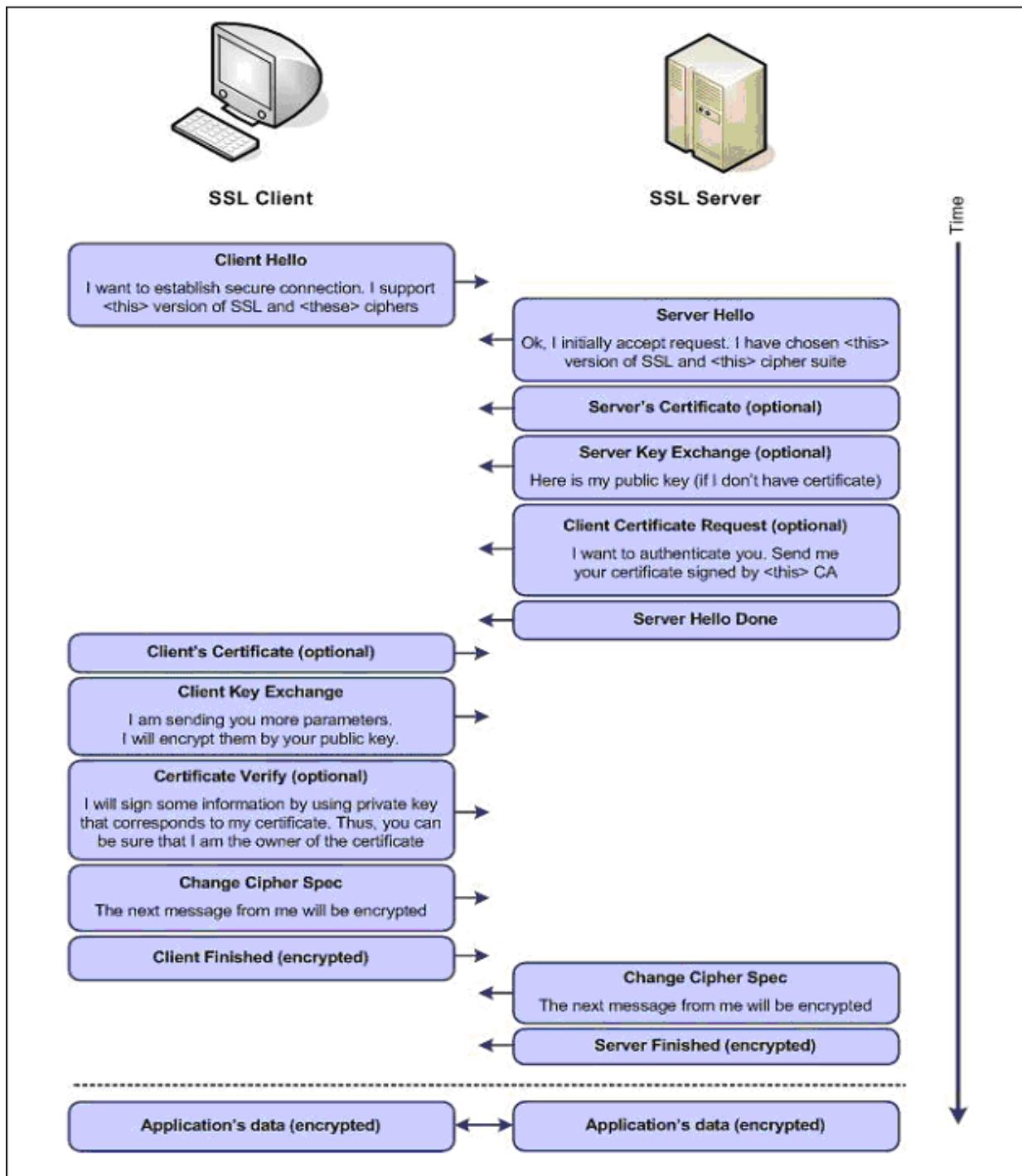


Figura 33 Establecimiento de Conexión SSL paso a paso (Imagen securityfocus)

### 6.1.11.3. Proceso de instalación

1. En el fichero **httpd.conf** de Apache que está en **\$APACHE\_HOME/conf** debemos añadir el módulo SSL. Para realizar esta operación activamos las líneas:

```
“
LoadModule ssl_module modules/mod_ssl.so
Include conf/extra/httpd-ssl.conf
”
```

2. Copiar las librerías **libeay32.dll** y **ssleay32.dll**, desde el directorio **\$APACHE\_HOME/bin** al directorio system32 de la instalación de Windows, normalmente en **C:\WINDOWS\system32**.
3. En este punto generaremos los certificados. Los certificados digitales los emiten autoridades certificadoras. Las autoridades certificadoras principales están instaladas ya en los navegadores y se encargan de dar fe de que el certificado emitido es válido.

Estos certificados no son gratuitos. El precio aproximado de un certificado de servidor ronda sobre los 1000€ dependiendo de la entidad certificadora es la FNMT, Verisign, etc., por lo que nosotros vamos a crear nuestro propio certificado autofirmado. Para crear el certificado vamos usar el paquete openssl que venía con la distribución de Apache. Este incluye herramientas de administración y librerías relacionadas con la criptografía.

Más adelante en el siguiente punto de la memoria veremos como crear nuestra propia entidad certificadora mediante SSL. Para ello tendremos que realizar la instalación de OpenSSL y Perl.

En este punto vamos a usar Certificados autofirmados por lo que simplemente usaremos un script de generación de los mismos. Un certificado autofirmado tiene la problemática, que los navegadores no reconocerán ninguna entidad certificadora por lo que nos dará un aviso.



Figura 34 Aviso Conexión segura



A continuación se muestra el script que automatizará todo el proceso de creación de certificado del servidor y clave privada:

```
echo off

if not defined apache_dir set
apache_dir=C:\CKyosei\Herramientas\Apache2.2

if not defined apache_conf_dir set
apache_conf_dir=%apache_dir%\conf

if not defined openssl_conf set
openssl_conf=%apache_conf_dir%\openssl.cnf

if not defined openssl_opts set openssl_opts=-config
"%openssl_conf%"

if not defined openssl set openssl=%apache_dir%\bin\openssl.exe

if not exist "%apache_dir%" (
echo Directory not found: "%apache_dir%"
goto :eof
)

if not exist "%apache_conf_dir%" (
echo Directory not found: "%apache_conf_dir%"
goto :eof
)

if not exist "%openssl_conf%" (
echo File not found: "%openssl_conf%"
goto :eof
)

if not exist "%openssl%" (
echo File not found: "%openssl%"
goto :eof
)

pushd "%apache_conf_dir%\ssl"

"%openssl%" req %openssl_opts% -new -out server.csr || goto
:eof
```

```

"%openssl% rsa -in privkey.pem -out server.key || goto :eof
"%openssl% x509 -in server.csr -out server.crt -req -signkey
server.key -days 365
del .rnd
popd

```



**Tabla 15 CKyosei\_SSL\_script.bat**

4. Creamos en el directorio **\$APACHE\_HOME** un fichero **CKyosei\_SSL\_script.bat** que contenga el script.
5. Crear la carpeta SSL dentro de **\$APACHE\_HOME /conf/**
6. Ejecutar el script de creación del certificado, introduciendo claves e informaciones según sean requeridas. El Common Name que se nos solicita debe coincidir exactamente con el nombre del dominio con que hayamos configurado en Apache, en nuestro caso vamos a usar el dominio alpha.com.localhost que usamos en las configuraciones anteriores. El script genera diversos archivos en el directorio **\$APACHE\_HOME/conf/ssl**.
7. .Configuración del archivo **\$APACHE\_HOME/conf/extra/httpd-ssl.conf**. Este fichero es el equivalente al httpd.conf pero para conexiones SSL –realizadas a través del puerto 443 por defecto.
8. Primeramente realizaremos una copia de seguridad del fichero **httpd-ssl.conf** a **httpd-ssl.conf.original**.
9. Creamos un nuevo archivo **httpd-ssl.conf** con un virtualHost que será casi identifico al creado en la interconexión de apache-tomcat. Solo que servirá las peticiones seguras al puerto 443. (Normalmente solo uno de los 2 virtualHost estará activo, se suele hacer que las invocaciones al protocolo http sean enviadas automáticamente al protocolo https, es decir, al puerto 443.)

Para consultar las directivas ssl de Apache puede hacerlo en la URL [http://httpd.apache.org/docs/2.2/mod/mod\\_ssl.html](http://httpd.apache.org/docs/2.2/mod/mod_ssl.html).



“

```
SSLMutex default
SSLRandomSeed startup builtin
SSLSessionCache none
SSLOptions +StdEnvVars +ExportCertData
Listen 443
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl .crl
<Virtualhost _default_:443>
SSLEngine on
SSLCipherSuite
ALL:!ADH:!EXP56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
SSLCertificateFile
"C:/CKyosei/Herramientas/Apache2.2/conf/ssl/server.crt"
SSLCertificateKeyFile
"C:/CKyosei/Herramientas/Apache2.2/conf/ssl/server.key"
#SSLVerifyClient optional_no_ca
JkMount /interconApacheTomcat/*.jsp ajp13
<Location "/interconApacheTomcat/WEB-INF/">
Deny from all
</Location>
</Virtualhost>
```

”

Tabla 16 httpd-ssl.conf

En las primeras líneas indicamos dónde se encuentra el certificado de servidor y solicitamos el certificado de cliente (sin especificar la CA). Especificamos asimismo el puerto en que funcionarán las conexiones SSL. En caso de que el puerto estándar 443 esté siendo ya utilizado por algún otro servicio, deberás especificar el puerto en que funcionará el SSL, por ejemplo el 444.

En las siguientes líneas damos permisos a la aplicación Tomcat que se quiere que trabaje mediante conexiones SSL. En este caso probaremos la aplicación Web interconApacheTomcat que configuramos anteriormente mediante SSL.

#### 10. Configuración del archivo **\$TOMCAT\_HOME/conf/jk/mod\_jk.conf**

Al igual que en el archivo **httpd-ssl.conf**, tendremos que indicar al conector de Tomcat que vamos a trabajar con SSL.

Si nos fijamos en el archivo **mod\_jk.conf**, el fichero **\$TOMCAT\_HOME/conf/jk/mod\_jk.conf** tenía las directivas de SSL comentadas.

Estas directivas eran las siguientes:

```
“
#
# Should mod_jk send SSL information to Tomcat (default is On)
JkExtractSSL On

# What is the indicator for SSL (default is HTTPS)
JkHTTPSIndicator HTTPS

# What is the indicator for SSL session (default is
SSL_SESSION_ID)

JkSESSIONIndicator SSL_SESSION_ID

# What is the indicator for client SSL cipher suit (default is
SSL_CIPHER)

JkCIPHERIndicator SSL_CIPHER

# What is the indicator for the client SSL certificated
(default is SSL_CLIENT_CERT)

JkCERTSIndicator SSL_CLIENT_CERT
”
```

Tabla 17 Directivas SSL mod\_jk.conf

El fichero resultante quedaría como el siguiente:

```
“
<IfModule !mod_jk.c>
    LoadModule jk_module "$APACHE_HOME/modules/mod_jk.so"
”
```



```

</IfModule>

JkWorkersFile
"C:/CKyosei/Herramientas/Tomcat6.0/conf/jk/workers.properties"

JkLogFile "C:/CKyosei/Herramientas/Tomcat6.0/logs/mod_jk.log"
JkLogLevel emerg

# Should mod_jk send SSL information to Tomcat (default is On)
JkExtractSSL On

# What is the indicator for SSL (default is HTTPS)
JkHTTPSIndicator HTTPS

# What is the indicator for SSL session (default is
SSL_SESSION_ID)
JkSESSIONIndicator SSL_SESSION_ID

# What is the indicator for client SSL cipher suit (default is
SSL_CIPHER)
JkCIPHERIndicator SSL_CIPHER

# What is the indicator for the client SSL certificated
(default is SSL_CLIENT_CERT)
JkCERTSIndicator SSL_CLIENT_CERT

# Use name-based virtual hosting.

NameVirtualHost 127.0.0.1:80

#Directorio raíz de contenidos de los virtualHost
<Directory "www/">
    Options -Indexes -Includes -FollowSymLinks Multiviews
    AllowOverride None
    Order Deny,Allow
    Allow from all
</Directory>

```



**Tabla 18 Fichero resultante mod\_jk.conf**

Como se puede observar simplemente hemos introducido las directivas SSL y hemos quitado la definición del virtualHost, al haberlo definido en el archivo **httpd-ssl.conf**, aunque la definición podría estar en cualquiera de los 2 ficheros.

11. Reiniciamos el servidor web Apache.
12. Para probar solicitaríamos en el navegador una conexión segura (especificando el puerto en caso de que no sea el estándar):

`https://localhost[:puerto]`

`https://localhost[:puerto]/interconApacheTomcat/`

Puede ocurrir el caso de que se desee que Apache solicite un certificado cliente para establecer la autenticación de quién se conecta a la aplicación.

Con estas directivas le indicamos al servidor apache que debe solicitar un certificado al cliente, y además se puede obligar a que este certificado haya sido emitido por una determinada entidad certificadora.

Para realizar esta operación simplemente tendríamos que añadir las siguientes directivas al fichero `httpd-ssl.conf`

```
“
sslverifyclient require
SSLVerifyDepth 10
#SSLCACertificateFile
"C:/CKyosei/Herramientas/Apache2.2/conf/ssl/cacert.crt"
”
```

**Tabla 19 Directivas SSL mod\_jk.conf**

Tras solicitarnos el correspondiente certificado de cliente, se nos mostrará la página de inicio de Apache en una conexión segura.

Si no tenemos instalado el certificado cliente obtendremos un error similar la siguiente:

(Código de error: `ssl_error_handshake_failure_alert`)

En caso de no tener la entidad certificadora instalada en el navegador podremos obtener un mensaje como el siguiente:

(Código de error: `sec_error_untrusted_issuer`)



### **6.1.11.4. Instalación de OpenSSL**

#### **6.1.11.4.1. Introducción**

En el apartado anterior vimos como se establecía la seguridad en Apache, usando certificados autofirmados, y la problemática que conllevaba. En este punto vamos a ver como podemos solucionarlo usando una entidad certificadora propia. Para ello vamos a usar *OpenSSL*.

El software *OpenSSL* es un proyecto de software desarrollado por los miembros de la comunidad Open Source. Es un robusto juego de herramientas que le ayudan a implementar el *Secure Sockets Layer (SSL)* en su sistema, así como otros protocolos relacionados con la seguridad, como el *Transport Layer Security (TLS)*.

*OpenSSL* nos provee de una infraestructura de cifrado y certificación digital que podemos aprovechar para tareas tan diversas como cifrar comunicaciones, autenticar los dos extremos de la conexión, estampas de firma de tiempo y firma de código.

#### **6.1.11.4.2. Requerimientos**

- Paquete de instalación de Visual C++ 2008 Redistributables, **vcredist\_x86.exe**:  
<http://www.microsoft.com/downloads/details.aspx?familyid=9B2DA534-3E03-4391-8A4D-074B9F2BC1BF&displaylang=en>
- Paquete de instalación de OpenSSL, **Win32OpenSSL-0\_9\_8i.exe**  
[http://www.slproweb.com/download/Win32OpenSSL-0\\_9\\_8i.exe](http://www.slproweb.com/download/Win32OpenSSL-0_9_8i.exe)

*Hemos de instalar la misma versión de OpenSSL que disponga el servidor Apache (el nombre del paquete de instalación de Apache indica la versión de OpenSSL que utiliza), ya que la instalación sobrescribe las librerías libeay32.dll y ssleay32.dll del System32 de Windows.*

---

#### 6.1.11.4.3. Proceso de instalación

1. La versión 0.9.8i de *OpenSSL* requiere de la instalación previa de los Visual C++ 2008 Redistributable. Por ello, empezamos ejecutando el archivo **Win32OpenSSL-0\_9\_8i.exe**.
2. Ejecutar **Win32OpenSSL-0\_9\_8i.exe**. Aceptamos primeramente la licencia y pulsamos **Next**.
3. En la siguiente pantalla indicamos la ruta de instalación,  
**C:\CKyosei\Herramientas\OpenSSL** y proseguimos con la instalación pulsando **Next**.

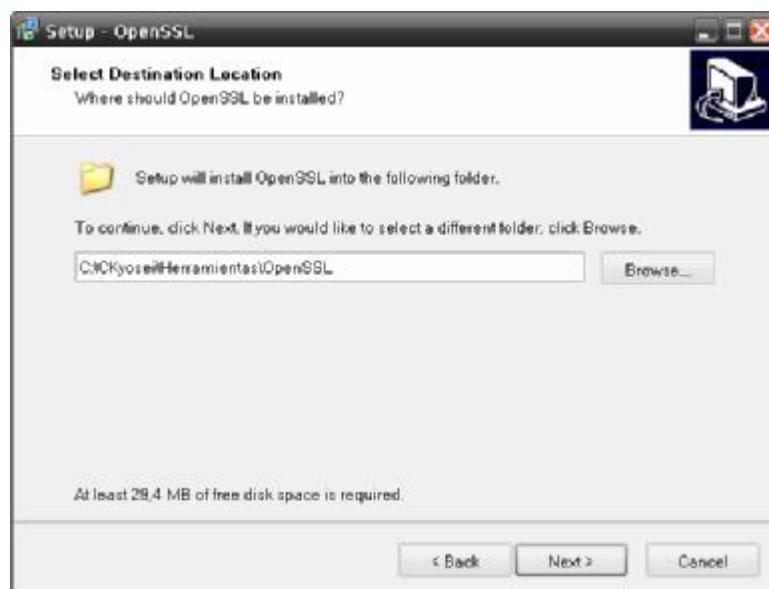


Figura 35 Pantalla de selección de directorio para OpenSSL

4. Indicamos que deben copiarse las bibliotecas DLL en el directorio del sistema de Windows, pulsamos **Next** nuevamente y al fin la opción **Install**, que instalará el programa en el directorio indicado.

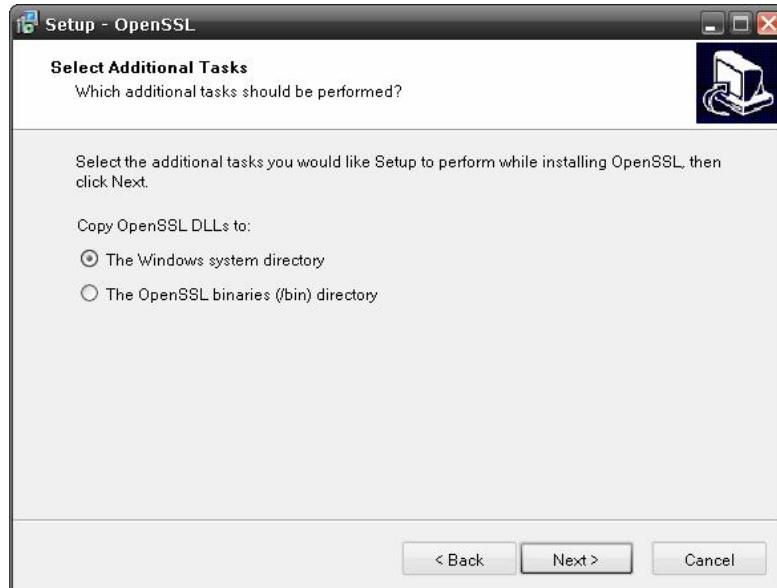


Figura 36 Selección de tareas adicionales de OpenSSL

5. Crear la variable de entorno `OPENSSL_HOME` al directorio donde se haya instalado OpenSSL, y añadir el directorio `bin` de OpenSSL a la variable `PATH`, agregándole `; %OPENSSL_HOME%\bin`, para que encuentre el ejecutable `openssl.exe`, así como los scripts de Perl que nos suministra *OpenSSL* para facilitarnos las labores de creación de certificados, su firma, etc.

También crearemos la variable `OPENSSL_CONF`, que indica dónde se encuentra por defecto el fichero de configuración, para que apunte al directorio `$APACHE_HOME\conf\ssl\ssl.conf`. Así utilizaremos el fichero de configuración incluido con la instalación de Apache.

## 6.1.11.5. Instalación de Active Perl

### 6.1.11.5.1. Introducción

*Active Perl* es un intérprete del lenguaje PERL para el entorno Windows, que nos ofrecerá principalmente la funcionalidad propia de llamadas específicas de la API Win32.

### 6.1.11.5.2. Requerimientos

- Interprete de Perl, **ActivePerl-5.10.0.1000-Beta-MSWin32-x86-283192.msi**

<http://downloads.activestate.com/ActivePerl/Windows/5.10/ActivePerl-5.10.0.1000-Beta-MSWin32-x86-283192.msi>

### 6.1.11.5.3. Proceso de instalación

1. Ejecutar **ActivePerl-5.10.0.1000-Beta-MSWin32-x86-283192.msi**. Aceptamos la licencia y pulsamos **Next**.
2. En la siguiente pantalla indicamos la ruta de instalación **C:\CKyosei\Herramientas\Perl\** y pulsamos **Next**.
3. Aceptamos la creación de las variables de entorno y pulsamos **Next**.

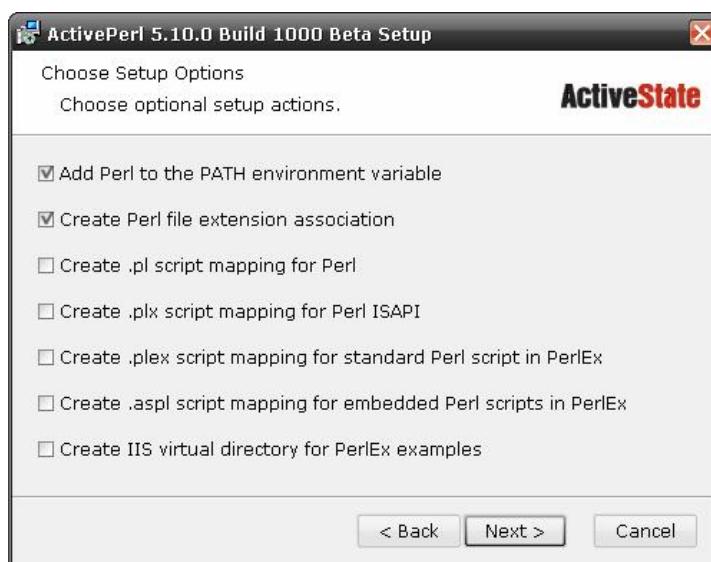


Figura 37 Pantalla instalación ActivePerl

4. Finalmente pulsamos la opción **Install**, que instalará el programa en el directorio indicado.



## 6.1.12. Creación Entidades Certificadoras y certificados firmados

### 6.1.12.1. Introducción

Una Entidad Certificadora CA (*Certificate Authority*) es una entidad que emite certificados, al firmar con su propia firma digital las firmas de terceros. Un CA debe guardar las claves, los certificados y un registro completo de los certificados que han sido revocados. La firma de un certificado por un CA implica que el CA está avalando la identidad de la entidad a quien se le firma su certificado, de la misma forma que un notario lo hace en una notaría.

En otras palabras: una CA es el organismo encargado de garantizar que un certificado pertenece a su legítimo propietario.

Por otro lado, un certificado digital es un documento digital que verifica que una clave pública pertenece a una determinada persona o entidad.

Los certificados digitales están sustentados en 3 principios fundamentales:

- Criptografía de clave pública
- Firma digital
- Elementos confiables

*OpenSSL* se configura mediante el archivo **openssl.cnf**, en el cual podemos especificar las carpetas donde queremos que se guarden los certificados y las claves e información que incluiremos en los certificados en la sección **req\_distinguished\_name**.

Para usar el producto se han desarrollado unos script en Perl que permiten su fácil uso. Estos script son **CA.pl** y **CA.sh**, y normalmente se encuentran bajo la carpeta **\$OPENSSL\_HOME/bin** o en **\$OPENSSL\_HOME/apps**, aunque su localización puede variar según la distribución.

### 6.1.12.2. Implementación de una Autoridad Certificadora

#### 6.1.12.2.1. Introducción

Para que un certificado sea válido a nivel global, este debe de estar firmado por una entidad certificadora válida también en todo el mundo. Es decir, que se encuentre en el almacén de certificados de la mayor parte de sistemas operativos para que los clientes puedan validar la autenticidad de los certificados de servidor.

Podemos, así, seguir varios caminos si queremos certificar nuestra aplicación:

- Comprar los certificados firmados por una compañía como *Verisign*
- Usar CAcert que nos provee de certificados raíz y de cliente gratuitos (este certificado todavía no aparece en todos los almacenes de confianza, aunque se está estudiando incluirlo en Linux), pero que no es muy compatible con Microsoft.
- Generar nuestra propia Autoridad Certificadora si los certificados van a ser para uso interno, podemos así instalar el certificado de nuestra CA (autoridad certificadora) en los clientes en los que queramos usar los mecanismos de *OpenSSL*.

Esta última opción es la que vamos a emplear para generar nuestros certificados digitales.

### 6.1.12.2. Creación de Entidades Autorizadoras

1. El primer paso es crear la estructura para nuestra Autoridad Certificadora.
  - 1.1. Primeramente crearemos un directorio donde alojaremos toda la estructura, por ejemplo **C:\CKyosei\Seguridad**
  - 1.2. Ejecutaremos el comando **CA.pl -newca** para crear la entidad Autorizadora.

```
C:\CKyosei\Seguridad>CA.pl -newca
CA certificate filename (or enter to create): ENTER
Making CA certificate ...
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:CLAVE
Verifying - Enter PEM pass phrase:CLAVE

-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:MADRID
Locality Name (eg, city) []:ALCALA
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CKYOSEI
Organizational Unit Name (eg, section) []:ENTER
Common Name (eg, YOUR name) []:ckyosei.org
Email Address []:info@ckyosei.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:ENTER
An optional company name []:ENTER
Using configuration from C:\CKyosei\Herramientas\Apache2.2\conf\openssl.cnf
Loading 'screen' into random state - done
Enter pass phrase for ./demoCA/private/cakey.pem:CLAVE
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number:
        ff:58:53:92:0b:a8:87:49
    Validity
        Not Before: Nov 19 23:56:47 2008 GMT
        Not After : Nov 19 23:56:47 2011 GMT
    Subject:
        countryName          = ES
        stateOrProvinceName = MADRID
```



```

organizationName      = CKYOSEI
commonName           = ckyosei.org
emailAddress         = info@ckyosei.org
x509v3 extensions:
    X509v3 Subject Key Identifier:
        37:5B:3F:A7:56:15:5D:1F:82:57:FD:A9:E4:84:45:68:3E:D9:16:45
    X509v3 Authority Key Identifier:
        keyid:37:5B:3F:A7:56:15:5D:1F:82:57:FD:A9:E4:84:45:68:3E:D9:16:45
DirName:/C=ES/ST=MADRID/O=CKYOSEI/CN=ckyosei.org/emailAddress=info@ckyosei.org
        serial:FF:58:53:92:0B:A8:87:49

    X509v3 Basic Constraints:
        CA:TRUE
Certificate is to be certified until Nov 19 23:56:47 2011 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated

```

La contraseña que pongamos en la creación de la CA es de vital importancia, ya que la utilizaremos siempre que queramos firmar un certificado. Sin ésta, *OpenSSL* dará error y abortará la operación. En este caso vamos a poner de clave **CLAVE**.

Al ejecutar el comando anterior se creará por defecto un directorio **demoCA** dentro del directorio anterior según indica la configuración por defecto del archivo **openssl.cnf**

```

“
[ CA_default ]
dir  = ./demoCA
# Where everything is kept
Certs = $dir/certs
# Where the issued certs are kept
crl_dir= $dir/crl
# Where the issued crl are kept
database      = $dir/index.txt
# database index file.
new_certs_dir= $dir/newcerts
# default place for new certs.
.....

```

Tabla 20 Configuración por defecto OpenSSL.conf

La ejecución del comando también creará un certificado así como una clave privada que se usará para firmar los certificados que se generen posteriormente.

El certificado (parte pública) en `./demoCA/cacert.pem` y la clave privada en `./demoCA/private/cakey.pem`

Transformamos el formato PEM de la clave pública a tipo DER que reconoce Windows y abrimos el certificado observamos que Windows no es capaz de reconocer la CA que acabamos de crear.

```
C:\CKyosei\Seguridad\demoCA>openssl x509 -in cacert.pem -outform DER -out ca.der
```

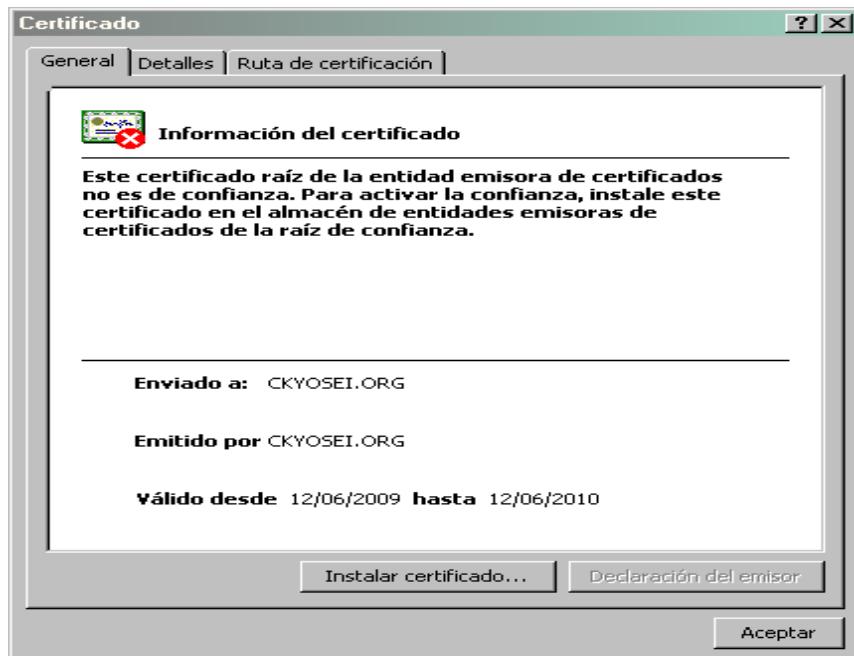
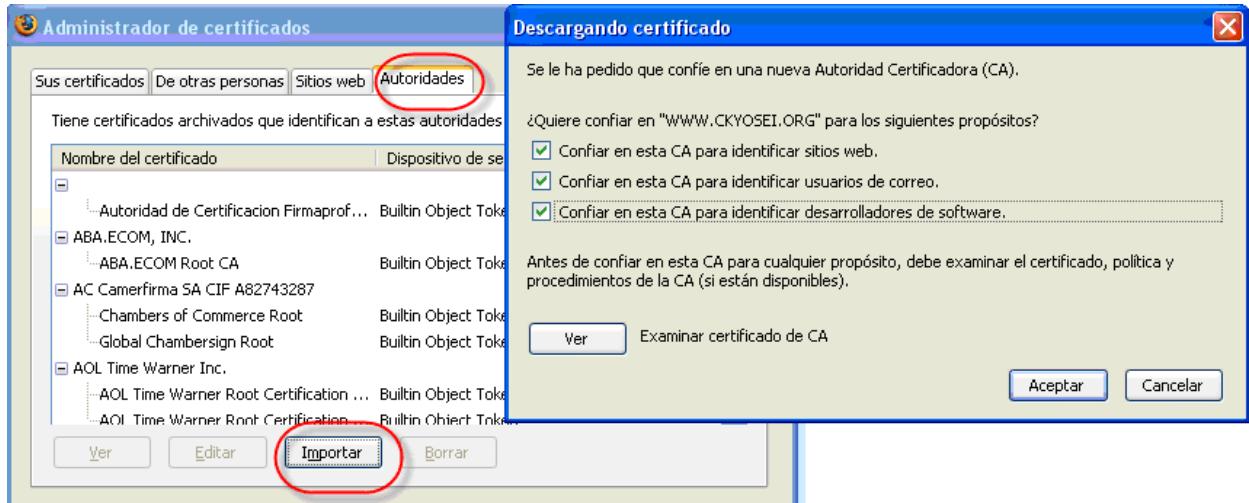


Figura 38 Instalación del certificado de la entidad certificadora en Windows.

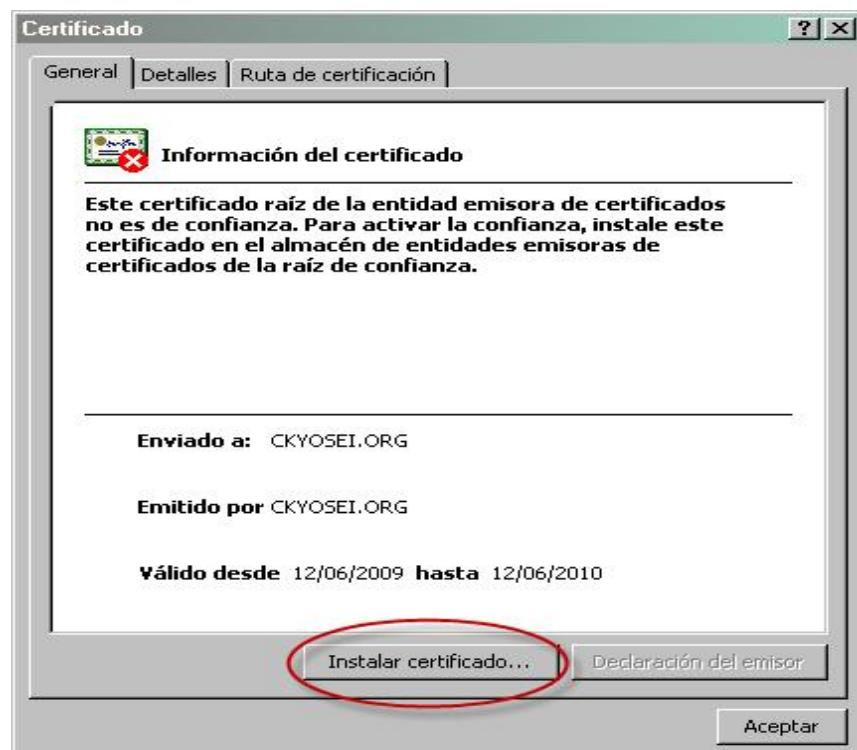
Para solventar este problema importaremos el certificado tanto en los navegadores como una entidad Autorizadora, como en el almacén de certificados de Windows. A continuación se muestra la importación en Firefox. Tendríamos que repetir el mismo procedimiento para Iexplorer u otro navegador.

Para importarla en Firefox, seleccionamos la pestaña de autoridades en el Administrador de certificados:



**Figura 39 Importación de entidad certificadora**

Para instalar la Entidad certificadora en Windows, abriremos el .DER y pulsaremos el botón de instalar certificado.



**Figura 40 Instalación del certificado de la entidad certificadora en windows.**

A partir de este momento tanto Windows como los navegadores reconocerán tanto la entidad Certificadora como los certificados que emitamos con ella.

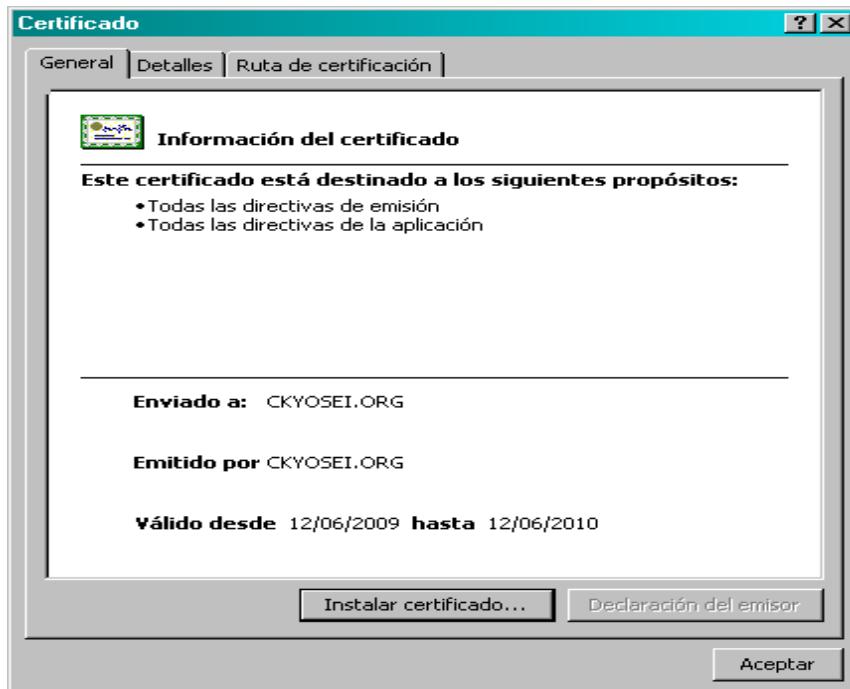


Figura 41 Información del certificado reconocido por Windows

### 6.1.12.3. Creación de Certificados

#### 6.1.12.3.1. Introducción

La generación de certificados con *OpenSSL* se puede realizar de varias formas, usando directamente el comando `openssl` o usando los scripts de Perl que *OpenSSL* nos proporciona para realizar esta misión.

Primeramente lo vamos a realizar usando el script y posteriormente por medio de comandos. Vamos a crear dos certificados diferentes: uno para el servidor Apache, es decir, un certificado de servidor, y un certificado para un cliente de la futura plataforma Kyosei-Polis.

El certificado de servidor lo crearemos usando los comandos de Perl del script y el certificado personal lo crearemos usando comandos de *OpenSSL* directamente.



### 6.1.12.3.2. Creación de un Certificado de Servidor

1. Nos situamos en el directorio C:\CKyosei\Seguridad
2. Generación de una Solicitud de Firma de Certificado (CERTIFICATE SIGNING REQUEST)

```
C:\CKyosei\Seguridad>ca.pl -newreq
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'newkey.pem'
Enter PEM pass phrase:server
Verifying - Enter PEM pass phrase:server

-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:MADRID
Locality Name (eg, city) []:ALCALA DE HENARES
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CKYOSEI
Organizational Unit Name (eg, section) []:UAH
Common Name (eg, YOUR name) []:alpha.com.localhost
Email Address []:info@ckyosei.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Request is in newreq.pem, private key is in newkey.pem
```

Mediante este comando generamos una petición para enviar a una Autoridad de Certificación y solicitar que firme nuestra clave y nos devuelva un certificado.

La información que vamos a introducir en los campos del certificado va a ser similar a la que pusimos para la entidad certificadora, ya que va a ser un certificado de servidor para la misma plataforma, la diferencia va a estar en el CN (common name), que será el del dominio nuevo que se cree para el portal. En este caso pondremos **alpha.com.localhost** puesto que es de prueba. También diferirá la contraseña, que será distinta de la de la entidad certificadora.

Este comando generará unos archivos **newreq.pem** y **newkey.pem** que incluirán la clave privada y la petición de certificado.

3. A continuación vamos a proceder a firmar el certificado como la entidad certificadora que creamos en el punto anterior para [www.ckyosei.org](http://www.ckyosei.org). El comando nos solicitará la clave privada de la entidad certificadora.

```
C:\CKyosei\Seguridad>ca.pl -sign
Using configuration from C:\CKyosei\Herramientas\Apache2.2\conf.cnf
Loading 'screen' into random state - done
Enter pass phrase for ./demoCA/private/cakey.pem:CLAVE
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number:
        e4:ee:33:59:4e:73:78:39
    Validity
        Not Before: Jan 27 19:34:01 2008 GMT
        Not After : Jan 26 19:34:01 2009 GMT
    Subject:
        countryName          = ES
        stateOrProvinceName = MADRID
        localityName         = ALCALA DE HENARES
        organizationName     = CKYOSEI
        organizationalUnitName= UAH
        commonName            = alpha.com.localhost
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        B3:5F:16:EC:FB:A7:32:DB:B8:DE:3B:59:7D:A9:30:1F:EC:2C:E7:12
    X509v3 Authority Key Identifier:
        keyid:C9:5E:A5:61:A4:28:59:F3:26:49:29:28:E3:49:3C:E8:78:8D:ED:AD
        DirName:/C=ES/ST=MADRID/L=ALCALA DE HENARES/O=CKYOSEI/OU=UAH/CN=
WWW.CKYOSEI.ORG/emailAddress=INFO@CKYOSEI.ORG
        serial:E4:EE:33:59:4E:73:78:38

Certificate is to be certified until Jan 26 19:34:01 2009 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Signed certificate is in newcert.pem
```

Como se puede observar en la salida el nuevo certificado firmado quedará en el fichero **newcert.pem**. La ejecución del script nos muestra 2 preguntas: Si queremos firmar el certificado y si actualizamos la base de datos de certificados firmados. La entidad certificadora (entidad de confianza) deberá guardar todos los certificados que firma, ya que tiene que dar fe de su correcta validez, integridad, etc.

Al realizar la operación de firma se crea dentro del directorio **demoCA** en el fichero index.txt una nueva entrada con la siguiente información:

```
V 090126193401Z  E4EE33594E737839      unknown
/C=ES/ST=MADRID/L=ALCALA
DE
HENARES/O=CKYOSEI/CN=alpha.com.localhost
```



Y dentro del directorio **newcerts** se crea una copia del certificado firmado con el nombre **E4EE33594E737839.pem** que corresponde al número de serie del certificado.

Con este certificado y su clave privada podremos ya introducir SSL en el servidor Apache. Modificando el archivo de configuración **httpd-ssl.conf**, sustituyendo el certificado autofirmado que generamos con el script **CKyosei\_SSL\_script.bat** por el emitido por la CA, CKYOSEI.ORG.

En windows es necesario eliminar el password de la clave. El comando nos pedirá el password con el que lo ciframos para confirmar la operación.. Esto se puede realizar ejecutando el comando

```
openssl dsa -in newkey.pem -out server.key
```

o

```
openssl rsa -in newkey.pem -out server.key
```

Renombramos el archivo **newcert.pem** a **server.crt** y la clave **newkey.pem** a **server.key**. Extracto del fichero **\$APACHE\_HOME\conf\ssl.conf**

```
<Virtualhost localhost:443>
.....
SSLCertificateFile conf/ssl/server.crt
SSLCertificateKeyFile conf/ssl/server.key
.....
```

Figura 42 Fichero de configuración SSL de Apache

### 6.1.12.4. Creación de un Certificado de cliente

Para crear un certificado de cliente podríamos usar directamente el comando que ofrece openssl en su script de perl **ca.pl -newcert** o usar **openssl.exe**. En este caso vamos a usar este último para ver su utilización.

1. Primeramente crearemos una clave privada para nuestro certificado. La clave privada se puede añadir en un fichero con contraseña, con el comando

```
openssl genrsa -des3 > cliente.key 1024
```

o sin cifrar:

```
openssl genrsa > cliente.key 1024
```

```
C:\CKyosei\Seguridad>openssl genrsa > cliente.key 1024
Loading 'screen' into random state - done
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
```

2. Generación de una Solicitud de firma de certificado(CERTIFICATE SIGNING REQUEST)

Ahora generamos una petición para enviar a una Autoridad de Certificación y solicitar que firme nuestra clave y nos devuelva un certificado. En este caso usaremos la entidad certificadora que acabamos de crear en el punto 5.14.2. de la guía de instalación.

```
C:\CKyosei\Seguridad>openssl req -new -key cliente.key > cliente.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:MADRID
Locality Name (eg, city) []:ALCALA DE HENARES
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CKYOSEI
Organizational Unit Name (eg, section) []:PROYECTO ALPHA
Common Name (eg, YOUR name) []:NOMBRE ZAPATERA PILO ROBERTO CARLOS - NIF
09234044J
Email Address []:Rxxxxx@GMAIL.COM

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Como se puede observar en el Common Name: hemos seguido la estructura del certificado de la FMNT, para mantener la compatibilidad con esta.

### 3.Firmar la petición de certificado por la entidad certificadora CKYOSEI.ORG

Para firmar la petición vamos a introducir una serie de OIDs que mantendrán la compatibilidad con los certificados de la FMNT.



La FMNT introduce en el Subject Alternative Names (Nombre alternativo del certificado) del certificado los siguientes Oids con la información de cada certificado cliente.

### **1.3.6.1.4.1.5734.1.1=Nombre**

### 1.3.6.1.4.1.5734.1.2=Apellido1

1.3.6.1.4.1.5734.1.3=Apellido2

### 1.3.6.1.4.1.5734.1.4=Nif

Nosotros vamos a tomar como identificador de nuestra CA CKYOSEI.ORG **9999**.

Lo que implica que los que los OIDs creados tendrán la estructura 1.3.6.1.4.1.9999

Modificamos el archivo de configuración `openssl.conf`, con esta información para certificado cliente para ello vamos a crear una nueva sección, en el archivo que llamaremos cada vez que firmemos el certificado. Esta incluirá la llamada a los datos de `dir_sect`.

6

```
subjectAltName=dirName:dir_sect

[ v3_signCA ]

# Extensions for a typical CA

# PKIX recommendation.

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

# This is what PKIX recommends but some broken software chokes
on critical

# extensions.

#basicConstraints = critical,CA:true

# So we do this instead.

basicConstraints = CA:true

# Key usage: this is typical for a CA certificate. However
since it will

# prevent it being used as an test self-signed certificate it
is best

# left out by default.

# keyUsage = cRLSign, keyCertSign

# Some might want this also
```

```

# nsCertType = sslCA, emailCA

# Include email address in subject alt name: another PKIX
recommendation

subjectAltName=dirName:dir_sect

# Copy issuer details

# issuerAltName=issuer:copy

# DER hex encoding of an extension: beware experts only!

# obj=DER:02:03

# Where 'obj' is a standard or added object

# You can even override a supported extension:

# basicConstraints= critical, DER:30:03:01:01:FF


[dir_sect]

OID.1.3.6.1.4.1.9999.1.1=Roberto Carlos
OID.1.3.6.1.4.1.9999.1.2=Zapatera
OID.1.3.6.1.4.1.9999.1.3=Pilo
OID.1.3.6.1.4.1.9999.1.4=*****

```



**Tabla 21 Configuración openssl.conf con los nuevos OIDs**

```

C:\CKyosei\Seguridad>openssl ca -policy policy_anything -out cliente.crt -
infiles cliente.csr -extensions v3_signCA
Using configuration from D:\Desarrollo\openssl-0.9.8k\openssl.cnf
Loading 'screen' into random state - done
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number:
        ab:d2:47:20:d4:25:11:37
    Validity
        Not Before: Jun 24 10:35:55 2009 GMT
        Not After : Jun 24 10:35:55 2010 GMT
    Subject:
        countryName                = ES
        stateOrProvinceName         = MADRID
        localityName               = ALCALA DE HENARES
        organizationName           = CKYOSEI
        organizationalUnitName     = PROYECTO ALPHA
        commonName                 = NOMBRE ZAPATERA PILO ROBERTO CARLOS -
NIF 09234044J
        emailAddress               = Rxxx@GMAIL.COM

```



```
X509v3 extensions:  
    X509v3 Subject Key Identifier:  
        9D:53:51:63:49:5D:8A:DA:F7:52:90:27:D6:E8:08:04:1B:D4:4B:B2  
    X509v3 Authority Key Identifier:  
  
keyid:41:60:79:25:84:F5:41:DD:B7:D8:B7:5E:F8:54:19:75:B7:F6:E8:A4  
  
DirName:/C=ES/ST=MADRID/O=UAH/OU=CKYOSEI/CN=CKYOSEI.ORG/emailAddress=ADMIN@CKYOSEI.ORG  
    serial:AB:D2:47:20:D4:25:11:20  
X509v3 Basic Constraints:  
    CA:TRUE  
X509v3 Subject Alternative Name:  
    DirName:/1.3.6.1.4.1.5734.1.1=Roberto  
Carlos/1.3.6.1.4.1.5734.1.2=Zapatera/1.3.6.1.4.1.5734.1.3=Pilo/1.3.6.1.4.1.5734.1.4=09234044J  
Certificate is to be certified until Jun 24 10:35:55 2010 GMT (365 days)  
Sign the certificate? [y/n]:y  
1 out of 1 certificate requests certified, commit? [y/n]y  
Write out database with 1 new entries  
Data Base Updated  
Signed CA certificate is in newcert.pem
```

De nuevo hemos actualizado la base de datos de la entidad certificadora con este nuevo certificado.

Si observamos en detalle toda la información que contiene el certificado. Observaremos los OIDs que hemos introducido.

```
openssl x509 -in cliente.crt -noout -text  
  
Certificate:  
  
.....  
    X509v3 Subject Alternative Name:  
        DirName:/1.3.6.1.4.1.9999.1.1=Roberto  
Carlos/1.3.6.1.4.1.9999.1.2=Zapatera/1.3.6.1.4.1.9999.1.3=Pilo/1.3.6.1.4.1.9999.1.4=09234044J  
.....
```

3. Por último pondremos el certificado en un formato que sea aceptado por los navegadores, para poder importarlo en nuestro navegador y así usarlo para la autenticación en el navegador, firmar, etc. Para ello, el certificado deberá llevar también su clave privada.

```
C:\CKyosei\Seguridad>openssl pkcs12 -export -out cacert.p12 -in cliente.crt -inkey cliente.key  
Loading 'screen' into random state - done  
Enter Export Password:claveprivada  
Verifying - Enter Export Password:claveprivada
```

### 6.1.12.5. Añadir el certificado al navegador

Buscaremos la pestaña de certificados dentro de la configuración del navegador y desde allí seleccionaremos la opción para importar certificados. Nos pedirá la contraseña que introdujimos al exportarlo.

En el caso del navegador Firefox es un poco más complejo que para el Internet Explorer de Microsoft, ya que nos pide, antes de incorporar el certificado, la contraseña maestra del su almacén de certificados. Esta contraseña se crea la primera vez que incorporas un certificado.

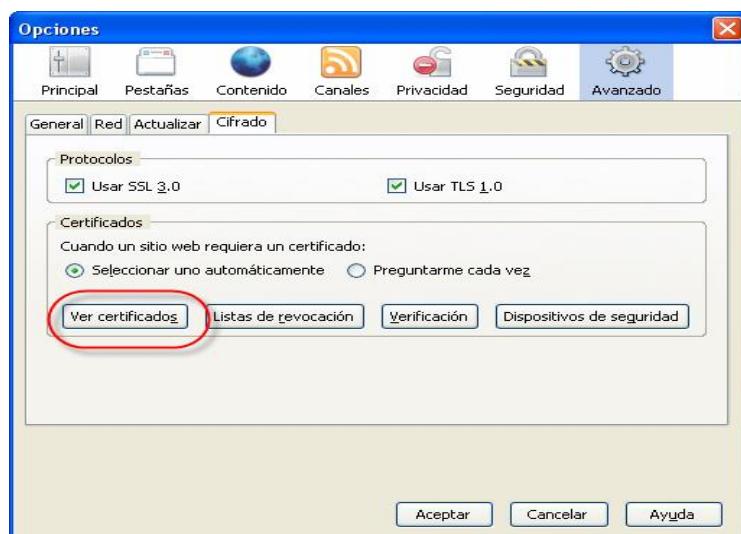


Figura 43 Ver certificados del navegador



Figura 44 Importación de certificados

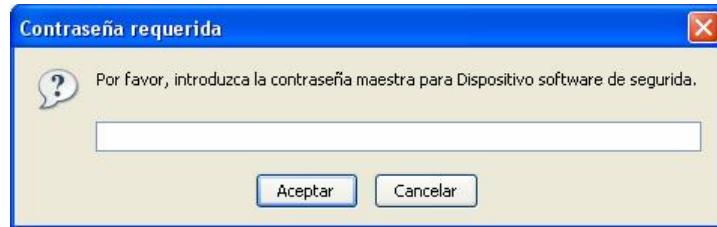


Figura 45 En firefox pide contraseña maestra almacén de certificados

Posteriormente nos pide la contraseña del propio certificado, para importarlo.

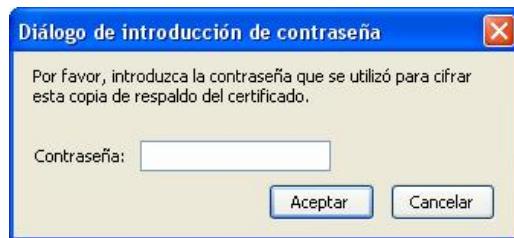


Figura 46 Contraseña de cifrado del fichero de certificados

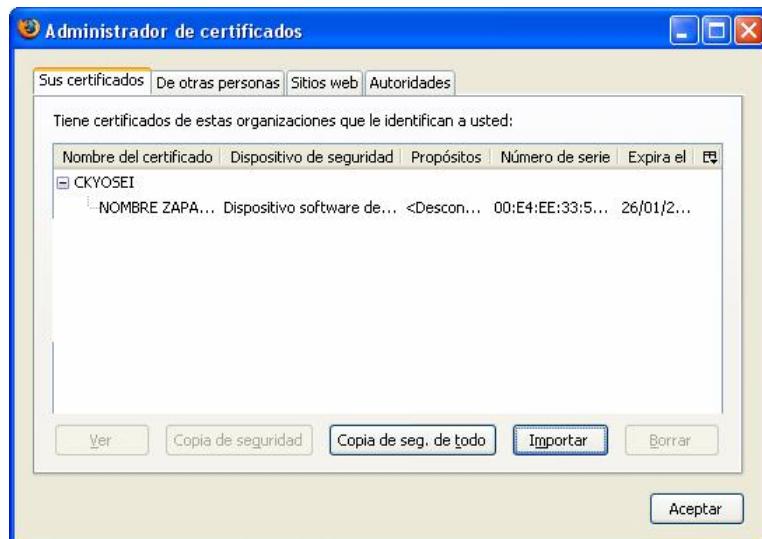
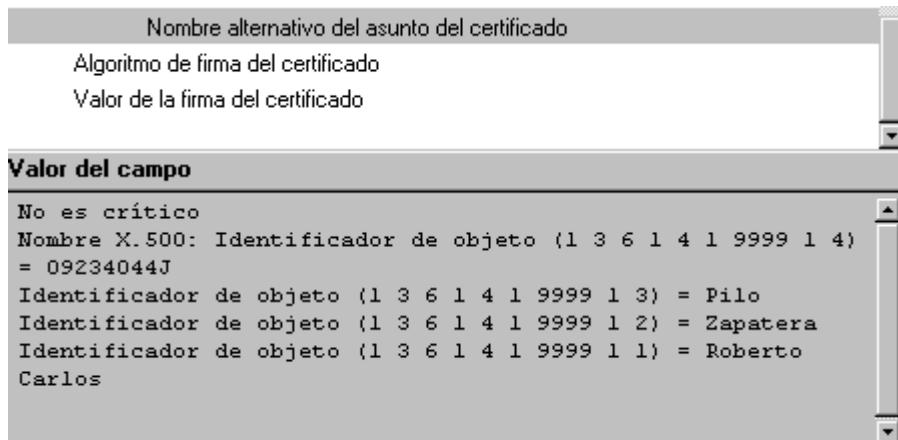


Figura 47 Certificado importado en el navegador



**Figura 48 Información del certificado OIDs Creados.**

## 6.1.12.6. OPCIONAL: Aplicativo de prueba SSL

### 6.1.12.6.1. Introducción

En el punto 6.1.9.1 configuramos la aplicación interconApacheTomcat para trabajar con SSL. En este apartado añadiremos funcionalidad al aplicativo para que lee los datos del certificado digital de un cliente, obtenido a partir del servidor Web Apache, y los muestra por pantalla. El certificado de cliente es transferido a través del modulo mod\_jk a Tomcat para su tratamiento.

La configuración del fichero **httpd-ssl.conf** quedaría:

```

SSLMutex default
SSLRandomSeed startup builtin
SSLSessionCache none
SSLOptions +StdEnvVars +ExportCertData
sslverifyclient require
SSLVerifyDepth 10
SSLCACertificateFile
"C:/CKyosei/Herramientas/Apache2.2/conf/ssl/cacert.crt"
Listen 443 https
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl .crl

```



```
<Virtualhost _default_:443>
    SSLEngine on
    SSLCipherSuite
        ALL:!ADH:!EXP56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
    SSLCertificateFile
        "C:/CKyosei/Herramientas/Apache2.2/conf/ssl/conf/ssl/server.crt"
    SSLCertificateKeyFile
        "C:/CKyosei/Herramientas/Apache2.2/conf/ssl/conf/ssl/server.key"
    ServerAdmin webmaster@alpha.com.localhost
    DocumentRoot "www/alpha.com.localhost"
    ServerName alpha.com.localhost
    ErrorLog "logs/alpha.com.localhost-error.log"
    CustomLog "logs/alpha.com.localhost-access.log" common
    JkMount /interconApacheTomcat/*.jsp ajp13
    JkMount /interconApacheTomcat/*.do ajp13
    <Location "/interconApacheTomcat/WEB-INF/">
        Deny from all
    </Location>
</Virtualhost>
```



Tabla 22 Fichero httpd-ssl.conf

Como se observa hemos añadido que todos los \*.do se redirijan a *Tomcat*.

El aplicativo consta únicamente de un controlador que se encarga de obtener el certificado y usa un patrón de diseño builder para que devuelva la clase de tratamiento de información del certificado, dependiendo del emisor del mismo.

Este aplicativo de momento sólo contempla el uso de certificados digitales cliente expedidos por la FNMT (Fábrica Nacional de Moneda y Timbre), y de la entidad certificadora UAH para CKYOSEI.ORG.

### 6.1.12.6.2. Diagrama de Clases

Como puede verse en la Figura 4, se ha creado una clase abstracta CertificadoDigital con la información que queremos recuperar del certificado. La clase abstracta tiene un método setup que deben sobreescribir todas las clases hijas para conformar la información del certificado digital.

La clase Director es el manager del patrón Builder y nos devuelve la implementación de la clase de tratamiento del certificado según sea éste de la FNMT o de otro tipo.

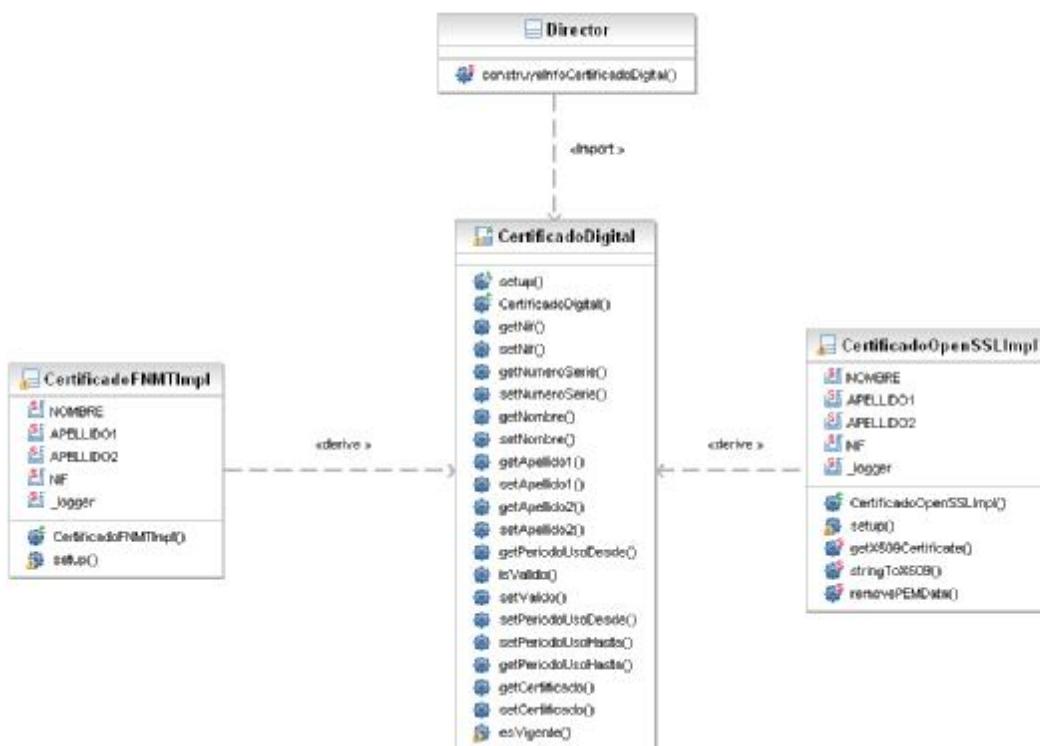


Figura 49 Diagrama de dependencias del paquete org.ckyosei.demossll.ssl

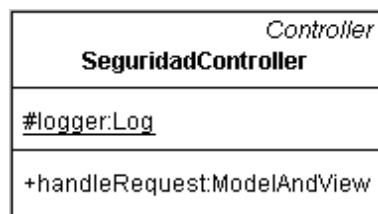


Figura 50 Diagrama Clases paquete org.ckyosei.demossll.web



### 6.1.12.6.3. Diagramas de secuencia

En este punto vamos a mostrar los diagramas de secuencia del método construyeInfoCertificadoDigital y del método del controlador de la aplicación la clase SeguridadController:

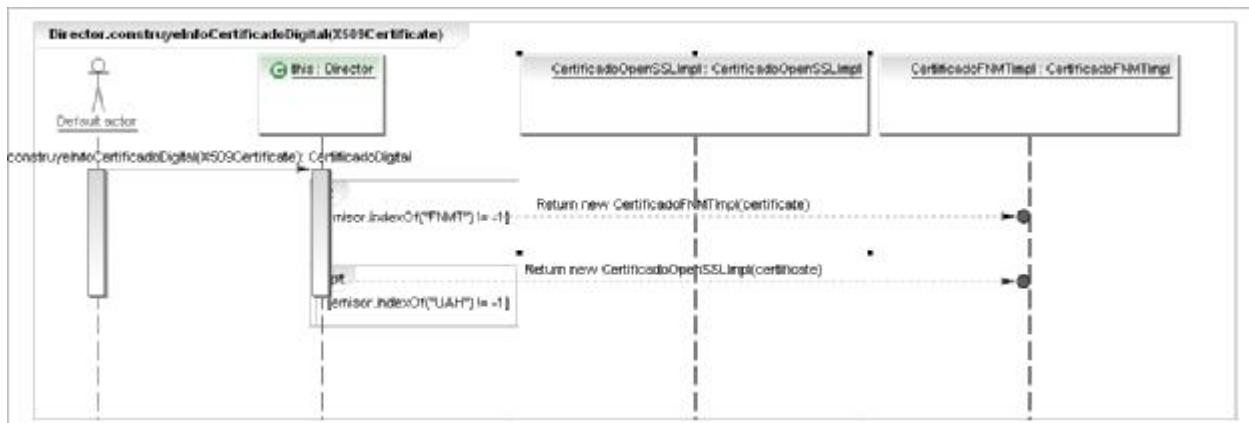


Figura 51 Diagrama de Secuencia del método de negocio de Director

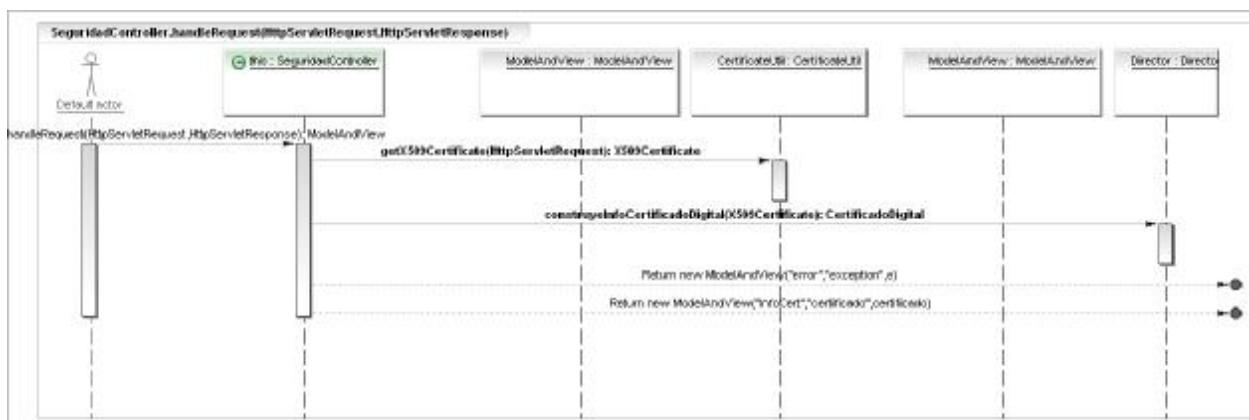


Figura 52 Diagrama de Secuencia del método del Controlador

### 6.1.12.6.4. Funcionamiento del aplicativo desde el Navegador

Puedes desplegar el aplicativo puede descargarlo del repositorio de subversion de ckyosei <http://www.ckyosei.org/svn/repos/ckyosei/interconApacheTomcat/trunk/>.

Cuando realizamos la llamada al aplicativo de seguridad desde nuestro navegador, con <https://alpha.com.localhost/interconApacheTomcat/cert.do>, deberemos seleccionar un certificado digital, cuya información será mostrada.

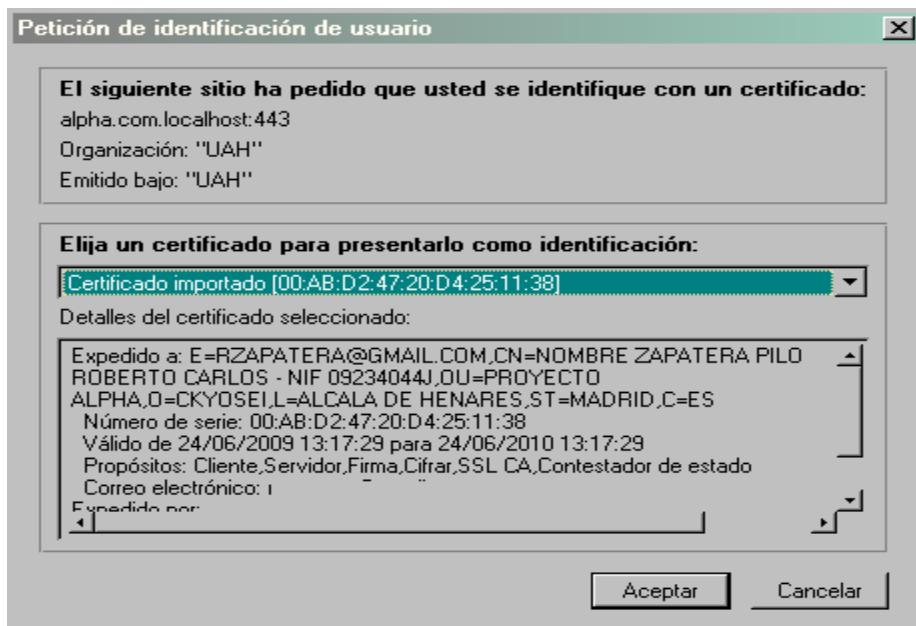


Figura 53 Petición de identificación mediante certificado Cliente del navegador.

Información Del Certificado	
CAMPO	DETALLE
OID-1.2.840.113549.3.2.5	SubjectDN
OID-1.2.840.113549.3.2.5.1	OU=PROYECTO
OID-1.2.840.113549.3.2.5.2	ALCALA DE HENARES,ST=MADRID,C=ES
OID-1.2.840.113549.3.2.5.3	00:AB:D2:47:20:D4:25:11:38
OID-1.2.840.113549.3.2.5.4	24/06/2009 13:17:29
OID-1.2.840.113549.3.2.5.5	24/06/2010 13:17:29
OID-1.2.840.113549.3.2.5.6	Cliente,Servidor,Firma,Cifrar,SSL CA,Contestador de estado
OID-1.2.840.113549.3.2.5.7	r...@gmail.com

Figura 54 Información del certificado autenticado



## 6.1.13. Instalación de Subversion e integración con Apache 2.2.x

### 6.1.13.1. Introducción: Sistemas de control de versiones

Un sistema de control de versiones es un software que administra el acceso a un conjunto de ficheros y mantiene un historial de cambios realizados en ellos. El control de versiones es útil para guardar cualquier documento que cambie con frecuencia, como es el caso del código fuente de un programa.

Normalmente, consiste en una copia maestra, hospedada en un repositorio central, y un programa cliente con el que cada usuario sincroniza su copia local. Esto permite compartir los cambios sobre un mismo conjunto de ficheros. Además, el repositorio guarda un registro de los cambios realizados por cada usuario y permite volver a un estado anterior en caso de necesidad.

Existen multitud de sistemas de control de versiones. El más popular es *CVS (Concurrent Versions System)*. *CVS* tuvo el merito de ser el primer sistema usado por el movimiento de código abierto para que los programadores colaboraran remotamente mediante el envío de parches. Es de uso gratuito, código abierto y emplea fusión de cambios.

*Subversion* se creó para igualar y mejorar la funcionalidad de *CVS*, preservando su filosofía de desarrollo.

### 6.1.13.2. Requerimientos

- § *Subversion para Apache 2.2.x*, versión estable de la serie 1.5.X (versión 1.5.4 en el momento de redacción de este manual), **svn-win32-1.5.4.zip**:  
<http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=8100>
- § *TortoiseSVN*, versión estable correspondiente de la serie 1.5.X (versión 1.5.5 en el momento de redacción de este manual), **TortoiseSVN-1.5.5.14361-win32-svn-1.5.4.msi**:  
[http://sourceforge.net/project/showfiles.php?group\\_id=138498](http://sourceforge.net/project/showfiles.php?group_id=138498)

### 6.1.13.3. Proceso de instalación

La distribución de *Subversion* incluye un cliente remoto (svn), un servidor (svnserver) y varias utilidades. *TortoiseSVN* es un cliente *Subversion*, implementado como una extensión al shell de *Windows*. Es software libre liberado según la licencia GNU GPL.

*TortoiseSVN* permite utilizar *Subversion* directamente desde el explorador de Windows.



Figura 55 TortoiseSVN en el explorador Windows

Antes de comenzar la instalación de *Subversion* es importante considerar que no se deben utilizar espacios en blanco en las rutas que el proceso de instalación nos irá solicitando, ya que *Subversion* no será capaz de gestionarlos.

1. Instalación de *Subversion* Descomprimir el paquete de instalación de *Subversion*, **svn-win32-1.54.zip** en el directorio:

C:\CKyosei\Herramientas\svn-win32-1.5.4

2. Establecer una variable de entorno **SUBVERSION\_HOME** cuyo valor sea el directorio donde haya descomprimido *Subversion*. Añadimos también el subdirectorio **bin** de *Subversion* a la variable **PATH** de *Windows*, agregándole ;%SUBVERSION\_HOME%\bin, para que se encuentren automáticamente sus ejecutables.
3. Instalación del cliente *TortoiseSVN*, ejecutando el paquete de instalación. Aceptamos la licencia, y en la siguiente ventana establecemos para su instalación el directorio C:\CKyosei\Herramientas\ TortoiseSVN\ y continuamos hasta su término la instalación.

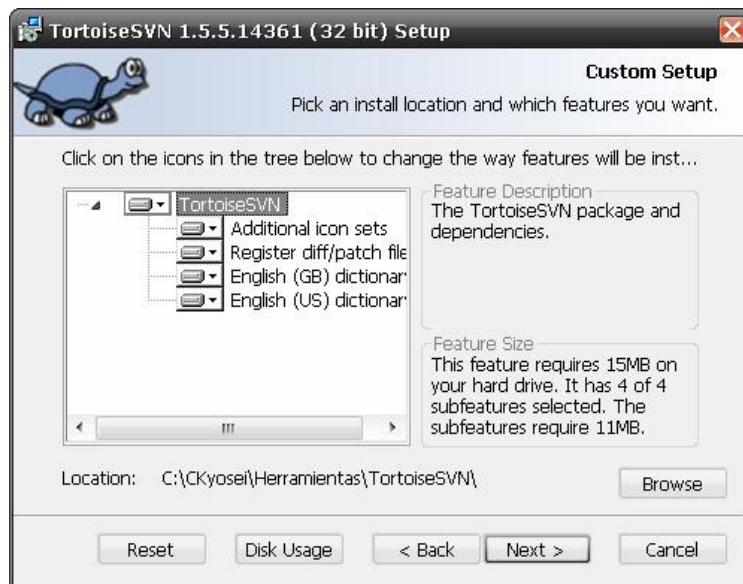


Figura 56 Pantalla instalación TortoiseSVN



### 6.1.13.4. Creación de repositorios

Veamos cómo configurar *Subversion* y *Apache* para hacer accesible el repositorio sobre una red. Se debe tener por lo menos un repositorio, aunque *Subversion* permite la existencia de múltiples repositorios, uno por cada proyecto.

Vamos a crear 3 repositorios:

- § El primero de ellos para el código fuente del aplicativo alpha, así como para el arquetipo y demás módulos del proyecto. Le denominaremos **ckyosei**.
- § El segundo servirá de repositorio de versiones finales de los diferentes productos que se vayan obteniendo. Le denominaremos **Versiones**.
- § El tercero servirá como repositorio de versiones **snapshots**. Versiones sobre las que estamos desarrollando. Le denominaremos **Snapshots**

#### 1. Configuración de los repositorios de *Subversion*

1.1. Creación, dentro del directorio C:\CKyosei\Repositorios de los subdirectorios **Versiónes**, **Snapshot** y **ckyosei**.

1.2. Los repositorios pueden crearse de dos maneras: con el comando **svnadmin** en una, o usando *TortoiseSVN* -situándose sobre cada una de las carpetas en el Explorador de *Windows* y pulsando con el botón derecho del ratón-.

1.2.1. Creación mediante comando svnadmin:

```
svnadmin create C:\CKyosei\Repositorios\Versiónes  
svnadmin create C:\CKyosei\Repositorios\Snapshot  
svnadmin create C:\CKyosei\Repositorios\ckyosei
```

1.2.2. Creación mediante *TortoiseSVN*:



Figura 57 Creación de repositos mediante TortoiseSVN

Si entramos en el interior de un repositorio, lo que vemos son ficheros y directorios de una instancia de la base de datos Berkeley. Esta es una novedad frente al CVS: los proyectos no se guardan en el sistema de ficheros, sino en una base de datos. Esto asegura que cualquier transacción que realicemos posea las propiedades *ACID (Atomic, Consistent, Isolated, Durable)* propias de toda base de datos.

Nota: No debe modificarse manualmente el contenido de los directorios que generara la utilidad anterior.

#### 6.1.13.5. Integración SVN-Apache 2.2.x

Para integrar nuestro servidor *Subversion* con *Apache*, y que así pueda accederse al repositorio utilizando el servidor web, debemos realizar los siguientes cambios en la configuración de *Apache*:

1. Copiar los archivos **mod\_authz\_svn.so** y **mod\_dav\_svn.so** procedentes de la instalación de *Subversion* \$SUBVERSION\_HOME\bin al directorio **modules** de la instalación de *Apache*
2. Copiar todas los archivos .dll del directorio anterior al directorio **bin** de *Apache 2.2.x*
3. Modificar el archivo **httpd.conf**, en \$APACHE\_HOME\conf, descomentar las líneas:

```
“  
LoadModule dav_module modules/mod_dav.so  
LoadModule dav_fs_module modules/mod_dav_fs.so  
”
```

4. Añadir a continuación las siguientes líneas:

```
“  
LoadModule dav_svn_module modules/mod_dav_svn.so  
LoadModule authz_svn_module modules/mod_authz_svn.so  
”
```



5. Añadir al final del fichero las líneas:

```
“
# Subversion connections
Include conf/extra/svnserver.conf
”
```

6. Creación de Permisos para usuarios.

- 6.1. Crear un directorio para los ficheros de configuración de usuarios que tendrán permiso para acceder a *Subversion*, en **C:\CKyosei\Repositorios\ConfiguracionSubversion**.
- 6.2. Con el comando **htpasswd**, que se encuentra en el directorio **bin** de la instalación de *Apache*, crearemos el fichero de usuarios así como el de privilegios que los usuarios tienen sobre los repositorios.

```
htpasswd -cm C:/CKyosei/Repositorios/ConfiguracionSubversion svn-auth-file
admin
New password: admin
Re-type new password: admin
Adding password for user admin
```

Repetiremos la acción para tantas usuarios como deseemos, pero para crear los permisos de éstos sólo tendremos que usar la opción **-m**. Si volviéramos a usar la opción **-cm** sobrescribiríamos el fichero de usuarios, ya que la opción **-c** crea el archivo denominado **svn-auth-file** en el que se almacena la información relativa a los usuarios. La opción **-m** indica que se use el algoritmo MD5 para enmascarar las claves.

- 6.3. Modificar los privilegios (ACL) de los usuarios para darle acceso a los repositorios. Para realizar esta operación crearemos un fichero de nombre **svn-acl**, en el directorio **C:/CKyosei/Repositorios/ConfiguracionSubversion/**, con el contenido siguiente:

“

```
# specify groups here
#
[groups]
team1 = admin
#
# team1 group has a read/write access to ckyosei repository
# all subdirectories
# all others have read access only
#
[ckyosei:/]
@team1 = rw
* = r
[versiones:/]
@team1 = rw
* = r
[snapshots:/]
@team1 = rw
* = r
```

”

**Tabla 23 Fichero configuración permisos de usuarios Subversion svn-acl**

Podremos crear tantos grupos, con tantos usuarios como necesitemos. Asimismo podemos darle acceso desde la raíz o desde cualquier subdirectorio, dependiendo de las necesidades del proyecto.

7. Crear dentro del directorio \$APACHE\_HOME/conf/extra un fichero de configuración para los repositorios de *Subversion*, de nombre **svnserver.conf** y que incluya la siguiente configuración.



```
<Location /svn/ckyosei>
    DAV svn
    SVNPath C:/CKyosei/Repositorios/ckyosei
    AuthType Basic
    AuthName "Subversion ckyosei repository"
    AuthUserFile C:/CKyosei/Repositorios/ConfiguracionSubversion svn-auth-file
    Require valid-user
    AuthzSVNAccessFile C:/CKyosei/Repositorios/ConfiguracionSubversion svn-acl
</Location>

<Location /svn/versiones>
    DAV svn
    SVNPath C:/CKyosei/Repositorios/versiones
    AuthType Basic
    AuthName "Subversion versiones repository"
    AuthUserFile C:/CKyosei/Repositorios/ConfiguracionSubversion svn-auth-file
    Require valid-user
    AuthzSVNAccessFile C:/CKyosei/Repositorios/ConfiguracionSubversion svn-acl
</Location>

<Location /svn/snapshots>
    DAV svn
    SVNPath C:/CKyosei/Repositorios/snapshots
    AuthType Basic
    AuthName "Subversion snapshots repository"
    AuthUserFile C:/CKyosei/Repositorios/ConfiguracionSubversion svn-auth-file
    Require valid-user
    AuthzSVNAccessFile C:/CKyosei/Repositorios/ConfiguracionSubversion svn-acl
</Location>
```



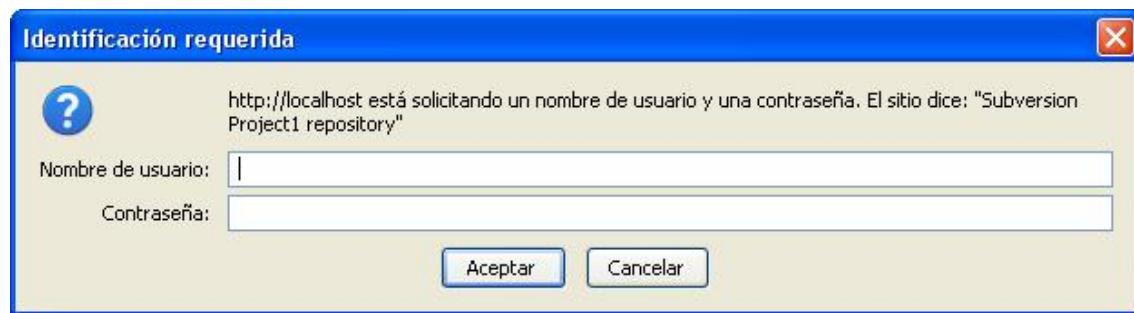
Tabla 24 Configuración de acceso a los Repositorios en Apache

### 8. Reiniciar el servidor Apache.

Se accede a los repositorios utilizando el navegador con la dirección:

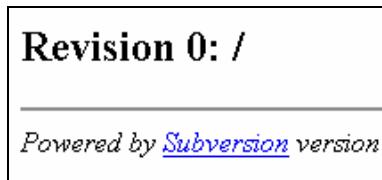
`http://localhost[:puerto]/svn/versiones/`

Nos pedirá el usuario y contraseña que hemos creado anteriormente en el punto 7



**Figura 58 Acceso a un repositorio**

Al introducir el usuario veremos algo similar a la siguiente figura, cuando el repositorio está totalmente vacío.



**Figura 59 Pantalla de bienvenida del repositorio**



## 6.1.14. Instalación de Maven

### 6.1.14.1. Introducción

**Maven** (<http://maven.apache.org>) es una herramienta para la gestión de proyectos de software basada en POM (Project Object Model). **Maven** es un *framework* que nos proporciona un enfoque íntegro de administración de proyectos software. Esta herramienta permite la automatización del ciclo de vida del software.

### 6.1.14.2. Requerimientos

- Paquete de instalación del **Maven**, versión estable de la serie 2.0.X (**apache-maven-2.0.9-bin.zip** *en el momento de redacción de este manual*):  
<http://maven.apache.org/download.html>
- Variable JAVA\_HOME predefinida en el sistema.

### 6.1.14.3. Instalación

1. Descomprimir el zip de **Maven** en C:\CKyosei\Herramientas\.
2. Establezca la variable de entorno MAVEN\_HOME al directorio donde haya instalado **Maven** C:\CKyosei\Herramientas\apache-maven-2.0.9
3. Agregue el directorio bin de **Maven** al PATH del sistema: ;%MAVEN\_HOME%\bin

Comprobar la correcta instalación abriendo una ventana de comandos y ejecutando mvn -version. Se mostrará la versión de la distribución de **Maven** que hayamos descargado e instalado.

### 6.1.14.4. Inicialización repositorio local de Maven

**Maven** almacena las dependencias de los proyectos (las librerías jar) en determinados sitios de forma que queden accesibles para otros proyectos maven. Estos lugares se conocen como repositorios. En Internet hay múltiples repositorios de librerías de **Maven**. Además, **Maven** permite configurar repositorios locales. Un mismo proyecto puede disponer de n repositorios de dependencias.

Cuando ejecutamos el comando `mvn install` por primera vez, **Maven** crea un repositorio local en el directorio `$HOME/.m2/repository` (situado, normalmente, en el directorio `C:\Documents and Settings\<usuario>\.m2`). **Maven** permite que se cambie su localización aunque no es conveniente porque el plugin de eclipse no permite su cambio.

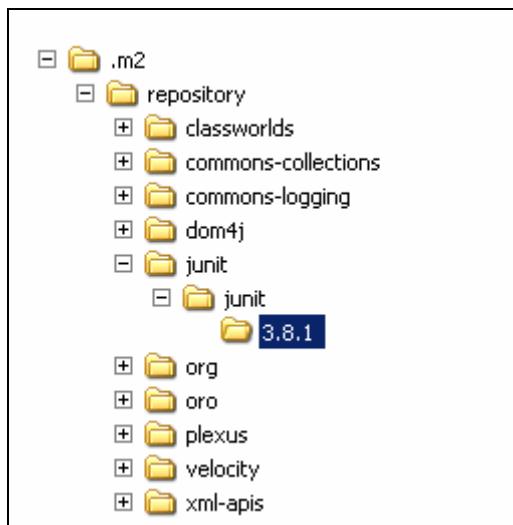


Figura 60 Repositorio local de Maven

Se cargarán, asimismo, los artefactos declarados en el archivo `POM.xml`. Si lo ejecutamos sin ningún archivo `POM`, **Maven** creará el repositorio por defecto pero nos dará un error indicándonos que no se encuentra el archivo de configuración.

#### 6.1.14.5. Creación de un repositorio Maven interno (Integración con Apache)

##### 6.1.14.5.1. Introducción

**Maven** es una herramienta software para la gestión y comprensión de proyectos Java. Dado su enorme potencial, cada día es más normal encontrarse con proyectos que se apoyan en Maven para su desarrollo.

**Maven** utiliza un sistema de repositorios para buscar las dependencias o librerías requeridas para cada proyecto. Debido a que existen librerías de terceros –como es el caso de `jax`, etc.– que por su tipo de licencia no pueden ser compartidas en un repositorio público, a menudo se hace necesario que los proyectos de desarrollo se doten de un repositorio interno de **Maven** en el que hospedarán todas las librerías necesarias para su proyecto, para que queden accesibles para todos los desarrolladores.

**Maven** utiliza, asimismo, la conexión a Internet para descargar las librerías necesarias de repositorios públicos, con lo que podríamos tener problemas si la conexión a Internet es lenta, intermitente, etc.



### 6.1.14.5.2. Configuración de WebDav en Apache 2.2.x Server

El protocolo *WebDAV*, que fue desarrollado por la *Internet Engineering Task Force*, proporciona funcionalidades para crear, cambiar y mover documentos en un servidor remoto (típicamente un servidor web). El término *WebDAV* es un acrónimo para "Edición y versionado distribuidos sobre la Web"

En este punto explicaremos cómo se configura *WebDAV* para compartir librerías de terceros, así como la documentación que se vaya generando de cada proyecto.

Antes de empezar a trabajar con *WebDav* se deben realizar los siguientes cambios:

1. Modificar el archivo **httpd.conf** de Apache:

- 1.1. Añadir los módulos mediante el comando **LoadModule**, des-comentando o añadiendo las siguientes líneas, si es que no estuviesen ya activos:

```
LoadModule dav_module modules/mod_dav.so  
LoadModule dav_fs_module modules/mod_dav_fs.so  
LoadModule setenvif_module modules/mod_setenvif.so  
LoadModule alias_module modules/mod_alias.so  
LoadModule auth_digest_module modules/mod_auth_digest.so  
LoadModule authn_file_module modules/mod_authn_file.so
```

Tabla 25 Fichero configuración Apache inclusión librerias WebDav

1.2. Incluir el fichero de configuración de *WebDav*, des-comentando o añadiendo la línea:

```
“Include conf/extra/httpd-dav.conf”
```

Tabla 26 Inclusión fichero configuración (extra Apache) WebD

2. Crear los directorios que almacenarán las librerías y la documentación en el directorio de repositorios creado previamente:

```
C:\CKyosei\Repositorios\librerias
```

```
C:\CKyosei\Repositorios\librerias
```

3. Crear el directorio `$APACHE_HOME\var`.

4. Editar el fichero de configuración de *WebDav* `$APACHE_HOME\conf\extra\httpd-dav.conf`. Verificamos que existe la línea siguiente:

```
“DavLockDB “C:/CKyosei/Herramientas/Apache2.2/var/DavLock””
```

Y añadimos las siguientes líneas para dar acceso a los directorios que hemos creado:



“

```
Alias /librerias "C:/CKyosei/Repositorios/librerias"
<Directory "C:/CKyosei/Repositorios/librerias">
    Dav On
    Options Indexes
    Order Allow,Deny
    Allow from all
    AuthType Basic
    AuthName "Subversion repository"
    AuthUserFile    "C:/CKyosei/Repositorios/ConfiguracionSubversion/svn-
auth-file"
    Require valid-user
    <LimitExcept GET OPTIONS PROPFIND>
        require user admin
    </LimitExcept>
</Directory>
Alias /documentacion "C:/CKyosei/Repositorios/documentacion"
<Directory "C:/CKyosei/Repositorios/documentacion">
    Dav On
    Options Indexes
    Order Allow,Deny
    Allow from all
    AuthType Basic
    AuthName "Subversion repository"
    AuthUserFile    "C:/CKyosei/Repositorios/ConfiguracionSubversion/svn-
auth-file"
    Require valid-user
    <LimitExcept GET OPTIONS PROPFIND>
        require user admin
    </LimitExcept>
</Directory>
```

”

Tabla 27 Fichero httpd-dav.conf

Como puede verse, utilizaremos para la autentificación el fichero de usuarios de *Subversion*, dado que nos pueste que nos interesa que todos los desarrolladores del *Sistema Kyosei-Polis* puedan tener acceso al recurso compartido de librerías de terceros, así como poder actualizarlo. Alternativamente, podría utilizarse el comando `htdigest -c "$APACHE_HOME/user.passwd" DAV-upload admin`, para crear un archivo de configuración de los permisos de los usuarios (y en tal caso utilizar autorización de tipo Digest, como recomienda Apache).

#### 6.1.14.6. Acceso al repositorio interno

Desde el navegador, accedemos a la dirección:

`http://localhost/librerias`

Nos pedirá el usuario y contraseña (usuarios que configuramos para los repositorios de *Subversión*)

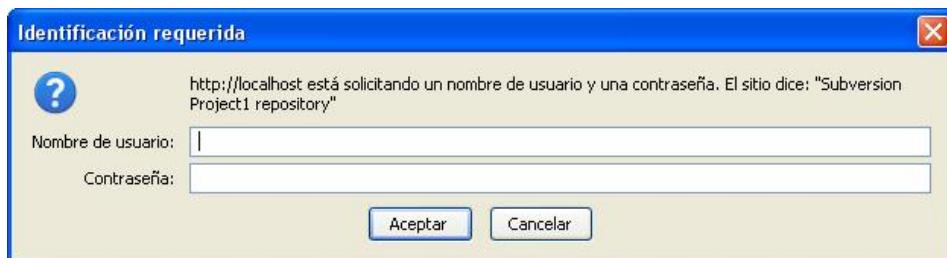


Figura 61 Acceso a repositorio interno

#### Index of /librerias

- [Parent Directory](#)
- [javax/](#)
- [org/](#)

Figura 62 Repositorio interno

De manera similar, podríamos acceder al repositorio de documentación:

`http://localhost/documentacion`



## 6.1.15. Instalación de servidor de correo James Server

### 6.1.15.1. Introducción

James Server es un servidor de correo electrónico SMTP, POP3 y de noticias NNTP. Es un proyecto de la Apache Software Foundation, 100% hecho en Java, siendo un motor de e-mail completo y portable, basado en los actuales protocolos de código abierto.

### 6.1.15.2. Requerimientos

- Paquete de instalación de James (**james-binary-2.3.1.zip** *en el momento de redacción de este manual*):  
<http://james.apache.org/download.cgi>

### 6.1.15.3. Proceso de instalación

1. Descomprimir paquete de instalación en **C:\CKyosei\Herramientas\james-2.3.1**
2. Abrimos una consola MS-DOS y accedemos al directorio Bin.

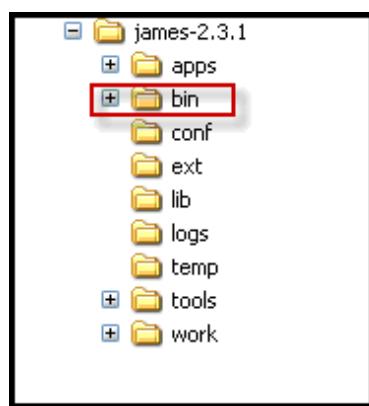


Figura 63 Directorio de instalación de Apache James Server

3. Ejecutamos **run.bat**. Puede ser necesario parar el servicio IIS Admin (lo que detendrá también el servicio SMTP Server asociado), para liberar el puerto estándar del servidor SMTP:

```

C:\CKyosei\Herramientas\james-2.3.1\bin>run.bat
Using PHOENIX_HOME: C:\CKyosei\Herramientas\james-2.3.1
Using PHOENIX_TMPDIR: C:\CKyosei\Herramientas\james-2.3.1\temp
Using JAVA_HOME: C:\CKyosei\SDK\jdk1.6.0_10

Phoenix 4.2

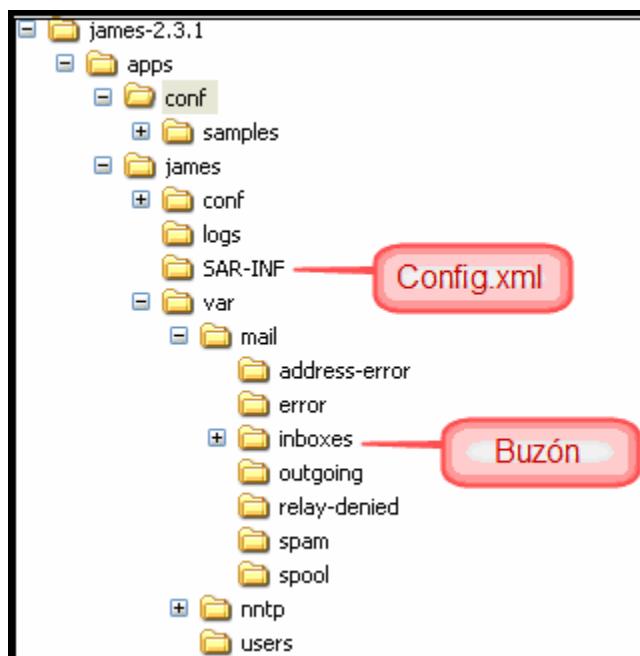
James Mail Server 2.3.1
Remote Manager Service started plain:4555
POP3 Service started plain:110
SMTP Service started plain:25
NNTP Service started plain:119
FetchMail Disabled

```

4. Es necesario iniciar el servidor James una vez para que se extraiga sus directorios y se generen los archivos de configuración. Para detener el servidor, se pulsarían las teclas **Ctrl+c**.

#### 6.1.15.4. Proceso de configuración

1. Con el servidor James ejecutándose, abrimos una sesión telnet al puerto de administración remota con el comando: **telnet localhost 4555**. El usuario inicial es **root** y la contraseña también es **root**.
2. Si ejecutamos **help** veremos la lista de comandos.
3. Para crear cuentas de usuarios se usa el comando **adduser usuario password**. Para cambiar el usuario y la contraseña del RemoteManager hay que modificar el el archivo de configuración general, en **apps/james/SAR-INF/config.xml**. La ubicación por defecto de los buzones de usuarios es: **/apps/james/var/mail/inboxes/usuario**





La ubicación por defecto de los logs es **apps/james/logs**. El dominio por defecto nada más instalar James es **localhost** con lo que el correo de los usuarios sera **uno@localhost**, etc.

4. Para cambiar el dominio tendremos que dirigirnos al directorio **apps\james\SAR-INF** y editar el archivo **config.xml**

```
“
<servernames>
    <servername> domain </servername>
</servernames>
”

```

Tabla 28 Fichero configuración Apache James config.xml

5. Para cambiar los DNS buscamos la entrada en el archivo **config.xml** del punto anterior y los añadimos. En Windows se pueden determinar usando el comando **ipconfig /all** y en Linux mirando el archivo **/etc/resolv.conf**.

```
“
<dnsserver>
    <servers>
        <server>xxx.xxx.xxx.xxx</server>
        <server>xxx.xxx.xxx.xxx</server>
    </servers>
    <authoritative>false</authoritative>
</dnsserver>
”

```

Tabla 29 Fichero configuración DNS resolv.conf.

### 6.1.15.5. Prueba del servidor SMTP

Creamos dos usuarios emisor y receptor:

1. Arrancamos el servidor con **run.bat**
2. Abrimos una consola MS-DOS y creamos dos usuarios emisor y receptor. Para ello usamos la administración remota, abriendo una sesión Telnet: **telnet localhost 4555**

```
JAMES Remote Administration Tool 2.3.1
Please enter your login and password
Login id:
root
Password:
root
Welcome root. HELP for a list of commands
adduser emisor emisor
Unknown command adduser emisor emisor
adduser emisor emisor
User emisor added
adduser receptor receptor
User receptor added
```

3. Abrimos una nueva sesión telnet al puerto 25 (SMTP). Para enviar un mensaje, iniciamos la comunicación escribiendo el comando:

```
220 CKYOSEI01 SMTP Server (JAMES SMTP Server 2.3.1) ready Mon, 19 Jan 2009
13:18:08 -0600(CST)
ehlo localhost
250-CKYOSEI01 Hello localhost (localhost [127.0.0.1])
250-PIPELINING
250 ENHANCEDSTATUSCODES
```

4. Escriba el comando siguiente para indicar al servidor SMTP de destino de quién proviene el mensaje:

```
mail from:<emisor@localhost>
250 2.1.0 Sender <emisor@localhost> OK
```

Esta dirección puede ser cualquier dirección SMTP que desee, pero conviene que sea una cuenta válida, para poder determinar si el mensaje experimenta algún problema de envío, porque el informe de no entrega (NDR) no puede llegar a una dirección IP que no es válida.

5. Escriba el siguiente comando con la dirección SMTP de la persona a la que desea realizar el envío:

```
rcpt to:<receptor@localhost>
250 2.1.5 Recipient <receptor@localhost> OK
```

6. Escriba el comando siguiente para indicar al servidor SMTP que está listo para enviar datos:

```
data
354 Ok Send data ending with <CRLF>.<CRLF>
```



Ahora está listo para comenzar a escribir la sección del mensaje (Las RFC aceptadas son RFC 822, RFC 2822). El usuario verá esta parte del mensaje en su bandeja de entrada.

Escriba el comando siguiente para agregar una línea de asunto:

```
Subject: mensaje de prueba
```

Presione ENTRAR dos veces. No se recibe ninguna respuesta de este comando.

**Nota** Los dos comandos ENTRAR cumplen con la Request for Comments (RFC, petición de comentarios) 822 y 2822. Los comandos 822 deben ir seguidos de una línea en blanco.

A continuación escriba el texto del mensaje.

```
Este es un mensaje de prueba, no verá ninguna respuesta a este comando.  
Finalice el mensaje con punto en una línea independiente y pulse ENTRAR.  
.250 2.6.0 Message received
```

Finalice la sesión con el comando Quit

```
quit
```

Para comprobar la recepción del mensaje:

Entramos al buzón del receptor en **james-2.3.1\apps\james\var\mail\inboxes\receptor**

Vemos que aparecen 2 ficheros. Si abrimos el fichero **.FileStreamStore** veremos el mensaje enviado anteriormente:

MIME-Version: 1.0  
Content-Type: text/plain; charset=Cp1252  
Content-Transfer-Encoding: quoted-printable  
Delivered-To: receptor@localhost  
Received: from localhost ([127.0.0.1])  
by 127.0.0.1 (JAMES SMTP Server 2.3.1) with SMTP ID 747  
for <receptor@localhost>;  
Mon, 19 Jan 2009 13:02:27 +0100 (CET)  
Date: Mon, 19 Jan 2009 13:02:27 +0100 (CET)  
From: emisor@localhost  
Subject: mensaje de prueba  
Este es un mensaje de prueba, no verá ninguna respuesta a este comando.  
Finalice el mensaje con punto en una línea independiente y pulse ENTRAR.



## **6.2. Instalación y Configuración del entorno de desarrollo**

En esta segunda parte, se va a ver la configuración del entorno de desarrollo mínimo que se debería tener instalado para el desarrollo de un proyecto del sistema.

### **6.2.1. Preparación del entorno**

Para mantener todos los programas relacionados con el Entorno de Desarrollo de Kyosei-Polis, conviene que crees un directorio llamado **CKyosei**, y dentro de él los directorios **SDK** (para el Java SDK), **Herramientas**, **WorkSpace** (para el código de la aplicación) y **Repositorios** (para los repositorios Subversion).

Algunas de las descargas del sistema poseen un tamaño considerable. En caso de que encuentres problemas para la descarga de los instalables, te recomendamos utilizar alguna aplicación de gestión de descargas gratuita.

## **6.2.2. Instalación de Java Development Kit**

Ver punto 6.1.4 de la instalación del servidor.



### 6.2.3. Instalación de Maven

En el punto 6.1.14 de la instalación del servidor ya vimos como se realizaba la instalación de Maven. Como se inicializaban los repositorios. Así mismo también vimos como se creaba un repositorio interno para Maven usando el protocolo WebDav para instalar las librerías necesarias para todo el sistema con la finalidad de tenerlas disponibles en todo momento para todos los programadores, y no depender de repositorios públicos en caso de no disponer de conexión a Internet en momentos puntuales.

En este punto vamos a ver como se puede usar esos repositorios desde el punto de vista de un desarrollador de aplicaciones.

#### 6.2.3.1. Instalación de Librerías en el repositorio

Para instalar librerías de terceros en el repositorio interno vamos a usar un plugin de *Maven* que nos permitirá realizar esta operación de manera muy sencilla. Este plugin es *Maven Wagon WebDAV*, y se encarga de crear la estructura de directorios *GroupId*, *artifactId* y *version* dentro del repositorio de librerías. (El plugin viene disponible con la instalación de Maven)

Cuando introducimos una dependencia en el fichero **pom.xml**, *Maven* intentará automáticamente descargarla de un repositorio público, pero si no la encuentra nos dará un mensaje similar a:

La estructura de un repositorio de Maven

```
.m2
+--- repository
    +--- GroupId
        +--- ArtifactId
            +--- version
```

Se podrían usar otros clientes *WebDav*, como *cadaver*, pero teniendo en cuenta que debemos crear los directorios como los solicita *Maven*, es decir: *GroupId*, *artifactId*, *version*, conviene utilizar el plugin propuesto.

Cuando introducimos una dependencia en el fichero **pom.xml**, *Maven* intentará automáticamente descargarla de un repositorio público, pero si no la encuentra nos dará un mensaje similar a:

```
JTA.
[INFO] Failed to resolve artifact.
Missing:
-----
1) javax.transaction:jta:jar:1.0.1B

Try downloading the file manually from:
http://java.sun.com/products/jta
```

En estos casos, para instalar la librería en cuestión, deberemos:

1. Crear un directorio en nuestra máquina local, **C:\CKyosei\librerias3os**, donde descargaremos la librería desde las páginas que nos indique *Maven* al ejecutarse la librería.
2. Crearemos un archivo **pom.xml** en ese directorio con el siguiente contenido.



```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.ckyosei</groupId>
  <artifactId>webdav-deploy-pom</artifactId>
  <packaging>pom</packaging>
  <version>1</version>
  <name>Webdav Deployment POM</name>
  <build>
    <extensions>
      <extension>
        <groupId>org.apache.maven.wagon</groupId>
        <artifactId>wagon-webdav</artifactId>
        <version>1.0-beta-2</version>
      </extension>
    </extensions>
  </build>
</project>
```



Tabla 30 Pom.xml para la importación de librerías de 3os.

3. Editaremos el fichero **\$MAVEN\_HOME\conf\setting.xml** de *Maven* y configuraremos el usuario que fue creado previamente para acceder al repositorio de librerías y al Subversión, en la configuración de los permisos de Subversión (Se recuerda que se han utilizado los mismos usuarios



de la configuración de Subversión para los permisos de acceso a los repositorios WebDav). En la sección <servers> añadimos una nueva configuración de servidor:

```
<servers>
  ...
  <server>
    <id>Ckyosei-repository</id>
    <username>admin</username>
    <password>admin</password>
  </server>
</servers>
```

Tabla 31 Fichero de configuración setting.xml donde configuramos los servidores

4. Ejecutamos el siguiente comando de instalación:

```
mvn deploy:deploy-file -DgroupId=***** -DartifactId=***** -Dversion=** -Dpackaging=jar -Dfile=C:/CKyosei/librerias3os/****.jar -DrepositoryId=Ckyosei-repository -Durl=dav:http://localhost/librerias
```

- DgroupId**: Nombre de los directorios donde almacenaremos las librerías.
- DartifactId**: Nombre de la librería que guardamos.
- Dversion**: Versión de la librería.
- Dfile**: La librería descargada previamente.
- DrepositoryId**: Configuración de servidor del punto anterior.

Al ejecutar el comando anterior la librería es instalada en el repositorio remoto y en el repositorio local que cada usuario tenga configurado en su fichero **setting.xml**.



**Tabla 32 Fragmento del fichero setting.xml de Maven donde se configura el repositorio local**

Ejemplos de algunas de las librerías que tendremos que instalar en el servidor interno serán:

- ü JTA - <http://java.sun.com/products/jta>
- ü JMXRI y JMXTools - <http://java.sun.com/products/JavaManagement/download.html>
- ü JMS <http://java.sun.com/products/jms/docs.html>
- ü etc.

### 6.2.3.2. Creación de *releases* o versiones

En nuestros proyectos en múltiples ocasiones generamos librerías de utilidades que son usadas en más de un proyecto y diferentes versiones del proyecto a medida que va siendo desarrollado. Interesará que estas librerías y proyectos estén disponibles para todos los usuarios, correctamente versionadas.

1. Para poder versionar las librerías y proyectos con *Maven* tendremos también que añadir un **tag** al fichero de configuración **pom.xml** de cada aplicativo, dentro del **tagProject**.



2.

```

<distributionManagement>
    <repository>
        <id>Ckyosei-versiones</id>
        <name>Ckyosei-versiones</name>
        <url>svn:http://localhost:puerto/svn/versiones/</url>
    </repository>
    <site>
        <id> Ckyosei-site</id>
        <url>dav:http://
localhost:puerto/documentacion/${groupId}/${artifactId}/${version}
}</url>
    </site>
    .....
</distributionManagement>

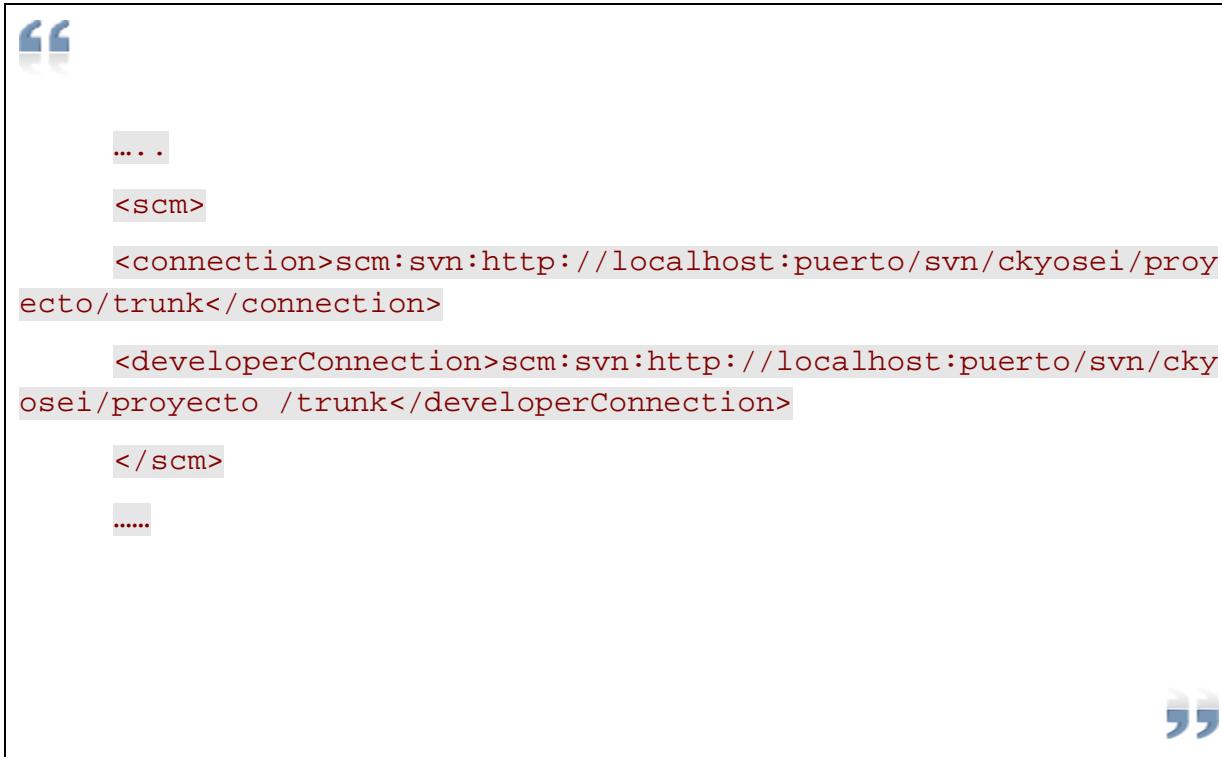
```

Tabla 32 Fragmento de un Fichero Pom.xml donde se configura las distribuciones

Con este *tag* indicaremos dónde queremos que se ubiquen las versiones que obtenemos de nuestros proyectos al generar una nueva versión del mismo (*Release o Deploy*), así como su documentación (*site*)

3. Asimismo, puesto que nuestro proyecto está versionado en un sistema de control de versiones como *Subversion*, añadiremos un *tag*(*scm*) que controle que el código del área de trabajo está correctamente sincronizado con el de *Subversion*

4.



```
“  
.....  
<scm>  
  <connection>scm:svn:http://localhost:puerto/svn/ckyosei/proy  
  ecto/trunk</connection>  
  <developerConnection>scm:svn:http://localhost:puerto/svn/cky  
  osei/proyecto /trunk</developerConnection>  
</scm>  
.....”
```

Tabla 33 Fragmento de un fichero Pomxml donde se configura el SCM

En caso de haber conflictos se recibirá un error indicando que el contenido de la copia local es distinto del versionado (esta operación la realiza el plugin `maven-scm-plugin`; para más información visitar la página: <http://maven.apache.org/scm/plugins/usage.html>).

Como se observa, tanto el servidor como el nombre del proyecto deberán sustituirse por el nombre real utilizado. Ambas conexiones apuntan al hilo *trunk* (tronco) de *Subversion*, en el que se desarrolla el hilo principal de los proyectos.

5. Debemos también añadir, dentro del tag `build` del fichero `pom.xml`, la extensión de *Wagon* para trabajar con el protocolo *svn*, y el plugin que nos permitirá realizar una *release*



```
<build>
.....
<extensions>
    <extension>
        <groupId>org.jvnet.wagon-svn</groupId>
        <artifactId>wagon-svn</artifactId>
        <version>1.8</version>
    </extension>
.....
</extensions>
<plugins>
    <plugin>
        <artifactId>maven-release-plugin</artifactId>
        <configuration>
            <username>${svn.username}</username>
            <password>${svn.password}</password>
        </configuration>
    </plugin>
.....
<plugins>
.....
```

**Tabla 34 Fragmento de un fichero Pomxml donde se configura los plugins**

El usuario y password deberán ser los definidos en el fichero de usuarios de **Subversion**. En cuanto al **tag Tagbase**, indica el directorio donde se almacena la instantánea del proyecto, cada vez que creamos una versión. Una “instantánea” podríamos decir que es el estado del proyecto en el instante de crear la versión.

6. Finalmente, para generar y desplegar en un servidor web la documentación del proyecto (**site**), habremos de modificar la configuración de **Maven** en el fichero **setting.xml**.

```
<servers>
...
<server>
    <id>Ckyosei-site</id>
    <username>admin</username>
    <password>admin</password>
</server>
</servers>
```

7. La creación de una versión o **release** con **Maven** se realiza en 2 pasos:

```
Mvn release:prepare
Mvn release:perform
```

El primero de ellos se encarga de crear el fichero de propiedades de la versión, así como de comprobar la coherencia del área de trabajo con el SCM, y nos pedirá los diferentes datos de la versión; nombre, número de versión, número de la siguiente versión del producto, etc.

### 6.2.3.3. Ejemplo Práctico

#### 6.2.3.3.1. Introducción

En este punto vamos a realizar un ejemplo simple, de creación una aplicación java desde su punto inicial hasta su conclusión, con la finalidad de afianzar todos los conocimientos adquiridos en el documento y resolver las posibles dudas. Para ello nos vamos apoyar en **edipsey** y **Maven**.



### 6.2.3.3.2. Requisitos

- § Instalados el servidor *Apache 2.2.x*, *Maven*, *Subversion Server* y el cliente *TortoiseSVN*.
- § Configurados el acceso a *Subversion* por medio de *Apache*
- § Configurados los repositorios de *Subversion* Versiones, Snapshot y ckyosei (fueron creados previamente en C:/CKyosei/Repositorios/)
- § Configurado *WebDav* para librerías a terceros y documentación

### 6.2.3.3.3. Proceso

1. Creación del proyecto java. Desde la línea comandos nos situamos en el directorio donde esté situada el área de trabajo de eclipse y ejecutamos:

```
mvn archetype:create -DgroupId=org.ckyosei -DartifactId=LifeCycleExample
```

2. Añadir información necesaria para trabajar con el proyecto desde eclipse.

```
cd LifeCycleExample  
mvn eclipse:eclipse
```

3. Desde *eclipse* importamos el proyecto como área de trabajo existente.

4. Modificamos el archivo **pom.xml** como sigue:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"  
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
                           http://maven.apache.org/maven-v4_0_0.xsd">  
  
    <modelVersion>4.0.0</modelVersion>  
  
    <groupId>org.ckyosei</groupId>  
  
    <artifactId>LifeCycleExample</artifactId>  
  
    <packaging>jar</packaging>  
  
    <version>1.0-SNAPSHOT</version>  
  
    <name>LifeCycleExample</name>  
  
    <url>http://maven.apache.org</url>  
  
    <pluginRepositories>
```

```

<pluginRepositories>
    <pluginRepository>
        <id>java.net</id>
        <name>Java.net Repository for Maven2</name>
        <url>http://download.java.net/maven/1/</url>
        <layout>legacy</layout>
    </pluginRepository>
</pluginRepositories>

<!-- 0 - Añadimos tags para indicar dónde está el hilo principal del control de versiones -->
<scm>

<connection>scm:svn:http://localhost:80/svn/ckyosei/LifeCycleExample/trunk</connection>
<developerConnection>scm:svn:http://localhost:80/svn/ckyosei/LifeCycleExample/trunk</developerConnection>
</scm>

<!-- 1 - Añadimos tags para indicar dónde guardaremos las versiones generadas y documentación-->
<distributionManagement>
    <repository>
        <id>Ckyosei-versiones</id>
        <name>iam Repository</name>
        <url>svn:htt://localhost:80/svn/versiones</url>
    </repository>
    <!--snapshotRepository>
        <id>iam-snapshotRepository</id>
        <name>iam Snapshot Repository</name>
        <url>svn:htt://localhost:80/svn/versiones</url>
    </snapshotRepository-->
    <site>
        <id>Ckyosei-site</id>
        <url>dav:htpp://localhost:80/documentacion/\${groupId}/\${artifactId}/\${version}</url>
    </site>
</distributionManagement>
<build>
    <plugins>
        <!-- 2 - Añadimos Plugin para trabajar con el control de versiones del proyecto-->

```



```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-scm-plugin</artifactId>
    <version>1.0-rc1</version>
    <configuration>
        <connectionType>connection</connectionType>
        <username>${svn.username}</username>
        <password>${svn.password}</password>
    </configuration>
</plugin>
<!-- 3 - Añadimos Plugin para realizar versiones del proyecto-->
<plugin>
    <artifactId>maven-release-plugin</artifactId>
    <configuration>
        <username>${svn.username}</username>
        <password>${svn.password}</password>
    </configuration>
</plugin>
</plugins>
</build>
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>
</dependencies>
</project>
```



Tabla 35 Pom.xml totalmente configurado

5. Cerramos *eclipse* y pasamos a crear el proyecto en el repositorio, así como sus carpetas para el hilo principal (*trunk*), instantáneas (*tags*) y versiones distintas del proyecto (*branch*).

```
svn mkdir -m "Making Project dir." http://localhost:80/svn/ckyosei/LifeCycleExample
svn mkdir -m "Dir trunk." http://localhost:80/svn/ckyosei/LifeCycleExample/trunk
svn mkdir -m "Dir branch." http://localhost:80/svn/ckyosei/LifeCycleExample/branch
svn mkdir -m "Dir tags." http://localhost:80/svn/ckyosei/LifeCycleExample/tags
```

6. Importamos el proyecto en Subversion.

- 6.1. Nos situaremos dentro del proyecto y eliminaremos las clases compiladas y recursos del directorio **target**. Importaremos el proyecto mediante el cliente de *Subversion*, ya que el plugin *maven-scm-plugin* no soporta el comando *import*.

```
C:\CKyosei\WorkSpace\LifeCycleExample>mvn clean
C:\CKyosei\WorkSpace\LifeCycleExample>svn import -m"Versión inicial"
http://localhost:80/svn/ckyosei/LifeCycleExample/trunk
```

También podría utilizarse *TortoiseSVN* para realizar la importación.

- 6.2. Después de ejecutar el `svn import` nuestro directorio **LifeCycleExample** no es todavía un directorio de trabajo. Es necesario efectuar un `svn checkout` en alguno de sus directorios. Para realizar esta operación, y dado que *Subversion* no permite realizar una copia o *checkout* del código versionado sobre un directorio con ficheros, haremos una copia de la carpeta **WorkSpace/LifeCycleExample** en un directorio temporal y eliminaremos todo su contenido (Realizar una copia sirve de backup en caso de producirse un error en el repositorio, una vez confirmado que está correcto procedemos a eliminarlo)

- 6.3. Realizamos el *checkout* ejecutando el comando:

```
C:\CKyosei\WorkSpace>mvn scm:checkout -
DconnectionUrl="scm:svn:http://localhost:80/svn/ckyosei/LifeCycleExample/trunk" -
DcheckoutDirectory=~/LifeCycleExample
```

- 6.4. Para realizar manualmente un *commit* de los cambios en el servidor se podría usar el comando:

```
C:\CKyosei\WorkSpace>mvn scm:checkin -
DconnectionUrl="scm:svn:http://localhost:80/svn/ckyosei/LifeCycleExample/trunk" -
Dmessage=<commit_log_here>"
```



## 6. Instalación Maven

7. Cuando ejecutamos el comando de preparación de una *release*, *Maven* nos pregunta el número de versión de la *release*, cómo queremos que se versione en el SCM la instantánea y cuál va a ser el número de la siguiente versión.

```
mvn release:prepare
```

```
[INFO] Checking dependencies and plugins for snapshots ...
What is the release version for "LifeCycleExample"? (org.ckyosei:LifeCycleExample)
1.0: :
What is SCM release tag or label for "LifeCycleExample"?
(org.ckyosei:LifeCycleExample) LifeCycleExample-1.0: :
What is the new development version for "LifeCycleExample"?
(org.ckyosei:LifeCycleExample) 1.1-SNAPSHOT: :
[INFO] Transforming 'LifeCycleExample'...
```

En este caso dejamos por defecto 1.1- SNAPSHOT. Este comando generará un fichero de propiedades con la configuración de la versión (*release.properties*) y automáticamente generará el fichero **pom.xml** a esta nueva versión. También creará una copia de seguridad del fichero **pom.xml** antiguo, por si se quiere dar marcha atrás (**pom.xml.releaseBackup**).

8. Creación de la *Release* con el comando:

```
mvn release:perform
```

Podemos ahora comprobar que la versión se encuentra realmente en el servidor:

```
http://localhost/svn/versiones/
```

Revision : /org/ckyosei/LifeCycleExample/1.0

- [LifeCycleExample-1.0-javadoc.jar](#)
- [LifeCycleExample-1.0-javadoc.jar.md5](#)
- [LifeCycleExample-1.0-javadoc.jar.sha1](#)
- [LifeCycleExample-1.0-sources.jar](#)
- [LifeCycleExample-1.0-sources.jar.md5](#)
- [LifeCycleExample-1.0-sources.jar.sha1](#)
- [LifeCycleExample-1.0.jar](#)
- [LifeCycleExample-1.0.jar.md5](#)
- [LifeCycleExample-1.0.jar.sha1](#)
- [LifeCycleExample-1.0.pom](#)
- [LifeCycleExample-1.0.pom.md5](#)
- [LifeCycleExample-1.0.pom.sha1](#)

Figura 64 Repositorio de versiones con el artefacto generado

<http://localhost/documentacion/>

## Index of /documentacion

- [Parent Directory](#)
- [org.ckyosei/](#)

Figura 65 Repositorio de Documentación

<http://localhost/documentacion/org.ckyosei/LifeCycleExample/1.0/index.html>

## LifeCycleExample

Last Published: 2009-10-10

LifeCycleExample

Project Documentation
Project Information
About
Continuous Integration
Dependencies
Issue Tracking
Mailing Lists
Plugin Management
Project License
Project Plugins
Project Team
Source Repository



### About LifeCycleExample

There is currently no description associated with this project.

Figura 66 Página Web de documentación generada.



## 6.2.4. Instalación de Ant

### 6.2.4.1. Introducción

*Ant* es una herramienta usada para la automatización de tareas. Es usado normalmente durante la fase de compilación y construcción de proyectos. Es similar a *Make* de Linux, pero sin las dependencias del sistema operativo que tiene éste. *Ant* es un proyecto de código abierto de la [Apache Software Foundation](#).

### 6.2.4.2. Requerimientos

- Paquete de instalación del *Ant*, versión estable de la serie 1.7.X ([apache-ant-1.7.1-bin.zip en el momento de redacción de este manual](http://apache-ant-1.7.1-bin.zip)):  
<http://ant.apache.org/bindownload.cgi>

*Para construir y utilizar Ant, es necesario disponer de un JAXP instalado y disponible en el classpath, como Xerces. JAXP es un API Java (definido por Sun Microsystems) que sirve para la manipulación y tratamiento de archivos XML. La distribución binaria de Ant incluye la última versión de Apache Xerces2 XML.*

### 6.2.4.3. Instalación

La distribución binaria de *Ant* consta de la siguiente estructura de directorios:

#### Ant

```
+ --- README, LICENCIA, fetch.xml, otros archivos de texto. // Información básica  
bin + --- // contiene los scripts de lanzamiento  
|  
+ --- lib // contiene librerías de dependencias  
|  
+ --- docs // contiene documentación  
| |  
| + --- images // diversos logotipos de la documentación html  
| |  
| + --- manual // Ant documentación (recomendable leer)  
|  
--- +, etc // contiene xsl utilizados para:
```

// - Crear un mejor informe de la salida XML de diversas tareas.  
// - Migrar sus ficheros de construcción y deshacerse de avisos de información obsoleta  
// ... Y más ;-)

1. Descomprimir el zip de Ant en **C:\CKyosei\Herramientas\**.
2. Establezca la variable de entorno **ANT\_HOME** asociándola al directorio donde haya instalado Ant.
3. Agregue el directorio **bin** de Ant al Path del sistema, añadiendo **;%ANT\_HOME%\bin**



## 6.2.5. Instalación de Eclipse

### 6.2.5.1. Introducción

*Eclipse* es una plataforma de desarrollo de software de Código Abierto, que es usada para desarrollar Entornos Integrados de Desarrollo (*IDE*), como el IDE de Java (Java Development Toolkit JDT).

*Eclipse* fue desarrollado originalmente por IBM como el sucesor de *VisualAge*. Ahora es desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

### 6.2.5.2. Requerimientos

- *Eclipse IDE for Java EE Developers Package*, versión estable de la serie 3.4.x (3.4.1 en el momento de redacción de este manual), **eclipse-jee-ganymede-SR1-win32.zip**:

<http://www.eclipse.org/downloads/packages/>

### 6.2.5.3. Proceso de instalación

1. Descomprimir el fichero **eclipse-jee-ganymede-SR1-win32.zip** en la carpeta **C:\CKyosei\Herramientas\eclipse**, que denominaremos en esta guía como **\$ECLIPSE\_HOME**. El paquete *Eclipse IDE for Java EE Developers* incluye el programa *eclipse SDK 3.4.1*, configurado junto con un paquete de herramientas de diseño Web y de desarrollo JEE. Este conjunto de *plugins* permite generar aplicaciones Web usando el IDE *eclipse*
2. Arrancar *eclipse* desde la línea de comandos usando la opción **-clean**, para que se reinicialice la caché de dependencias y los registros de extensiones de *eclipse*

```
eclipse -clean
```

3. Seleccionar la ruta donde se creará el área de trabajo: **C:\CKyosei\Workspace\**. Un área de trabajo es un contenedor donde *eclipse* guarda los proyectos. Cuando creamos una nueva área de trabajo *eclipse* crea un directorio dentro de la carpeta seleccionada, llamado **/metadata**. Este directorio no debe ser modificado, ya que *eclipse* guarda en él la información sobre los proyectos y su configuración.

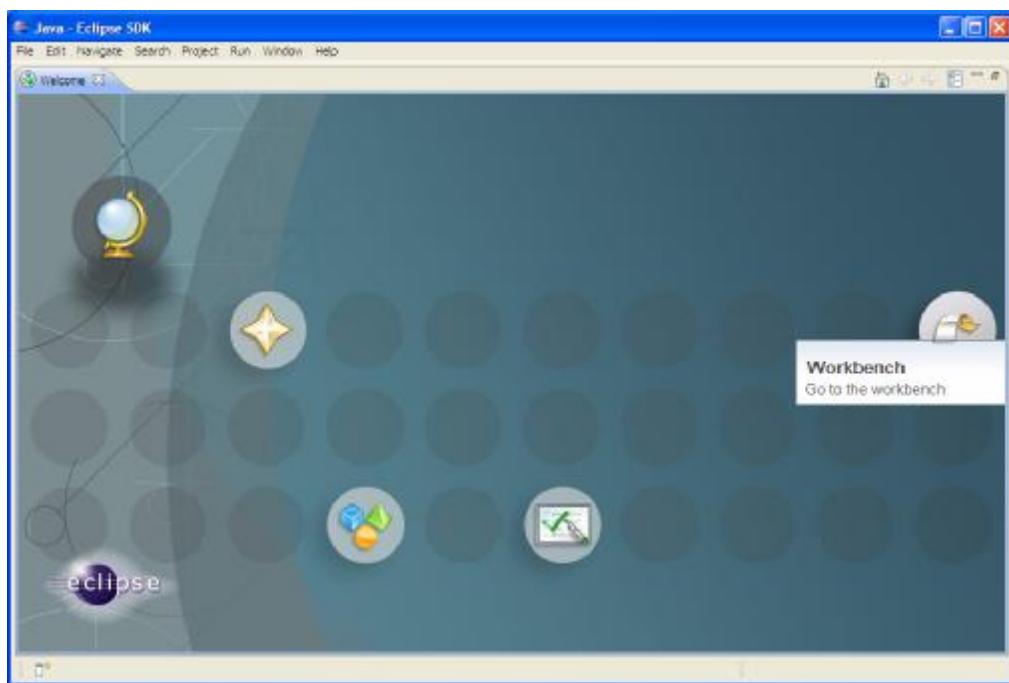
Puesto que no es conveniente tener una única área de trabajo con muchos proyectos, es mejor no establecer una “área de trabajo por defecto”. Es preferible crear tantas áreas de trabajo como necesitemos agrupando proyectos similares o con dependencias comunes en la misma área de trabajo. De esta manera no ralentizaremos el IDE.



**Figura 67 Selección Workspace Eclipse**

#### **6.2.5.4. OPCIONAL: Prueba de eclipse**

1. En la pantalla de bienvenida, ir a **Workbench**, para acceder al entorno de trabajo.



**Figura 68 Bienvenida Eclipse**

2. Para crear un nuevo proyecto Java, verificamos que estamos en la perspectiva Java, en la parte superior derecha de la ventana de eclipse:



## 6. Instalación de Eclipse



Figura 69 Vistas de Eclipse

3. Pulsamos **File - New - Java Project.**

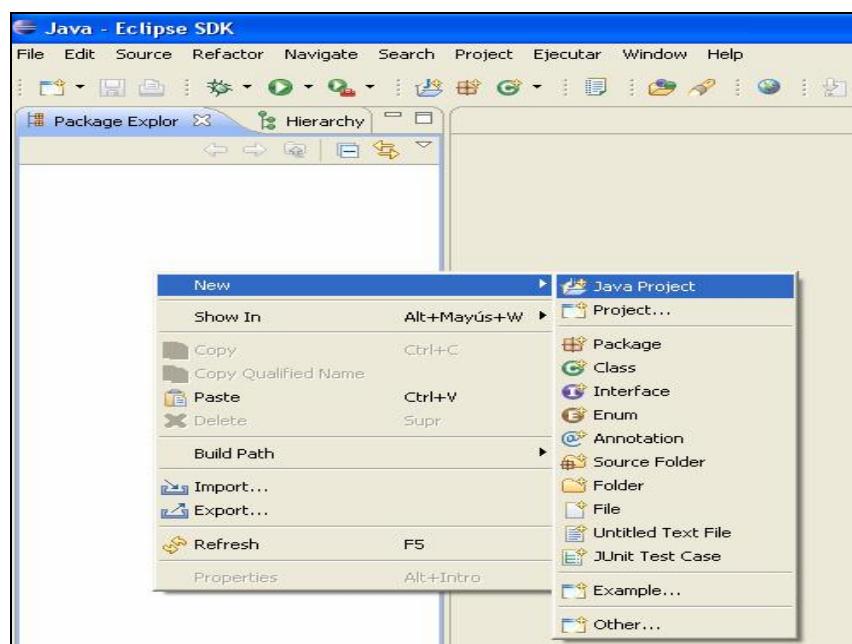
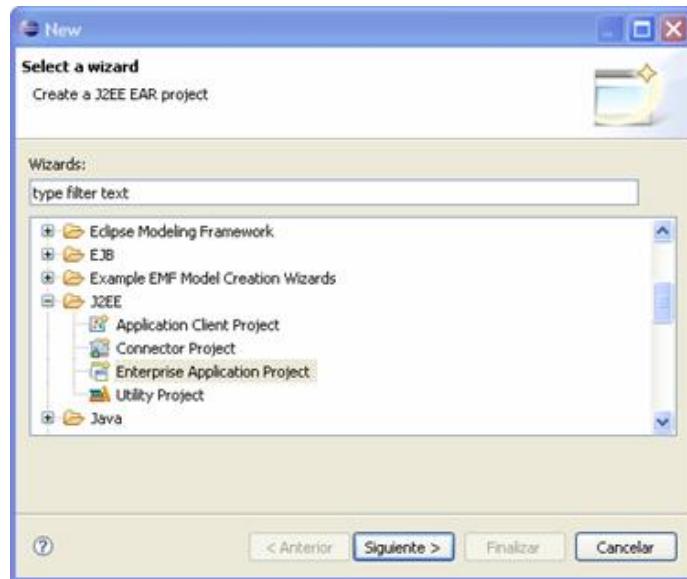


Figura 70 WorkSpace de Eclipse

4. Si deseamos crear una aplicación web, EJB, etc. realizaremos la misma operación que en el punto anterior, seleccionando **File - New - Other....**



**Figura 71 Selección Proyecto**

5. Creacion de aplicación típica de prueba "Hola Mundo".

- 5.1. Seleccionamos **File - New - Java Project** para crear un proyecto Java. Introducimos el nombre del proyecto y pulsamos Next.
- 5.2. Indicaremos el nombre del proyecto Java de Eclipse (en nuestro caso **PrimeraAplicacion**), indicaremos la carpeta donde queramos guardar el proyecto Eclipse (Create new project in workspace o bien Create project from existing source). También podemos seleccionar el Java runtime environment (JRE) que queramos utilizar para el proyecto. Pulsamos Next para continuar con la configuración del proyecto:

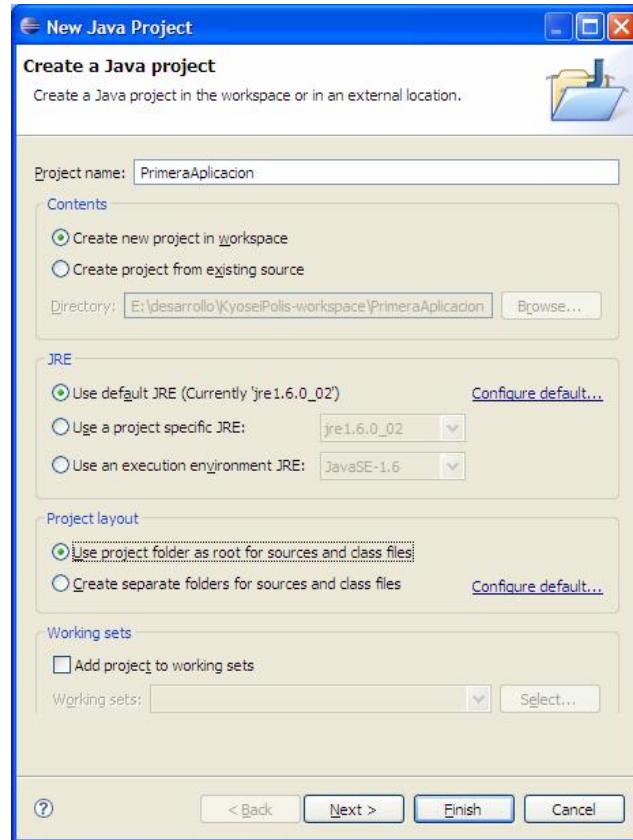


Figura 72 Creación proyecto Java

5.3. Configurar más opciones para el proyecto, como dónde se guardarán los ficheros .class. Se recomienda crear una carpeta como **PrimeraAplicacion/bin** para separar el código fuente de los ficheros compilados. Pulsamos **Finish** para concluir el asistente e iniciar el desarrollo.



**Figura 73 Selección de fuentes**

5.4. A continuación creamos una nueva clase Java para crear el típico mensaje "Hola Mundo". Para ello pulsamos con el botón derecho del ratón sobre **PrimeraAplicación**, en el explorador de paquetes (Package Explorer), seleccionamos **New - Class**:

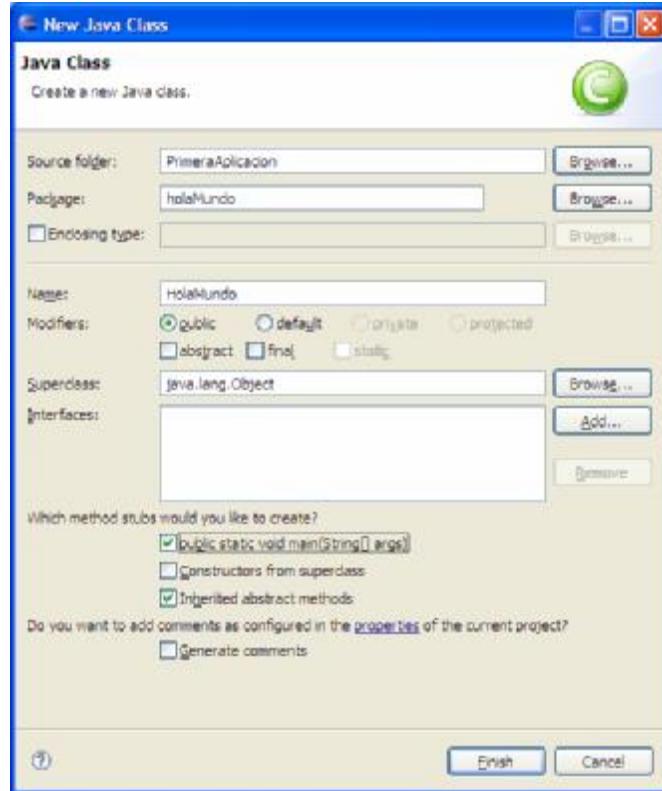


Figura 74 Creación de una Clase

- 5.5. Pulsamos **Finish** y el esqueleto del código para la clase es generado por eclipse. Añadimos la línea de código: `System.out.println("Hola mundo");`

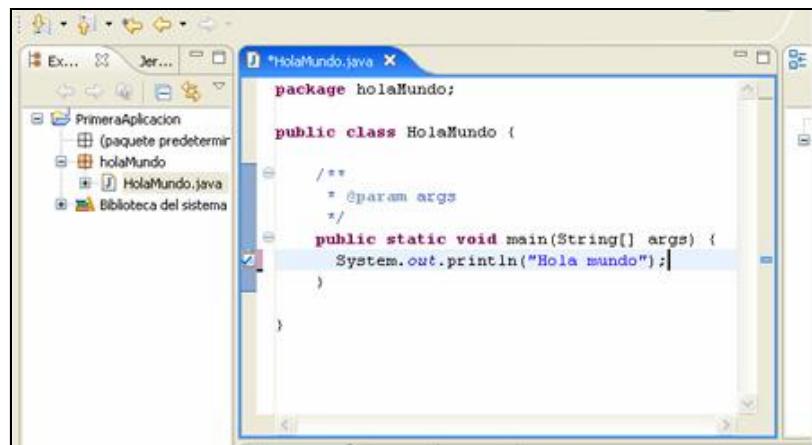


Figura 75 Área de código de Eclipse

- 5.6. Para compilar nuestra aplicación Java Eclipse basta con pulsar el botón de **Save** o **Ctrl.+s**.

- 5.7. Para ejecutar la aplicación pulsar el botón **play**.



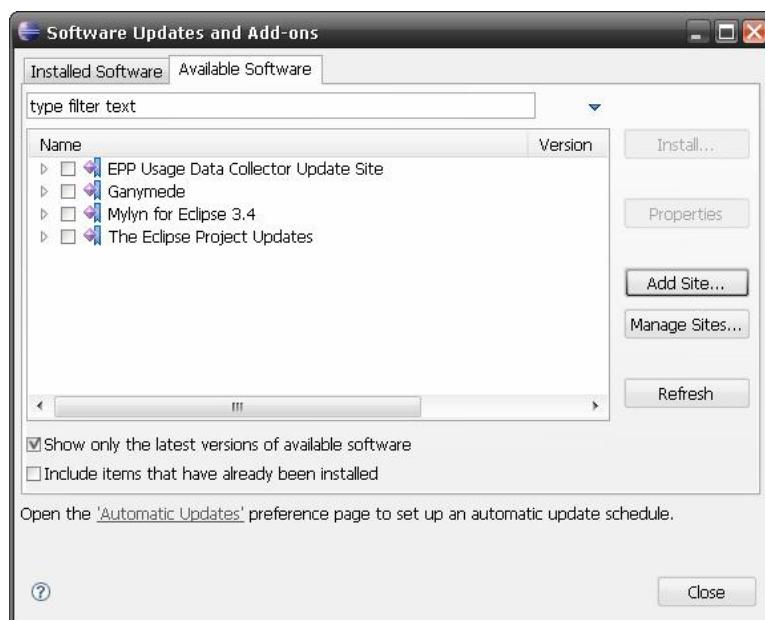
**Figura 76 Ejecución de la aplicación**



## 6.2.6. Instalación de plugins de Eclipse

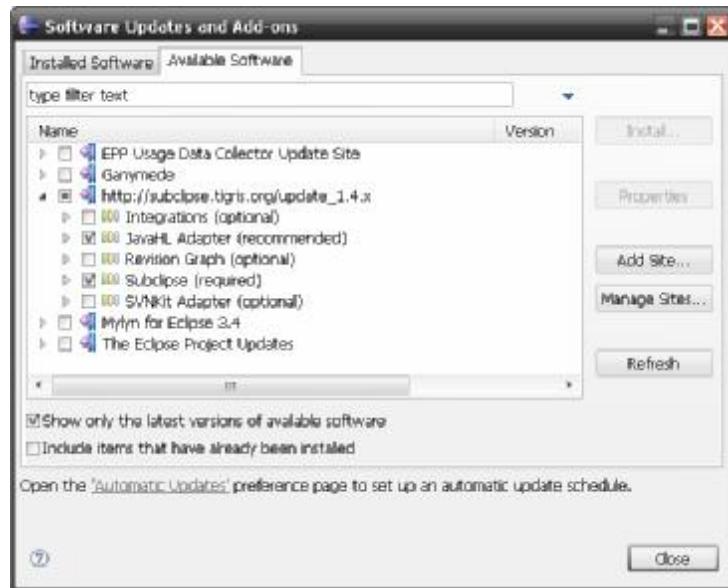
### 6.2.6.1. Instalación de Subclipse plugin en Eclipse 3.4.x

1. Comenzaremos la instalación desde el menú de ayuda de eclipse, seleccionando **Software Updates...**
2. Pulsamos sobre la pestaña **Available Software**, y se deberá seleccionar el botón **Add Site...**.



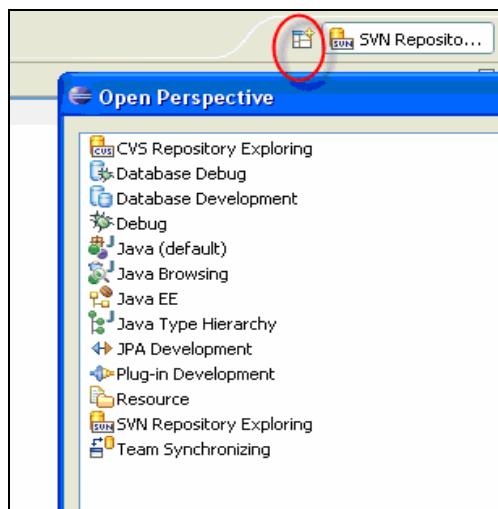
**Figura 77 Actualización Plugins Eclipse**

3. En la ventana que se abre, introducimos el URL de repositorio de actualización del proyecto Subclipse: [http://subclipse.tigris.org/update\\_1.4.x](http://subclipse.tigris.org/update_1.4.x), y pulsamos **Ok**.
4. En la ventana anterior aparecerá el sitio nuevo que hemos añadido con sus componentes. Seleccionamos el plugin **Subclipse** y **JavaHL Adapter** y pulsamos el botón **Install...**



**Figura 78 Selección plugin Subversion**

5. Pulsamos el botón **Next** para confirmar la selección de componentes y en la siguiente ventana aceptamos la licencia y pulsamos el botón **Finish**.
6. Se descargarán de los paquetes de instalación y se instalará el plugin. Al término de la instalación, *eclipse* solicita reiniciarse. Pulsamos **Yes** para reiniciar *eclipse*
7. Seleccionar en las perspectivas **Other... > SVN Repository Exploring**.



**Figura 79 Perspectiva de Eclipse para Subversion**

8. Por último localizar el repositorio.

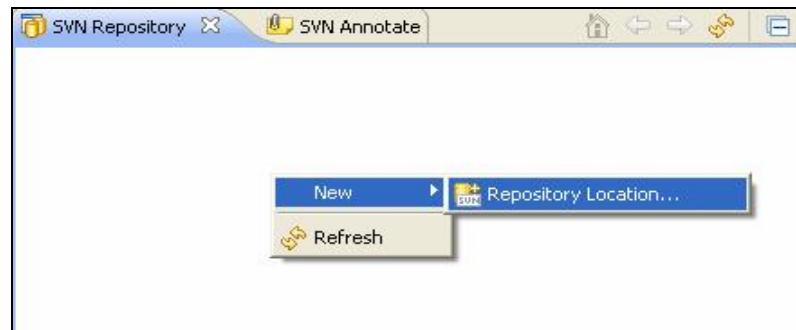


Figura 80 Creación de un nuevo acceso a repositorio

9. Introduce el URL del repositorio del proyecto:

`svn://www.ckyosei.org/svn/repos/ckyosei`

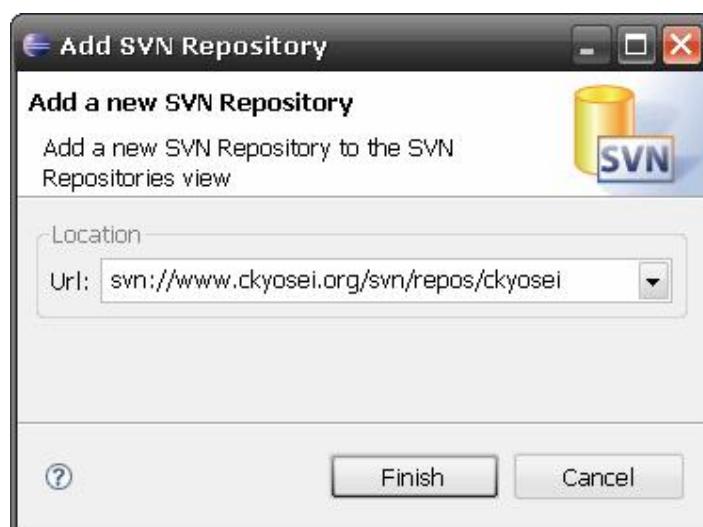


Figura 81 URL acceso a repositorio

10. Ahora puedes descargar las fuentes del proyecto utilizando *Subversion*

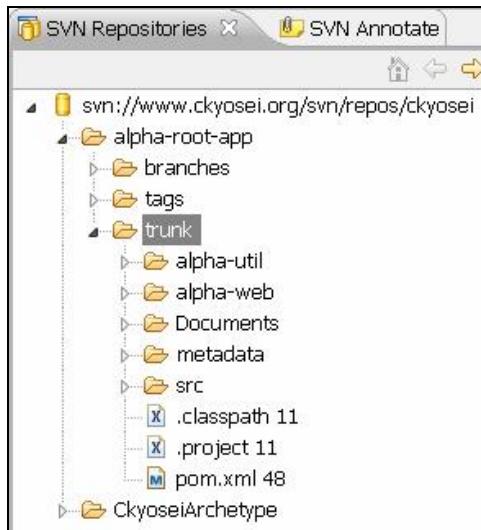


Figura 82 Explorador de Repositorio

## 6.2.6.2. Instalación de SQL Explorer plugin en Eclipse 3.4.x

### 6.2.6.2.1. Introducción

Se trata de un plugin gratuito para Eclipse que permite acceder y manipular bases de datos (Oracle, MySQL, SQL Server, etc.) desde el propio IDE de Eclipse. Eclipse SQL Explorer permite visualizar el contenido y ejecutar consultas contra cualquier base de datos para la que exista un controlador JDBC. Es una herramienta extensible mediante plugins, lo cual permite incorporar funcionalidades específicas para una determinada base de datos. Está disponible como una aplicación independiente de escritorio o como un plugin mas para Eclipse 3.2 y su licencia de uso es LGPL.

### 6.2.6.2.2. Requerimientos

- Paquete de instalación del plugin, versión estable de la serie 3.5.~~x~~ (*0.RC6 en el momento de redacción de este manual*), [sqlexplorer\\_plugin-3.5.0.RC6.zip](http://sourceforge.net/project/showfiles.php?group_id=132863)  
[http://sourceforge.net/project/showfiles.php?group\\_id=132863](http://sourceforge.net/project/showfiles.php?group_id=132863)
- Conector de MySQL, controlador JDBC para utilizar java como cliente de la base de datos, [mysql-connector-java-5.1.~~x~~-rc.zip](http://dev.mysql.com/downloads/connector/j/5.1.html)  
<http://dev.mysql.com/downloads/connector/j/5.1.html>



### 6.2.6.2.3. Proceso de instalación

1. Descargar el paquete de instalación de SQL Explorer y descomprimir en el directorio de instalación de Eclipse.
2. Arrancar eclipse desde la línea de comandos usando la opción `-clean`, para que se regenere la caché de dependencias y los registros de extensiones de eclipse.
3. Durante la instalación de Tomcat ya se descargó el driver conector para la base de datos, y se extrajo en el directorio de librerías de Tomcat `$TOMCAT_HOME\lib`. Verificar que el fichero `mysql-connector-java-5.1.7-bin.jar` se encuentra allá, y si no es así extraerlo.

### 6.2.6.2.4. OPCIONAL: Configuración y prueba del plugin

1. En el asistente de creación `File - New - Other...`, podemos comprobar que nos aparecerán las opciones del nuevo plugin.

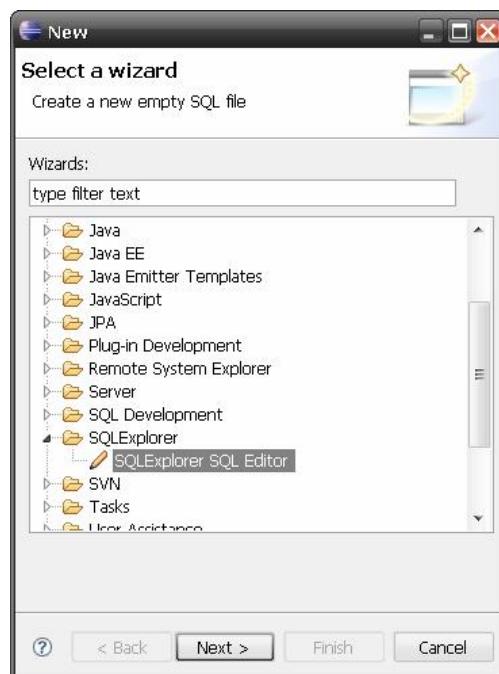


Figura 83 Selección proyecto SQLEditor en eclipse

2. En la ventana `Open Perspective` seleccionamos `SQL Explorer` y pulsaremos `OK`.



Figura 84 Selección perspectiva SQLExplorer en eclipse

- Precisamos indicarle a Eclipse la ubicación del driver JDBC de MySQL. Para ello, pulsamos **Window - Preferences**, y dentro de las preferencias de **SQL Explorer - JDBC Drivers**, seleccionamos el driver **MySQL Driver** y pulsamos **Edit**.

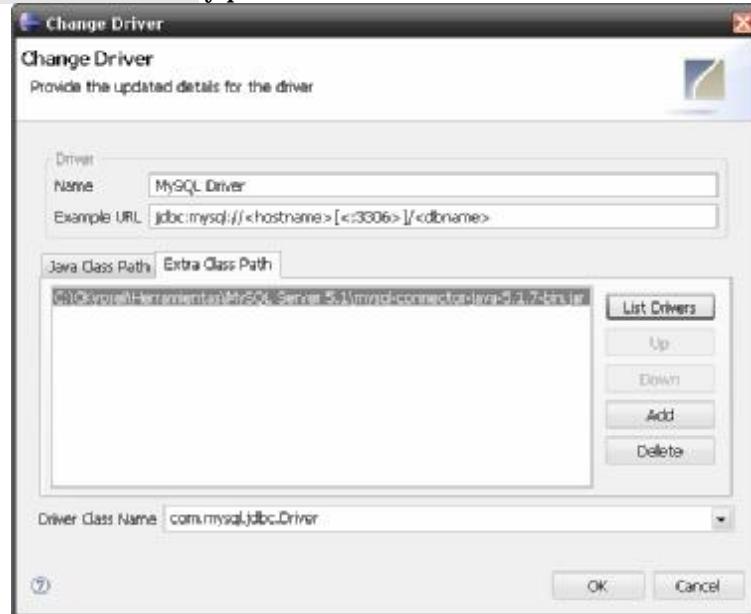


Figura 85 Selección de driver

- En la ventana **Change Driver** en el campo **Name** debe aparecer el valor **MySQL Driver** y en el campo **Example URL** el valor



5. `jdbc:mysql://<hostname>[:<:3306>]/<dbname>`. Pulsamos en la pestaña **Extra Class Path** y posteriormente en el botón **Add**. Seleccionamos el archivo del conector, `mysql-connector-java-5.1.7-bin.jar`, que habíamos previamente situado en el directorio de librerías de Tomcat. Pulsamos el botón **List Drivers** para que aparezcan los drivers en el campo **Driver Class Name**. En este campo seleccionamos el driver `com.mysql.jdbc.Driver` y pulsamos finalmente **Ok**.



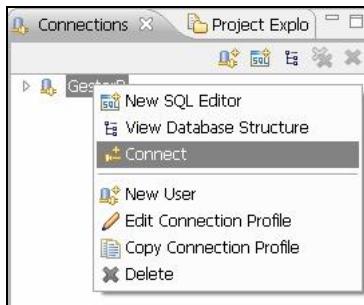
**Figura 86 Configuración de conexión**

6. En la ventana **Create New Connection Profile** introduciremos los siguientes datos:

- **Name:** el nombre del nuevo perfil de conexión, `GestorP`
- **Driver:** en el desplegable aparecen los drivers por defecto, vamos a seleccionar `MySQL Driver`.
- **Url:** Pondremos la URL de conexión a nuestra base de datos, `jdbc:mysql://localhost:3306/Spring-security`
- Seleccionamos la opción **Auto logon**.
- **Username:** `root`
- **Password:** `password`

Y pulsamos **Ok**.

7. Para realizar una conexión pulsaremos con el botón derecho del ratón sobre el perfil de conexión creado `Spring-security` y seleccionaremos **Connect**:



**Figura 87 Establecimiento de conexión**

8. En **Database Structure** podremos ver en forma de árbol todos los esquemas de nuestro servidor MySQL con todas las tablas y sus campos, índices, procedimientos. Al pulsar sobre una tabla, veremos en la pestaña **Database Detail** los detalles de la composición de la tabla, así como la vista previa de todos los registros y campos (**Preview**), número de registros (**Row count**), claves primarias, claves foráneas, índices, privilegios, etc. Por último podemos usar el Editor de SQL para realizar las operaciones oportunas sobre la base de datos.

#### 6.2.6.3. Instalación de Maven plugin en Eclipse 3.4.x

1. De manera similar a con el plugin de Subclipse, comenzaremos la instalación desde el menú de ayuda de eclipse, seleccionando **Software Updates...**
2. Pulsamos sobre la pestaña **Available Software**, y se deberá seleccionar el botón **Add Site**.
3. En la ventana que se abre, introducimos el URL de repositorio de actualización del proyecto Maven: <http://m2eclipse.sonatype.org/update>, y pulsamos **Ok**.
4. En la ventana anterior aparecerá el sitio nuevo que hemos añadido con sus componentes. Seleccionamos el plugin entero y pulsamos el botón **Install...**
5. Aceptamos la licencia y continuamos con la instalación hasta el final. Reiniciamos eclipse.
6. Para usar el plugin de maven en eclipse, es necesario asegurarse de que éste sea ejecutado en un JDK (Java Development Kit), ya que el plugin de maven utiliza jars del JDK. Esto puede conseguirse añadiendo al archivo de configuración de **eclipse.ini**, una línea como:

```
-vm C:\CKyosei\SDK\jdk1.6.0_10\bin\
```



## 6. Instalación Plugins de Eclipse

Alternativamente, se puede modificar el acceso directo que se use para arrancar eclipse para que incluya dicho parámetro:

```
C:\CKyosei\Herramientas\eclipse\eclipse.exe -vm C:\CKyosei\SDK\jdk1.6.0_10\bin
```

7. El plugin de maven requiere, asimismo que los JRE definidos en eclipse correspondan a un JDK. Para ello, verifica que los JRE que aparecen en **Windows – Preferences – Java – Installed JREs**, correspondan a un JDK. Añade, en su caso, la referencia al JDK que hemos instalado:

```
C:\CKyosei\SDK\jdk1.6.0_10\
```

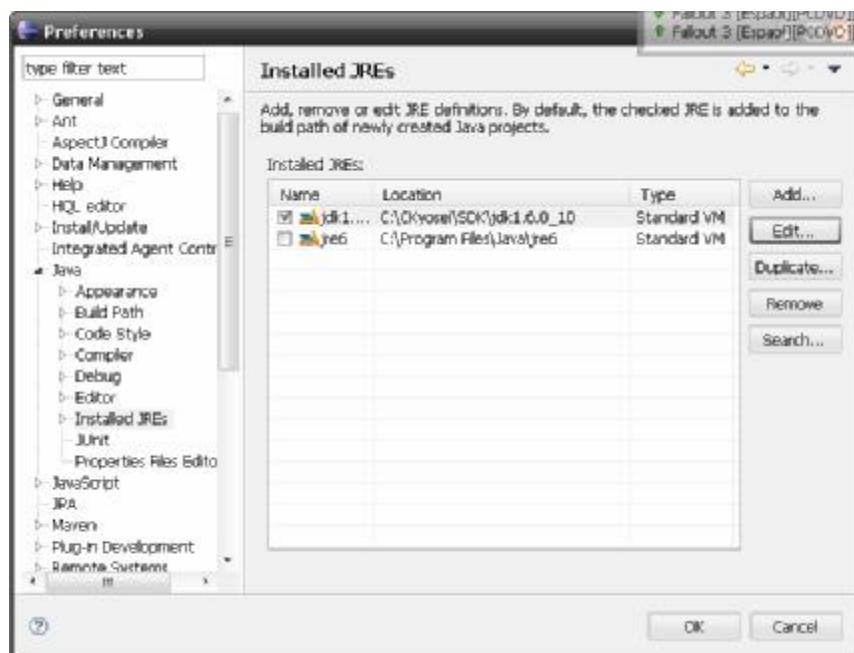


Figura 88 Selección de JRE de eclipse.

### 6.2.6.4. OPCIONAL: Configuración y prueba del plugin

1. Cómo importar un proyecto Maven (externo a eclipse)

- 1.1. El comando de Maven que permite crear un proyecto es:

```
mvn archetype:create -DgroupId=org.ckyosei -DartifactId=CkyoseiAPP
```

- 1.2. Para realizar la importación a **eclipse** vamos al menú **File - Import - General - Maven Projects**. Establecemos en **Root directory** el directorio de Maven donde está el proyecto que acabamos de crear y pulsamos **Finish**.
- 1.3. Alternativamente, **eclipse** permite ya crear directamente un proyecto Maven, desde el menú **File - New - Other... - Maven - Maven Project**. Seleccionamos la opción **Create simple project** y especificamos un Workspace para el proyecto e introducimos, como en el comando maven previo, un **groupId** y un **artifactId**. Con ambas informaciones, el proyecto será creado.

### 6.2.6.5. Configuración de Ant en Eclipse

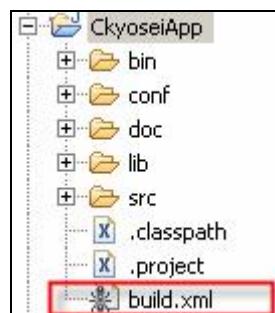
Eclipse incluye un plugin para usar Ant. Pero es recomendable usar la última versión de Ant. Para realizar esta operación tendremos que realizar los siguientes pasos:

1. Abrir el IDE **eclipse** y en el menú superior ir a **Window - Preferences - Ant** y pulsando sobre el botón **Ant Home...** introducimos el directorio donde hemos instalado Ant:

```
C:\CKyosei\Herramientas\apache-ant-1.7.1
```

#### 6.2.6.5.1. OPCIONAL: Utilización de Ant en Eclipse

1. Para crear un fichero build de Ant en Eclipse, en el explorador de paquetes hay que pulsar con el botón derecho sobre el proyecto donde deseemos crearlo y seleccionar **New - Other - General - File**. Como nombre de fichero escribir **build.xml**.



2. Un ejemplo de fichero **build.xml**, para la aplicación de configuración de Spring **CkyoseiApp.jar**. La tarea por defecto del script es **CreateExecutableJar**, que a su vez depende de **clean**, **init**, **compile** y **doc**. Lo que hace la tarea principal es empaquetar la aplicación generando una versión del aplicativo **CkyoseiAPP\_\${version}.jar**, generándose además la documentación de las clases.



```
<?xml version="1.0" encoding="UTF-8"?>
<project name="CkyoseiApp" basedir="." default="CreateExecutableJar">
```



```
<property name="dist.dir" value="dist" />
<property name="src.dir" value="src"/>
<property name="lib.dir" value="lib"/>
<property name="classes.dir" value="bin"/>
<property name="doc.dir" value="doc"/>
<property name="conf.dir" value="conf"/>
<property name="version" value="0.1"/>

<target name="clean">
    <delete includeemptydirs="true">
        <fileset dir="${classes.dir}" includes="**/*"/>
    </delete>
    <delete includeemptydirs="true">
        <fileset dir="${doc.dir}" includes="**/*"/>
    </delete>
    <delete includeemptydirs="true">
        <fileset dir="${dist.dir}" includes="**/*"/>
    </delete>
</target>
<target name="doc" depends="init">
    <javadoc
        destdir="${dist.dir}/${doc.dir}"
        author="true"
        version="0.1"
        use="true"
        windowtitle="Ckyosei API">
        <packageset dir="${src.dir}" defaultexcludes="yes">
            <include name="org/ckyosei/**"/>
        </packageset>
        <doctitle>
            <! [CDATA[<h1>Ckyosei API</h1> ]]></doctitle>
        <bottom>
            <! [CDATA[<i>Ckyosei Application dummy</i> ]]></bottom>
    </javadoc>
</target>
```

```

<path id="master-classpath" description="Master CLASSPATH for this script">
  <fileset dir="${lib.dir}">
    <include name="*.jar" />
  </fileset>
  <pathelement location="${classes.dir}" />
</path>

<target name="init" description="Setup for build script">
  <mkdir dir="${dist.dir}/${doc.dir}" />
  <mkdir dir="${dist.dir}" />
  <mkdir dir="${dist.dir}/lib" />
</target>

<target name="compile" depends="init"
       description="Compiles .java files to WAR directory">
  <javac srcdir="${src.dir}" destdir="${classes.dir}" debug="true"
         failonerror="true" classpathref="master-classpath"
         depend="true" />
  <copy todir="${classes.dir}">
    <fileset dir="${src.dir}">
      <include name="*.xml" />
      <include name="*.properties" />
    </fileset>
  </copy>
</target>

<target name="CreateExecutableJar" depends=" clean ,init,compile,doc">
  <jar destfile="${dist.dir}/CkyoseiAPP_${version}.jar" basedir=".bin"
       manifest=".conf/MANIFEST.MF"/>
  <copy todir="${dist.dir}/lib">
    <fileset dir="${lib.dir}">
      <include name="*.jar" />
    </fileset>
  </copy>
</target>
</project>

```

”

**Tabla 36 Fichero build.xml**

3. Para ejecutar el script de Ant tenemos 2 opciones:



- Pulsar el botón derecho del ratón sobre el fichero **build.xml** del explorador de paquetes y seleccionar **Run As - Ant Build**, con lo que se ejecuta la tarea por defecto que se haya asignado en el script de Ant, mostrándose los resultados en la consola de Eclipse. También puede ver una parte del resultado de la ejecución seleccionando el ícono **Problems** que aparece en la parte inferior de Eclipse.
- Desde la ventana **Outline**, una vez abierto el fichero **build.xml**, pulsar el botón derecho del ratón sobre la tarea y ejecutar **Ant build**.

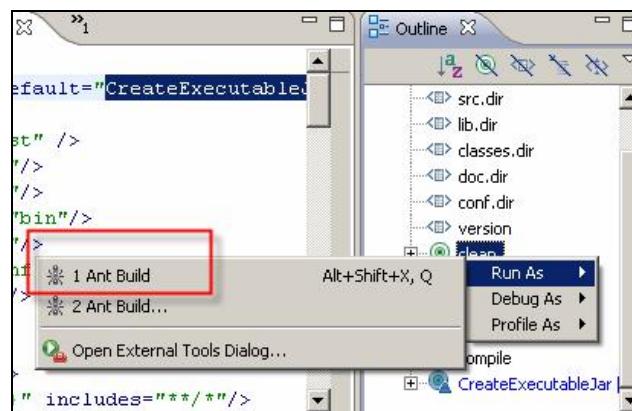


Figura 89 Ejecución de Ant desde eclipse

## **6.3. Instalación y Configuración de Frameworks**

En esta tercera parte, se va a ver la configuración de los frameworks usados en el entorno de desarrollo.



## 6.3.1. Instalación de Hibernate

### 6.3.1.1. Requerimientos

- Paquete de instalación del *Hibernate*, versión estable de la serie 3.2.~~x~~ (*Hibernate-3.2.6.ga.zip en el momento de redacción de este manual*):

[http://sourceforge.net/project/showfiles.php?group\\_id=40712&package\\_id=127784](http://sourceforge.net/project/showfiles.php?group_id=40712&package_id=127784)

### 6.3.1.2. Proceso de instalación

1. Descomprimir el zip de *Hibernate* en C:\CKyosei\Herramientas\.

La librería de *Hibernate* es **hibernate3.jar** que se encuentra en el directorio raíz. Esta librería requiere para su uso una serie de paquetes adicionales que vienen en el interior de la estructura de directorios. La librería de Hibernate requiere para su uso una serie de paquetes adicionales que se encuentran en la carpeta **lib**: **antlr.jar**, **asm.jar**, **asm-attrs.jars**, **cglib.jar**, **commons-collections.jar**, **commons-logging.jar**, **dom4j.jar**, **jta.jar** y **log4j.jar**.

### 6.3.1.3. OPCIONAL: Proceso de inclusión de Hibernate en aplicaciones

1. Vamos a ilustrar cómo sería el proceso de creación de una aplicación java pura y de una aplicación Web, que utilizasen Hibernate.
  2. Para crear una aplicación Java pura (aplicación **main**), debemos ir a **File - New - Project - Java - Java Project**. Introducimos el nombre del proyecto y pulsamos **Next**.
    - 2.1. Para añadir las librerías al **dasopath** del proyecto, en la siguiente ventana pulsamos sobre la pestaña **Libraries** y pulsamos el botón **Add External Jars...**, buscamos el directorio de Hibernate y añadimos la librería **hibernate3.jar** y el resto de librerías de las que ésta depende.
    - 2.2. Una vez añadidas las librerías debemos de crear el fichero de configuración **hibernate.cfg.xml**. En este fichero describe cómo se debe conectar Hibernate a la base de datos y cuáles son los ficheros xml que describen los mapeos entre las clases y las tablas de la base de datos. Este fichero se puede crear a mano o mediante el plugin de Eclipse Hibernate Tools, tal como se explicó en la sección de instalación del plugin de eclipse Hibernate Tools. Recomendamos esta segunda opción. El archivo lo creamos en el directorio **src**, que es donde se sitúan los fuentes de java.

- 2.3. Si exportáramos la aplicación a un archivo **.jar** para poder ejecutar la aplicación sobre una JVM independiente del IDE, deberíamos indicar las dependencias de estas librerías en el archivo **Manifest.MF**.

La estructura de este archivo será similar a la siguiente:

```
“
Manifest-Version: 1.0
Main-Class: com.ckyosei.Principal
Class-Path: .    ./commons-lang.jar
            ./commons-logging.jar
            .....
”
```

**Tabla 37 Fichero Manifest**

**Main-Class** indica la clase que se ejecutará cuando se inicie la aplicación.

**Class-Path** indica las librerías que hemos añadido y que tendrá que cargar para poder ejecutar el aplicativo correctamente.

Para ejecutar la aplicación ejecutaremos el siguiente comando.

```
java -jar nombrepaquete.jar
```

3. Para crear una aplicación Web: **File - New - Other - Web - Dynamic Web Project**. Introducimos el nombre del proyecto y pulsamos **Next**.

- 3.1. Repetimos la inclusión de las librerías, como en el caso anterior, pero esta vez las copiaremos directamente dentro del directorio **WebContent/WEB-INF/lib** del proyecto, ya que al ser una aplicación web, serán buscadas dentro de dicha carpeta al ser desplegada.
- 3.2. Deberemos nuevamente crear el fichero de configuración Hibernate. En este caso para ejecutar la aplicación dentro de un servidor web debemos exportarla a un fichero **.war** y situarla dentro de un contenedor de aplicaciones web. En nuestro caso la situaríamos dentro de **\$STOMCAT\_HOME/webapps**.



### **6.3.1.4. Instalación de Hibernate tools en Eclipse 3.4.x**

#### **6.3.1.4.1. Introducción**

*Hibernate* es una herramienta de mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación. Esta relación se establece mediante archivos de configuración (XML) que permiten establecer estas asociaciones. *Hibernate* es software libre, distribuido bajo los términos de la licencia GNU LGPL.

El objetivo principal de *Hibernate* es facilitar la utilización simultánea de los dos modelos utilizados hoy en día para organizar y manipular datos:

- Ü El usado en la memoria de la computadora y en la programación: orientación a objetos.
- Ü El usado en las bases de datos: modelo relacional.

Para lograr esta meta *Hibernate* permite definir al desarrollador cómo es su modelo de datos, las relaciones existentes, su cardinalidad, etc. Con esta información *Hibernate* le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la *OOP* (Programación Orientada a Objetos).

*Hibernate* está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

*Hibernate* ofrece también un lenguaje de consulta de datos llamado *HQL* (*Hibernate Query Language*), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria").

*Hibernate Tools* (<http://tools.hibernate.org/>) es un conjunto de utilidades para *Ant* y *Eclipse* que nos facilitan el uso de *Hibernate*.

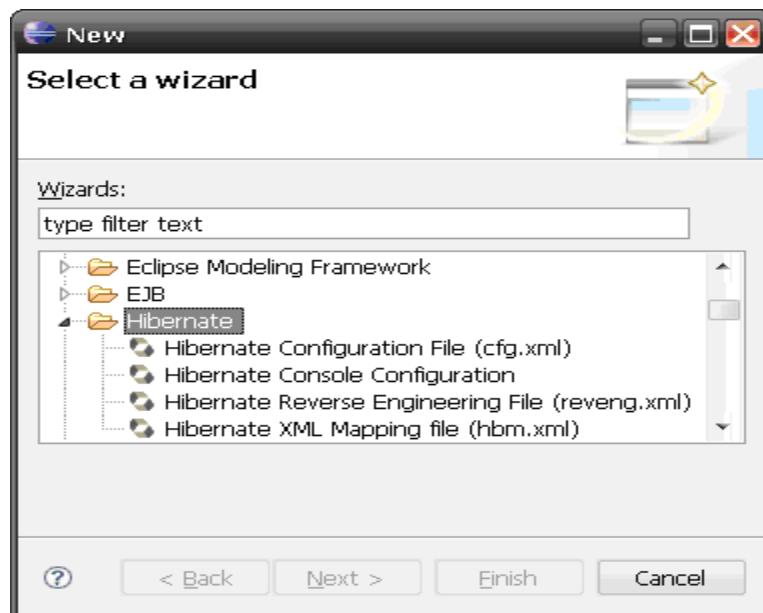
#### **6.3.1.4.2. Requerimientos**

- Paquete de instalación de las *Hibernate Tools*, versión estable de la serie 3.2.X (3.2.4.Beta1 en el momento de redacción de este manual), **HibernateTools-3.2.4.Beta1-R200810311334.zip**:  
<http://www.hibernate.org/30.html>

#### **6.3.1.4.3. Proceso de instalación**

1. Comenzaremos la instalación descargando el paquete de *Hibernate tools*. Una vez descargado descomprimirlo en el directorio donde se instaló Eclipse.

2. Arrancar *eclipse* desde la línea de comandos usando la opción `-clean`, para que se regenere la caché de dependencias y los registros de extensiones de eclipse.
3. En el asistente de creación `File -> New -> Other...`, podemos comprobar que nos aparecerán las nuevas opciones del plugin de *Hibernate*



**Figura 90 Wizar de hibernate**

### 6.3.1.5. OPCIONAL: Configuración y prueba de Hibernate

1. Seleccionando la primera opción, *Hibernate Configuration File (cfg.xml)* el plugin nos permite generar el fichero de configuración normalmente llamado *hibernate.cfg.xml*. En este fichero es donde describiremos cómo *Hibernate* debe conectarse a la base de datos y cuáles son los ficheros xml que describen los mapeos entre las clases y las tablas de la base de datos. Seleccionamos el directorio de ubicación del fichero de configuración, *src*, y pulsamos *Next*.



Figura 91 Selección de directorio para la Configuración de Hibernate

2. En la siguiente ventana, introducimos los valores mostrados en la imagen:

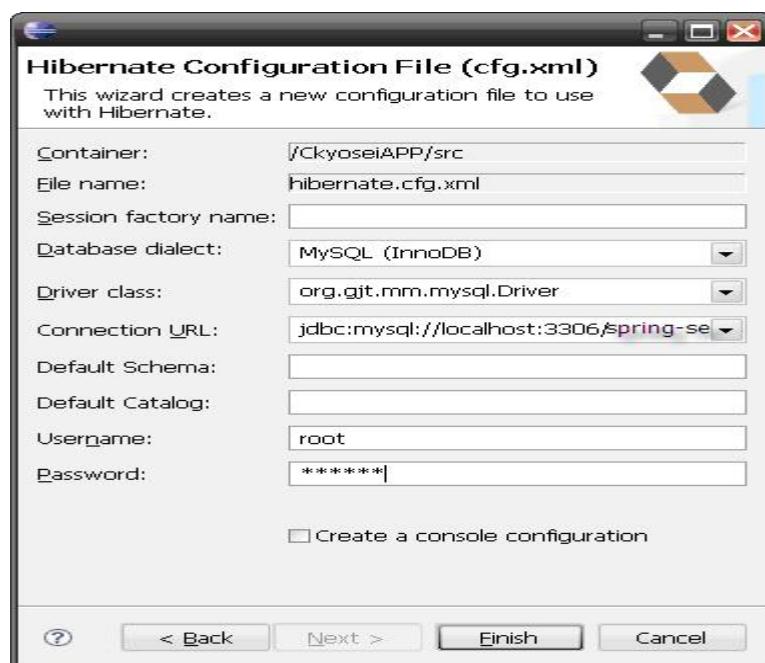


Figura 92 Configuración de la base de datos

3. Pulsamos **Finish** y se creará el siguiente fichero de configuración, que en definitiva contiene los datos que hemos introducido, incluyendo el dialecto utilizado por *Hibernate* para comunicarse con la base de datos, en este caso *MySQL* con el engine *InnoDB* para admitir transacciones.

“

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://Hibernate.sourceforge.net/Hibernate-
configuration-3.0.dtd">
<Hibernate-configuration>
<session-factory>
<property
name="Hibernate.connection.driver_class">org.gjt.mm.my
sql.Driver</property>
<property
name="Hibernate.connection.password">*****</property>
<property
name="Hibernate.connection.url">jdbc:mysql://localhost
:3306/spring-security</property>
<property
name="Hibernate.connection.username">root</property>
<property
name="Hibernate.dialect">org.Hibernate.dialect.MySQLIn
noDBDialect</property>
</session-factory>
</Hibernate-configuration>
```

”

**Tabla 38 Fichero Configuración hibernate**

4. En los siguientes puntos, vamos a configurar la consola de *Hibernate* que es el corazón del plugin. A través de ella, podremos realizar cualquier otra operación, lanzar sentencias HQL, crear código, etc.

Para crear una nueva configuración de la consola de *Hibernate* hacemos **File -> New -> Other... -> Hibernate -> Hibernate Console Configuration**

En la pantalla de configuración introduciremos el nombre de esta configuración, el proyecto asociado, y el fichero de configuración **hibernate.cfg.xml** creado en el punto anterior, donde está configurada nuestra conexión.



Figura 93 Con sola de hibernate

En la pestaña **Options** podríamos indicar si disponemos de alguna clase que nos permita establecer una estrategia de nombrado. Éstas permiten que si, por ejemplo, todas nuestras tablas se llamaran CK\_TABLA, podamos trabajar sólo con el nombre de la tabla y que la clase se encargue automáticamente de la traducción.

5. Antes de pulsar **Finish** seleccionamos la pestaña **Classpath**. Pulsando **Add External JAR** añadimos el paquete donde se encuentra el driver de conexión a la base de datos seleccionada, **mysql-connector-java-5.1.7-bin.jar**, que se encuentra en el directorio de instalación de MySQL.
6. La consola está accesible desde la perspectiva: **Open perspective - Hibernate**

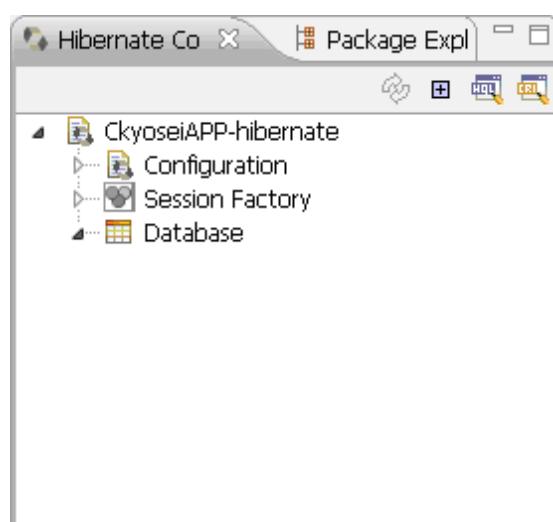
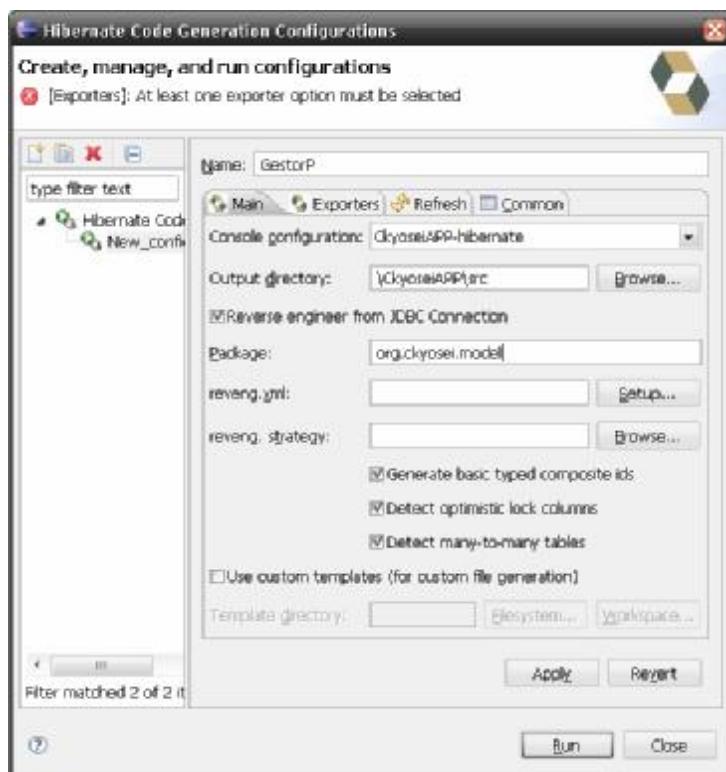


Figura 94 Vista de la consola de Hibernate de la base de datos

7. Para generar el código fuente a partir de las tablas creadas en la base de datos de ejemplo `gestorp`, debemos usar el icono que se creó en la barra de herramientas al instalar las *Hibernate Tools*, seleccionando **Hibernate Code Generation Configurations...**.



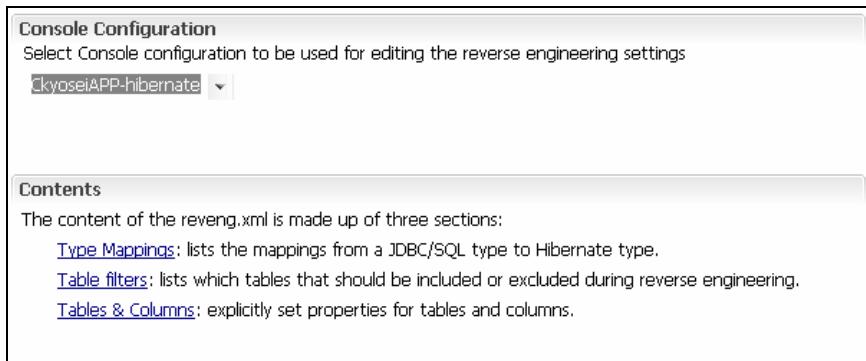
8. En la ventana que aparece, pulsamos sobre el primer ícono, **New launch configuration**. Rellenamos los campos en la zona de la derecha. Seleccionamos la opción **Reverse engineer from JDBC Connection** para realizar ingeniería inversa sobre la base de datos: generar las clases a partir de la información de las tablas contenidas en la base de datos. Rellenamos según muestra la imagen.



9. Seleccionamos la pestaña **Exporters**. Marcamos los **Domain code (java)** para generar los POJOs e **Hibernate XML Mappings (.hbm.xml)** para generar los ficheros de configuración XML donde se describe el mapeo entre las clases y las tablas de la base de datos. Seleccionamos **Use Java 5 syntax**, que permitirá usar el nuevo modelo de colecciones existente a partir de esta versión. **Generate EJB3 annotations** genera POJOs (Plain Old Java Objects) anotados, según el estándar EJB3. Esta es una alternativa a los ficheros xml de mapeo, en la que el mismo POJO, por medio de las anotaciones generadas, indica cómo se debe mapear con la base de datos. Como no vamos a usar EJBs, dejamos esta casilla sin marcar.



10. Pulsamos **Run** para que los mapeos y POJOS sean generados.
  
  
  
  
  
  
  
  
  
11. Mediante el fichero **revenge.xml**, *Hibernate Tools* permite la personalización de los *POJOs*, así como de las relaciones entre las tablas. *Hibernate* usará por defecto tipos primitivos para establecer la relación entre los campos de la base de datos y del modelo. Hay veces en las que interesa que estos campos permitan valores nulos, ya que *Hibernate* es capaz de distinguir si la entidad ya existe en la base de datos o si se trata de una nueva entidad que habrá que añadir. En el caso de las relaciones, *Hibernate* permite por defecto bidireccionalidad en las mismas, pero dado que hay veces que se recupera la información en un solo sentido, no sería necesario el otro. Editando el fichero **revenge.xml** podemos realizar todas estas operaciones, para ello volvemos a abrir la pantalla de Configuración de la generación de código *Hibernate* y pulsaremos el botón **Setup** que aparece al lado del campo **revenge.xml**. En la ventana que se abre, pulsamos **Create new...** para crear el fichero.
  
12. Aparece una nueva pantalla en la que seleccionamos el lugar donde guardar el fichero **revenge.xml**, en el directorio **src**. Pulsando **Next** aparece la pantalla de operaciones. En la parte izquierda, tras pulsar **Refresh** se muestran las tablas de la base de datos. Seleccionamos las tablas que queremos personalizar, y pulsamos **Include...** para que sean añadidas. Pulsamos **Finish** para que se cree el fichero.
  
13. Estas tablas serán incluidas una vez generado el fichero **revenge.xml**, que es abierto automáticamente. Alternativamente podríamos buscar el fichero dentro del proyecto y hacer doble clic sobre él.



14. En la pestaña **Type Mappings** podemos indicar cómo se deben mapear las tipos de la base de datos con los tipos de **Java**. Por ejemplo, en esta sección podemos añadir un mapeo del tipo INTEGER de JDBC al tipo Integer de **Java**, cambiando int por Integer.
15. En la pestaña **Source** se deben realizar a mano los cambios en el tipo de relación. De momento **Hibernate Tools** no permite realizar la personalización gráficamente. Para demostrar como funciona la personalización vamos a hacer que el campo id de las tablas sea autoincremental. Si dispusiéramos de secuencias se añadiría el nombre de la misma. Para realizar esta operación, en la pestaña **Table & Columns** pulsamos **Add** y añadimos las tablas. Posteriormente, las seleccionamos una a una y vamos pulsando **Add Primary key**.
16. Desde la pestaña **Source** podemos modificar manualmente el contenido del archivo **revenge.xml**. Modificamos el contenido del **Tag primary key** tal como se muestra en el fragmento de código abajo. Grabamos los cambios y relanzamos el proceso de generación de código. Si observamos los mapeos generados, veremos que han cambiado, indicando que la clave primaria es autoincremental.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN"
"http://Hibernate.sourceforge.net/Hibernate-reverse-engineering-3.0.dtd" >
<Hibernate-reverse-engineering>
<type-mapping>
<sql-type jdbc-type="INTEGER" Hibernate-type="Integer" not-null="true"></sql-type>
</type-mapping>
<table-filter match-catalog="spring-security" match-name="tabla" />
<table catalog="spring-security" name="tabla">
<primary-key>
<generator class="increment" />
</primary-key>
</table>
</Hibernate-reverse-engineering>
```

**Tabla 39 Fichero Revenge.xml hibernate**

## 6.3.2. Instalación de Spring

### 6.3.2.1. Requerimientos

- Paquete de instalación del *Spring Framework*, versión estable de la serie 2.5.X (**spring-framework-2.5.6-with-dependencies.zip** en el momento de redacción de este manual):

<http://www.springsource.com/download/community?project=Spring%20Framework>

### 6.3.2.2. Proceso de instalación

1. Descomprimir paquete *Spring* en **C:\CKyosei\Herramientas\**. Las librerías de la distribución de *Spring* se encuentran dentro del directorio **dist**. *Spring* es un *framework* modular, lo que permite que se puedan usar independientemente la mayoría de sus módulos. El paquete **Spring.jar**, es el paquete más grande porque es el paquete completo. Los módulos independientes se encuentran dentro de la carpeta **dist/modules**.

### 6.3.2.3. OPCIONAL: Proceso de inclusión de Spring en aplicaciones

1. Vamos a ilustrar cómo sería el proceso de creación de una aplicación java pura y de una aplicación Web, que utilizasen *Spring*.
2. Para crear una aplicación Java pura (aplicación **main**), debemos ir a **File - New - Project - Java - Java Project**. Introducimos el nombre del proyecto y pulsamos **Next**.
  - 2.1. Para añadir las librerías al **classpath** del proyecto, en la siguiente ventana pulsamos sobre la pestaña **Libraries** y pulsamos el botón **Add External Jars...**, buscamos el directorio de *Hibernate* y añadimos la librería **spring.jar** o las librerías de los diferentes módulos que se utilicen.
  - 2.2. Una vez añadidas las librerías debemos de crear el fichero de configuración. *Spring* usa el patrón de inyección de dependencias, soportando las 2 formas de configuración posibles:
    - ü **Setter-based injection** (inyección mediante métodos set en los beans)
    - ü **Constructor-based injection** (inyección mediante constructores)



El interfaz `org.springframework.beans.factory.BeanFactory` es el contenedor *IoC*. Se trata de una interfaz que dirige esencialmente la configuración y la gestión de la aplicación encargándose de crear las instancias de los Beans, bien sean definidos a través de código o, como en este caso, a través de archivos XML. Los Beans pueden ser cualquier tipo de objetos, pero comúnmente son JavaBean estilo “*dassesthat*” es decir, clases con métodos “set” y “get”

El `org.springframework.context.ApplicationContext` extiende al `BeanFactory` y añade nuevas utilidades como el acceso a recursos, la integración con Spring AOP, mensaje de la manipulación de recursos, etc. Por otra parte, `WebApplicationContext` hereda de `ApplicationContext` añadiendo utilidades de acceso al contexto de una aplicación web.

El fichero de configuración de Spring se suele denominar `ApplicationContext.xml` y no tiene porque ser único, ya que puede hacer referencia mediante importaciones a otros ficheros xml, en los que parametrizar los diferentes Beans o componentes que usemos en el proyecto. Por ejemplo, podrían separarse los Beans del MVC de los del Middle tier, etc.

4. Para crear una aplicación Web: `File - New - Other - Web - Dynamic Web Project`. Introducimos el nombre del proyecto y pulsamos `Next`.
  - 4.1. Realizaremos la inclusión de las librerías, pero esta vez las copiaremos directamente dentro del directorio `WebContent/WEB-INF/lib` del proyecto, ya que al ser una aplicación web, serán buscadas dentro de dicha carpeta al ser desplegada.
  - 4.2. De forma similar, crearemos los archivos de configuración de Spring, pero esta vez dentro del aplicativo web. Para ejecutar la aplicación dentro de un servidor web sólo tenemos que exportarla a un fichero `.war` y situarla dentro de un contenedor de aplicaciones web.

### **6.3.3. Sistemas de Trazas**

#### **6.3.3.1. Introducción**

En los sistemas actuales, tanto en desarrollo como en producción es necesario disponer de sistemas de control de trazas, es decir, sistemas que nos permitan un fácil depurado de los métodos que se están ejecutando, con la finalidad de poder rápidamente localizar los errores en el código programado.

Antiguamente esta función se realizaba con `System.out.println` para escribir en la consola. Este método se ha demostrado que es totalmente ineficiente, puesto que las llamadas a System consumen recursos del sistema operativo. Es complicado encontrar los errores en la consola de un servidor ya que es compartida con el resto de aplicaciones que se ejecutan en el mismo. Así mismo no se pueden desactivar los mensajes, con lo que se ralentiza el flujo del aplicativo ya que se produce una operación de E/S para escribir en el fichero que los servidores vuelcan la consola por cada traza.

Debido a estos problemas comentados, han surgido sistemas que nos permiten el correcto control de trazas como `log4j` o el `java.util.Logging`, integrado ya por Sun desde su versión del jdk1.4.

En este documento no vamos a entrar en detalle en el funcionamiento de ambos sistemas de trazas, simplemente vamos a dar una visión global de ambos sistemas mencionando sus características más importantes

#### **6.3.3.2. Log4j**

El primer sistema de éxito que se hizo eco de esta problemática fue log4j.

Log4j es una librería open source desarrollada en Java por la Apache Software Foundation que permite a los desarrolladores de software elegir la salida y el nivel de granularidad de los mensajes (logging) en tiempo de ejecución y no en tiempo de compilación como es comúnmente realizado.

Características de Log4j.

- Diferentes niveles de traza ordenados jerárquicamente.
- Redirección de las trazas a diferentes destinos.
- Diferentes formatos de visualización personalizables mediante Layout.
- Configuración por ficheros tanto de properties, como XML.
- Filtros según categoría.

##### **6.3.3.2.1. Requerimientos**



- Paquete de instalación de *Log4j*, versión estable de la serie 1.2.~~x~~ ([apache-log4j-1.2.15.zip en el momento de redacción de este manual](http://www.apache.org/dyn/closer.cgi/logging/log4j/)):

<http://www.apache.org/dyn/closer.cgi/logging/log4j/>

### 6.3.3.2.2. Niveles de prioridad de trazas en Log4j

*Log4j* está organizado en niveles jerárquicos. *Log4j* posee 7 niveles que se muestran a continuación organizados desde el más bajo al más alto.

En la configuración de *Log4j* indicamos el nivel de traza que queremos poner, desactivando las trazas de los niveles inferiores al indicado, es decir, si indicamos que el nivel de trazas es INFO, el aplicativo mostrará todas las trazas con mayor o igual nivel que INFO.

- **ALL:** este es el nivel más bajo posible, habilita todos los logs. Puede ser especialmente útil en entornos de desarrollo.
- **DEBUG:** se utiliza para escribir mensajes de depuración de código. Por ejemplo dónde empieza y dónde terminan los métodos de una clase.
- **INFO:** se utiliza para mensajes informativos en las aplicaciones.
- **WARN:** se utiliza para mensajes de aviso sobre eventos que se desea monitorizar.
- **ERROR:** se utiliza en mensajes de error de la aplicación que se desean registrar, estos eventos afectan al programa pero no finalizan su ejecución.
- **FATAL:** Errores del programa irrecuperables. Se registrarán y el programa finalizará su ejecución
- **OFF:** este es el nivel más alto posible, deshabilita todos los logs.

Como hemos indicado antes existen 3 formas de configurar log4j.

### 6.3.3.2.3. Configuración del Log4j

- Una es mediante programación usando el API de *Log4j*
- Mediante fichero de propiedades **log4j.properties**
- Mediante fichero XML **log4j.xml**.

Con las dos últimas posibilidades, estos ficheros deberán estar disponibles en el **classpath** del sistema, o se deberá indicar mediante programación dónde se encuentran.

#### *Ejemplo de configuración a nivel de clase*

Para configurar log4j a nivel de clase se puede usar la clase **BasicConfigurator** y el método estático **configure**. Por ejemplo, para que la salida estándar sea por consola:

“

```
package test;

import org.apache.log4j.Logger;
import org.apache.log4j.BasicConfigurator;
public class Log4jTesting1 {

    static             Logger          logger      =
Logger.getLogger("test.Log4jTesting1");

    static public void main(String[] args) {
        BasicConfigurator.configure();
        logger.info("Hello world.");
    }
}
```

”

**Tabla 40 Clase de Test log4j**

**Salida:**

```
0 [main] INFO test.Log4jTesting1 - Hello world.
```

**Ejemplo de configuración mediante fichero log4j.properties**

Si añadimos al classpath el fichero **log4j.properties** y eliminamos la línea de configuración de la clase anterior **BasicConfigurator.configure()**:



**Fichero log4j.properties.**

```
log4j.rootLogger=DEBUG, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c]
- <%m>%n
```

**Tabla 41 log4j.properties**

Si no se pusiera el fichero de configuración en el classpath se puede cargar mediante la clase `PropertyConfigurator.configure("$path/logConfigURL");` del paquete `org.apache.log4j.PropertyConfigurator`, cambiando `$path` por la ruta donde se encuentre el fichero de configuración.

**Salida:**

```
2007-11-21 13:28:43,087 INFO [test.Log4jTesting1] - <Hello
world.>
```

“

```
log4j.rootLogger=DEBUG, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c]
- <%m>%n
```

”

**Tabla 42 Fichero log4j.properties**

“

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
    <appender name="stdout" class="org.apache.log4j.ConsoleAppender">
        <param name="Threshold" value="DEBUG" />
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="[%t] %-5p %c - %m%n" />
        </layout>
    </appender>
    <root>
        <priority value="DEBUG" />
        <appender-ref ref="stdout" />
    </root>
</log4j:configuration>
```

”

**Tabla 43 Fichero log4j.xml**



Si no se pusiera el fichero de configuración en el **classpath** se puede cargar mediante la clase `DOMConfigurator.configure( "$path/logging.xml" );` del paquete `org.apache.log4j.xml.DOMConfigurator`, cambiando `$path` por la ruta donde se encuentre el fichero de configuración.

**Salida:**

```
2007-11-21 13:28:43,087 INFO [test.Log4jTesting1] - <Hello  
world.>
```

**6.3.3.2.4. Categorías y niveles**

*Log4j* funciona según un modelo de herencia para las diferentes categorías que se configuren en los ficheros de configuración. A continuación pondremos varios ejemplos para clarificar esto.

Configuramos el root a nivel de Debug y el paquete x lo dejamos por defecto.

Logger name	Assigned level	Effective level
root	DEBUG	DEBUG
x	none	DEBUG

“

```
package x;

import org.apache.log4j.Logger;

public class Uno {

    static Logger logger = Logger.getLogger("x.Uno");

    static public void main(String[] args) {

        logger.debug("Hello world.");
    }
}

log4j.rootLogger=DEBUG, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c]
- <%m>%n
```

”

**Tabla 44 Ejemplo Log 1**

**Salida:**

2007-11-21 23:35:42,406 DEBUG [x.Uno] - <Hello world.>

Como se observa heredamos el nivel del log padre Root.

<b>Logger name</b>	<b>Assigned level</b>	<b>Effective level</b>
root	DEBUG	DEBUG
x	ERROR	ERROR
x.y	INFO	INFO
x.y.z	DEBUG	DEBUG



Si modificamos el fichero **log4j.properties** para la anterior configuración, se puede observar que para las trazas del paquete x sólo sacarán aquellas trazas cuyo nivel sea mayor o igual que ERROR. Para el x.y sólo sacara las que sean mayores que INFO, etc.



```
log4j.rootLogger=DEBUG, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c]
- <%m>%n
log4j.logger.x=ERROR
log4j.logger.x.y=INFO
log4j.logger.x.y.z=DEBUG

package x.y;

import org.apache.log4j.Logger;

public class Dos {
    static Logger logger = Logger.getLogger("x.y.Dos");
    static public void main(String[] args) {
        logger.info("Hello world Two.");
    }
}

package x;

import org.apache.log4j.Logger;

public class Uno {
    static Logger logger = Logger.getLogger("x.Uno");
```

```

        static public void main(String[] args) {
            logger.error("Hello world One.");
        }

    }

package x.y.z;

import org.apache.log4j.Logger;

public class Tres {
    static Logger logger = Logger.getLogger("x.y.z.Tres");
    static public void main(String[] args) {
        logger.debug("Hello world Three.");
    }
}

```



**Tabla 45 Log4j ejemplo 2**

### 6.3.3.2.5. Appenders

Una vez establecido el nivel de trazas del código hay que configurar dónde queremos que las trazas sean escritas. Como ya indicamos las trazas pueden ser guardadas en varios lugares, el más corriente es la consola o un fichero. Aunque Log4j, también puede ser configurado para almacenar las trazas en la base de datos o que estas sean enviadas por correo electrónico.

#### Appender consola

```

log4j.rootLogger=DEBUG, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender

```

#### Appender RollingFileAppender



```
log4j.rootLogger=DEBUG, stdout, Fichero  
log4j.appender.Fichero= org.apache.log4j.RollingFileAppender  
log4j.appender.Fichero.File=c:/log/Trazas.log
```

Además de estos *Appenders*, que son los dos más utilizados, se pueden encontrar otros en la documentación del proyecto *log4j*.

### 6.3.3.2.6. Layouts

El último de los elementos configurables es *Layout*, que sirve para configurar la forma en la que se representan la información en cada traza. A cada *appender* se le puede añadir un *layout*.

El más común es **PatternLayout**, que recibe un patrón que indica cómo es la salida. Por ejemplo, para mostrar hora `%d` Prioridad `%p` [categoría] `[%c]` <mensaje> `<%m>` salto de línea `%n`, de la siguiente forma:

```
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout  
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] -  
<%m>%n
```

### 6.3.3.3. java.util.logging

Es el sistema de trazas introducido por Sun a partir de la JDK1.4 es muy similar al anterior sistema descrito en el punto anterior. Con diferentes niveles y en lugar de *appenders*, se utilizan *handlers* o manejadores.

#### 6.3.3.3.1. Niveles de prioridad de trazas en java.util.logging

- ü SEVERE (valor mas alto)
- ü WARNING
- ü INFO
- ü CONFIG
- ü FINE
- ü FINER
- ü FINEST (valor mas bajo)

### **6.3.3.3.2. Handler**

En lugar de *Appenders* como en Log4j este sistema de trazas utiliza *handlers*. Por defecto viene con 5 *handlers* para poder configurarlo.

- |   |         |
|---|---------|
| ü | Stream  |
| ü | Console |
| ü | File    |
| ü | Socket  |
| ü | Memory  |

A cada *handler* se le puede asociar, como con Log4j, un formato (*layout* en log4j), en este caso podemos usar un String (SimpleFormatter) o un XML de formato (XMLFormatter).

## Programa básico de Log:

```
package test;

import java.io.*;
import java.util.logging.*;

public class TestSunLogging {
    public static void main(String[] args) {
        Logger logger = Logger.getLogger("test. TestSunLogging ");

        // Log a few message at different severity levels
        logger.severe("severe message");
        logger.warning("warning message");
        logger.info("info message");
        logger.finest("finest message");
    }
}
```

**Tabla 46 Programa básico de Log**



### **6.3.3.4. Jakarta-Commons-Logging (JCL)**

Como hemos visto hay varios sistemas de trazas que actualmente se pueden elegir, bajo el proyecto **commons** de Jakarta, se ha creado un pequeño subproyecto gracias a el cual se puede intercambiar de un sistema de trazas a otro sin tener que cambiar ni una línea de código.

#### **6.3.3.4.1. Requerimientos**

- Paquete de instalación de *Jakarta common-logging* versión estable de la serie 1.1.X (**commons-logging-1.1.1.zip en el momento de redacción de este manual**):

[http://commons.apache.org/downloads/download\\_logging.cgi](http://commons.apache.org/downloads/download_logging.cgi)

Para realizar esta operación se necesita:

- Fichero **common-logging.properties**
- Añadir al classpath el jar **commons-logging.jar**

En el fichero **common-logging.properties** indicamos el sistema de trazas que queremos utilizar.

Así, en lugar de utilizar las clases directamente de *Log4j* o *java.util.Logging* usamos la factoría suministrada por JCL, que consulta en el fichero **common-logging.properties** qué sistema de trazas ha de configurar. De este modo podemos pasar de un sistema a otro de forma transparente.

#### **common-logging.properties**



```
# commons logging config - use Log4J
org.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JCategoryLog

package x;
//import org.apache.log4j.Logger;
import org.apache.commons.logging.*;
public class Uno {
    //static Logger logger = Logger.getLogger("x.Uno");
```

```
    static Log logger = LogFactory.getLog(Uno.class);  
  
    static public void main(String[] args) {  
        logger.error("Hello world One..");  
    }  
}
```



**Tabla 47 common-logging.properties**

## **7 DESARROLLO APLICACIONES**

---





## ***7. Desarrollo aplicaciones.***

---

Este bloque, proporciona una vista previa de las tecnologías clave que hemos usado en el proyecto de creación del Sistema Kyosei-Polis y del proceso de desarrollo vamos a utilizar en el mismo.

En este bloque, se obtendrá una visión general de los siguientes puntos:

- ü Resumen de tecnologías y herramientas de desarrollo utilizadas para construir la aplicación alpha. Para más detalle ver documento de instalación del entorno.
- ü El proceso de desarrollo de software utilizado para construir la aplicación de ejemplo.

## 7.1. Tecnologías empleadas en el desarrollo

Tecnología seleccionada	Categoría	Alternativas libres o OpenSource
Spring Framework ( <a href="http://springframework.org">springframework.org</a> )	Framework web	Struts framework, Tapestry, JSF, otros
Spring-security	Framework de seguridad (antiguo Acegi)	
DisplayTags ( <a href="http://DisplayTag">DisplayTag</a> )	Creación de Tablas	JMesa
Hibernate ( <a href="http://Hibernate.org">Hibernate.org</a> )	Framework de persistencia	JDO, EJB, otros.
Apache Tomcat ( <a href="http://tomcat.apache.org">tomcat.apache.org</a> )	Servidor web/ contenedor servlets	Caucho Resin, Jetty, Jboss, etc.
Apache web ( <a href="http://http.server.apache.org">http.server.apache.org</a> )	Servidor Web	Cherokee, lighttpd
Eclipse SDK ( <a href="http://eclipse.org">eclipse.org</a> )	IDE	NetBeans, otros.
<b>Maven</b>	Administración/ configuración/ generación	Ant, MKS make, etc...
JUnit ( <a href="http://junit.org">junit.org</a> )	Test y pruebas unitarias	Mock, etc.
Mysql	Base de datos	Hsqldb, postgress, etc.
Mozilla Firefox	Navegador	N.Navitagor
Sitemesh	Decorador	Title



### **Spring Framework**

Spring Framework (también conocido simplemente como Spring) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa y un sustituto del modelo de Enterprise JavaBean. Por su diseño, el framework ofrece mucha libertad a los programadores en Java y soluciones muy bien documentadas y fáciles de usar, prácticas comunes en la industria.

Spring Framework está diseñado como una serie de módulos que pueden trabajar independientemente uno de otro. Además, intenta mantener un mínimo acoplamiento entre la aplicación y el propio framework de forma que podría ser desvinculado de él sin demasiada dificultad.

### **Spring-security**

Antes conocido como Acegi es una API open source que provee de servicios de autenticación y autorización a las aplicaciones basadas en Spring. Desde su primera release se ha convertido en una de las mejores opciones y más extendidas para este fin.

### **DisplayTags**

Es una librería de tags jsp open source que es de gran utilidad a la hora de trabajar con una arquitectura MVC.

Esta librería de tags sirve de gran utilidad cuando se quiere mostrar una colección de datos en forma de tabla, permitiendo, además, entre otras cosas, agregarle estilos, decoradores (decorators) a los datos mostrados, exportar a distintos formatos, utilizar paginación y ordenación de los datos representados.

### **Hibernate**

Hibernate es una herramienta de Mapeo objeto-relacional para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

### **Apache Tomcat**

Es un servidor web (http) y funciona como un contenedor de servlets. Es la implementación de referencia de las especificaciones de servlets 2.5 y de Java Server Pages (JSP) 2.1, especificaciones para Java Community Process, usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Apache Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

### **Servidor Apache web**

Servidor web (Acrónimo de "a patchy server") de distribución libre y de código abierto, siendo el más popular del mundo desde abril de 1996.

### **Entorno de trabajo Eclipse SDK**

Eclipse es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios Web, programas en C++ o aplicaciones Java.

Se trata de un entorno de desarrollo integrado (IDE) en el que encontrarás todas las herramientas y funciones necesarias para tu trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar.

### **Maven**

**Maven** es una herramienta software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002.

**Maven** es una herramienta open source, mediante la cual podemos administrar ciertas etapas del ciclo de vida de nuestros proyectos, entre variadas cosas más.

### **JUnit**

JUnit es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

### **Base de datos Mysql**

El servidor de bases de datos MySQL es la base de datos de fuente abierta más popular en el mundo. Su arquitectura lo hace extremadamente rápido y fácil de adaptar.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario

### **Mozilla Firefox y herramientas**

Mozilla Firefox es un navegador de Internet libre y de código abierto descendiente de Mozilla Application Suite, desarrollado por la Corporación Mozilla, la Fundación Mozilla y un gran número de voluntarios externos.

Además de ser un navegador rápido y ágil, contiene un conjunto de plugins que permiten el desarrollo dinámico y comodo de portales web, como firebug, Developer tools, otros.

### **Sitemesh**

Sitemesh es un framework de decoración y layout para páginas web, que facilita la creación de grandes sitios que contienen gran cantidad de páginas, las cuales requieren un consistente “look and feel”, navegación y complejo sistema de capas.



## **7.2. Metodología empleada.**

Para la resolución de este proyecto y de los futuros proyectos usaremos una combinación de las mejores prácticas y técnicas recomendadas por:

- ü XP, Programación eXrema (<http://extremeprogramming.org>)
- ü AMDD, modelo ágil de desarrollo (<http://agilemodeling.com>)

AMDD se centra en el modelado (modelo de la interfaz de usuario, por ejemplo).

XP se centra en el ciclo de vida completo.

### **7.2.1. Principios del Desarrollo Ágil.**

- ü Nuestra mayor prioridad es satisfacer al cliente a través de la entrega temprana y continua de software con valor.
- ü Los responsables de negocio y los programadores deben trabajar juntos diariamente a lo largo del proyecto.
- ü Conversación cara a cara.
- ü La atención continua a la excelencia técnica y los buenos diseños mejoran la agilidad.
- ü A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo.
- ü Las mejores arquitecturas, requisitos y diseños surgen de equipos que se auto-organizan.
- ü Construimos proyectos con profesionales motivados.
- ü Simplicidad es esencial.
- ü Entregamos software frecuentemente.
- ü Software que funciona es la principal medida de progreso.
- ü Aceptamos requisitos cambiantes, incluso en etapas avanzadas.
- ü Los procesos ágiles promueven el desarrollo sostenible.

*«Lo que es nuevo en los procesos ágiles no son las prácticas que usan, sino que reconocen a las personas como primeros implicados en el éxito de un proyecto, además de un intenso foco en la efectividad y la manejabilidad. Esto genera una nueva combinación de valores y principios que definen una visión ágil del mundo.»*

*(Highsmith & Cockburn 2001)*

---

## 7.2.2. Características de un método ágil

El desarrollo de software debe ser incremental, cooperativo, simple y adaptativo.

- Ú Incremental: Pequeñas entregas y ciclos cortos.
- Ú Cooperativo: Cliente y programadores trabajarán codo a codo para avanzar rápido y sin desviaciones en el proyecto.
- Ú Simple: el método es fácil de aprender y modificar.
- Ú Adaptativo: Siempre abierto a incorporar cualquier cambio de última hora que estime el cliente.

## 7.2.3. ¿Qué es eXtreme Programming?

La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Fue desarrollada por Kent Beck.

*«Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar»*

(Kent Beck)

---

Siempre es una buena idea definir qué problema se plantea y como se está resolviendo. Para quién se está resolviendo y por qué nuestra solución es importante para el cliente. Además, es importante entender cuáles son las expectativas del cliente, por ejemplo, para cuándo se requiere la solución y cuál es el alcance del proyecto dentro de su organización.

La metodología XP se sustenta en 4 pilares básicos.

- Ú Coste: Máquinas, especialistas y oficinas.
- Ú Tiempo: Total y de Entregas.
- Ú Calidad: Externa e Interna.
- Ú Alcance: Intervención del cliente.

## 7.2.4. ¿Por qué eXtreme Programming?

- Ú Se consiguen productos usables con mayor rapidez.
- Ú El proceso de integración es continuo, por lo que el esfuerzo final para la integración es nulo.
- Ú Se consigue integrar todo el trabajo con mucha mayor facilidad.



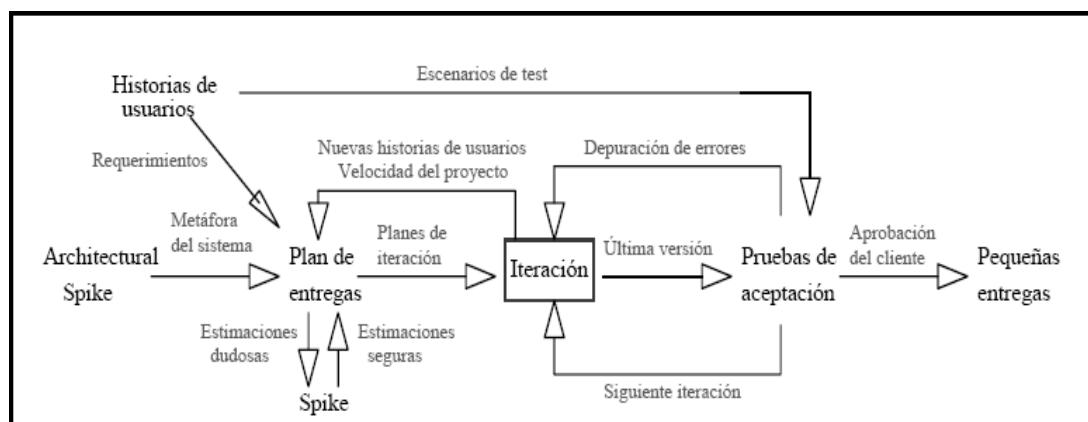
- ü Se atienden las necesidades del usuario con mayor exactitud. Esto se consigue gracias a las continuas versiones que se ofrecen al usuario y a la comunicación continua con él.
- ü Se consiguen productos más fiables y robustos ante los errores, gracias al diseño de los test de forma previa a la codificación.
- ü Obtenemos código más simple y más fácil de entender, reduciendo el número de errores.

Existen otras múltiples ventajas de trabajar con programación extrema pero en este documento sólo he citado las más significativas.

### 7.2.5. Cuando usar la Metodología XP

El uso de la metodología XP es recomendable siempre que se cumplan una serie de condiciones:

- ü Semanas de 40 horas de trabajo.
- ü Todos los programadores deben estar en una única habitación.
- ü Todos los días comienzan con una reunión de apertura.
- ü Alta velocidad de trabajo.



**Figura 95 Ciclo de vida de XP**

## 7.2.6. Trabajando con XP

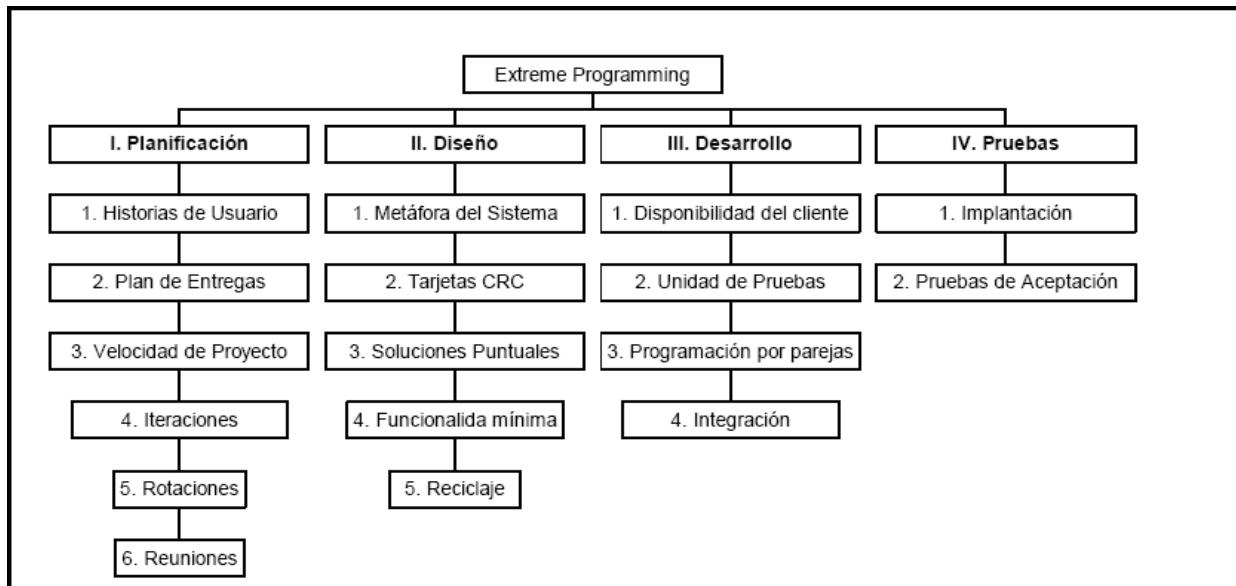


Figura 96 Fases de la metodología XP

### 7.2.6.1. Fases del la Metodología XP

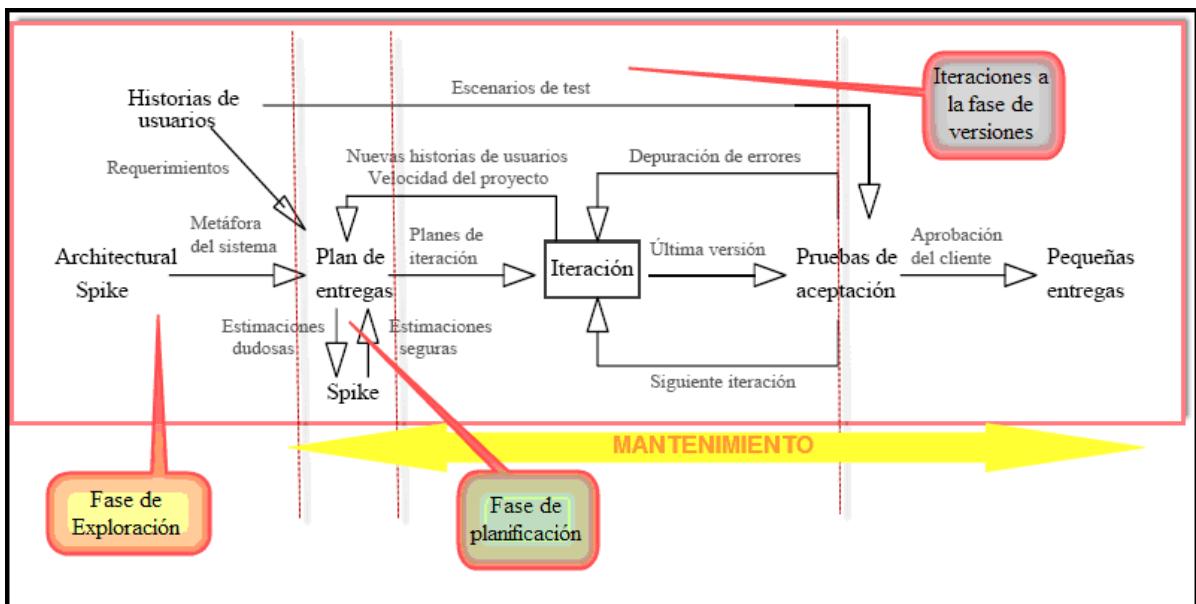


Figura 97 Fase de XP en detalle



### 7.2.6.1.1. Fase de Exploración:

Algunas actividades que podrían tener lugar en esta fase de exploración del proyecto:

- Ú Un modelo de dominio que nos ayudará a definir los principales conceptos de negocio (entidades) y las relaciones entre ellos.
- Ú Prototipos de Interfaces de usuario y flujo de pantallas (*Storyboard*).
- Ú Historias de usuarios.
- Ú Definición del alcance Es importante definir el alcance del proyecto por adelantado a fin de que se conozca qué es lo que hay que desarrollar inmediatamente y lo que puede ser aplazado.

### 7.2.6.1.2. Fase de Planificación:

Debe incluir:

- Ú Plan de liberación para la próxima versión de un sistema. Enumera todas las historias de usuario que se incluirá en la próxima versión del sistema, agrupadas en varias iteraciones.
- Ú Un plan de iteración, este plan se desarrolla antes de cada iteración. Incluye las historias de usuario que el cliente quiere aplicar en la próxima iteración.
- Ú Definir los sistemas de nombrado de código, así como de base de datos.

En este punto AMDD incluye un extra que es la participación activa de los interesados.

Éstos (los usuarios o sus representantes) tienen la autoridad y la capacidad de proporcionar la información relativa al sistema que se está construyendo y puede hacer efectivas las decisiones relativas a los requisitos y el establecimiento de prioridades del mismo.

### 7.2.6.1.3. Fase de Iteraciones hacia una versión (construcción incremental del software)

El desarrollo iterativo significa que cada iteración incluye el diseño, codificación, la aceptación de los usuarios, y el despliegue de código en el entorno de prueba/producción.

Cada iteración incluyen las siguientes actividades:

- Ú Tareas de desarrollo, las estimaciones de los Programadores, y un plan para la próxima iteración.
- Ú Diseño de tarjetas CRC, diagramas UML, y así sucesivamente.
- Ú Código de prueba en primer lugar, refactorizar código/ base de datos / arquitectura, según sea necesario, optimizar más tarde.
- Ú Pruebas de aceptación del usuario (UAT).
- Ú Desplegar la iteración.

#### **7.2.6.1.4. Alcance del proyecto.**

Definir una tabla con las historias que tenemos que incluir y con lo que podemos aplazar.



## 7.3. La aplicación alpha

En informática, una aplicación es un programa informático diseñado para facilitar al usuario la realización de un determinado tipo de trabajo. Búsqueda de soluciones a un problema que se plantea.

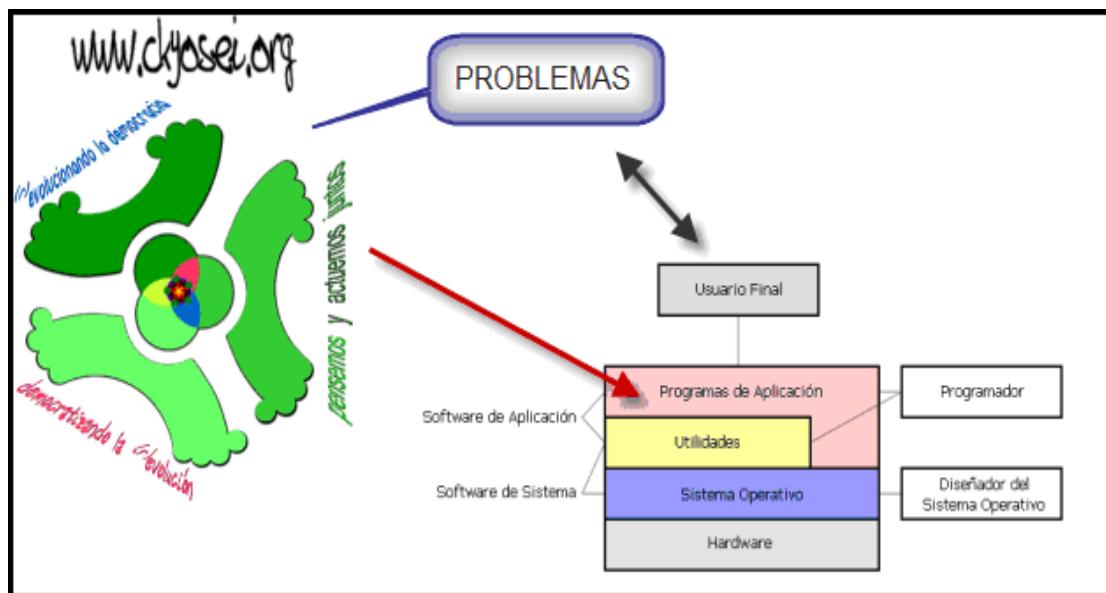


Figura 98 Problemática Ckyosei

En este proyecto vamos a poner la primera piedra, de las múltiples que se necesitarán para el desarrollo del sistema global de *Kyosei-Polis*.

El Sistema Kyosei-Polis Fue iniciado en el año 2005 como una investigación dentro del programa de doctorado de la *Universitat Oberta de Catalunya*, con el título:

**e-Participa: Diseño de un Entorno Virtual de Participación Ciudadana Municipal a partir de un análisis interdisciplinar de la participación ciudadana**

Que buscaba delimitar cuáles son los requisitos que debe cumplir un Entorno Virtual para la Participación Ciudadana municipal y el fomento de las redes ciudadanas que pudiese ser utilizado en municipios de países en desarrollo. Por ello, gran parte del trabajo de campo tuvo lugar en la ciudad de Fortaleza (Brasil) y en Guatemala.

A medida que se desarrollaba el proyecto, sus objetivos fueron creciendo. No sólo se pretende ahora proporcionar la descripción detallada del sistema sino iniciar asimismo su construcción como proyecto de Software Libre.

Este es el objetivo en el que esta basado este proyecto, primero de todo proporcionar el entorno tecnológico sobre el cual se implantará el sistema. Se trata de crear el soporte tecnológico sobre el cuál

giren todos los futuros desarrollos. Además de establecer este entorno se quiere establecer una filosofía de trabajo basada en metodologías ágiles y programación extrema, para desarrollo de las siguientes partes del sistema.

Además de establecer el marco tecnológico, se va a crear una aplicación que sirva de modo de prueba de todo el sistema, y que sirva de punto inicial a todas las futuras aplicaciones.

Esta aplicación que denominamos alpha, va a constar de 3 módulos principales:

1. Arquetipo *Maven* desarrollado para el sistema: Este arquetipo creará mediante la ejecución de un comando una aplicación Web basada en *Spring framework*. Esta aplicación a su vez llevará ya incluida la autenticación contra base de datos con el framework de seguridad *Spring security*.

La inclusión de *Spring security* va a proporcionar dos pilares que todas las aplicaciones informáticas van a usar, autenticación y permisos o ACLs.

- 1.1. La primera básica en todo sistema informático, permitirá saber quien es la persona que interactúa con el sistema y que páginas tiene derecho a ver, según su Rol en el mismo.
- 1.2. La segunda permitirá, definir la creación de entidades y compartir las a usuarios o grupos independientes, aportando la seguridad necesaria a la operación.

Además de *Spring security*, el arquetipo configurará la aplicación Web generada para usar *Sitemesh*. Como ya hemos visto en el apartado de tecnologías. *Sitemesh* es un framework de decoración y creación de capas (*layout*) para páginas web, que facilita la creación de grandes sitios que contienen gran cantidad de páginas, las cuales requieren un consistente aspecto (look and feel), navegación y complejo sistema de capas..

La aplicación generada, incluirá un sistema de *Taglibs* para la creación de menús dinámicos basados en un fichero de configuración *XML*. Este sistema permitirá definir todos los menús de un aplicativo, en un fichero *XML*, decidir que Roles tienen acceso a ellos, y ser invocado simplemente mediante la inclusión de una librería de *Taglibs*.

Además esta aplicación Web, ya quedará con todos los archivos necesarios de configuración para el uso de *Hibernate* como framework de persistencia de datos, *DisplayTags* como herramienta de presentación de tablas.

Esta aplicación también incluirá los archivos necesarios de internacionalización basados en el estándar I18N.

Por último el arquetipo incluirá el *framework* de CSS de *Mike Stenhouse* que permite crear portales con un aspecto robusto, y lo que es más importante su evolución en el tiempo, ya que cambiar el aspecto no supone nada más que cambiar hojas de estilo no cambiar toda la página web completa (código, CSS, ).



2. Paquete de utilidades: Se va a crear un paquete de utilidades que se incluirá en todos los proyectos que permitirá operaciones comunes a todos los proyectos Web como es el acceso a base de datos mediante DAO, acceso a lógica de negocio y transacciones (*Manager o Service*) .

Aplicación Web que denominaremos alpha, generada a partir del arquetipo, y que use el paquete de utilidades, que sirva de ejemplo al futuro desarrollo de los diferentes módulos del sistema.

3. Esta aplicación servirá también de ejemplo para aquellos programadores que no hayan trabajado con **Spring framework**, ya que en ella vamos a explicar algunos de los diferentes Modelos Vista Controlador de **Spring** Así como el sistema de Inyección de dependencias de **Spring** y otras capas del Framework. También haremos uso de la capa de acceso a datos, para ello usaremos el paquete de utilidades del punto anterior.

La aplicación estará construida a partir de la aplicación generada por el arquetipo. Como dijimos en el punto primero, el arquetipo ya integra la autenticación de usuarios basada en **Spring security**.

Para esta aplicación vamos a crear 2 Roles, cada uno con una funcionalidad propia, y dependiendo del usuario conectado en la aplicación, este podrá realizar diferentes operaciones.

Los roles son **ROLE\_ADMINISTRADOR**, **ROLE\_USER**, correspondientes al usuario Administrador (Usuario con privilegios), y a un Usuario normal o sin privilegios.

El rol **ROLE\_USER** podrá realizar las siguientes operaciones:

- ü Completar su perfil de datos personales.
- ü Ver y Subscribirse a un Grupo de Trabajo.
- ü Crear/ Eliminar/Modificar/Visualizar sus Documentos.
- ü Dar Permisos/Eliminar Permisos sobre sus documentos.
- ü Ver documentos de otros usuarios a los que ha sido autorizado.

El usuario **ROLE\_ADMINISTRADOR**:

- ü Todas las operaciones del rol anterior.
- ü Crear / Modificar/ Visualizar /Eliminar Grupos de trabajo.

También veremos comoharemos uso de un Framework de CSS desarrollando una versión propia para la aplicación alpha, comprobando la facilidad para cambiar de aspecto de forma rápida, vistosa.

Combinaremos la creación de estos estilos con el uso de **Sitemesh** para crear la estructura de capas y aspecto definitivo que tendrán las páginas.

Qué vamos a ver en este punto:

1. Establecer una metodología de desarrollo de software basado en la de programación extrema (XP) y Modelado Ágil (AMDD).
2. Comprensión de requerimientos de negocio del aplicativo.
3. Desarrollar algunos artefactos de alto nivel de tales como un modelo de dominio, la interfaz de usuario prototipos, etc.
4. Estableceremos un plan de entregas.

### 7.3.1. Historias de usuario

El concepto de las historias de usuario (*user-stories*) tiene algo que ver con los famosos casos de uso, utilizados en el tradicional ciclo de desarrollo de software en cascada, ya que su cometido es similar. Pero las “historias de usuario” no son realmente lo mismo que los “casos de uso”. Nos permiten sustituir unos largos requerimientos por una serie de “historias del usuario” y además nos permiten hacer una estimación del tiempo para el lanzamiento de las futuras versiones de nuestro sistema.

Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema.

Las historias de usuario nos ayudan a crear test de aceptación. Estos son pruebas que se aplicarán al sistema para ver si cumplen una determinada “historia del usuario”, lo que viene a ser lo mismo que cumplir un determinado requisito en otros modelos de desarrollo.

Estas historias serán creadas en tarjetas físicas, en las que todos los asistentes a la reunión de trabajo escribirán mientras dialogan acerca de las necesidades del sistema.

Así, el proceso consiste en que el cliente enuncia una necesidad y se habla sobre ella,

Procurando que no quede ningún punto oscuro o indefinido. Cada participante irá anotando en las tarjetas lo que cada uno cree que puede ser una buena definición informal. Estas definiciones serán puestas en común hasta refinarlas en unas nuevas tarjetas.

El objetivo es conseguir es conseguir una serie de tarjetas, que cada una “historia de usuario” detallada en 4 o 5 líneas, de forma que la carga de trabajo para desarrollarla sea de más o menos una semana.

En el proyecto los integrantes de estas reuniones fueron:

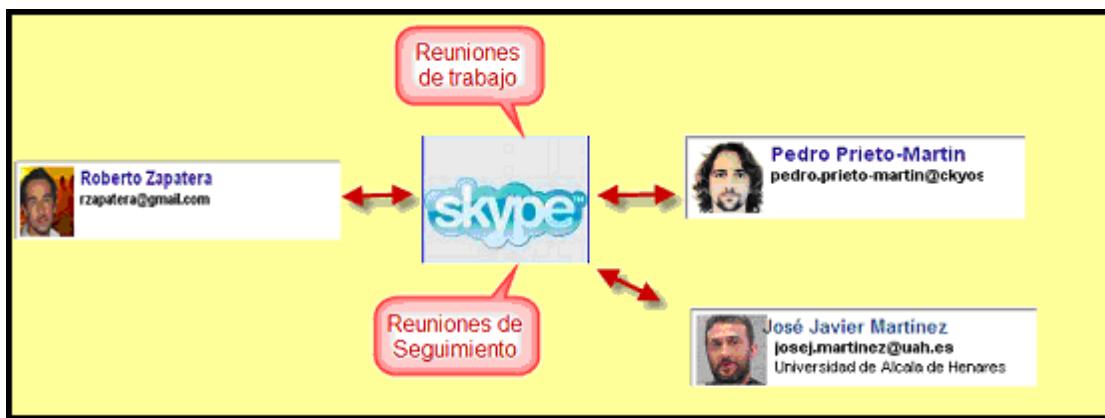
- ü Pedro Prieto Martín como Coordinador del proyecto de creación del Sistema Kyosei-Polis.
- ü Roberto Zapatera, Programador de la fase 0 del sistema (pre-alpha).

Siendo el Dr. José Javier Martínez, profesor de la Universidad de Alcalá, el responsable académico del proyecto por parte de la U.A.H.



Las reuniones, en contra de lo recomendado por la XP, fueron realizadas a través de Skype un software para realizar llamadas sobre Internet (VoIP), ya que debido a que el coordinador del proyecto de creación del Sistema Kyosei-Polis reside en Guatemala y era imposible realizarlas físicamente.

Cabe reseñar que esto no ha sido ninguna rémora en el desarrollo de las reuniones ni para la puesta en común de ideas, ya que la tecnología hoy en día permite tener reuniones virtuales igual que si todos estuviéramos en la misma sala.



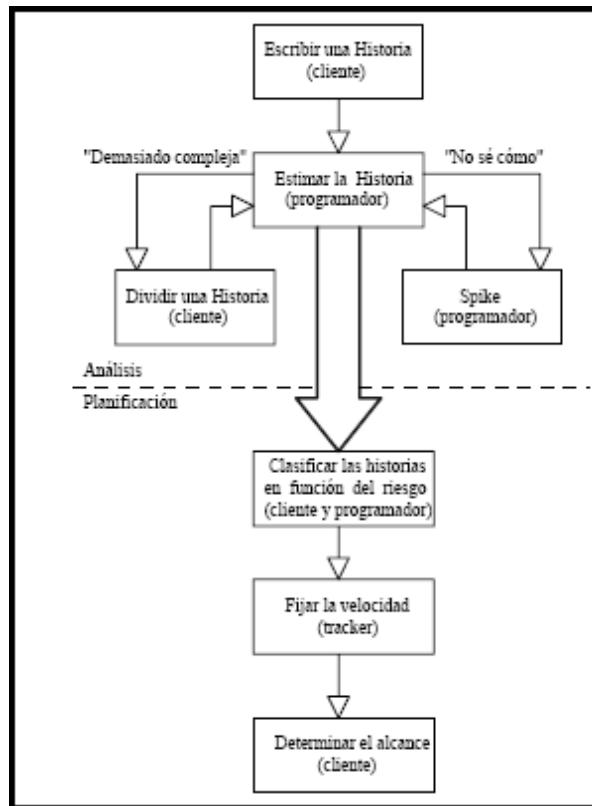
**Figura 99 Esquema reunión Skype**

En el documento vamos a incluir sólo las historias de usuario inicial y las historias ya refinadas para las tareas específicas que tenía que acometer el sistema. Estas historias han sido informatizadas para su inclusión en el documento para mayor claridad.

Las historias de usuario, no son las típicas historias de un Cliente – Jefe Proyecto, en las que el cliente desconoce totalmente la tecnología y sólo tiene conocimiento sobre su dominio. Este suceso aunque anormal, es debido al perfil técnico de todos los integrantes de la misma.

Coordinador del Proyecto Kyosei-Polis	Pedro Prieto
	NEW X FIX
Se pretende el diseño de una aplicación inicial o alpha, que trabaje con las tecnologías que se han tomado como base para el proyecto, y que servirá como punto de partida inicial al futuro sistema. El sistema debe ser desarrollado mediante el uso de metodologías de trabajo XP.	
La aplicación que se propone deberá gestionar el alta de usuarios, agruparlos por roles, así como la creación de atributos específicos para los usuarios.	
Así mismo se deberá suministrar mecanismos de autenticación de usuarios y establecer un acceso restringido a las diferentes partes de las aplicaciones.	
Por otro lado se pretende la agrupación de usuarios en grupos.	
Que los usuarios puedan adjuntar documentos y permitir el acceso a ellos por parte de otros usuarios.	

	<b>A. Funcional /Programador del proyecto.</b>	<b>Roberto Zapatera</b>
	<b>NEW</b> X <b>FIX</b>	
<p>Diseño de un sistema que pruebe las tecnologías del proyecto. Creando un aplicativo que permita la gestión de usuarios, roles y autenticación, agrupación en listas de usuarios, creación de grupos y documentos y ACL o permisos sobre éstos.</p> <p>Se propone la creación de un arquetipo usando la funcionalidad de <i>Maven</i> que genere una aplicación inicial ya configurada, que sirva como base para todos los desarrollos futuros, incluido el alpha.</p> <p>La aplicación generada estará configurada para el uso de Spring, Hibernate, Sitemesh, CSS framework,etc. Esta aplicación Web podrá entonces ser adaptada para el dominio concreto del problema.</p> <p>Una vez desarrollado el arquetipo, crear el aplicativo a partir de el..</p> <p>Así mismo, se plantea la creación de un paquete de utilidades genéricas que permita el rápido desarrollo de los proyectos futuros.</p>		
<p>A partir de estas dos “Historias de usuario” iniciales, en las múltiples reuniones posteriores se generaron tres nuevas “Historias de usuario” principales, que fueron la base para el trabajo de desarrollo del arquetipo y la versión alpha del sistema.</p>		



## **Figura 100 Diagrama generación historias de usuario**

	<b>A. Funcional /Programador del proyecto.</b>	<b>Roberto Zapatera</b>
	<b>NEW X FIX</b>	
Diseño de librería de utilidades común a todos los proyectos.		

	<b>A. Funcional /Programador del proyecto.</b>	<b>Roberto Zapatera</b>
	<b>NEW X FIX</b>	
	<b>Diseño del Arquetipo.</b>	

	<b>A. Funcional /Programador del proyecto.</b>	<b>Roberto Zapatera</b>
	<b>NEW X FIX</b>	
Diseño de la aplicación alpha basada en el arquetipo y usando la librería de utilidades.		

A continuación se muestra una tabla con las diferentes “tareas” para cada una de las historias anteriores, en la que estimamos sus prioridades, así como los puntos necesarios para su desarrollo. Estos puntos pueden ser días o semanas, según la unidad de medida que se establezca para el proyecto. En la siguiente tabla, vamos a tomarlos como días.

Lamentablemente la velocidad de desarrollo no es tan alta como sería recomendable por XP debido a que todos los integrantes del proyecto no podíamos dedicarles las 8 horas que recomienda la programación extrema para un proyecto, ya que teníamos que compatibilizar la realización del proyecto con los compromisos laborales.

### Diseño de librería de utilidades

NOMBRE HISTORIA (TAG)	DESCRIPCIÓN	PRIORIDAD	PUNTOS
ESTRUCTURA DE PAQUETES	DISEÑO ESTRUCTURA DE PAQUETES	1	1
EXCEPCIONES	CREACIÓN CLASES EXCEPCIONES	2	2
DAO	CREACIÓN DE CLASES DE DAOs GENÉRICOS	2	3
SERVICE	CREACIÓN DE CLASES DE LÓGICA DE NEGOCIO GENÉRICA	2	2
MODEL	CLASE PADRE DE LOS MODELOS DE DOMINIO EMPLEADOS EN LOS APLICATIVOS	3	1/4
SEGURIDAD	CLASE PARA CODIFICAR UNA CONTRASEÑA SEGÚN ALGORITMO ESPECIFICADO SHA/MD5	4	1
TOTAL			9.25



### DISEÑO DEL ARQUETIPO

NOMBRE HISTORIA (TAG)	DESCRIPCIÓN	PRIORIDAD	PUNTOS
ARQUITECTURA ARQUETIPO	DISEÑO DE LA ARQUITECTURA	1	4
ESTRUCTURA DE PAQUETES	DISEÑO ESTRUCTURA DE PAQUETES	1	1
DISEÑO PANTALLAS	DISEÑO DE LA ESTRUCTURA QUE TENDRÁ LA FUTURA APLICACIÓN	2	10
MENÚS DEL APLICATIVO	DISEÑO MENÚS DEL APLICATIVO.	3	5
ESTILOS DEL APLICATIVO	DISEÑOS DE CSS	4	10
PRESENTACIÓN DEL APLICATIVO	PAGINA PRINCIPAL QUE INTEGRA, MENÚS CON LOS ESTILOS SELECCIONADOS. PRESENTANDO LAS PANTALLAS DEL FUTURO APLICATIVO	5	2
AUTENTICACIÓN DEL APLICATIVO	PAGINA AUTENTICACIÓN Y ACCESO ZONA PRIVADA.	6	8
REGISTRO	PAGINA REGISTRO	6	2
<b>TOTAL</b>			<b>41</b>

## DISEÑO DEL APlicativo ALPHA

NOMBRE HISTORIA (TAG)	DESCRIPCIÓN	PRIORIDAD	PUNTOS
GENERACIÓN DEL APlicativo CON EL ARQUETIPO	CREACIÓN APlicativo INICIAL	1	1
ENTRADA EN EL APlicativo	SISTEMA AUTENTICACIÓN Y REGISTRO	1	5
SALIDA DEL APlicativo	SALIDA DEL APlicativo	2	1
MENÚ DE OPERACIONES	DISEÑO DEL MENÚ DEL APlicativo	3	1
ALTA DOCUMENTOS	ALTA DOCUMENTOS	4	1
BAJA DOCUMENTOS	BAJA DOCUMENTOS	4	2
LISTADO DE DOCUMENTOS	LISTADO DE DOCUMENTOS	4	2
ACTUALIZACIÓN DOCUMENTOS	ACTUALIZACIÓN DOCUMENTOS	4	1
ALTA GRUPOS	ALTA GRUPOS	5	1
BAJA GRUPOS	BAJA GRUPOS	5	1
LISTADO GRUPOS	LISTADO GRUPOS	5	2
ACTUALIZACIÓN GRUPOS	ACTUALIZACIÓN GRUPOS	5	1
ACL SOBRE DOCUMENTOS	ACL SOBRE DOCUMENTOS	6	5
TOTAL			24



### 7.3.2. Aplicando XP y AMDD al arquetipo

#### 7.3.2.1. Arquitectura de las futuras aplicaciones diseñadas a partir del arquetipo.

En este punto vamos a ver las 2 tendencias actuales en el desarrollo de aplicaciones Web, J2EE, puro y la arquitectura de Spring. Y las razones por las que nos decantamos por el uso de esta última.

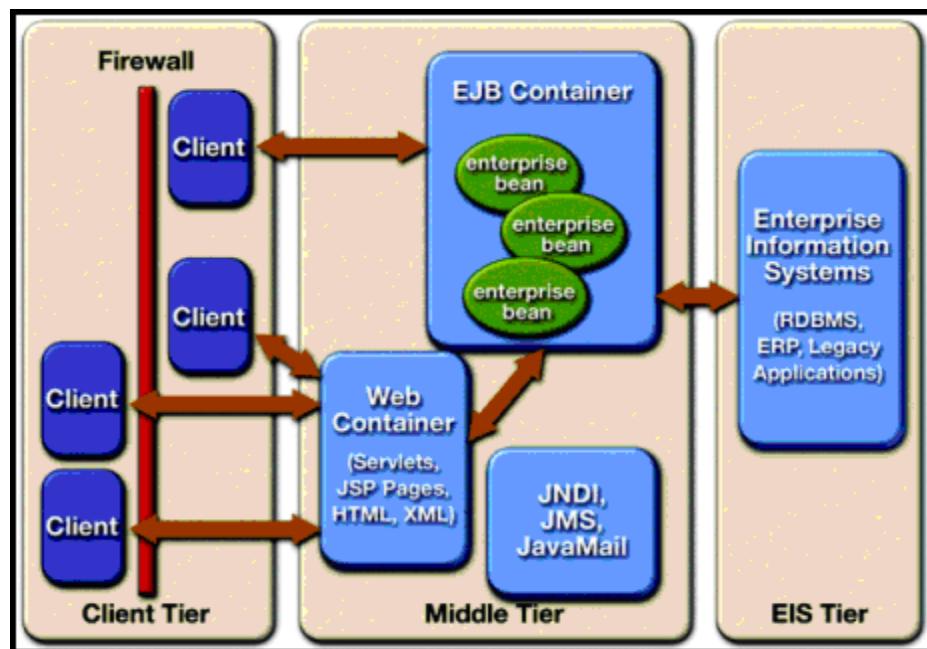


Figura 101 APPLICACIONES MEDIANTE J2EE

J2EE: la edición de Java para la construcción de aplicaciones empresariales.

J2EE no es un producto, sino un conjunto de especificaciones desarrolladas por el JCP (Java Community Process).

J2EE aporta:

- ü Separación entre capas.
- ü Transparencia de la ubicación.
- ü Manejo de transacciones: Pools de recursos (conexiones a bases de datos, objetos de negocio, etc.).
- ü Manejo de hilos de ejecución (threads).
- ü Seguridad.
- ü Etc.

La especificación de J2EE define 2 contenedores:

- ü • Web Container
- ü • EJB Container

Estos contenedores son los responsables de manejar los componentes correspondientes.

El **Web Container** maneja los componentes de presentación: *Servlets y páginas JSP, etc*

El **EJB Container** maneja componentes de lógica de negocio: *Enterprise JavaBeans*

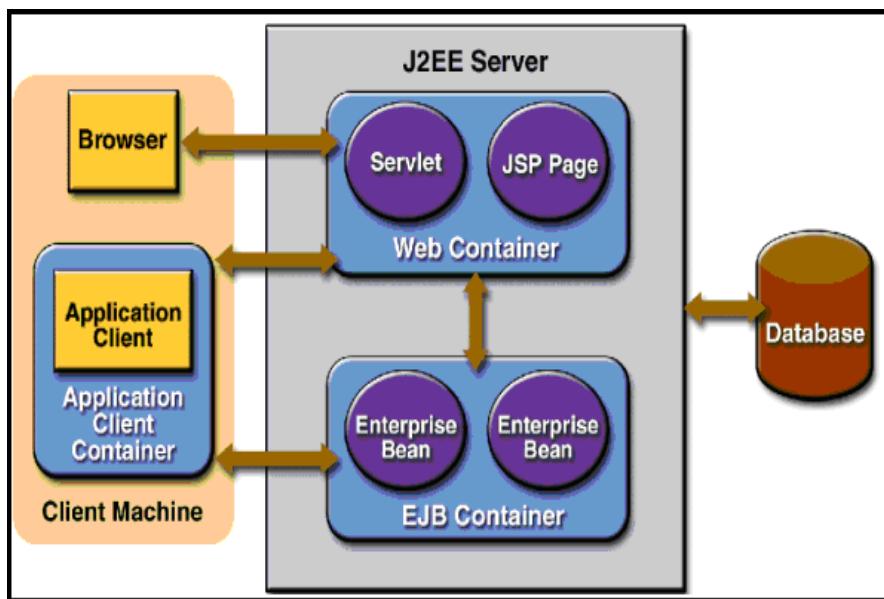
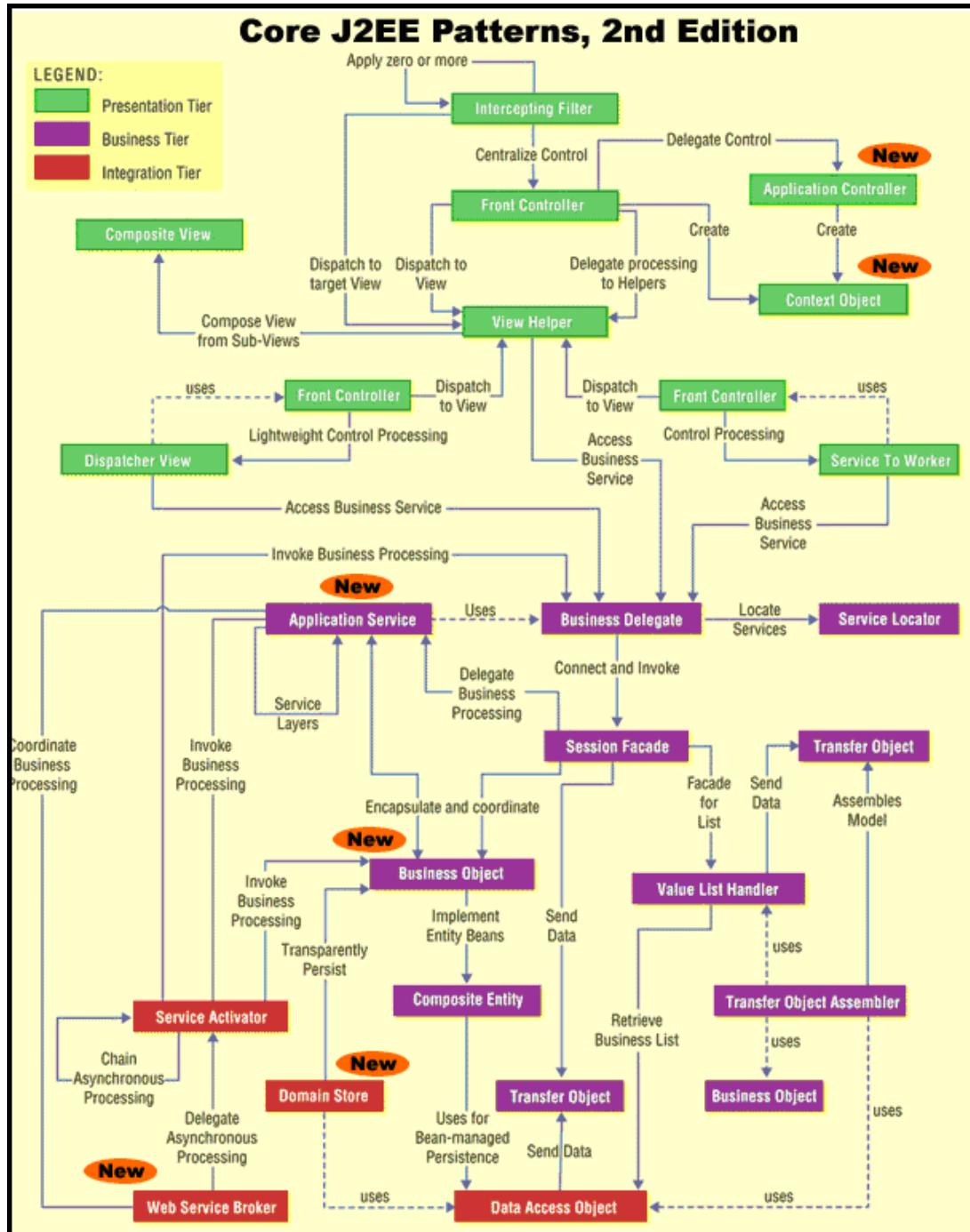


Figura 102 Esquema funcioneamiento J2EE

**J2EE Blueprints:** recomendaciones para el desarrollo de aplicaciones J2EE, provistas en el sitio Web de Java (<http://java.sun.com>)

*Blueprints* suministra un conjunto de patrones de diseño para la construcción de una aplicación web. Como se puede observar en la siguiente figura este diseño, es bastante complejo, lo que hace que la construcción de aplicaciones web sea difícil y complicado.

No necesitamos tanta complicación para el desarrollo de aplicaciones web. No necesitamos desarrollar completamente *Blueprints*. No todo son *EJB* para la lógica de negocio, existen alternativas más rápidas y ligeras.



### **Figura 103 Patrones de Diseño Blueprint**

## Spring Framework

También conocido simplemente como Spring, es un framework de código abierto para el desarrollo de aplicaciones Java.

**Spring Framework** se ha popularizado en la comunidad de programadores en Java al considerársela una alternativa y sustituto del modelo de *Enterprise JavaBean*. Por su diseño el *Framework*, ofrece mucha libertad a los Programadores en Java, así como soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria.

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

Spring nos aporta una gran variedad de controladores que nos permiten la utilización del patrón MVC.

- ü **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de éstos y permite derivar nuevos datos.
- ü **Vista:** Este presenta el modelo en un formato adecuado para interactuar con él, usualmente la interfaz de usuario.
- ü **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

La arquitectura elegida para el aplicativo será los MVC suministrados con *Spring* más la capa de servicios que nos suministra acceso a la lógica de negocio del aplicativo.

A su vez utilizaremos una capa de persistencia suministrada por *Hibernate* a través de los paquetes que suministra *Spring* el *SpringORM*.

Esta arquitectura queda más claro viendo las 3 figuras siguientes, en las que se pone de manifiesto las separaciones en capas comentadas.

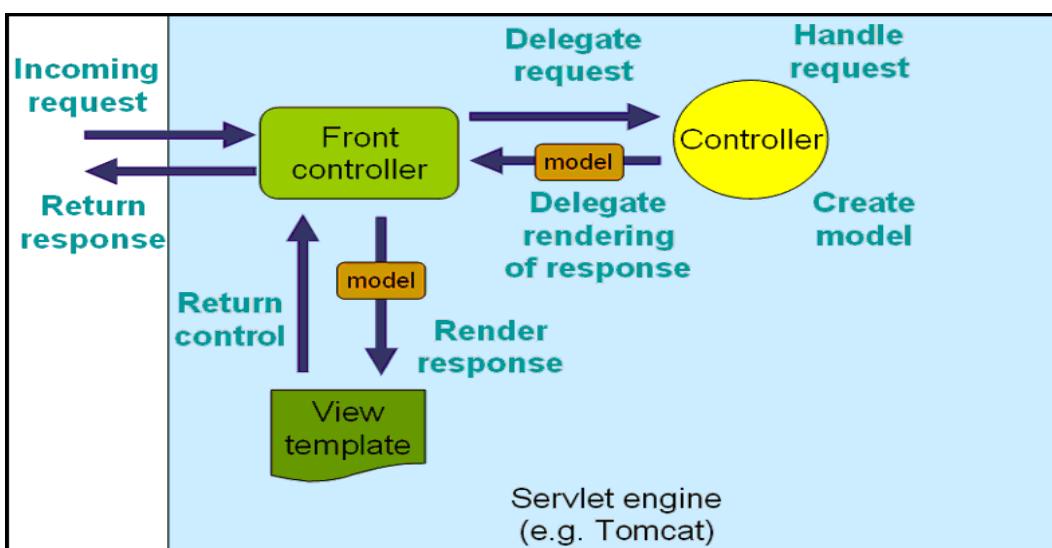


Figura 104 MVC DE SPRING



La arquitectura está formada por 3 capas principales.

**La capa de presentación:** En esta capa se usará JSP y JSTL, y la pieza más importante del mismo es su fichero de configuración. En él tendrá que declararse los beans de vista como la navegabilidad de la aplicación

**La capa de negocio:** Es de las capas más importantes de toda la aplicación ya que alberga la lógica de negocio de la misma. La lógica de negocio, tendrán que ser declarados en el fichero de configuración de Spring, `applicationContext.xml`.

**La capa de acceso a datos:** *Data Access Object* (DAO, Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, como una Base de datos.

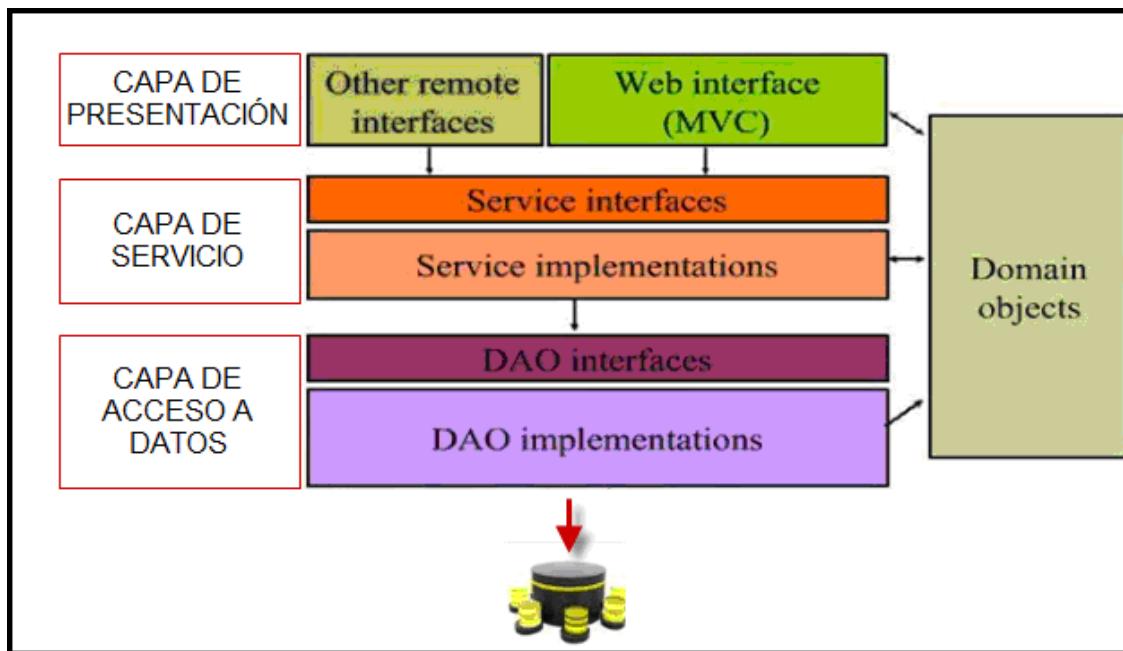
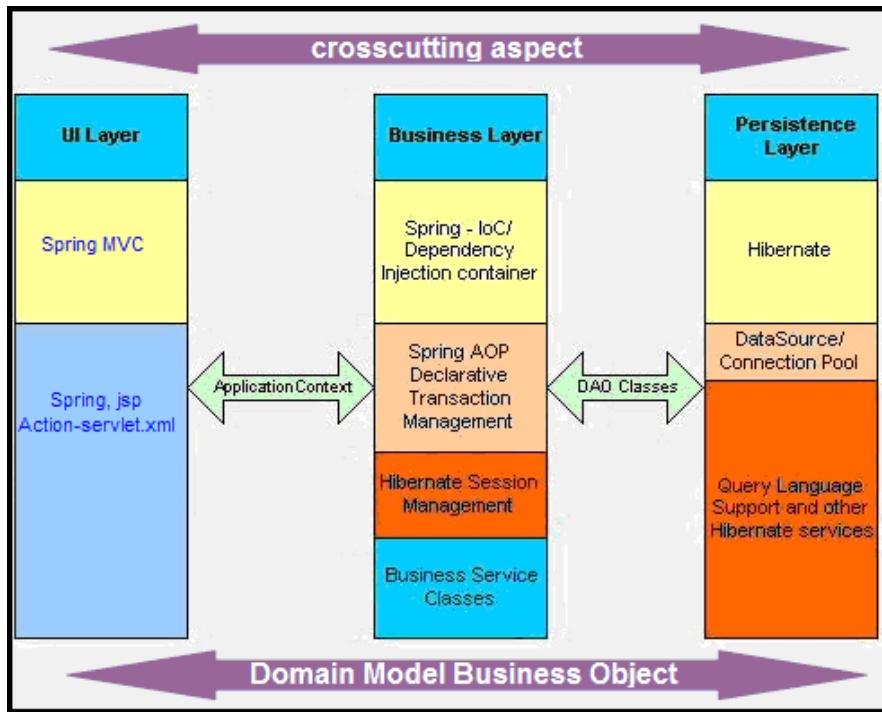


Figura 105 Modelo de capas de Aplicaciones con Spring



**Figura 106 Modelo de Capas Spring - Hibernate**

Para implementar estas clases de modelo de negocio haremos uso de las clases Java conocidas como *clases POJOs (Plain Old Java Objects)*. Estas clases no son nada más que Java Beans.

Un *JavaBean* o Bean es un componente hecho en software que se puede reutilizar y que puede ser manipulado visualmente por una herramienta de programación en lenguaje Java.

Las propiedades de un Bean pueden examinarse y modificarse mediante métodos o funciones miembro, que acceden a dicha propiedad, y pueden ser de dos tipos:

- ü **getter**: lee el valor de la propiedad
- ü **setter**: cambia el valor de la propiedad.

*Hibernate* permite establecer relaciones entre clases Java y tablas de una base de datos. Para ello hace uso de ficheros XML en los que se describe dicha relación que indicará a *Hibernate* como persistir las clases Java



### 7.3.2.2. Estructura del arquetipo.

A continuación vamos a mostrar la arquitectura del arquetipo y de las futuras aplicaciones desarrolladas con él.

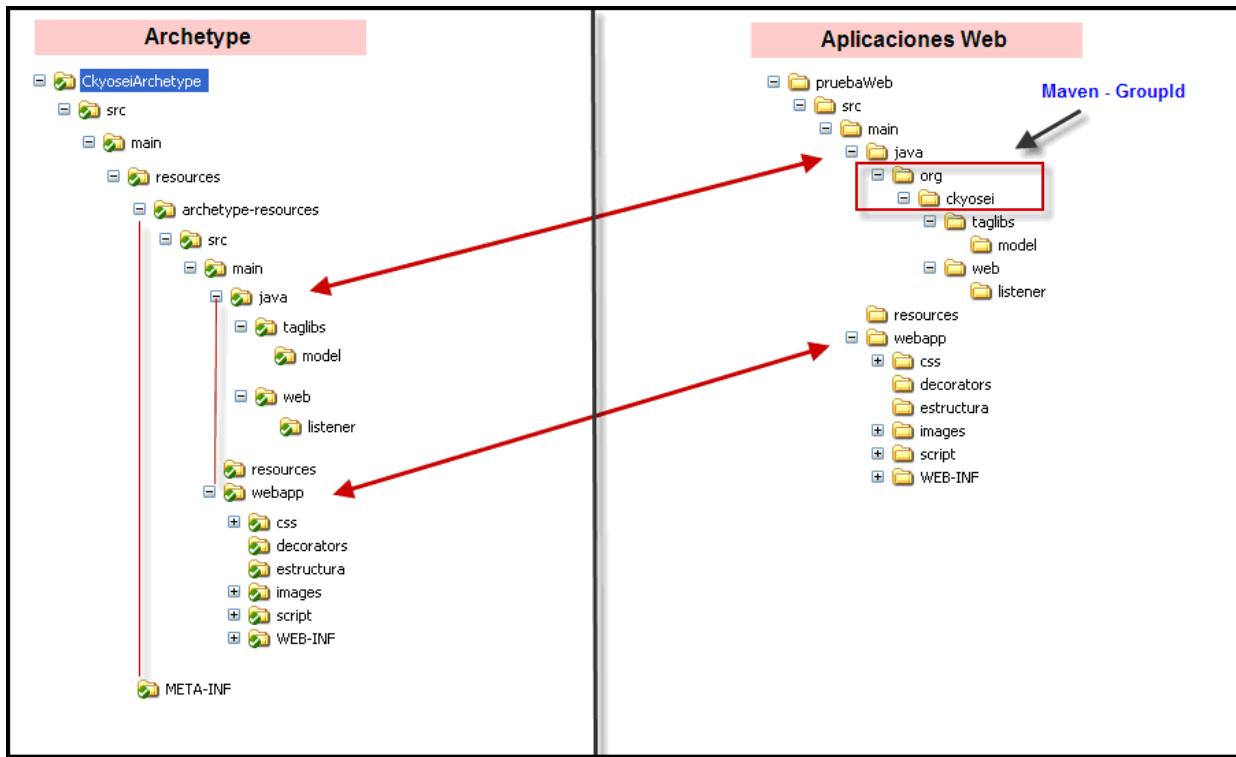


Figura 107 Estructura del arquetipo Ckyosei

Las aplicaciones Web generadas tendrán siempre la estructura siguiente:

La carpeta **src/main/java** directorio de código fuente.

La carpeta **src/main/resources** directorio donde se colocarán los recursos necesarios para ser incluidos en el classpath del aplicativo, de los cuales hará uso el código fuente. Por ejemplo **log4j.properties**, ficheros de internacionalización **I18N.properties** etc.

El arquetipo no generará explícitamente carpetas para el código de los pruebas o test así como para los recursos de los mismos. Será función del programador crearlos ambos.

Para mantener la estructura de **Maven** de directorios la ruta donde se deben crear será:

La carpeta **src/test/java** directorio de código fuente donde colocaremos los test de **junit**, **mock**, etc.

La carpeta **src/test/resources** directorio de recursos necesarios para la ejecución de los test de junit, mock, etc.

Tampoco se generará por defecto la carpeta para la creación del **Site** o documentación del proyecto, puesto que se desconoce la estructura del mismo.

Esta carpeta en caso de querer generar documentación mediante **Maven**, se creará en la estructura **src/site** donde incluiremos el descriptor del site **site.xml** y las plantillas apt para la generación de la documentación, necesarias para el plugin de **Maven**.

Para más información de la generación de documentación por medio de **Maven** ver las siguientes referencias:

Para más información de la generación de documentación por medio de Maven ver las siguientes referencias:

<http://Maven.apache.org/plugins/Maven-site-plugin/> documentación del plugin site de Maven 2.0

<http://Maven.apache.org/guides/mini/guide-apt-format.html>. El formato APT de Maven.

Organización de las carpetas de contenido Web (**webapp**).

Esta carpeta esta organizada en 2 partes fundamentales como todas las aplicaciones Web:

- ü Contenidos públicos.
- ü Contenidos privados.

Los contenidos privados estarán situados bajo la carpeta **Web-inf**. Dentro de ella se situarán los ficheros de configuración de **Spring** y el descriptor de despliegue de la aplicación web.

También situaremos a este nivel la carpeta de vistas **jsp**. Al hacer esta operación haremos que ninguna vista pueda ser invocada directamente a través del navegador.

Dentro de esta carpeta se situarán como en todas las aplicaciones Web, las librerías (**lib**) y el directorio de clase compiladas.

La parte pública estará organizada de la siguiente forma.

Carpeta **CSS** dentro de esta carpeta situaremos los estilos de la aplicación.

Carpeta **decorators** dentro de esta carpeta situaremos los decoradores de **Sitemesh**

Estos decoradores no son nada más que archivos JSP con la estructura de las capas que contendrán las páginas a las cuales aplicaremos el decorador.

La información sobre esta carpetas se verá más adelante.

Carpeta **script** dentro de esta carpeta situaremos los javascript de la aplicación.



Carpeta **estructura** dentro de esta carpeta situaremos los jsps comunes a todas las páginas. Cabeceras, metas, pie de página. Estas JSPs que llamaré de estructura serán incluidas en las JSPs de los distintos decoradores de la carpeta decorators

Carpeta **images** dentro de esta carpeta situaremos las imágenes de la aplicación.

### 7.3.2.3. Modelo de dominio

Un arquetipo no es más que una “plantilla” a partir de la cual generaremos un aplicativo. Por lo que el modelo de dominio de todo el arquetipo no tiene sentido.

Lo que vamos a ver en este punto, son los modelos de dominio comunes a todas las aplicaciones que se desarrollen a partir del arquetipo. Estos modelos de dominio comunes, son los menús y la gestión de usuarios (autenticación y registro)..

La siguiente figura muestra el modelo de dominio de la creación de los diferentes menús del un aplicativo.

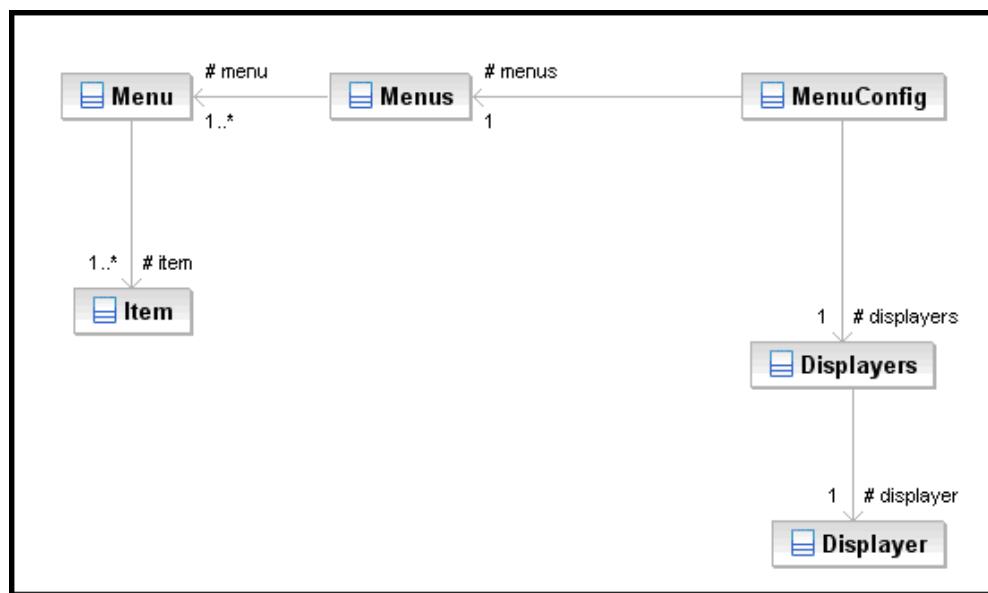


Figura 108 Modelo de dominio de Menús

Un menú, va a estar estructurado de la siguiente manera.

Una Objeto de configuración **MenuConfig**, podrá tener un objeto **Menus** (Menus), y un visualizador (**Displayers**).

A su vez el objeto “Menús” puede tener de 1.\* Objetos “Menú”. (Menús individuales).

Cada objeto Menú contendrá 0.\* objetos **Item**.

Estos menús podrán ser verticales como horizontales.

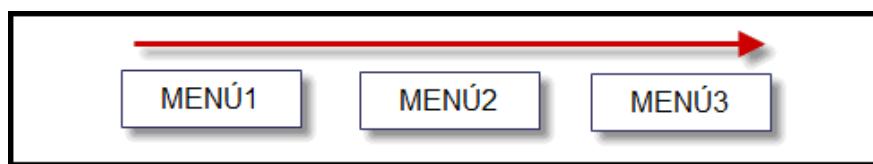


Figura 109 Estructura de menús horizontales usados para navegación global

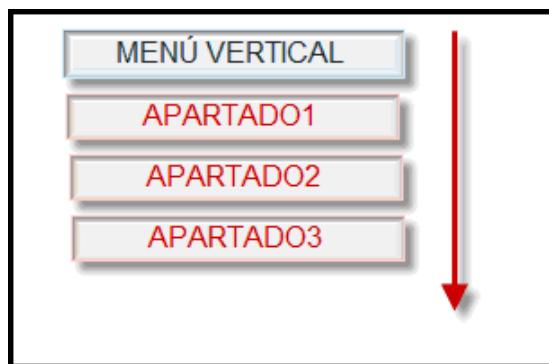


Figura 110 Estructura de menús verticales usados para navegación local

Para la autenticación vamos a usar los modelos de dominio que proporciona **Spring-security**.

Este framework de seguridad permite la autenticación de un usuario con su Rol.

O la autenticación de Usuario, por pertenencia a un grupo, con un determinado Rol. Según el siguiente Modelo.

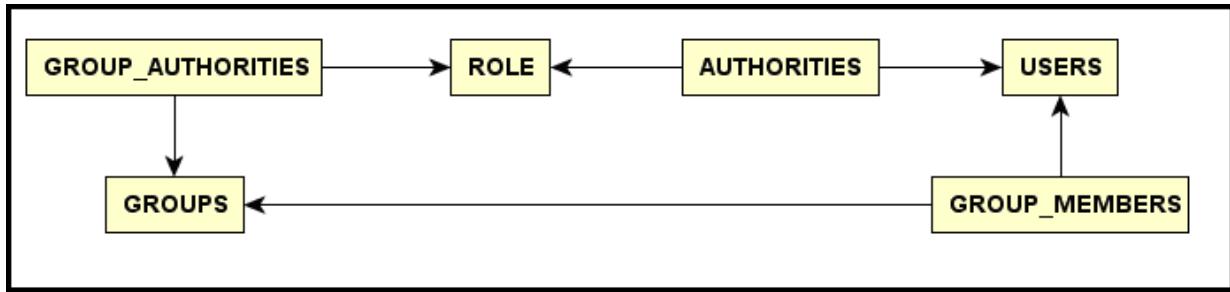


Figura 111 Modelo lógico de autenticación de Spring security

**Users**: almacenará la información relativa al usuario.

**Role**: almacenará los Roles definidos en el aplicativo.

**Authorities**: establecerá la relación entre usuarios y roles.

**Groups**: Almacenará los grupos del aplicativo.

**Groups\_Members**: Almacenará la relación usuarios y su pertenencia a un grupo.

**Group\_Authorities**: Establece un Rol para un grupo.

En este proyecto se ha elegido la implementación sencilla de Usuario / Rol. Por lo que el objeto de dominio quedará según la siguiente figura.



Figura 112 Modelo de dominio de Autenticación

### 7.3.2.4. Prototipos de interfaces de usuario

Las pantallas del futuro aplicativo tendrán el siguiente aspecto inicialmente.

Estarán compuestas básicamente de una cabecera, un rastro de migas, un pie y 2 menús de navegación, contenido, y zona de información. El menú izquierdo es el menú de navegación local para cada aplicativo. Siendo el menú horizontal (parte superior), el acceso a otras secciones o aplicativos del sistema.

Todas las capas pueden ser configuradas dependiendo de la estructura de la página visualizada. Pudiéndose ocultar dependiendo del tipo de presentación deseado.

## Pantalla principal.



Figura 113 Prototipo de la pantalla principal de las futuras aplicaciones

El cuerpo de las futuras páginas de información podrá estar distribuido en 1 o 2 columnas. Pero respetará la estructura de menús, para mantener la homogeneidad de los aplicativos permitiendo que no se pierdan las personas que naveguen por ellos.

## Pantalla de Identificación y Registro

La pantalla de autenticación no posee ningún menú, ni de navegación global, ni de navegación local al aplicativo. También prescindiremos del apartado de información del lado izquierdo de la página.



**Figura 114 Pantalla de Identificación y Registro**

### Estructura de las pantallas

La estructura de pantallas está formada a partir de la combinación del framework Sitemesh y de una adaptación del framework de CSS de Mike Stenhouse.

El framework CSS estructura la forma en que se compondrán las páginas definiendo varios ficheros. Uno de capas, otro de fuentes, formularios, mensajes, etc. Estos ficheros se combinan mediante importaciones en un archivo, Conformando un Tema. Archivo **theme.css**

Un tema estará compuesto por una estructura de capas, intercambiables entre sí para conformar las páginas. De una sola columna, de 2 columnas. Con rastro de Migas, Sin rastro. Con navegación global, etc.

Cada página decidirá la estructura interna que tiene sobreescribiendo las capas definidas en **layout.css**.

### Estructura del archivo de estilos **theme.css**

“

```
@import url("typo.CSS");  
@import url("layout.CSS");  
@import url("displaytag.CSS");  
@import url("form.CSS");  
@import url("message.CSS");
```

”

Esta forma de trabajo, nos permitirá cambiar toda la apariencia de una aplicación simplemente importando otro tema distinto, al actual.

“

```
<link rel="stylesheet" href='${ctx}/CSS/${theme}/theme.CSS'  
type="text/CSS" />
```

”

**\${theme}** Parámetro de contexto definido en el descriptor de despliegue del aplicativo.

A continuación se muestran algunas partes básicas de la estructura del aplicativo, como el contenedor, contenido, cabeceras, etc.

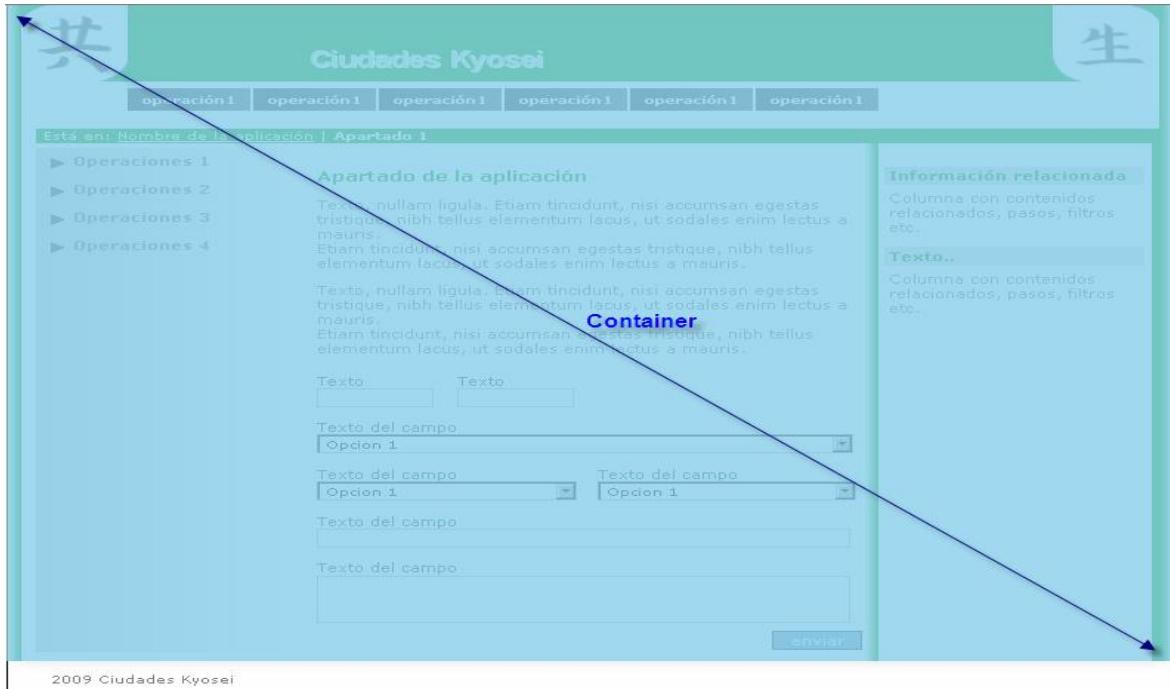


Figura 115 Estructura de capas del CSS - Contenedor Global

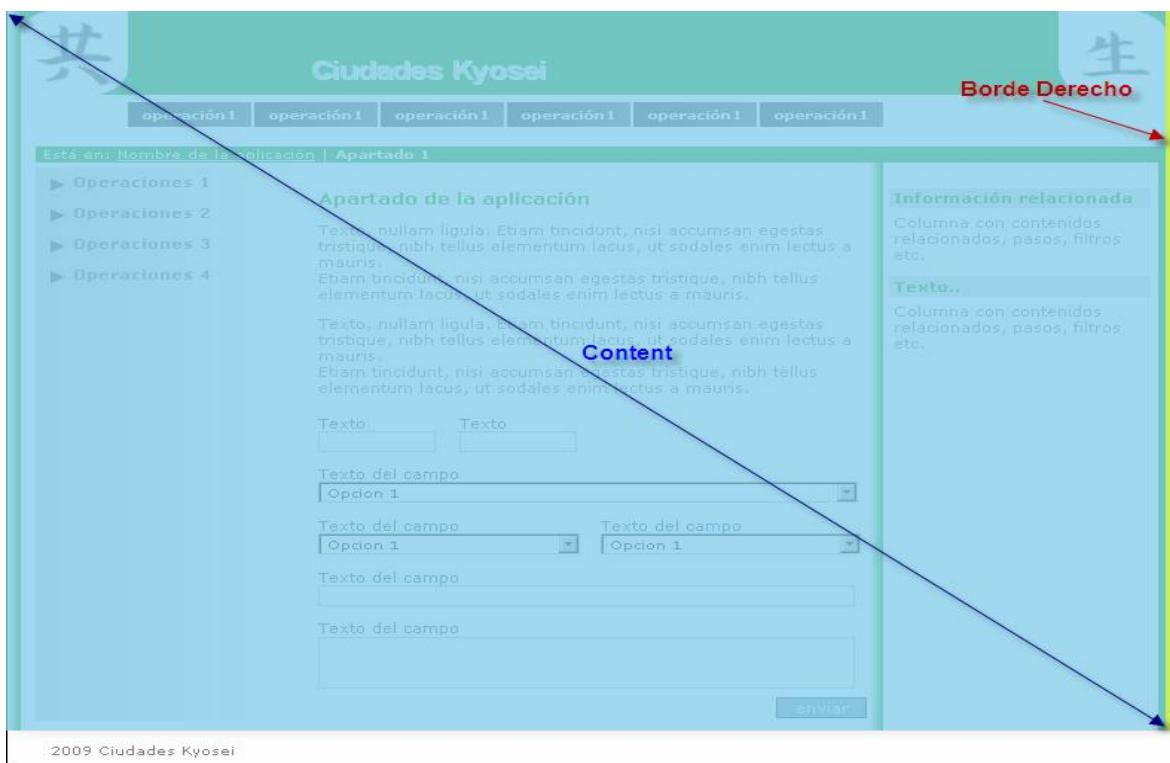


Figura 116 Estructura de capas del CSS - Contenedor Global

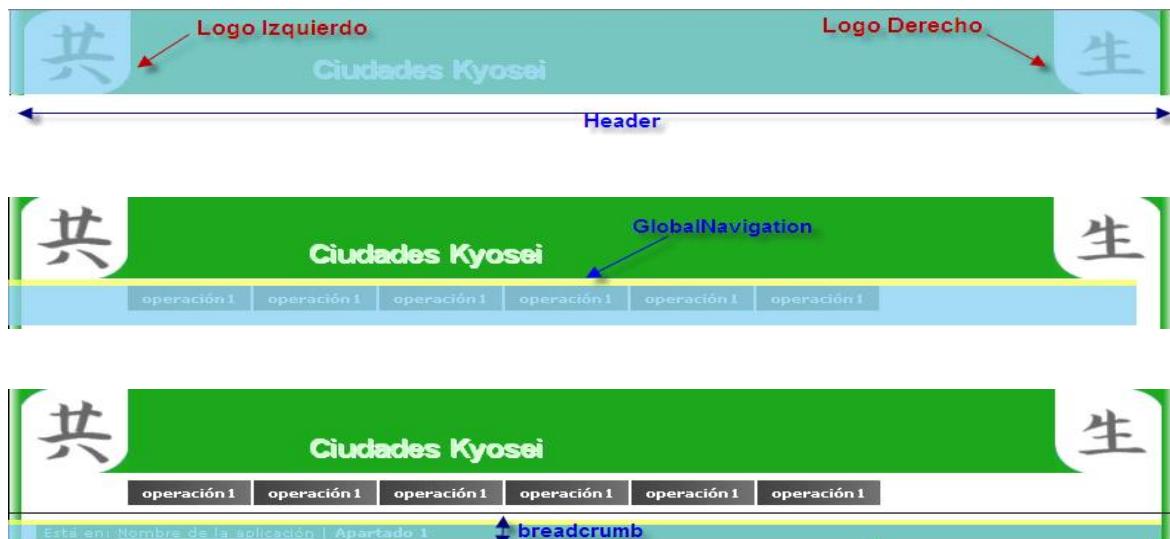


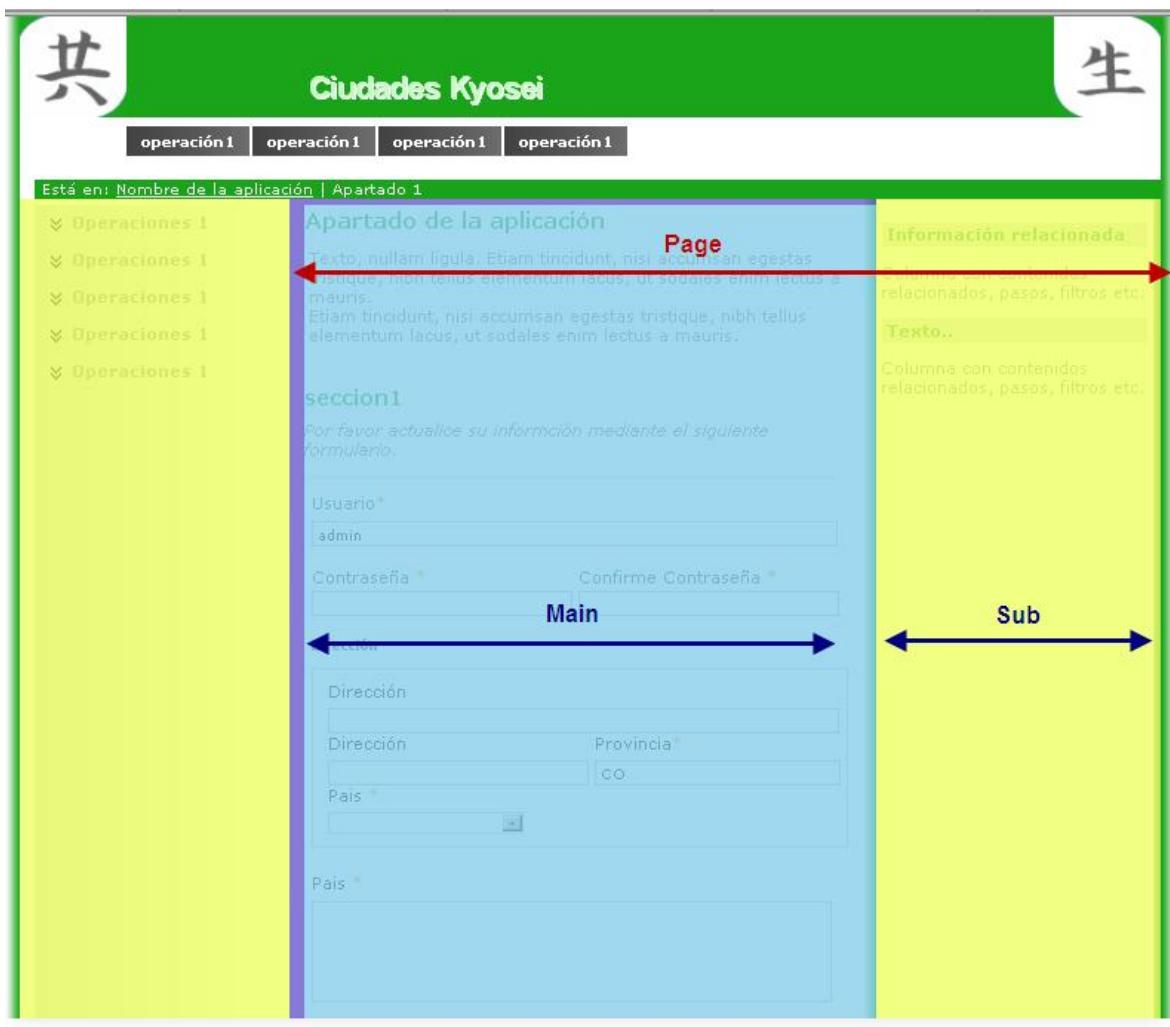
Figura 117 Estructura de capas del CSS - Cabeceras, Navegación Global y rastro de migas

Este diagrama ilustra la jerarquía de capas del CSS para el contenido de la página:

- Apartado de la aplicación:** La sección principal que contiene texto y campos de formulario.
- Información relacionada:** Una columna lateral que incluye un encabezado y un apartado "Texto".
- Botones:** Un grupo de botones de acción ubicados en la parte inferior derecha de la sección principal.

2009 Ciudades Kyosei

Figura 118 Estructura de capas del CSS - Contenido



**Figura 119 Estructura de capas del CSS - Contenido en detalle**

A continuación se muestra el funcionamiento de Sitemesh para la estructura anterior.

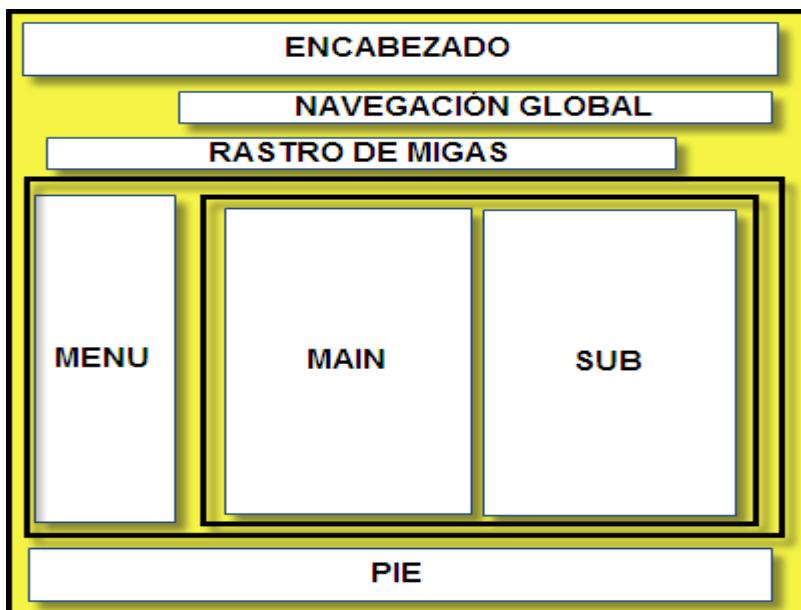


Figura 120 Esquema estructura global de capas del aplicativo

Partimos de un modelo de capas inicial como el de la figura siguiente:

La invocación de la página inicial **index.do**, invocará al controlador (MVC) que nos devolverá la vista **index.jsp**.

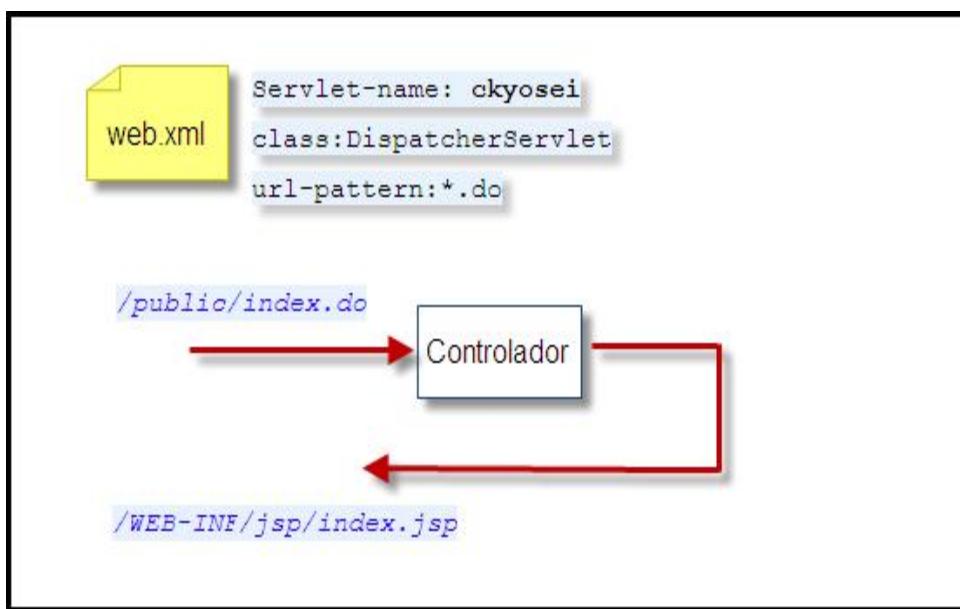


Figura 121 Estructura Modelo Vista Controlador



La vista **index.jsp** será una combinación de una plantilla + el contenido de la propia vista.

Generándose la respuesta dinámicamente.

Pare realizar esta operación o se hace uso de un filtro (**Servlet Filter**) para agregar capas a las páginas de la aplicación. Este filtro es proporcionado por la librería Open Source **Sitemesh**. El filtro intercepta todas las peticiones que regresan contenido HTML y las decora.

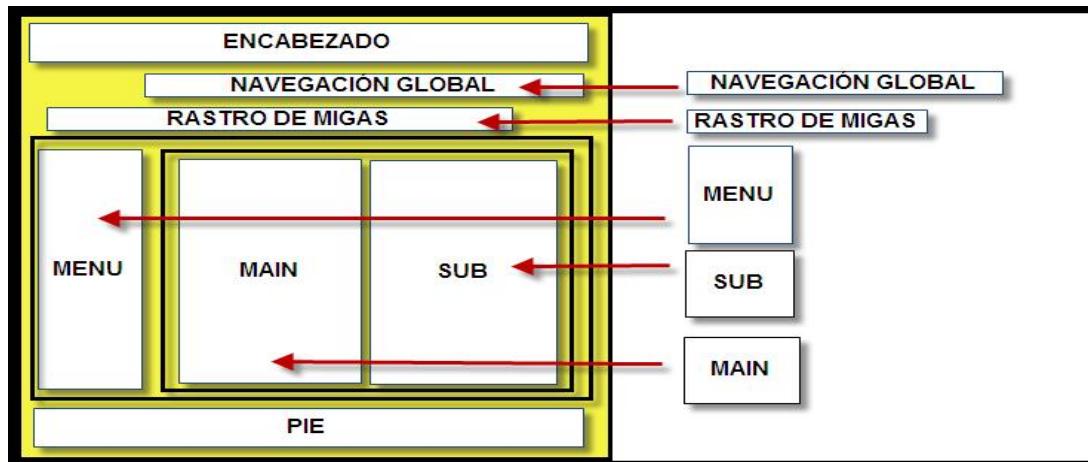


Figura 122 Esquema estructura global de capas del aplicativo Usando Sitemesh

En la figura siguiente se muestra en detalle el proceso.

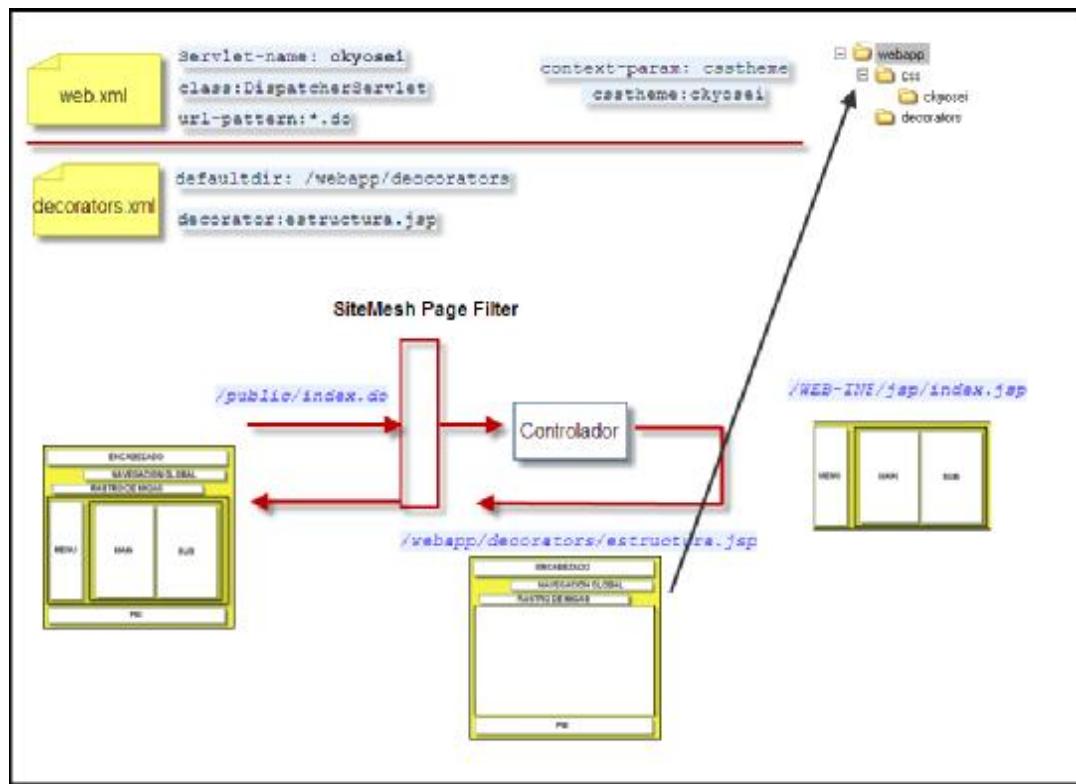


Figura 123 Funcionamiento en detalle de Sitemesh

Como se observa en las vistas solo se modifican los elementos concretos que queramos incorporar a la página resultante. Los elementos comunes cabeceras, pies, etc. los añade la plantilla o decorador.

El tema de la hoja de estilos es seleccionado en un parámetro de contexto del descriptor de despliegue del aplicativo Web, pudiéndose fácilmente intercambiar por otro valor simplemente cambiando el parámetro y reiniciando el aplicativo.



La forma de trabajar entonces será la siguiente:

1. En la plantilla o decorador incluiremos el tema que se desee usar.



```
<link rel="stylesheet" href='${ctx}/CSS/${theme}/theme.CSS' type="text/CSS" />
```



Este tema añadirá los layout por defecto que compondrán la página, los estilos de las fuentes, (**typo.CSS**). Formularios (**form.CSS**) etc.

Y realizaremos la llamada para incluir los contenidos que conformarán la pagina mediante el tag **<decorator:getProperty>**



```
<decorator:getProperty property="page.layout" />
```



Este tag nos permitirá incluir el contenido definido en un tag denominado por ejemplo “layout” de las páginas JSP.

Mediente este tag incorporaremos los layout de las páginas JSP concretas así como los contenidos.

En las páginas JSP concretas los estilos y contenidos de la página.



```
<content tag="layout">  
    <link rel="stylesheet"  
        href='${ctx}/CSS/${theme}/layout-navtop-localleft.CSS'  
        type="text/CSS" />  
</content>
```



La forma de definirlos es con un tag “content” con el nombre de la propiedad usada en la plantilla.



### 7.3.2.5. Storyboard

Mediante el diagrama de flujo o *StoryBoard*, presentaremos la navegación del aplicativo.

En este caso vamos a presentar simplemente la navegación inicial que tendrá al crearse el aplicativo. Esta navegación será ampliada al desarrollarse el mismo.

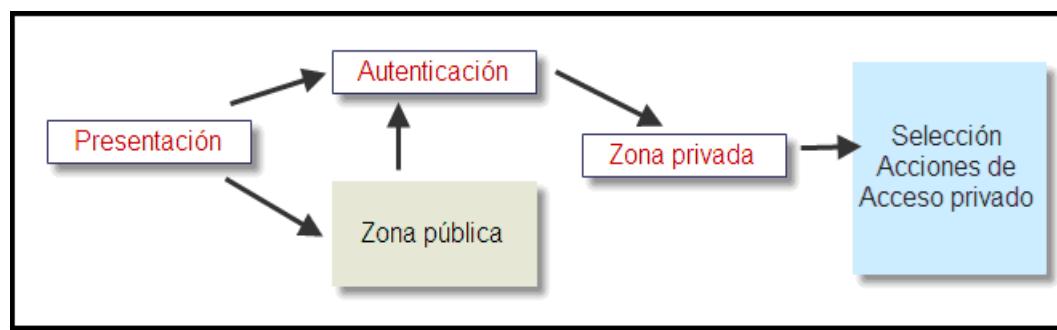


Figura 124 Storyboard

Al invocar la página principal o **index.do**, se mostrará la página de presentación del aplicativo. Desde esta página se podrá acceder a las diversas funcionalidades que se quieran incluir en los menús global o local del mismo.

Dependiendo del grado de privacidad de los mismos, los contenidos que se presenten podrán ser públicos o privados. Para acceder a estos últimos el usuario deberá validarse a través de la pantalla de autenticación.

#### Plan de Entregas del Arquetipo

Iteración	Funcionalidad	FECHA
0	Instalación entorno de trabajo. JDK, Eclipse, Plugin eclipse, Servidores Apache, Tomcat, Conector, Base datos.	Por definir
1	Pegueña release con prioridad 1, 2, 3 historias de usuario	Por definir
2	Pegueña release con prioridad 4	Por definir
3	Pegueña release con prioridad 4	Por definir

4	Pegueña release con prioridad 5	Por definir
5	Pegueña release con prioridad 6	Por definir

### 7.3.2.6. Enfoque y diseño del Artefacto

La figura siguiente muestra algunos de los posibles artefactos que pueden producir en la liberación de una versión o a nivel de cada iteración. Estos artefactos no son obligatorios para cada proyecto, de manera que dependerá del proyecto su generación dado que en muchos casos no tendrán alguno sentido.

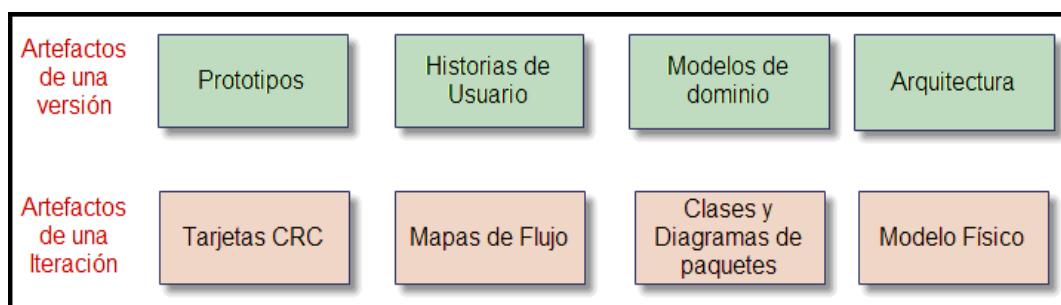


Figura 125 Artefactos y iteraciones

### 7.3.2.7. Arquitectura

A continuación se muestra la arquitectura de alto nivel del aplicativo que generará el arquetipo. La arquitectura es bastante sencilla. Aplicaremos el estándar de arquitectura Web de tres niveles. El nivel cliente (navegador web), El nivel medio (servidor de aplicaciones), y el nivel de datos (base de datos).

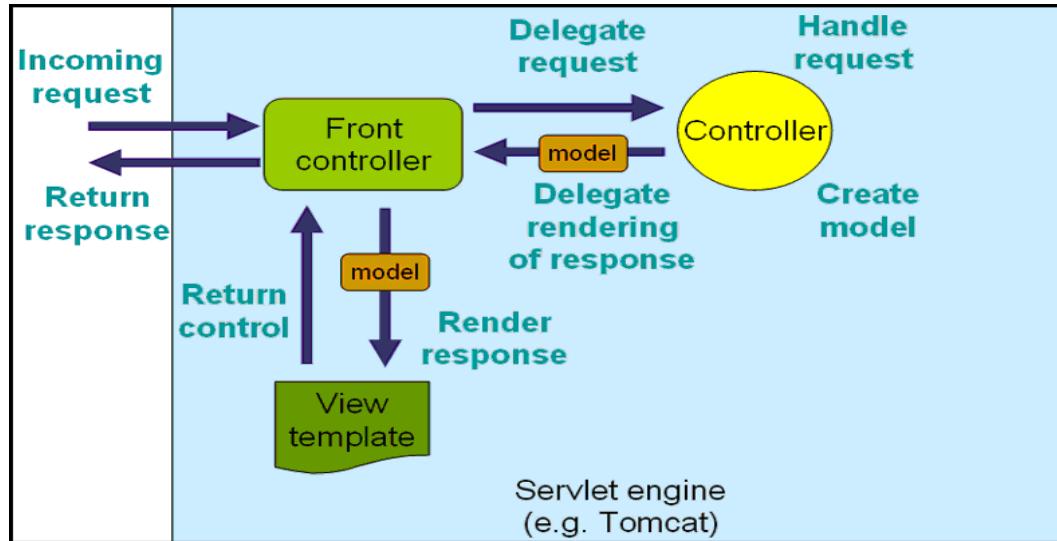


Figura 126 MVC DE SPRING

Spring tiene implementado un servlet que realiza las tareas de frontal (Front Controller), esto significa que cada uno de las peticiones (request) que son realizados por el usuario, pasan a través de este servlet. El nombre que recibe este servlet es **Dispatcher Servlet** ).

La petición (Request) llega al **Dispatcher** el cual tiene la responsabilidad de delegar a otro componente el procesamiento de la misma (Controller o Controlador).

Para obtener el nombre del componente que recibirá la petición, Spring utiliza lo que se denomina el **Handler Mapping**.

El **Handler Mapping** tiene el objetivo de indicar al **Dispatcher** cual será el controlador que debe recibir la petición enviada por el usuario.

<b>Handler Mapping</b>	<b>Descripción</b>
BeanNameUrlHandlerMapping	Mapea controladores a URL basándose en el nombre del Bean
SimpleUrlHandlerMapping	Mapea controladores a URL basándose en una colección de propiedades que se definen en el Spring application context.
ControllerClassNameHandlerMapping	Mapea controladores a URL utilizando el nombre de la clase del controlador.
CommonsPathMapHandlerMapping	Mapea controladores a URL usando metadatos en el código del controlador. Esta metadata es definida usando Jakarta Commons Atributes.

Después de que el **Handler Mapping** le entrega el nombre del Controlador que se encargará de la petición, el Dispatcher Servlet le envía la petición (Request) al Controller.

Para poder implementar un Controlador sobre Spring es necesario que se cree una clase que herede de los Controladores ya implementados por Spring.



Tipo de Controlador	Clase Implementación	Descripción
View	ParameterizableViewController UrlFilenameViewController	Cuando un controlador solo necesita desplegar información
Simple	Controller (interface) AbstractController	Para controladores simples que sólo se utilizan como Simples Servlet
Multiaction	MultiActionController	Para implementar una serie de acciones con similar lógica
Command	BaseCommandController AbstractCommandController	Los controladores reciben parámetros, estos son manejados dentro de un objeto
Throwaway	ThrowawayController	Para manejar los request como un Comando
Form	AbstractFormController SimpleFormController	Permite desplegar y procesar un formulario, bajo el mismo componente
Wizard	AbstractWizardFormController	Para realizar formularios guiados

Una vez que el **Controller** recibe la petición, se construye un Objeto que se denomina **ModelAndView**. Este objeto se encargará de poner nombre a la Vista y al Modelo asociado a la misma. Además injectará el objeto Model que tiene los datos que serán presentados por la vista.

Una vez terminado, el objeto **ModelAndView** es devuelto al **Dispatcher** y este componente delega la responsabilidad de elegir la vista a utilizar al componente **ViewResolver**.

El **ViewResolver** es el encargado de realizar el mapeo entre el nombre lógico de la vista y el componente.

<b>Handler Mapping</b>	<b>Descripción</b>
InternalResourceViewResolver	Resuelve el nombre lógico utilizando el mapeo para Velocity o JSP
BeanNameViewResolver	Resuelve el nombre lógico utilizando Bean definidos en el Spring Context
ResourceBundleViewResolver	Define el mapping entre los nombres lógicos y las vistas asociadas, definiéndolo en un archivo de propiedades
XmlViewResolver	Define el mapping entre los nombres lógicos y las vistas asociadas, definiéndolo en un archivo XML

Una vez que la vista realiza el procesamiento, el Dispatcher devuelve la respuesta de retorno al usuario.

Por lo tanto para configurar Spring MVC hay que seguir los siguientes pasos:

1. Colocar las librerías necesarias.
2. Configurar al **DispatcherServlet** como **FrontController**.

<Bucle\_inicio>

Escribir un controlador y agregarlo al contexto del **DispatcherServlet**.

Escribir una vista (JSP u otro formato).

<Bucle\_Fin>

3. Configurar un **Handler Mapping** para asociar las URLs a los controladores.
4. Configurar un **View Resolver** para asociar las vistas a los controladores.



### 7.3.2.8. CRC Cards

**CRC (Class, Responsibilities, and Collaborators)**, método que se basa en el uso de tarjetas. Un método muy práctico para definir las clases de un sistema y la interacción que hay entre ellas.

La estructura que vamos a presentar en las tarjetas será la siguiente:

Nombre de la Clase	
Responsabilidades (obligaciones de esta clase, tales como los métodos de negocio, manejo de excepciones, métodos de seguridad, los atributos y variables)	Colaboradores (otras clases necesarias para proporcionar una solución completa)

**Tarjetas CRC para la creación de un Menú.**

#### CRC0

MenuConfig	
MenuConfig, clase encargada de guardar la configuración del menú.	Menus y Displayers. Que contienen los diferentes menús, y los diferentes visualizadores.

#### CRC1

Menus	
Menus, Clase contenedora de todos el conjunto de menús del aplicativo.	Menu. Contiene un menú individual

## CRC2

<b>Menu</b>	
Menus, Clase contenedora un menú con sus diferentes opciones (ítems)	Item. Apartado de un menú.

## CRC3

<b>Displayers</b>	
Conjunto de visualizadores de un menú. Reservado para usos futuros	<b>Display. Visualizador del menú.</b>

## CRC4

<b>Display</b>	
Cada Visualizador de un menú. Reservado para usos futuros	

**Taglibs desarrollados para la creación de los menús.**

## CRC5

<b>ConfigMenuTag</b>	
Se encarga de leer el fichero de menús y cargarlos en la configuración, y de configurar la seguridad de los menús.	Heredará de la clase TagSupport del paquete javax.servlet.jsp.tagext.TagSupport que permite la creación de Taglibs



## CRC6

<b>MenuTag</b>	
Se encarga de dibujar cada uno de los menús dependiendo de su configuración horizontal, vertical.	Heredará de la clase TagSupport del paquete javax.servlet.jsp.tagext.TagSupport que permite la creación de Taglibs

**Tarjetas CRC para la autenticación de usuarios.**

## CRC7

<b>Users</b>	
Clase que contendrá los datos de autenticación de usuarios necesarios de Spring-security.	Userdetail. Interface de Spring-Security que permite la personalización de la autenticación.

## CRC8

<b>Role</b>	
Clase que contendrá la información sobre el Rol de un usuario dentro del aplicativo.	GrantedAuthority: Interface de Spring-security para la personalización de las entidades de autorización.

## CRC9

<b>ROLEDAO</b>	
INTERFAZ PARA EL DAO DE ROLES.	ROLE. CLASE PARA ALMACENAR LA INFORMACIÓN DE UN ROL.

## CRC 10

<b>USERDAO</b>	
INTERFAZ PARA EL DAO DE USUARIOS.	USERS. CLASE PARA ALMACENAR LA INFORMACIÓN DE UN USUARIO.

## CRC 11

<b>UserDaoHibernate</b>	
Implementación del interfaz UserDao, usando el Framework de persistencia de Hibernate.	Users. Clase para almacenar la información de un Usuario. UserDao Interfaz para el DAO de Usuarios.

## CRC 12

<b>RoleDaoHibernate</b>	
Implementación del interfaz RoleDao, usando el Framework de persistencia de Hibernate.	Role. Clase para almacenar la información de un Usuario. RoleDao Interfaz para el DAO de Roles.

## CRC 13

<b>userManager</b>	
Interfaz para la lógica de negocio de usuarios.	Users. Clase para almacenar la información de un Usuario.

## CRC 14



## 7. Aplicando XP y AMDD a la aplicación alpha.

<b>UserManagerImpl</b>	
Implementación de la lógica de negocio de la gestión de usuarios.	Users. Clase para almacenar la información de un Usuario. UserDao. Interfaz para el DAO de Usuarios.

### CRC 15

<b>RoleManager</b>	
Interfaz para la lógica de negocio de roles	Role. Clase para almacenar la información de un Rol.

### CRC 16

<b>RoleManagerImpl</b>	
Implementación de la lógica de negocio de la gestión de usuarios	Role. Clase para almacenar la información de un Rol. RoleDao Interfaz para el DAO de Roles.

Taglibs desarrollados para la creación de los controladores MVC.

### CRC 17

<b>Controlador</b>	
Controlador que presentará la página principal	

## CRC 18

MenuController	
Controlador que pinta la página principal de la zona de acceso restringido	

## CRC 19

IncripcionController	
Controlador de Registro en el aplicativo	

### 7.3.2.9. Mapa de flujo Aplicación Inicial generada con el Arquetipo

En la siguiente tabla se muestra la relación entre las historias de usuario, CRC.

Story Tag	VIEW	Controller Class	COLLABORATORS
Presentación	Index	Controlador	----
Autenticación del aplicativo	Menu	MenuController	
Registro	Registro	IncripcionController	

### 7.3.2.10. Diagramas de Clase UML

A continuación se presentan los diagramas UML del arquetipo que nos permitirá el uso de Menús. Como del controlador del aplicativo.



## Ménus

Para el diseño de los menús vamos a usar el API de java, JAXB. Este API nos permite la transformación de XML a código Java y viceversa de una forma muy rápida y sencilla.

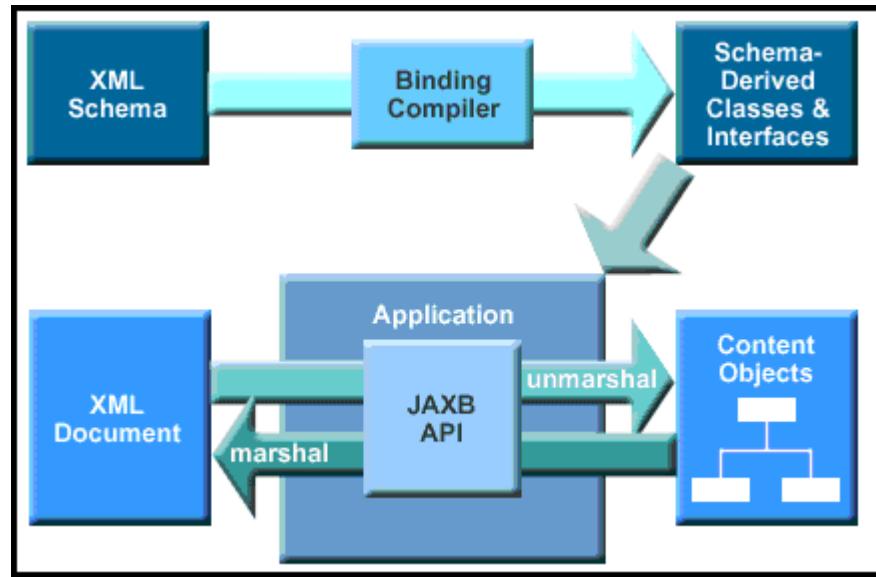


Figura 127: Generación de clases mediante al Api Jaxb

Para ello JAXB suministra mecanismos de generación de código Java a partir de un XML esquema.

**XML Schema** es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa.

La arquitectura de clases que buscamos con el esquema será la siguiente:

**ConfigMenuTag** crea la configuración de los Menús. (**MenuConfig**)

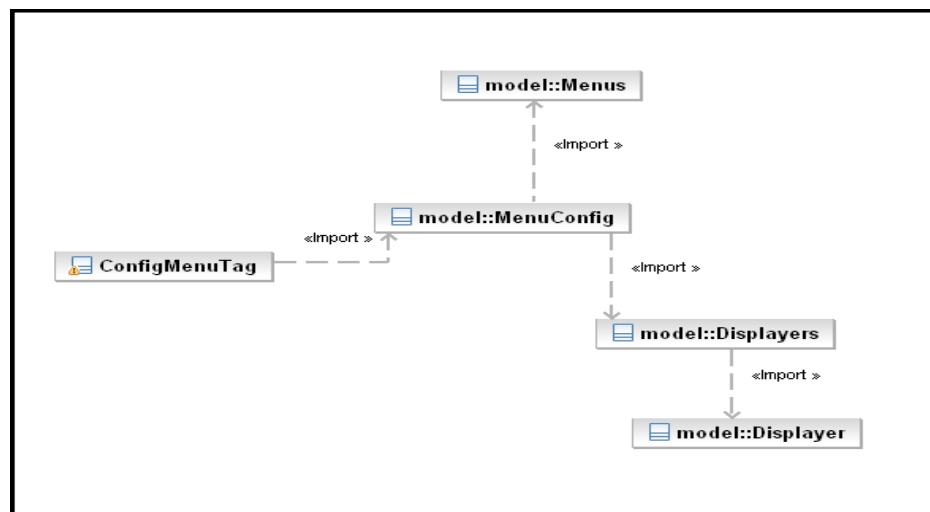


Figura 128 Diagrama de dependencias de la clase ConfigMenuTag

## Diagrama de dependencia de MenuTag

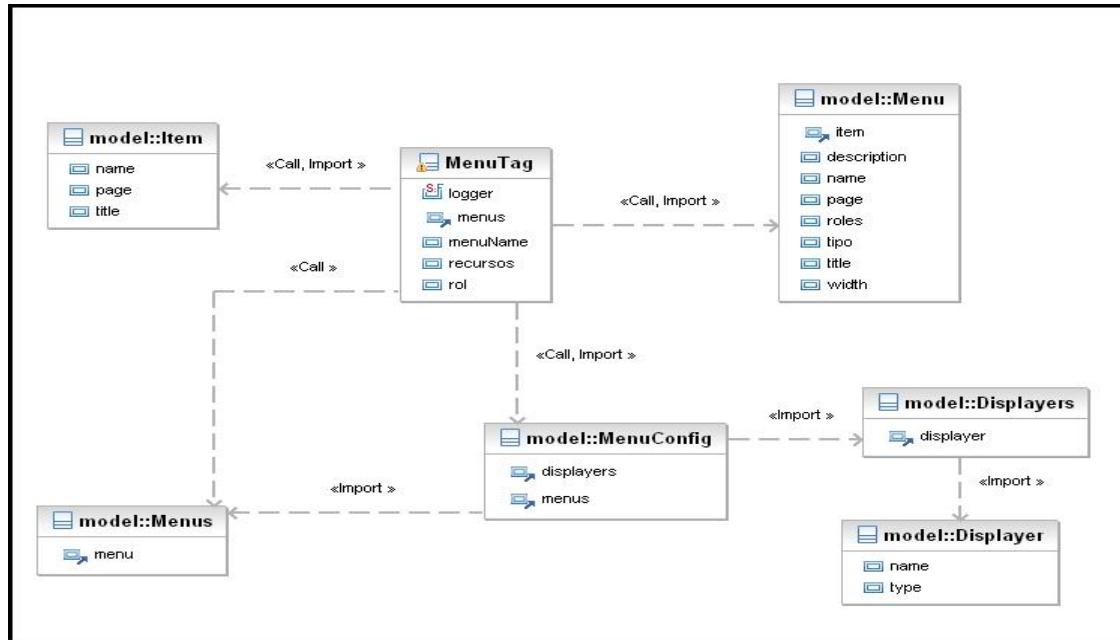


Figura 129 Diagrama de dependencia de MenuTag

**MenuConfig** está formado por **Menus** y **Displayers**. Siendo los primeros cada uno de los menus del aplicativo.



Figura 130 Diagrama dependencias MenuConfig



**Displayers** contendrá un conjunto de visualizadores (uso futuro).

**Menus** puede contener de 1 a n menús.

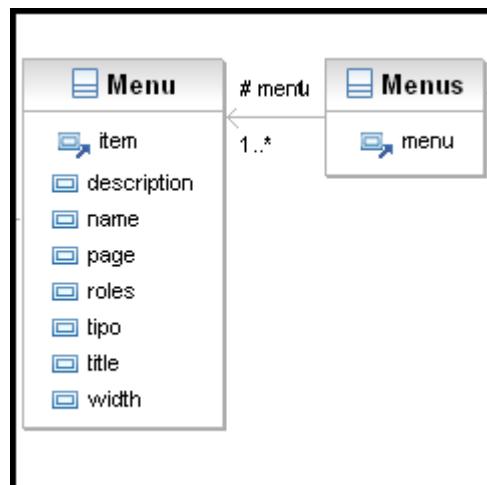


Figura 131 Diagrama asociación clase Menus

Por último un menú puede tener 0 a N apartados (Item).

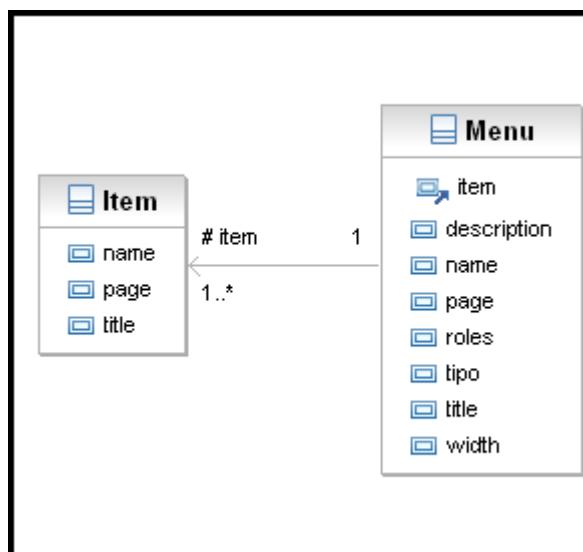


Figura 132 Diagrama asociación clase Menus

## Diagrama de Asociación de los modelos.

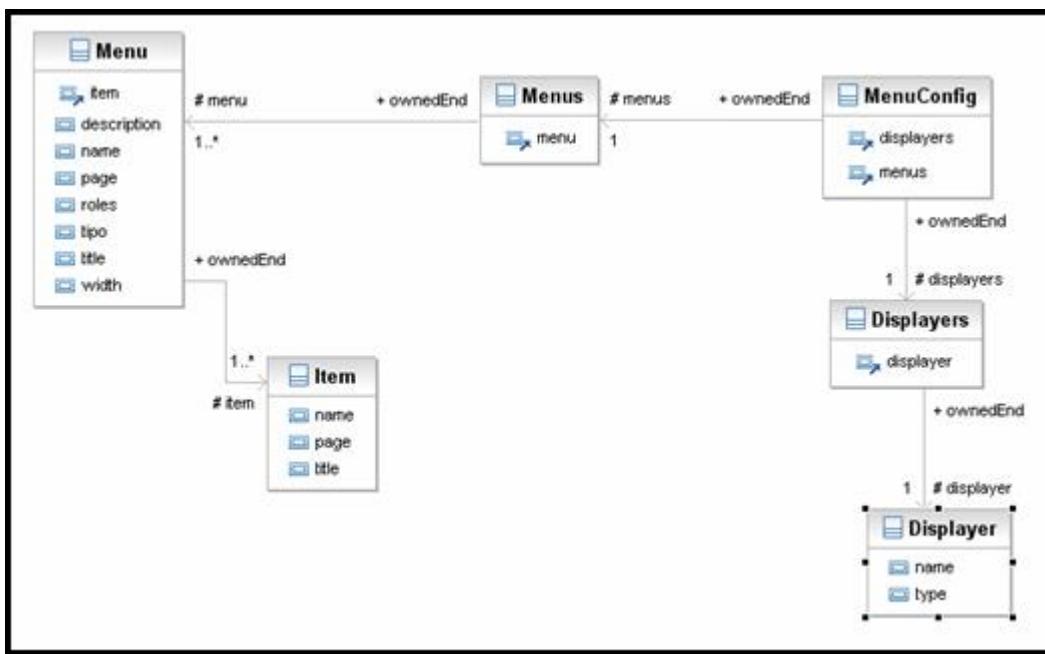


Figura 133 Diagrama asociación de los modelos

Con los diagramas *UML* anteriores nos es muy fácil diseñar un esquema (*XSD*) que cumpla con lo anterior.

```
<?xml version="1.0" encoding="UTF-8"?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema">
<xss:element name="MenuConfig">
<xss:complexType>
<xss:sequence>
<xss:element ref="Displayers" />
<xss:element ref="Menus" />
</xss:sequence>
</xss:complexType>
</xss:element>
<xss:element name="Menus">
<xss:complexType>
```



```

<xss:sequence>
<xss:element ref="Menu" maxOccurs="unbounded" />
</xss:sequence>
</xss:complexType>
</xss:element>

<xss:element name="Menu">
<xss:complexType>
<xss:sequence>
<xss:element ref="Item" minOccurs="0" maxOccurs="unbounded" />
</xss:sequence>
<xss:attribute name="width" type="xs:int" use="optional"/>
<xss:attribute name="title" type="xs:string" use="required"/>
<xss:attribute name="roles" type="xs:string" use="optional"/>
<xss:attribute name="page" type="xs:string" use="optional"/>
<xss:attribute name="name" type="xs:string" use="required"/>
<xss:attribute name="description" type="xs:string" use="optional"/>
<xss:attribute name="tipo" use="optional">
<xss:simpleType>
<xss:restriction base="xs:string">
<xss:enumeration value="vertical"/>
<xss:enumeration value="horizontal"/>
</xss:restriction>
</xss:simpleType>
</xss:attribute>
</xss:complexType>
</xss:element>

<xss:element name="Item">
<xss:complexType>
<xss:attribute name="title" type="xs:string" use="required"/>
<xss:attribute name="page" type="xs:string" use="required"/>
<xss:attribute name="name" type="xs:string" use="required"/>
</xss:complexType>
</xss:element>

<xss:element name="Displayers">

```

```

<xs:complexType>
  <xs:sequence>
    <xs:element ref="Displayer" />
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Displayer">
  <xs:complexType>
    <xs:attribute name="type" type="xs:string" use="required" />
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
</xs:schema>

```



Aplicando **Jaxb** generaremos las clases antes diseñadas en los diagramas de UML.

Para ello podemos usar variadas opciones. El plugin de **Maven** para desarrollar jaxb, ant creando un **build.xml**. No vamos a entrar en profundidad en la generación, puesto que no es el objetivo de este documento.

Para más información sobre el proceso de generación de puede visitar la pagina web

<https://jaxb.dev.java.net/>



### Controladores y escuchadores (Listener)

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. En nuestro caso tenemos 2 controladores: un *SimpleFormController* que permitirá la inscripción de usuarios y un controlador que nos llevará al menú de la aplicación.

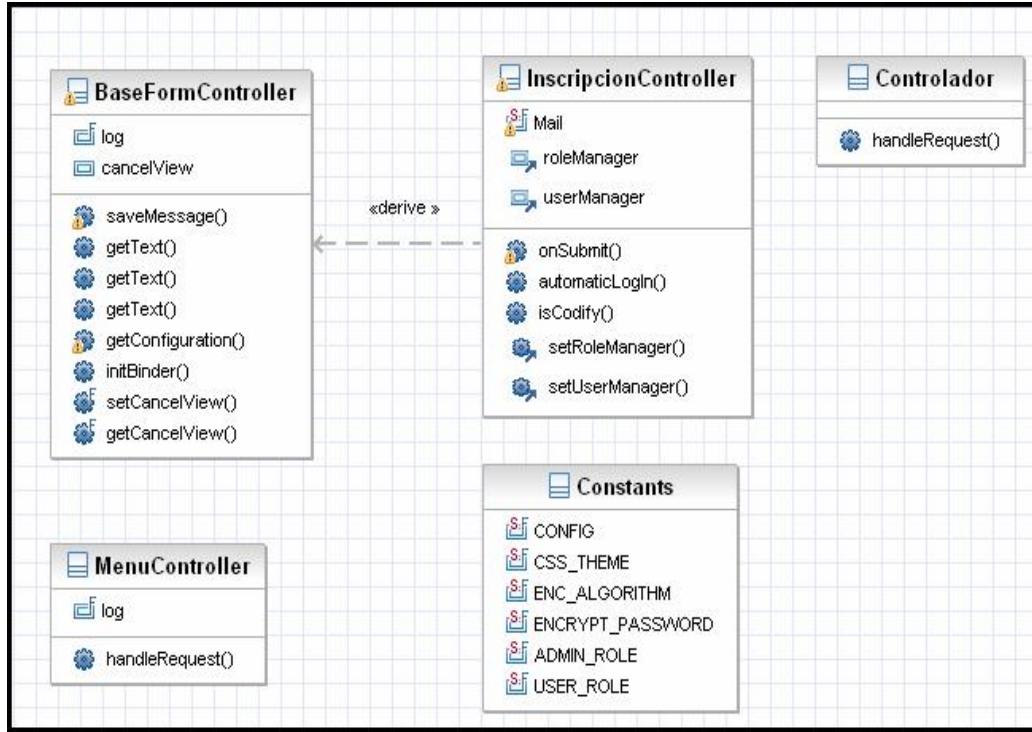


Figura 134 Diagrama de dependencias controladores

**Listener:** Situaremos en este paquete todos los escuchadores j2ee del aplicativo.

En este caso **ConfigListener**, que permite configurar el aplicativo al arrancar.



Figura 135 Listener de configuración

### 7.3.2.11. UML Package Diagram

A continuación presentamos la estructura inicial de paquetes que tendrá el futuro aplicativo.

Tomando como groupId *orgdkyosi*.

En el paquete *Web* situaremos todos los controladores.

En el paquete *listener*; situaremos todos los listener o escuchadores (j2ee) que necesite el aplicativo.

En el paquete *taglibs* se situarán las clases de generación de Menús. Han sido incluidas en cada aplicativo en lugar de en una biblioteca por si se desea la personalización del código por parte de los futuros programadores.

En el paquete *security* irán las entidades de seguridad así como las clases de autenticación de usuarios (Daos y lógica de negocio).

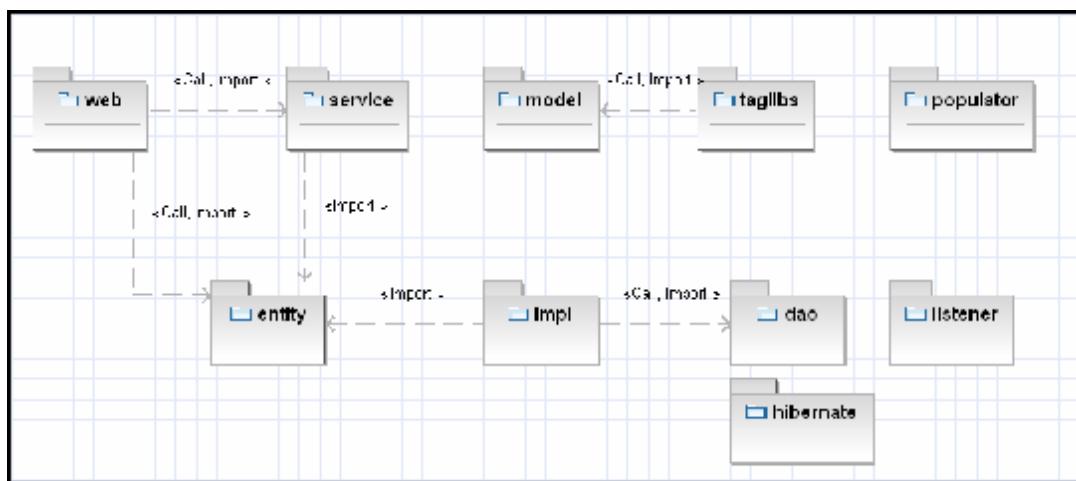


Figura 136 Diagrama de paquetes UML

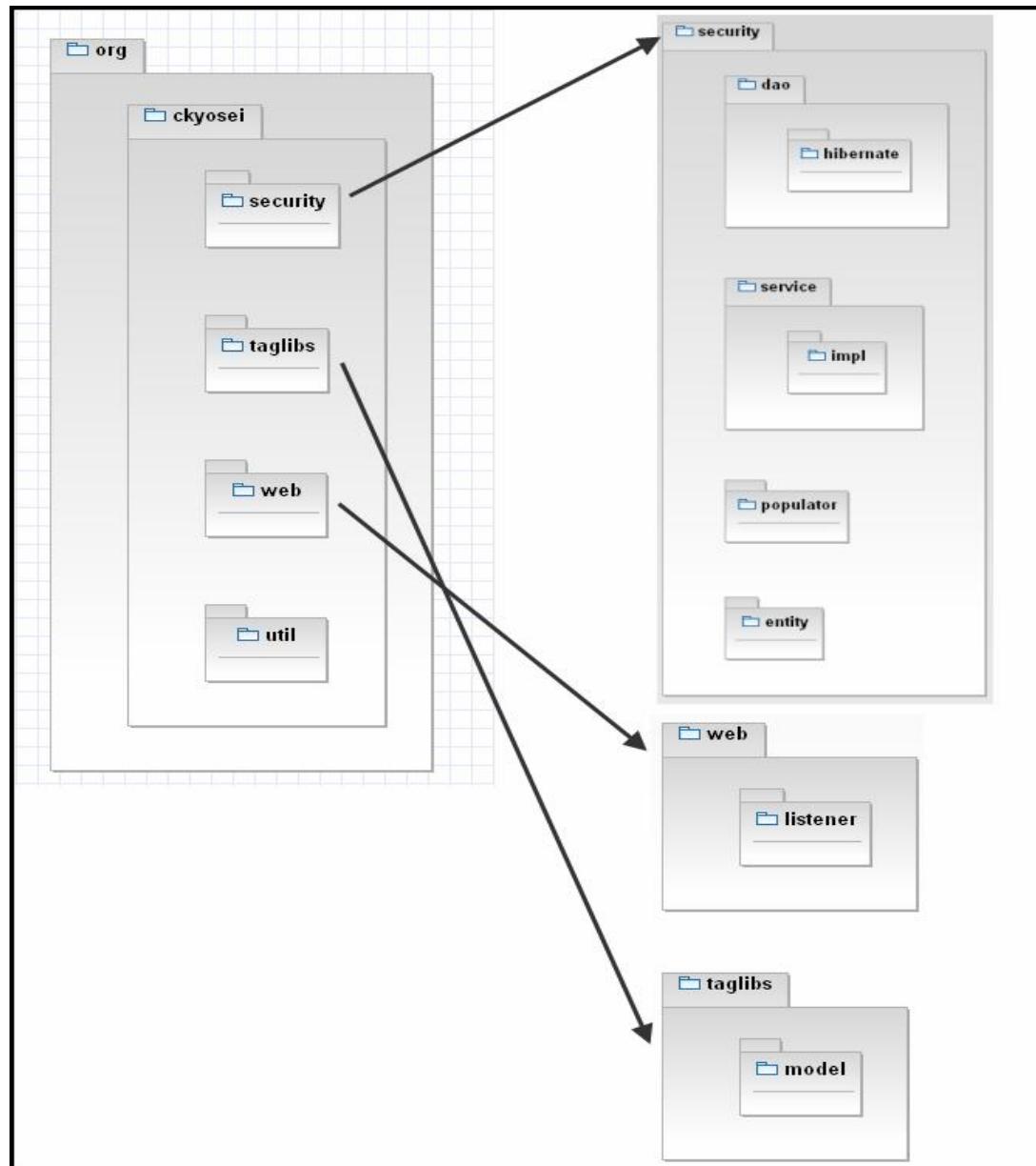


Figura 137 Diagrama de paquetes UML para aplicación generada con el GroupId `org.ckyosei`

### 7.3.2.12. Creación de Test y Aceptación

En los últimos años se han desarrollado un conjunto de herramientas que facilitan la elaboración de pruebas unitarias en diferentes lenguajes. Dicho conjunto se denomina XUnit. De entre dicho conjunto, JUnit es la herramienta utilizada para realizar pruebas unitarias en Java.

El concepto fundamental en estas herramientas es el caso de prueba (test case), y la suite de prueba (test suite). Los casos de prueba son clases o módulos que disponen de métodos para probar los métodos de una clase o módulo concreto.

Como se observa en el diagrama UML los Casos de prueba heredarán directamente de la clase TestCase, dispondrán de 1 método de inicialización y otro de destrucción del test.

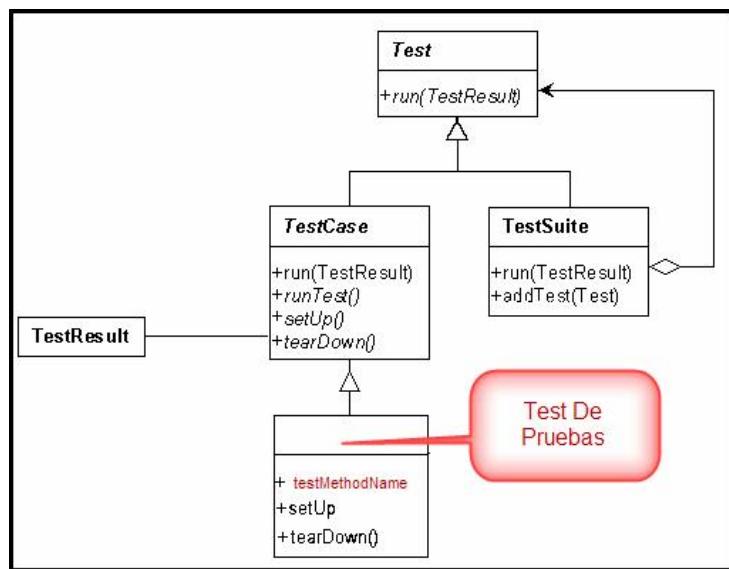


Ilustración 119: Estructura UML de test de Junit

Mediante las suites podemos organizar los casos de prueba, de forma que cada suite agrupa los casos de prueba de módulos que están funcionalmente relacionados.

Para el caso del arquetipo el Test de pruebas que vamos a realizar es sencillo. Verificaremos que se crean correctamente los menús y que se muestra correctamente la presentación.

- ü Creación de un Menú a partir de un XML de configuración.
- ü Creación de n Menús a partir del fichero de configuración.
- ü Creación de un Menú con apartados.
- ü Creación de un Menú vertical y Creación de un Menú horizontal.



### 7.3.2.13. Creación de Arquetipos en Maven

Para la creación del aplicativo inicial, que sirva de guía a futuros desarrollos vamos a emplear *Maven* y *eclipse*. Inicialmente vamos a crear un arquetipo que permita generar la estructura de una futura aplicación de Spring Framework + Hibernate + Sitemesh + etc. desde cero.

Un “arquetipo” para *Maven* es una plantilla o template. A partir del arquetipo, *Maven* es capaz de generar una estructura de directorios y ficheros inicial tan compleja como se desee.

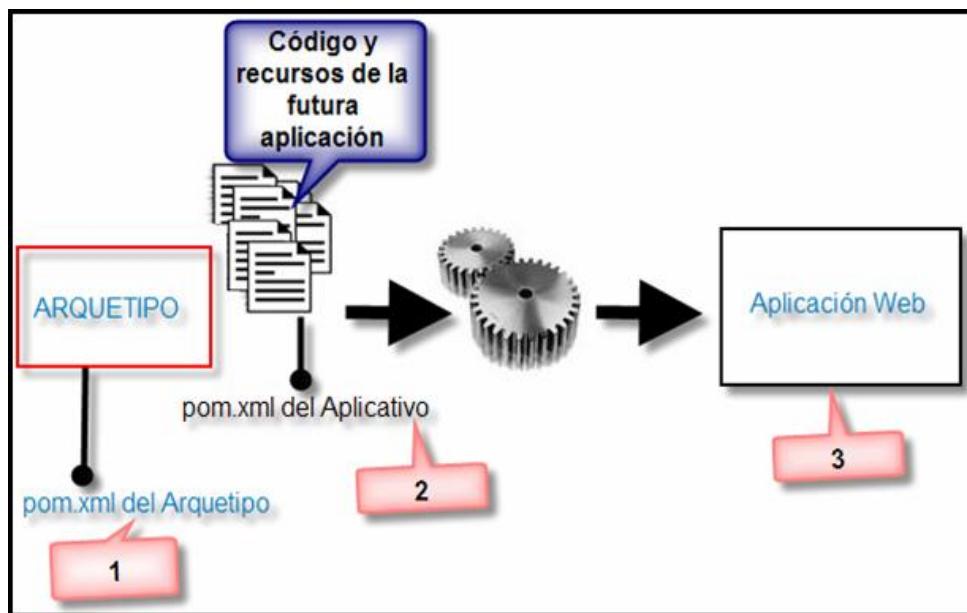


Figura 138 Arquetipos en Maven

*Maven* ya dispone de arquetipos primarios para crear diferentes tipos de aplicaciones, como aplicaciones Java o aplicaciones Web, pero lo que buscamos en nuestro proyecto es facilitar aún más las tareas de los futuros programadores del Sistema Kyosei-Polis. Para ello vamos a crear un arquetipo que genere una “aplicación inicial ya configurada”, a la cuál sólo tengan que añadir la diferente funcionalidad que se necesite implementar para la misma.

### 7.3.2.14. Creación de un Arquetipo para el Sistema Kyosei-Polis

Para la creación del arquetipo y de la futura aplicación alpha que generaremos a partir de él vamos a crear un área de trabajo nueva en *eclipse*. Cuando arranca *eclipse* le indicaremos donde queremos que nos cree el área de trabajo o workspace. *Eclipse* nos creará los ficheros internos que necesita para gestionar la nueva área de trabajo (.metadata). Estos ficheros no deben ser manipulados salvo por usuarios avanzados, ya que podrían inutilizar el área de trabajo.

El área de trabajo de la que voy a partir se encuentra en **C:\CKyosei\Workspace-arquetipo**. A partir de este punto todos los comandos estarán referidos a esta ruta.

Para indicar que a un proyecto java ya creado en eclipse, le podemos añadir la funcionalidad de Maven activando la administración de dependencias en cualquier momento.



Figura 139 Habilitar administrador de dependencias en eclipse

Se generará la estructura de directorios con la que trabaja *Maven*, que separa el código java del aplicativo del código de los test y de los recursos.

```
src/main/java  
src/main/resources  
src/test/java  
src/test/resources
```

Cuando se compila obtendremos una estructura similar a la siguiente:

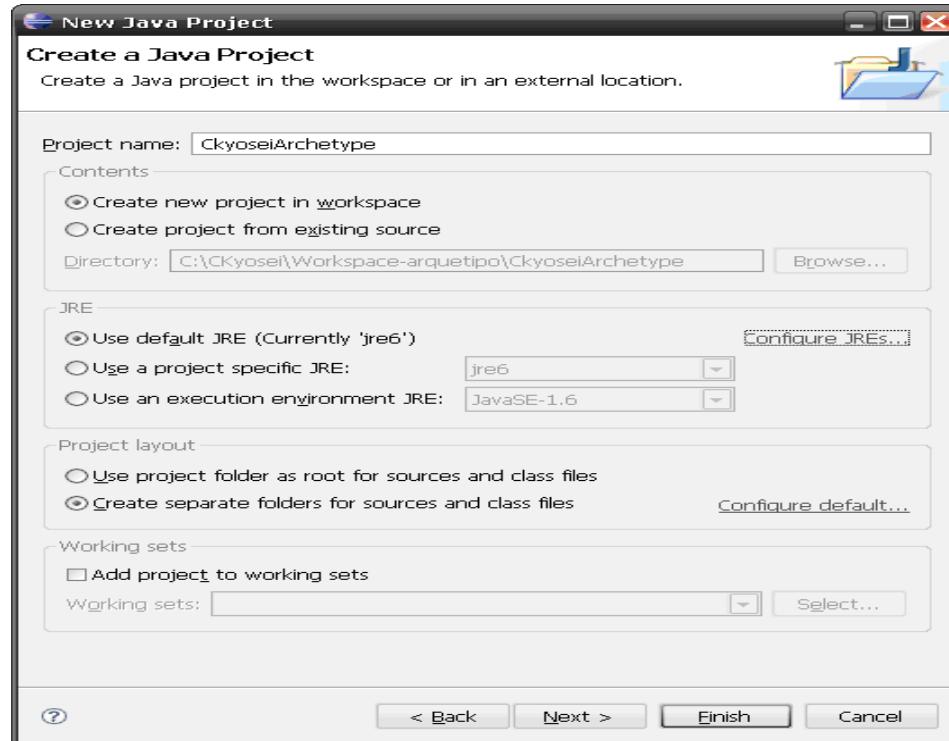


Todos los recursos de la carpeta `src/main/resources` son añadidos al Classpath de classes y los de test al de test-classes. La carpeta **Surefire-Reports**, es donde *Maven* deja los ficheros de trazas de los test de Junit.

Hay dos formas diferentes de crear un arquetipo: desde eclipse, creando un nuevo proyecto java y añadiendo mediante el plugin de *Maven* el fichero `pom.xml`, o creando directamente un proyecto *Maven* en eclipse.

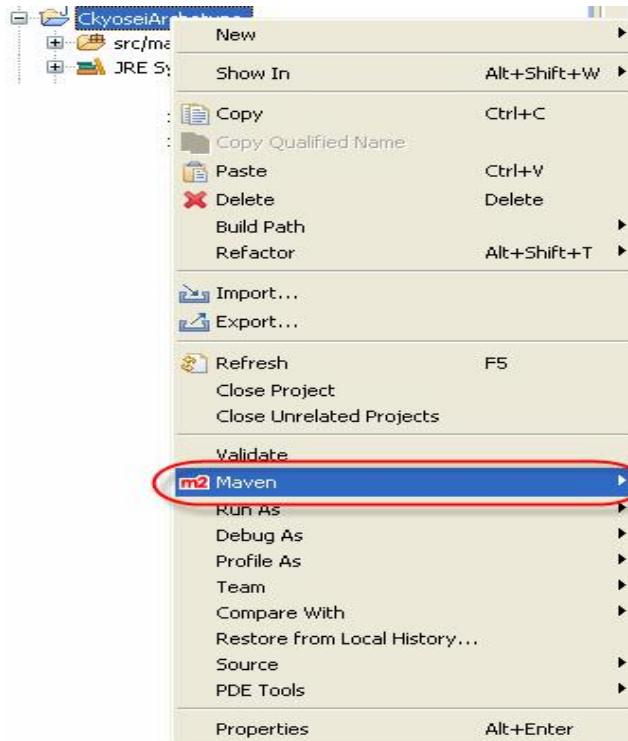
## 1. Crear un proyecto java nuevo desde eclipse.

### 1.1. Accedemos menú `File – New – Java project`



**Figura 140 Creación el proyecto en eclipse para el arquetipo**

1.2. Seleccionando el proyecto y pulsando el botón derecho del ratón habilitaremos el uso de Maven para el proyecto. Ello creará automáticamente el archivo pom.xml.



**Figura 141 Selección de Maven en eclipse.**

- 1.3. El fichero **pom.xml** es el descriptor que usaremos para introducir la información del arquetipo, su **GroupId** y **ArtifactId**, etc.
- 1.4. Editamos los tags **GroupId** y **ArtifactId** del fichero **pom.xml**, determinan el grupo de aplicaciones al que pertenece nuestra aplicación y el identificador de la propia aplicación. Con packaging se indica el tipo de empaquetado del proyecto (jar, war, ear, pom...):

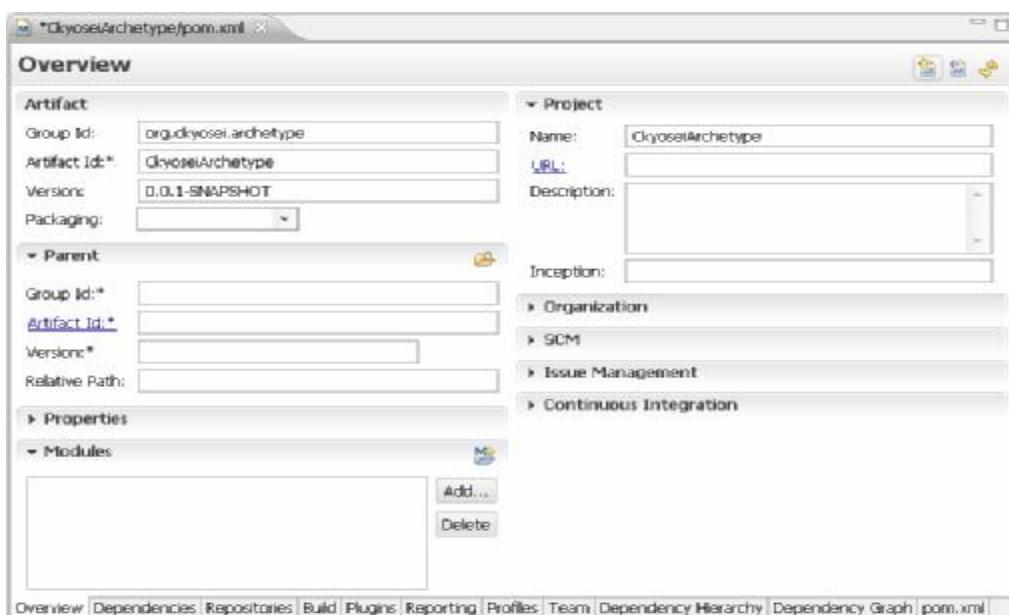


Figura 142 Datos del fichero Pom.xml, del arquetipo

Para entender exactamente qué estamos haciendo tenemos que recordar que **Maven** usa un repositorio local donde se guardan los diferentes objetos o dependencias de código que se van necesitando y que se descargarán de los repositorios remotos.

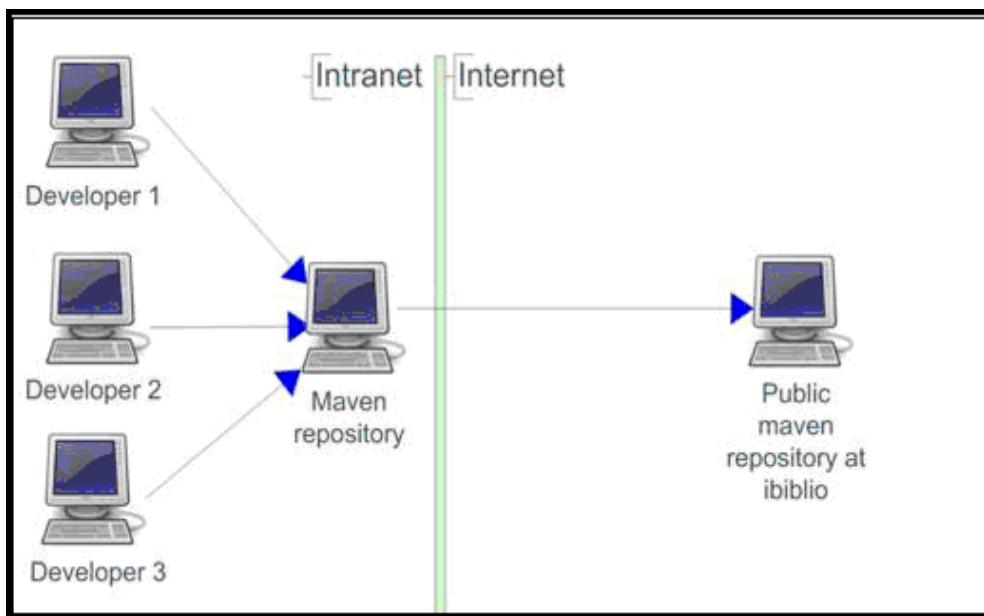


Figura 143 Figura 143 Esquema funcionamiento repositorios de Maven



En nuestro repositorio local *Maven* lo guardaría usando el nombre del *GroupId* como carpeta padre, luego el *artifactId*, y luego las diferentes versiones que tengamos de la librería en nuestro repositorio.

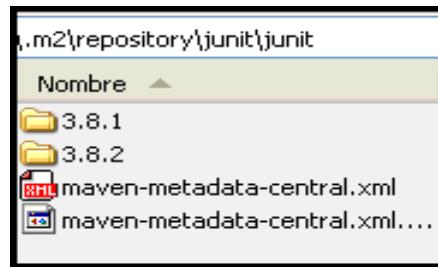


Figura 144 : Ejemplo instalación de un artefacto en el repositorio en este caso Junit

Por lo tanto con los *tags* del fichero *pom.xml* del arquetipo, *GroupId* y *ArtifactId*, simplemente indicamos la ruta donde se almacenará el arquetipo dentro del repositorio, así como el nombre que guardará el mismo cuando se despliegue y la versión que se debe guardar.

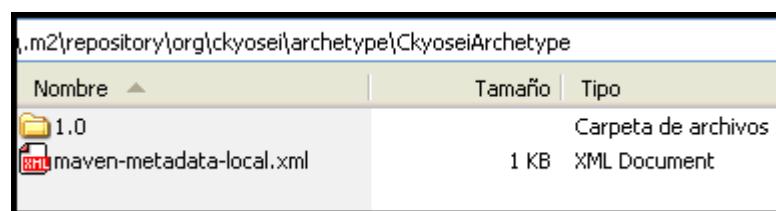


Figura 145 Arquetipo de Ckyosei una vez implantado en el repositorio local

Con *version* se indica la versión del proyecto inicial con la que vamos a trabajando. Al indicar *SNAPSHOT* se quiere decir que es una versión evolutiva, es decir que estamos trabajando para obtener una versión estable.

En este fichero *pom.xml* también se pueden definir las dependencias de librerías que tenga el proyecto.

2. Alternativamente, podemos crear directamente el proyecto *Maven* en Eclipse, con *File – New – Maven Project*:

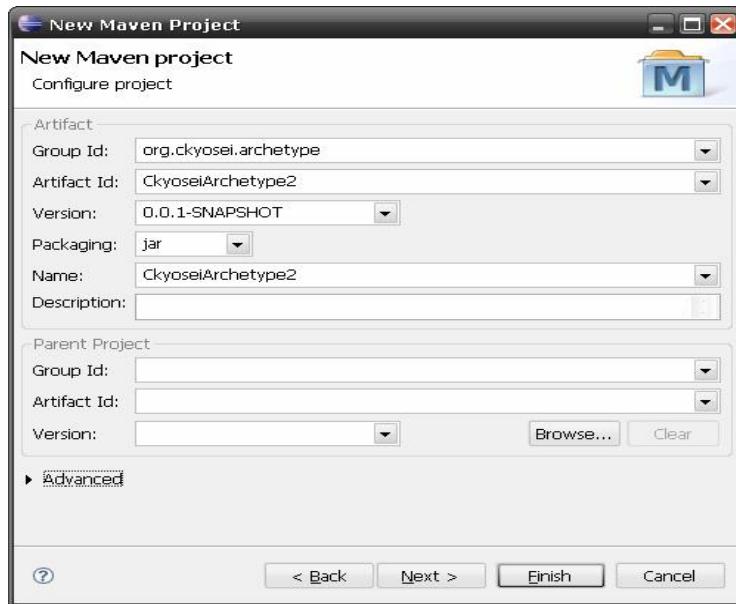


Figura 146 Creación del arquetipo como Proyecto Maven

### 3. Creación del descriptor del Arquetipo.

- #### 3.1. Dentro del directorio `src/main/resources` creamos una nueva carpeta `META-INF` donde situaremos el fichero `archetype.xml`. Las diferentes secciones que podemos tener en el fichero descriptor son las siguientes:

```
<sources> = src/main/java  
<resources> = src/main/resources  
<testSources> = src/test/java  
<testResources> = src/test/resources  
<siteResources> = src/site
```

(Para más información ver la documentación de *Maven* para la creación de Arquetipos)

El fichero `archetype.xml` contendrá las rutas a los diferentes recursos que se incluirán al generar una aplicación, como clases, jsps, CSS y scripts. Se pueden incluir todos los elementos que queramos que tenga la aplicación inicialmente.

En nuestro caso, generaremos la siguiente estructura:



“

```
<archetype>
    <id>CkyoseiArchetype</id>
    <sources>
        <!-- Controlador primario de aplicación -->
        <source>src/main/java/web/Controlador.java</source>
        .....
        <!-- Constantes de la aplicación de la capa de Controladores -->
        <source>src/main/java/web/Constants.java</source>

        <!-- Configuración de aplicación -->
        <source>src/main/java/web/listener/ConfigListener.java</source>

        <!-- Configuración de menus -->
        <source>src/main/java/taglibs/ConfigMenuTag.java</source>
        <source>src/main/java/taglibs/Constants.java</source>
        <source>src/main/java/taglibs/MenuTag.java</source>
        .....
        </sources>
        <resources>
            <!-- ficheros de estilos -->
            <resource>src/main/webapp/CSS/ckyosei/theme.CSS</resource>
            .....
            <!-- ficheros de imagenes -->
            <resource>src/main/webapp/images/ckyosei/logo.jpg</resource>
            <!-- ficheros de script -->
            <resource>src/main/webapp/script/ckyosei/sdmenu.js</resource>

            <resource>src/main/webapp/decorators/estructura.jsp</resource>
            <resource>src/main/webapp/estructura/header.jsp</resource>
            .....
        </resources>
    </sources>
</archetype>
```

```

<resource>src/main/webapp/taglibs.jsp</resource>
.....
<!-- ficheros de configuración -->
<resource>src/main/webapp/WEB-INF/ckyosei-
servlet.xml</resource>
<resource>src/main/webapp/WEB-
INF/applicationContext.xml</resource>
<resource>src/main/webapp/WEB-
INF/dataAccessContext.xml</resource>
<resource>src/main/webapp/WEB-INF/config.properties</resource>
<resource>src/main/webapp/WEB-INF/jdbc.properties</resource>
<resource>src/main/webapp/WEB-INF/web.xml</resource>
<resource>src/main/webapp/WEB-INF/menu.tld</resource>

<!-- ficheros de configuración Sitemesh -->
<resource>src/main/webapp/WEB-INF/Sitemesh.xml</resource>
<resource>src/main/webapp/WEB-INF/decorators.xml</resource>
<resource>src/main/webapp/WEB-INF/jsp/index.jsp</resource>
<resource>src/main/webapp/index.html</resource>
<!-- ficheros de internacionalización -->

<resource>src/main/resources/ArchetypeResources_es.properties</resourc
e>
<resource>src/main/resources/menu-config.xml</resource>
</resources>
</archetype>

```





De la estructura del archivo anterior caben destacar los ficheros de configuración de la futura aplicación estos ficheros se sitúan dentro de **WEB-INF**.

- Ü **ckyosei-servlet.xml**, donde definiremos los controladores del MVC de Spring.
- Ü **applicationContext.xml**, donde definiremos los beans de la lógica de negocio.
- Ü **dataAccessContext.xml**, que definirá los beans de acceso a las bases de datos.
- Ü **config.properties**, servirá para definir propiedades genéricas del aplicativo.
- Ü **jdbc.properties**, servirá para definir propiedades de acceso a las bases de datos del aplicativo.
- Ü **Sitemesh.xml** Archivo de configuración de Sitemesh.
- Ü **decorators.xml** Archivo de configuración de los decoradores de Sitemesh.

Estos ficheros tendrán que ser modificados para añadir la diferente funcionalidad de cada aplicativo en las futuras aplicaciones generadas.

Los archivos **JSP** del aplicativo, por su parte, están dentro de **WEB-INF/JSP** zona restringida de acceso no visible desde el exterior.

#### 4. Crear los archivos de la futura aplicación así como el fichero descriptor del aplicativo **pom.xml**.

En este punto vamos a crear el archivo **pom.xml** del futuro aplicativo, con sus dependencias iniciales. Este archivo se modificará posteriormente una vez creado el aplicativo en función del proyecto. ¡Ojo!, no confundir este fichero **pom.xml** con el fichero inicial que generamos al crear el proyecto java. El fichero **pom.xml** inicial, sirve para trabajar sobre el arquetipo y el que vamos a crear ahora servirá para la futura aplicación que generemos a partir de este arquetipo. Para no confundirnos vamos a referirnos a partir de este momento como “**pom.xml** del arquetipo” y “**pom.xml** del aplicativo”.

Creamos un directorio **archetype-resources** dentro del directorio **META-INF**, y dentro de él crearemos el fichero **pom.xml** del aplicativo.

A continuación se muestra un fragmento del fichero **pom.xml** del aplicativo. Como se observa \${groupId}, \${artifactId} y \${version} son parámetros que serán recogidos al invocar al arquetipo desde la línea de comandos:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>${groupId}</groupId>
  <artifactId>${artifactId}</artifactId>
  <packaging>war</packaging>
  <name>${artifactId}</name>
  <version>${version}</version>
  <inceptionYear>2009</inceptionYear>
  <organization>
    <name>Ckyosei</name>
    <url>http://www.ckyosei.org</url>
  </organization>
  <build>
    <finalName>${artifactId}</finalName>
    .....
  </build>
```

También situaremos los ficheros que queremos que se añadan a la estructura inicial, los ficheros cuyos rutas de acceso o path indicamos en el fichero anterior **archetype.xml**. De forma que al final vamos a tener una estructura como la figura siguiente.

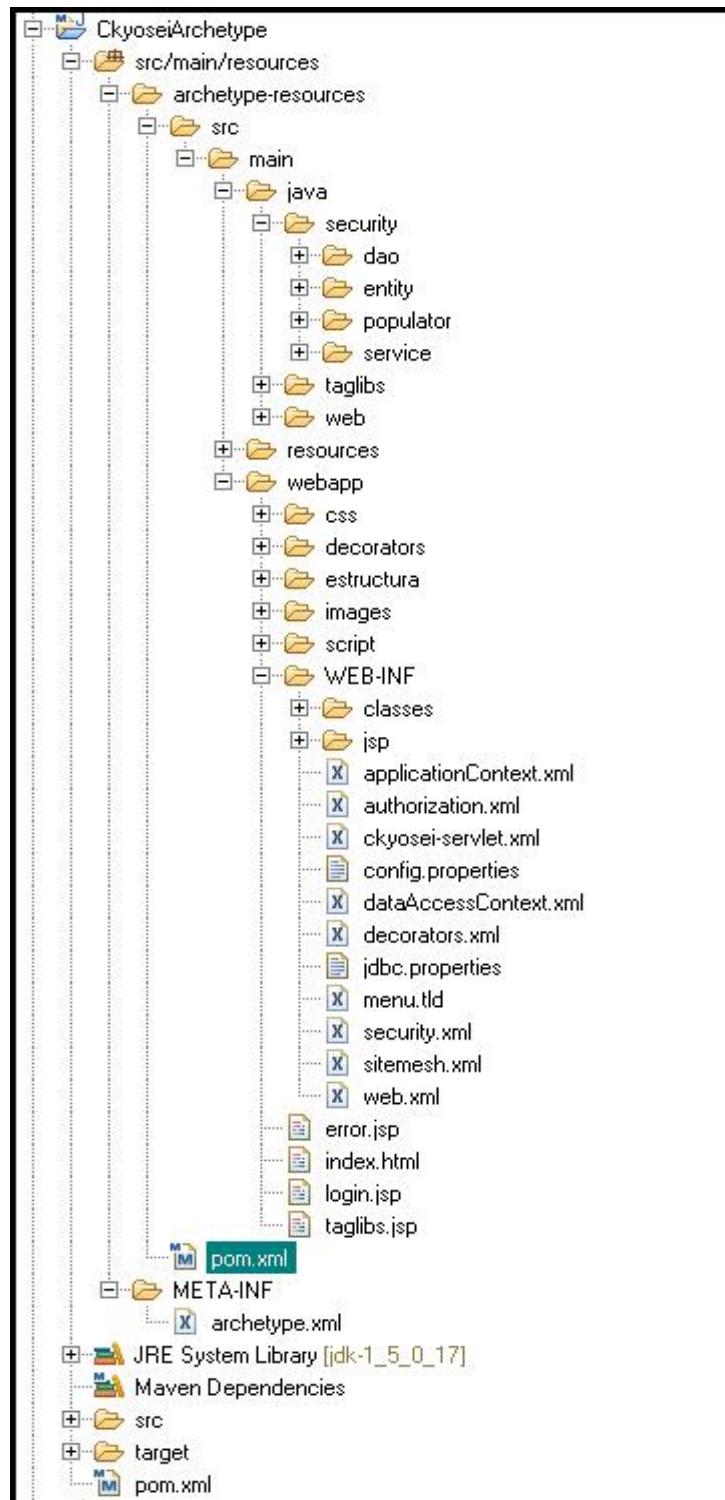
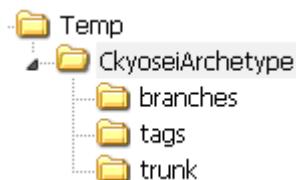


Figura 147 : Estructura global del arquetipo CkyoseiArchetype

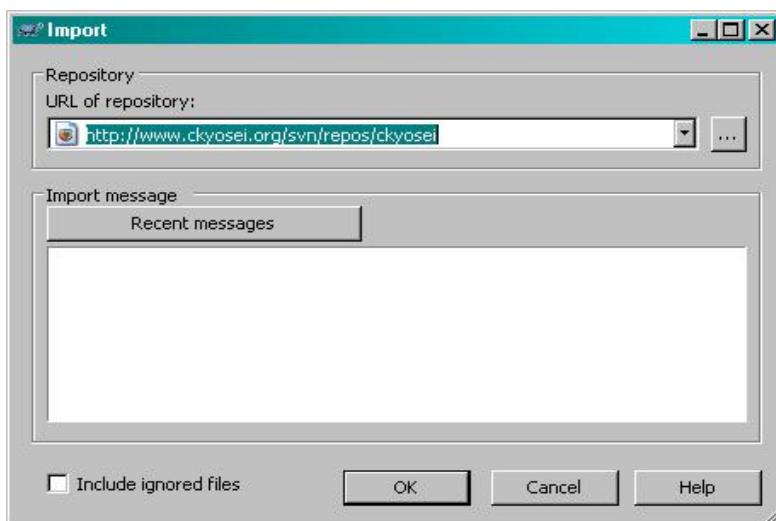
## 5. Instalación del arquetipo en el control de versiones (**Subversion**).

En este punto damos por supuesto que ya tenemos configurado Subversion y que tenemos los repositorios necesarios que indicamos anteriormente (el de código, el de versiones y el directorio de documentación).

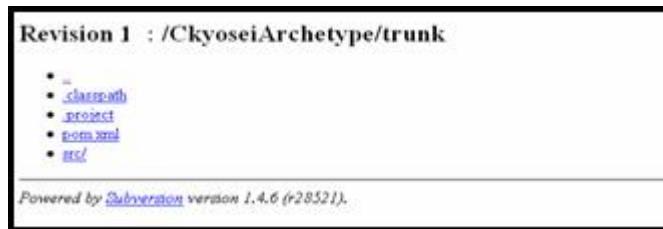
En el explorador de archivos creamos una estructura como:



Situándonos encima de la carpeta accedemos al cliente TortoiseSVN pulsando el botón derecho del ratón y seleccionamos la opción Import. La carpeta **Temp** o raíz que usemos no se importará al svn.



## 6. Subimos la estructura del arquetipo.



En este punto tenemos todo nuestro código versionado. Cualquier cambio podrá ser fácilmente detectado por el SVN (subversion).



## 7. Aplicando XP y AMDD a la aplicación alpha.

### 7. Instalación del arquetipo en el repositorio local.

7.1. Abrimos una consola DOS y ejecutamos:

```
C:\CKyosei\Workspace-arquetipo\CkyoseiArchetype>mvn install
[INFO] Scanning for projects...
[INFO] -----
-----
[INFO] Building CkyoseiArchetype
[INFO]   task-segment: [install]
[INFO] -----
-----
[INFO] Ignoring available plugin update: 2.4 as it requires Maven version
2.0.8
[INFO] [resources:resources]
[INFO] Using default encoding to copy filtered resources.
[INFO] [compiler:compile]
[INFO] No sources to compile
[INFO] [resources:testResources]
[INFO] Using default encoding to copy filtered resources.
[INFO] [compiler:testCompile]
[INFO] No sources to compile
[INFO] [surefire:test]
[INFO] No tests to run.
[INFO] [jar:jar]
[INFO] Building jar: C:\CKyosei\Workspace-
arquetipo\CkyoseiArchetype\target\CkyoseiArchetype.jar
[INFO] Preparing javadoc:javadoc
[INFO] -----
-----
[INFO] Building CkyoseiArchetype
[INFO] -----
-----
.....
[INFO] Installing C:\CKyosei\Workspace-
arquetipo\CkyoseiArchetype\target\CkyoseiArchetype.jar to C:\Documents and
Settings\ZAPA\.m2\repository\org\
ckyosei\archetype\CkyoseiArchetype\1.0\CkyoseiArchetype-1.0.jar
[INFO] -----
--
[INFO] BUILD SUCCESSFUL
[INFO] -----
--
[INFO] Total time: 11 seconds
[INFO] Finished at: Thu Sep 18 17:21:30 CEST 2008
[INFO] Final Memory: 11M/23M
```

Como se observa en la salida del comando, se genera un archivo **jar**, que se instala en el directorio donde tengamos configurado el repositorio de **Maven**, en este caso en **C:\Documents and Settings\\*\*\*\*\.m2\repository\**.

En este punto acabaría la creación del arquetipo a partir del cual se podrán crear tantas aplicaciones como deseemos. A continuación vamos a mostrar los comandos que permitirán la creación de una aplicación a partir del arquetipo.

## 8. Instalación del arquetipo en el repositorio remoto

Para instalar el arquetipo en el repositorio remoto que creamos. Podemos usar el plugin de Maven **maven-release-plugin** para lo que tendremos que ejecutar 2 comandos:

```
mvn release:prepare  
mvn release:perform
```

Uno para preparar la versión (prepare) y otro que ejecutará la subida o simplemente ejecutar el comando deploy.

```
mvn release:deploy
```



El resultado en ambos casos es idéntico, solo que el primer comando comprobará la coherencia del área de trabajo con el SVN, e incrementará al final del mismo el número de versión del artefacto.

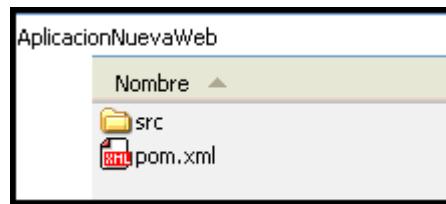
## 1. Creación de un aplicativo a partir del arquetipo.

### 1.1. Nos situamos el directorio raíz del área de trabajo

```
C:\CKyosei\Workspace-arquetipo>mvn archetype:create -DarchetypeGroupId=org.ckyosei.archetype -DarchetypeArtifactId=CkyoseiArchetype -DarchetypeVersion=1.2 -DgroupId=test.aplicacion.nueva -DartifactId=AplicacionNuevaWeb
[INFO] Archetype created in dir: C:\CKyosei\workspace-arquetipo\AplicacionNuevaWeb
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 4 seconds
[INFO] Finished at: Thu Sep 18 17:26:46 CEST 2008
[INFO] Final Memory: 4M/9M
[INFO] -----
--
```

En caso de no tener instalado el arquetipo en el repositorio local, se le puede indicar al comando anterior en qué repositorio remoto se encuentra añadiendo el parámetro `-DremoteRepositories` y la url donde se encuentra el repositorio.

```
C:\CKyosei\Workspace-arquetipo>mvn archetype:create -DarchetypeGroupId=org.ckyosei.archetype -DarchetypeArtifactId=CkyoseiArchetype -DarchetypeVersion=1.0 -DgroupId=test.aplicacion.nueva -DartifactId=AplicacionNuevaWeb -DremoteRepositories=http://URL
```



**Figura 149 Aplicativo generado a partir del arquetipo CkyoseiArchetype**

### 1.2. Entramos en el aplicativo que ha sido generado:

El nuevo aplicativo si refrescamos el área de trabajo de eclipse es invisible para el entorno, porque no tiene todavía los ficheros que eclipse necesita para reconocerlo como suyo. Para realizar esta operación necesitamos ejecutar el siguiente comando:

```
C:\CKyosei\Workspace-arquetipo\AplicacionNuevaWeb>mvn eclipse:eclipse -Dwtpversion=2.0
```

Con **mvn eclipse:eclipse** lo que hace es crear los ficheros **.project** y **.classpath**. La extensión **-Dwtpversion** indica que debe añadirse el soporte para aplicación web. La versión depende de la versión de Web Tools Platform (WTP) que se tenga instalada.

1.3. Ahora podemos importar el proyecto en eclipse, con: **File - Import - General - Existing Projects into Workspace**.

Si queremos desplegar la aplicación en Tomcat, lo más lógico es incluir su entorno de ejecución para comprobar la correcta compatibilidad con sus librerías.

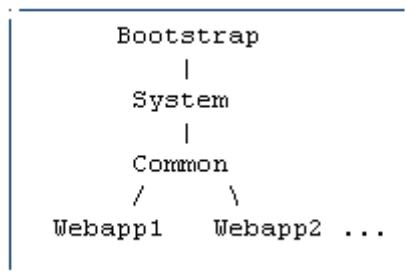


Figura 150 Cargador clases Tomcat

Cuando Tomcat 6 se inicia, se van cargando las clases de arriba hacia abajo según la figura siguiente:

**Common** equivale a **lib** según la documentación y como se observa está por encima de todos los aplicativos que se desplieguen. Esta carpeta ya contiene las librerías antes mencionadas.

Para incluir el runtime de Tomcat en eclipse haremos clic con el botón derecho del ratón sobre el proyecto y seleccionaremos **Properties – Java Build Path – Add Library – Server Runtime – Apache Tomcat v6**.

Una vez que hemos incluido el runtime de Tomcat refresharemos el proyecto pulsando F5 sobre el mismo, y forzamos su compilación con: **Project – Clean**. El proyecto quedará sin ningún error de compilación.

## 2. Despliegue del nuevo aplicativo en el entorno Tomcat de eclipse.

Desde la vista J2EE iremos a la pestaña Server y pulsaremos botón derecho del ratón: **New –Server**. Seleccionamos un servidor de tipo **Apache – Tomcat v6.0** y proporcionamos la ruta de instalación.

Desde el navegador llamaremos a la URL:

**http://localhost:8080/ AplicacionNuevaWeb**



**Ciudades Kyosei**

**Cabecera**

**Menú Global**

**Menú Local**

**Información**

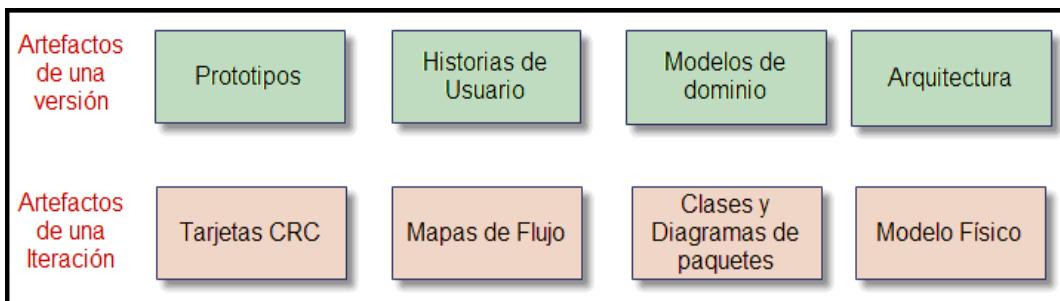
- \* Zona de acceso público  
Esta situado en la zona de acceso público.
- \* Acceso a contenidos restringidos  
Para acceder a esta zona pulse en enlace zona privada.

**Zona de información**

Figura 151 Página principal de la aplicación generada con el arquetipo

### 7.3.3. Aplicando XP y AMDD a la librería de utilidades

Como ya vimos en un punto anterior los pasos que estamos dando para la generación del artefacto son los siguientes.



En este caso la generación de prototipos, historias de usuario y modelo de dominio no tienen mucho sentido. Por lo que vamos a ir directamente a la arquitectura.

#### 7.3.3.1. Arquitectura de la librería.

Como ya hemos indicado a lo largo del documento, la librería de utilidades va a contener un conjunto de clases que servirán para agilizar los desarrollos de las aplicaciones creadas del sistema. Por lo que no tiene sentido hablar de arquitectura global.

Principalmente la librería en su primera versión se va a centrar en la capa de acceso a datos y a facilitar el diseño de los test de pruebas.

Esta librería va a contener clases genéricas que permitirán tanto la creación de DAOs, como lógica de negocio de una manera rápida y unificada para trabajar con el framework de persistencia de Hibernate.

Básicamente vamos a generar clases que utilicen la combinación del patrón DAO para acceso a datos y el patrón proxy usando Hibernate para la generación de los transferObject.

Los DAO permiten:

- ü Todos los acceso a bases de datos en el sistema se realiza a través de un DAO para lograr la encapsulación.
- ü Cada instancia DAO es el responsable de un dominio principal objeto o entidad. Si un objeto de dominio tiene un ciclo de vida independiente, debe tener su propio DAO.
- ü El DAO es el responsable de creaciones, dice (por clave primaria), actualizaciones y supresiones - es decir, CRUD - en el dominio objeto.



- Ú El DAO puede permitir que las consultas sobre la base de criterios que no sean la clave principal. Me refiero a estos métodos como buscador o buscadores. El valor de retorno de un buscador es normalmente una colección del dominio objeto del que el DAO es el responsable.

El DAO no es responsable para el manejo de las operaciones, los períodos de sesiones o las conexiones.

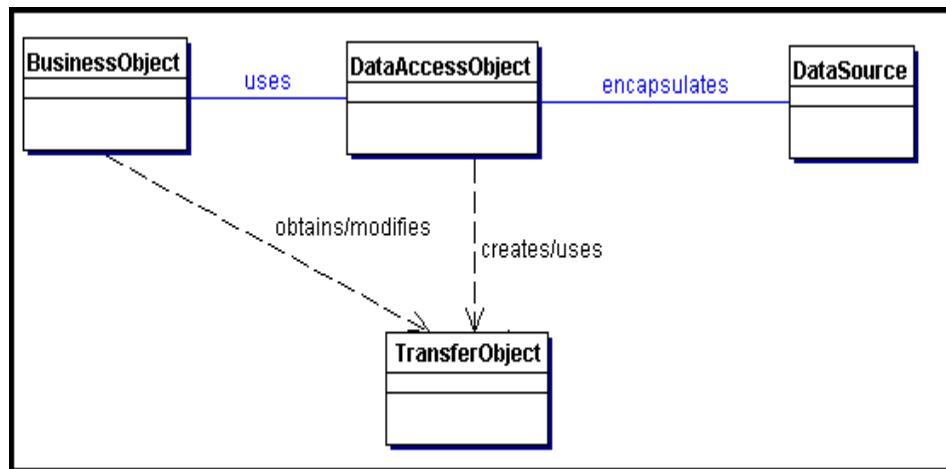
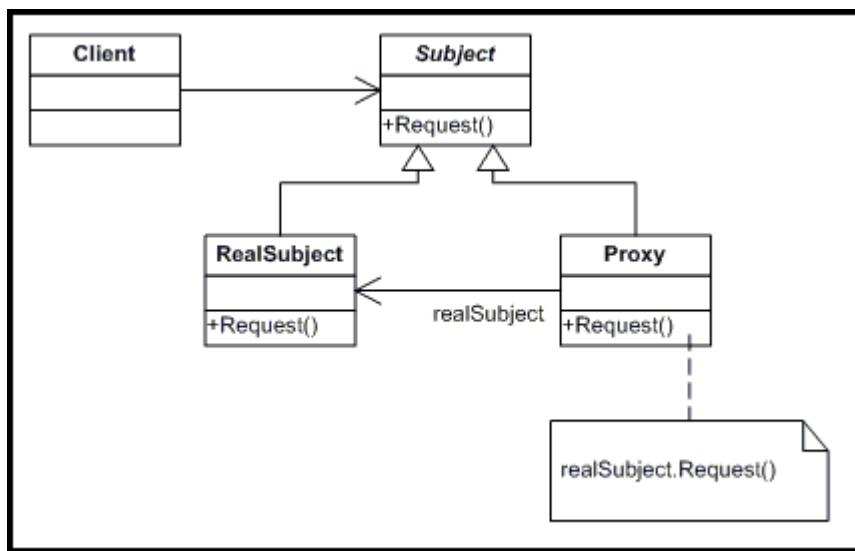


Figura 152 Patrón diseño Dao

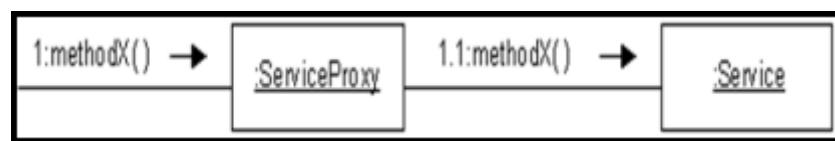
Estos son manejados fuera de la DAO para lograr flexibilidad.

El patrón proxy:

- Ú Este patrón, nos proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.
- Ú Un objeto Proxy, recibe llamadas a métodos que pertenecen a otros objetos.
- Ú El patrón obliga a que las llamadas a métodos de un objeto ocurran indirectamente a través de un objeto Proxy.
- Ú Útil cuando se desea retrasar la creación de un objeto hasta que sea necesario usarlo (optimiza operaciones costosas, como invocar imagen).



**Figura 153 Patrón de diseño Proxy**



### 7.3.3.2. CRC Cards

La estructura que vamos a presentar en las tarjetas será la siguiente:

Nombre de la Clase	
Responsabilidades (obligaciones de esta clase, tales como los métodos de negocio, manejo de excepciones, métodos de seguridad, los atributos y variables)	Colaboradores (otras clases necesarias para proporcionar una solución completa)



### Tarjetas CRC para la creación de DAOs.

#### CRC0

<b>Dao</b>	
Proporcionar métodos comunes a todos los Daos que no soportan Template (java 1.4 -).	BaseHibernateDAOImpl. Clase implementa el interface para usar el framework de Hibernate..

#### CRC1

<b>GenericDAO&lt;T, PK extends Serializable&gt;</b>	
Clase implementa el interface para usar el framework de Hibernate..	GenericHibernateDAO<T, PK extends Serializable>. Clase implementa el interface para usar el framework de Hibernate..

#### CRC2

<b>BaseHibernateDAOImpl</b>	
Clase implementa el interface para usar el framework de Hibernate..	HibernateDaoSupport Implementación de Spring para trabajar con Hibernate

#### CRC3

<b>GenericHibernateDAO&lt;T, PK extends Serializable&gt;.</b>	
Clase implementa el interface para usar el framework de Hibernate..	<b>HIBERNATEDAO SUPPORT</b> <b>IMPLEMENTACIÓN DE SPRING PARA TRABAJAR CON HIBERNATE</b>

#### CRC4

<b>FinderNamingStrategy</b>	
Se utiliza para localizar el nombre de una consulta.	<b>SimpleFinderNamingStrategy</b> Clase implementa el interface. Busca el nombre de la consulta de Hibernate basada en el nombre del método de invocación.

#### CRC5

<b>FinderExecutor</b>	
Interface con los métodos que permitirá ejecutar usando AOP.	GenericHibernateDAO<T, PK <b>extends Serializable</b> > Implementación de los métodos que evitan reescribir las consultas y usar Hibernate query name..

#### CRC6

<b>FinderArgumentTypeFactory</b>	
Se utiliza para localizar cualquier tipo de mapeo específico que podría ser necesario para una aplicación dao	SimpleFinderArgumentTypeFactory Enumeraciones mapas personalizados a un tipo de usuario de Hibernate.

#### CRC7

<b>ExtendedFinderNamingStrategy</b>	
Se utiliza para localizar cualquier tipo de mapeo específico que podría ser necesario para una aplicación dao	Ver SimpleFinderNamingStrategy.



### CRC8

<b>FinderIntroductionAdvisor</b>	
Conecta Spring AOP con Hibernate DAO	FinderIntroductionInterceptor. org.springframework.aop.support.DefaultIntroductionAdvisor

### CRC9

<b>FinderIntroductionInterceptor</b>	
Conecta la Spring AOP con Hibernate DAO	org.springframework.aop.IntroductionInterceptor

### Tarjetas CRC para la creación de lógica de negocio.

Estas clases van a ser la continuación de los DAO. Solo que proporcionarán transacciones a las operaciones.

### CRC10

<b>Service</b>	
Proporcionar métodos comunes a todos los Daos que no soportan Template (java 1.4 -).	ServiceImpl. Clase implementa el interface para usar el framework de Hibernate..

### CRC11

<b>GenericService&lt;T, ID extends Serializable&gt;</b>	
Clase implementa el interface para usar el framework de Hibernate.	GenericServiceImpl<T, ID extends Serializable> Clase implementa el interface para usar el framework de Hibernate..

## CRC12

<b>BaseHibernateDAOImpl</b>	
Clase implementa el interface para usar el framework de Hibernate..	HibernateDaoSupport Implementación de Spring para trabajar con Hibernate

## CRC13

<b>GenericHibernateDAO&lt;T, PK extends Serializable&gt;.</b>	
Clase implementa el interface para usar el framework de Hibernate..	HibernateDaoSupport Implementación de Spring para trabajar con Hibernate

## Tarjetas CRC para la creación de objetos de dominio o model.

## CRC14

<b>BaseObject</b>	
Clase Padre de la que heredarán todos los objetos de dominio que proporciona métodos comunes a todas los modelos, como toString, equals, etc.	

## Tarjetas CRC para la creación de excepciones.

## CRC15

<b>BaseException</b>	
Clase Padre de la que heredarán todos las excepciones genéricas.	



### CRC16

<b>BaseRuntimeException</b>	
Clase Padre de la que heredarán todos las excepciones en tiempo de ejecución.	

### CRC17

<b>BaseRuntimeException</b>	
Clase Padre de la que heredarán todos las excepciones en tiempo de ejecución.	

### CRC18

<b>ConfigException</b>	
Clase Padre de la que heredarán todos las excepciones de configuración.	

### CRC19

<b>DebuggableException</b>	
Clase Padre de la que heredarán todos las excepciones de depurables.	

### CRC20

<b>DebuggableRuntimeException</b>	
Clase Padre de la que heredarán todos las excepciones de depurables en tiempo de ejecución.	

### 7.3.3.3. Diagramas de Clase UML.

A continuación se presentan los diagramas UML de la librería que nos facilitará la creación de aplicaciones.

#### Interfaces genéricos DAOs

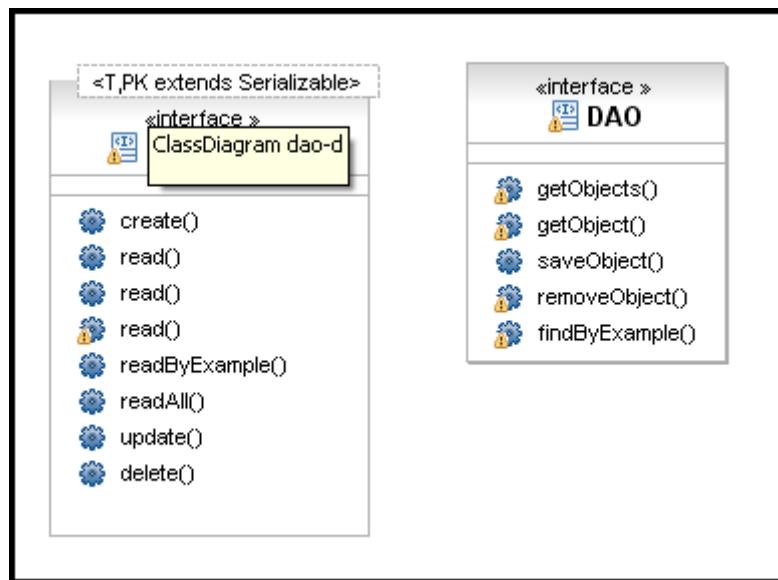


Figura 154 Interfaces genéricos DAOs

#### Interfaces finder para usar Spring AOP con los DAO



Figura 155 Interfaces finder para usar Spring AOP con los DAO

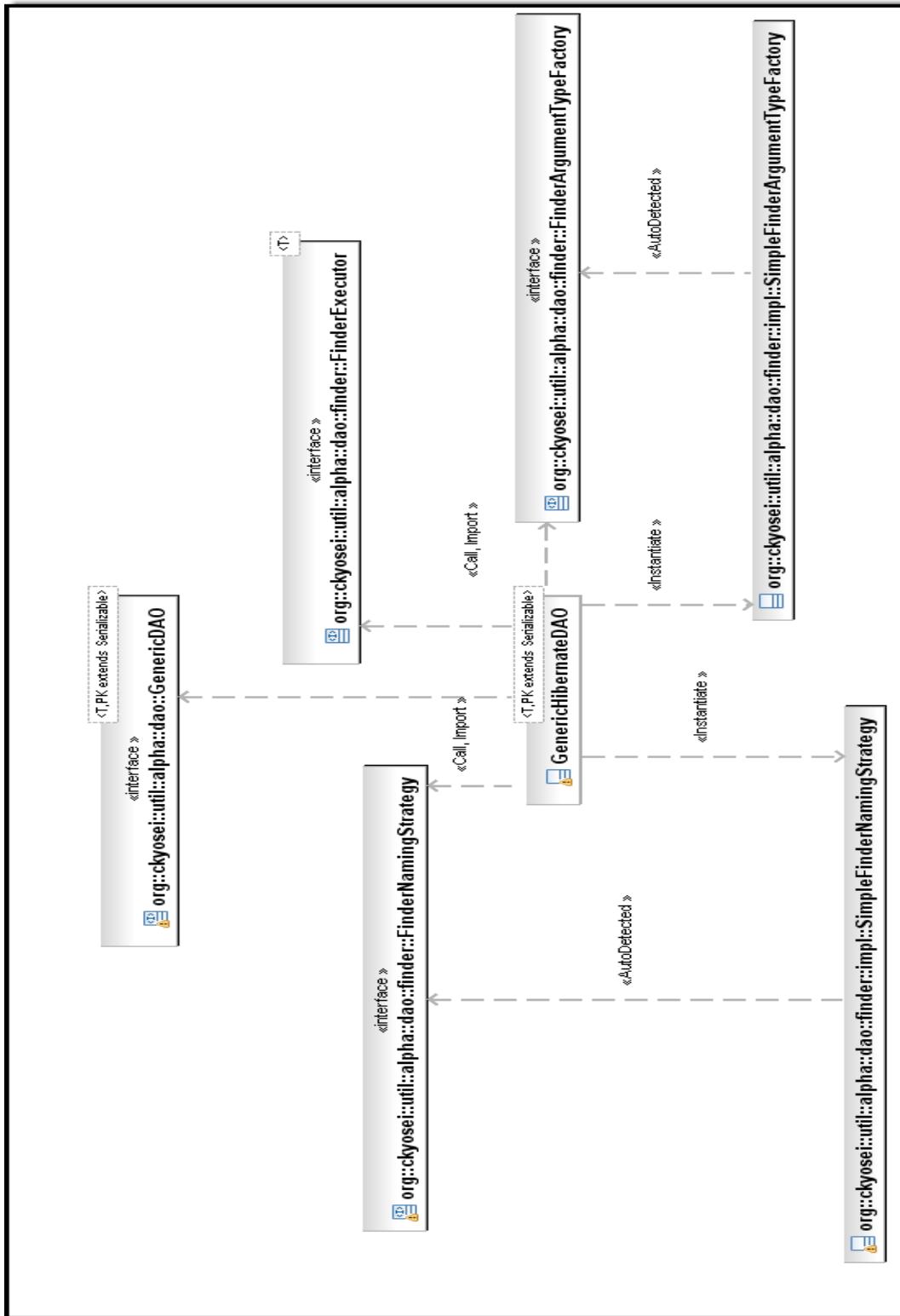


Figura 156 Diagrama de dependencias genericdao

## Implementación de los DAO

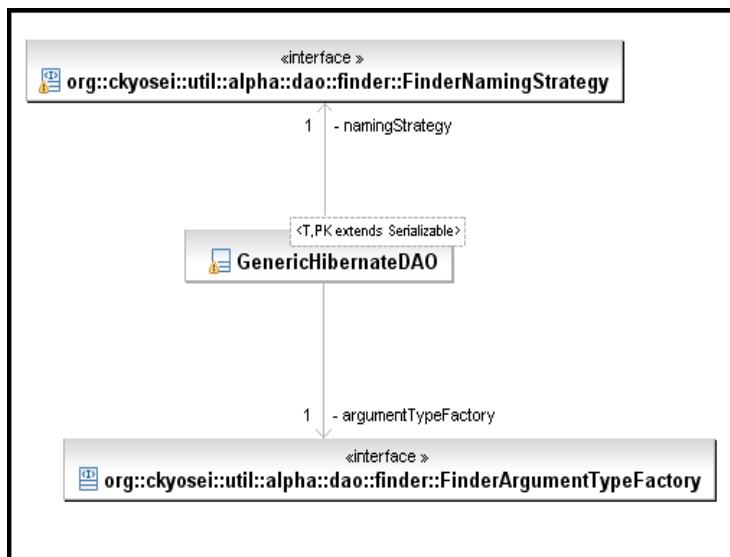


Figura 157 Diagrama de asociación de genericdao



Figura 158 Diagrama de dependencias de BaseHibernateDAOImpl

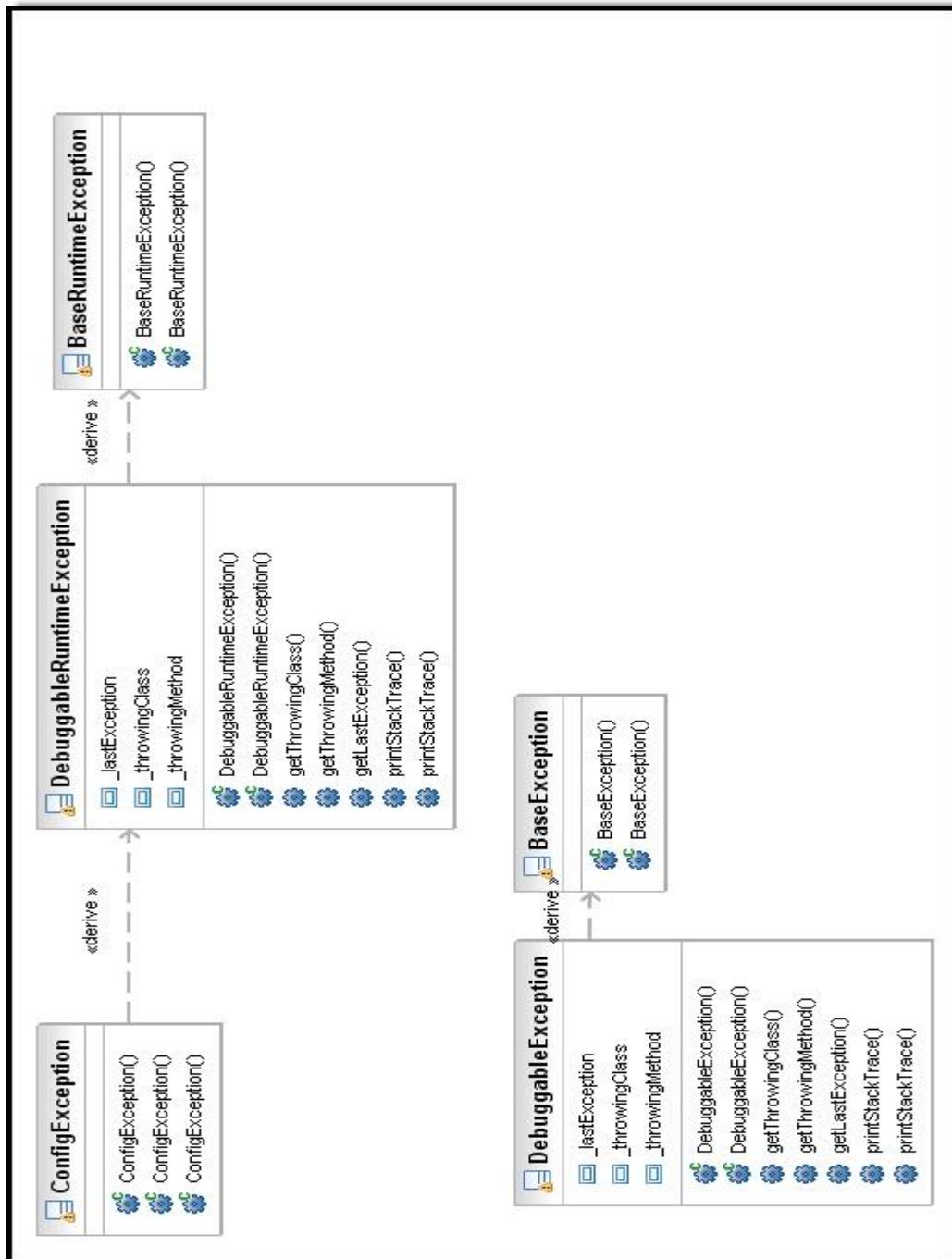


Figura 159 Diagrama de Dependencias paquete exception

## Interfaces de la lógica de negocio



Figura 160 Interfaces de la lógica de negocio

## Diagrama de dependencia de la Implementación de GenericService

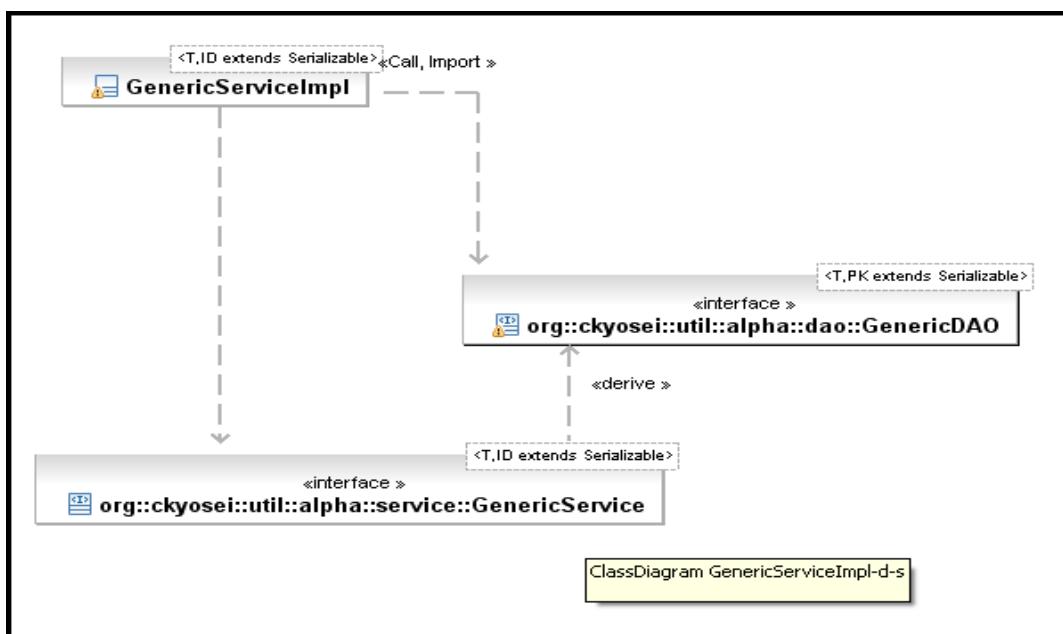
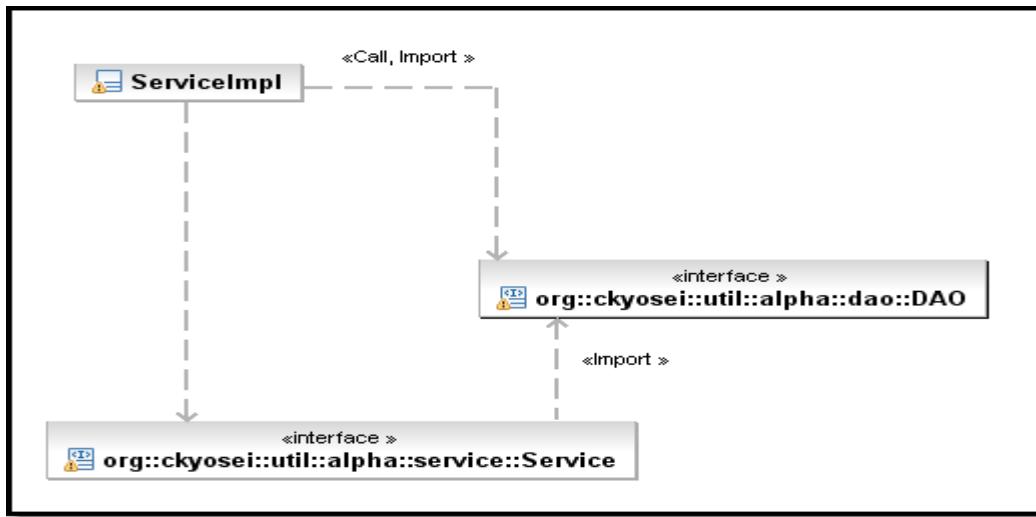


Figura 161 Diagrama de dependencia de la implementación de genericservice



**Diagrama de dependencias de la implementación de Service.**



**Figura 162 Diagrama de dependencias de implementación de service**

## 7.3.4. Aplicando XP y AMDD a la aplicación alpha

### 7.3.4.1. Arquitectura de la aplicación

La aplicación va a ser generada a partir del arquetipo por lo que la información referente al arquetipo, es válida en este punto.

Para crear la aplicación usaremos el arquetipo, creado para la realización de aplicaciones del sistema:

```
C:\CKyosei\Workspace-arquetipo>mvn archetype:create -  
DarchetypeGroupId=org.ckyosei.archetype -  
DarchetypeArtifactId=CkyoseiArchetype -DarchetypeVersion=1.2 -  
DgroupId=test.aplicacion.nueva -DartifactId=alpha-web  
[INFO] alpha-web created in dir: C:\CKyosei\workspace\alpha-web  
[INFO] -----  
--  
[INFO] BUILD SUCCESSFUL  
[INFO] -----  
--  
[INFO] Total time: 4 seconds  
[INFO] Finished at: Thu Sep 18 17:26:46 CEST 2008  
[INFO] Final Memory: 4M/9M  
[INFO] -----  
--
```

La estructura generada será idéntica a la del ejemplo de creación de aplicaciones con el arquetipo ya visto.



### 7.3.4.2. Modelo de dominio

El modelo de dominio del aplicativo alpha va a ser una combinación, de los modelos de dominio suministrados por el arquetipo que nos suministraban, la funcionalidad sobre los usuarios, autenticación y permisos, y los modelos propios de la funcionalidad que se va a generar para el propio aplicativo.

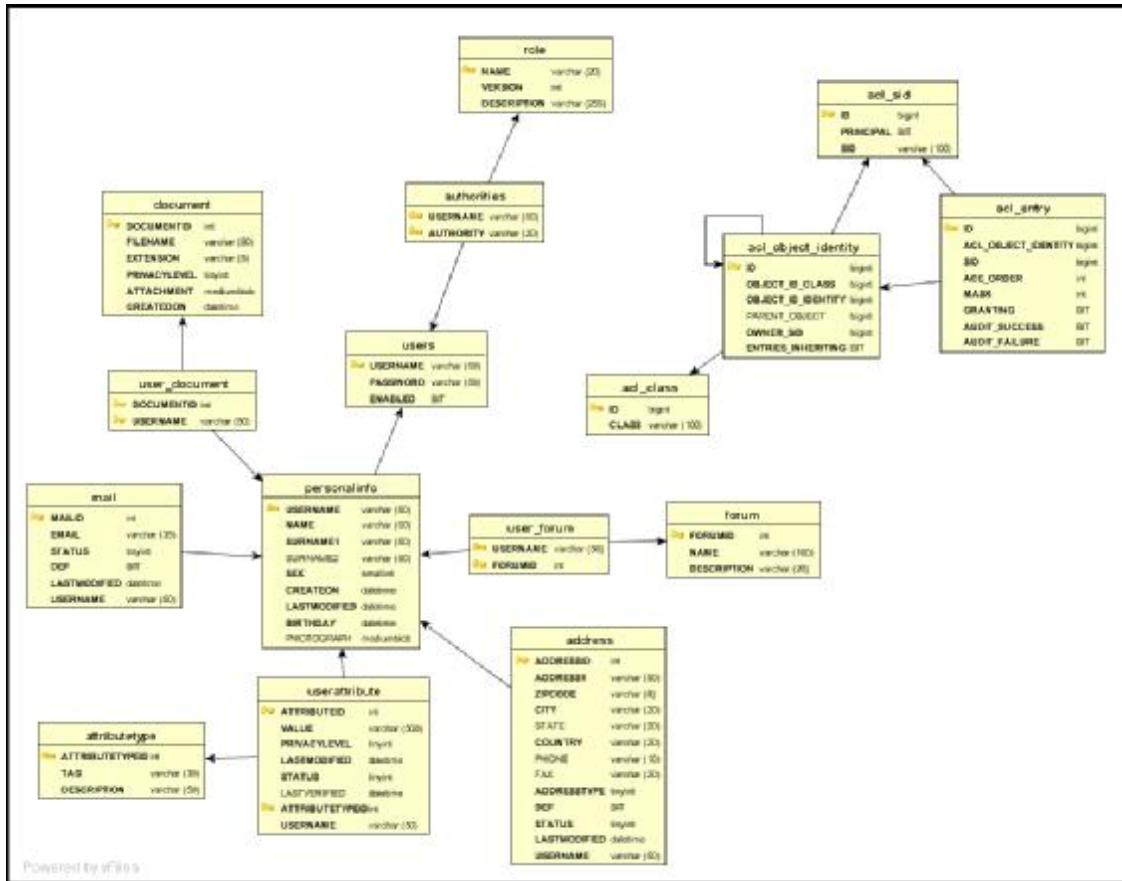


Figura 163 Modelo de dominio del aplicativo alpha

Para más información acerca del modelo lógico ver el apéndice “Modelo lógico alpha” al final del documento.

### 7.3.4.3. Prototipos de interfaces de usuario

Las pantallas del aplicativo alpha están basadas en los estilos definidos para el arquetipo.

En el aplicativo, vamos a tener 3 tipos de pantallas

- Ú Pantallas públicas: Accesible por todos los usuarios, anónimos, Usuario, Administrador.
- Ú Pantallas para el rol usuario. Sólo accesible para el rol Usuario.
- Ú Pantallas para el rol administrador .Sólo accesible para el rol Administrador.

#### Pantalla principal (acceso público)

El menú superior de navegación global permite ir al portal de CKYOSEI.ORG.

Mediante los menús verticales podemos dirigirnos a la página principal de la U.A.H, o ver información sobre Ckyosei.

Usaremos el CSS. *layout-navtop-localleft.css*.

Que incluye el menú de navegación global. Y el menú de navegación local a la izquierda.



Figura 164 Pantalla principal del aplicativo alpha



### Pantalla de validación de usuarios para acceso a zona segura (acceso público)

En la pantalla de validación de usuarios utilizaremos el sistema de capas donde se desactivan los menús globales y de navegación local.

El CSS que permitirá esto es *layout-2col.css*

Por favor actualice su información mediante el siguiente formulario.

Usuario\*

Clave \*

Enviar

Registrarse

© Copyright 2008 ckyosei

Figura 165 Pantalla de validación de usuarios aplicativo alpha

### Pantalla registro para usuarios no autenticados en el sistema (acceso público)

Usaremos el sistema de capas igual que en la pantalla anterior.

El CSS que permitirá esto es *layout-2col.css*

Datos de registro

Por favor introduzca sus datos mediante el siguiente formulario. Una vez completada la operación recibirá un correo que permitira activar su usuario.

Usuario\*

Clave \*

Enviar

© Copyright 2008 ckyosei

Figura 166 Pantalla de registro de usuarios aplicativo alpha

## Pantalla principal para el Rol usuario ya validado. (Acceso restringido)

Usaremos el CSS. *layout-navtop-localleft.css*.



Figura 167 Pantalla de menú del Rol Usuario

## Pantalla principal para el Rol administrador ya validado.(Acceso restringido)

Usaremos el CSS. *layout-navtop-localleft.css*.



Figura 168 Pantalla de menú del Rol Administrador



**Pantalla de Datos básicos común para ambos roles.(Acceso restringido)**

Mediante esta pantalla rellenaremos el perfil del usuario autenticado.

Usaremos el CSS. *layout-navtop-localleft.css*.

Está en: | Menú privado

▲ Menú principal

Datos Básicos

Por favor actualice su información mediante el siguiente formulario.

Avatar

Nombre\*

NOMBRE USUARIO

Apellido 1\* Apellido 2\*

APELLIDO1 APELLIDO2

Fecha Nac.\* sex\*

10/10/1960 HOMBRE

Email\*

CORREO@CORREO.ES

Dirección\*

CALLE ALCALA 12 BAJO 5

Ciudad\* CP\*

Madrid 28001

País\*

ESPAÑA

Enviar

Información

Perfil usuario

Rellene la información de perfil de usuario.

**Figura 169 Pantalla de datos básicos de ambos roles**

## Pantalla de Listado de foros común para ambos roles.(Acceso restringido)

Esta pantalla muestra el listado de todos los foros disponibles en el sistema. Estos foros solo pueden ser creados por el administrador.

Haciendo click en el nombre del mismo, nos podemos suscribir al mismo.

Usaremos el CSS *layout-navtop-localleft-1col.css*, que permite 1 sola columna de contenido.

Nombre	Descripción
foro1	foro1
foro2	foro2
foro3	foro3
foro4	foro4

Figura 170 Pantalla de Listado de Foros de ambos Roles.

## Pantalla de Mis foros común para ambos roles.(Acceso restringido)

Esta pantalla permite la suscripción a un foro por parte de un usuario. Para anular la suscripción al mismo, solo tendremos que hacer click en el nombre del mismo.

La suscripción a un foro permite compartir documentos con los miembros del mismo.

Usaremos el CSS *layout-navtop-localleft-1col.css*



**Figura 171 Pantalla de Mis Foros de ambos Roles.**

### Pantalla de Mis documentos común para ambos roles.(Acceso restringido)

Esta pantalla permite la administración de los documentos de un usuario.

Esta pantalla mostrará 2 tipos de documentos.

- ü Documentos que son propiedad del usuario conectado.
- ü Documentos que están autorizados a este usuario.

Además desde esta pantalla se podrá ir a subir documentos, borrar, ver usuarios autorizados para un documento o por último autorizar.

Usaremos el CSS *layout-navtop-localleft-1col.css*



© Copyright 2008 ckyosei

**Figura 172 Pantalla mis documentos ambos roles**

### Pantalla Subir documentos común para ambos roles.(Acceso restringido)

Mediante la pantalla siguiente se permite la subida de 2 tipos de documentos.

Imágenes *JPG* o Archivos *DOC*. Solo se han elegido dos extensiones para que sirvieran de ejemplo aunque se podría modificar el controlador para que se aceptarán todas las deseadas.

© Copyright 2008 ckyosei

**Figura 173 Pantalla de subida de documentos ambos roles**



### Pantalla permisos sobre documentos común para ambos roles.(Acceso restringido)

Realizando un click en la pantalla Mis documentos en Autorizados obtenemos todos los usuarios autorizados a ver el documento. Haciendo Click sobre el nombre le eliminaremos los permisos, en caso de ser propietarios del documento.

Usaremos el CSS **layout-navtop-localleft-1col.css**

**Figura 174 Pantalla de permisos sobre documentos ambos roles**

### Pantalla de autorización sobre documentos común para ambos roles.(Acceso restringido)

La autorización de documentos se va a realizar en 2 pasos principales.

- ü Selección del Foro del cual forma parte la persona que queremos autorizar.(Solo se permite autorizar a usuarios de un foro del cual forma parte el usuario autorizado)
- ü Selección de usuario dentro del foro seleccionado en el paso previo.

Usaremos el CSS **layout-navtop-localleft-1col.css** para ambas pantallas.

**Ciudades Kyosei**

operación1 | operación2 | operación3 | operación4 | operación5

Está en: | Mis Foros

▲ Menú principal

- Datos Básicos**
- Listado Foros**
- Mis foros**
- Mis documentos**

**Listado Foros Subscrito**

Seleccione un foro de la lista.

Total: 3 Mostrados todos los Foros.

Nombre	Selección
foro1	<input checked="" type="checkbox"/>
foro2	<input type="checkbox"/>
foro3	<input type="checkbox"/>

**Enviar**

© Copyright 2008 ckyosei

**Figura 175 Listado de foros suscrito**

**Ciudades Kyosei**

operación1 | operación2 | operación3 | operación4 | operación5

Está en: | Listado Usuarios

▲ Menú principal

- Datos Básicos**
- Listado Foros**
- Mis foros**
- Mis documentos**

**Listado Usuarios Subscriptos al foro**

Seleccione un usuario a autorizar.

Total: 3 Mostrados todos los Usuarios.

	NOMBRE	APELLIDO	APELLIDO
<input checked="" type="checkbox"/>	ROBERTO	ZAPATERA	PILO
<input type="checkbox"/>	PEDRO	PRIETO	MARTÍN
<input type="checkbox"/>	NOMBRE USUARIO	APELLIDO1	APELLIDO2

**Enviar**

© Copyright 2008 ckyosei

**Figura 176 Selección de usuarios para autorizar**



### Pantalla de creación de Foros para el Rol Administrador.(Acceso restringido)

Mediante esta pantalla podremos crear nuevos foros. Estos aparecerán en la opción listado de Foros una vez creados y permitirán a los usuarios suscribirse a ellos.

Figura 177 Pantalla de creación de Foros para el Rol Administrador

#### 7.3.4.4 Storyboard

El Storyboard va a ser una combinación del ya visto en el arquetipo, más la nueva funcionalidad de acciones de la parte privada vista en el prototipo de ventanas.

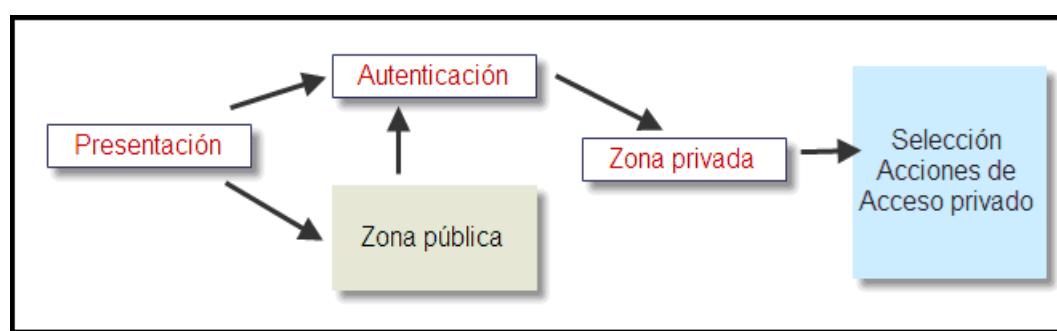


Figura 178 StoryBoard parte pública aplicación alpha

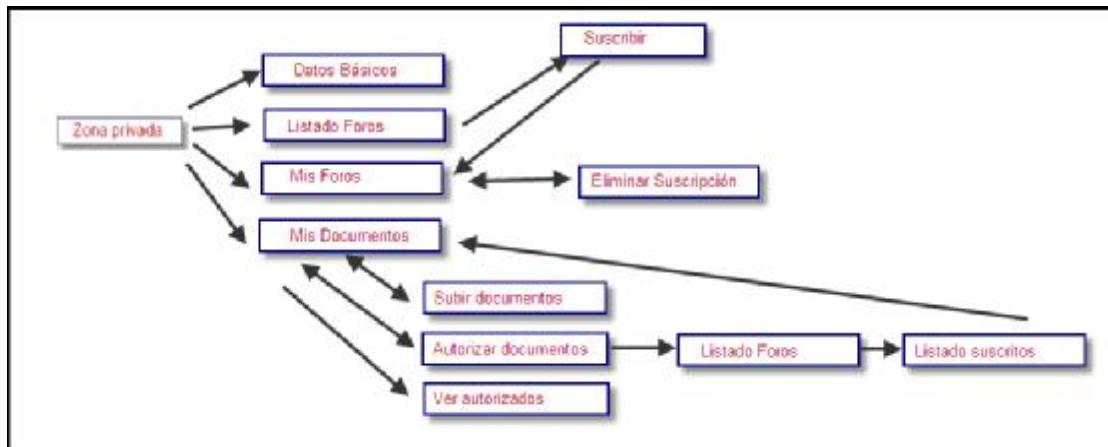


Figura 179 StoryBoard parte privada aplicación alpha

### 7.3.4.5. Plan de Entregas

Iteración	FUNCIONALIDAD	FECHA
0	Pegueña release con prioridad 1, 2, 3 historias de usuario	Por definir
1	Pegueña release con prioridad 4	Por definir
2		Por definir
3	Pegueña release con prioridad 4	Por definir
4	Pegueña release con prioridad 5	Por definir
5	Pegueña release con prioridad 6	Por definir
6	Pegueña release con prioridad 6	Por definir

## **8 CONCLUSIÓN**

---





## 8. Conclusión

El Software libre, es aquel que puede ser distribuido, modificado, copiado y usado; cualquier persona que lo deseé, esto implica que debe estar acompañado de del código fuente y de una mínima documentación, para hacer efectivas las libertades que lo caracterizan.

Ahora bien, el SF libre, tiene ciertas desventajas sobre el propietario, que hay que tener muy claras antes de embarcarse en proyectos de envergadura apoyándose en el mismo.

No se tienen garantías del autor ni de quien lo provee, y no existen compañías únicas que respalden toda la tecnología. Además esta el inconveniente mayor de todos, que es el cambio constante de las versiones del mismo.

La mayoría de las configuraciones del SF, no son intuitivas, y están poco documentadas, se necesita dedicar recursos a la reparación de errores, la mayoría de sus soportes están en los foros de la Web y ante un error o Bug, lo único que puedes hacer es intentar repararlo o comunicarlo hasta que se solucione en una versión posterior.

Por lo que hay que tener claro una serie de recomendaciones antes de implantar un proyecto de SF libre:

- Ü Elegir proyectos con futuro y pasado. (Proyectos que siguen desarrollándose, y que no son productos puntuales que surgen en un momento dado, que terminarán abandonándose).
- Ü Elegir proyectos con una suficiente documentación como para no perderte en un laberinto del que es muy difícil salir, y que hacen que los costes se disparen por tener que asignar muchos recursos ante un problema puntual.

Durante la realización de este proyecto, se me han planteado varios retos, tenía que implantar una plataforma tecnológica, que fuera lo suficientemente innovadora, para abarcar el máximo de proyectos de nueva generación, a la vez que debíamos poder implantar otros proyectos ya existentes de SF libre.

Así mismo esta plataforma debe de seguir viva en el tiempo, por lo que tenía que elegir proyectos que se sigan desarrollando, corrigiendo fallos y evolucionando. Como los proyectos de Apache.

Con las tecnologías de desarrollo me encontré el mismo problema, que lenguajes elegir y que metodología de trabajo, no podía elegir una tecnología que pareciera muy puntera y que en 2 días nadie la utilizara, tenía que elegir un lenguaje muy potente, vivo, y que sigue creciendo como Java.

Por lo tanto todos los proyectos de SF libre que se han elegido para este proyecto han sido fruto del azar han sido seleccionados, por criterios de uso actual, pasado y sobre todo futuro.

En este proyecto hemos planteado la base tecnológica, la metodología de trabajo, y hemos creado el primer ladrillo de lo que deberá ser el Sistema kyosei-Polis, un marco de participación para todos.

Un Entorno Virtual de Participación Ciudadana donde tanto los ayuntamientos como los ciudadanos y sus asociaciones pueden poner a disposición de los ciudadanos, otros colectivos, medios de comunicación, etc. todas las informaciones referidas a las actividades participativas que realizan en la ciudad.



## **9 FUTURAS LÍNEAS DE TRABAJO**

---





A partir del entorno tecnológico implantado en el proyecto final de carrera se pretende crear, un Entorno Virtual de Participación Ciudadana donde tanto los ayuntamientos como los ciudadanos y sus asociaciones pueden poner a disposición de los ciudadanos, otros colectivos, medios de comunicación, etc. todas las informaciones referidas a las actividades participativas que realizan en la ciudad.

Además, el Entorno Virtual de Participación Ciudadana ofrecerá, en función de las necesidades del municipio, herramientas especiales y espacios virtuales que complementen y apoyen sus actividades participativas presenciales, como por ejemplo:

- ü Envío de preguntas a los distintos departamentos del ayuntamiento u organizacion;
- ü Encuestas y cuestionarios, para que el ciudadano transmita su opinión sobre ciertos temas;
- ü Procesos de recogida de firmas para realizar una petición o iniciativa legislativa;
- ü Consulta de la "agenda participativa" del municipio o barrio, que es elaborada colaborativamente por todos los colectivos de la ciudad;
- ü Espacios para la realización virtual de procesos participativos complejos;
- ü Foros de debate con modelos avanzados de moderación que permitan realizar discusiones estructuradas y de calidad, identificar los consensos existentes, resolver posibles desacuerdos y elaborar documentos de conclusiones;
- ü Espacios privados de colaboración que faciliten el trabajo participativo interno del ayuntamiento y las organizaciones ciudadanas;
- ü Publicación y revisión participativa de documentos;
- ü Herramientas que faciliten que ciudadanos interesados por un cierto tema se pongan en contacto, tc.



## **10 PRESUPUESTO**

---





## ***10. Presupuesto***

---

El presente apartado contiene la estimación presupuestaria realizada para el desarrollo de este proyecto.

## **10.1. Introducción**

La estimación del presupuesto de este proyecto se realiza teniendo en cuenta dos pautas: por un lado se considera el precio de los equipos necesarios para el desarrollo del proyecto, y por otra parte se tiene en cuenta el coste que supone el tiempo empleado en la ejecución de dicho trabajo.



## 10.2. Presupuesto de Ejecución Material

Se denomina coste total de la ejecución material, a la suma del coste de los equipos y del coste por tiempo de trabajo.

### Coste de equipos

EQUIPO	PRECIO	DURACIÓN	USO	TOTAL
<b>Ordenador</b>	1200 euros	3 años	7 meses	233,00 Euros
<b>Impresora</b>	240 euros	3 años	1 mes	6,66 Euros

<b>Coste total de equipos:</b>	240 Euros
--------------------------------	-----------

### Coste por tiempo de trabajo

Como hemos visto en las Historias de usuario necesitamos 74,25 puntos para completar todas las tareas. Tomando de partida 1 punto como 6horas.

FUNCIÓN	Nº HORAS	EUROS/HORA	TOTAL
<b>Ingeniería</b>	445	36.061	16020 Euros
<b>Mecanografía</b>	208	15,02	3125.26 Euros

<b>Coste por tiempo de trabajo:</b>	19145,26 euros
-------------------------------------	----------------

### **Coste total de ejecución material**

Es la suma de los importes del coste de materiales y de la mano de obra.

<b>CONCEPTO</b>	<b>COSTE</b>
<b>Coste de equipos:</b>	240 Euros
<b>Coste por tiempo de trabajo:</b>	19145,26 Euros
<b>Coste de ejecución material:</b>	19385,26 Euros



## 10.3. Gastos Generales y Beneficio Industrial

Normalmente se trata de los gastos necesarios para disponer de instalaciones en las que desempeñar el trabajo, además de otros gastos adicionales. Los gastos generales y el beneficio industrial son el resultado de aplicar un recargo del 22% sobre el Coste Total de Ejecución Material, resultando:

Gastos Generales y Beneficio Industrial..... 4264,7572 Euros

### 10.3.1. Presupuesto de Ejecución por Contrata

El presupuesto de ejecución por contrata es el resultado de sumar el coste total de ejecución y los gastos generales.

Presupuesto de Ejecución por Contrata..... 23650,02Euros

### 10.3.2. Honorarios

Los honorarios facultativos por la ejecución de este proyecto se determinan de acuerdo a las tarifas de los honorarios de los ingenieros en trabajos particulares vigentes a partir del 1 de Septiembre de 1997, dictadas por el Colegio Oficial de Ingenieros Técnicos de Telecomunicación.

Importe	Coeficiente reductor	Porcentaje
Hasta 5 millones	C = 1	7 %
Desde 5 millones	C = 0,9	7 %

Los derechos del visado se calcularán aplicando la siguiente fórmula:  $0,07 \times P \times c$ , donde P es el presupuesto de ejecución material y c es el coeficiente reductor.

#### DERECHOS DE VISADOCOSTE

$0,07 \times 19385,26 \times 0,9 = 1221,27$

Total honorarios: 1221,27

## **10.4. Importe total del presupuesto**

El importe total del presupuesto de este proyecto se calcula sumando el presupuesto de ejecución por contrata y los honorarios. A dicho valor se le aplicará el 16% de IVA.

Presupuesto de Ejecución por Contrata 23650,02 Euros

Honorarios..... 1221,27euros

SUBTOTAL..... 24871,29euros

16% de IVA..... 3979,4064 euros

IMPORTE TOTAL ..... 28850,69 Euros

El importe total del proyecto asciende a la cantidad de..... 28850,69 Euros.

## **11 BIBLIOGRAFÍA**

---





## 11. Bibliografía

- [1] Cerami, E. (2002) *Web services essentials distributed applications with XML-RPC, SOAP, UDDI & WSDL*. O'Reilly, Indianapolis, EEUU.
- [2] Harrop, R., Machacek, J. (2005) *Pro Spring*. Apress, UK.
- [3] Hemrajani, A. (2006) *Agile Java Development with Spring, Hibernate and Eclipse*, Sams, EEUU.
- [4] Johnson, R., Hoeller, J., Arendsen, A., Risberg, T., Sampaleanu, C. (2005) *Professional Java Development with the Spring Framework*. Wrox., EEUU.
- [5] Johnson, R., Hoeller, J. (2004) *Expert One-on-One J2EE Development without EJB*. Wrox., July. EEUU.
- [6] Ladd, S., Davison, D., Devijver, S., Yates C. (2006) *Expert Spring MVC and Web Flow*, Apress, UK.
- [7] Miller, P. y Webb, M. (2007) "Flesh, steel and Wikipedia. How government can make the most of online collaborative tools", The Collaborative State. How working together can transform public services. S. Parker y N. Gallagher (Eds.), Edinburgh, International Teledemocracy Centre, Napier University.
- [8] Raible, M. (2005) *Spring Live*. SourceBeat LLC., Colorado, EEUU.
- [9] Tate, B.A. y Gehtland, J. (2004) *Better, Faster, Lighter Java2*. O'Reilly, EEUU.
- [10] Tate, B.A. y Gethland, J. (2005) *Spring: A Developer's Notebook*. O'Reilly, EEUU.
- [11] Walls, C., Breidenbach, R. (2005) *Spring in Action*. Manning Publications, EEUU.
- [12] Miyata, C. (1998), "La informática más accesible para los discapacitados. Regulación de estándares por Cocemfe y AENOR", *Computerworld*, 756, May, 29, <http://www.idg.es/computerworld/articulo.asp?id=52422>.
- [13] Moreno, M.J. (1998), "El acceso de los usuarios discapacitados a las paginas Web: Análisis comparativo del estado de la cuestión en América del norte y Europa", FESABID, *VI Jornadas Españolas de Documentación*, [http://fesabid98.florida-uni.es/Comunicaciones/mj\\_moreno/mj\\_moreno.htm](http://fesabid98.florida-uni.es/Comunicaciones/mj_moreno/mj_moreno.htm).
- [14] NI4 (2005), "Ley 34/2002 sobre accesibilidad en Administraciones Pùblicas", Artículos: Soluciones NI4, <http://www.ni4.org/modules.php?name=News&file=article&sid=39>
- [15] The Apache Software Foundation (2008-2009), *Maven Getting Started Guide*, <http://maven.apache.org/guides/index.html>.
- [16] The Apache Software Foundation (2009), *Apache HTTP Server Version 2.2 Documentation* <http://httpd.apache.org/docs/2.2/>.
- [17] The Apache Software Foundation (2008), *Apache Tomcat 6.0*, <http://tomcat.apache.org/tomcat-6.0-doc/index.html>.
- [18] The Apache Software Foundation (2008-2009), *The Apache Tomcat Connector*, <http://tomcat.apache.org/connectors-doc/>.
- [19] Porter, B., Odea, M.. (2009) *Apache Maven 2: Effective Implementation*. Packt Publishing, EEUU
- [20] Van Zyl, J., Fox, B., Casey, J., Snyder, J., O'Brien, T., Redmond, E. (2009) *Maven: The Definitive Guide*. O'Reilly., EEUU
- [21] O'Brien,T., Massol, V. (2005) *Maven: A Developer's Notebook*. O'Reilly., EEUU
- [22] Elliott, J. (2004) *Hibernate: A Developer's Notebook*. O'Reilly., EEUU
- [23] Elliott, J. (2008) *The Criteria API - Harnessing Hibernate*. O'Reilly., EEUU
- [24] Elliott, J. (2008) *Getting Started with Hibernate 3, 1st Edition*. O'Reilly., EEUU

[25] Elliott, J., O'Brien, M. (2009) *Hibernate Types - Harnessing Hibernate*. O'Reilly., EEUU

## **12 GLOSARIO DE TÉRMINOS**

---





**A**

**Apache:** servidor web de código abierto.

**B**

**BER:** el formato Basic Encoding Rules (BER) es uno de los formatos de codificación definidos como parte de la norma ASN.1. Para certificados digitales. Se trata de una norma para la codificación de información abstracta en una transmisión de datos concreta. Dos subconjuntos de BER y DER (Distinguished Encoding Rules) y CER (Canonical Encoding Rules).

**C**

**.cer, .crt:** un archivo con la extensión .crt o .cer contiene un certificado X.509 único que utiliza el formato Distinguished Encoding Rules (DER), un conjunto de reglas definidas por la norma ASN.1 para dar formato a datos binarios.

**D**

**DNS:** El Domain Name System (DNS) es una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio

**DER:** el formato Distinguished Encoding Rules (DER) se utiliza en criptografía para asegurar que los datos, como los de un certificado X.509, que debe llevar una firma digital, producen una representación en serie exclusiva. El formato DER hace posible firmar y verificar certificados X.509.

**F**

**framework:** es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.

**FMNT:** Fabrica Nacional de Moneda y Timbre. Entidad certificadora Española.

**H**

**Hibernate:** Framework de persistencia de datos.

**J**

**ISP:** Proveedor de servicios de Internet (Internet Service Provider)

**JSF:** Acronimo de Java Server Faces. es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.

**JSP:** es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

**M**

**Maven:** herramienta de software para la gestión y construcción de proyectos Java.

**Mysql:** MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario

## O

**OpenSSL:** Consiste en un robusto paquete de herramientas de administración y librerías relacionadas con la criptografía

## P

**PHP:** es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas

**plugins:** es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica

**.p7b, .p7c, .p7s:** los archivos con una extensión .p7b tienen un mensaje según la norma de criptografía de claves públicas núm. 7 (PKCS#7) que contiene uno o más certificados X.509. El estándar de correo seguro S/MIME utiliza PKCS#7 para sus mensajes firmados digitalmente y encriptados.

**.p12, .pfx:** los archivos con una extensión .p12 tiene un formato de archivo encriptado según lo establecido en la norma de criptografía de claves públicas núm. 12 (PKCS#12). Se trata de un formato portable para el almacenamiento o transporte de las claves privadas, certificados y secretos varios de un usuario. Este estándar admite la transferencia directa de información personal en diferentes modos de privacidad e integridad, desde el uso de claves públicas/privadas hasta una menor seguridad en la forma de un sistema de privacidad basado en contraseñas.

**.pem:** el .pem o Privacy Enhanced Mail es un formato para el envío y recepción de correo electrónico seguro mediante el uso de criptografía de claves públicas. Se trata de una codificación únicamente de texto que permite su uso en las cabeceras de los mensajes S/MIME. El formato .pem es una codificación de tipo base64 del certificado X.509 (formato DER binario) rodeado de cabeceras de texto.

## S

**Spring Framework:** Framework de desarrollo que permite dar cobertura a todo el ciclo de vida del software.