ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

BASE DE DATOS WEB SOBRE CÓDIGOS ÉTICOS PARA PROFESIONALES DE LA INFORMÁTICA Y UNA PROPUESTA ECLÉCTICA

Realizado por

MIGUEL ÁNGEL SERNA MATA

Dirigido por

JOSÉ GALINDO GÓMEZ

Departamento

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

MÁLAGA, DICIEMBRE 2005

UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Reunido el tribunal examinador en el día de la fecha, constituido por:

Presidente Dº/Dª. _____

Secretario Dº/Dª. _____

Vocal Dº/Dª. _____

para juzgar el proyecto Fin de Carrera titulado:

BASE DE DATOS WEB SOBRE CÓDIGOS ÉTICOS PARA PROFESIONALES DE LA INFORMÁTICA Y UNA PROPUESTA ECLÉCTICA

del alumno Dº/Dª. Miguel Ángel Serna Mata

dirigido por Dº/Dª. José Galindo Gómez

ACORDÓ POR _____ OTORGAR LA CALIFICACIÓN DE _____

Y PARA QUE CONSTE, SE EXTIENDE FIRMADA POR LOS COMPARECIENTES DEL TRIBUNAL, LA PRESENTE DILIGENCIA.

Málaga, a de del 200__

El Presidente El Secretario El Vocal

Fdo: Fdo: Fdo:

Agradecimientos y dedicatorias

Deseo expresar mi agradecimiento más sincero a José Galindo Gómez, tutor de este proyecto, por su dedicación, implicación y paciencia. También quiero dedicar este trabajo y agradecer el apoyo de mi familia, de mi amigo Carlos y de mi novia María José.

Índice general

Introducción

A lo largo de toda su vida, el ser humano se enfrenta continuamente a dilemas éticos, en los que se pregunta si el realizar o no cierta acción es bueno o malo, correcto o incorrecto. Debe, por tanto, ser capaz de discernir el bien del mal en incontables ocasiones. De ahí la importancia que tiene la ética, la parte de la filosofía que se ocupa del obrar del hombre, de sus acciones.

En sus orígenes (Aristóteles e incluso sus predecesores), la ética se preocupaba de cómo actuar para alcanzar la propia felicidad. En la actualidad se entiende también como la forma de actuar para respetar a los demás y a sus intereses, o incluso para respetar la naturaleza y sus formas de vida. Aunque a primera vista pueda parecerlo, en cierta forma, estos puntos de vista no son contradictorios, sino complementarios. El primero de ellos es el más básico y puede ser visto como un punto de vista «egoísta» en el que el ser humano sólo se preocupa de sí mismo y de su bienestar personal. En cambio, los otros dos puntos de vista tienen un concepto más «amplio» de felicidad en el que intervienen otros elementos, como son el resto de la humanidad y la naturaleza en su conjunto (incluidos todos los seres vivos que la habitan). Estos dos criterios implican que la felicidad personal es imposible si no se incluye en ella la de los demás, complementando así al primer criterio.

Una parte muy importante de la vida del hombre es la que éste dedica a ejercer una profesión. Por ello, no tuvo que pasar mucho tiempo para que se empezara a aplicar la ética a cada una de las profesiones humanas, naciendo así la ética profesional. De esta manera, la ética ha ido estableciendo, para cada una de las profesiones, los principios éticos básicos que deben regir el comportamiento profesional de cada uno sus miembros. Las profesiones más antiguas, como son la medicina o el derecho, han tenido muchos siglos para enunciar y revisar sus normas de conducta profesional. En cambio, los profesionales de la informática, una ciencia muy reciente, apenas han dispuesto de tiempo para establecer un marco ético común.

La sociedad actual es una sociedad altamente informatizada. En muy poco tiempo, los ordenadores se han convertido en piezas fundamentales sin las cuales no se podría vivir de la manera en que se vive. La informática, y de forma más general, las Tecnologías de la Información y las Comunicaciones (TIC), han provocado cambios significativos en la estructura económica y social, y en el conjunto de las relaciones sociales. Así, la creciente dependencia, tanto social como económica, de la informática y la imparable evolución de Internet han provocado la aparición de nuevas situaciones y nuevos problemas, así como han transformado o agravado otros.

En este contexto, se hace necesario tener una serie de reglas que ayuden al profesional

informático a enfrentarse a los dilemas éticos. La forma de conseguir esto es creando un código ético en el que se recojan todas estas normas, principios, directrices o pautas de actuación. Un código de ética no es más que un documento escrito desarrollado por una asociación profesional con el propósito de guiar a los profesionales en lo que se refiere a su conducta profesional en determinadas circunstancias.

Motivación y objetivos

Aunque en estos últimos años cada vez más asociaciones informáticas, algunas de la talla de la ACM (*Association for Computing Machinery*) o el IEEE (*Institute of Electrical and Electronics Engineers*), han formulado sus propios códigos deontológicos, a día de hoy no existe un código estándar que sea universalmente aceptado por la profesión informática. Por ello, resulta interesante disponer de una herramienta para almacenar y consultar todos estos códigos.

En este trabajo pretendemos desarrollar una aplicación web de tales características que, además de esto, implemente distintos tipos de búsqueda: por palabras, por organización, etc., aprovechando así las ventajas que nos ofrece el uso de una base de datos. La aplicación desarrollada también deberá poder emplearse para la clasificación de todo tipo de «códigos legales» en cualquier contexto y para cualquier ámbito, independientemente de la estructura de estos códigos (en secciones, artículos...). Estará disponible en español e inglés, pudiendo ser traducida a otros idiomas de una manera fácil y rápida.

Con este proyecto también se persigue recopilar en una base de datos un número razonable de códigos éticos de asociaciones informáticas, prestando especial atención a su traducción al español. Con esto pretendemos ampliar la difusión de los códigos deontológicos a la comunidad hispanohablante.

También realizaremos un estudio global en el que veremos qué temas son los que se tratan en los códigos éticos de las asociaciones profesionales informáticas. El estudio consistirá en clasificar todos los códigos que logremos recopilar, utilizando para ello una lista de temas que elaboraremos para tal fin. Al final, obtendremos una estadística que nos dirá cuantas veces aparece cada tema en los diferentes códigos.

Por último, presentaremos un «código ecléctico» con todo lo aprendido de este estudio. Este código intentará recoger todas las virtudes de los códigos éticos estudiados, subsanando las deficiencias que se observen en los mismos.

Descripción de los capítulos

De una forma rápida y general, podemos describir el contenido de la presente memoria explicando brevemente el contenido de cada uno de los capítulos:

- **Capítulo 1: Códigos éticos informáticos:** Se da una introducción a la ética informáti-

ca. También se habla sobre las tareas de búsqueda, traducción, clasificación y análisis estadístico de los códigos éticos informáticos recopilados.

- **Capítulo 2: Tecnologías web:** En este capítulo se introducen los conceptos y tecnologías relacionados con la web. Se explica cómo funciona la web y se presentan algunos de los estándares web más importantes que actualmente se utilizan en el desarrollo de sistemas y aplicaciones web.

- **Capítulo 3: Herramientas:** Se describen las herramientas software empleadas en la elaboración de la aplicación web del presente proyecto. Todas ellas son de libre distribución. Se incluye una breve introducción al *software open source*.

- **Capítulo 4: MySQL:** En este capítulo se incluye una introducción a MySQL, el sistema gestor de base de datos (SGBD) que hemos utilizado, así como una pequeña comparativa entre el SGBD elegido y otros disponibles en el mercado. Además, se describen brevemente algunos de los posibles programas clientes que pueden ser utilizados para trabajar con MySQL. Por último, se habla de JDBC (*Java Database Connectivity*), la interfaz que nos permite comunicarnos con cualquier SGBD a través del lenguaje de programación Java. Este es el lenguaje que utilizaremos para nuestra aplicación.

- **Capítulo 5: Struts:** Este capítulo es una introducción a Struts, el *framework* que hemos utilizado para construir nuestra aplicación web. Primero, se describen las tecnologías utilizadas por Struts. Después, se presentan los dos tipos de arquitecturas para aplicaciones web Java: Modelo 1 y Modelo 2. Nos centraremos en el Modelo 2, también llamado patrón MVC (*Model-View-Controller*), explicando su estructura y funcionamiento, así como la forma en que Struts implementa dicho patrón. En último lugar, introduciremos los distintos componentes de Struts, describiendo brevemente las funciones que realizan.

- **Capítulo 6: La aplicación web: Codetic:** Este capítulo explica el proceso de desarrollo de la aplicación web del proyecto. Está dividido en tres apartados, cada uno de los cuales se corresponde con una de las distintas fases del proceso: análisis, diseño e implementación.

- **Capítulo 7: Manuales:** Contiene los tres manuales suministrados con la aplicación web: el manual de instalación, que detalla paso a paso la instalación del sistema, el manual de usuario, que explica las distintas pantallas a las que tienen acceso los usuarios públicos o visitantes, y el manual de administración, que documenta la zona de administración de la aplicación web.

- **Capítulo 8: Propuesta ecléctica:** Este capítulo recoge el código ecléctico elaborado como parte de este proyecto. Está dividido en nueve apartados que agrupan las normas de conducta incluidas en el código.

Este trabajo se completa con unos apéndices que incluyen una lista de temas para la clasificación de los artículos de los códigos éticos (Apéndice A) y el código de ética y de conducta profesional de la ACM (Apéndice B).

Capítulo 1

Códigos éticos informáticos

Este capítulo es una introducción a la ética informática y a los códigos éticos elaborados por asociaciones de informática para sus socios. También habla de las tareas de recopilación, traducción y análisis de códigos éticos informáticos que han sido llevadas a cabo para este proyecto.

El capítulo se divide en los siguientes apartados:

- Introducción a la ética informática. Se explica en qué consiste la ética, haciendo especial hincapié en la ética informática. También se habla sobre los códigos éticos y su utilidad [25, 35, 45, 51].
- Traducción y análisis de códigos éticos informáticos. Este apartado describe todo el proceso de búsqueda, traducción, clasificación y análisis de todos los códigos éticos recopilados para este proyecto [1, 5, 26].

1.1. Introducción a la ética informática

En este apartado hablaremos sobre el concepto de ética, centrándonos en la ética aplicada a la informática, así como de los códigos deontológicos desarrollados por asociaciones de profesionales informáticos.

1.1.1. Ética

La ética es la parte de la filosofía que se preocupa de las cuestiones de valor, es decir, se encarga de estudiar la bondad o maldad del comportamiento humano. La ética tiene que ver con las normas, valores, principios morales, etc. que cada uno usa en sus decisiones o actos.

En una sociedad en la que el ordenador cobra cada vez más un papel importante, y dado el importante crecimiento de Internet y el gran desarrollo en las áreas de las técnicas de la información y la informática, es necesario establecer una serie de normas que rijan el comportamiento profesional de los informáticos.

La tecnología informática plantea nuevas situaciones y nuevos problemas, y gran parte de estas nuevas situaciones y problemas son de naturaleza ética. Evidentemente existen intentos de resolver estos problemas aplicando las actuales reglas y soluciones éticas de carácter general, por lo que es importante echar la vista al pasado en busca de los mejores valores éticos tradicionales. Las TIC (Tecnologías de la Información y de las Comunicaciones) cambiarán el mundo que conocemos actualmente, y desde el punto de vista ético, lo más importante y evidente es la necesidad de una nueva ética que estudie y solucione los problemas propios de las TIC. Esta nueva ética será llamada ética de la informática.

1.1.2. Ética Informática: EI

Existen varias definiciones para el concepto de Ética de la Informática o Ética Informática (EI). Parte de las que presentamos en este trabajo están sacadas del artículo «¿Qué es la ética de la informática?», escrito por José María Guibert [35].

De forma restrictiva, podemos considerar la EI como la disciplina que analiza los problemas éticos creados, transformados o agravados por el uso de la tecnología informática. En este punto, la pregunta que se plantean muchos autores es si acaso la EI corresponde a un campo de estudio completamente nuevo, o si en realidad se encuentra como una extensión de antiguos dilemas éticos.

Otras definiciones de la EI son mucho más amplias. No se reducen a un nuevo campo de ética aplicada sino que, por ejemplo, para Moor [51], la EI es el análisis de la naturaleza y el impacto social de las tecnologías de la información y la correspondiente formulación y justificación de políticas para un uso ético de dicha tecnología. Para este autor, la EI está relacionada con el vacío normativo que ha ocasionado la tecnología de la información. El problema, según él, es que existe una falta de reglamentación en cómo utilizar estas nuevas tecnologías que posibilitan nuevas actividades para las cuales no hay o no se perciben con nitidez principios de actuación claros. Los profesionales informáticos cada vez han de tomar más decisiones para las que no existen documentos legales ni reglamentos que les puedan servir de ayuda. Por eso, según Moor, la tarea de la EI es aportar guías de actuación cuando no hay reglamentación o cuando la existente es obsoleta.

Por su parte, Terrel Bynum [14], tomando como base la definición de Moor, nos ofrece una visión aún más amplia de la EI en la que ésta se define como la disciplina que identifica y analiza los impactos de las tecnologías de la información en los valores humanos y sociales. Estos valores son: la salud, la riqueza, el trabajo, la libertad, la democracia, el conocimiento, la privacidad, la seguridad y la autorrealización personal.

Conger [17] opina que la intención de los que escriben (o escribimos) sobre esta materia no es la de adoctrinar o inculcar un conjunto de principios éticos concretos, sino la de crear una conciencia social en torno a los distintos usos y aplicaciones de las tecnologías de la información, y ayudar a los informáticos a utilizar dichas tecnologías de una forma éticamente

correcta. Para Pecorino [56], el objetivo principal de la EI consiste en conseguir que la toma de decisiones sobre temas tecnológicos sea consistente con la afirmación de los propios valores que uno profesa o con los derechos humanos en general.

Para conseguir esto, la ética informática se plantea varios objetivos intermedios. Por un lado, según Parker [55], es preciso encontrar y articular dilemas éticos clave en informática, y determinar en qué medida son agravados, transformados o creados por la tecnología informática. Por otro lado, Moor [51] expresa la necesidad de analizar y proponer un marco conceptual adecuado y formular principios de actuación para determinar qué hacer en las nuevas actividades ocasionadas por la informática en las que no se perciben con claridad líneas de actuación.

Para realizar lo anterior, la EI pretende tener en cuenta dos aspectos. Por un lado, utilizar la teoría ética para clarificar los dilemas éticos y detectar errores en el razonamiento ético [55], y, por otro, colaborar con otras disciplinas en ese debate. Sin embargo, la EI puede ir más allá. Además de proponer normas de conducta y estudiar a qué valores afectan éstas, también puede reconsiderar determinados valores que son implícitamente aceptados. Por ejemplo, es posible replantearse el concepto de propiedad con respecto al *software*, un tipo de propiedad que no encaja perfectamente en el concepto de propiedad tradicional. La EI puede analizar qué tipo de propiedad es el *software*, pero puede plantearse un debate más profundo preguntándose por qué ha de existir propiedad intelectual. Esto supone plantearse de manera nueva valores antiguos y reconsiderar su vigencia [51].

Vamos a terminar haciendo una observación sobre la expresión «Ética de la Informática». Como comenta Guibert [35] en su artículo, si Moor hablaba de la EI como el estudio del impacto de los ordenadores, años más tarde Bynum lo hacía como el impacto de la Tecnología de la Información (TI), aunque en la actualidad sería más apropiado hablar del impacto de las Tecnologías de la Información y las Comunicaciones (TIC). Por esto, para Guibert, es necesario replantearse el uso de la expresión «Ética de la Informática» y buscar en su lugar otra expresión que incluya no sólo a la informática sino a todo el conjunto de las Tecnologías de la Información y las Comunicaciones, aunque hemos de añadir que quizás bastaría con incluir dentro de la ciencia informática las modernas tecnologías de las comunicaciones.

1.1.3. Problemas tratados por la ética informática

Hasta el día de hoy no existe unanimidad con respecto a los contenidos que conforman la EI. Aun así, vamos a presentar la recopilación que Guibert incluye en su artículo [35] y que contiene los temas y problemas más importantes que han sido tratados por los distintos autores:

- *Ética profesional general*: Abarca aquellos problemas que son comunes a otras actividades ocupacionales. Estos problemas pueden dividirse en tres grupos:

    - Problemas personales de los profesionales: criterios, obligaciones y responsabilidades personales de los profesionales.

  - Problemas interiores a la empresa: relaciones entre los empleados, lealtad organizacional, interés público, comercialización de productos similares a los de la empresa, etc. Además, existe otro grupo de problemas que han sido creados o agravados por las nuevas tecnologías: control empresarial sobre el uso del ordenador (correo, chat, mensajería instantánea, navegación web...) con finalidades extraproductivas, investigación de los registros personales de los empleados para detectar uso de drogas, etc.
  - Problemas relativos a las prácticas comerciales (incluyendo contratos, acuerdos y conflictos de interés): proponer programas informáticos inferiores, comercializar *software* sabiendo que tiene fallos (*bugs*), etc.

- *La utilización de la información*: Recoge aquellos problemas relativos al uso no autorizado de los servicios informáticos o de la información contenida en ellos. Se plantean problemas como los siguientes: invasión de la intimidad, falta de confidencialidad en la información, uso de datos personales sin autorización, inspeccionar registros personales ajenos, desarrollar tarjetas de crédito inteligentes que almacenen información personal sin que los sepan los titulares de la tarjeta, delimitación o censura de determinados contenidos (pornografía infantil, racismo...), acceso igualitario a las redes de la información, etc.

- *Lo informático como nueva forma de bien o propiedad*: Este grupo de problemas hace referencia a que el *software* es un tipo de propiedad especial que no se ajusta fácilmente al concepto tradicional de propiedad. A primera vista, la solución pasaría por proteger el *software* mediante las leyes actuales de propiedad intelectual. Sin embargo, como dice Guibert, para proteger adecuadamente el *software* antes debemos plantearnos que es en sí un programa: «¿Es un algoritmo o una idea que no puede ser poseída por nadie porque pertenece al patrimonio cultural de la humanidad? ¿es propiedad intelectual que puede ser poseída y protegida?».

  Este planteamiento origina la aparición de nuevos problemas de propiedad, pirateo, plagio, derechos de autor, secretos industriales, etc., como los siguientes: cesión de *software* comercial, desarrollo de un programa a partir de otro comercial, reclamación de los derechos de propiedad intelectual relativos a un programa realizado en la universidad o en la empresa, etc.

- *Lo informático como instrumento de actos potencialmente dañinos*: Este tema, a veces considerado como parte específica de la EI y a veces no, comprende los hechos en los cuales se cometen acciones que provocan daño a terceras personas mediante el uso de cualquier tipo de medio informático. Aquí se incluyen las consecuencias de los errores en datos y algoritmos, los problemas que se pueden causar por la falta de protección en la seguridad de sistemas con datos sensibles o que implican riesgos en la salud de

clientes, los actos de terrorismo lógico, las acciones de fanáticos, el espionaje de datos, las introducciones de virus y gusanos [73].

Aparte de perseguir estas acciones, también es importante concienciar a las personas de las consecuencias que se pueden derivar del uso de los equipos y aplicaciones informáticas, instaurando en la mismas un sentimiento de responsabilidad sobre sus actos.

- *Miedos y amenazas de la informática*: Esta parte de la EI se encarga de estudiar las implicaciones éticas de la inteligencia artificial (sistemas expertos, redes neuronales...). Un ejemplo lo constituye el caso de los sistemas de soporte de decisión (SSD). Estos sistemas permiten tratar y gestionar la complejidad y la incertidumbre de manera racional, son eficientes y actúan según criterios consistentes. No obstante, también plantean problemas éticos [44]. Por un lado, los relativos a los criterios que han de seguir los propios sistemas (por ejemplo, cómo gestionar los riesgos para la salud humana o cómo hacer equivalencias, si es que es posible, entre la vida humana y ciertas cantidades de dinero). Por otro lado, posibles desviaciones ocultas en el proceso de toma de decisiones. Por último, hay que determinar hasta qué punto los diseñadores de estos sistemas son responsables de los resultados de los mismos.

- *Dimensiones sociales de la informática*: La implantación de la informática también ha traído consigo consecuencias negativas. Entre otras cosas, ha contribuido a aumentar el abismo existente entre los países desarrollados y en desarrollo, y ha jugado un papel importante en la globalización de la economía. También ha influido en el sector empresarial desplazando masivamente empleo (sobre todo de tipo administrativo), deshumanizando las condiciones laborales, aumentando la competitividad, etc. [48].

  Otros problemas a tener en cuenta son la desigual distribución de la información (la cual crea ricos y pobres en información: la llamada brecha digital), el desigual acceso a los medios técnicos (incluyendo a las redes de información), el modo en el que la tecnología de la información refuerza la actual distribución de poder... [48].

  Por último, cabe mencionar que los informáticos han sido una pieza fundamental en la investigación, desarrollo y producción de la tecnología militar. Por ello, desde la EI se podría concienciar a los informáticos sobre si es ético desarrollar modos «superinteligentes» para idear sufrimiento y destrucción humanos [9].

1.1.4. Códigos éticos

Un código ético no es más que un conjunto de reglas, directrices, normas, definidas para un dominio o grupo. Un ejemplo de código ético breve y conciso es el de los diez mandamientos de la ética informática, creado por el *Computer Ethics Institute* (CEI)[1]:

[1] Disponible en http://www.brook.edu/its/cei/overview/Ten_Commanments_of_Computer_Ethics.htm. Podemos consultar una traducción al español en http://www.geocities.com/virtualserinformatica/mandamientos.html

1. No usarás una computadora para dañar a otros.
2. No interferirás en el trabajo ajeno.
3. No indagarás en los archivos ajenos.
4. No utilizarás una computadora para robar.
5. No utilizarás la informática para realizar fraudes.
6. No copiarás o utilizarás *software* que no hayas comprado.
7. No utilizarás los recursos informáticos ajenos sin la debida autorización.
8. No te apropiarás de los derechos intelectuales de otros.
9. Deberás evaluar las consecuencias sociales de cualquier código que desarrolles.
10. Siempre utilizarás las computadoras de manera que se consideren y respeten los derechos de los demás.

El Apéndice B contiene el código ético de la asociación ACM (*Association for Computing Machinery*). Este es un código bastante más completo que puede servir para que el lector se haga una mejor idea de qué es un código ético y de qué trata.

Aunque las diferentes asociaciones de profesionales y empresas relacionadas con la informática han desarrollado distintos códigos deontológicos, todos ellos intentan cumplir unas determinadas funciones. En su artículo [35], Guibert nos presenta una lista con estas funciones, basada en los trabajos de Berleur [7] y Holvast [38]:

- El que existan normas éticas para una profesión quiere decir que un profesional, en este caso un técnico, no es sólo responsable de los aspectos técnicos del producto, sino también de las consecuencias económicas, sociológicas y culturales del mismo.
- Sirven también como un instrumento flexible como suplemento a las medidas legales y políticas, ya que éstas en general van muy lentas comparadas con la velocidad del desarrollo de las tecnologías de la información. Los códigos hacen de suplemento a la ley y sirven de ayuda a los cuerpos legislativos, administrativos y judiciales.
- Sirven como concienciación pública, ya que crear unas normas así hace al público consciente de los problemas y estimula un debate para designar responsabilidades.
- Estas normas tienen una función sociológica ya que dan una identidad a los informáticos como grupo que piensa de una determinada manera; es símbolo de su estatus profesional y parte de su definición como profesionales.

- Estas normas sirven también como fuente de evaluación pública de una profesión y son una llamada a la responsabilidad que permiten que la sociedad sepa qué pasa en esa profesión; aumenta la reputación del profesional y la confianza del público.

- En las organizaciones internacionales estas normas permiten armonizar legislaciones o criterios divergentes existentes (o ausentes, en su caso) en los países individuales.

Sin embargo, como señala Guibert en el mismo artículo, «la crítica que se hace a estas asociaciones es que han hecho poco por hacerlos cumplir, por imponer sanciones si no se cumplen o por comprobar si se aplican o si son relevantes o pertinentes. De hecho hay códigos que no son conocidos por los miembros de sus profesiones y menos por sus clientes».

No obstante, el hecho de que las asociaciones de profesionales de la informática se preocupen de crear códigos deontológicos ya es algo positivo [8]. Los códigos suponen un paso adelante en el proceso de concienciación de las sociedades y organizaciones que quieren mejorar situaciones en las que los impactos sociales del desarrollo tecnológico no se tienen en cuenta.

La aplicación web desarrollada en este proyecto tiene como objetivo facilitar la difusión de dichos códigos éticos. Al estar diseñada considerando la posibilidad de tener los códigos traducidos en varios idiomas, se hace todavía más fácil esta difusión, ya que cualquier persona interesada podrá traducir un determinado código a su lengua y enviárnosla para que la introduzcamos en la base de datos.

1.2. Traducción y análisis de códigos éticos informáticos

Una parte importante del proyecto ha sido la de buscar, traducir y clasificar códigos éticos. En total, hemos conseguido los códigos de veintiocho asociaciones informáticas de nivel internacional, continental, nacional y regional. La mayor parte de estos códigos han sido traducidos al español.

Una vez traducidos, hemos dividido a todos los códigos en partes más pequeñas, llamadas artículos o secciones, para así poder introducirlos en la base de datos. El proceso de clasificación ha consistido en asociar a cada uno de estos artículos uno o más temas relacionados con la ética informática (honradez e integridad, medio ambiente, etc.).

Finalmente, hemos calculado una pequeña estadística sobre los temas y los grupos.

1.2.1. Recopilación de códigos éticos informáticos

El primer paso que hemos tenido que dar ha sido recopilar el mayor número posible de códigos éticos de asociaciones relacionadas con las TIC. Esta búsqueda ha sido llevada a cabo a través de la Web, visitando los sitios oficiales de dichas asociaciones. No todas las asociaciones ofrecen acceso a sus códigos éticos a través de su sitio en la Web. Por ello, también se enviaron

Asociación	Irish Computer Software (ICS)
Descripción	El ICS fue fundado en 1967 como el cuerpo nacional de Profesionales de la Tecnología de la Información y la Comunicación (TIC) de Irlanda. Desde su fundación, el ICS ha promovido el desarrollo continuo del conocimiento y las habilidades profesionales en el campo de las TIC organizando seminarios, conferencias y actividades relacionadas.
Ámbito	Nacional
Localización	Irlanda
Fecha asociación	1967
Web asociación	http://www.ics.ie/
Fecha código	Junio 1994
Web código	http://www.ics.ie/code.shtml

Tabla 1.1: Información recopilada sobre la asociación Irish Computer Software (ICS).

mensajes de correo electrónico a distintas asociaciones, pidiéndoles sus respectivos códigos deontológicos.

Adicionalmente al propio contenido, hemos intentado obtener la siguiente información de cada código ético:

- Fecha de adopción del código.
- Dirección web del código.

Además de los códigos deontológicos, también se ha recogido información relativa a cada asociación:

- Nombre de la asociación.
- Descripción de la asociación.
- Ámbito de la asociación: internacional, continental, etc.
- Localización geográfica de la asociación. Ejemplos: Europa, España, etc.
- Año de creación de la asociación.
- Sitio web de la asociación.

Como ejemplo, en la Tabla 1.1 mostramos toda la información recopilada sobre la asociación Irish Computer Software (ICS).

Finalmente, en la Tabla 1.2 podemos ver un listado de las asociaciones relacionadas con las TIC cuyos códigos deontológicos hemos recopilado y que formarán parte de la base de datos a la que accederá nuestra aplicación web. Las asociaciones son mostradas ordenadas según su localización geográfica, junto con la dirección web de su sitio oficial.

Ámbito	Asociación	Web
África		
Sudáfrica	Computer Society of South Africa (CSSA)	www.cssa.org.za
América		
Argentina	Consejo Profesional en Ciencias Informáticas (CPCI)	www.cpci.org.ar
Canadá	Canadian Information Processing Society (CIPS)	www.cips.ca
Costa Rica	Colegio de Profesionales en Informática y Computación (CPIC)	www.cpic.or.cr
EEUU	American Society for Information Science and Technology (ASIS&T)*	www.asis.org
EEUU	Association of Information Technology Professionals (AITP)*	www.aitp.org
EEUU	SANS Institute*	www.sans.org
Asia		
Hong Kong	Hong Kong Computer Society (HKCS)*	www.hkcs.org.hk
Japón	Japan Information Technology Services Industry Association (JISA)*	www.jisa.or.jp/en
Singapur	Singapore Computer Society (SCS)*	www.scs.org.sg
Europa		
Continental	Council of European Professional Informatics Societies (CEPIS)*	www.cepis.org
España	Colegio Oficial de Ingenieros en Informática de Cataluña (COEIC)	www.coeic.org
España	Colegio Oficial de Ingenieros en Informática de la Comunidad Valenciana (COIICV)	www.coiicv.org
España	Colegio Oficial de Ingenieros en Informática del País Vasco (COIIE)	www.coiie.org
España	Colegio Oficial de Ingenieros en Informática del Principado de Asturias (COIIPA)	www.coiipa.org
España	Colegio Oficial de Ingenieros Tecnicos en Informática del Principado de Asturias (CITIPA)	www.citipa.org
España	Asociación de Técnicos de Informática (ATI)	www.ati.es
España	Instituto Español De Ética Informática (IEDEI)	www.iedei.org
Irlanda	Irish Computer Society (ICS)*	www.ics.ie
Malta	Computer Society of Malta (CSM)*	www.csm.org.mt
Reino Unido	British Computer Society (BCS)*	www.bcs.org.uk
Oceanía		
Australia	Australian Computer Society (ACS)*	www.acs.org.au
Australia	System Administrators Guild of Australia (SAGE-AU)*	www.sage-au.org.au
Nueva Zelanda	New Zealand Computer Society (NZCS)	www.nzcs.org.nz
Internacional		
	ACM/IEEE Computer Society*	www.computer.org
	Association of Computing Machinery (ACM)*	www.acm.org
	Association of Service and Computer Dealers International (ASCDI)*	www.ascdi.com
	Institute of Electrical and Electronics Engineers (IEEE)*	www.ieee.org

* Asociaciones cuyos códigos han sido traducidos al español para la base de datos de este proyecto.

Tabla 1.2: Asociaciones relacionadas con las TIC cuyo código ético está incluido en la base de datos.

1.2.2. Traducción de los códigos

Una vez hemos recopilado la información y los códigos de las asociaciones, el siguiente paso ha sido traducir dichos textos. Únicamente se han traducido las descripciones y los códigos.

Aunque la base de datos estará diseñada para almacenar traducciones en cualquier idioma, nos limitaremos a dos lenguas: el español y el inglés. Por restricciones de tiempo y conocimientos, en este proyecto sólamente procederemos a traducir al español aquellos códigos escritos en inglés. En la Tabla 1.2 aparecen con un asterisco los códigos que hemos traducido al español. Por otro lado, todas las descripciones estarán disponibles tanto en inglés como en español. En total, hemos traducido diecinueve códigos, con lo que nuestra base de datos puede ofrecer en Internet la mayor recopilación de códigos éticos informáticos escritos en español.

Antes de ponernos directamente a traducir, hemos realizado una búsqueda, por la Web, de posibles traducciones, oficiales o no, de los códigos de las asociaciones. De esta manera, evitamos hacer un trabajo ya hecho. Pues bien, gracias a esto, hemos encontrado las traducciones de cuatro códigos, una de ellas de carácter oficial. Cómo es lógico, la base de datos está diseñada para almacenar toda la información relativa a la autoría de las traducciones: traductor y dirección web de la traducción.

Puesto que las traducciones realizadas estarán accesibles a través de la aplicación web desarrollada, cualquier persona podrá ponerse en contacto con nosotros, transmitiéndonos todas aquellas sugerencias que consideren útiles. De esta forma, las traducciones realizadas en este proyecto podrán ser revisadas y mejoradas. Además, dado el carácter plurilingüe de la base de datos, siempre se podrán incorporar traducciones de los códigos a otros idiomas distinto del español y el inglés, así como de las reseñas de las asociaciones.

1.2.2.1. Cómo resolver las dudas terminológicas

Existen varias alternativas para resolver las dudas terminológicas (términos que nos resultan difíciles de traducir) [1]:

- Consultar a los especialistas, tanto expertos sobre el tema del texto, como profesionales de la traducción y de la lengua.
- Listas y foros, como las siguientes:
    - Lista Spanglish, especializada en informática y tutelada por el Grupo De Lengua e Informática de la ATI. http://delfos.sci.uma.es/mailman/listinfo/spanglish.
    - Lista del Español Urgente, dedicada a resolver cuestiones generales de lengua y tutelada por el equipo de filólogos de la Agencia EFE. http://lists.albura.net/efe.es/apuntes/.
    - Foros del Centro Virtual de Cervantes. http://cvc.cervantes.es/foros/.

- Google. Ayuda al traductor en dos aspectos: a buscar traducciones para los distintos términos y a comprobar que las traducciones propuestas se están utilizando. Google proporciona además información sobre dónde (España, Hispanoamérica, etc.) y quienes (universidades, fabricantes, periodistas, etc.) usan los términos. http://www.google.es.

- Diccionarios, bases de datos terminológicas y glosarios. Debido al «medio» en el que trabajamos, el medio informático, se tiende a utilizar los recursos disponibles en la Red:

  - Eurodicautom, el banco de datos terminológicos de la Comisión Europea. http://europa.eu.int/eurodicautom/Controller.
  - Termite, la base de datos terminológica de la Unión Internacional de Comunicaciones (UIT). http://www.itu.int/terminology/index.html.
  - Glosario básico inglés-español para usuarios de Internet, de Rafael Fernández Calvo. http://www.ati.es/novatica/glointv2.html.
  - ORCA: Glosario de informática inglés-español. http://www.linux.org.ni/LuCAS/LuCAS/ORCA/glosario.html.
  - Errores habituales de Spanglish de los informáticos... y también de los no informáticos, de Ángel Álvarez. http://maja.dit.upm.es/ aalvarez/pitfalls/.
  - Diccionario informático inglés-castellano GIAIT, http://es.tldp.org/Otros/diccionario-us-es/.
  - Términos debatidos en la lista Spanglish. http://www.gsi.dit.upm.es/ gfer/spanglish/.
  - Diccionario electrónico Lexibase-Collins inglés-español.
  - Diccionario de la Real Academia Española. http://buscon.rae.es/diccionario/drae.htm.
  - Dudas resueltas por el Departamento de Español Urgente de la Agencia EFE. http://www.efe.es/esurgente/lenguaes/.
  - Tesauro de SIGNUM, diccionario *online* de sinónimos y antónimos del español. http://www.lenguaje.com/herramientas/tesauro/.
  - Diccionario y tesauro de inglés Merriam-Webster. http://www.m-w.com/.

- *Software* de traducción:

  - Free Translation, traductor *online* de textos y de páginas webs. http://www.freetranslation.com/.
  - L&H Power Translator Pro 7.0.

1.2.3. Clasificación temática de los códigos

Puesto que la clasificación de un código consiste en asociar uno o más temas a las distintas partes o artículos del código, lo primero que necesitamos es una lista de temas. Como punto de partida hemos utilizado la lista de temas que aparece en el artículo «Un sistema de información para el estudio de códigos de ética informática», de Eugeni Zamora y Miquel Barceló [26]. Empleando este listado, realizamos una primera clasificación de todos los códigos. Tras esta primera clasificación, nuestro listado sufrió modificaciones: unimos algunos temas, eliminamos otros..., quedando un número final de temas muy elevado, treinta y siete para ser exactos. Por esto, vimos necesario clasificar, a su vez, dichos temas en grupos más grandes. La lista definitiva de temas, formada por treinta y siete temas divididos en ocho grupos, que hemos obtenido es la siguiente:

- Apoyos:
  - Apoyo y defensa a la asociación.
  - Apoyo y defensa a la empresa.
  - Apoyo y defensa a la nación.
  - Promoción y defensa de la Informática.
- Desarrollo Profesional:
  - Capacitación y formación profesional.
  - Desarrollo y carrera profesional.
  - Docencia.
- Desempeño de la Profesión:
  - Buen uso de recursos.
  - Dirección y liderazgo.
  - Lenguaje comprensible.
  - Uso del correo electrónico.
- Ética y Legalidad:
  - Comportamiento no ético.
  - Cumplimiento del código (alcance, sanciones, denuncias).
  - Dilema ético.
  - Justificación del código ético.

- Leyes, normas, estándares.
- Promoción del comportamiento ético.
- Propiedad intelectual.

- Relaciones Humanas:
  - Clientes de nuestra empresa.
  - Colegas de profesión (solidaridad, cooperación, respeto, competencia).
  - Proveedores.
  - Usuarios.

- Sociedad y Medio Ambiente:
  - Acceso a las Tecnologías de la Información.
  - Derechos humanos (discriminación, justicia, salud, seguridad).
  - Interés público (bien social).
  - Medio Ambiente.

- Valores Personales y Profesionales:
  - Confidencialidad y privacidad.
  - Honradez e integridad.
  - Imparcialidad (desinterés personal).
  - Libertad profesional.
  - Profesionalidad y diligencia (responsabilidad, buen servicio, calidad).
  - Transparencia de la información.

- Otros Temas:
  - Calidad del lugar del trabajo.
  - Condiciones económicas justas (sueldo, tarifas).
  - Conflicto de intereses.
  - Código fuente abierto (Open Source).
  - Opinión profesional.

Tema	Interés público (bien social)
Descripción	Los profesionales perseguirán el bien de la sociedad en general. Se preocuparán por el interés del público, así como se encargarán de aumentar la apreciación y el conocimiento públicos de la informática.
Palabras Clave	Interés público, bien social, sociedad, calidad de vida, derechos humanos, diversidad cultural, bienestar humano, responsabilidad social...

Tabla 1.3: Información sobre el tema «Interés público (bien social)».

Este es el listado de temas que hemos utilizado finalmente para realizar una segunda clasificación, la última y definitiva. Todos estos temas tienen una descripción y una lista de palabras clave que sirven para explicar cada tema. Esta información también está almacenada en la base de datos, tanto en inglés como en español. Por ejemplo, la información relativa al tema «Interés público (bien social)» la podemos ver en la Tabla 1.3. En el Apéndice A se pueden encontrar las descripciones y palabras clave de todos los temas.

Más que pensar en una clasificación teóricamente perfecta, hemos pensado en una clasificación temática útil. O sea, más que buscar temas «disjuntos» entre sí, hemos buscado temas que engloben un asunto o problemática suficientemente interesante o cuya búsqueda pueda tener interés y/o que existan suficientes artículos que puedan ser asociados a dichos temas.

Aunque en el Apéndice A se explica brevemente cada tema, ello no ha supuesto menospreciar el poner interés en que los títulos de cada tema sean suficientemente claros y explicativos.

En los casos en los que hemos dudado entre asociar un tema a un artículo o no hacerlo, en general, nos hemos decantado por «asociarlo», ya que creemos que es preferible que salga ese artículo en la búsqueda, aunque luego el usuario decida que ese no le sirve o que no era eso lo que buscaba. Hay que decir que puede haber artículos con varios temas a los que posiblemente se les pueda asociar otros o eliminar algunos, según distintas perspectivas. La explicación/descripción de un tema es una ayuda y no obliga a nada, por lo que se deja a la elección del clasificador el asociar o no un tema a una sección.

1.2.4. Resultados de la clasificación

Para terminar, nos parecía interesante obtener una estadística que nos sirviera para observar aproximadamente cuánta importancia se le dá a cada tema. Las variables estadísticas que hemos calculado para cada tema han sido la frecuencia absoluta (cuántos artículos tienen asociados dicho tema) y la frecuencia relativa (la proporción entre la frecuencia absoluta y el número total de asociaciones entre artículos y temas). La Tabla 1.4 recoge las estadísticas de todos los temas, que aparecen en orden decreciente por frecuencia relativa.

Observando la tabla, vemos que no existen saltos especialmente significativos entre las frecuencias de los distintos temas. El tema con mayor frecuencia es «Profesionalidad y diligencia

Tema	**Frec. Rel. %**	**Frec. Abs.**
Profesionalidad y diligencia (responsabilidad, buen servicio, calidad)	9.23	242
Honradez e integridad	7.82	205
Colegas de profesión (solidaridad, cooperación, respeto, competencia)	6.56	172
Clientes de nuestra empresa	5.72	150
Leyes, normas, estándares	5.38	141
Transparencia de la información	5.26	138
Capacitación y formación profesional	5.18	136
Apoyo y defensa a la empresa	4.77	125
Dirección y liderazgo	4.38	115
Interés público (bien social)	4.19	110
Derechos humanos (discriminación, justicia, salud, seguridad)	4.19	110
Cumplimiento del código (alcance, sanciones, denuncias)	3.77	99
Promoción y defensa de la Informática	3.58	94
Confidencialidad y privacidad	3.58	94
Opinión profesional	3.58	94
Imparcialidad (desinterés personal)	3.55	93
Propiedad intelectual	1.83	48
Apoyo y defensa a la asociación	1.79	47
Desarrollo y carrera profesional	1.68	44
Usuarios	1.56	41
Promoción del comportamiento ético	1.49	39
Docencia	1.30	34
Proveedores	1.30	34
Comportamiento no ético	1.26	33
Conflicto de intereses	1.11	29
Medio ambiente	0.99	26
Buen uso de recursos	0.95	25
Justificación del código ético	0.91	24
Condiciones económicas justas (sueldo, tarifas)	0.91	24
Dilema ético	0.42	11
Acceso a las Tecnologías de la Información	0.42	11
Lenguaje comprensible	0.34	9
Calidad del lugar del trabajo	0.30	8
Libertad profesional	0.30	8
Uso del correo electrónico	0.27	7
Código fuente abierto (Open Source)	0.08	2
Apoyo y defensa a la nación	0.04	1
Total	100	2623

Tabla 1.4: Frecuencias absoluta y relativa de los temas.

Grupo de Temas	**Frec. Rel. %**	**Frec. Abs.**
Valores Personales y Profesionales	29.74	780
Relaciones Humanas	15.14	397
Etica y Legalidad	15.06	395
Apoyos	10.18	267
Sociedad y Medio Ambiente	9.80	257
Desarrollo Profesional	8.16	214
Otros Temas	5.99	157
Desempeño de la Profesión	5.95	156
Total	100	2623

Tabla 1.5: Frecuencias absoluta y relativa de los grupos de temas.

(responsabilidad, buen servicio, calidad)», que trata sobre la profesionalidad y diligencia de los profesionales informáticos, que deben ofrecer productos y servicios de calidad así como cumplir las obligaciones de su trabajo profesional de acuerdo con los objetivos de plazo y presupuesto acordados.

Existen doce temas con menos de un uno por ciento de frecuencia relativa. Aunque podríamos pensar en eliminarlos, creemos que merece la pena tenerlos debido a que nos parecen temas interesantes y curiosos.

Puesto que tenemos grupos que engloban a los temas, también merece la pena tener una estadística de dichos grupos. Los datos estadísticos a calcular son los mismos que para los temas: frecuencia absoluta y frecuencia relativa. En la Tabla 1.5 se recogen las estadísticas de todos los grupos de temas, que aparecen ordenados en orden decreciente por frecuencia relativa.

En el caso de los grupos, existe uno que sobresale del resto, «Valores Personales y Profesionales», con un porcentaje de casi un treinta por ciento. Este tema engloba a los dos temas con mayor frecuencia: «Profesionalidad y diligencia (responsabilidad, buen servicio, calidad)» y «Honradez e integridad».

Al contrario que para los temas, no existen grupos con un porcentaje demasiado bajo (menor al uno por ciento). Todos los grupos tienen una frecuencia relativa superior al cinco por ciento.

Es importante aclarar que el listado de temas y grupos que hemos presentado en este proyecto es relativo y discutible. Aunque hemos intentado abarcar todos aquellos asuntos que nos han parecido interesantes, desde otras perspectivas se podrían plantear la supresión, unión o separación de temas o grupos ya existentes, así como la creación de nuevos temas o grupos.

Capítulo 2

Tecnologías web

Este capítulo es una introducción a los conceptos y tecnologías relacionados con la web. Está compuesto por los siguientes apartados:

- Introducción a la web. Se explica cómo funciona la web y se presentan los tres estándares básicos en los que se apoya ésta: HTTP, URL y HMTL [6, 16].
- Estándares web. Presentamos los estándares web establecidos por la W3C y otras organizaciones, usados para crear e interpretar contenido basado en la web [6, 16, 19, 20, 24, 42].
- Sistemas para la web. Este apartado muestra las diferencias entre los dos tipos de sistemas web existentes: sitio web y aplicación web [16].
- Páginas web. Por último, hablaremos sobre los documentos básicos de la web, es decir, las páginas web. Las clasificaremos en dos tipos: estáticas y dinámicas [16, 18].

2.1. Introducción a la web

Puede haber cierta confusión en cuanto a qué es exactamente Internet y en qué se diferencia de la web (WWW, *World Wide Web*). Internet es una red física de redes a escala mundial de millones de computadoras interconectadas con el conjunto de protocolos TCP/IP. La web, por otro lado, es uno de los muchos servicios ofertados en la red Internet (ver la Figura 2.1). La web es un sistema de información que emplea la red Internet como medio de transmisión.

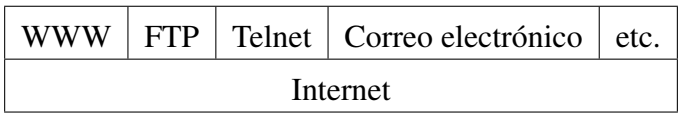

Figura 2.1: Servicios disponibles en Internet

Algunos de los servicios disponibles en Internet, aparte de la web, son el acceso remoto a otras máquinas (telnet y ssh), transferencia de archivos (FTP), correo electrónico (e-mail), boletines electrónicos (news o grupos de noticias), conversaciones en línea (chat), mensajería instantánea (ICQ, YIM, Jabber), etcétera.

Los primeros sistemas web, creados por Tim Berners-Lee para el CERN (*Centre Européen pour la Recherche Nucléaire*, Centro Europeo para la Investigación Nuclear), formaban un sistema hipermedia distribuido que permitía a los investigadores tener acceso a la información y documentos de las computadoras de sus compañeros.

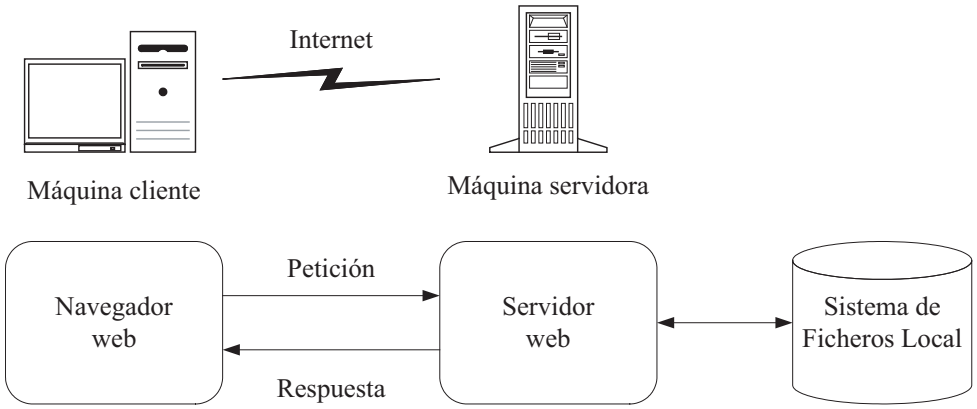

Figura 2.2: Sistema web básico

Como servicio, la web define dos partes, un cliente web (o navegador web) y un servidor web, que utilizan Internet para comunicarse; ver la Figura 2.2. El navegador web es una aplicación software ejecutada en la máquina cliente (o cliente) que permite al usuario recuperar y visualizar documentos alojados en máquinas servidoras (o servidores). Para visualizar un documento, el usuario debe introducir en el navegador web el nombre del documento y el nombre de la máquina servidora donde se encuentra dicho documento. Entonces el navegador web envía una petición del documento a la máquina servidora. Esta petición es manejada por el servidor web, una aplicación software ejecutada (normalmente como servicio o demonio[1]) en la máquina servidora, que monitoriza un puerto[2] especial, normalmente el puerto 80, en espera de peticiones llevadas a cabo por un navegador web. Cuando un servidor web recibe la petición de un documento, localiza este documento en el sistema de ficheros de la máquina servidora y le envía una respuesta al navegador web que envió la solicitud.

La web es un sistema hipermedia distribuido. Es un sistema hipermedia ya que los documentos del sistema están interconectados entre sí mediante hipervínculos (o enlaces). Estos hipervínculos posibilitan la navegación por los recursos del sistema. Los recursos no sólo pue-

[1]En Unix y otros sistemas operativos, un demonio (*daemon*) es una clase especial de programa que se ejecuta en segundo plano (*background*), es decir, sin interactuar con un humano. Los programas demonio reciben distintas denominaciones según el sistema operativo del que se trate. En Windows los demonios son llamados servicios.

[2]Un puerto es una forma genérica de denominar a una interfaz por la cual diferentes tipos de datos pueden ser enviados y recibidos. Dicha interfaz puede ser física o a nivel de software.

den ser documentos de texto (hipertexto), sino que también incluyen a otro medios como audio, vídeo, etc.; de ahí el término hipermedia. Es un sistema distribuido, puesto que los recursos están distribuidos en una red de computadoras, donde la existencia de múltiples computadoras es transparente al usuario.

2.1.1. Los tres estándares básicos: HTTP, HTML y URL

La funcionalidad elemental de la web se basa en tres estándares: el protocolo HTTP, que especifica cómo el navegador y el servidor intercambian información en forma de peticiones y respuestas, el lenguaje HTML, un método para codificar la información de los documentos y sus enlaces, y la codificación URL, que especifica cómo a cada documento se le asocia una «dirección» única en la que encontrarlo. Veamos cada uno de ellos.

2.1.1.1. HTTP

Hemos visto que el modelo de comunicación usado en la web es el modelo de petición/respuesta. Pues bien, los navegadores y los servidores web utilizan el protocolo HTTP (*Hypertext Transfer Protocol*, Protocolo de Transferencia de Hipertexto)[3] para comunicarse. Este protocolo especifica el formato que deben utilizar los navegadores y los servidores web para los mensajes de petición y respuesta. Estos mensajes son simplemente un conjunto de líneas de caracteres.

Un mensaje de petición HTTP consiste en tres cosas: una línea de petición, cabeceras de petición y opcionalmente un cuerpo de petición. Un posible mensaje de petición sería el siguiente:

```
GET /index.html HTTP/1.1
Host: www.maquina.com
Accept-Language: es
```

La línea de petición comienza con el nombre del método de petición, en este caso es `GET`, seguido de un identificador de recurso y de la versión del protocolo usado por el navegador, en nuestro ejemplo `/index.html` y HTTP/1.1, respectivamente. El método `GET` es el más usado. Como el propio nombre indica, una petición `GET` es usada para obtener un recurso del servidor. Es el método de petición por defecto, de forma que cuando se escribe una URL (más adelante veremos lo que es) en la barra de direcciones de un navegador y pulsamos la tecla *intro*, o cuando hacemos clic en un enlace, la petición es enviada al servidor como una petición `GET`.

Las cabeceras de petición (`Host` y `Accept-Language` en este caso) proporcionan información adicional que el servidor puede usar a la hora de procesar la petición. El cuerpo del mensaje es incluido sólo en algunos tipos de peticiones, como la petición `POST`.

Además de los métodos `GET` y `POST`, HTTP especifica otros métodos. La Tabla 2.1 muestra todos los métodos de petición de HTTP 1.1. La especificación HTTP 1.0 sólo soportaba los métodos `GET`, `HEAD` y `POST`.

[3]El protocolo HTTP está definido en la especificación RFC-2616, disponible en http://www.ietf.org/rfc/rfc2616.txt

Método	Descripción
`GET`	Obtiene el recurso identificado por la URL de la petición
`HEAD`	Obtiene las cabeceras del recurso especificado por la URL de la petición
`POST`	Envía datos al servidor
`PUT`	Guarda el cuerpo de la petición en el servidor bajo la URL de la petición
`DELETE`	Borra el recurso del servidor asociado a la URL de la petición
`OPTIONS`	Obtiene los métodos HTTP soportados por el servidor
`TRACE`	Obtiene las cabeceras enviadas junto a la petición TRACE

Tabla 2.1: Métodos de petición HTTP 1.1

Los servidores web suelen configurarse para que usen un directorio particular de la máquina servidora como directorio raíz del sistema web. Cuando un servidor web recibe una petición, éste busca el recurso pedido como relativo a dicho directorio y genera un mensaje de respuesta que es enviado al navegador.

La estructura de un mensaje de respuesta es similar a la de un mensaje de petición. También se divide en tres partes: una línea de estado (o línea de respuesta), cabeceras de respuesta y opcionalmente un cuerpo de mensaje. Veamos un ejemplo:

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  <body>
    <h1>¡Hola Mundo!</h1>
  </body>
</html>
```

La línea de estado comienza con el nombre del protocolo, seguida de un código de estado y de una breve descripción del código de estado. Aquí estos valores son HTTP/1.1, 200 y `OK`. El código de estado 200 significa que la petición se ha ejecutado satisfactoriamente. En el ejemplo, significa que se ha encontrado la página web solicitada. Las cabeceras de petición sirven para dar información adicional al navegador web. Una línea en blanco separa las cabeceras del cuerpo del mensaje, que en este caso es una página HTML simple (veremos HTML más adelante).

Por supuesto, el cuerpo de la respuesta puede contener una página HTML más compleja o cualquier otro tipo de contenido. Por ejemplo, la petición podría devolver una página HTML con elementos `<img>` (una etiqueta HTML utilizada para insertar imágenes en las páginas). Cuando un navegador lea la primera respuesta y encuentre los elementos `<img>`, éste enviará una nueva petición para el recurso identificado por cada uno de estos elementos, frecuentemente

en paralelo. El servidor devuelve una respuesta por cada imagen pedida, donde el cuerpo de la respuesta contiene los *bytes* que componen la imagen, indicando el tipo de imagen mediante la cabecera de respuesta `Content-Type`. Entonces, el navegador combina todas las respuestas recibidas para visualizar la página completa. Esta interacción es ilustrada en la Figura 2.3. Una ventaja de esto es que permite que un navegador puede desactivar la visualización de las imágenes para acelerar la carga de páginas.

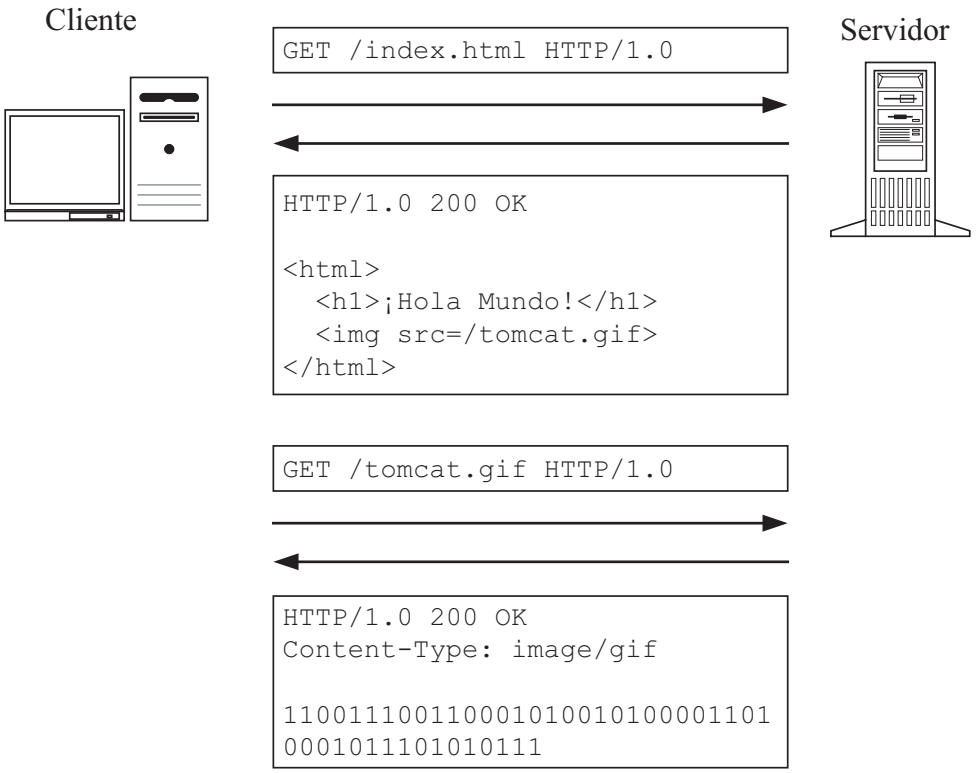

Figura 2.3: Interacción entre un cliente web y un servidor

2.1.1.2. HTML

HTML (*HyperText Markup Language*, Lenguaje de Marcas de Hipertexto) es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. HTML es un lenguaje de formato de documentos definido de acuerdo con SGML (*Standard Generalized Markup Language*, Lenguaje de Marcas Estándar Generalizado), un lenguaje mucho más extenso usado para definir lenguajes de marcas para propósitos específicos. Al igual que HTTP, HTML es un estándar mantenido por el W3C (*Worl Wide Web Consortium*, Consorcio de la WWW). La recomendación actual del W3C es HTML 4.01[4]. Para una descripción más detallada de HTML, ver el Apartado 2.2.1.

[4]Su especificación está disponible en http://www.w3c.org/TR/html4

2.1.1.3. URL

Un URL (*Uniform Resource Locator*, Localizador Uniforme de Recurso) es el identificador completo utilizado para referenciar y obtener un recurso de la web. Es simplemente una cadena de caracteres sin espacios en blanco. El URL de un recurso identifica el protocolo usado, el nombre de la máquina servidora, un número de puerto opcional y la ruta del recurso. Veamos un ejemplo:

```
http://www.maquina.com/busqueda/busqueda_simple.html
```

Este URL identifica a una página HTML llamada `busqueda_simple.html` que se encuentra disponible mediante protocolo HTTP en el directorio `busqueda`, relativo al directorio raíz del sistema web, de la máquina servidora llamada `www.maquina.com`. Por defecto, para peticiones HTTP se asume que el puerto es el 80.

Un URL es una especialización de un URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso). Un URL es un tipo de URI que identifica el recurso por su localización, por ejemplo el nombre del servidor que contiene el recurso. Existe otro tipo de URI, llamado URN (*Uniform Resource Name*, Nombre Uniforme de Recurso), que identifica el recurso por su nombre.

2.2. Estándares web

Los estándares web son tecnologías, establecidas por la W3C y otras organizaciones, usadas para crear e interpretar contenido basado en la web. Estas tecnologías están diseñadas para ser usadas por los documentos publicados en la web y hacer esos documentos todo lo accesibles que sea posible.

Los estándares web pueden clasificarse en:

- Lenguajes estructurales: sirven para describir la estructura lógica de un documento.
  - HTML (*Hypertext Markup Language*) 4.01
  - XHTML (*Extensible Hypertext Markup Language*) 1.0, 1.1
  - XML (*Extensible Markup Language*) 1.0
- Lenguajes de presentación: sirven para definir la forma en la que un documento debe ser visualizado.
  - CSS (*Cascading Style Sheets*) Nivel 1, Nivel 2 Revisión 1, Nivel 3 (en desarrollo)
  - MathML (*Mathematical Markup Language*)
  - SVG (*Scalable Vector Graphics*)

- Modelos de objetos: permiten acceder y manipular, mediante la programación, documentos estructurados.
    - DOM (*Document Object Model*) Nivel 1, Nivel 2, Nivel 3
- Lenguajes de *script*: permiten ejecutar código en la máquina del cliente.
    - ECMAScript 262 (la versión estandarizada de JavaScript)

Algunas razones para usar los estándares web son:

- Desarrollo y mantenimiento más sencillos: Usar un HTML más estructurado y semántico hace más fácil y rápido comprender el código creado por otra persona.
- Compatibilidad con navegadores web futuros: Cuando se usan estándares y código válido, aseguramos que los documentos puedan ser interpretados correctamente por navegadores web futuros.
- Descarga y visualización más rápida de las páginas web: Usar CSS significa usar menos código HTML para cada página, lo que conlleva tamaños de archivos más pequeños y descargas más rápidas.
- Mejor accesibilidad: Un HTML semántico, donde la estructura y la presentación están separados, facilita que cualquier dispositivo utilizado para navegar por la web sea capaz de interpretar correctamente el contenido de las páginas web.
- Adaptación más sencilla: Un documento marcado semánticamente puede ser adaptado fácilmente para ser impreso o para ser visualizado en dispositivos de navegación alternativos, como PDAs o teléfonos celulares, simplemente enlazando el documento a diferentes archivos CSS. También permite hacer cambios a la presentación de todo un sitio web, editando un único archivo.

2.2.1. Creación de páginas web básicas: HTML

El lenguaje HTML define un conjunto de etiquetas (*tags*, elementos) que son usadas para decirle al navegador como visualizar algo (texto, imágenes, etc.) o para definir un enlace a otra página web. Todas las etiquetas se encierran entre los símbolos < y >. Las etiquetas normalmente se usan por parejas envolviendo el texto que quieren formatear, con una etiqueta de apertura y otra de cierre. La etiqueta de cierre es marcada con el símbolo `/`. Por ejemplo, para presentar una palabra en cursiva se usa la etiqueta `<i>`. Veamos una sentencia de ejemplo y el código HTML asociado:

```
Palabra en <i>cursiva</i>.
```

```
Palabra en <i>cursiva</i>.
```

Algunos elementos no tienen contenido. Se les llama elementos vacíos y no deben llevar etiqueta de cierre. El elemento `<br>` es un ejemplo de elemento vacío, ya que su función es provocar un salto de línea dentro de un mismo párrafo.

Algunas etiquetas aceptan atributos, los cuales se colocan dentro de los caracteres < y > y normalmente constan de un nombre de atributo seguido de un signo igual y el valor del atributo encerrado entre comillas dobles. El siguiente fragmento de HTML crea un hipervínculo a otro documento web usando la etiqueta `<a>`:

```
La especificación de HTML se pueden encontrar en
<a href="http://www.w3c.org">el sitio web de la W3C</a>.
```

La mayoría de los navegadores, por defecto, subrayan los hipervínculos:

```
La especificación de HTML se puede encontrar en
el sitio web de la W3C.
```

El elemento `<a>` usa el atributo `href` para definir la localización y el tipo del enlace. La Tabla 2.2 muestra una breve referencia de las principales etiquetas HTML.

También es posible incluir comentarios dentro del código HTML. Estos comentarios no serán visualizados por el navegador y van escritos entre las etiquetas <!-- y -->.

El lenguaje HTML ha sido diseñado teniendo presentes dos normas básicas:

1. Define la estructura y componentes de un documento, no la forma concreta en que estos componentes se presentan cuando un cliente los visualiza.

2. No está atado a ningún entorno particular. El contenido de un documento HTML puede ser interpretado y visualizado en ordenadores con características muy diferentes.

No nos interesa en este momento profundizar demasiado en la sintaxis del lenguaje HTML, ya que existe abundante bibliografía al respecto y no es el objetivo del presente proyecto. Lo que sí vamos a ver son los elementos arquitecturalmente importantes del lenguaje, aquellos que serán modelados. Por ejemplo, el modelo de diseño de un sistema web no se interesa en el tamaño o el color de la fuente; en cambio, si está interesado en el conjunto de páginas web por las que se pueden navegar.

2.2.1.1. Vínculos

Un vínculo (hipervínculo, enlace) a un documento web es creado con la etiqueta `<a>`. Esta etiqueta puede tener varios atributos, siendo el más importante `href`. También puede ser usada

Etiqueta	Descripción	Ejemplo
	Texto en negrita	Texto en negrita
	Texto en cursiva	Texto en cursiva
<u>	Texto subrayado	<u>Texto subrayado</u>
 	Salto de línea	
<hr>	Línea horizontal	<hr>
	Tipo de letra	Texto grande
<img>	Imagen o gráfico	<img src="grafico.gif">
<a>	Enlace a otro documento	Ir a página
	Párrafo	Párrafo
<h1>	Encabezado de primer nivel	Título
<h2>	Encabezado de segundo nivel	Título
<h3>	Encabezado de tercer nivel	Título
<h4>	Encabezado de cuarto nivel	Título
<h5>	Encabezado de quinto nivel	Título
<h6>	Encabezado de sexto nivel	Título
	Lista ordenada	 Item 1 Item 2</l2>
	Lista no ordenada	 Item 1 Item 2</l2>
	Elemento de una lista	Item
<table>	Tabla	<table> <tr> <td>Celda 1 de la fila 1</td> <td>Celda 2 de la fila 1</td> </tr> <tr> <td>Celda 1 de la fila 2</td> <td>Celda 2 de la fila 2</td> </tr> </table>
<tr>	Fila de una tabla	<tr><td>Celda</td></tr>
<td>	Celda de una tabla	<td>Celda</td>

Tabla 2.2: Resumen de etiquetas HTML

sin especificar el valor `href`, pero en este caso la etiqueta es usada como un marcador interno de la página. El atributo `href`, el cual especifica la URL del documento enlazado, puede contener una URL relativa. Esta URL relativa no especifica la URL completa, sino que espera que el navegador utilice como máquina servidora la máquina suministrada por la página que contiene el vínculo. Por ejemplo, el siguiente vínculo es correcto:

```
Nosotros tenemos muchos <a href="prod.html">productos</a>
a la venta.
```

En este ejemplo, el vínculo es a una página en la misma máquina y en el mismo directorio que la página que contiene el vínculo. Además de la localización y el nombre del documento web, un vínculo puede pasar parámetros junto a la petición de página. Cuando se especifican parámetros en una petición, normalmente es porque la página solicitada es ejecutable. La página web solicitada es capaz de acceder a los parámetros pasados con la petición y construir una página de respuesta. Los parámetros son pasados como pares `nombre=valor` separados por el carácter `&` y separados del resto de la URL mediante el carácter `?`. La siguiente petición de página contiene dos parámetros: `ProductoID` y `Categoria`. Al parámetro `ProductoID` le asignamos el valor `452` y al parámetro `Categoria` el valor `B`.

```
http://www.mialmacen.com/productos.jsp?ProductoID=452&Categoria=B
```

La página deseada es `productos.jsp`. La extensión de la página web da una pista de que la tecnología de servidor usada es JSP (*Java Server Pages*). Otras posibles extensiones son PHP (*Hypertext Preprocessor*), ASP (*Active Server Pages*), etc.

2.2.1.2. Formularios

Un formulario HTML es la parte de una página web que puede aceptar entradas del usuario. Un formulario HTML es una colección de campos que permiten a los usuarios introducir texto o seleccionarlo de una lista. Además de cuadros de texto (*text boxes*), los campos de un formulario pueden ser botones (*buttons*), casillas de verificación (*check boxes*) y botones de verificación (*radio buttons*). Si una página web tiene un formulario, el navegador visualizará dicho formulario con los controles de interfaz de usuario apropiados y permitirá al usuario introducir o cambiar sus valores. La mayoría de los formularios tienen un botón especial (*submit*) que cuando el usuario hace clic en él, envía el formulario y sus contenidos al servidor web como parte de una petición HTTP de página. Entonces el servidor web recibe una petición para una página web especial, o un módulo ejecutable, que es capaz de leer los valores de los campos del formulario y procesarlos en el servidor. El resultado final es una nueva página HTML que es devuelta al navegador que hizo la petición.

El concepto general es que la página web ejecutable es usada por el servidor web para

procesar los valores del formulario y producir una nueva página HTML que es devuelta al navegador. Muy frecuentemente, el procesamiento conlleva la comunicación con objetos del servidor o con bases de datos.

Un formulario es definido mediante la etiqueta `<form>`. Los dos atributos principales son `action` y `method`. El atributo `action` es la URL de la página web ejecutable que procesa el formulario. El atributo `method` especifica cómo serán enviados los datos al servidor, es decir, especifica el método HTTP de la petición de página. Los dos valores válidos son `GET` y `POST`. Cuando el valor `GET` es usado, los valores de todos los campos del formulario son concatenados a la URL como parámetros. El servidor web ve el envío del formulario como una petición `GET` típica, como si fuera realizada desde una etiqueta `<a>` estándar. La W3C no recomienda usar `GET`, ya que tiene algunos problemas de internacionalización y no funciona para formularios grandes. En cambio, es mejor usar `POST`. El método `POST` le dice al navegador que envíe los valores del formulario dentro del cuerpo de la petición HTTP de página.

Los principales elementos de los formularios HTML son: `<input>`, `<select>` y `<textarea>`. La etiqueta `<input>` es una etiqueta sobrecargada que puede ser configurada para actuar como un botón, una casilla de verificación, un botón de verificación o un campo de entrada de texto. La etiqueta `<select>` especifica un cuadro de lista desplegable o multiselección en el que el usuario puede seleccionar algo. La etiqueta `<textarea>` es un control de entrada de texto multilínea y permite a los usuarios introducir grandes bloques de texto.

2.2.2. Preparándonos para el futuro: XHTML

Es posible usar HTML 4.01 para construir sitios web estructurados y basados en estándares. Sin embargo, para hacer la transición a un marcado semántico y limpio, y estar mejor preparados para una posible transición a XML (Apartado 5.2.5) y otros futuros lenguajes de marcas, es recomendable emplear XHTML 1.0 Strict.

XHTML 1.0 es una reformulación de HTML 4 en XML 1.0 y fue desarrollada para sustituir a HTML. XHTML 1.0 estricto no permite el marcado de presentación (como, por ejemplo, usar la etiqueta `<font>`, usar el atributo `bgcolor`, etc.), con lo que obliga a separar la estructura de la presentación.

Vamos a ver una lista de las cosas más importantes que hay que considerar a la hora de utilizar XHTML 1.0 estricto en vez de HTML:

- Siempre usar minúsculas y poner entre comillas todos los atributos: Todos los nombres de etiquetas y atributos deben ir en minúscula. Todos los valores de los atributos deben estar entrecomillados.

  ✏ Incorrecto: `<A HREF="index.html" CLASS=interna>`
  Correcto: `<a href="index.html" class="interna">`

- Siempre cerrar todas las etiquetas: En HTML, se permite que algunas etiquetas no sean cerradas. Dichas etiquetas son automáticamente cerradas cuando se encuentra el comienzo de la siguiente etiqueta. XHTML no permite eso. Todas las etiquetas deben ir cerradas, incluso los elementos vacíos, es decir, aquellas etiquetas que en HTML no van en pareja, como `<img>`. Los elementos vacíos deben tener bien una etiqueta de cierre, bien terminar su etiqueta de apertura con `/>`.

  ✏ Incorrecto: `<li>Item 1`
  Correcto: `<li>Item 1</li>`

  ✏ Incorrecto: `<img src="image.jpg" alt="">`
  Correcto: `<img src="image.jpg" alt="" />`

- Los atributos no pueden ser minimizados: En HTML, ciertos atributos pueden ser minimizados. XHTML no permite esto.

  ✏ Incorrecto:
  ```
  <input type="checkbox" id="checkbox1"
    name="checkbox1" checked>
  ```
  Correcto:
  ```
  <input type="checkbox" id="checkbox1"
    name="checkbox1" checked="checked" />
  ```

- No usar elementos desaprobados: Un elemento o atributo desaprobado es aquel que ha quedado anticuado por la presencia de estructuras nuevas. Algunos elementos y atributos permitidos en HTML 4.01 transicional y XHTML 1.0 transicional son desaprobados en XHTML 1.0 estricto (y en HTML 4.1 estricto). Algunos ejemplos son `<font>`, `<center>`, `alink`, `align`, `width`, `height` (para algunos elementos) y `background`.

Todos los documentos HTML y XHTML deben tener una declaración de tipo de documento. El tipo de documento indica la versión de HTML o XHTML que utiliza el documento, y es usado a la hora de validar el documento. Si un documento es XHTML 1.0 estricto, deberemos definir la siguiente declaración de tipo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Esta declaración indica que versión de HTML o XHTML se usa en la página, de forma que los navegadores pueden saber qué sintaxis y gramática se usa, y los validadores puedan comprobar su validez.

2.2.3. Separar el estilo del contenido: CSS

Las hojas de estilo en cascada (CSS, *Cascading Style Sheets*)[5] son un mecanismo que permiten aplicar formato a los documentos escritos en HTML (y en otros lenguajes estructurados, como XML) separando el contenido de las páginas de su apariencia. Para el diseñador, esto significa que la información estará contenida en la página HTML, pero este archivo no debe definir cómo será visualizada esa información. Las indicaciones acerca de la composición visual del documento estarán especificadas en el archivo de la CSS. Además, hoy día es posible utilizar CSS para el diseño de las páginas web, ya que el soporte para CSS de los navegadores web se ha visto mejorado en los últimos años.

Hay varias versiones del estándar: CSS1 y CSS2, con CSS3 en desarrollo por el W3C. Los navegadores modernos implementan CSS1 bastante bien, aunque existen pequeñas diferencias de implementación según marcas y versiones de los navegadores. CSS2, sin embargo, está solo parcialmente implementado en los más recientes.

2.2.3.1. Introducción a CSS

Una hoja de estilo consiste en un conjunto de reglas de estilo que indican al navegador cómo presentar un documento. Hay varias formas de enlazar dichas reglas de estilo a nuestros documentos HTML, pero la forma más fácil es emplear el elemento HTML llamado `<style>`. Esta etiqueta se coloca dentro del cuerpo del elemento `<head>` y contiene las reglas de estilo para la página.

Es importante observar que aunque utilizar el elemento `<style>` es un buen método para experimentar con las hojas de estilo, éste tiene desventajas que deberían ser consideradas antes de utilizar este método en la práctica. Nosotros adoptaremos este método para introducir brevemente el uso de las hojas de estilo en esta sección.

Cada regla de estilo está formada por un selector, normalmente un elemento HTML como `<body>`, `<p>` o `<em>`, y el estilo aplicado a dicho selector, que no es más que un conjunto de propiedades. Cada propiedad toma un valor que, junto a la propiedad, describen cómo debería ser presentado el selector.

La sintaxis general de una regla de estilo es:

```
selector { propiedad1: valor1;
           propiedad2: valor2;
           ...
           propiedadN: valorN }
```

Como ejemplo, el siguiente fragmento de código define las propiedades `color` y `font-size` para los elementos `<h1>` y `<h2>`.

[5] La información oficial de la W3C sobre CSS está disponible en http://www.w3.org/Style/CSS/

```
<head>
<title>Ejemplo de CSS</title>
<style type="text/css">
  h1 { font-size: x-large; color: red }
  22 { font-size: large; color: blue }
</style>
</head>
```

La hoja de estilo del ejemplo le dice al navegador que muestre las cabeceras de primer nivel (elementos `<h1>`) en tamaño extra-largo y en rojo, y que muestre las cabeceras de segundo nivel con una fuente grande y azul.

Existen varias formas de aplicar hojas de estilo a los elementos de un documento HTML:

- Externa: Son varias las ventajas de mantener todas las reglas CSS en uno o más archivos independientes. El documento HTML reduce su tamaño, los archivos CSS son almacenados en la caché del navegador, por lo que sólo es necesario descargarlos una vez. Además, editando un único archivo (la hoja de estilo) podemos cambiar la presentación de un sitio web completo. Un archivo CSS externo podría ser el siguiente:

  ```
  h1 {
      font-weight:bold;
  }
  ```

  Podemos enlazar un archivo CSS a un documento CSS usando el elemento `<link>`.

  ```
  <link rel="stylesheet" type="text/css" href="styles.css" />
  ```

  o usando la regla import en un elemento `style`:

  ```
  <style type="text/css">
  @import url("styles.css");
  </style>
  ```

- En línea: Podemos aplicar estilos CSS directamente a un elemento HTML usando el atributo `style`:

  ```
  <h1 style="font-weight:bold;">Rubrik</h1>
  ```

  Aunque podemos hacerlo de esta manera, debemos evitarlo para evitar mezclar la estructura con la presentación.

- Interna: Las reglas de estilo están contenidas en el elemento `<style>`, que a su vez está contenido dentro del elemento `<head>` de un documento HTML:

```
<style type="text/css">
h1 {
    font-weight:bold;
}
</style>
```

  También debemos evitar este método, ya que lo mejor es tener el HTML y el CSS en archivos independientes.

2.2.3.2. Soporte de los navegadores

Como hemos comentado antes, en estos últimos años el soporte CSS ofrecido por los navegadores web se ha visto mejorado. Desafortunadamente no todos los desarrolladores de navegadores parecen estar interesados en implementar estándares abiertos, de forma que el nivel de soporte varía de un navegador a otro.

En la actualidad, los navegadores web que tienen un mejor soporte de CSS son Mozilla, Opera y Safari, al igual que los navegadores basados en los motores de navegación Gecko (Firefox, Camino, Netscape 6+) y WebCore (OmniWeb 4.5 y las posteriores versiones).

En cuanto al navegador Internet Explorer (IE) de Microsoft, la versión actual para Windows, IE 6, ofrece un soporte CSS inferior al de los anteriores navegadores. Aun así permite hacer las cosas más básicas. Por otro lado, IE 5 para Macintosh es compatible con CSS1 y no soporta gran parte de CSS2. La versión para Windows implementa bastantes de las características de CSS, aunque hay que tener cuidado con el uso de determinadas reglas CSS. Tanto las primeras versiones de Internet Explorer como de Netscape Navigator, o no tienen soporte CSS o éste es deficiente y con bastantes errores.

Puesto que la mayoría de la gente usa Internet Explorer para Windows, frecuentemente tomaremos ese navegador como el mínimo común denominador.

No todos los navegadores actualmente en uso tienen el nivel de soporte CSS requerido para visualizar correctamente un sitio web que hace un uso completo de CSS para el diseño de las páginas web. Afortunadamente, en la mayoría de los sitios web, sólo un pequeño porcentaje de los visitantes están utilizando un navegador demasiado antiguo como para no visualizar apropiadamente un diseño basado en CSS.

Es importante observar que esas personas no estarán privadas de acceder al contenido del sitio web. En los años 90, era muy frecuente usar *scripts* que comprobaban el navegador utilizado por el visitante, de forma que si empleaban un navegador «inadecuado» (normalmente cualquiera distinto a Internet Explorer para Windows) eran redireccionados a una página en la que se les decía que actualizarán su navegador para poder acceder al sitio.

Hoy día podemos tratar mejor este problema. Una gran ventaja de usar un XHTML semántico y lógico es que hace a los documentos accesibles y usables incluso sin CSS. La presentación, es decir, la apariencia del documento, no será la misma que en un navegador que admita CSS, pero el contenido estará allí. En la mayoría de los casos, para la mayor parte de los visitantes de un sitio, el contenido es realmente más importante que la forma en que es presentado. Por esto, es mejor enviar una página sin estilo a los navegadores sin soporte CSS que privarles de visitar el sitio.

Hay varias formas de hacer esto. Una de las más comunes es usar la directiva `@import` para enlazar con el archivo CSS asociado. Netscape 4 y los navegadores antiguos no saben interpretar la directiva `@import` y no importarán el archivo CSS.

2.2.4. Documentos estructurados con DOM

El DOM (*Document Object Model*, Modelo de Objetos de Documento) es una forma de representar documentos estructurados (tales como una página web HTML o un documento XML) que es independiente de cualquier lenguaje orientado a objetos.

Su finalidad es definir el conjunto de objetos que pueden componer documentos HTML o XML, así como las estructuras que se definen dentro de él, sus propiedades y sus métodos, independientemente del lenguaje de programación utilizado, con el fin de evitar problemas de compatibilidad entre navegadores.

En efecto, el DOM es una API (*Application Programming Interface*)[6] para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript (Javascript).

2.2.5. Páginas dinámicas con JavaScript

JavaScript es una tecnología del lado del cliente, es decir, que se ejecuta en el navegador del usuario. Permite dotar de dinamismo a las páginas web, así como que el cliente ejecute parte de la lógica de negocio de una aplicación web. JavaScript es un lenguaje de *script* (no compilado) que tiene sus raíces en el lenguaje de programación Java, aunque fue creado con la intención de ser más fácil de aprender y usar que este último.

JavaScript está basado en objetos, pero no es un lenguaje orientado a objetos, es decir, se pueden usar objetos predefinidos pero no soporta la definición de clases por parte del usuario ni la herencia. Además, es un lenguaje dirigido por eventos.

Algunas funcionalidades de JavaScript son:

- Validar los campos de los formularios HTML.
- Crear el efecto *rollover* (imágenes que cambian al pasar el cursor por encima).

[6] Una API es un conjunto de funciones ya definidas que permiten hacer solicitudes al sistema operativo o a una aplicación.

- Compensar las incompatibilidades del navegador.
- Abrir nuevas ventanas emergentes con un tamaño preciso.
- Comprobar la presencia o ausencia de Flash, QuickTime u otros *plugins*.
- Alternar contenido e imágenes dependiendo de la hora del día, del número de veces que el usuario haya visto cierta página, o simplemente al azar.

Existen varias formas de incluir código JavaScript en una página. La más frecuente es utilizar el elemento `<script>`, una extensión de HTML, usado para encerrar código JavaScript. Puesto que la etiqueta `<script>` puede ser usada para incluir otros lenguajes, debemos indicar el lenguaje utilizado mediante el atributo `language`. En el siguiente ejemplo, el elemento `<script>` contiene una llamada a una función que muestra una ventana de diálogo al usuario.

```
<script language="JavaScript">
  alert('Hola mundo.')
</script>
```

Desafortunadamente, se nos plantean ciertos problemas a la hora de definir *scripts*. No todos los navegadores soportan JavaScript y, aunque los navegadores se construyen para ser robustos y tolerantes a fallos, los navegadores más antiguos no tratan adecuadamente el código JavaScript, que no saben interpretar. El problema proviene del hecho de que si no entienden una etiqueta, el navegador la ignora. Normalmente, esto funciona. Por ejemplo, un navegador que no entienda la etiqueta `<font>` simplemente la ignora y continua usando la fuente por defecto. El problema sucede cuando un navegador antiguo ignora la etiqueta `<script>`. Las etiquetas de apertura y cierre son ignoradas, pero el texto contenido entre ellas, el código JavaScript, no es ignorado. El navegador intenta analizar esa información como si fuera cualquier otra parte del documento. Ya que JavaScript no sigue las mismas reglas de formateo y estructuración que HTML, es probable que un navegador antiguo reaccione mal y no visualice correctamente las páginas que contengan código JavaScript. Para evitar esta situación son usados los comentarios HTML y JavaScript para proteger las regiones donde se podría generar un error en un navegador no compatible con JavaScript. En el siguiente ejemplo, la etiqueta `<script>` define una función llamada `isPositiveNum()`. Esta función será llamada probablemente por la función de validación de un formulario antes de ser enviado al servidor. La primera línea del código es el inicio de un comentario HTML.

```
<script language="JavaScript">
<!-- Oculta el script a los navegadores antiguos
function esPositivo(s) {
  return (parseInt(s) > 0)
  }
```

```
// Fin de comentario -->
</script>
```

La última línea es un comentario JavaScript y es necesario para que sea ignorada por un intérprete de JavaScript.

2.2.6. Normas para mejorar la accesibilidad: WAI

Hablar de accesibilidad web es hablar de un acceso universal a la web, independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura, localización geográfica y capacidades de los usuarios.

Con esta idea de accesibilidad nace la Iniciativa de Accesibilidad Web, conocida como WAI (*Web Accessibility Initiative*)[7]. Se trata de una actividad desarrollada por el W3C, cuyo objetivo es facilitar el acceso de las personas con discapacidad, desarrollando pautas de accesibilidad, mejorando las herramientas para la evaluación y reparación de accesibilidad web, llevando a cabo una labor educativa y de concienciación en relación a la importancia del diseño accesible de páginas web, y abriendo nuevos campos en accesibilidad a través de la investigación en este área.

La idea principal radica en hacer la web más accesible para todos los usuarios independientemente de las circunstancias y los dispositivos involucrados a la hora de acceder a la información.

Teniendo esta idea como base, una página accesible lo será tanto para una persona con discapacidad, como para cualquier persona que se encuentre en circunstancias externas que dificulten el acceso a la información (en caso de ruidos externos, en situaciones donde nuestra atención visual y auditiva no estén disponibles, etc.).

Para hacer el contenido web accesible, se han desarrollado las denominadas pautas WCAG (*Web Content Accessibility Guidelines*, Pautas de Accesibilidad al Contenido en la Web)[8], cuya función principal es guiar el diseño de páginas web hacia un diseño accesible, reduciendo de esta forma barreras a la información. WCAG consiste en 14 pautas que proporcionan soluciones de diseño y que utilizan como ejemplo situaciones comunes en las que el diseño de una página puede producir problemas de acceso a la información. Las pautas contienen además una serie de puntos de verificación que ayudan a detectar posibles errores.

Cada punto de verificación está asignado a uno de los tres niveles de prioridad establecidos por las pautas.

- Prioridad 1: Son aquellos puntos que un desarrollador web tiene que cumplir ya que, de otra manera, ciertos grupos de usuarios no podrían acceder a la información del sitio web.

[7] La información oficial de la W3C sobre la WAI está disponible en http://www.w3.org/WAI
[8] Disponible en http://www.w3.org/TR/WCAG

- Prioridad 2: Son aquellos puntos que un desarrollador web debería cumplir ya que, si no fuese así, sería muy difícil acceder a la información para ciertos grupos de usuarios.
- Prioridad 3: Son aquellos puntos que un desarrollador web debería cumplir ya que, de otra forma, algunos usuarios experimentarían algunas dificultades para acceder a la información.

En función a estos puntos de verificación se establecen los niveles de conformidad:

- Nivel de Conformidad «A»: todos los puntos de verificación de prioridad 1 se satisfacen.
- Nivel de Conformidad «Doble A»: todos los puntos de verificación de prioridad 1 y 2 se satisfacen.
- Nivel de Conformidad «Triple A»: todos los puntos de verificación de prioridad 1,2 y 3 se satisfacen.

Las pautas describen cómo hacer páginas web accesibles sin sacrificar el diseño, ofreciendo esa flexibilidad que es necesaria para que la información sea accesible bajo diferentes situaciones y proporcionando métodos que permiten su transformación en páginas útiles e inteligibles.

Igualmente se han desarrollado Pautas de Accesibilidad para Herramientas de Autor cuyo objetivo es ayudar a los desarrolladores de software a la hora de crear herramientas de autor para producir contenido web accesible. También se han desarrollado Pautas de Accesibilidad para XML donde se explica cómo asegurar la accesibilidad de aplicaciones basadas en XML. Y por último Pautas de Accesibilidad para Agentes de Usuario 1.0, donde se explica cómo hacer accesible los navegadores, reproductores multimedia y otras tecnologías asistentes.

Por otro lado se han desarrollado otro tipo de documentos como las Técnicas para Pautas de Accesibilidad al Contenido en la Web, que ofrecen una serie de ejemplos de etiquetado y explicaciones muy detalladas de cómo implementar las Pautas de Accesibilidad al contenido en la web. Entre ellas se pueden destacar Técnicas esenciales para Pautas de Accesibilidad al Contenido en la Web 1.0, las Técnicas HTML para Pautas de Accesibilidad al Contenido a la Web 1.0 y las Técnicas CSS para Pautas de Accesibilidad al Contenido en la Web 1.0.

Un ejemplo de prioridad 1 sería la identificación clara de cualquier cambio de idioma que se pueda producir en el texto de un documento. Es decir, si se utilizan diferentes idiomas es necesario que cualquier cambio esté claramente señalado con el atributo `lang`.

Un ejemplo de código correcto sería el siguiente:

```
<p>Buenos días Philip</p>
<p><span lang="en">Good morning</span>. Respondió Philip en inglés.</p>
<p>¿Qué tal estás?</p>
<p><span lang="fr">Très bien.</span>
   Volvió a responder, pero esta vez en francés.</p>
```

Otro ejemplo de prioridad 1 sería la utilización de la etiqueta `alt` para incorporar texto equivalente al contenido de una imagen cuando se quieren utilizar gráficos; esto permitiría que dispositivos o personas que no pueden visualizar los gráficos, obtengan una representación alternativa textual. El código correspondiente sería:

```
<img src="wai.png" alt="Web Accessibility Initiative" />
```

A continuación presentamos una guía breve para crear sitios web accesibles, desarrollada por el W3C:

- Imágenes y animaciones: Use el atributo `alt` para describir la función de cada elemento visual.
- Mapas de imagen: Use el elemento `map` y texto para las zonas activas.
- Multimedia: Proporcione subtítulos y transcripción del sonido, y descripción del vídeo.
- Enlaces de hipertexto: Use texto que tenga sentido leído fuera de contexto. Por ejemplo, evite «pincha aquí».
- Organización de las páginas: Use encabezados, listas y estructura consistente. Use CSS para la maquetación donde sea posible.
- Figuras y diagramas: Descríbalos brevemente en la página o use el atributo `longdesc`.
- *Scripts*, *applets* y *plugins*: Ofrezca contenido alternativo si las funciones nuevas no son accesibles.
- Marcos: Use el elemento `noframes` y títulos con sentido.
- Tablas: Facilite la lectura línea a línea. Resuma.
- Revise su trabajo: Verifique. Use las herramientas, puntos de comprobación y pautas de WCAG.

2.2.7. Validación de páginas webs

La validación consiste en verificar si la codificación de una página web cumple con los estándares que el W3C ha definido. Aquí seguiremos las recomendaciones sugeridas por el Grupo de Interés de Aseguramiento de la Calidad del W3C (QAIG, *Quality Assurance Interest Group*).

2.2.7.1. Validación de HTML

Un validador HTML simplemente comprueba la corrección de un documento frente al DOCTYPE declarado.

Si se quiere verificar la validez de un documento al ser mostrado por un navegador web, podemos usar el validador HTML del W3C. El validador HTML devolverá una lista de errores de acuerdo al DOCTYPE HTML elegido.

Además del validador HTML del W3C, existen otros:

- Validador HTML del W3C: http://validator.w3.org
- Validador HTML del WDG: http://www.htmlhelp.com/tools/validator
- Doctor HTML: http://www.doctor-html.com/RxHTML/cgi-bin/single.cgi

2.2.7.2. Validación de enlaces

Una cuestión importante en muchos sitios web es la persistencia de las URLs. Cuando en los documentos se incluyen enlaces a otros sitios web, se esperan que aquellos enlaces permanezcan estables y persistentes. Esto significa que la información a la que se quiere apuntar aún estará allí cuando alguien que lea el sitio web haga clic en uno de los enlaces. También podríamos querer verificar y garantizar que los enlaces proporcionados a otras páginas web u otras secciones del propio sitio web tampoco contienen errores. Hay una herramienta que ha sido desarrollada con este propósito: W3C Link Checker[9].

Link Checker genera un informe sobre los enlaces. La longitud del informe depende del tiempo que lleve alcanzar y verificar todos los enlaces contenidos en la página. Para verificar el enlace, el programa maneja las peticiones de cabeceras HTTP del documento (peticiones HEAD). Si el servidor está mal configurado, podemos obtener un informe erróneo aunque el enlace sea correcto, pero sólo porque el servidor es incapaz de proporcionar la cabecera. En este caso, deberíamos escribir al *webmaster* del sitio web para pedirle que arregle la configuración del servidor.

```
Checking link http://webstandards.org/
HEAD http://webstandards.org/  fetched in 0.1s
```

Lo anterior es un ejemplo de lista. Proporciona el tiempo que tarda en alcanzar el enlace.

Tras la lista de enlaces, tendremos un informe de los enlaces que están rotos o redireccionados, lo cual nos ayudará a corregir enlaces incorrectos.

[9]Disponible en http://validator.w3.org/checklink

2.2.7.3. Validación de CSS

Igual que con el servicio de validación HTML, con el servicio de validación CSS[10] del W3C podremos comprobar la validez de las hojas de estilo.

2.2.7.4. Validación de la accesibilidad

W3C no ha desarrollado su propia herramienta de validación automática. Sí nos ofrece la posibilidad de insertar un logotipo de cumplimiento de los distintos niveles de prioridad de sus pautas WAI.

WAI-W3C ha aceptado la validación que realiza el programa desarrollado por CAST (*Center for Applied Special Technology*) y denominado Bobby[11]. Recientemente ha sido adquirido por la empresa Watchfire quien presta servicio gratuito de validación en línea (la validación se hace página por página). Este programa va por su versión 4 y da la posibilidad de descargarlo en nuestro ordenador, previo pago, para analizar las páginas en modo local.

Al igual que en los casos anteriores, después del análisis se nos ofrecen los resultados con las soluciones a los posibles errores encontrados. Hay que tener presente que algunos puntos de verificación no se pueden analizar de forma automática y deberemos verificarlos personalmente por otros medios no automatizados.

En España se ha desarrollado una herramienta de validación de accesibilidad denominada TAW[12]. Está en su segunda versión. Ya se puede descargar y nos permite analizar página a página nuestro sitio o todo en su conjunto y, además, está en castellano y es gratuito.

Siguiendo el mismo procedimiento que el resto de los validadores, nos ofrece los resultados del análisis y la posible solución a los errores. TAW tiene la misma limitación que Bobby en lo que respecta al análisis manual (o personal) de alguno de los puntos de verificación, que siempre tendremos que tener en cuenta.

2.3. Sistemas para la web

Las aplicaciones web usan tecnologías de servidor (como JSP, ASP, PHP, ColdFusion, etc.) para hacer sus contenidos dinámicos y permitir a los usuarios del sistema afectar a la lógica de negocio[13] en el servidor. La distinción entre sitios y aplicaciones web es sutil y depende de la capacidad que tenga el usuario para afectar o influir en el estado de la lógica de negocio en el servidor. Ciertamente, si no existe lógica de negocio en el servidor, el sistema no debería ser denominado aplicación web. Por tanto, una aplicación web es un sistema en el que el servidor

[10]Disponible en http://jigsaw.w3.org/css-validator

[11]Disponible en http://bobby.watchfire.com

[12]Disponible en http://www.tawdis.net

[13]La lógica de negocio es la parte más importante de una aplicación. En ella se implementan las reglas que definen la funcionalidad del sistema y necesita de forma habitual conceptos como el acceso a datos, la gestión de transacciones y la exposición por servicios web.

web o, en su caso, el servidor de aplicaciones[14], permite que el cliente pueda modificar la lógica de negocio desde un navegador web.

La distinción es todavía más sutil en el caso de los motores de búsqueda (*search engines*), en los que el usuario introduce criterios de búsqueda. Los motores de búsqueda que son sitios web, simplemente aceptan esta información, la utilizan para realizar una consulta `SELECT` a una base de datos y devuelven los resultados. Cuando el usuario termina de usar el sistema, no hay cambios observables en el estado del motor de búsqueda, excepto en los registros de uso y los contadores de visitas. Esto contrasta con las aplicaciones web que, por ejemplo, aceptan información de registro *online*. Un sitio web que permite al usuario inscribirse en un curso de inglés tiene un estado diferente cuando el usuario termina de utilizar la aplicación.

La arquitectura de un sitio web es simple. Contiene los componentes principales de un sitio web: un servidor web, una conexión de red y los navegadores de los clientes. Las aplicaciones web también incluyen un servidor de aplicaciones. El uso de un servidor de aplicaciones web permite al sistema manejar la lógica de negocio y el estado del sistema.

2.4. Páginas web

Siguiendo la clasificación propuesta por Conallen [16], podemos diferenciar dos tipos de páginas web:

- Estáticas. Una página web estática es aquella cuya información de los formularios y cuyo contenido no es procesado por el servidor antes de ser enviada a un cliente. Típicamente, estas páginas contienen alguna explicación textual, tales como direcciones o información de ayuda, o formularios HTML. Cuando un servidor web recibe una petición de una página HTML, el servidor simplemente envía el archivo sin someterlo a ningún procesamiento.

- Dinámicas. Una página web dinámica es aquella que es procesada, de alguna forma, por el servidor antes de ser enviada a un cliente. Normalmente, estas páginas son implementadas como páginas de *script* en el servidor (páginas ASP, JSP, PHP, Cold Fusion, etc.) que son procesadas por un servidor de aplicaciones o por módulos ejecutables (ISAPI, NSAPI). Estas páginas potencialmente tienen acceso a todos los recursos del servidor, incluyendo los componentes lógicos de negocio, bases de datos, etc.

También se suelen denominar dinámicas aquellas páginas que contienen código JavaScript (páginas dinámicas del lado del cliente). Realmente, estas páginas son enviadas a los clientes sin ningún tipo de procesamiento, por lo que estrictamente serían páginas estáticas según la

[14]Los servidores de aplicaciones son aquellas aplicaciones que trabajan junto con los servidores web, dotando a estos últimos de capacidad de procesamiento. De esta manera, proporcionan a los desarrolladores la oportunidad de elaborar páginas web dinámicas. Un servidor de aplicaciones JSP muy conocido es Tomcat.

clasificación anterior. Lo que pasa es que a los ojos de un cliente estas páginas pueden presentar algún tipo de dinamismo (*rollover* de imagen, desplazamiento de imágenes...) proporcionado por el código JavaScript asociado.

Por tanto, otra clasificación de las páginas web sería:

- Estáticas. Son aquellas que se presentan sin movimiento y sin funcionalidades más allá de los enlaces.

- Dinámicas. Son aquellas páginas que tienen efectos especiales y en las que podemos interactuar. Estas, a su vez, las podemos dividir en dos grupos en función de dónde se lleva a cabo el procesamiento de la página, es decir, el computador que cargará con el peso adicional que supone que la página realice efectos y funcionalidades:

  - Dinámicas de cliente. Son las páginas dinámicas que se procesan en el cliente. En estas páginas toda la carga de procesamiento de los efectos y funcionalidades la soporta el navegador.

    Usos típicos de las páginas de cliente son efectos especiales para webs como *rollovers* o control de ventanas, presentaciones en las que se pueden mover objetos por la página, control de formularios, cálculos, etc.

    Las páginas dinámicas de cliente se escriben en dos lenguajes de programación principalmente: JavaScript y Visual Basic Script (VBScript). Este último es un lenguaje de programación de *scripts* del lado del cliente, pero sólo compatible con Internet Explorer. Es por ello que su utilización está desaconsejada a favor de JavaScript.

  - Dinámicas de servidor. Son las páginas dinámicas que son reconocidas, interpretadas y ejecutadas por el propio servidor.

    Las páginas del servidor son útiles en muchas ocasiones. Con ellas se puede hacer todo tipo de aplicaciones web. Desde agendas a foros, sistemas de documentación, estadísticas, juegos, chats, etc. Son especialmente útiles en trabajos que se tiene que acceder a información centralizada, situada en una base de datos en el servidor, y cuando por razones de seguridad los cálculos no se pueden realizar en el ordenador del usuario.

    Existen varios lenguajes con los que escribir páginas dinámicas de servidor: ASP, PHP, JSP, ColdFusion, Perl (mediante la interfaz CGI[15]), etc.

    Las ventaja principal de este tipo de programación es que el cliente no puede ver los *scripts*, ya que se ejecutan y transforman en HTML antes de enviarlos. Además son

[15]CGI (*Common Gateway Interface*, Interfaz Común de Pasarela) es una interfaz de intercambio de datos estándar mediante la cual se organiza el envío y recepción de datos entre un navegador y los programas residentes en servidores web. Al tratarse de una interfaz, no existe ningún tipo de dependencia con el lenguaje de programación empleado, por lo que, en principio, cualquier lenguaje es susceptible de ser utilizado para desarrollar programas CGI.

independientes del navegador del usuario, ya que el código que reciben es HTML fácilmente interpretable.

Como desventajas se puede señalar que será necesario un servidor más potente y con más capacidades que el necesario para las páginas de cliente. Además, estos servidores podrán soportar menos usuarios concurrentes, porque se requerirá más tiempo de procesamiento para cada uno.

Capítulo 3

Herramientas

En este capítulo presentamos las herramientas software empleadas en la elaboración de la aplicación web del presente proyecto. Todas estas herramientas funcionan bajo Windows XP, el sistema operativo que hemos utilizado para este proyecto, aunque todas ellas también están disponibles para los sistemas Linux. Debido al presupuesto disponible para el mismo, hemos optado por utilizar aplicaciones *open source* (código fuente abierto), que suelen ser de bajo coste (o incluso nulo). De esta manera ofrecemos una alternativa a la hora de desarrollar un sistema web, intentando evitar la piratería, una práctica éticamente incorrecta que es muy habitual en nuestros días.

El capítulo se divide en los siguientes apartados:

- Introducción al *open source*. Presentamos el movimiento *open source*, así como los beneficios que nos puede ofrecer [49, 60].

- Servidor HTTP: Tomcat. Breve introducción al servidor web utilizado durante el desarrollo de la aplicación web [3, 63].

- Desarrollo web. En este apartado, hablaremos sobre las herramientas empleadas para la creación del sistema web: el IDE Eclipse, el navegador Firefox y la aplicación de tratamiento de imagen GIMP [33, 36, 65, 78].

- Modelado de bases de datos: DBDesigner. Hablaremos de la aplicación que hemos utilizado para construir el modelo lógico de la base de datos de la aplicación.[4, 27].

- Modelado UML: Poseidon for UML. Finalmente, presentaremos la herramienta utilizada para la creación de los modelos UML (*Unified Modeling Language*, Lenguaje Unificado de Modelado) de la aplicación.

3.1. Introducción al Open Source

La idea que late detrás del *open source* es bien sencilla: cuando los programadores en Internet pueden leer, modificar y redistribuir el código fuente de un programa, éste evoluciona, se desarrolla, mejora. Los usuarios lo adaptan a sus necesidades, corrigen sus errores. Y esto puede ocurrir a tal velocidad que el que esté acostumbrado al ritmo de desarrollo de los programas comerciales no lo puede concebir.

Open source podría traducirse como «código fuente abierto»: un programa que ofrece al usuario la posibilidad de entrar en sus tripas para poder estudiarlo o modificarlo. Pero no sólo hace referencia al libre acceso al código fuente. Las condiciones de distribución de un programa *open source* deben cumplir una serie de criterios. La intención de la Definición de *Open Source* (*Open Source Definition*)[1], definida por la OSI (*Open Source Initiative*)[2] es establecer que esos criterios contengan la esencia de lo que los programadares quieren que signifique: que aseguren que los programas distribuidos con licencia *open source* estarán disponibles para su continua revisión y mejora para que alcancen niveles de fiabilidad que no pueda conseguir ningún programa comercial cerrado[3].

A la idea esencial del *open source*, ofrecer programas con acceso al código fuente, van unidas una serie de conceptos:

- Bajo coste. Las aplicaciones *open source* tienen un coste muy bajo o incluso nulo en muchos casos, lo cual repercute directamente en el coste de un proyecto de desarrollo de software.
- Flexibilidad. Si el código fuente está disponible, los desarrolladores pueden modificar los programas a su antojo. Además, se produce un flujo constante de ideas que mejora la calidad de los programas.
- Fiabilidad y seguridad. Con varios programadores a la vez escrutando el mismo trabajo, los errores se detectan y corrigen antes, por lo que el producto resultante es más fiable y eficaz que el comercial.
- Rapidez de desarrollo. Las actualizaciones y ajustes se realizan a través de una comunicación constante vía internet.
- Relación con el usuario. El programador se acerca mucho más a las necesidad real de su cliente, y puede crear un producto específico para él.

La conexión entre el movimiento de código fuente abierto y las empresas de software ha sido importante para dar validez a sus programas informáticos dentro del mundo corporativo.

[1] Disponible en http://www.opensource.org/docs/definition.php
[2] Sitio web oficial de la OSI: http://www.opensource.org
[3] Un programa cerrado o de código cerrado es aquel cuyo código fuente no se puede inspeccionar.

Empresas como IBM, Sun o Netscape, aunque no son empresas integradas a este movimiento, han publicado parte de sus programas en régimen de código fuente abierto, y han dado así un empuje a esta filosofía. Aun así, todas estas empresas todavía tienen aplicaciones basadas en el modelo de software cerrado.

Por otra parte, existen empresas como RedHat, Suse o Ximian que forman parte de este movimiento: llevan a cabo el desarrollo de todas sus aplicaciones con un proceso de código fuente abierto y distribuyen sus programas a un coste muy bajo. Los beneficios los obtienen con la venta de servicios para estos programas gratuitos, como la formación, el asesoramiento técnico o su personalización. Los usuarios no pagan por el programa, sino por los servicios basados en él. Así, pues, un estudiante universitario puede utilizar un programa totalmente gratis, mientras que un usuario profesional que necesite los servicios de instalación y mantenimiento del mismo programa tendrá que pagar por ellos. Esto es muy positivo para el usuario, ya que puede disponer del programa, utilizarlo y sólo pagar por los servicios que realmente necesita, a diferencia del modelo actual, en que hay que pagar siempre una licencia por el uso de un programa en cada ordenador.

3.2. Servidor HTTP: Tomcat

El servidor Tomcat[4] es una aplicación Java desarrollada por el Grupo Apache (*Apache Group*)[5] que sirve para interpretar las páginas web con tecnología JSP (Apartado 5.2.2) y Servlet (Apartado 5.2.1). Se distribuye bajo la licencia Apache Software License (ASF), aprobada como licencia *open source* por la OSI. Es la implementación de referencia de Sun para los contenedores de servlets. Puede actuar como un pequeño servidor web pero no está realmente diseñado para ello.

Durante el desarrollo de la aplicación web, Tomcat se ha usado como servidor web de forma local y sólo para el desarrollo la aplicación, ya que es lento y no ofrece las características que otros servidores web ofrecen. La configuración más normal en un entorno de producción es la instalación del servidor Apache[6] junto con Tomcat, donde Tomcat añade el soporte JSP al servidor Apache uniéndolos mediante un módulo de conexión.

3.3. Desarrollo web

Existen multitud de aplicaciones para la creación de sistemas web. Desde un simple editor de texto hasta un editor WYSIWYG (*What You See Is What Tou Get*)[7]. Actualmente, el programa

[4]Disponible en su sitio web oficial http://jakarta.apache.org/tomcat

[5]Sitio web del Grupo Apache: http://www.apache.org

[6]Disponible en su sitio web oficial http://httpd.apache.org

[7]WYSIWYG es el acrónimo de *What You See Is What You Get* («lo que ves es lo que obtienes»). Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso.

más utilizado en el sector del diseño y la programación web es Macromedia Dreamweaver, un editor WYSIWYG de páginas web, desarrollado y comercializado por la empresa Macromedia. Todas las licencias de uso de este programa son de pago.

Para este proyecto nos hemos decantado por usar el entorno de desarrollo Eclipse, con el que crearemos las páginas web, tanto HTML como JSP, así como las hojas de estilo CSS. Utilizaremos también el navegador Firefox, útil a la hora crear el diseño de la web y para la corrección de los errores JavaScript de nuestras páginas. Para la creación de imágenes para la web, nos hemos decidido por la aplicación GIMP, la alternativa de fuente abierta para el conocido Adobe Photoshop.

3.3.1. Eclipse

Eclipse[8] es un IDE (*Integrated Development Tool*, Entorno Integrado de Desarrollo) de código abierto, multiplataforma y extensible. Aunque fue creado por IBM, en la actualidad es la Fundación Eclipse, una organización sin ánimo de lucro, la que se encarga de su desarrollo.

Programado en Java e inicialmente orientado a Java, el entorno Eclipse es extensible para el desarrollo en otros lenguajes a través de un avanzado sistema de *plugins*: existe un *plugin* para trabajar en PHP, otro para trabajar en Perl, otro para UML, etc. También existen *plugins* para la edición de XML, para iniciar/detener Tomcat, etc. Podemos encontrar gran cantidad de *plugins* disponibles para Eclipse en [23] y [22].

La versión 3.x de Eclipse ofrece las siguientes características:

- Editor de texto con resaltado de sintaxis.
- Tests unitarios con JUnit, una herramienta diseñada para implementar y automatizar la realización de pruebas a software orientado a objetos.
- Control de versiones con CVS (*Control Version System*), existiendo un *plugin* para SVN (*Subversion*).
- Integración con Ant, una herramienta para compilar programas Java.
- Asistentes (*wizards*) para creación de proyectos, clases, tests, etc.
- Refactorización de código.

3.3.1.1. *Plugins* necesarios

Eclipse tiene un tipo de proyecto para el desarrollo de aplicaciones Java, normalmente insuficiente para el desarrollo de aplicaciones web Java. Una aplicación web Java está caracterizada por una estructura de directorios especial y un conjunto de archivos específicos: archivos JSP,

[8] Sitio web oficial en http://www.eclipse.org

archivos de configuración XML, servlets, etc. Para poder suplir esta carencia necesitamos instalar un *plugin* de Sysdeo[9]. Éste no sólo nos proporciona un proyecto específico para aplicaciones web sino que también nos ofrece la posibilidad de controlar el servidor Tomcat desde el propio entorno Eclipse. Para una explicación detallada del proceso de instalación y configuración de Eclipse y el *plugin* de Sysdeo, consultar [36].

También necesitaremos un *plugin* denominado Struts Console[10]. Este *plugin* nos permite editar gráficamente todos los archivos de configuración XML de Struts, el *framework* que hemos empleado para construir la aplicación web de este proyecto.

Otro *plugin* muy útil es Eclipse Colorer[11], que ofrece el coloreado de sintaxis para más de cien lenguajes.

3.3.2. Mozilla Firefox

Mozilla Firefox[12] es un navegador web *open source* y multiplataforma de la Fundación Mozilla[13]. Entre sus características encontramos la navegación por pestañas, el bloqueo de ventanas emergentes (*popups*), un buscador integrado en la interfaz, un gestor de descargas y la posibilidad de ampliar las funcionalidades de Firefox mediante la instalación de extensiones.

El navegador Firefox incluye diversas herramientas que resultan muy útiles para el desarrollo y la depuración de errores de sitios web:

- El inspector DOM. Es una herramienta que se puede usar para examinar y editar la jerarquía DOM (árbol de elementos HTML presentes en un documento: `h1`, `h2`, `div`, etc., Apartado 2.2.4) de cualquier documento web, permitiéndonos ver la página exactamente como el navegador la ve. El inspector DOM viene con Firefox, pero durante el proceso de instalación es necesario indicar que lo deseamos instalar, ya que por defecto no lo hace.

- La consola JavaScript. Este programa viene «de serie» en Firefox y es útil para corregir errores en páginas o rastrear problemas en el código. Nos muestra todos los errores, advertencias y mensajes generados por el código JavaScript (Apartado 2.2.5) analizado.

- La extensión Web Developer[14]. Se trata de una nueva barra de herramientas totalmente integrada a la interfaz del navegador web Firefox, de gran utilidad para los desarrolladores web, ya que cuenta con un amplio número de potentes utilidades de manipulación CSS (Apartado 2.2.3), opciones de validación (Apartado 2.2.7), así como herramientas que permiten controlar el tipo de información visualizada acerca de la página web.

[9] Disponible en http://www.sysdeo.com/eclipse/tomcatPlugin.html
[10] Disponible en http://www.jamesholmes.com/struts/console
[11] Disponible en http://colorer.sourceforge.net
[12] Disponible en http://www.mozilla-europe.org/es/products/firefox
[13] Sitio web oficial de la Fundación Mozilla: http://www.mozilla-europe.org/es
[14] Disponible en http://chrispederick.com/work/firefox/webdeveloper/

Esta extensión ofrece potentes características CSS, como la posibilidad de aplicar una hoja de estilo personal a una página web o incluso editar el CSS de la página web visualizada. Adicionalmente, permite ver el estilo aplicado a un elemento de la página web abierta con sólo seleccionarlo.

En cuanto a las opciones de validación, permite validar el CSS, HTML, WAI, los enlaces y la sección 508 (normativa estadounidense sobre la accesibilidad en la web) de una página web.

También podemos obtener información acerca del tamaño y resolución de las imágenes, ocultarlas, cambiar la imagen de fondo, limpiar la caché y el historial, borrar los cookies, abrir el inspector DOM, redimensionar la ventana del navegador a 800×600 u otro tamaño, editar formularios, visualizar el código fuente, así como obtener numerosa información acerca de la página web.

3.3.3. GIMP

GIMP (*GNU Image Manipulation Program*) es un programa para el tratamiento de imágenes desarrollado bajo la licencia GPL (*GNU General Public License*, Licencia Pública General de GNU)[15], erigiéndose como la alternativa libre por excelencia a otras aplicaciones similares de pago. Es ideal para tareas como retocar fotografías, crear imágenes para la web, etc.

El programa dispone de un completísimo paquete de herramientas y *plugins*, soporta capas y canales, dispone de una base de datos integrada, capaz de realizar llamadas a funciones internas de GIMP desde aplicaciones externas. También posee funcionalidad de *scripts*, *undos* indefinidos, editor de animaciones, etc.

3.4. Modelado de bases de datos: DBDesigner

Para crear el modelo lógico de base de datos (diagrama Entidad/Relación o diagrama ER) de la base de datos hemos utilizado la herramienta DBDesigner[16] en su versión 4, un entorno visual integrado para el diseño, modelado, creación y mantenimiento de bases de datos relacionales. Es una aplicación gratuita, distribuida bajo licencia GPL, y disponible para plataformas Windows y Linux. DBDesigner 4 ha sido creado para funcionar íntegramente con MySQL (Capítulo 4), aunque también puede ser capaz de soportar otros sistemas de bases de datos por medio de la inclusión de *plugins*.

El listado de características a destacar de DBDesigner 4 es el que sigue:

- Creación de bases de datos, tablas, campos, relaciones, etc. de una manera totalmente visual.

[15] Disponible en http://es.gnu.org/licencias/gples.html
[16] Disponible en su sitio web oficial http://www.fabforce.net/dbdesigner4/

- Generación automática del código SQL a partir del modelo ER.
- Ingeniería inversa (generar el modelo ER a partir de una base de datos existente) para MySQL, Oracle, MSSQL (Microsoft SQL Server) y cualquier base de datos ODBC (*Open Database Connectivity*).
- Sincronización del modelo visual con la base de datos, para ir aplicando los últimos cambios que se hacen sobre el modelo relacional directamente a la base de datos.
- Guarda los proyectos en XML, lo que nos permite trabajar con ellos a través de otras aplicaciones.
- Posee un editor con el que podemos crear y ejecutar consultas SQL.
- Dispone de un sistema de *plugins* para extender y personalizar la herramienta.
- Posibilidad de exportar el modelo como imagen.

3.5. Modelado UML: Poseidon for UML

La herramienta de modelado UML que hemos elegido para este proyecto es Poseidon for UML Communty Edition v3.1. La edición Community Edition (CE)[17] es gratuita y ofrece las siguientes características:

- Está escrita en Java, por lo que es altamente portable.
- Soporta los 9 diagramas de UML 1.4.
- Permite guardar los diagramas en formato UML 2.0 Diagram Interchange Standard.
- Utiliza XMI (*XML Metadata Interchange*) 1.2 como formato estándar para guardar los diagramas. Además, puede cargar diagramas XMI 1.0, 1.1 y 1.2.
- Opciones avanzadas de impresión.
- Exportación de diagramas a formatos gif, ps, eps, svg, wmf, jpg y png.
- Internacionalización y localización para inglés, alemán, ruso, francés, español y chino.
- Generación de código Java.
- Instalación y actualizaciones sencillas con Java Web Start.

En [34] puede verse una comparación de algunas herramientas UML de libre distribución (o con pocas restricciones) y en [54] puede encontrar una lista de herramientas UML, tanto comerciales como no comerciales, bastante actualizada.

[17] Disponible en su sitio web oficial http://www.gentleware.com

Capítulo 4

MySQL

A lo largo de este capítulo presentaremos el sistema gestor de bases de datos (SGBD) utilizado para la realización de este proyecto: MySQL. El capítulo se divide en los siguientes apartados:

- Introducción. Se explica qué es MySQL y qué características posee [53].
- Por qué usar MySQL. Este apartado presenta una comparativa entre el SGBD elegido y otros disponibles en el mercado [21, 52].
- Programas clientes. Introduciremos brevemente tres de los posibles programas clientes que pueden ser utilizados para trabajar con MySQL: la herramienta de líneas de comandos `mysql` y las herramientas gráficas MySQL Administrator y MySQL Query Browser [29, 53].
- Acceso a bases de datos con Java: JDBC. Finalmente, terminaremos hablando del API JDBC (*Java Database Connectivity*), el cuál proporciona acceso a cualquier tipo de fuente de información, incluida la base de datos MySQL, a través del lenguaje Java [29].

4.1. Introducción

MySQL[1] es un sistema gestor de bases de datos, desarrollado por la compañía MySQL AB. Una base de datos es una colección estructurada de información. Puede ser una simple lista de la compra, una galería de imágenes o la enorme cantidad de información de una red corporativa. Para añadir, acceder o procesar información almacenada en una base de datos, se necesita un sistema que gestione la base de datos. Este sistema es llamado SGBD.

Existen varios tipos de SGBD. MySQL es un SGBD relacional. Una base de datos relacional almacena información en tablas separadas en vez de poner todos los datos en un gran «almacén».

[1] Disponible en su sitio web oficial http://www.mysql.com/products/mysql

Esto añade velocidad y flexibilidad. MySQL soporta SQL (*Structured Query Language*, Lenguaje de Consultas Estructurado), el lenguaje estándar comúnmente usado para acceder a bases de datos relacionales.

Por otro lado, MySQL es software libre (*free software*), usando la licencia GPL (*GNU General Public License*, Licencia Pública General de GNU [32]). También existen licencias comerciales no GPL para aquellos que no quieran estar restringidos por los términos de GPL.

4.1.1. Características principales

Con el fin de tener una visión global de las capacidades del servidor MySQL, expondremos en la presente sección sus características más destacables. Para una información más detallada, acudir a [53].

- Implementa gran parte de la sintaxis ANSI SQL 99, junto con extensiones propias. También reconoce la sintaxis de otros sistemas de bases de datos, facilitando así la creación de aplicaciones portables.

- Está disponible para una gran variedad de plataformas, tales como Linux, Microsoft Windows, FreeBSD, Sun Solaris, IBM's AIX, Mac OS X, HP-UX, AIX, QNX, Novell NetWare, SCO OpenUnix, SGI Irix, and Dec OSF.

  Podemos conectarnos a un servidor MySQL prácticamente desde cualquier plataforma y a través de casi cualquier lenguaje de programación, utilizando para ello la biblioteca cliente estándar o alguno de los productos de la familia Connector de controladores de la base de datos MySQL.

  La familia Connector está formada por dos controladores de bases de datos: Connector/ODBC y Connector/J. EL controlador Connector/ODBC es una biblioteca que implementa las funciones del API ODBC (*Open Database Connectivity*) de Microsoft Corporation. El API ODBC permite a las aplicaciones acceder a diferentes SGBDs sin tener que escribir una versión distinta del código para cada una de las bases de datos. El controlador ODBC se encarga de procesar las llamadas a funciones ODBC, enviar peticiones SQL al servidor MySQL y devolver el resultado a la aplicación. La función del controlador Connector/J es la misma, salvo que implementa el API JDBC (Apartado 4.4) de Sun, creado para facilitar el acceso a bases de datos desde el lenguaje de programación Java.

- Motores de almacenamiento independientes. Se puede elegir el tipo de almacenamiento más apropiado para cada situación. Si necesitamos bloqueos y transacciones a nivel de fila, podemos usar el motor InnoDB. Si, en cambio, no se requieren transacciones, entonces podemos usar el motor MyISAM para conseguir así un máximo rendimiento.

- Incluye control de transacciones, usando los motores InnoDB o Berkeley DB. El motor InnoDB también admite la creación de restricciones de clave foránea.

- Sistema de seguridad flexible, incluyendo soporte para SSL (*Secure Sockets Layer*), un protocolo desarrollado por la empresa Netscape Communications Corporation para permitir confidencialidad y autenticación en Internet. La idea que persigue SSL es encriptar la comunicación entre servidor y cliente mediante el uso de claves y algoritmos de encriptación.
- Uso de caché para las consultas. Esta característica permite un gran aumento en la velocidad de las consultas ejecutadas más frecuentemente, guardando en una caché la repuesta de la consulta y devolviéndosela al cliente en vez de ejecutar la consulta de nuevo.
- Replicación de bases de datos.
- Búsqueda e indexación en textos. Permite buscar en campos de texto palabras específicas o frases, pudiendo establecer órdenes de relevancia.
- Biblioteca de servidor MySQL embebido (*embedded MySQL server library*, `libmysqld`). Esta biblioteca hace posible ejecutar un servidor MySQL dentro de una aplicación cliente. Los principales beneficios son el incremento de la velocidad y la simplificación de la gestión propios de las aplicaciones embebidas.

4.2. Por qué usar MySQL

Hay muchas bases de datos disponibles en el mercado, pero MySQL ofrece una buena combinación de rendimiento, precio y características.

4.2.1. Rendimiento

En el sitio web de MySQL, se pueden encontrar bancos de prueba (*benchmarks*) comparando MySQL y otras bases de datos. Estos bancos de prueba generalmente muestran MySQL sólidamente mejor que sus competidores. Aunque conviene ser prudente ante todos los bancos de prueba, especialmente los diseñados por los creadores de los productos, otros tests independientes indican que MySQL está entre los productos más rápidos.

Los resultados de los bancos de prueba del sitio web de MySQL pueden encontrarse en [52].

La revista eWeek realizó en el 2002 una serie de pruebas [21], repetidas en el 2003, para comparar cinco sistemas gestores de bases de datos. Los productos estudiados fueron:

- DB2 7.2 de IBM (con FixPack 5)
- SQL Server 2000 Enterprise Edition de Microsoft Corp. (con Service Pack 2)
- MySQL 4.0.1 Max de MySQL AB
- Oracle9i Enterprise Edition 9.0.1.1.1 de Oracle Corp.

- ASE (Adaptive Server Enterprise) 12.5.0.1. de Sybase Inc..

Todas las bases de datos se probaron en la misma plataforma *hardware* (servidores HP NetServer LT 6000r con cuatro CPUs 700MHz Xeon, 2GB de RAM y 24 discos duros Ultra3 SCSI 10.000-rpm 9,1GB usados para el almacenamiento de la base de datos) y en el mismo sistema operativo (Windows 2000 Advanced Server con Service Pack 2). Para generar la carga de trabajo se utilizó una aplicación web (concretamente, una librería *online*) escrita en JSP llamada Nile y el programa de pruebas e-Test Suite 6.0 de Empirix, que permite realizar y monitorizar pruebas repetitivas.

De forma global, Oracle9i y MySQL obtuvieron el mejor rendimiento y escalabilidad, con Oracle9i un poco por delante de MySQL en la mayor parte de la ejecución de las pruebas. ASE, DB2, Oracle9i y MySQL acabaron empatados con un máximo de 550 usuarios webs conectados simultáneamente. En este punto, el rendimiento de ASE se estabilizó en 500 páginas por segundo, unas 100 páginas menos que Oracle9i, y MySQL se estabilizó en unas 600 páginas por segundo. El rendimiento de DB2 cayó sustancialmente, estabilizándose alrededor de las 200 páginas por segundo bajo altas cargas.

Debido a sus significativos problemas con el driver JDBC, SQL Server estuvo limitado a 200 páginas por segundo durante toda la prueba.

Los tres factores que tuvieron mayor impacto en el rendimiento fueron los drivers, la puesta a punto de la memoria y el diseño de la base de datos. La puesta a punto tiene una gran incidencia en la productividad. En la prueba realizada se logró doblar la productividad gracias a ella.

Los drivers de Oracle y MySQL lograron la mejor combinación de estabilidad y conjunto completo de características JDBC.

SQL Server y MySQL fueron los más fáciles de afinar (optimizar el rendimiento), mientras que Oracle9i fue el más dificil debido a que tiene muchas cachés separadas que pueden ser ajustadas. Este problema es peor todavía en Oracle9i porque requiere la mayor cantidad de memoria por conexión concurrente a la base de datos (sobre 400KB de RAM). En comparación, DB2 requiere 177KB de RAM por conexión, y SQL Server, MySQL y ASE requieren alrededor de 50KB de RAM por conexión. Por tanto, las cachés de datos y consultas de Oracle9i tienen que ser menor que la de los otros SGBDs debido a la memoria utilizada para las conexiones de usuarios.

El gran rendimiento de MySQL se debió principalmente al uso de la caché en memoria para las consultas y la capacidad de usar diferentes motores de bases de datos, una característica exclusiva de MySQL.

Por último, se observó que mejorando el diseño de la base de datos, mediante la adición de índices y la ordenación de las filas de la tabla en el mejor orden físico para el conjunto de consultas a realizar, se consiguen mejoras en el rendimiento, aunque estos efectos son menores que los de los drivers y la puesta a punto de la memoria.

Aunque más adelante tocaremos el tema del precio, es importante observar que en estas pruebas uno de los productos que ha sobresalido es gratis, mientras que el otro tiene un precio

de 160.000 dólares (40.000 dólares por procesador).

La velocidad siempre ha sido uno de los principales objetivos de MySQL. Sólo se añaden nuevas características al servidor MySQL cuando éstas no perjudican al rendimiento. Algunas veces esto significa que las características se incorporan más lentamente de lo que desearían los usuarios. Sin embargo, esto asegura que MySQL siempre va a ser rápido.

4.2.2. Precio

El precio es tal vez el punto más fácil de comparar. Para muchos propósitos, MySQL es una aplicación gratuita. La licencia GPL permite usar el *software*, alterar el código fuente y redistribuir MySQL bajo GPL. No obstante, existen ciertas circunstancias, como el deseo de redistribuir MySQL como parte de un producto comercial, en las que es necesario comprar una licencia comercial. La licencia para un único servidor cuesta 220€ o 440€ (en la fecha en que este texto fue escrito), dependiendo de si quiere utilizar el tipo tabla InnoDB. En otras palabras, MySQL usa un esquema dual de licencias, donde el uso gratuito es controlado por GPL y el uso comercial lo es a través de los acuerdos estándares de la industria, como son los EULAs (*End-User License Agreements*) y los OEM (*Original Equipment Manufacturer*). La regla general de MySQL AB es «si tú eres gratuito, nosotros también lo somos; si tú eres comercial, nosotros también lo somos».

Los principales competidores son comerciales, con complejos esquemas de precios que dependen del uso pensado, el número de procesadores de cada servidor y el número de usuarios que se conectarán. Las base de datos de Oracle, Microsoft SQL Server y DB2 de IBM pueden costar decenas de miles de dólares en escenarios moderados y cientos de miles de dólares en un servidor con muchos procesadores y muchos clientes conectados.

MySQL es a veces comparado con otras bases de datos de código abierto, como PostgreSQL y Firebird. De estas bases de datos, MySQL es el único producto con una compañía detrás suyo, siendo los propietarios de los derechos de propiedad intelectual y ofreciendo completas licencias comerciales.

4.2.3. Estabilidad

Los desarrolladores de MySQL siempre han dado una importancia primordial a la estabilidad. Todas las versiones de MySQL distribuidas como binarios, incluso las versiones *alpha* (aquellas con una gran cantidad de código nuevo que no ha sido probado al cien por cien), deben pasar el MySQL Test Suite. Este proceso prueba las funciones y otras características, así como los resultados de operaciones donde se reparó un error en el pasado; por tanto, se intenta asegurar que los errores nunca sean reintroducidos accidentalmente.

Los desarrolladores también deben dar una mayor prioridad a la corrección de errores que a sus otras tareas de desarrollo. Básicamente, su otro trabajo se paraliza hasta que cualquier

error relacionado con su campo de especialización es corregido. La regla es que las versiones liberadas de MySQL deberían estar libres de todos los errores conocidos y reproducibles.

Finalmente, la calidad se asegura a través de los clientes y de la comunidad de MySQL. Con más de cuatro millones de usuarios en todo el mundo trabajando en una amplia variedad de entornos, esto proporciona la oportunidad de encontrar errores en incluso etapas tempranas de desarrollo. El sistema de notificación y tratamiento de errores en MySQL es público, de modo que cualquiera puede ver lo que otras personas han comunicado y añadir sus propios comentarios.

4.2.4. Facilidad de uso

Otra característica importante de MySQL es su facilidad de uso. No se necesita ningún complicado procedimiento de configuración para poder comenzar a usarlo. El servidor MySQL trabaja de forma adecuada por defecto, estando configurado para un uso mínimo de recursos de disco y memoria. Para un rendimiento óptimo y para requisitos específicos de producción tales como accesos al sistema (*login*), será necesaria la puesta a punto de la configuración por defecto. Como ayuda, se incluyen unos archivos de configuración de ejemplo con MySQL.

4.3. Programas clientes

En este apartado vamos a introducir brevemente tres de los programas cliente que podemos utilizar para trabajar con MySQL: `mysql`, MySQL Administrator y MySQL Query Browser. Todos ellos se distribuyen bajo la licencia GPL. El primero se incluye en la distribución del servidor MySQL.

4.3.1. La herramienta de línea de comandos: `mysql`

El programa cliente `mysql` es un simple intérprete de órdenes (*shell*) SQL. Puede usarse para conectarse o desconectarse del servidor MySQL y para ejecutar sentencias SQL.

Para conectarse al servidor es necesario introducir el nombre de usuario MySQL al invocar `mysql` y, probablemente, una contraseña. Si el servidor se ejecuta desde un equipo distinto al que se está conectando, también es necesario indicar un nombre de anfitrión (*host*):

```
> mysql -h hostname -u username -p
```

La opción `-p` indica que la cuenta de usuario posee una clave no vacía y, al no indicarla en la línea de comandos, el intérprete nos pedirá que la introduzcamos. Al pulsar *Intro* podrá ver lo siguiente:

```
Introduzca contraseña: *******
```

Los asteriscos representan la contraseña. Al introducirla, podrá ver cierta información seguida por la línea `mysql>`. Una vez realizada la conexión con éxito, se puede desconectar en cualquier momento escribiendo `QUIT` en la línea de comandos, en cuyo caso recibirá u mensaje de despedida:

```
mysql> QUIT
Adiós
```

Una vez conectados al servidor, podemos ejecutar órdenes SQL en el intérprete `mysql`. El siguiente es un sencillo ejemplo que sirve para preguntar al servidor la fecha actual:

```
mysql> SELECT CURRENT_DATE;
+--------------+
| CURRENT_DATE |
+--------------+
| 2004-05-26   |
+--------------+
```

Podemos apreciar que la instrucción SQL termina en punto y coma «;». Existen excepciones en el que este signo no se necesita, como el comando `QUIT` antes utilizado. También hemos de notar que no es obligatoria la cláusula FROM en este caso.

4.3.2. Dos herramienta gráficas: MySQL Administrator y MySQL Query Browser

MySQL Administrator[2] y MySQL Query Browser[3] son dos herramientas visuales para la gestión y administración de bases de datos MySQL, desarrolladas por MySQL AB, la misma empresa que se encarga del desarrollo de MySQL.

MySQL Administrator es una herramienta que permite realizar tareas administrativas sobre servidores de MySQL, incluyendo:

- La configuración de las opciones de inicio de los servidores.
- Inicio y detención de servidores.
- Monitorización de conexiones al servidor.
- Administración de usuarios.
- Monitorización del estado del servidor, incluyendo estadísticas de uso.

[2]Disponible en http://www.mysql.com/products/tools/administrator/
[3]Disponible en http://www.mysql.com/products/tools/query-browser/

- Visualización de los *logs* (archivos de registro) del servidor.
- Gestión de copias de seguridad y recuperaciones.

Por su parte, MySQL Query Browser es una herramienta que permite crear, ejecutar y optimizar consultas SQL en un entorno gráfico. Sus principales características son:

- Ejecuta consultas y permite editar resultados.
- Exporta conjuntos de resultados CSV (*Comma Separated Values*), HTML y XML.
- Tiene un editor SQL con coloreado de sintaxis.
- Marcadores de consultas e historia para recuperar fácilmente consultas ejecutadas.
- Compara resultados de consultas.

Ambas se distribuyen, al igual que el servidor MySQL, bajo licencia GPL, y están disponibles para plataformas Windows, Unix y Mac.

4.4. Acceso a bases de datos con Java: JDBC

En este último apartado, vamos a explicar la forma en que accederemos a nuestra base de datos MySQL desde el lenguaje de programación Java, que será el lenguaje que utilizaremos para construir nuestra aplicación web.

El API que nos permite ejecutar instrucciones SQL es distinto en cada motor de bases de datos. Sin embargo, los programadores de Java no están sujetos a las normas de portabilidad de bases de datos, ya que disponen de un solo API, la Conectividad de bases de datos de Java (*Java Database Connectivity*, JDBC)[4], que se puede transportar de un motor a otro.

La biblioteca JDBC dispone de una interfaz para ejecutar instrucciones SQL que ofrece la funcionalidad básica para el acceso de datos. Las clases e interfaces que componen el API JDBC son conceptos comunes al acceso a bases de datos para todas las bases de datos. Facilita el envío de instrucciones SQL a sistemas de bases de datos relacionales y es compatible con todos los dialectos de SQL. Sin embargo, el API JDBC 2.0 supera los límites del lenguaje SQL ya que ofrece la posibilidad de interactuar con otros tipos de orígenes de datos, como archivos que contengan tablas de datos. En otras palabras, con la biblioteca JDBC no es necesario escribir un programa para acceder a la base de datos MySQL, otro para acceder a una base de datos Oracle, etc. Basta con escribir un programa Java que utilice la biblioteca JDBC para que pueda enviar SQL u otras instrucciones a la fuente de datos adecuada. Evidentemente, se trata de conservar el enfoque de Java «*escriba una vez, utilícelo siempre*».

[4]Disponible en http://java.sun.com/products/jdbc

4.4.1. Controladores

Aunque se puede utilizar el API JDBC para acceder a prácticamente cualquier origen de datos desde una aplicación Java, existe una pequeña salvedad: hay que tener acceso al controlador específico de la base de datos. Un controlador es la implementación de la interfaz JDBC adecuada a una base de datos concreta. Se trata de un nivel intermedio que traduce el método Java en llamadas API a una base de datos propietaria que sirven para manipular la base de datos. Los detalles exactos relativos a la forma en que el controlador activa la comunicación entre la aplicación Java y el API de la base de datos depende del controlador en sí. Con un controlador JDBC se pueden conseguir tres cosas:

1. Establecer una conexión con una fuente de datos.

2. Enviar sentencias SQL a las fuentes de datos.

3. Recuperar resultados.

Existen diversos controladores para MySQL y la información relativa a dichos controladores se puede encontrar en la página de descargas de MySQL[5], dentro de la sección JDBC. En nuestro caso, vamos a utilizar el controlador MM.MySQL, sujeto a la licencia GPL. Permite a los diseñadores de Java realizar conexiones a los servidores MySQL desde aplicaciones Java y *applets*. Es aconsejable descargar el binario de este controlador y ubicarlo en el sitio correcto (como puede ser la carpeta Tomcat `common\lib`) y asegurarse de que la variable `CLASSPATH` apunta a dicha carpeta. En el sitio web de Sun, se puede encontrar la lista completa de controladores disponibles[6].

4.4.2. Cómo acceder a una base de datos utilizando JDBC

Los pasos necesarios para utilizar JDBC y acceder a una base de datos son los siguientes:

1. Cargue y registre el controlador JDBC adecuado con ayuda de DriverManager (Administrador de controladores).

2. Determine el vínculo entre el controlador y la fuente de datos por medio de una URL JDBC.

3. Cree un objeto de conexión con la URL JDBC.

4. Inserte una instrucción SQL en un método de ejecución para ejecutar la instrucción.

[5]http://www.mysql.com/downloads
[6]Lisa de controladores JDBC http://industry.java.sun.com/products/jdbc/drivers

4.4.3. Cómo cargar el controlador JDBC

La clase `DriverManager` es la capa de administración tradicional de JDBC que actúa entre el usuario y los controladores. Realiza el seguimiento de los controladores que se encuentran disponibles en el sistema así como la conexión a la base de datos a través del controlador adecuado.

El primer paso para utilizar un controlador y conectarse a la base de datos es, evidentemente, cargar el controlador correspondiente. Para ello, es necesario añadir la clase controlador a la propiedad del sistema `jdbc.drivers`. Seguidamente, como la clase `DriverManager` se ha iniciado, busca el valor de dicha propiedad del sistema y, si encuentra los nombres de uno o más controladores, trata de cargarlos.

Para añadir los nombres de controladores a la propiedad del sistema es necesario utilizar el método `System.setProperty()`, como se puede comprobar a continuación:

```
System.setProperty(\jdbc.drivers","org.gjt.mm.mysql.Driver");
```

Hemos añadido el nombre de clase del controlador `MM.MySQL` a la propiedad de sistema `jdbc.drivers`. Se puede añadir más de un controlador a la propiedad; basta con separar los nombres de los controladores por medio de comas. Sin embargo, puede que no tenga permiso para modificar la propiedad de sistema de esta forma, debido a la política de seguridad de su sistema. Si éste es el caso, tiene que emplear una técnica más directa y cargar el controlador de forma explícita utilizando el método `Class.forName()`:

```
Class.forName(\org.gjt.mm.mysql.Drivers");
```

Al cargar una clase de controlador, ésta crea un objeto controlador y registra dicha instancia con la clase `DriverManager` invocando, automáticamente, el método `DriverManager .registerDriver()`. No es necesario invocar directamente este método para registrar una clase controlador.

4.4.4. URL de JDBC

El siguiente paso hacia la conexión a la base de datos es determinar una dirección URL del controlador JDBC que vincule un controlador con una fuente de datos.

La primera parte de una dirección URL indica el protocolo utilizado para acceder a la información, seguido siempre por dos puntos. El resto de la dirección ofrece información sobre la ubicación del origen de datos. La sintaxis estándar de una URL de JDBC es la siguiente:

```
jdbc:<subprotocol>:<data source identifier>
```

Las tres partes de esta URL de JDBC son:

- `jdbc`: el protocolo.

- `<subprotocol>`: el nombre del controlador.
- `<data source identifier>`: el nombre de la fuente de datos.

Un ejemplo de URL es la siguiente:

```
jdbc:mysql://localhost:8080/Proyecto?user=pepe&password=pepe
```

4.4.5. Cómo establecer la conexión

Una vez cargado el controlador y definida la URL para vincularlo con una fuente de datos, se puede establecer la conexión con ayuda del método `DriverManager.getConnection()`, como se aprecia :

```
String url =
"jdbc:mysql://localhost:8080/Proyecto?user=pepe&password=pepe";
Connection con = DriverManager.getConnection(url);
```

Hemos creado una cadena (`string`) URL de JDBC y la hemos insertado en el método `getConnetion()`.`DriverManager` prueba cada uno de los controladores registrados para ver si se puede establecer la conexión. Puede darse el caso de que haya más de un controlador JDBC para realizar la conexión con una determinada URL. El método `DriverManager` trata de usar cada uno de los controladores en el orden en que fueron registrados.

El método devuelve una conexión o, para ser más específicos, devuelve un objeto que implementa la interfaz `java.sql.Connection`. El objeto conexión permite crear y ejecutar comandos SQL.

4.4.6. Creación de instrucciones

Una vez establecida la conexión a una determinada base de datos, se puede utilizar para enviar instrucciones SQL. Para ello, es necesario crear un objeto instrucción que nos permita diseñar un comando SQL y ejecutarlo.

El API JDBC proporciona tres interfaces para enviar sentencias SQL a la base de datos, además de métodos en la interfaz `Connection` que crean instancias de ellos. Las interfaces para enviar instrucciones SQL y los métodos de `Connection` que permiten crear instancias de dichos interfaces son los siguientes:

1. Interfaz `java.sql.Statement`. Objetos creados por el método `Connection.createStatement()`. Los objetos `Statement` son usados para enviar sentencias SQL sin parámetros.

Ejemplo 4.1 Uso de un objeto `Statement`.

Una vez que se establece una conexión a una base de datos particular, esta conexión puede ser usada para enviar sentencias SQL. Un objeto `Statement` es creado con el método `createStatement()` del objeto `Connection`, como en el siguiente fragmento de código:

```
Connection con = DriverManager.getConnection(url,
   "sunny", "");
Statement stmt = con.createStatement();
```

La sentencia SQL que será enviada a la base de datos es proporcionada como el argumento de uno de los métodos de ejecución (Apartado 4.4.7) de un objeto `Statement`. Esto lo podemos ver a continuación:

```
ResultSet rs = stmt.executeQuery("SELECT a, b, c
                                  FROM Table2");
```

La variable `rs` contiene un objeto `ResultSet`, que contiene el resultado de ejecutar la consulta SQL, es decir, contiene las filas que satisfacen las condiciones de la consulta.

□

2. Interfaz `java.sql.PreparedStatement`. Objetos creados por el método `Connection.prepareStatement()`. Esta interfaz es una extensión de la anterior. Los objetos `PreparedStatement` contienen sentencias SQL precompiladas. Estas sentencias pueden tener uno o varios parámetros de entrada (IN) para que la aplicación pueda, de forma dinámica, asignar diferentes valores. Un parámetro IN es aquel cuyo valor no es especificado cuando la sentencia SQL es creada. En cambio, la sentencia tiene un símbolo de cierre de interrogación «?» en el lugar de cada parámetro IN. Cualquier aplicación tiene que dar valores a estos marcadores de parámetros de entrada antes de ejecutar la instrucción.

   Un objeto `PreparedStatement` tiene el potencial de ser más eficiente que un objeto `Statement` ya que ha sido precompilado y almacenado para su uso futuro. Por tanto, suelen ser usados cuando es necesario ejecutar una determinada instrucción SQL en repetidas ocasiones.

   Ejemplo 4.2 Uso de un objeto `PreparedStatement`.

   El siguiente fragmento de código, donde `con` es un objeto `Connection`, crea un objeto `PreparedStatement` que contiene una sentencia `UPDATE` con dos marcadores (indicados con el símbolo «?») para parámetros de entrada (IN):

```
PreparedStatement pstmt = con.prepareStatement(
  "UPDATE table4 SET m = ? WHERE x = ?");
```

EL objeto `pstmt` contiene ahora la sentencia «`UPDATE table4 SET m = ? WHERE x = ?`», que ya ha sido enviada al SGBD y preparada para la ejecución.

EL siguiente código ilustra cómo establecer valores para los dos parámetros de entrada y ejecutar `pstmt` 10 veces (para una descripción del método `executeUpdate()`, Apartado 4.4.7). En este ejemplo, al primer parámetro se le da el valor «Hi» y permanece constante. Al segundo parámetro se le da un valor diferente en cada iteración, comenzando con 0 y terminando con 9.

```
pstmt.setString(1, "Hi");
for (int i = 0; i < 10; i++) {
  pstmt.setInt(2, i);
  int rowCount = pstmt.executeUpdate();
}
```

□

3. Interfaz `java.sql.CallableStatement`. Objetos creados por el método `Connection.prepareCall()`. Esta interfaz es una extensión de la anterior. Los objetos `CallableStatement` son usados para ejecutar procedimientos almacenados SQL (un grupo de sentencias en un lenguaje de programación, el cual debe disponer de mecanismos para ejecutar sentencias SQL, que se almacenan en la base de datos y se ejecutan internamente). Estos procedimientos son almacenados en la base de datos. Un objeto `CallableStatement` contiene la llamada a uno de estos procedimientos. Esta llamada es escrita en una sintaxis que puede tomar dos formas: una con un parámetro de resultado y otra sin él. Un parámetro de resultado, un tipo de parámetro de salida, es el valor devuelto por el procedimiento almacenado. Ambas formas pueden tener parámetros de entrada (IN), de salida (OUT) o de entrada/salida (INOUT). Un símbolo de cierre de interrogación «?» sirve como marcador de cada uno de estos parámetros.

Ejemplo 4.3 Uso de un objeto `CallableStatement`.

Los objetos `CallableStatement` son creados con el método `prepareCall` del objeto `Connection`. El siguiente código, donde `con` es un objeto `Connection` activo, crea una instancia de `CallableStatement`.

```
CallableStatement cstmt = con.prepareCall(
  "{call getTestData(?, ?)}");
```

La variable `cstmt` contiene una llamada la procedimiento almacenado `getTestData`, el cuál tiene dos parámetros de entrada y ningún parámetro resultado. Que los marcadores «?» sean parámetros IN, OUT o INOUT depende del procedimiento almacenado `getTestData`. □

4.4.7. Ejecución de instrucciones

Después de crear un objeto `Statement`, se puede ejecutar un comando SQL invocando distintos métodos de ejecución sobre el objeto. El método adecuado que se debe utilizar viene determinado por el resultado que se quiere obtener tras ejecutar la instrucción SQL, que se envía a la base de datos como argumento del método de ejecución.

La interfaz `java.sql.Statement` ofrece tres métodos distintos para ejecutar instrucciones SQL:

- `executeUpdate()`
  Este método permite ejecutar instrucciones `INSERT`, `UPDATE` o `DELETE`, o instrucciones SQL DDL como `CREATE TABLE`, `DROP TABLE` y `ALTER TABLE`. El valor que devuelve es un entero (el número de actualización) que indica el número de filas que se han visto afectadas. En instruccinoes como `CREATE TABLE` o `DROP TABLE`, que no operan en filas, el valor devuelto del método `executeUpdate()` siempre es cero.

- `executeQuery()`
  Este método está diseñado para instrucciones que devuelven un solo conjunto de resultados, como instrucciones `SELECT`.

- `execute()`
  Este método nos permite ejecutar instrucciones que devuelven más de un conjunto de resultados, más de un número de actualización o una combinación de ambos.

Las interfaces `PreparedStatement` y `CallableStatement` extienden la interfaz `java.sql.Statement`, por lo que heredan los métodos de ejecución anteriores.

Capítulo 5

Struts

Este capítulo es una introducción a Struts, el *framework* que hemos utilizado para construir nuestra aplicación web. Se supone que el lector está familiarizado con la programación orientada a objetos y conoce el lenguaje de programación Java.

Está compuesto por los siguientes apartados:

- Introducción. Se explica qué es Struts y qué características posee [29].
- Tecnologías necesarias. Presentamos una breve introducción a las tecnologías utilizadas por Struts [29, 39, 64].
- Modelo 1 frente a Modelo 2. Este apartado muestra las diferencias entre los dos tipos de arquitecturas para aplicaciones web Java [39].
- El patrón MVC. Se explica la estructura y funcionamiento del patrón de diseño MVC (*Model-View-Controller*, Modelo-Vista-Controlador), y cómo implementa Struts dicho patrón [64].
- Componentes de Struts. En último lugar, introduciremos los distintos componentes de Struts, describiendo brevemente las funciones que realizan [15, 59, 61, 64].

5.1. Introducción

Struts es un *framework*[1] de código abierto usado para construir aplicaciones web en Java, basado en el patrón de diseño MVC y diseñado utilizando principalmente las tecnologías JavaServer Pages (JSP) y Servlets. Nos permite separar la lógica de negocio, la lógica de control y el código de presentación de una aplicación, lo que permite mejorar su posterior utilización y su mantenimiento. Además, Struts nos proporciona una estructura básica sobre la cual construir

[1] Un *framework* es la extensión de un lenguaje mediante una o más jerarquías de clases que implementan una funcionalidad y que (opcionalmente) pueden ser extendidas.

nuestra aplicación, facilitando la implementación de funcionalidades como el manejo de excepciones, la internacionalización, etc. Según Ford [30], usar Struts ahorra al desarrollador escribir entre un 30 y un 40 por ciento del código que normalmente se requiere para una aplicación web típica.

El *framework* Struts forma parte del proyecto Jakarta[2], gestionado por la Fundación Apache Software[3].

Struts ofrece las siguientes funcionalidades:

- Un servlet que actúa como controlador del paradigma MVC.
- Bibliotecas de etiquetas JSP para la administración de *beans*, generación de HTML y JavaScript, manejo de plantillas, control de flujo en JSP y extensión de las etiquetas Struts para permitir anidamiento.
- Una estructura para internacionalizar los mensajes, lo que significa que cualquier mensaje que aparezca al utilizar la aplicación se encuentre escrito en el idioma del usuario.
- Implementación de JDBC (*Java Database Connectivity, Conectividad de Bases de Datos de Java* para definir las fuentes de datos y un *pool* de conexiones a bases de datos.
- Un mecanismo general de resolución de errores y excepciones.
- Análisis de sintaxis (*parsing*) XML.
- Utilidades para carga de archivos.
- Utilidades de conexión.

Aparte de Struts, existen otros *frameworks* para construir aplicaciones web en Java que implementan el patrón MVC: Spring Framework, WebWork, Tapestry, JSF, etc. Nos hemos decantado por Struts por las siguientes razones:

- Es útil, potente y no demasiado complejo.
- Es el *framework* más utilizado.
- Tiene una bibliografía bastante extensa, a diferencia de muchos otros *frameworks*.
- Ofrece características que se adecúan a nuestras necesidades:
  - Estructuración de las aplicaciones gracias al patrón de diseño MVC (Apartado 5.4).
  - Soporte para la internacionalización (Apartado 5.5.4.3)

[2]Sitio web del proyecto Jakarta: http://jakarta.apache.org/
[3]Sitio web de la Fundación Apache Software: http://www.apache.org/

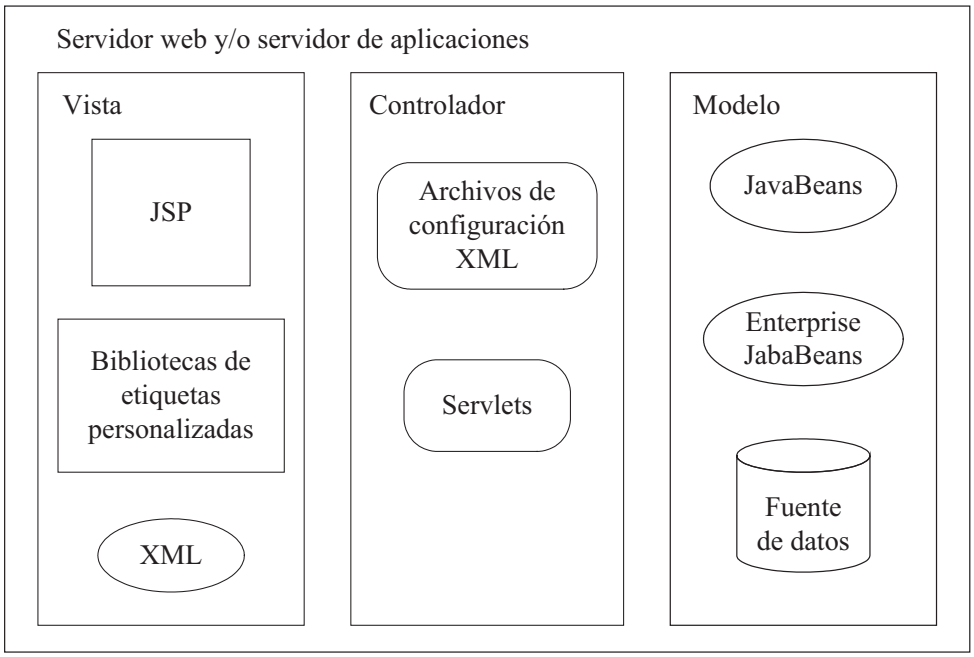

Figura 5.1: Tecnologías utilizadas por Struts

- Posee una bibliotecas de etiquetas que nos facilita el trabajo con los formularios HTML y nos permite usar plantillas para nuestra páginas web.

- Su bajo coste (es gratuito).

Para este proyecto se ha utilizado la versión 1.2.7 de Struts.

5.2. Tecnologías necesarias

Hoy día es muy normal en el campo del desarrollo de aplicaciones web tener que trabajar con muchas tecnologías interrelacionadas. Frecuentemente es necesario tener un conocimiento bastante profundo de un importante número de tecnologías para poder llevar a cabo un proyecto.

Las principales tecnologías utilizadas por Struts son los Servlets y JSP. Sin embargo, otras que también es preciso conocer son las bibliotecas de etiquetas JSP, los JavaBeans y XML.

En este apartado vamos a describir brevemente cada una de estas tecnologías, por lo que remitimos al lector que esté interesado en profundizar en estas tecnologías a los recursos bibliográficos. La Figura 5.1 muestra cómo estas tecnologías se relacionan con cada uno de los componentes de Struts: el modelo, la vista y el controlador (Apartado 5.5).

5.2.1. Servlets

Un servlet es un componente o programa que genera contenido dinámico. Es como un pequeño servidor web: recibe una petición y devuelve una respuesta. Un servlet es simplemente una clase Java que ha sido compilada a bytecode, como cualquier otro objeto Java. Puesto que

están basados en Java, los servlets son una tecnología muy portable que puede utilizarse en cualquier sistema operativo que tenga una Máquina Virtual de Java (MVJ, o JVM por sus siglas en inglés). Los servlets se definen por medio del API Java Servlet.

Para que los servidores web convencionales puedan tener acceso a los servlets, éstos deben ejecutarse dentro de un servidor de aplicaciones compatible con Java, también llamado contenedor, como puede ser Tomcat. El contenedor de servlets debe configurarse para que coopere con un servidor web. Se puede declarar, para cada servlet, qué direcciones URL serán las que deba procesar. Cuando se recibe una petición que esté asociada a algún servlet en particular, el servidor web pasa la petición al contenedor, y el conteneder invoca al servlet encargado de procesar dicha petición.

La tecnología Servlet es mencionada ya que el controlador de la arquitectura Struts es un servlet. Para más información, puede consultar [68].

5.2.2. JSP

JSP (*JavaServer Pages*, Páginas de Servidor Java) es una tecnología usada para desarrollar aplicaciones web con contenido dinámico. Una página JSP es un componente del lado del servidor formado por código HTML o XML, etiquetas específicas de JSP, y fragmentos de código Java llamados scriptlets. JSP es usado para la capa de presentación en las arquitecturas web organizadas por niveles con el fin de separar la presentación del contenido. En el *framework* Struts, las páginas JSP representan la Vista en el patrón de diseño MVC (*Model-View-Controller*, Modelo-Vista-Controlador) sobre el que hablaremos en el Apartado 5.4.

Cuando se solicita por primera vez una JSP, ésta es traducida a un archivo Java y entonces es compilada en una clase Servlet. De esta manera, la JSP se convierte en un servlet.

Para más información sobre JSP visite [67] o [41] para un tutorial.

Ejemplo 5.1 Creación de una página JSP. Vamos a crear una página JSP sencilla. Para ello debemos introducir las siguientes líneas de código en un archivo que debe tener extensión .jsp, para así indicar al servidor web que se trata de una página JSP:

```
<html>
 <head>
   <title>Mi primera JSP</title>
 </head>
 <body>
   ¡Hola mundo!<br />
   La fecha y hora actual es <%= new java.util.Date() %>
 </body>
</html>
```

Si observamos el código, se trata de una página HTML que contiene código Java. JSP introduce una etiqueta especial, `<%= ... %>`, denominada expresión scriptlet, para distinguirse de HTML. Dentro de esas etiquetas hay escrito código Java. Las palabras `new java.util.Date()` le dicen a Java que cree un nuevo objeto `Date` y los símbolos `<%= ... %>` le dicen a Java que imprima los contenidos de ese objeto (la fecha de hoy) en HTML.

El siguiente paso sería guardar el archivo dentro del directorio raíz de un servidor web con soporte para JSP, como puede ser Tomcat. En la Figura 5.2 se muestra la pantalla del navegador Internet Explorer visualizando nuestra página de ejemplo. Podemos observar que ya no aparecen las palabras `new java.util.Date()`, sino la fecha y hora actuales. Esto se debe a que el contenedor JSP ha ejecutado todo el código Java de la página y lo ha reemplazado por el resultado de la ejecución.

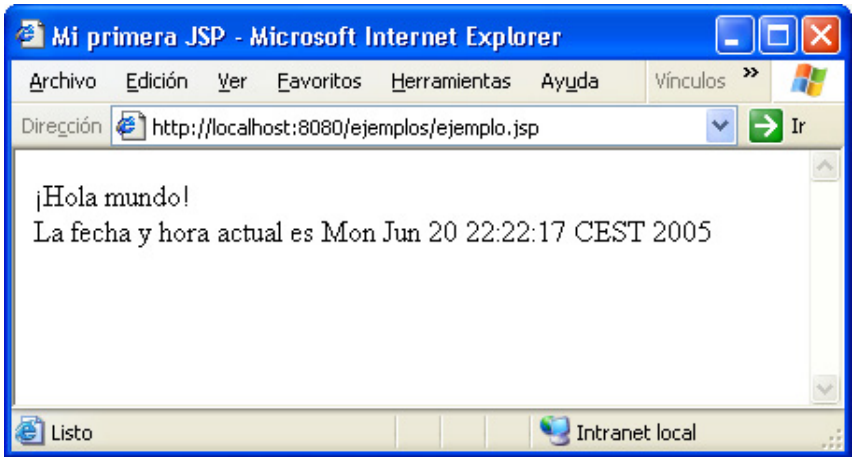

Figura 5.2: Pantalla del Internet Explorer visualizando la página JSP de ejemplo

□

5.2.3. JavaBeans

Un JavaBean o bean es simplemente una clase que mantiene algunos datos (las propiedades) y sigue unas determinadas convenciones de codificación. Estas convenciones proporcionan un mecanismo para el soporte automatizado. Este soporte automatizado significa que el motor de JSP, por ejemplo Tomcat, puede inspeccionar el bean y descubrir las propiedades que tiene.

Las propiedades de un JavaBean se exponen públicamente utilizando los llamados métodos getter y setter. Estos métodos siguen convenciones de denominación simples, que entenderemos más fácilmente con un ejemplo. Si tenemos una propiedad denominada `color`, los métodos getter y setter se llamarían `getColor()` y `setColor()`. Los nombres del método son sólo el nombre de la propiedad comenzando por letra mayúscula y precedidos por get o set. Para construir un JavaBean todo lo que tenemos que hacer es escribir una clase Java que siga estas reglas.

Ejemplo 5.2 Construcción de un JavaBean. Definamos una clase JavaBean llamada `CocheBean` que podríamos utilizar como componente de un sitio web de venta de coches. Será un componente que modelará un coche y tendrá una propiedad: la marca del coche.

```
public class CocheBean {

  public CocheBean() {
  }

  private String marca = "Ford";

  public String getMarca() {
    return marca;
  }

  public void setMarca(String marca) {
    this.marca = marca;
  }
}
```

Hemos definido una propiedad `String` llamada `marca`, que por defecto toma el valor `"Ford"`. Si nos movemos por el código podemos ver que esta propiedad tiene dos métodos getter y setter. El método `getMarca()` devuelve el valor de la marca. El método `setMarca()` establece el valor de la propiedad `marca` como igual al argumento.

□

Inicialmete Sun[4] creó los JavaBeans para la construcción de componentes GUI (*Graphical User Interface*). Hoy día su uso se ha extendido a todos los campos del desarrollo con Java, incluido el de las aplicaciones web. Así podemos mover los datos y la funcionalidad presentes en los scriptlets de una página JSP a componentes beans. De esta manera aumentamos la reutilización del código y facilitamos el mantenimiento de nuestra aplicación.

Para poder utilizar beans en las páginas JSP debemos conocer el concepto de «ámbito». Cada bean tiene un ámbito. Existen cuatro ámbitos distintos: `application`, `session`, `request` y `page`. Por ejemplo, se puede asociar un bean con los datos comunes a todos los usuarios (ámbito `application`), un bean con información de usuario a la sesión (ámbito `session`), beans para pasar datos entre JSPs que tratan una petición (ámbito `request`) o un bean para guardar una variable que será usada dentro de una única página (ámbito `page`).

Antes de acceder a un bean desde una página JSP, debemos identificar y obtener una referencia al mismo. La etiqueta `<jsp:useBean>` intenta obtener una referencia a una instancia

[4] http://www.sun.com

de un bean ya existente. Lo hace usando el identificador y el ámbito del bean. Si el Bean ha sido creado con anterioridad en el ámbito especificado, se usará dicha instancia. En cambio, si no se encuentra la instancia, se creará una nueva. Una etiqueta que hiciera eso podría ser la siguiente:

```
<jsp:useBean id="usuario" class="proyecto.usuario"
             scope="session" />
```

Struts hace uso de los beans, por eso es importante al menos estar familiarizado con la terminología. La especificación de los JavaBeans puede encontrarse en [69].

5.2.4. Bibliotecas de etiquetas personalizadas JSP

Las bibliotecas de etiquetas personalizadas (*Custom Tag Libraries* o *Taglibs*) son una característica disponible a partir de la especificación JSP v1.1. Cada biblioteca contiene varias etiquetas JSP que son usadas de la misma forma que las etiquetas HTML. Una única etiqueta JSP puede representar a un gran número de sentencias Java, pero lo único que necesita conocer el programador es cómo insertar la etiqueta. El código asociado a la etiqueta está «oculto» en un archivo de clase Java.

Si un desarrollador desea usar el mismo código en otra página, sólo tiene que insertar la etiqueta dentro de esa página. Si el código asociado a la etiqueta cambia, todas las etiquetas usarán automáticamente la versión actualizada. Las páginas JSP que usen la etiqueta no necesitan ser revisadas.

Aunque nosotros podemos crear nuestras propias bibliotecas de etiquetas, existen varias bibliotecas de etiquetas JSP creadas por otros desarrolladores, como puede ser la biblioteca JSTL (*JSP Standard Tag Library*, Biblioteca de Etiquetas Estándar para JSP). Struts 1.2 posee cinco potentes bibliotecas de etiquetas: Struts-html, Struts-bean, Struts-logic, Struts-nested y Struts-tiles. Un buen sitio para comenzar a conocer las bibliotecas de etiquetas es [71].

Ejemplo 5.3 Uso de algunas etiquetas de la biblioteca Struts-logic. La biblioteca Struts-logic contiene etiquetas que permiten controlar el flujo de ejecución dentro de una página JSP, como, por ejemplo, la iteración o la evaluación condicional del cuerpo de una etiqueta.

Veamos algunos ejemplos que ilustran el uso de varias etiquetas de esta librería. El primero utiliza la etiqueta `<logic:equal>`.

```
<logic:equal name="coche" property="marca" value="Ford">
  El valor de coche.marca es Ford
</logic>
```

Esta etiqueta evalúa el cuerpo de la misma en caso de que haya un bean denominado `coche` que tenga una propiedad llamada `marca` cuyo valor sea igual a `Ford`. En el siguiente ejemplo, se emplea la etiqueta `<logic:greatherThan>`:

```
<logic:greatherThan name="coche" property="precio" value="6000">
  El valor de coche.precio es mayor que 6000
</logic:greatherThan>
```

Esta etiqueta evalúa el cuerpo en caso de que haya un bean denominado `coche`, que disponga de una propiedad `precio`, pero sólamente si el valor de la propiedad es mayor que 6000.

Otra etiqueta de esta biblioteca es `<logic:present>`. Como se puede deducir de su nombre, esta etiqueta comprueba si un determinado objeto está presente antes de evaluar el cuerpo de la etiqueta. En el siguiente ejemplo podemos ver la aplicación de esta etiqueta:

```
<logic:present name="coche" property="modelo">
  La propiedad coche.modelo está presente
</logic:greatherThan>
```

En este caso, la etiqueta evalúa el cuerpo si existe un bean denominado `coche` que disponga de una propiedad llamada `modelo`.

□

5.2.5. XML

Con el tiempo, el lenguaje XML (*eXtensible Markup Language*) se ha convertido en una tecnología omnipresente en el desarrollo de aplicaciones y servicios web. Hoy día es difícil construir una aplicación sin tener que usar XML. Este lenguaje es usado para representar documentos estructurados y datos en la web. Consiste en un conjunto de elementos (o etiquetas). El documento XML debe estar bien formado. Se dice que un documento está bien formado cuando todos sus elementos tiene etiquetas de apertura y cierre y están anidadas correctamente. Por ejemplo:

```
<asociacion>
  <nombre>Asociación de Técnicos de Informática</nombre>
  <siglas>ATI</siglas>
</asociacion>
```

representa tres etiquetas XML, asociacion, nombre y siglas. Es posible validar documentos XML con un DTD (*Document Type Declaration*, Definición de Tipo de Documento). Un DTD es un fichero lógico que contiene la definición formal de un tipo de documento XML en particular. En éste se describen los nombres de los elementos, dónde pueden aparecer y la interrelación entre ellos, en otras palabras, proporciona la gramática para una clase de documentos XML. Hoy día, los archivos de configuración de las aplicaciones suelen ser archivos XML. Struts no es una excepción y tiene dos archivos XML: `web.xml`, para configurar la aplicación web,

y `struts-config.xml`, para configurar las acciones que pueden ser llevadas a cabo por nuestra aplicación Struts.

Puede obtener más información sobre XML en [75] y [58].

5.3. Modelo 1 frente a Modelo 2

Las páginas JSP fueron creadas con la idea de facilitar la creación de páginas web dinámicas, como una alternativa a los servlets. De esta manera, los desarrolladores podían mezclar fácilmente HTML con código Java gracias a la tecnología JSP y tener todas las ventajas de los servlets.

Las aplicaciones web Java se convirtieron rápidamente en centradas en la página, puesto que las páginas JSP eran las responsables tanto de la lógica de control (recibir peticiones, redireccionar a otra JSP) como de la lógica de presentación (interactuar con el usuario). Toda esta lógica se codifica en las páginas JSP como scriptlets. En aplicaciones más grandes, esta combinación de lógica en páginas y la necesidad de asociar páginas empezó a causar problemas: fuerte dependencia entre las páginas, imposibilidad de reutilizar los scriptlets (al menos que cortemos/peguemos), etc. Claramente se necesitaba otro modelo.

Muchos desarrolladores se dieron cuenta de que las páginas JSP y los servlets se podrían usar juntos para construir aplicaciones web. Los servlets podrían encargarse de controlar el flujo de ejecución, y las JSP podrían encargarse de escribir el código HTML. Usar JSP y servlets juntos se ha dado ha conocer como el Modelo 2 (cuando usar sólo páginas JSP era el Modelo 1 o modelo centrado en la página). El Modelo 2 sigue el conocido patrón de diseño MVC (*Model-View-Controller*, Modelo-Vista-Controlador). Este modelo separa el código de presentación del de la lógica de control, resolviendo el problema que teníamos con el Modelo 1.

5.4. El patrón MVC

El `framework` Struts se basa en varios patrones de diseño[5]. Esto es una buena noticia puesto que el hecho de utilizar patrones facilita el trabajo. Los patrones de diseño no sólo proporcionan diseños más claros y probados, sino que acelera la curva de aprendizaje de los desarrolladores. Una vez que estamos familiarizados con un patrón, es fácil aplicar sus conceptos en otros proyectos. La estructura general del `framework` Struts está basada en el patrón MVC, también llamado Modelo 2 o MVC2.

El patrón MVC divide una aplicación en tres partes: el modelo, la vista y el controlador. Este patrón fue creado inicialmente para el diseño de interfaces gráficas de usuario (GUI), pero con el tiempo se ha acabado aplicando en las aplicaciones web. La ventaja de usar este patrón es que separa la lógica de negocio (modelo) de la presentación (vista). Esto posibilita una mayor

[5]Un patrón de diseño es, en resumen, una solución común aceptada como correcta a un problema de diseño concreto. Para más información, consultar [50].

reutilización de componentes y minimiza el acoplamiento entre capas o partes de la aplicación, de forma que podemos cambiar la implementación de una capa afectando mínimamente al resto de capas. Este es uno de los principales beneficios de Struts. A continuación vamos a ver la función de cada componente y cómo se relacionan entre sí. En la Figura 5.3 podemos ver el diagrama de componentes de la arquitectura MVC.

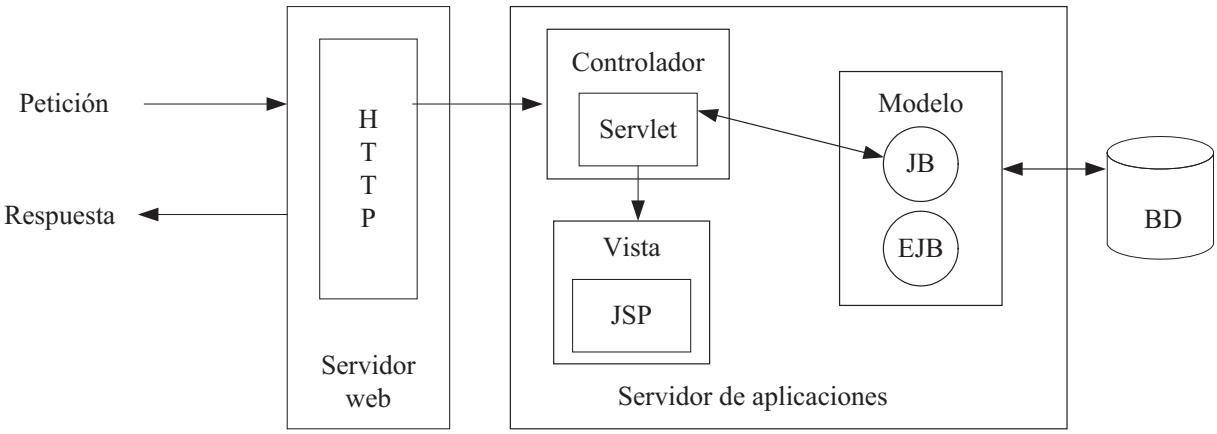

Figura 5.3: Diagrama de componentes de la arquitectura MVC

En la arquitectura MVC, el flujo de la aplicación está dirigido por un Controlador central. El Controlador recibe peticiones, en nuestro caso, peticiones HTTP, y las envía al manejador apropiado, el cuál se encarga de procesar la petición. En Struts, los manejadores también son llamados acciones (*actions*). Los manejadores están relacionados con el Modelo, de forma que cada manejador actúa como un adaptador o puente entre la petición y el Modelo. Los manejadores o acciones suelen delegar en uno o más JavaBeans o EJBs[6] la ejecución de la lógica de negocio. Así, cuando un manejador recibe una petición, recupera los datos necesarios de la petición y se los pasa a los JavaBeans o EJBs que se encargarán de realizar la lógica de negocio asociada al manejador.

El Modelo representa, o encapsula, el estado o lógica de negocio de la aplicación, en otras palabras, contiene los datos y la funcionalidad de la aplicación. Una vez que el manejador ha terminado de procesar la petición, normalmente devuelve el control al Controlador. Entonces el Controlador envía la petición a la Vista, que muestra la información al usuario de una cierta forma. En Struts, la Vista está formada por el conjunto de páginas JSP que contienen la lógica de presentación.

El Controlador sabe a qué manejador o Vista enviar cada petición gracias a un conjunto de asignaciones (*mappings*) que obtiene a partir de una base de datos o de un archivo de configuración. Esto proporciona un mínimo acoplamiento entre la Vista y el Modelo, haciendo a la aplicación más fácil de crear y mantener. En Struts, el Controlador obtiene dichas asignaciones

[6]Los EJBs (*Enterprise JavaBeans*) proporcionan un modelo de componentes distribuido estándar para el lado del servidor. No hay que confundir a los Enterprise JavaBeans con los JavaBeans. Los JavaBeans también son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones, pero no pueden utilizarse en entornos de objetos distribuidos.

del archivo de configuración `struts-config.xml`.

5.4.1. Implementación MVC de Struts

Ahora vamos a ver cómo implementa Struts el patrón de diseño MVC. Veamos primero la manera en que Struts implementa cada componente de la arquitectura MVC:

- El Modelo. Está formado por las clases Action[7], que actúan como manejadores o acciones. Estas clases proporcionan la lógica de negocio. Normalmente, estas clases harán uso de JavaBeans o EJBs para hacer el trabajo real. El Controlador sabe a qué Action debe enviar cada petición gracias al archivo `struts-config.xml`.

- La Vista. Está formada por páginas JSP y bibliotecas de etiquetas JSP. La Vista no contiene ni lógica de control, ni lógica de negocio, ni información sobre el modelo. Aunque no es obligatorio, en la mayoría de las situaciones suele ser beneficioso usar las bibliotecas de etiquetas proporcionadas por Struts. La clase `ActionForm` (la veremos más adelante, aunque, por ahora, basta decir que es utilizada para que las JSPs puedan pasar información al Modelo) y las bibliotecas de etiquetas ayudan a construir más rápidamente los formularios HTML de nuestras aplicaciones. Además, proporcionan un mecanismo para la internacionalización que está basado en el uso de archivos de recursos.

- El Controlador. Es un servlet, denominado `ActionServlet`, que implementa el patrón de diseño *Command Design*. Este servlet se encarga de recibir peticiones y enviárselas a las clases Action apropiadas. La forma de decirle al contenedor web que cree una instancia del `ActionServlet` es a través del archivo `web.xml`. Ese mismo archivo sirve para configurar el servlet. Las asignaciones petición-acción son definidas en el archivo `struts-config.xml`.

Antes de comenzar a verlos con más detalle, veamos cómo se interrelacionan entre sí estos componentes. La Figura 5.4 muestra todo el proceso que se desencadena en una aplicación Struts cuando se recibe una petición. Comentemos cada uno de los pasos de este proceso:

1. El Controlador recibe una petición por parte de un navegador web.

2. El Controlador envía la petición a la clase Action adecuada.

3. La clase Action hace uso de JavaBeans y/o EJBs para llevar a cabo la lógica de negocio. Puesto que ayuda a controlar el flujo de la aplicación sin ejecutar directamente código de la lógica de negocio, la clase Action también puede ser considerada como parte del Controlador en vez del Modelo.

[7]Existen dos posturas enfrentadas acerca de si la clase `Action` es parte del Controlador o del Modelo. En este texto se ha seguido el segundo enfoque.

4. Una vez que la clase Action ha terminado de procesar la petición, devuelve el control al Controlador. Además, le dice al Controlador a qué página JSP, u otro tipo de recurso, debe redirigir la petición.

5. El Controlador redirige la petición a la página JSP indicada por la clase Action.

6. La página JSP consulta los datos del Modelo que necesita para generar la respuesta, es decir, se ejecuta el código Java incluido en la JSP.

7. La página JSP ya ejecutada es enviada al navegador web como respuesta a su petición.

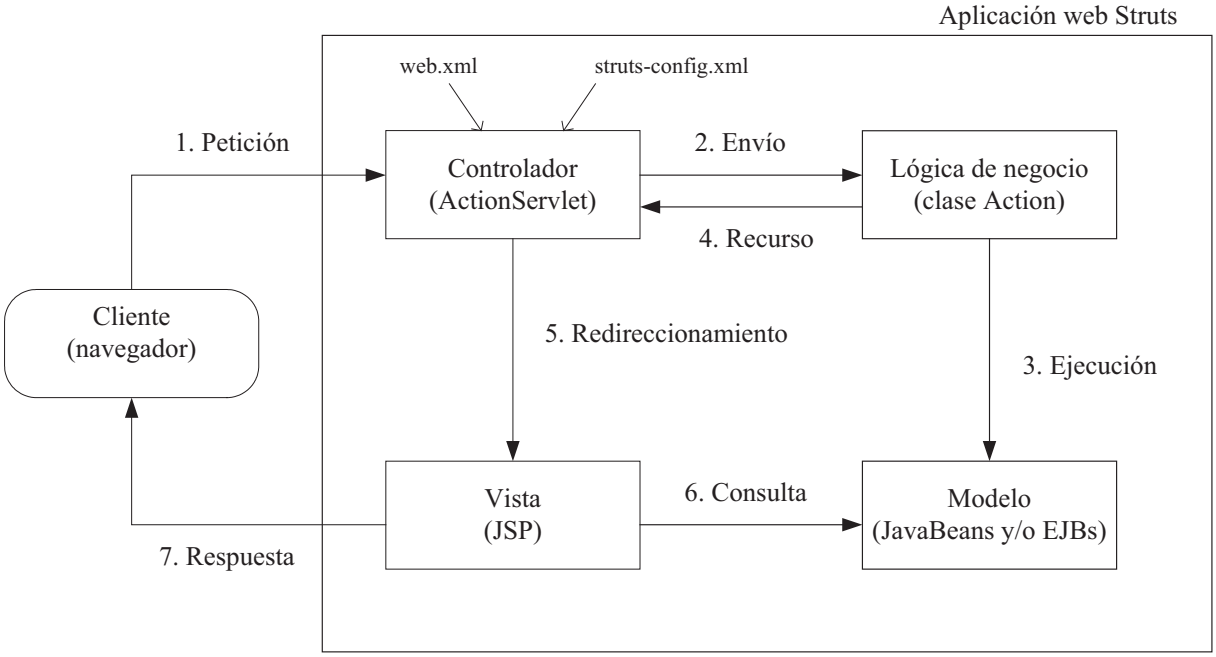

Figura 5.4: Diagrama de componentes de Struts

5.5. Componentes de Struts

En este apartado introduciremos los distintos componentes de Struts, describiendo brevemente las funciones que realizan. La idea es proporcionar una idea básica y global de dichos componentes, puesto que una explicación detallada requeriría todo un libro. Para quien quiera profundizar nos remitimos a la bibliografía de consulta.

5.5.1. Componentes del Controlador

En Struts sólo existe un componente controlador: el servlet `ActionServlet`.

5.5.1.1. La clase `org.apache.struts.action.ActionServlet`

El servlet `ActionServlet` representa al Controlador en el patrón de diseño MVC y se implementa por la clase `org.apache.struts.action.ActionServlet` (clase que hereda de la clase estándar `javax.servlet.http.HttpServlet`). Existe una única instancia de este servlet por cada aplicación web y su función es la de procesar todas las peticiones que afectan al estado o modelo de la aplicación. Para cada petición, el `ActionServlet` selecciona e invoca a la clase Action encargada de ejecutar la lógica de negocio asociada. Una vez que la clase Action lleva a cabo la lógica de negocio, envía un objeto `ActionForward` al `ActionServlet`. Este objeto identifica a la página JSP, u otro recurso, usada para generar la respuesta. Entonces, el controlador `ActionServlet` utiliza el método `RequestDispatcher.forward()` del API Servlet para pasar el control a la página JSP identificada por el `ActionForward`. Esta página será la respuesta que enviaremos al cliente.

El controlador ejecuta las siguientes operaciones durante el manejo de una petición:

1. A partir de la URI de la petición, determina la clase Action que debe ejecutarse y crea un objeto `ActionMapping`. Este objeto contiene toda la información que el controlador `ActionServlet` conoce acerca de la asignación entre la petición y la clase Action asociada.

2. Crea una instancia del bean `ActionForm`, en caso de que se haya declarado uno en la asignación petición-acción, e intenta establecer el valor de las propiedades del bean a partir de los parámetros de la petición, contenidos en el objeto `Request`. El objeto `Request` contiene toda la información relativa a la petición. Opcionalmente, ejecutará el método `validate()` del bean, que se encarga de validar las propiedades.

3. Invoca el método `execute()`, antiguo método `perform()` de las versiones 1.0.x de Struts, sobre la instancia de la clase Action, pasándole como parámetros los objetos `ActionMapping`, `ActionForm`, `Request` y `Response`. Este método se encarga de procesar la petición.

4. Acepta el `ActionForward` que devuelve el método `execute()` y le pasa el control al recurso determinado por el `ActionForward` mediante el método `forward()` del objeto `RequestDispatcher` o el método `sendRedirect()` del objeto `HttpServletResponse`.

La Figura 5.5 muestra todo este proceso.

Al igual que cualquier otro servlet, el `ActionServlet` debe ser declarado en el archivo `web.xml` y configurado de forma que se cargue durante el inicio.

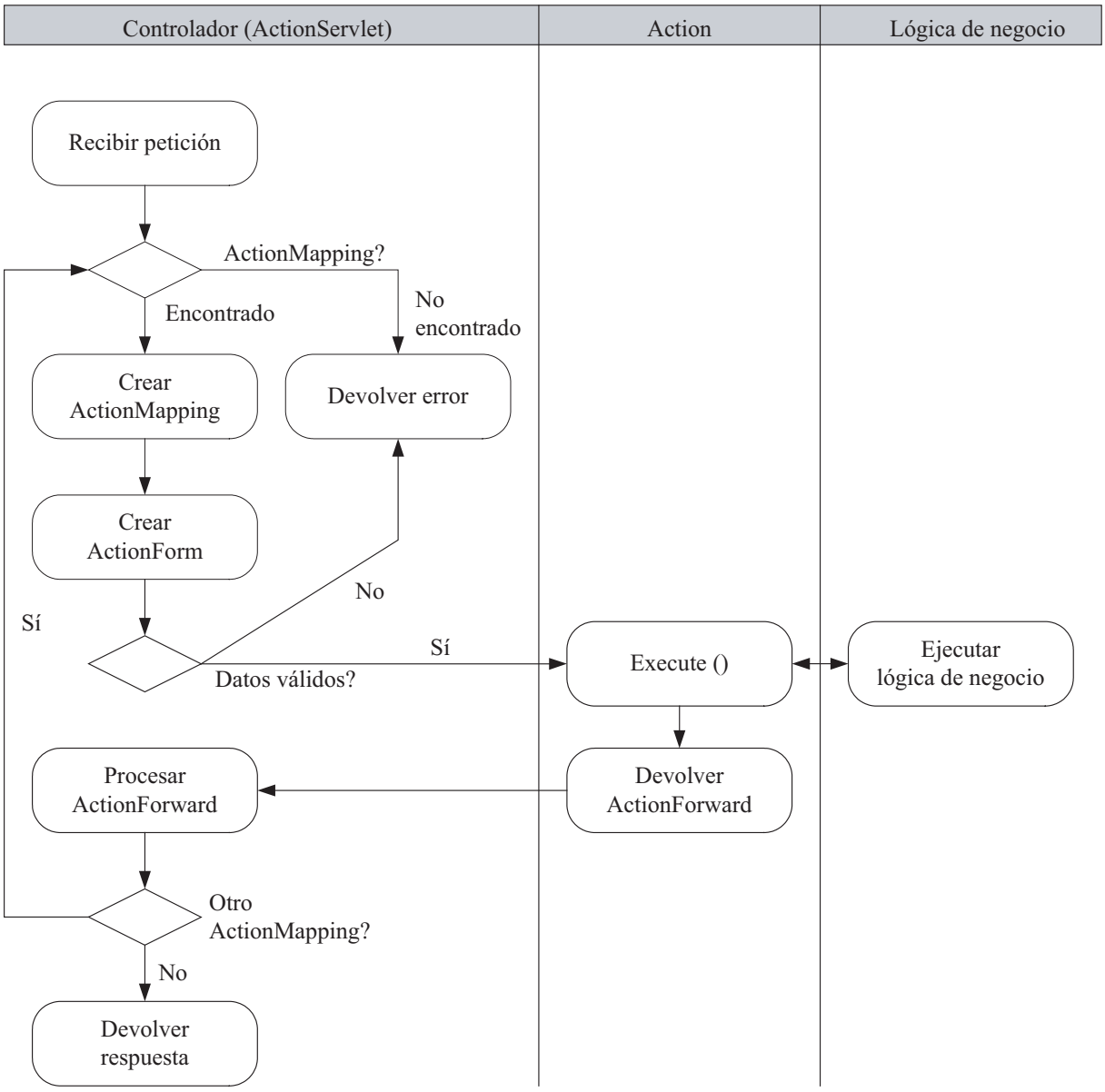

Figura 5.5: Diagrama de actividad petición/respuesta

5.5.2. Componentes del Modelo

Los componentes del Modelo son los encargados de mantener el estado del sistema y proporcionar acciones que puedan cambiar dicho estado. Dichas acciones constituyen la lógica de negocio del sistema y están representadas por las clases Action. El paquete `org.apache.struts.action.Action` contiene las clases que nos permiten construir nuestros propios componentes Action. Veamos algunas las clases fundamentales que componen este paquete.

5.5.2.1. La clase `org.apache.struts.action.ActionMapping`

Para poder operar satisfactoriamente, el `ActionServlet` necesita conocer varias cosas sobre cómo se debería mapear toda URI solicitada a una clase Action apropiada. El conocimiento requerido ha sido encapsulado en una clase Java, llamada `ActionMapping`. Un

`ActionMapping` representa la información que el `ActionServlet` conoce acerca de una asignación petición-Action concreta. Esta información es pasada al método `execute()` de la clase Action, permitiendo, de esta manera, que esta clase pueda acceder directamente a esta información. Los `ActionMapping` se definen mediante el elemento `<action>` del archivo de configuración `struts-config.xml`. Veamos un ejemplo de cómo asociar una URI a una clase Action determinada:

```
<action-mappings>
  <action path="/search"
          type="proyecto.actions.SearchAction"
          name="searchForm"
          scope="request"
          input="/search.jsp">
    <forward name="success" path="/display.jsp"/>
  </action>
</action-mappings>
```

La propiedad `path` contiene la URI de la petición.

La propiedad `type` especifica el nombre completo de la clase Action.

La propiedad `name` contiene el nombre del bean `ActionForm` asociado a la clase Action, en el caso de que lo tenga

La propiedad `scope` identifica el ámbito dónde se creará el bean `ActionForm`.

La propiedad `input` define la ruta relativa del formulario de entrada al que se debe devolver el control en caso de producirse un error de validación.

El elemento `<forward>` asigna un nombre lógico a un recurso web. Este nombre lógico puede ser utilizado por la clase Action para decirle al `ActionServlet` a qué recurso de la Vista (página JSP, servlet...) debe pasarle el control. Se pueden especificar múltiples elementos `<forward>` dentro de un mismo elemento `<action>`.

5.5.2.2. La clase `org.apache.struts.action.Action`

Como vimos, un Action es un adaptador entre el contenido de una petición HTTP y la lógica de negocio que debería ejecutarse para procesar dicho contenido. Podemos pensar en cada Action como si fuera el pegamento que une la petición del cliente con la lógica de negocio asociada a dicha petición. Las clases Action suelen ser clases que delegan en JavaBeans o EJBs para ejecutar la lógica de negocio. Para cada petición, el `ActionServlet` selecciona la clase `Action` apropiada, crea una instancia de la misma (si no existe) y llama al método `execute()`.

5.5.2.3. La clase `org.apache.struts.action.ActionForward`

Un objeto `ActionForward` identifica dónde debería reenviar el control el `ActionServlet` (por ejemplo a una JSP) para proporcionar la respuesta apropiada. Representa, por tanto, un destino, un recurso web al que enviar el control. La clase Action puede crear dinámicamente instancias de la clase `ActionForward` o utilizar las contenidas en el objeto `ActionMapping`.

Los `ActionForward` pueden definirse dentro de los elementos `<action-mapping>` del archivo de configuración `struts-config.xml`:

```
<forward name="exito" path="display.jsp" />
```

La propiedad `name` define un nombre lógico que identifica al recurso. Este nombre puede usarse para buscar un destino dentro de un `ActionMapping`.

La propiedad `path` define la URI del recurso web.

También es posible usar la sección `global-forwards` del archivo de configuración `struts-config.xml` para definir destinos globales. De esta manera, podemos crear nombres lógicos para los recursos web más usados sin necesidad de tener que definirlos para cada una de las acciones Struts. Una sección `global-forwards` de ejemplo podría ser la siguiente:

```
<global-forwards>
  <forward name="logoff"  path="/logoff.do"/>
  <forward name="logon"   path="/logon.jsp"/>
  <forward name="success" path="/mainMenu.jsp"/>
</global-forwards>
```

Todos estos destinos se ponen a disposición de todas las acciones Struts (a menos que se anulen dentro de su respectivo elemento `<action>`). La manera de acceder a estos destinos es a través de la instancia `ActionMapping`, por ejemplo, haciendo la llamada `actionForwardInstance.findForward("success")`.

5.5.2.4. La clase `org.apache.struts.action.ActionError`

El mecanismo que implementa Struts para devolver los errores ocurridos durante la validación de un `ActionForm` hace uso de la clase `ActionError`. Un `ActionError` encapsula un mensaje de error devuelto por el método `validate()` de un `ActionForm`. Cada error consiste en una clave, mediante la cuál se puede conseguir el mensaje de texto asociado a dicho error. Todos los mensajes de texto se encuentran en el archivo de recursos de mensajes de la aplicación. Los `ActionError` son almacenados dentro de una colección llamada `ActionErrors`. Un `ActionErrors` que no es más que una colección, un conjunto de pares

clave-valor, donde la clave es el nombre de la propiedad del `ActionForm` (si el error está asociado a una propiedad específica del `ActionForm`) o `ActionErrors.GLOBAL_ERROR` (si el error es global), y el valor es un objeto `ActionError`.

Veamos un método `validate()` de ejemplo para ver como se trabaja con los errores:

```
public ActionErrors validate(ActionMapping mapping,
                             HttpServletRequest request) {

  ActionErrors errors = new ActionErrors();

  if ((username == null) || (username.length() < 1)) {
    errors.add("username",
               new ActionError("error.username.required"));
  }
  return errors;
}
```

En este ejemplo, creamos una instancia de `ActionErrors`. Esta colección está vacía a menos que nuestro criterio de validación falle. En ese caso, creamos una instancia de un `ActionError` con la clave del mensaje y lo añadimos a la colección usando `username` como clave.

Cuando queramos visualizar todos los errores en una página JSP, utilizaremos la etiqueta `<html:errors>`.

En Struts 1.2, la clase `ActionError` ha sido desaprobada (*deprecated*) en favor de la clase `ActionMessage`. Por desaprobada se entiende que desaparecerá en futuras versiones de Struts. Por otro lado, la clase `ActionErrors` no ha sido desaprobada; sin embargo, se recomienda utilizar la clase `ActionMessages` en su lugar, siempre que sea posible.

5.5.2.5. La clase `org.apache.struts.action.ActionMessage`

Los `ActionMessage` (y los `ActionMessages`) funcionan igual que los `ActionError` (y los `ActionErrors`). La clase `ActionMessages` se introdujo en Struts 1.1. La diferencia es que los `ActionMessage` pueden usarse para pasar mensajes informativos que no son errores. Esto es útil cuando tenemos, por ejemplo, una página de confirmación de una aplicación de correo electrónico que muestra al usuario un mensaje informativo del tipo «Mensaje enviado satisfactoriamente».

Podemos visualizar los mensajes en una página JSP utilizando la etiqueta Struts `<html:messages>`. Esta etiqueta también sirve para visualizar los errores.

En Struts 1.2 los `ActionMessage` sirven para encapsular tanto errores como mensajes informativos, puesto que la clase `ActionError` ha sido desaprobada en dicha versión.

5.5.3. Componentes de la Vista

Los componentes de la Vista son las páginas JSP y las clases `ActionForm`, que son beans asociados a los formularios de las páginas JSP. Las bibliotecas de etiquetas de Struts también pueden considerarse componentes de la Vista. Veamos la clase `ActionForm`.

5.5.3.1. La clase `org.apache.struts.action.ActionForm`

Un `ActionForm` es un JavaBean opcionalmente asociado a uno o más `ActionMapping`. Los `ActionForm` se definen dentro del elemento `<form-beans>` del archivo de configuración `struts-config.xml`:

```
<form-beans>
  <form-bean name="searchForm" type="proyecto.forms.SearchForm"/>
</form-beans>
```

Normalmente tendremos más de un elemento `<form-bean>`. El atributo `name` asigna un nombre lógico al formulario. El atributo `type` especifica el nombre de la clase `ActionForm`. Esta clase debe extender la clase `org.apache.struts.action.ActionForm`.

Los `ActionForm` se utilizan para pasar información entre los componentes de la Vista y los componentes del Modelo. Esta transferencia de información es llevada a cabo por el `ActionServlet`, que realiza las siguientes operaciones:

- Comprueba si se ha configurado un `ActionForm` en el `ActionMapping`. En caso afirmativo, utiliza el atributo `name` del elemento `<action>` para buscar la información de configuración del bean.
- Comprueba si existe una instancia ya creada del `ActionForm`. En el caso de que no exista, se creará una nueva instancia.
- Invoca al método `reset()` del `ActionForm`.
- Establece el valor de las propiedades del bean a partir de los parámetros de la petición, contenidos en el objeto `Request`.
- Si el atributo `validate` del elemento `<action>` está a `true`, invoca al método `validate()` del bean. Si el método devuelve algún error, entonces el `ActionServlet` pasa el control al formulario de entrada correspondiente. Si no, llama al método `execute()` de la acción adecuada.

La Figura 5.6 muestra todo este proceso.

La clase `ActionForm` es una clase abstracta, por lo que los `ActionForm` que creemos deberán ser subclases de la misma. Estas subclases deben implementar los métodos getter y

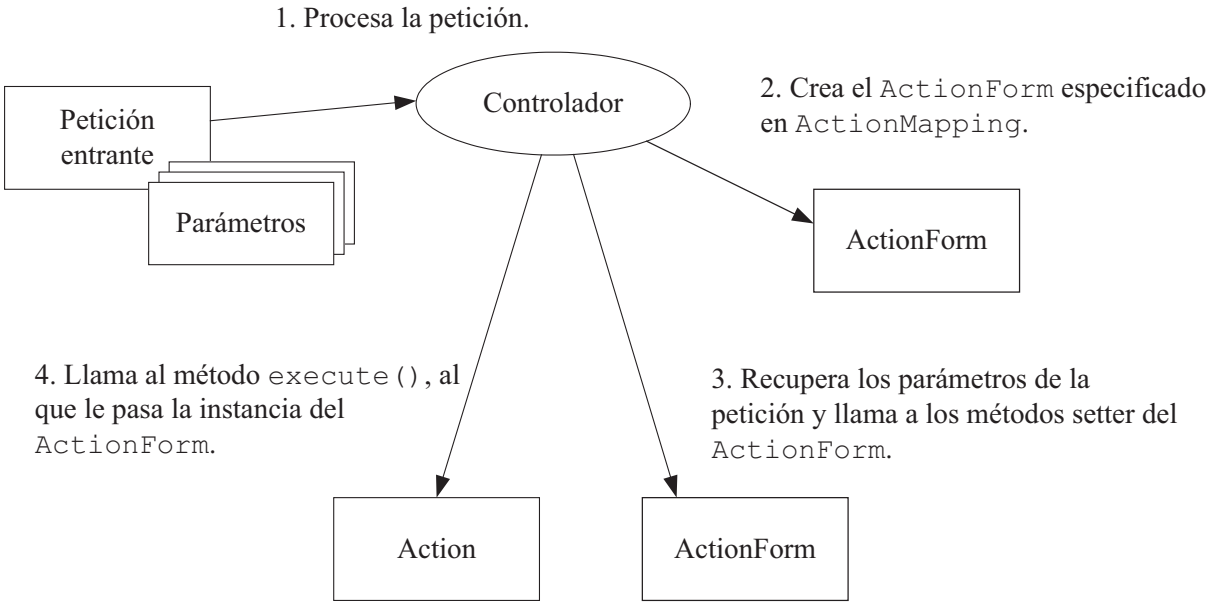

Figura 5.6: Procesamiento de un `ActionForm` por parte del controlador

setter para todas las propiedades del bean. Además, pueden redefinir los métodos, tanto públicos como protegidos, de su superclase. Generalmente, interesa redefinir los métodos `reset()` y `validate()`.

5.5.4. Otras clases y paquetes útiles

Hasta ahora hemos visto los componentes fundamentales de Struts. En este apartado vamos a exponer brevemente algunas características adicionales que pueden sernos de utilidad durante el desarrollo de nuestra aplicación.

5.5.4.1. Validación declarativa: framework Validator

El *framework* Validator nos permite definir validaciones del lado del servidor de forma declarativa. Estas validaciones se basan en unas reglas de validación que son declaradas en un archivo XML llamado `validation.xml`. De esta manera, no tenemos que implementar el método `validate()` del `ActionForm`.

5.5.4.2. Formularios dinámicos con `DynaActionForm`

La clase `DynaActionForm` se introdujo en Struts 1.1 y nos evita el tener que escribir los beans `ActionForm`. Esta clase nos permite crear propiedades dinámicas. Esto significa que podemos definir las propiedades en el archivo `struts-config.xml` y utilizar como tipo de formulario la clase `org.apache.struts.action.DynaActionForm`.

5.5.4.3. Internacionalización de la aplicación

Struts es considerado como un buen *framework* para construir aplicaciones que pueden ser usadas en múltiples idiomas y países. La base de la internacionalización (i18n) de Struts es Java, siendo esencial la clase `java.util.Locale`. Esta clase se utiliza para identificar una localidad (*locale*), es decir, un idioma particular y un país.

Podemos usar Struts para adaptar nuestras aplicaciones a distintas regiones e idiomas utilizando las siguientes características:

- Archivos de recursos de mensajes (`MessageResources`) que proporcionan mensajes y formatos adaptados a la localidad del usuario.
- Creación de objetos `Locale` basados en la configuración del navegador web del usuario.
- Especificación de patrones de formateo de fechas, números y monedas.
- Recuperación de imágenes específicas a la localidad.

Capítulo 6

La aplicación web: Codetic

El proceso de desarrollo de la aplicación se ha dividido en tres fases: análisis, diseño e implementación, adoptando así el modelo secuencial de desarrollo. La metodología utilizada ha sido UML (*Unified Modeling Language*, Lenguaje Unificado de Modelado), el estándar actual para desarrollos orientados a objetos. En la página de enlaces web del Wiki de Struts[1] se pueden encontrar varias aplicaciones desarrolladas con Struts cuyo estudio nos ha sido de utilidad.

El capítulo se divide en los siguientes apartados:

- Análisis. Se explican las tareas realizadas durante la fase de análisis de la aplicación: la especificación de requerimientos, el diagrama de casos de uso y el diagrama de clases del dominio [10, 47].
- Diseño. En la fase de diseño, presentaremos el modelo relacional de la base de datos y los diagramas de clases del sistema [10, 16, 47].
- Implementación. Finalmente, en la fase de implementación, hablaremos de la forma en que hemos construido la aplicación web [15, 59, 61].

6.1. Análisis

En el análisis hemos realizado la especificación de los requerimientos del sistema, es decir, hemos documentado lo que tiene que hacer el sistema. Además, también hemos identificado las clases principales del dominio del problema.

6.1.1. Especificación textual de requisitos

En el sistema tendremos dos tipos de usuarios: el Visitante (Usuario normal) y el Administrador de la aplicación web.

El Visitante podrá:

[1] Disponible en http://wiki.apache.org/struts/StrutsWebLinks

- Ver un listado de las asociaciones y sus códigos.
- Ver la información de una asociación.
- Ver un código ético.
- Realizar búsquedas de artículos de códigos. Se podrá buscar por palabras, por idiomas, por temas y por asociaciones.
- Ver un listado de enlaces relacionados con la ética y con los códigos de ética.
- Cambiar el idioma de la aplicación.

El Administrador, además de lo anterior, podrá:

- Conectarse al sistema (*login*).
- Desconectarse del sistema (*logout*).
- Administrar la información sobre asociaciones, códigos, temas, grupos de temas, ámbitos, idiomas y traducciones tanto de los códigos como de la información sobre asociaciones, temas, grupos de temas, ámbitos e idiomas. Se podrán insertar, consultar, modificar y eliminar asociaciones, códigos, temas, grupos de temas, ámbitos, idiomas, así como sus traducciones.

6.1.2. Diagrama de casos de uso

El diagrama de casos de uso constituye una especificación abstracta del comportamiento del sistema, es decir, nos ofrece una representación del sistema a un alto nivel.

Los objetivos de los casos de uso son los siguientes:

- Capturar los requisitos funcionales del sistema y expresarlos desde el punto de vista del usuario.
- Guiar el proceso de desarrollo de la aplicación.
- Proporcionar una herramienta de comunicación entre los usuarios, analistas y diseñadores.
- Proporcionar una visión general del sistema sin entrar en los detalles de cómo se realiza.

A continuación vamos a explicar los distintos elementos de un diagrama de casos de uso.

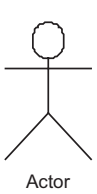

Figura 6.1: Representación gráfica de un actor.

Figura 6.2: Representación gráfica de un caso de uso.

6.1.2.1. Componentes del diagrama

Un diagrama de casos de uso es un grafo constituido por actores, casos de uso y relaciones entre los actores y los casos de uso. Veamos cada uno de estos elementos:

- **Actores**. Un actor es algo o alguien que se encuentra fuera del sistema y que interactúa con él. En general, los actores son los usuarios del sistema u otros sistemas externos. Un único actor puede representar a varios usuarios distintos, y un usuario puede actuar como diferentes actores. Un actor se representa gráficamente mediante un muñeco con su nombre debajo (Figura 6.1).

- **Casos de uso**. Un caso de uso representa el comportamiento que ofrece un sistema desde el punto de vista del usuario. Básicamente será un conjunto de transacciones ejecutadas entre el sistema y los actores. Gráficamente se representa con una elipse en cuyo interior se incluye el nombre del caso de uso (Figura 6.2).

- **Relaciones**. Existen varios tipos de relaciones: comunicación, especialización, inclusión y extensión. Estas relaciones se pueden dar entre casos de uso, entre actores, y entre un actor y los casos de uso. A continuación se realizará una descripción más detallada de cada una de ellas.

  - **Caso de uso-Caso de uso**. Las relaciones que se pueden dar entre los casos de uno son tres: generalización/especificación, inclusión y extensión. Veamos cada una de ellas:

    - **Generalización/Especificación**. En este tipo de relación, un caso de uso hijo hereda el comportamiento y el significado de otro caso de uso padre. El caso de uso hijo puede añadir comportamiento o bien redefinir el del padre. Gráficamente se representa mediante una flecha hueca dirigida al caso de uso padre (Figura 6.3).

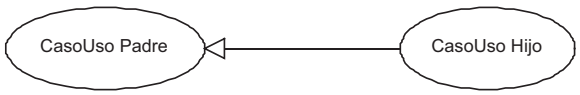

Figura 6.3: Representación gráfica de la relación de generalización/especificación entre casos de uso.

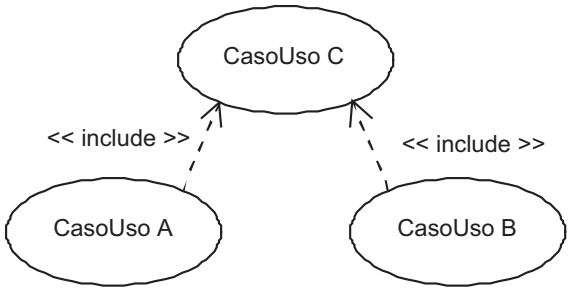

Figura 6.4: Representación gráfica de la relación de inclusión entre casos de uso.

  - **Inclusión**. La relación de inclusión representa un comportamiento común entre varios casos de uso. Un caso de uso A incorpora explícitamente el comportamiento de otro caso de uso B. Entonces este comportamiento común se recoge en un tercer caso de uso C que se relaciona con los casos de uso Ay B mediante una relación de inclusión. Su representación gráfica es mediante una flecha discontinua etiquetada con el estereotipo «usa» (o «include») hacia el caso de uso común (Figura 6.4).
  - **Extensión**. Esta relación representa un comportamiento opcional de un caso de uso. Por lo tanto, un caso de uso base A que esté relacionado con otro caso de uso B puede escoger o no el comportamiento del caso de uso B. Se representa gráficamente mediante una flecha etiquetada con el estereotipo «extiende» (o «extend») que llega al caso de uso base (Figura 6.5).

- **Actor-Actor**. La relación que pueden mantener dos actores es la relación de generalización. En este tipo de relación, el actor hijo hereda el comportamiento de un actor padre, además de poder definir los atributos propios que los diferencia de los demás. Gráficamente se representa mediante una flecha con la cabeza hueca dirigida al actor más general (Figura 6.6).
- **Actor-Caso de uso**. La relación que puede haber entre un actor y un caso de uso

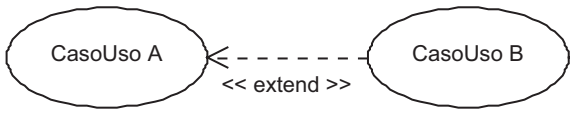

Figura 6.5: Representación gráfica de la relación de extensión entre casos de uso.

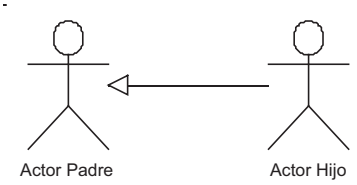

Figura 6.6: Representación gráfica de la relación de generalización entre actores.

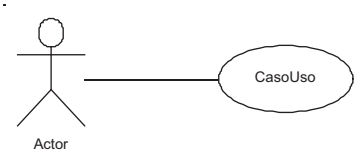

Figura 6.7: Representación gráfica de la relación de comunicación entre un actor y un caso de uso.

es una relación de comunicación, que puede ser unilateral o bilateral. Se representa gráficamente mediante una línea continua (Figura 6.7).

6.1.2.2. Diagrama de casos de uso de la aplicación

En la Figura 6.8 mostramos el diagrama de casos de uso de la aplicación web del proyecto.

6.1.3. Diagrama de clases del dominio

El objetivo principal de los diagramas de clases es la representación de los aspectos estáticos del sistema, utilizando para ello diversas técnicas de abstracción como pueden ser la clasificación, la generalización y la abstracción. Los diagramas de clases recogen las relaciones entre los distintos objetos que componen el sistema.

6.1.3.1. Componentes del diagrama

Los elementos que constituyen un diagrama de clases son:

- **Clases**. Las clases agrupan o abstraen objetos con atributos, relaciones y operaciones comunes y con la misma semántica y comportamiento. Se representan gráficamente como una caja separada en tres zonas por líneas horizontales (Figura 6.9). En la zona superior se identifica el nombre de la clase. La zona central contiene una lista de atributos y la zona inferior incluye una lista con las operaciones que proporciona la clase.
- **Objetos**. Elementos de una clase (instancias de esa clase).
- **Atributos**. Propiedad de una clase compartida por todos sus objetos. Pueden ser de dos tipos: base y derivados. La diferencia entre ellos es que los segundos son calculados a partir de los primeros. Los atributos de una clase representan los datos asociados a los

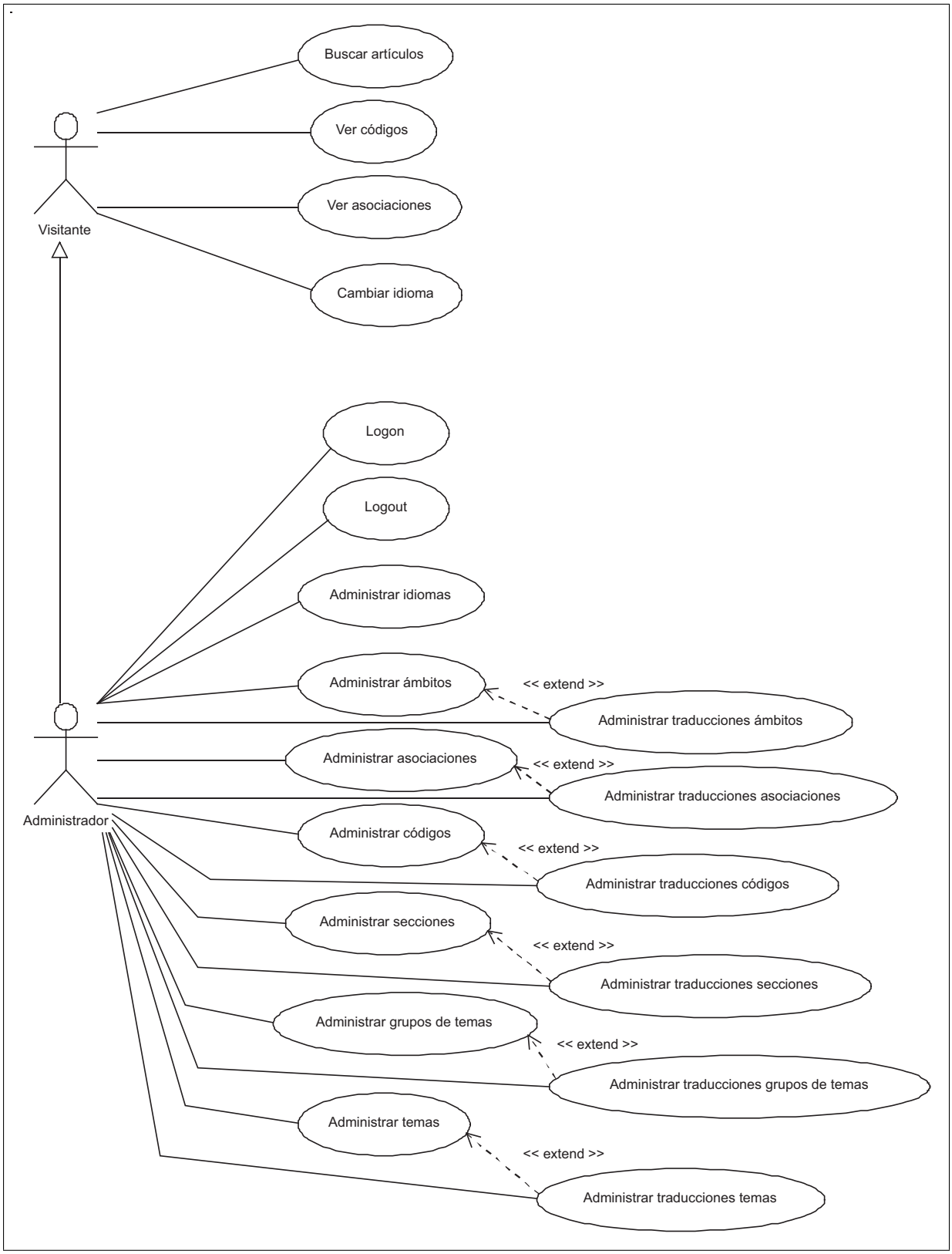

Figura 6.8: Diagrama de casos de uso de la aplicación.

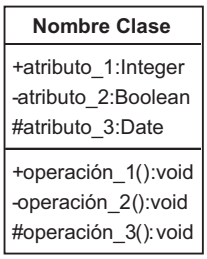

Figura 6.9: Representación gráfica de una clase.

objetos instanciados por esa clase. Cada atributo posee un nombre único en su clase y el conjunto de valores de los atributos determina el estado del objeto. Los atributos siguen el siguiente formato:

{visibilidad} atributo:tipo = valor-inicial

Existen tres tipos de visibilidad, tanto para los atributos como para las operaciones: público (+), privado (-) y protegido (#).

- **Operaciones**. Representan el comportamiento de los objetos de una clase y lo caracterizan. Las operaciones pueden modificar los datos de un objeto, o lo que es lo mismo, su estado. Siguen el siguiente formato:

  {visibilidad} operación (lista-de-argumentos):tipo-devuelto

- **Relaciones**.

  Las relaciones son las conexiones que pueden tener los distintos elementos. Existen varios tipos de relaciones.

  - **Asociación**. Es una relación estructural que especifica que los objetos de un elemento están conectados con los objetos de otros. Mientras no se indique lo contrario, se suponen bidireccionales. Pueden ser unarias, binarias, ternarias, n-arias. Cada relación tiene una multiplicidad que especifica cuántas instancias de una clase están asociadas a una instancia de la otra clase. Cuando no se especifica ninguna multiplicidad, se asume el valor uno (1) por defecto. La representación gráfica de una asociación es una línea continua entre las clases asociadas, junto a la que puede haber un nombre que defina la semántica o naturaleza de la asociación (Figura 6.10).
    Los distintos tipos de multiplicidad son los siguientes:

Cero a 1:	0..1
Muchos:	0..n ó 0..*
Uno o más:	1..n o 1..*
Número exacto:	7
Combinaciones:	0..1, 3..4, 6..*

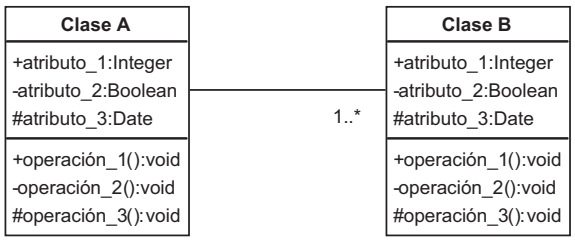

Figura 6.10: Representación gráfica de la relación de asociación entre clases.

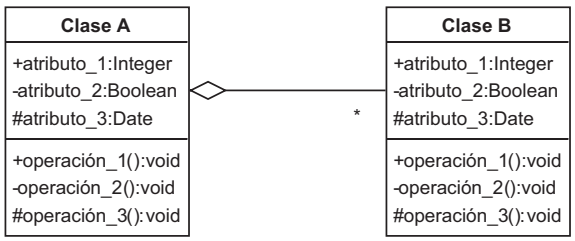

Figura 6.11: Representación gráfica de la relación de agregación entre clases.

- **Agregación**. La agregación, que es un caso particular de la asociación, es un tipo de relación jerárquica entre un objeto que representa la totalidad de ese objeto y las partes que lo componen. Permite el agrupamiento físico de estructuras relacionadas lógicamente. Los objetos «son parte de» otro objeto completo. La agregación se representa gráficamente mediante un rombo hueco en la clase cuya instancia es una agregación de las instancias de la otra (Figura 6.11).
- **Composición**. Una composición es una agregación más fuerte que implica:
  - Dependencia existencial: El elemento dependiente desaparece al destruirse el que lo contiene y, si es de cardinalidad 1, es creado al mismo tiempo.
  - Pertenencia fuerte: Se puede decir que el objeto contenido es parte constitutiva y vital del que lo contiene.
  - No compartición: Los objetos contenidos no son compartidos, esto es, no forman parte del estado de otro objeto.

  La composición se representa mediante un rombo relleno del lado de la clase que contiene a la otra en la agregación (Figura 6.12).
- **Dependencia**. Esta relación representa que una clase requiere a otra para proporcionar alguno de sus servicios. Su representación gráfica es una flecha con línea discontinua que apunta a la clase de la que depende (Figura 6.13).
- **Generalización o herencia**. La relación de generalización es una relación entre un elemento general (llamado superclase o clase padre) y una clase más específica de ésta (llamada subclase o clase hija). La clase hijo comparte la estructura (los atributos) y el comportamiento (las operaciones) de la clase padre. Además puede

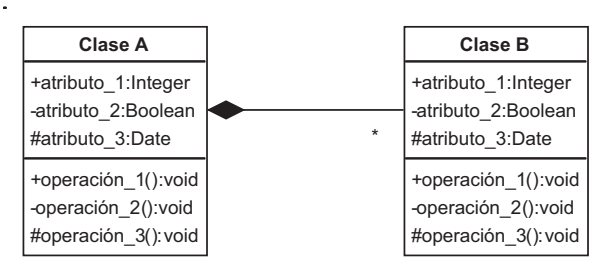

Figura 6.12: Representación gráfica de la relación de composición entre clases.

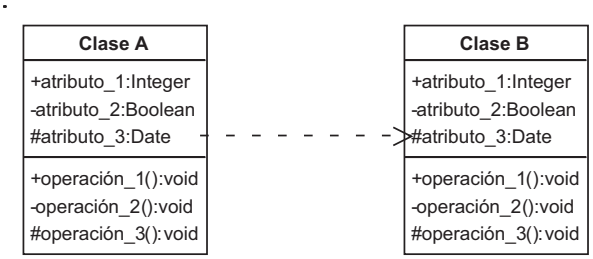

Figura 6.13: Representación gráfica de la relación de dependencia entre clases.

añadir atributos y operaciones propias y redefinir operaciones del padre. Existe dos tipos de herencia: simple y múltiple. La herencia simple se origina cuando la clase hija hereda de un solo padre. La herencia múltiple se da cuando la clase hija hereda de dos o más superclases. Esta relación se representa gráficamente mediante una flecha con una línea continua dirigida hacia la clase padre (Figura 6.14).

- **Paquetes**. Es el mecanismo de que dispone UML para organizar sus elementos en grupos. En el caso de los diagramas de clases, los paquetes se usan para dividir el modelo de clases agrupando clases u otros paquetes. Las dependencias entre los paquetes se definen a partir de las relaciones establecidas entre los distintos elementos que se agrupan en éstos. Estas dependencias se representan mediante flechas discontinuas entre los paquetes dependientes. La representación gráfica de un paquete es un icono con forma de carpeta (Figura 6.15).

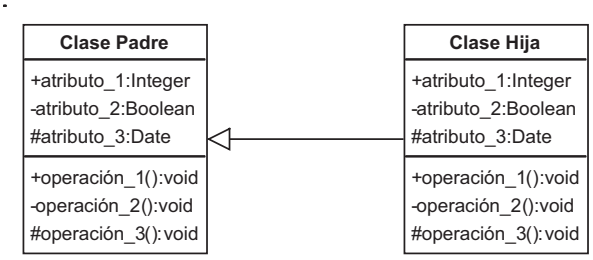

Figura 6.14: Representación gráfica de la relación de generalización (o herencia) entre clases.

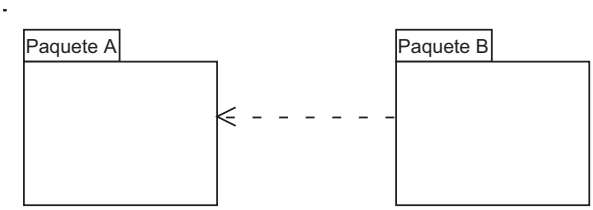

Figura 6.15: Representación gráfica de la dependencia entre paquetes.

6.1.3.2. Diagrama de clases de la aplicación

En la Figura 6.16 mostramos el diagrama de clases del dominio del problema. Este es un modelo descriptivo del problema que queremos resolver, en el que identificamos todos aquellos conceptos característicos del problema, susceptibles de expresarse como clases de objetos.

A continuación, vamos a explicar detalladamente cada una de las clases de este diagrama.

Clase Idioma

La clase Idioma encapsula todos los atributos necesarios para almacenar un idioma. Esta clase se relaciona con todas las traducciones de entidades, ya que estas traducciones deben estar escritas en algún idioma concreto.

Los atributos de esta clase son los siguientes:

- idIdioma. Representa el código de idioma ISO definido por el estándar ISO-639-1, disponible en http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt. Este estándar proporciona códigos de dos letras escritas en minúsculas para 136 lenguas del mundo. Por ejemplo, el código ISO-639-1 para el inglés es «en».

- nombre. Representa el nombre del idioma escrito en el mismo idioma. Por ejemplo, para el español sería «español», mientras que para el inglés sería «english».

Clase Ambito

Esta clase representa a los posibles ámbitos geográficos a los que pueden pertenecer las distintas asociaciones. Sus atributos son los siguientes:

- idAmbito. Un número entero que identifica unívocamente al ámbito.

- nombreAdmon. Un nombre que representa, independientemente del idioma, a un ámbito determinado y es utilizado para la administración de Codetic. Puesto que el idioma nativo del autor del proyecto es el español, este atributo estará escrito en español. Lo mismo pasará con el atributo nombreAdmon del resto de clases. Como ejemplo, un posible valor para este atributo sería «Internacional».

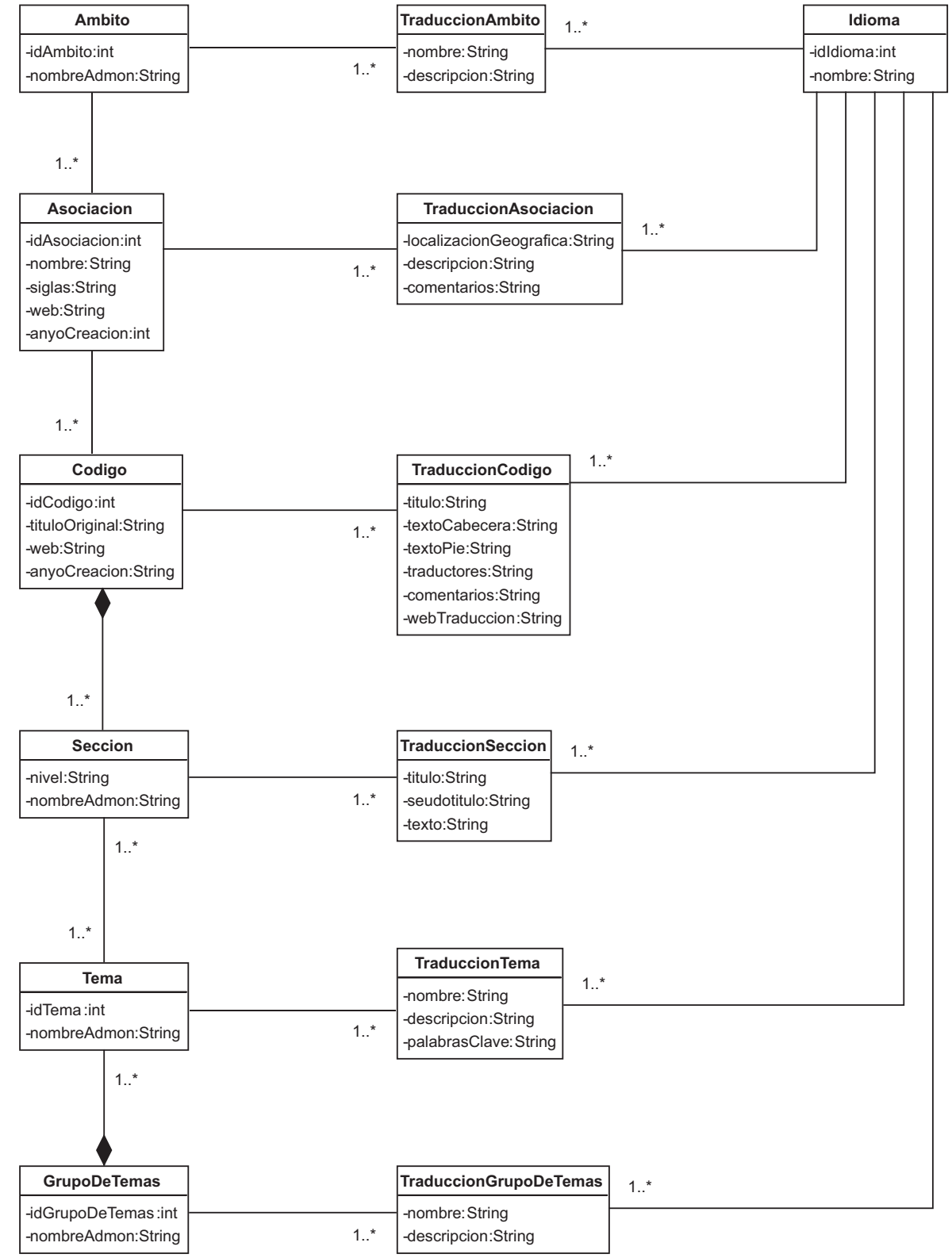

Figura 6.16: Diagrama de clases del dominio del problema.

Clase Asociacion

La clase Asociacion encapsula toda aquella información independiente del idioma que es relativa a una asociación informática. La información que depende del idioma, es decir, la información que puede ser escrita en distintas lenguas, es recogida en otra clase, la clase TraduccionAsociacion, que veremos más adelante.

La atributos de esta clase son los siguientes:

- idAsociacion. Es un número entero que identifica unívocamente a la asociación.
- nombre. Representa el nombre completo de la asociación escrito en el idioma «oficial» de esta. Para las asociaciones españolas este nombre estará escrito en español, para las italianas en italiano, para las internacionales en inglés, etc.
- siglas. Contiene las siglas de la asociación.
- web. Contiene la dirección URL del sitio web oficial de la asociación.
- anyoCreacion. Recoge el año en que se creó la asociación.

Clase Codigo

La clase Codigo es una abstracción de los códigos éticos. Al igual que la clase Asociacion, sólo recoge aquellos atributos que son independientes del idioma:

- idCodigo. Es un número entero que identifica unívocamente al código ético.
- tituloOriginal. Contiene el título del código escrito en su idioma original.
- web. Recoge la dirección URL de la página web oficial del código ético.
- anyoAprobacion. Contiene el año en que se creó (aprobó) el código ético por parte de la asociación.

Clase Seccion

La clase Seccion encapsula toda aquella información independiente del idioma que es relativa a una sección de un código ético. Por sección entendemos uno de los apartados en los que se ha dividido un código. Esta división puede ser tanto natural (impuesta por la propia estructura del código) como artificial (creada por el propio administrador de Codetic). Por lo tanto, el contenido de un código estará formado por el contenido de todas sus secciones.

Veamos los atributos de esta clase:

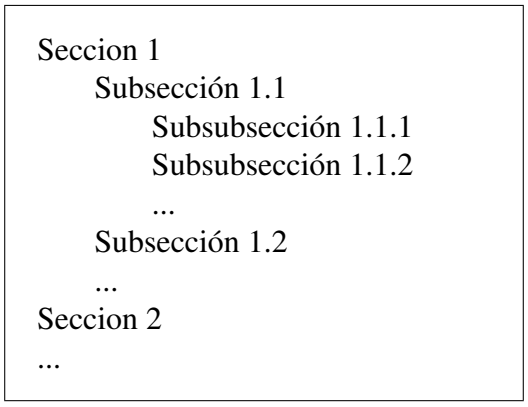

Figura 6.17: Estructura jerárquica de los códigos éticos.

- nivel. Indica todos los niveles a los que pertenece la sección. Se representa mediante una cadena formada por los niveles separados por un punto «.». De esta forma, si almacena «3.10.6», indica que se trata del artículo 6 de la subsección 10 de la sección 3. Este atributo nos permite establecer una jerarquía dentro de las secciones de un código (Figura 6.17). Así, tendremos secciones de primer nivel, con nivel=1,2..., de segundo nivel, con nivel=1.1,1.2..., etc.
- nombreAdmon. Contiene un nombre que representa, independientemente del idioma, a una seccion determinada y es utilizado para la administración de Codetic. Normalmente coincidirá con el título de la sección. Algunos posibles valores de este atributo son «Preámbulo», «Principio 1», etc.

Clase Tema

Esta clase representa a los posibles temas que pueden asociarse a las secciones. Un tema no es más que un asunto o problemática relacionado con la ética informática. Sus atributos son los siguientes:

- idTema. Identifica unívocamente al tema.
- nombreAdmon. Este atributo da nombre, independientemente del idioma, a un tema determinado y es utilizado para la administración de Codetic.

El Apéndice A muestra la lista de temas que hemos utilizado para clasificar los códigos éticos.

Clase GrupoDeTemas

La clase GrupoDeTemas encapsula toda aquella información independiente del idioma relativa a un grupo de temas. Como vimos en el Apartado 1.2.3, estos grupos permiten clasificar los temas.

Los atributos de esta clase son los siguientes:

- idGrupoDeTemas. Identifica unívocamente al grupo de temas.
- nombreAdmon. Contiene un nombre que representa, independientemente del idioma, al grupo de temas. Es utilizado para la administración de Codetic.

Clase TraduccionAmbito

Esta clase recoge toda aquella información, relativa a un ámbito, que varía según el idioma. Dicho de otra manera, contiene aquellos atributos que son dependientes del idioma. Por tanto, la clase TraduccionAmbito representa a las posibles traducciones de un ámbito.

Veamos los atributos de esta clase:

- nombre. Es una cadena de texto, escrita en un idioma concreto, que sirve para designar al ámbito.
- descripción. Contiene la descripción del ámbito escrita en un idioma determinado.

Clase TraduccionAsociacion

Esta clase representa las posibles traducciones de la información de una asociación concreta. Sus atributos son los siguientes:

- localizacionGeografica. Traducción de la zona geográfica cubierta por la asociación. Por ejemplo, «Nacional (España)» para el español, «National (Spain)» para el inglés, etc.
- descripcion. Contiene la descripción de la asociación escrita en un idioma determinado.
- comentarios. Sirve para añadir cualquier cosa sobre la asociación, que consideremos interesante, como puede ser el número de socios, la dirección postal, el director o presidente de la asociación, etc.

Clase TraduccionCodigo

La clase TraduccionCodigo encapsula toda la información necesaria para almacenar la traducción de un código ético. Posee los siguientes atributos:

- titulo. Título del código.
- textoCabecera. Texto que está situado entre el título del código y la primera sección. Lo normal es que contenga subtítulos, dedicatorias, etc.
- textoPie. Texto que va al final del código, inmediatamente después de la última sección. Normalmente trata sobre la autoría del código, posibles reconocimientos a personas o instituciones, la fecha de adopción del código, etc.

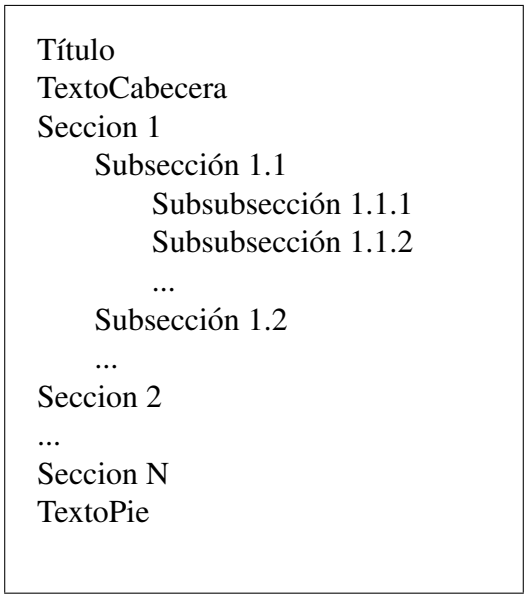

Figura 6.18: Estructura general de los códigos éticos.

- traductores. Autores de la traducción del código.
- comentarios. Comentarios sobre la traducción del código.
- webTraduccion. Página web de la traducción del código.

Con el fin de aclarar la forma en la que hemos organizado los códigos, en la Figura 6.18 podemos ver cual es la posición que ocupan los atributos textoCabecera y textoPie dentro de la estructura global de un código.

Clase TraduccionSeccion

Esta clase encapsula toda la información necesaria para almacenar la traducción de una sección. Sus atributos son los siguientes:

- titulo. Título «natural» de la sección. Es el título que tiene la sección en el código. Pueden existir secciones sin título.
- seudotitulo. Título «artificial» de la sección. Cuando una sección no tiene título, es necesario darle uno. Este es el título que usaremos para las secciones sin título «natural» cuando vayamos a mostrar los resultados de una búsqueda.
- texto. Texto de la sección.

Clase TraduccionTema

La clase TraduccionTema representa las posibles traducciones de un tema. Tiene los siguientes atributos:

- nombre. Nombre del tema.
- descripcion. Descripción del tema.
- palabrasClave. Lista de palabras, términos, frases, etc. que ayudan a describir y definir el tema.

Clase TraduccionGrupoDeTemas

Esta clase encapsula toda la información necesaria para almacenar la traducción de un grupo de temas. Sus atributos son los siguientes:

- nombre. Nombre del grupo.
- descripcion. Descripción del grupo.

6.2. Diseño

En la fase de diseño se determina la arquitectura general del sistema y su comportamiento dinámico, utilizando para ello la especificación realizada en la etapa de análisis. En nuestro caso, hemos modelado tanto la base de datos, mediante el modelo relacional del dominio, como la aplicación web, mediante un diagrama de clases que nos permite representar la interrelación que existe entre los distintos elementos de la misma.

6.2.1. Modelo relacional del dominio

El objetivo de esta tarea es obtener el modelo relacional de datos a partir del modelo de clases. En este modelo, todos los datos están estructurados a nivel lógico como tablas formadas por filas y columnas. Por lo tanto, se aplicarán una serie de reglas de transformación que conserven lo mejor posible la semántica del modelo de clases. Cada uno de los elementos del modelo de clases se tiene que transformar en un elemento del modelo relacional. En alguno casos, la transformación es directa, pero otras veces no existe esta correspondencia, por lo que es necesario buscar una transformación adecuada al modelo.

6.2.1.1. Reglas de transformación del diagrama de clases

Veamos las reglas de transformación que debemos aplicar a cada uno de los elementos del diagrama de clases:

- **Transformación de las clases**. Una clase se transforma en una tabla. Es posible que dos clases se transformen en una sola tabla cuando el comportamiento de una de ellas no sea importante para la base de datos.

- **Transformación de los atributos de las clases**. Cada atributo se transforma en una columna de la tabla en la que se transformó la clase a la que pertenece. El atributo que identifique únivocamente a cada objeto de la clase pasa a ser la clave primaria. En el caso de que hagan falta más atributos para la identificación de objetos, se tendrá en la tabla más de una clave primaria, formando así una clave primaria compuesta. Aquellas clases que dependen de otras obtendrán de éstas sus atributos únicos, los cuales se denominarán claves foráneas en la tabla dependiente.

- **Transformación de las asociaciones y agregaciones**. La transformación que se le aplica a este tipo de relaciones depende de la correspondencia que tengan éstas:

  - **Relaciones M:N**. Se transforman en una tabla, cuya clave primaria es la concatenación de las claves primarias de cada una de las tablas relacionadas, que serán claves foráneas en la nueva tabla.
  - **Relaciones 1:N**. Tenemos dos posibilidades:
    1. Propagar el identificador de la clase de cardinalidad 1 a la de cardinalidad N, teniendo en cuenta que:
       - La clave propagada es clave foránea en la tabla a la que se ha propagada.
       - Si la relación es «débil», la clave primaria de la tabla correspondiente a la clase débil estará formada por la concatenación de los identificadores de ambas clases.
    2. Transformar la relación en una tabla que tenga como clave primaria el identificador de la clase de cardinalidad máxima N si:
       - La relación posee atributos propios y se desea que aparezcan como tales.
       - Si se cree que en un futuro la relación puede convertirse en M:N.
       - El número de ocurrencias relacionadas de la clase que propaga su clave es muy pequeño.
  - **Relaciones 1:1**. Este es un caso particular de la relación 1:N y, por tanto, se puede crear una tabla única o propagar la clave. En principio, la clave se propaga en ambos sentidos, aunque hay que analizar la situación para elegir la solución que recoga mayor semántica y evite valores nulos.

- **Transformación de la herencia**. Las distintas formas que tenemos para tratar la transformacíón de la herencia son:

  1. Crear una tabla para la superclase que tenga como clave primaria el identificador de dicha clase y crear una tabla para cada una de las subclases con el identificador de la superclase como clave foránea. Esta es una solución adecuada cuando las subclases tienen muchos atributos distintos y se quieren conservar los atributos comunes en una tabla. Esta es la opción que mejor conserva la semántica.

2. Crear una tabla para cada subclase. Los atributos comunes aparecen en todas las subclases y la clave primaria para cada tabla es el identificador de la superclase. Esta opción mejora la eficiencia en los accesos a todos los atributos de una subclase.

3. Agrupar en una tabla todos los atributos de la superclase y las subclases. La clave primaria de esta tabla es el identificador de la superclase. Se añade un atributo que indique a qué subclase pertenece cada ocurrencia. Esta solución puede aplicarse cuando las subclases se diferencien en pocos atributos y tengan las mismas relaciones con otras clases.

6.2.1.2. Modelo relacional de la aplicación

La Figura 6.19 muestra el diagrama relacional de la base de datos del proyecto. Hemos utilizado la tabla de conversión entre tipos de datos MySQL y tipos de datos Java[2], para determinar el tipo de las columnas MySQL a partir del diagrama de clases del dominio.

Para optimizar la búsqueda de artículos, hemos creado índices sobre las columnas «título» y «texto» de la tabla «TraduccionesSecciones».

6.2.2. Diagrama de clases del sistema

UML proporciona un lenguaje estándar para modelar sistemas de software. Aun así, ningún lenguaje puede ser nunca suficiente para expresar todos los matices de todos los modelos en todos los dominios y en todo momento. Por esa razón, UML tiene varios mecanismos de extensión que permiten extender su sintaxis y su semántica para expresar los conceptos específicos de un determinado dominio de aplicación.

En el caso de los sistemas web, los bloques de construcción estándar que vienen con UML no son suficientes para expresar la relación que existe entre los distintos elementos de una aplicación web. Por ello, Conallen [16] desarrolló una extensión de UML, llamada WAE-UML (*Web Application Extension for UML*, Extensión de Aplicación Web para UML), para el modelado de aplicaciones web. Aunque esta extensión o perfil sirve para modelar cualquier tipo de aplicación web, hemos preferido crear una nueva extensión que nos permita modelar específicamente aplicaciones web Java que utilicen el *framework* Struts. La razón es que creemos que utilizando la nueva extensión se obtienen unos diagramas más sencillos de entender que con la extensión propuesta por Conallen. Por ejemplo, en WAE se debe utilizar una clase con estereotipo «HTML form» (6.2.2.1) por cada formulario HTML. En cambio, con nuestra extensión únicamente deberemos especificar en la clase Action el nombre del formulario asociado al mismo.

A continuación veremos los distintos mecanismos que ofrece UML para poder extender su sintaxis y su semántica. Después, explicaremos el perfil UML que hemos creado para poder

[2]Disponible en http://dev.mysql.com/doc/refman/5.0/en/cj-type-conversions.html

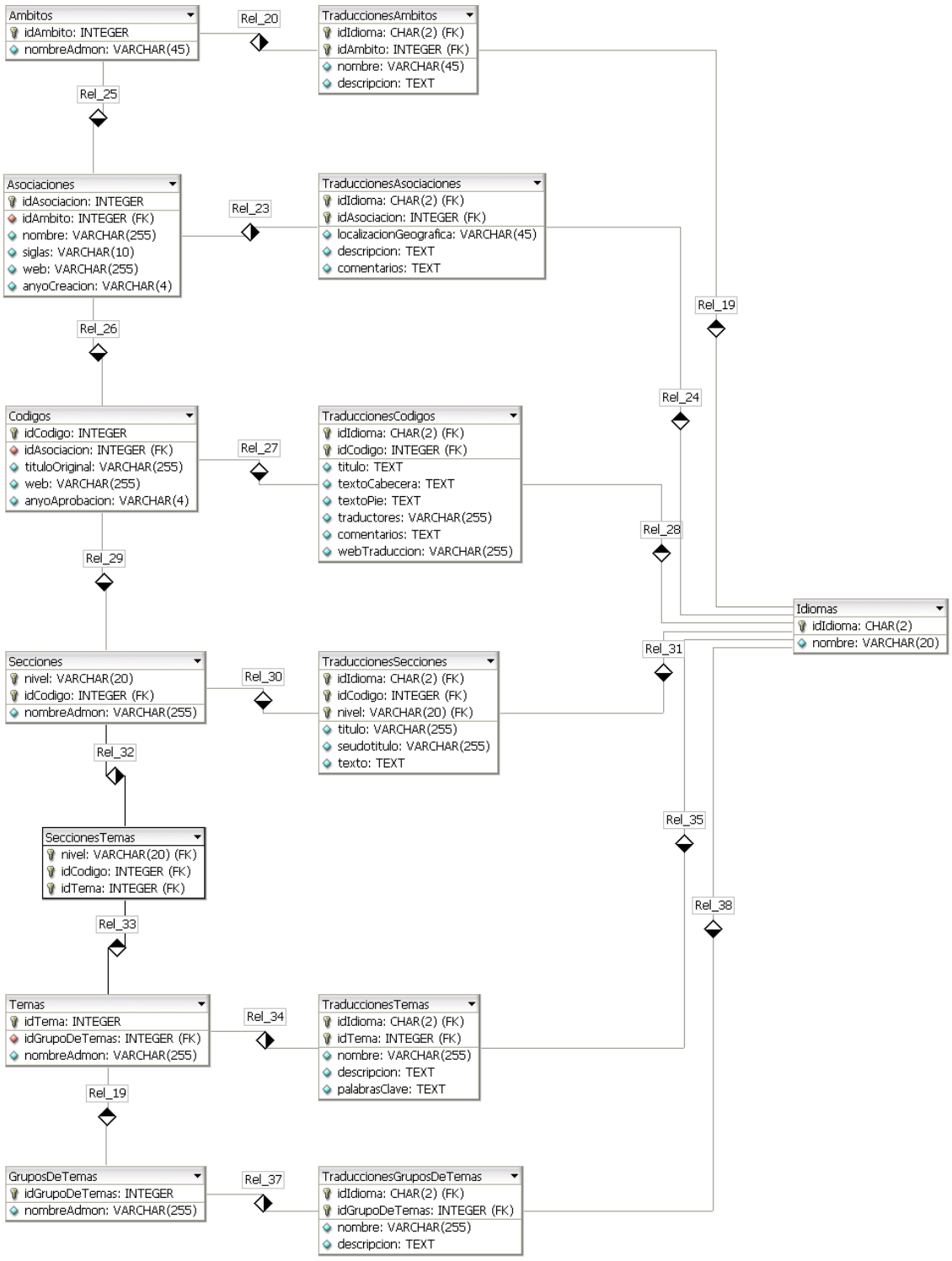

Figura 6.19: Modelo relacional del dominio del problema.

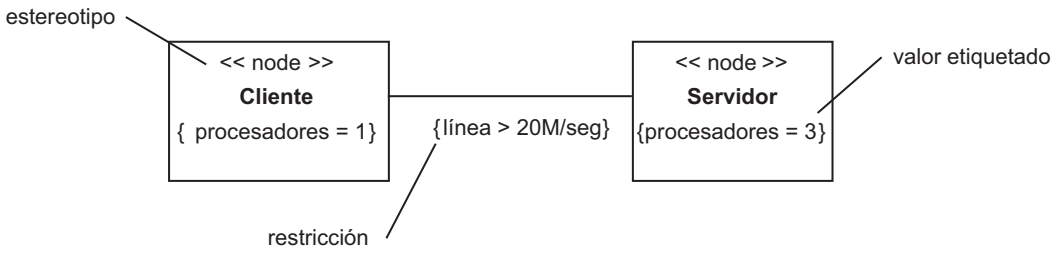

Figura 6.20: Mecanismos de extensión de UML: estereotipos, valores etiquetados y restricciones.

modelar nuestra aplicación web Struts. Por último, mostraremos los diagramas de clases de la aplicación.

6.2.2.1. Mecanismos de extensión de UML

Los mecanismos de extensión de UML incluyen:

- **Estereotipos** (*stereotypes*). Un estereotipo es una extensión del vocabulario de UML que permite crear nuevos tipos de bloques de construcción similares a los existentes, pero específicos del problema que se está modelando. Se representa gráficamente mediante un nombre entre comillas que va colocado sobre el nombre de otro elemento (Figura 6.20). Opcionalmente, el elemento con el estereotipo puede dibujarse con un nuevo icono asociado a ese estereotipo.

- **Valores etiquetados** (*tagged values*). Un valor etiquetado es una extensión de las propiedades de un elemento de UML, permitiendo añadir nueva información en la especificación de ese elemento. Se representa gráficamente como una cadena de caracteres entre llaves colocada debajo del nombre de otro elemento (Figura 6.20). Esta cadena incluye un nombre (etiqueta), un separador (el símbolo «=») y un valor (de la etiqueta). Se puede especificar sólo el valor si su significado no es ambiguo.

- **Restricciones** (*constraints*). Una restricción es una extensión de la semántica de un elemento de UML, que permite añadir nuevas reglas o modificar las existentes. Las restricciones se pueden escribir como texto libre o con el lenguaje OCL (*Objetc Constraint Language*, Lenguaje de Restricciones de Objetos). Se representa gráficamente como una cadena de caracteres entre llaves colocada junto al elemento al que está asociada o conectada a ese elemento o elementos por relaciones de dependencia (Figura 6.20). Otra alternativa es utilizar una nota para representar una restricción.

En conjunto, estos tres mecanismos de extensión permiten configurar y extender UML según las necesidades de un proyecto específico.

Estereotipo	Se aplica al símbolo	Significado
page	clase	Especifica una página JSP.
action	clase	Especifica una clase Action.
link	asociación	Especifica una relación unidireccional entre una página JSP y otra página JSP u otro Action. Una asociación «link» es una abstracción de la etiqueta `<a>` de HTML.
forward	asociación	Especifica una relación unidireccional entre un Action y una página JSP u otro Action. Una asociación «redirect» es una abstracción de un *forward* de Struts donde el atributo `redirect` está a `false`.
redirect	asociación	Especifica una relación unidireccional entre una página JSP o un Action y otra página JSP u otro Action. Una asociación «redirect» es una abstracción de un *forward* de Struts donde el atributo `redirect` está a `true`.
submit	asociación	Especifica una relación unidireccional entre un formulario HTML y un Action.

Tabla 6.1: Estereotipos del perfil UML para Struts.

6.2.2.2. Perfil UML para Struts

Un perfil UML es un conjunto bien definido de estereotipos, valores etiquetados y restricciones que permite expresar los conceptos específicos de un determinado dominio de aplicación. En nuestro caso, este dominio es el de las aplicaciones web desarrolladas con Struts.

Antes de ponernos a crear desde cero un nuevo perfil para Struts, lo primero que hemos hecho ha sido buscar perfiles ya existentes que sean específicos para Struts. A día de hoy, no existe ningún perfil estándar para Struts. Sin embargo, hemos encontrado dos programas que poseen sus propios perfiles UML y que nos han ayudado a crear el nuestro:

- MagicDraw. http://www.magicdraw.com/.
- Mia Generation. http://www.mia-software.com/.

Aparte, existe una herramienta de modelado gráfico específica para Struts, Scioworks Camino[3], que posee un modelo propio llamado *storyboard*, que también hemos consultado.

El perfil que hemos utilizado para modelar la aplicación web Struts está formado por los estereotipos de la Tabla 6.1 y las restricciones de la Tabla 6.2.

6.2.2.3. Diagramas de clases de la aplicación

Entre las Figuras 6.21 y 6.38 se muestran los diagramas de clases de la aplicación.

[3]Disponible en http://www.scioworks.com/scioworks_camino.html.

Valor etiquetado	**Se aplica al estereotipo**	**Significado**
ActionForm	action	Especifica el ActionForm asociado al Action.
parametros	link	Especifica los parámetros HTTP que son pasados a la página destino. El valor de esta etiqueta está formado por los nombres de los parámetros separados por el símbolo «&».
nombre	forward	Especifica el nombre lógico que identifica al destino del *forward*.

Tabla 6.2: Valores etiquetados del perfil UML para Struts.

Como comentamos con anterioridad, los valores etiquetados van escritos entre llaves. La versión actual de Poseidon for UML, la herramienta de modelado que hemos utilizado, no muestra estos valores etiquetados. Por ello, para especificar los valores etiquetados de las clases hemos utilizado los atributos de las mismas, y para los de las asociaciones hemos usado su propio nombre.

Por simplificación, hemos omitido del modelo lo siguiente:

- El cambio de idioma de las páginas.
- La navegación *breadcrumb* (migas de pan), que permite al usuario orientarse dentro del sitio web por el cual discurre su navegación. Un ejemplo de *breadcrumb* podría ser:

  Inicio >Sección 1 >Subsección A >Productos >Ficha de Producto

- En toda la zona de administración, el enlace que hay en las páginas que permite volver a las anteriores.
- La página de error que muestra los fallos de sistema (error de conexión a la base de datos, etc.)

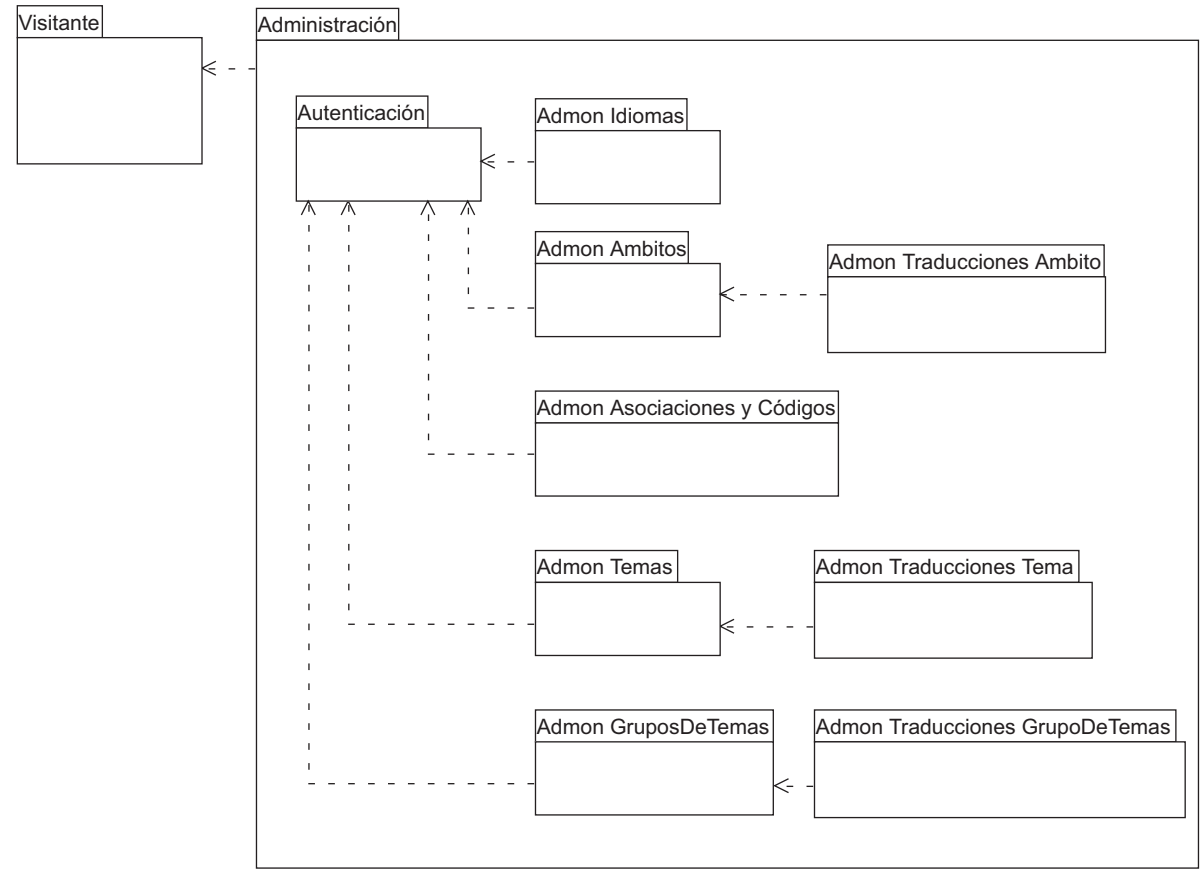

Figura 6.21: Diagrama de clases de la aplicación.

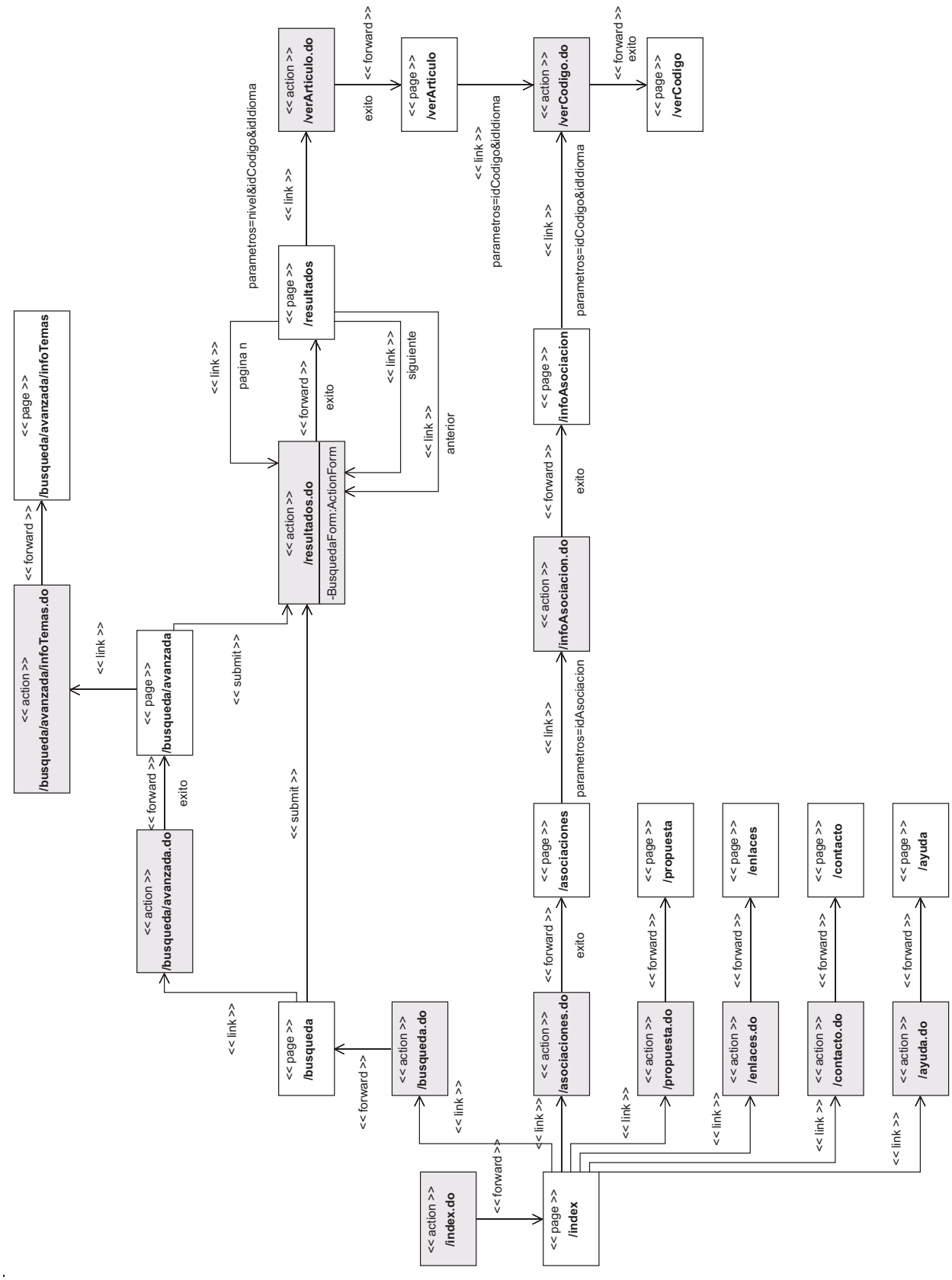

Figura 6.22: Diagrama de clases del paquete Visitante.

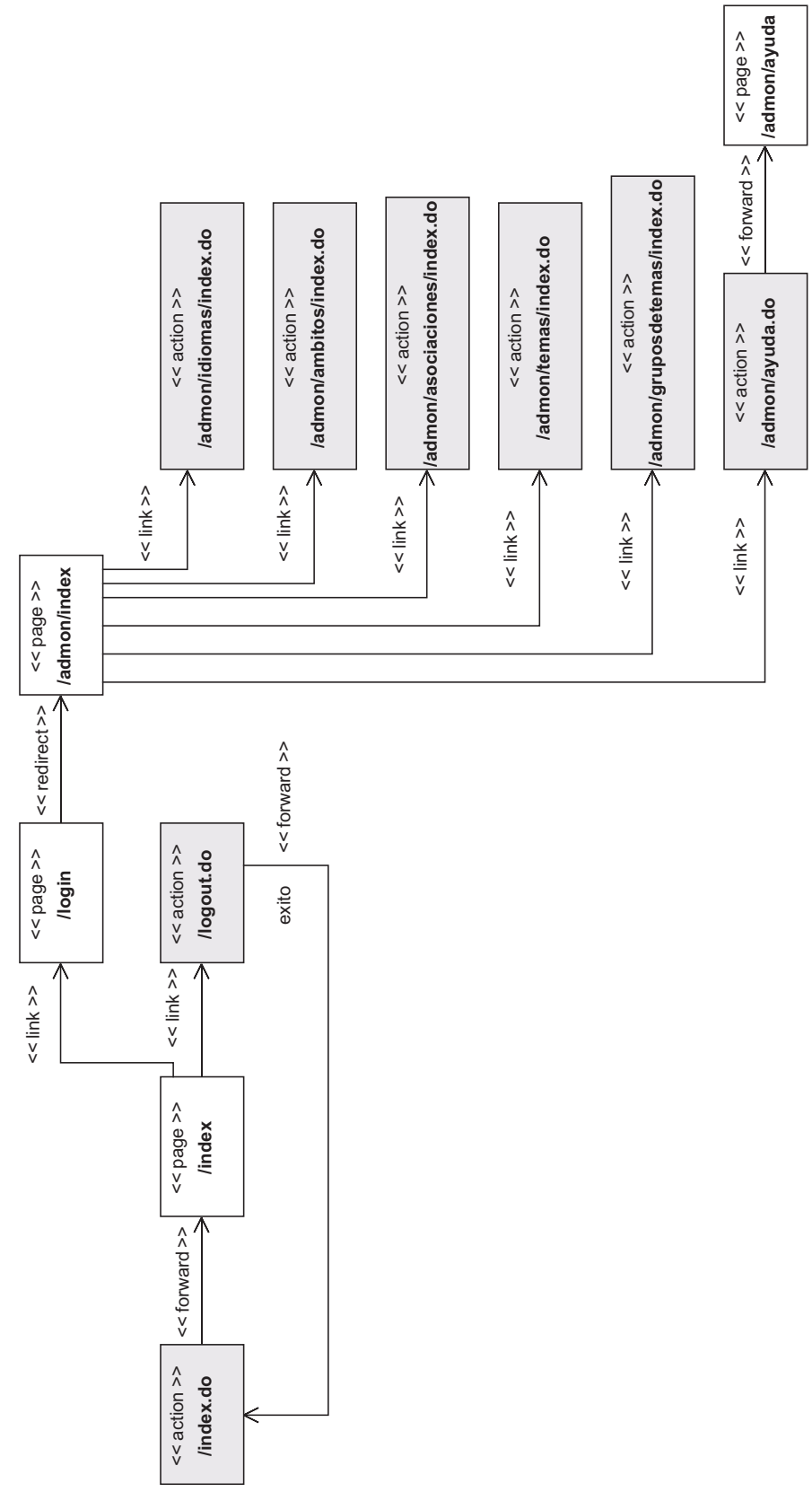

Figura 6.23: Diagrama de clases del paquete Autenticación.

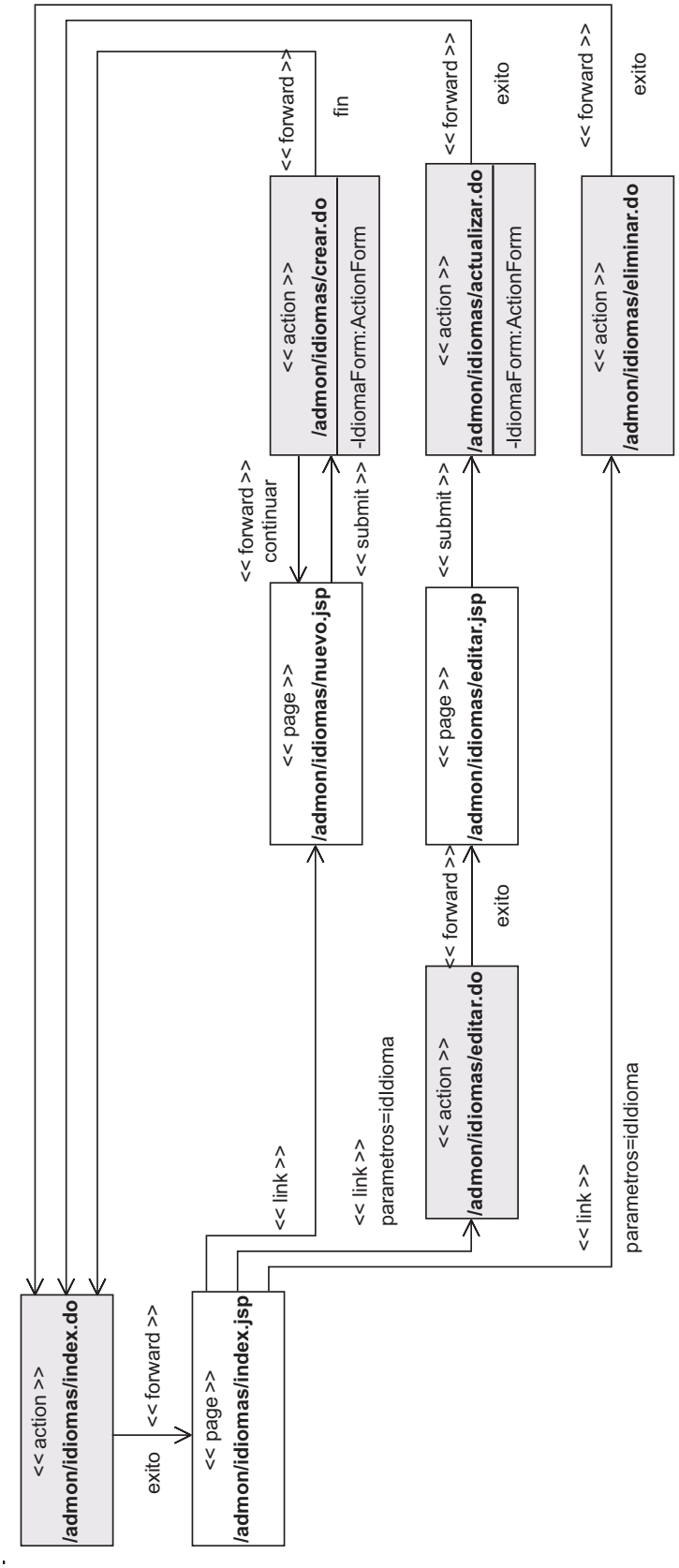

Figura 6.24: Diagrama de clases del paquete Admon Idiomas.

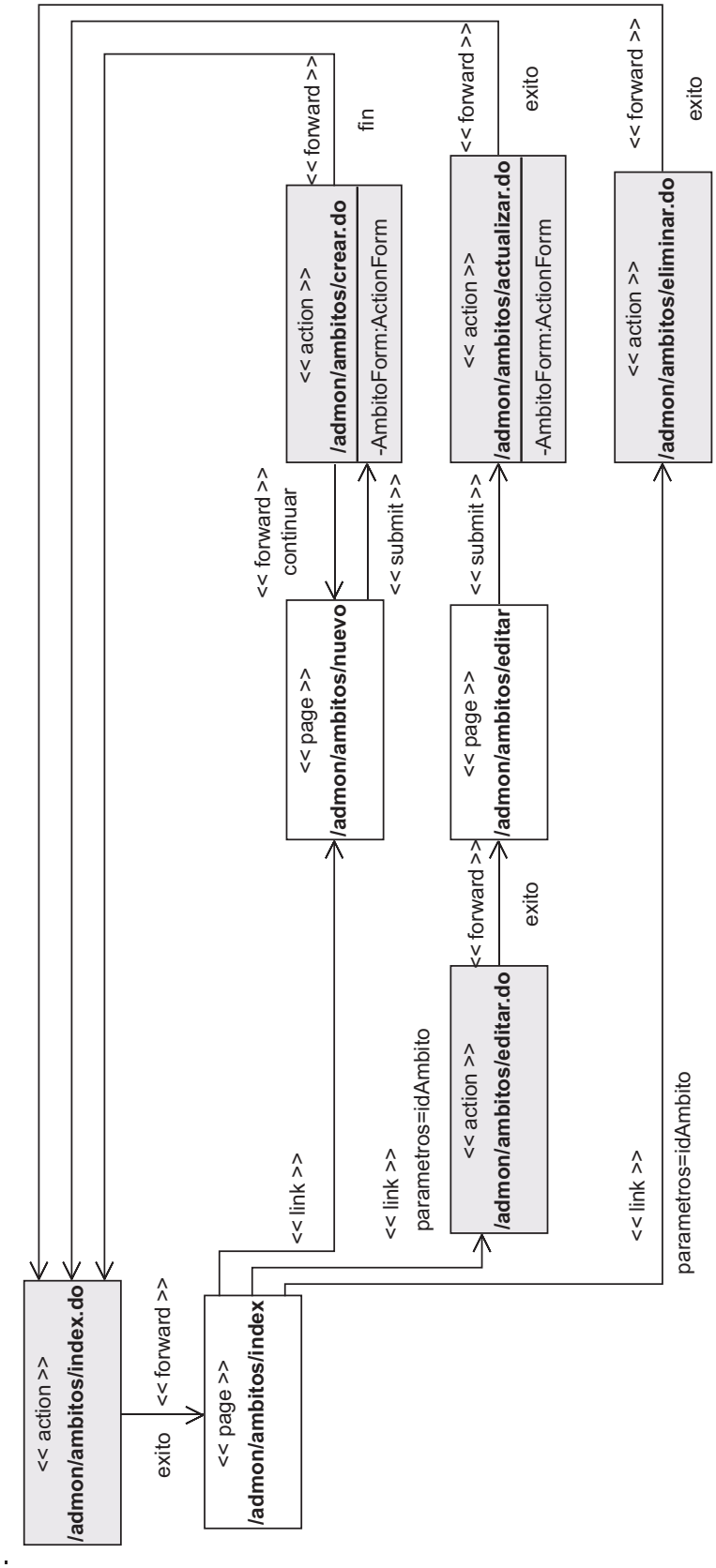

Figura 6.25: Diagrama de clases del paquete Admon Ambitos.

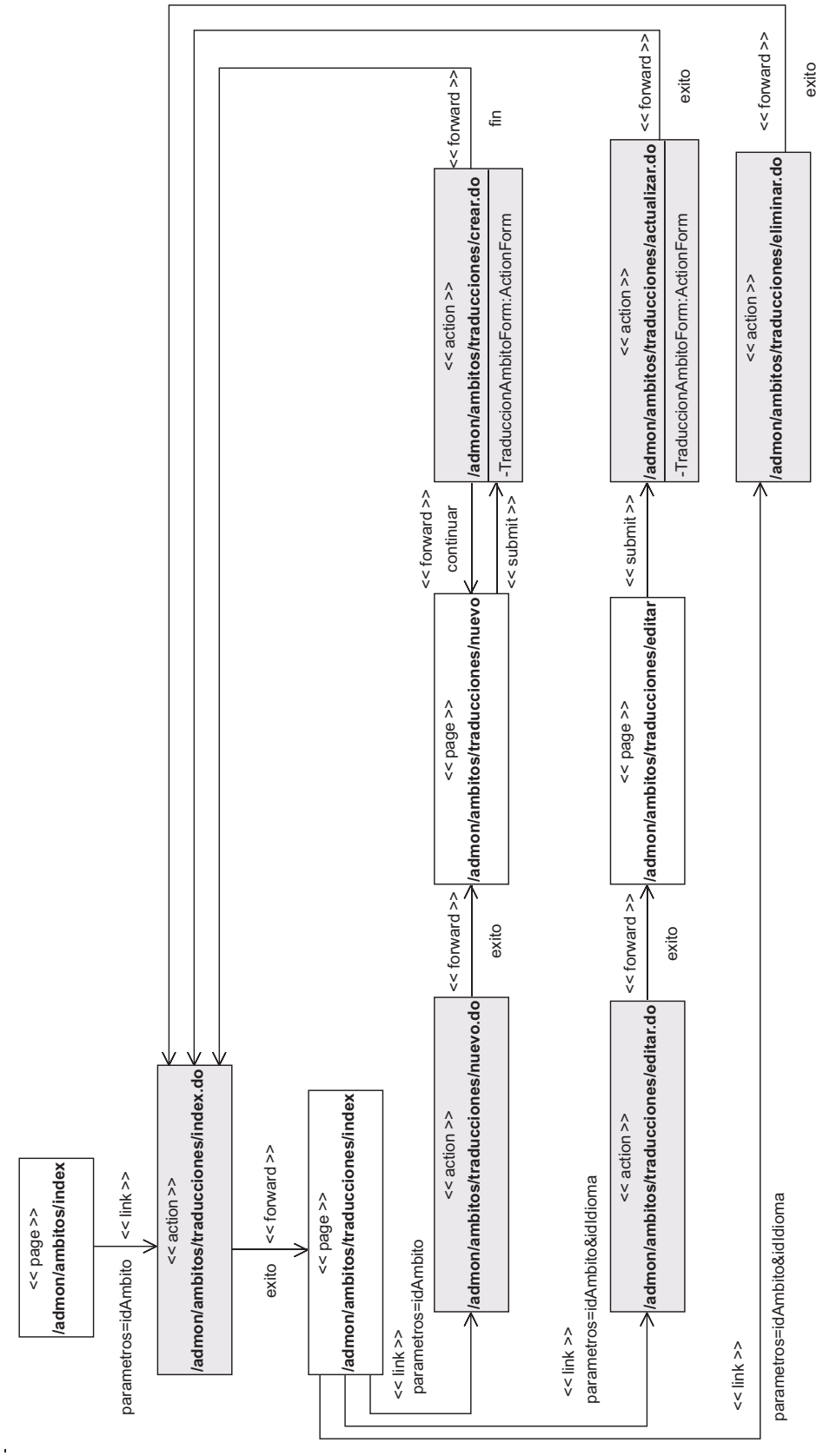

Figura 6.26: Diagrama de clases del paquete Admon Traducciones Ambito.

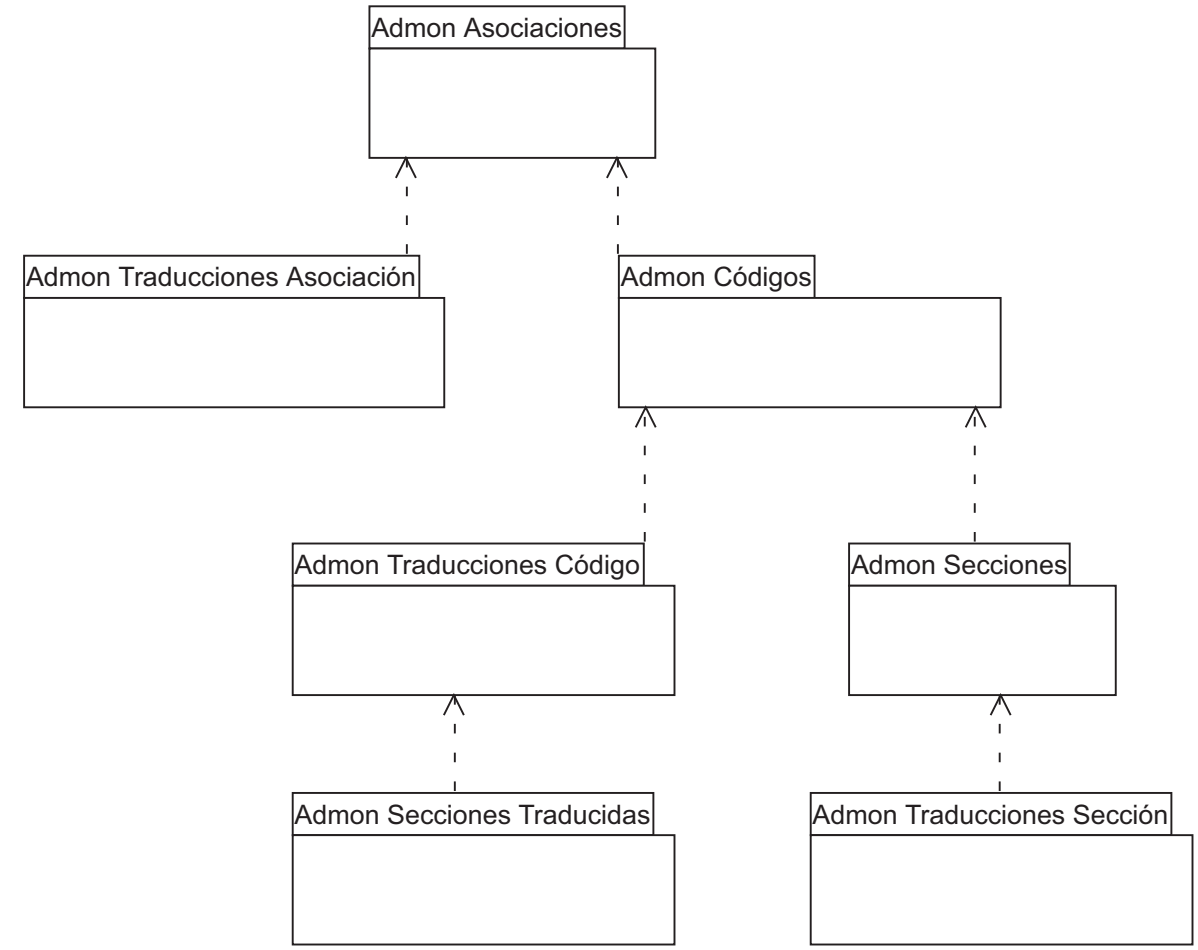

Figura 6.27: Diagrama de clases del paquete Admon Asociaciones y Códigos.

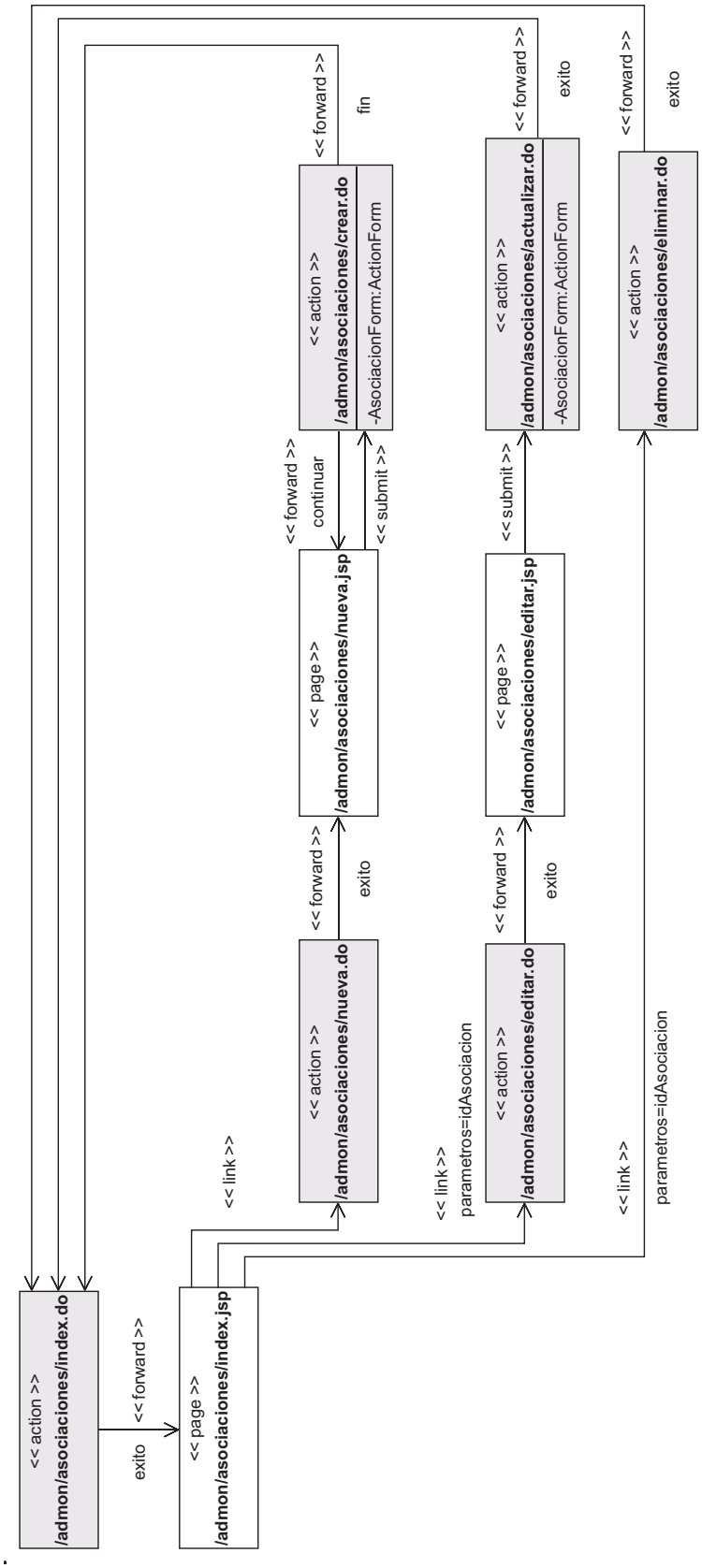

Figura 6.28: Diagrama de clases del paquete Admon Asociaciones.

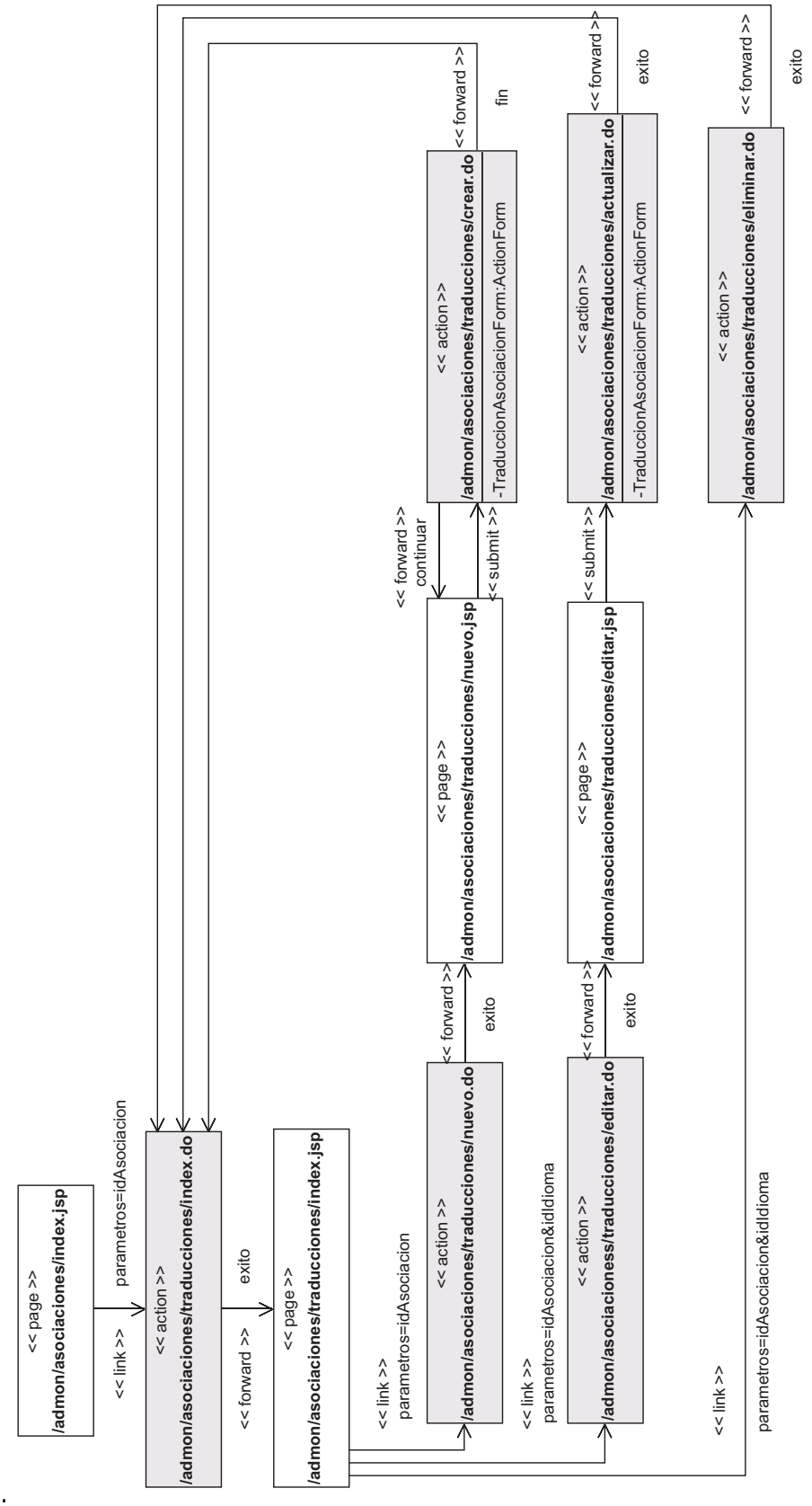

Figura 6.29: Diagrama de clases del paquete Admon Traducciones Asociación.

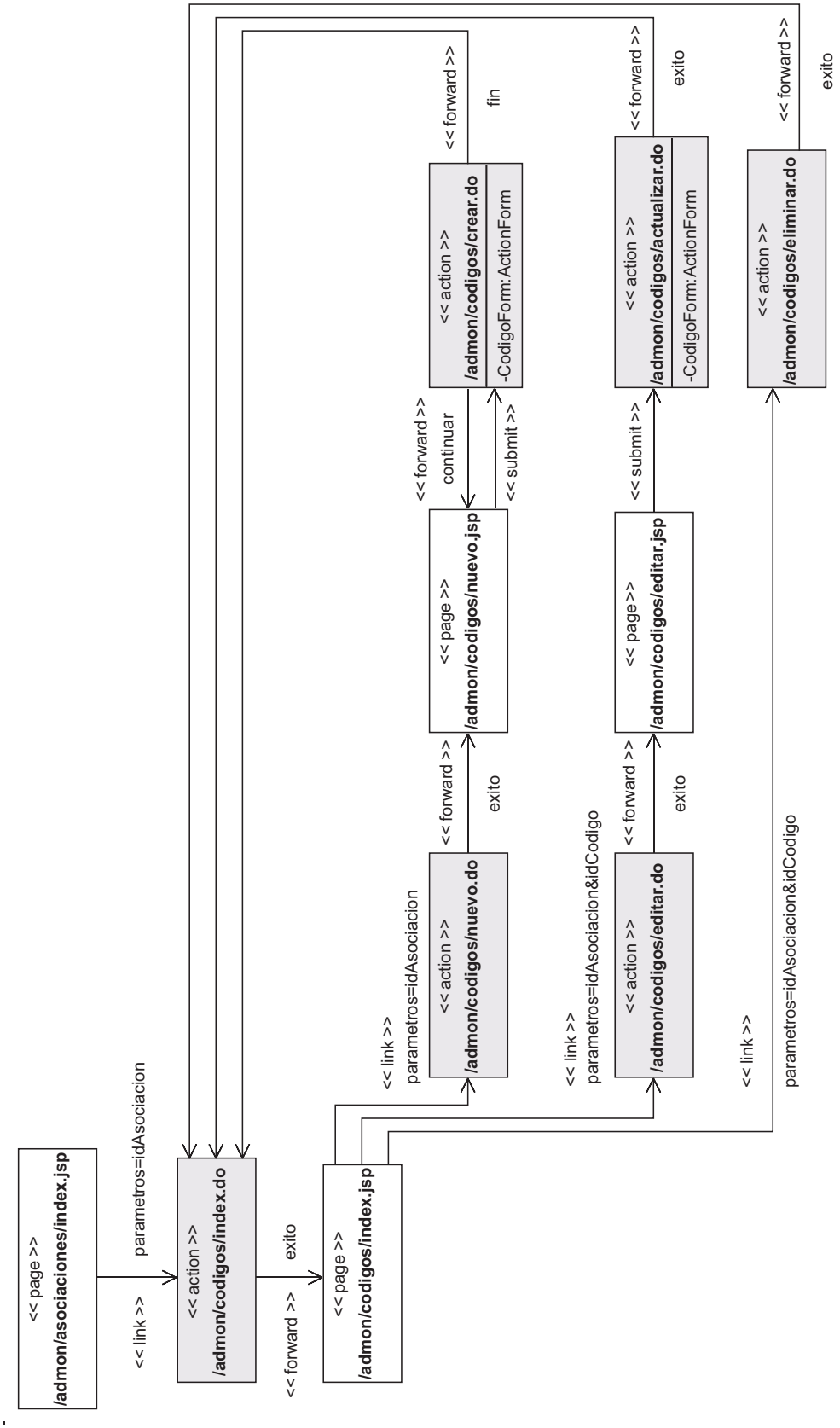

Figura 6.30: Diagrama de clases del paquete Admon Códigos.

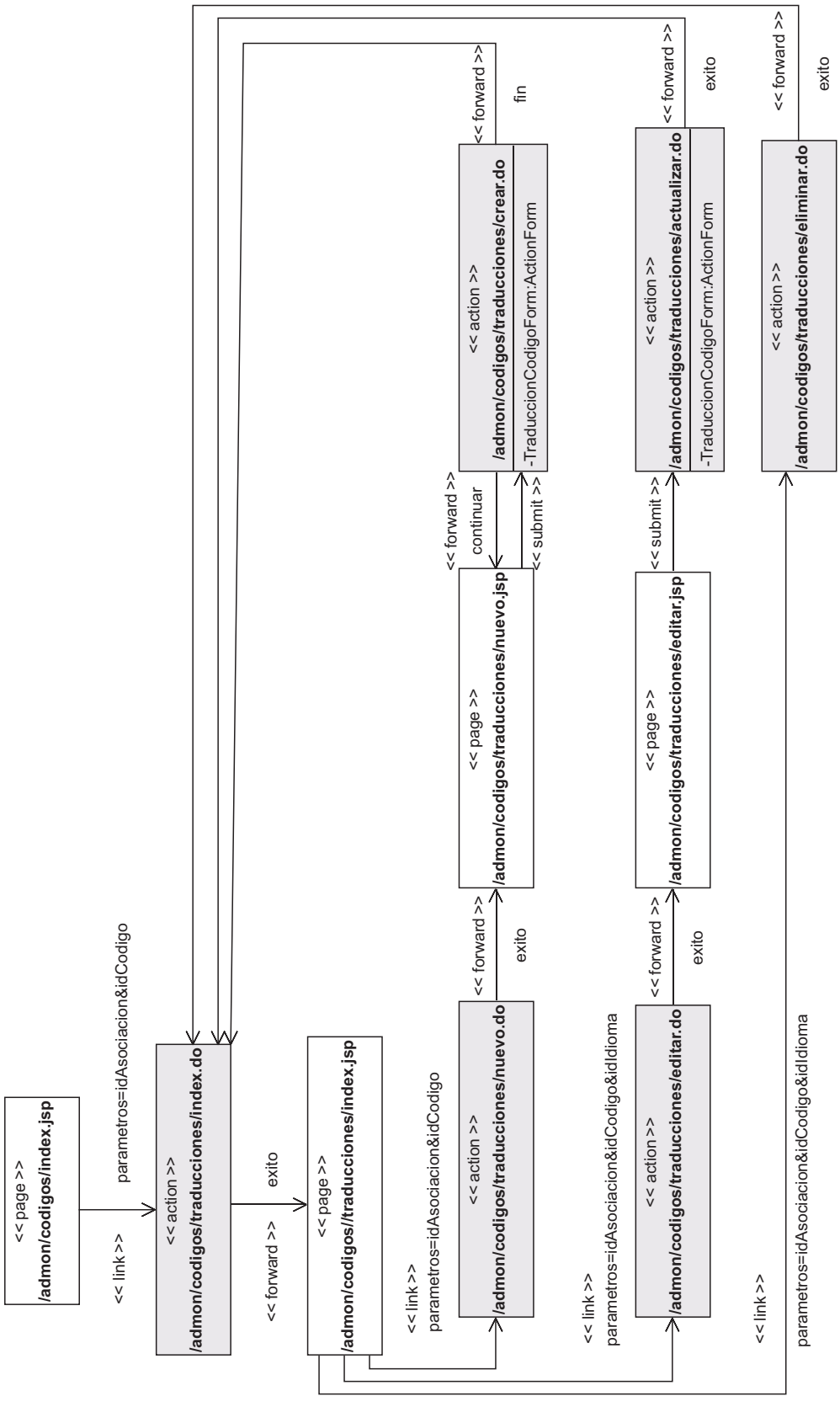

Figura 6.31: Diagrama de clases del paquete Admon Traducciones Código.

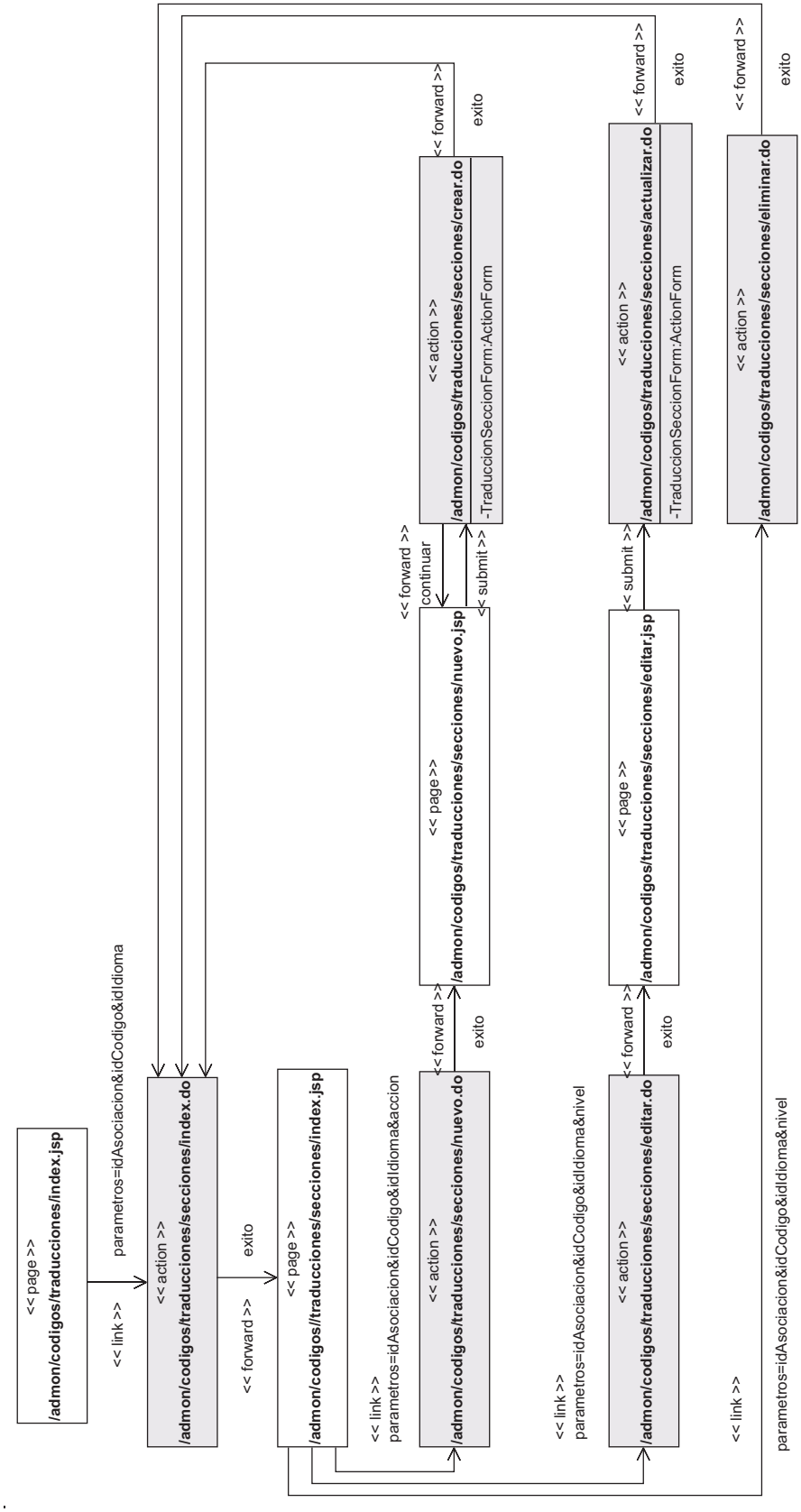

Figura 6.32: Diagrama de clases del paquete Admon Secciones Traducidas.

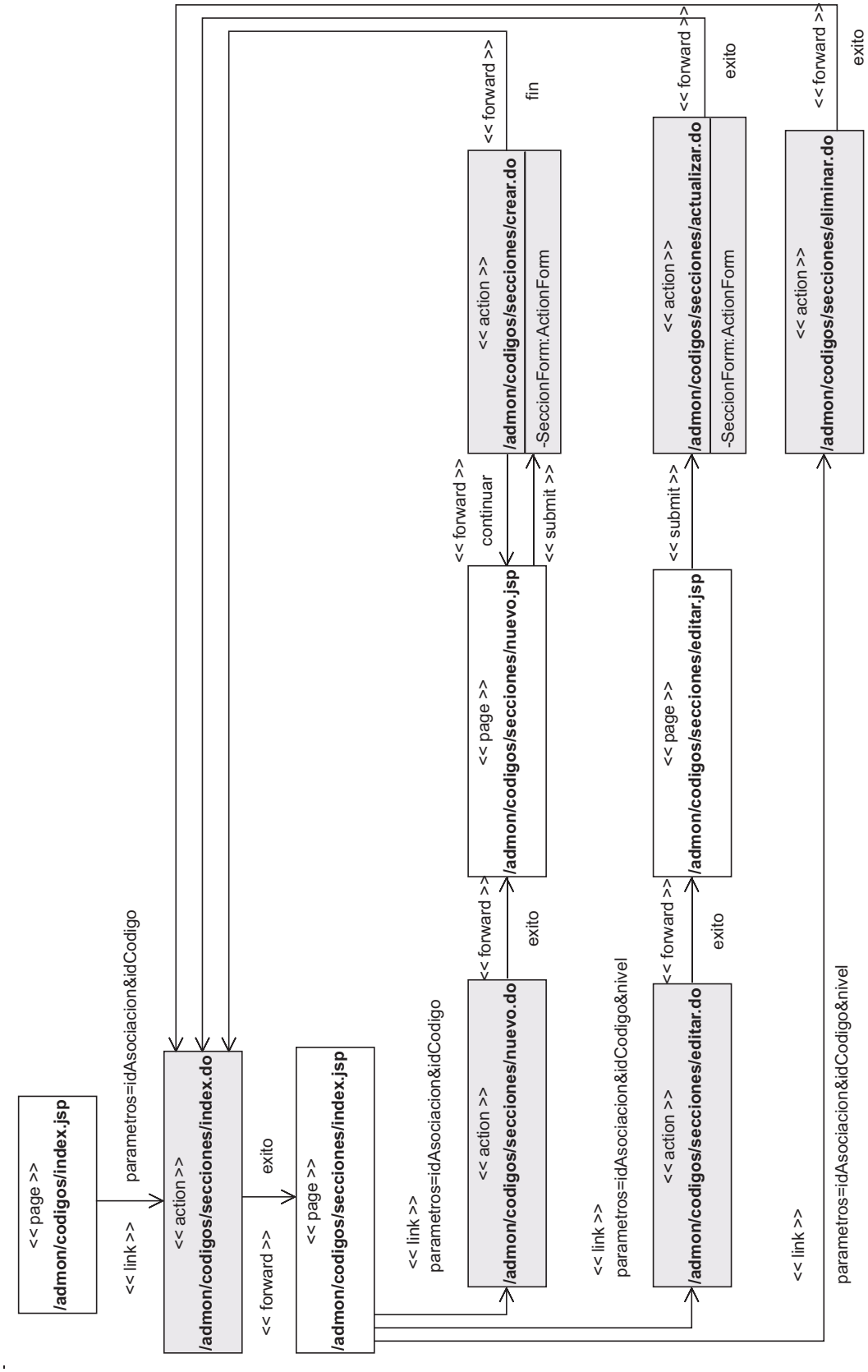

Figura 6.33: Diagrama de clases del paquete Admon Secciones.

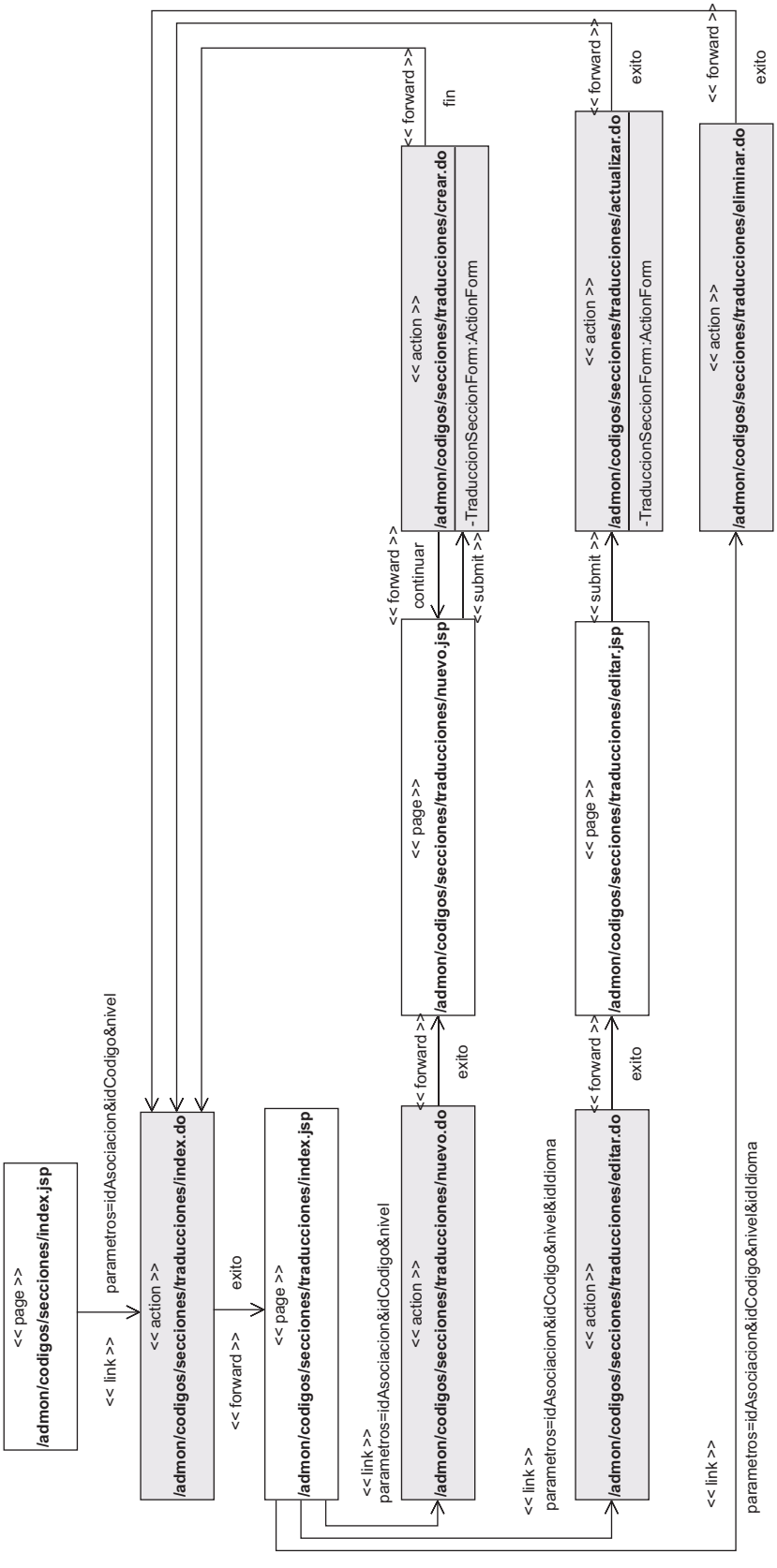

Figura 6.34: Diagrama de clases del paquete Admon Traducciones Sección.

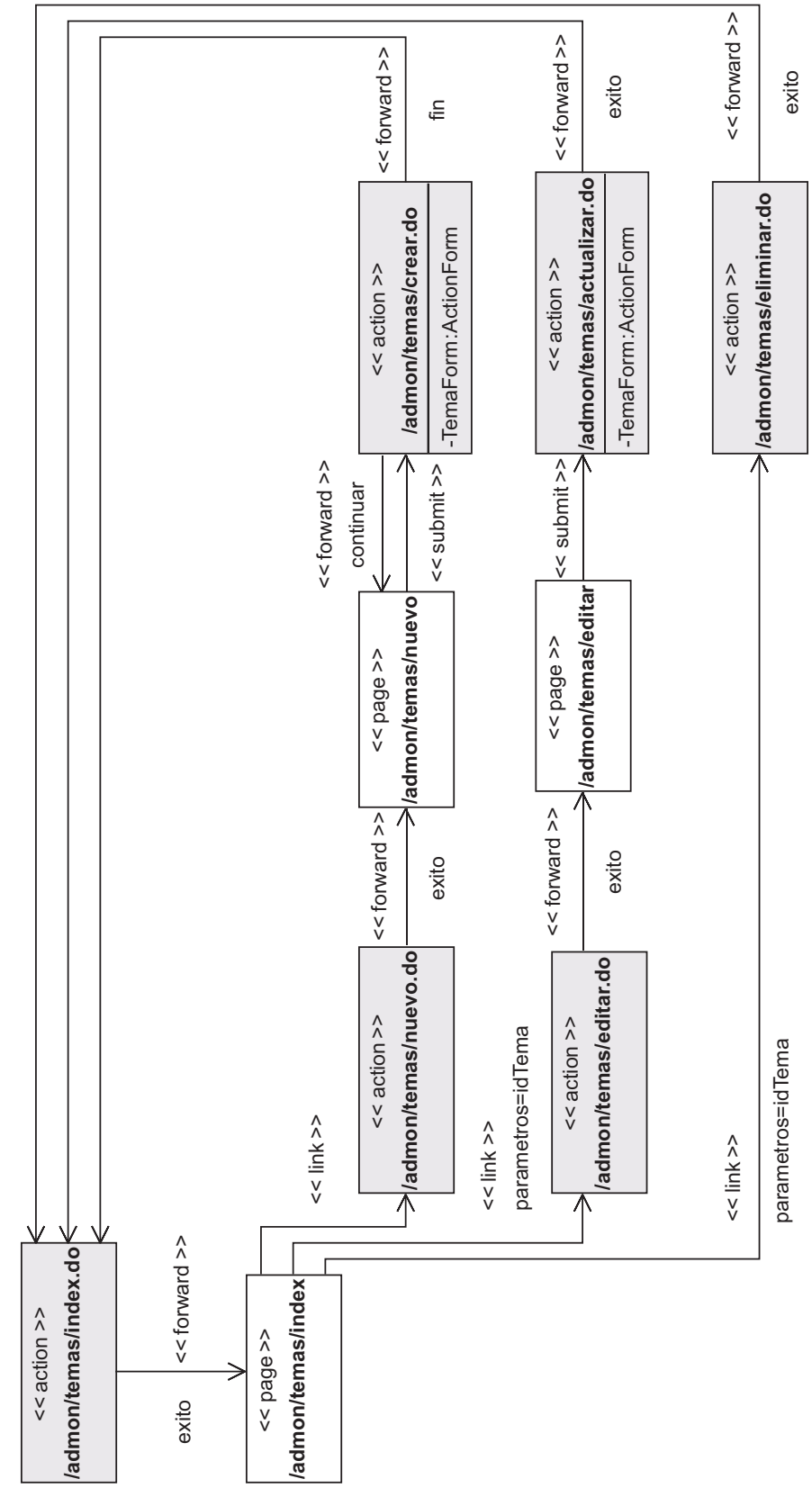

Figura 6.35: Diagrama de clases del paquete Admon Temas.

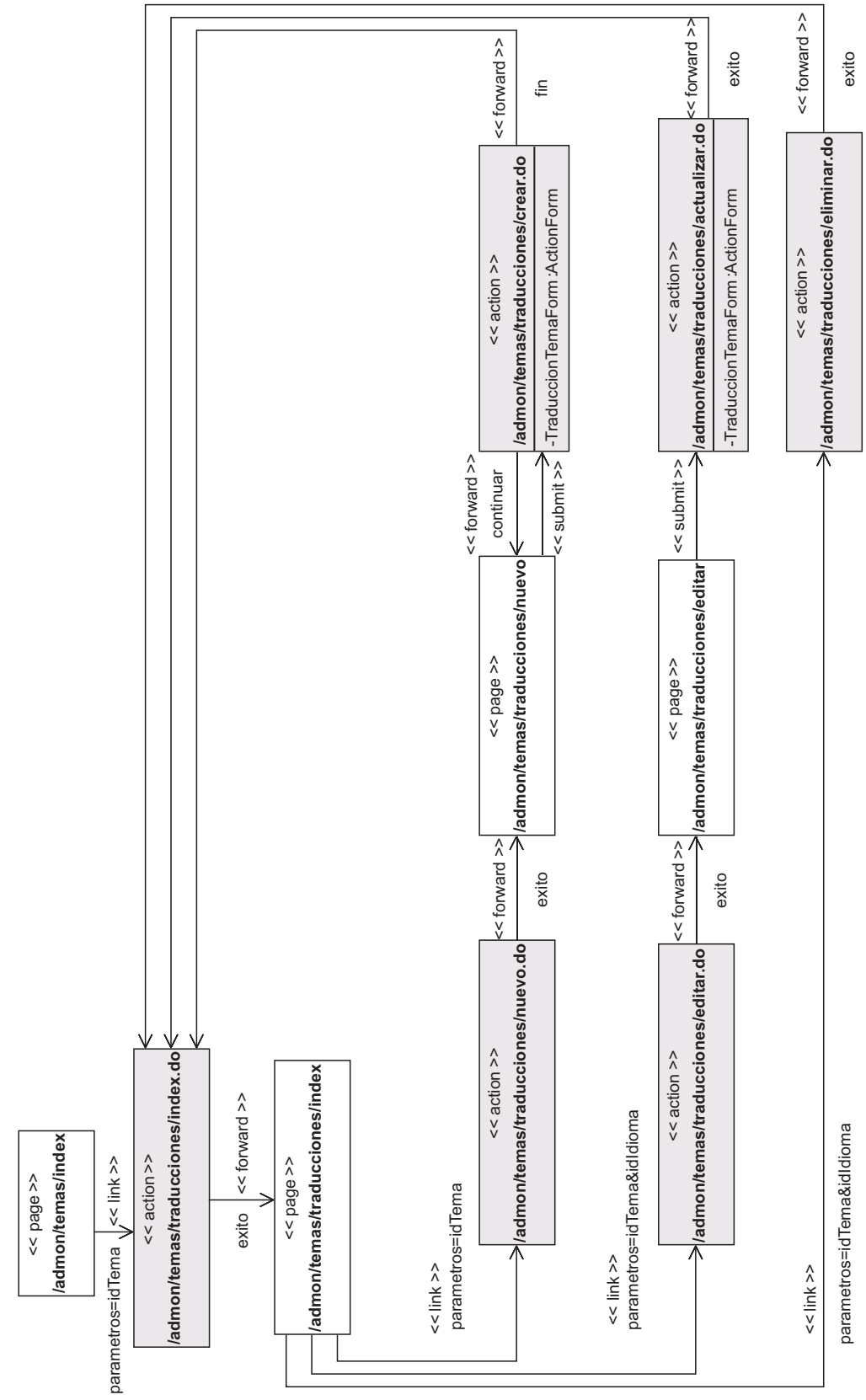

Figura 6.36: Diagrama de clases del paquete Admon Traducciones Tema.

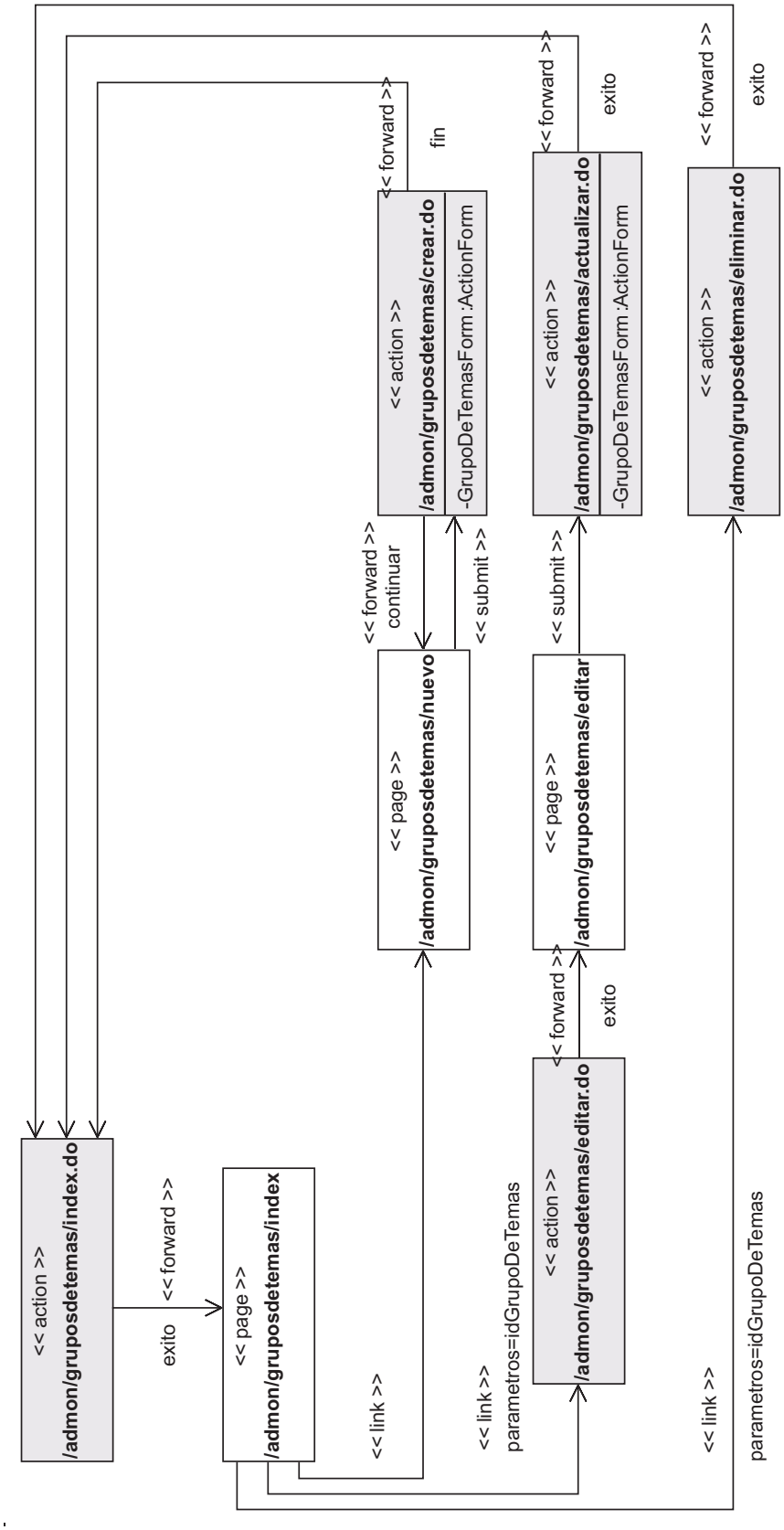

Figura 6.37: Diagrama de clases del paquete Admon GruposDeTemas.

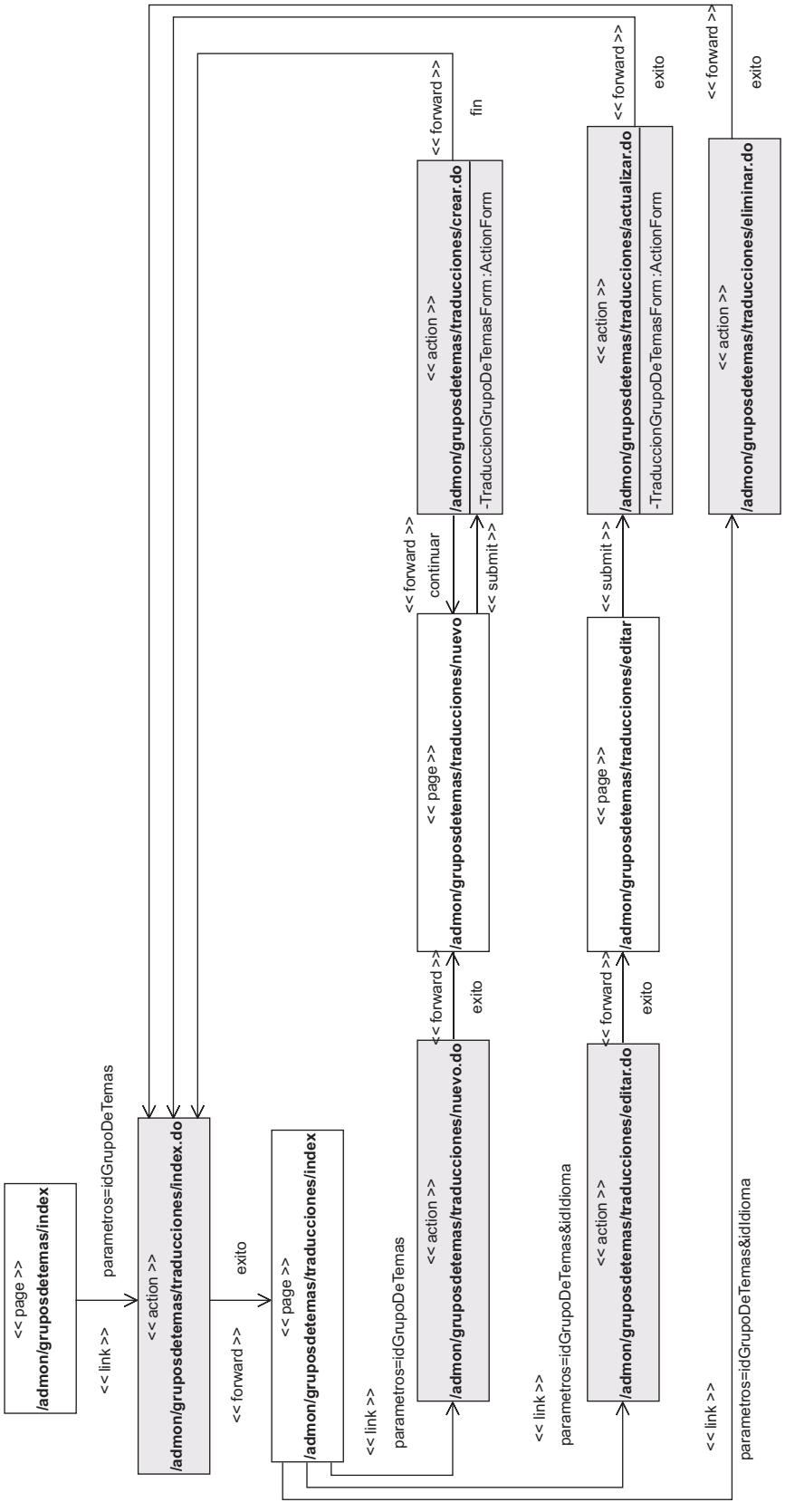

Figura 6.38: Diagrama de clases del paquete Admon Traducciones GruposDeTemas.

6.3. Implementación

Para el desarrollo de nuestra aplicación hemos tomado como punto de partida la aplicación `struts-blank.war` que viene con la versión 1.2.7 de Struts. Otra aplicación que puede servir como base para el desarrollo de una aplicación J2EE es AppFuse[4]. Esta es especialmente útil cuando queremos utilizar varios *frameworks* (como Hibernate, Spring, Struts, Ant/XDoclet, iBATIS, etc.) en un mismo desarrollo, ya que nos facilita la integración de todas estas tecnologías. También existe una versión más simple de AppFuse llamada Equinox[5].

Para comenzar, explicaremos cómo hemos implementado la seguridad de nuestra aplicación. Después hablaremos de los `ActionForms`, así como de los Actions y las páginas JSP de la aplicación. A continuación, hablaremos de la internacionalización de la aplicación, para, en último lugar, comentar brevemente la forma en que la hemos distribuido.

6.3.1. Seguridad

En nuestra aplicación necesitamos restringir el acceso a la zona de administración. Para ello hemos optado por utilizar las opciones de seguridad que nos ofrecen los servidores de aplicaciones J2EE (*Java 2 Platform, Enterprise Edition*). La tecnología servlet nos permite que la autenticación (proceso por el cual el usuario se identifica en una aplicación) y la autorización de recursos (proceso por el cual se autoriza al usuario identificado a acceder a determinados recursos de la misma) sea gestionada por el contenedor servlet, por ejemplo Tomcat, de manera que nos evita el tener que programar nosotros mismos la seguridad de nuestra aplicación.

La especificación J2EE define un mecanismo de seguridad en el que la seguridad de la aplicación es definida de forma declarativa en archivos de configuración. Para configurar correctamente esta seguridad declarativa se deben seguir cuatro pasos:

1. Definir los roles. El primer paso el definir los roles. Los roles representan un conjunto de permisos que se pueden asociar a un cierto grupo de usuarios. En nuestro caso, nosotros hemos definido un rol, llamado `codeticAdmin`, para los administradores del sistema.

2. Definir los reinos de seguridad (*realms*). El segundo paso es definir un reino de seguridad en el archivo de configuración `server.xml` del contenedor web. Un reino de seguridad identifica a un conjunto de usuarios, a sus contraseñas y a sus roles asociados. Existen cuatro tipos de reinos: reino basado en archivos (`UserDatabaseRealm`), reino JDBC (`JDBCRealm`), reino JNDI (`JNDIRealm`) y reino JASS (`JAASRealm`). Nosotros nos hemos decantado por el reino `UserDatabaseRealm`. Este tipo de reino se define en el archivo de configuración de usuarios del contenedor web. Para Tomcat, este archivo es `tomcat-users.xml`.

[4]Sitio web oficial de AppFuse: https://appfuse.dev.java.net
[5]Sitio web oficial de Equinox: https://equinox.dev.java.net

3. Proteger los recursos. El tercer paso es declarar los recursos de la aplicación que queremos proteger. Para ello se especifica en el archivo `web.xml` qué roles deben tener los usuarios para poder acceder a dichos recursos. Cuando se utiliza Struts, estas declaraciones se pueden hacer en dos sitios: en el archivo `struts-config.xml`, para proteger a las Actions, y en el archivo `web.xml`, para proteger otros recursos, como páginas JSP, páginas HTML o imágenes.

4. Definir los métodos de autenticación. El último paso es definir el mecanismo de autenticación. Cuando un usuario intenta acceder a un recurso protegido, el contenedor determina si el usuario ha sido autenticado. Si no, entonces lleva a cabo uno de las cuatro posibles métodos de autenticación: básico (`BASIC`), compendio (`DIGEST`), basado en formularios (`FORM`) y certificado cliente HTTPS (`CLIENT-CERT`). Nosotros hemos utilizado el método basado en formularios, puesto que nos permite diseñar páginas personalizadas de inicio de sesión y de error.

Uno de los inconvenientes del modelo de autenticación gestionada por el contenedor web es que el usuario debe intentar acceder a un recurso protegido para poder identificarse. Este comportamiento dificulta que el usuario se identifique proactivamente. Un truco muy común, que es el que hemos usado, para permitir que los usuarios inicien una sesión proactivamente es crear un enlace desde una página no protegida a una página JSP protegida. Puesto que la página JSP está protegida, el usuario deberá identificarse. La página JSP protegida simplemente redirecciona hacia la página no protegida inicial.

6.3.2. `ActionForms`

En vez de crear una clase `ActionForm` para cada formulario de la aplicación, hemos utilizado formularios dinámicos (Apartado 5.5.4.2), que nos permiten definir los formularios de manera declarativa en el archivo `struts-config.xml` y utilizar como tipo de formulario la clase `DynaActionForm` o `DynaValidatorForm` en caso de que usemos el *framework* Validator (Apartado 5.5.4.1) para realizar la validación de los formularios.

Para la validación de formularios hemos utilizado el *framework* Validator. La validación siempre debe realizarse como mínimo en el servidor, por si el usuario no tiene activado JavaScript, y añadirse opcionalmente en el cliente, para evitar flujos innecesarios de información. En nuestra aplicación, realizaremos las validaciones sólo en el servidor.

A continuación mostramos una lista con todos los formularios de la aplicación. Todos ellos tienen como tipo la clase `org.apache.struts.validator.DynaValidatorForm`:

- `BusquedaForm`
- `IdiomaForm`
- `AmbitoForm`

- `AsociacionForm`
- `CodigoForm`
- `SeccionForm`
- `TemaForm`
- `GrupoDeTemasForm`
- `TraduccionAmbitoForm`
- `TraduccionAsociacionForm`
- `TraduccionCodigoForm`
- `TraduccionSeccionForm`
- `TraduccionTemaForm`
- `TraduccionGrupoDeTemasForm`
- `LocaleForm`

Si observamos, podemos ver que, en general, existe un formulario por cada una de las tablas de la base de datos. Los campos de cada formulario son los mismos que los de su tabla asociada. Sólo existen cuatro formularios que no se corresponden exactamente con alguna tabla de la base de datos:

- `BusquedaForm`: Formulario utilizado para establecer el criterio de las búsquedas de artículos.
- `SeccionForm`: Está asociado con la tabla `Seccion` de la base de datos. Tiene un campo adicional llamado `temas` que es un array con los identificadores de los temas asociados a la sección. Este campo se utiliza para crear, recuperar, modificar y eliminar filas de la tabla `Seccion_Tema`. Además, tiene otro campo denominado `nivelAntiguo` que se utiliza para guardar el nivel que tiene una sección antes de ser modificada.
- `TraduccionSeccionForm`: Está asociado con la tabla `TraduccionSeccion` de la base de datos. Tiene un campo adicional llamado `temas` que es un array con los identificadores de los temas asociados a la sección. Este campo se utiliza para crear, recuperar, modificar y eliminar filas de la tabla `Seccion_Tema`.
- `LocaleForm`: Formulario utilizado para cambiar el idioma.

Sólo hemos utilizado los tipos `String` y `Boolean` para definir los campos de los formularios. Es mejor no usar el resto de tipos porque pueden dar problemas ya que las propiedades del objeto `ActionForm` son establecidas antes de la validación. Por ejemplo, si un usuario introduce una palabra en un campo definido como tipo numérico, p.e. `java.lang.Integer`, entonces el formulario no puede ser rellenado y el método `BeanUtils.populate()` lanza una excepción en tiempo de ejecución. En cambio, si usamos campos de tipo `String`, el formulario puede ser completado sin problemas. Una vez completado, la validación se encarga de verificar se la información introducida en el formulario es correcta.

6.3.3. Actions

En este apartado vamos a ver la forma en que hemos organizado e implementado las clases Action. Como vimos, una clase Action es un adaptador entre el contenido de una petición de cliente y la lógica de negocio que debería ejecutarse para procesar dicho contenido. Por tanto, podemos pensar en cada Action como si fuera el pegamento que une la petición HTTP con la lógica de negocio asociada a dicha solicitud.

En nuestra aplicación, las clases Action son clases que delegan en objetos de negocio (*Business Object*, BO) para ejecutar la lógica de negocio. A su vez, esto objetos de negocio delegan en objetos de acceso a datos (*Data Access Object*, DAO) para acceder a la base de datos. El traspaso de información entre todas estas capas se llevará a cabo mediante los objetos de transferencia de datos (*Data Transfer Object*, DTO), que actúan como portadores de información.

A continuación, explicaremos el patrón de diseño DAO que especifica cómo acceder a una fuente de datos mediante objetos de acceso a datos. Después, presentaremos las excepciones Java creadas para nuestra aplicación. Finalmente, hablaremos de las clases DTO, las clases DAO, las clases BO y las clases Action de la aplicación.

6.3.3.1. El patrón de diseño DAO

Cuando en un desarrollo orientado a objetos se utiliza una base de datos relacional, hay que adaptar el modelo orientado a objetos de nuestra aplicación al esquema relacional de la base de datos. A esa adaptación se le denomina ORM (*Object-Relational Mapping*, Mapeo Objeto-Relacional) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos, y viceversa.

Existen varias soluciones ORM como Hibernate, iBATIS, JDO, etc., que proporcionan mecanismos de mapeo objeto/relacional para definir cómo se almacenan, eliminan, actualizan y recuperan los objetos Java. La opción que hemos elegido nosotros consiste en codificar «a mano» la adaptación objeto/relacional. Para ello, hemos utilizado el patrón de diseño DAO (*Data Access Object*, Objeto de Acceso a Datos). Este patrón nos permite separar la lógica de negocio de la lógica de acceso a datos. De esta manera, el objeto DAO abstrae y encapsula todos los accesos a la fuente de datos, que en nuestro caso será una base de datos relacional. El DAO

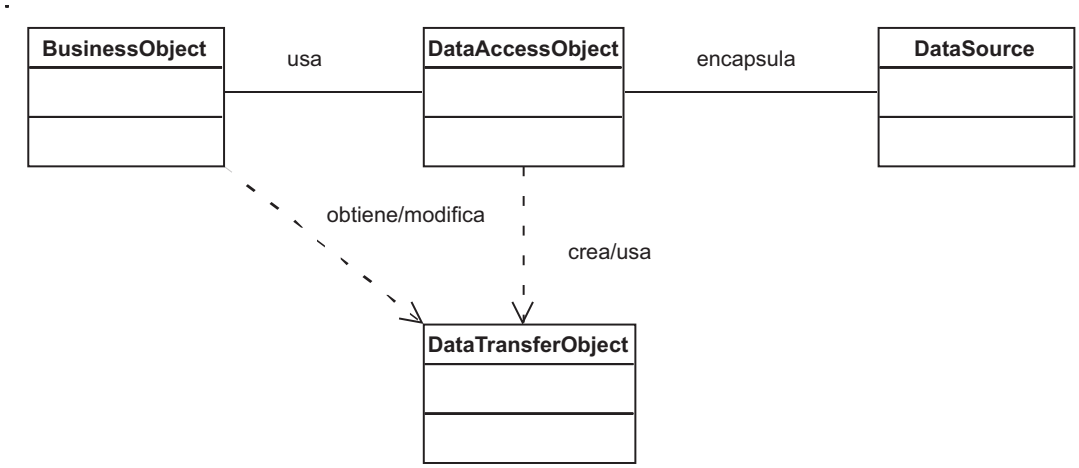

Figura 6.39: Diagrama de clases del patrón DAO.

maneja la conexión con la fuente de datos para obtener y almacenar datos, sin importar cómo esté implementada dicha fuente de datos. Por tanto, el principal beneficio del patrón DAO es definir un mecanismo de acceso a los datos transparente del tipo de almacenamiento.

En la Figura 6.39 podemos ver un diagrama de clases en el que se muestra cómo se relacionan los distintos elementos que forman parte del patrón DAO , mientras que la Figura 6.40 contiene el diagrama de secuencia que muestra cómo interactúan entre sí dichos elementos.

Los objetos que forman parte del patrón de diseño DAO son los siguientes:

- BusinessObject (BO). Este objeto abstrae las operaciones de negocio. Es el objeto que usa el DAO, ya que requiere acceder a la fuente de datos para almacenar y recuperar datos. No depende de la fuente de datos concreta.

- DataAccessObject (DAO). El objeto DataAccessObject es el más importante de todo el patrón. Abstrae las operaciones sobre la fuente de datos, proporcionando un API para acceder y manipular datos.

- DataSource. Este objeto representa la implementación de la fuente de datos. Una fuente de datos podría ser una base de datos relacional, una base de datos orientada a objetos, un fichero del sistema de archivos, etc.

- DataTransferObject (DTO). Es el objeto utilizado para transportar datos entre las distintas capas. Los objetos DAO utilizan objetos DTO para devolver los datos a la clase cliente. A su vez, un objeto DAO puede recibir datos del cliente a través de otro objeto DTO.

Para mayor información sobre este y otros patrones J2EE, puede consultar el catálogo de patrones J2EE[2].

6.3.3.2. Excepciones

El lenguaje Java define dos tipos de excepciones:

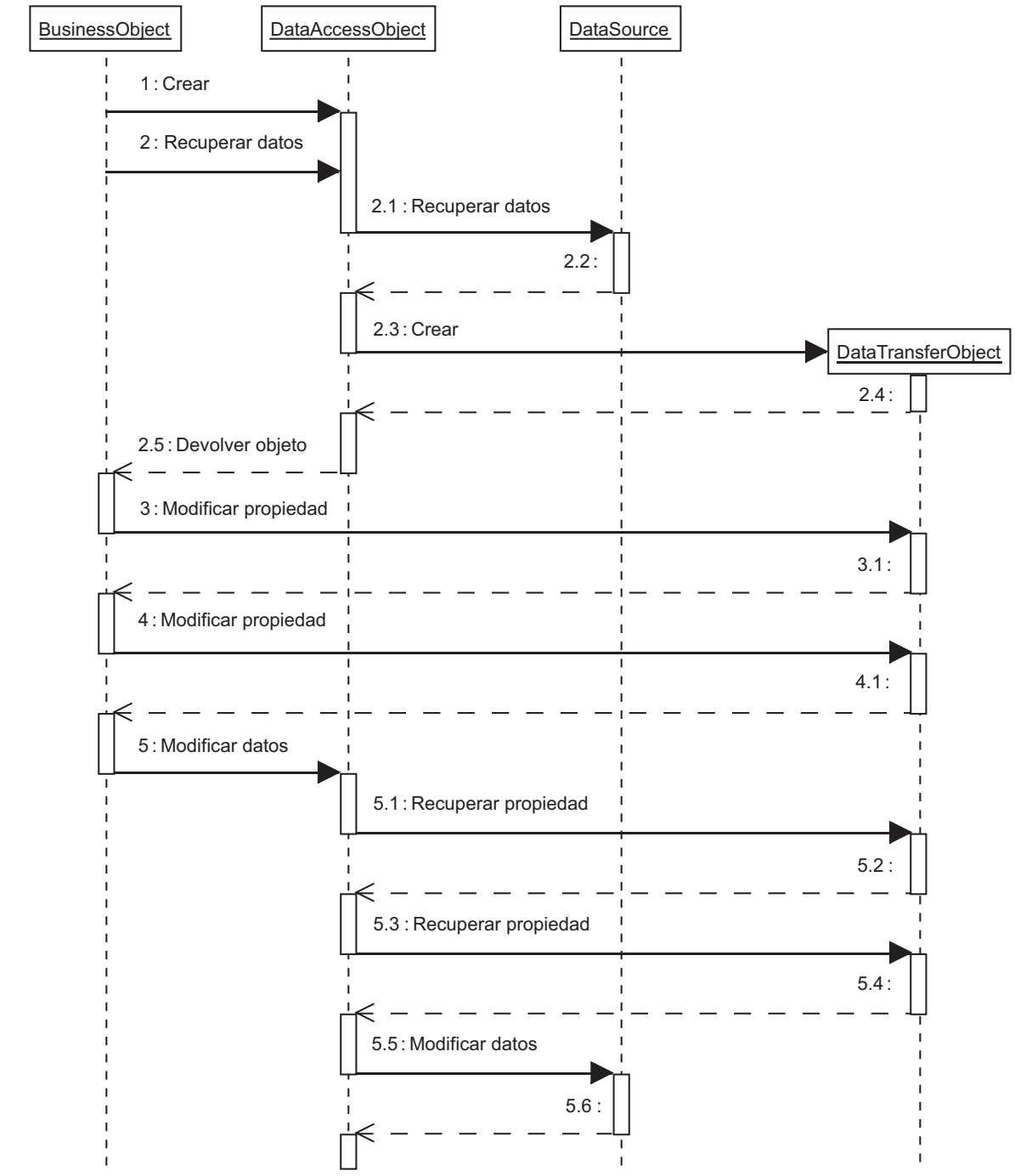

Figura 6.40: Diagrama de interacción del patrón DAO.

- Excepciones comprobadas (*checked exceptions*). Heredan de la clase `Exception` y deben ser manejadas obligatoriamente.
- Excepciones no comprobadas (*unchecked exceptions*). Heredan de la clase `RuntimeException` y no es necesario manejarlas.

Otra posible clasificación de las excepciones es la siguiente:

- Excepciones de sistema (*system exceptions*). Son excepciones críticas por naturaleza, normalmente debidas a problemas de bajo nivel (error al conectarse a una base de datos, etc.) que no están relacionados con la lógica de la aplicación, y que suelen ser irrecuperables. Por norma general, las excepciones del sistema son definidas como excepciones comprobadas puesto que no se espera que sean manejadas por la aplicación, ya que o bien no son errores de programación o bien son tan graves que no se puede hacer nada para recuperarse del error.
- Excepciones de aplicación (*application exceptions*). Son excepciones menos críticas que las excepciones de sistema y están relacionadas con la lógica de la aplicación. Estas excepciones ocurren cuando se infringe una regla de negocio o cualquier otra condición de la lógica de la aplicación. Por ejemplo, podríamos lanzar una excepción de aplicación cuando un usuario intente iniciar una sesión con una cuenta que está bloqueada. Aunque no es un error catastrófico, es un problema que se debe manejar y sobre el que se debe informar al usuario.

Visto todo esto, hemos decidido crear dos tipos de excepciones para nuestra aplicación: `DAOException` y `BusinessException`. Ambas excepciones son excepciones de aplicación que heredan de la clase `Exception`, por lo que deben ser manejadas a la fuerza. Estas excepciones encapsulan todos aquellos errores que deben ser comunicados a los usuarios finales de la aplicación: insertar un ámbito ya existente en la base de datos, intentar eliminar un idioma que no existe, etc.

Veamos brevemente que excepciones lanzan cada una de las capas de nuestra aplicación:

- Clases DAO. Lanzan excepciones `DAOException` para las excepciones de aplicación y excepciones `RuntimeException` para las excepciones de sistema.
- Clases BO. Lanzan excepciones `BusinessException` para las excepciones de aplicación y excepciones `RuntimeException` para las excepciones de sistema.
- Clases Action. No tienen código de manejo de excepciones.

Nuestra aplicación gestiona las excepciones de forma declarativa. Las excepciones `BusinessException` son mostradas al usuario final (mediante la etiqueta `<html:messages>` de la biblioteca `struts-html`). El resto de excepciones son tratadas por una página de error genérica que hemos creado para tal fin.

6.3.3.3. Clases DTO

Los objetos DTO se usan para transferir información entre las múltiples capas de una aplicación. Los objetos DAO pueden utilizar los DTO para devolver información al cliente. También pueden recibir información por parte del cliente en forma de objetos DTO y actualizar el contenido de la fuente de datos.

Hemos definido una clase DTO por cada tabla de la base de datos:

- `IdiomaDTO`
- `AmbitoDTO`
- `AsociacionDTO`
- `CodigoDTO`
- `SeccionDTO`
- `SeccionTemaDTO`
- `TemaDTO`
- `GrupoDeTemasDTO`
- `TraduccionAmbitoDTO`
- `TraduccionAsociacionDTO`
- `TraduccionCodigoDTO`
- `TraduccionSeccionDTO`
- `TraduccionTemaDTO`
- `TraduccionGrupoDeTemasDTO`

Aparte, hemos creado dos clases DTO especiales para las búsquedas:

- `BusquedaDTO`, que representa un criterio de búsqueda.
- `ResultadoDTO`, que representa el resultado de una búsqueda.

6.3.3.4. Clases DAO

Hemos creado un DAO por cada tabla de la base de datos. Cada clase DAO se encarga de la interacción con la base de datos:

- `IdiomaDAO`.
- `AmbitoDAO`.
- `AsociacionDAO`
- `CodigoDAO`
- `SeccionDAO`
- `SeccionTemaDAO`
- `TemaDAO`
- `GrupoDeTemasDAO`
- `TraduccionAmbitoDAO`
- `TraduccionAsociacionDAO`
- `TraduccionCodigoDAO`
- `TraduccionSeccionDAO`
- `TraduccionTemaDAO`
- `TraduccionGrupoDeTemasDAO`

Cada una de estas clases tiene los siguientes elementos:

- Un método para crear la entidad de la base de datos.
- Un método para eliminar la entidad de la base de datos.
- Un método para actualizar la información correspondiente a una entidad existente.
- Varios métodos para localizar entidades en la base de datos en función de criterios específicos. Como mínimo, cada DAO dispondrá de al menos un método para localizar la entidad a partir de su clave primaria, y otro método para recuperar todas las entidades de la tabla.

Aparte de los DAOs anteriores, hemos creado un DAO especial, llamado `BusquedaDAO`, que se comunica con varias tablas de la base de datos para realizar la búsqueda de artículos.

Todas estas clases usan el API JDBC con agrupamiento de conexiones para acceder a la base de datos MySQL. Se denomina agrupamiento de conexiones (*connection pool*) al manejo de una colección de conexiones abiertas a una base de datos de manera que puedan ser reutilizadas al realizar múltiples consultas o actualizaciones. De esta manera, el agrupamiento de conexiones permite hacer una gestión eficiente de los recursos disponibles en la base de datos.

También hemos creado una clase no extensible llamada `JDBCUtils` que contiene diversos métodos útiles para trabajar con JDBC, como, por ejemplo, obtener una conexión a la base de datos, cerrar una conexión, etc.

6.3.3.5. Clases BO

Los objetos BO son aquellos que contienen las reglas de negocio. Son los encargados de mantener las entidades del sistema, que, en nuestro caso, son las tablas de la base de datos. Los objetos BO hacen uso de los objetos DAO para obtener los datos que necesiten. En nuestra aplicación, tenemos una clase BO por cada clase DAO:

- `IdiomaBO`.
- `AmbitoBO`.
- `AsociacionBO`
- `CodigoBO`
- `SeccionBO`
- `SeccionTemaBO`
- `TemaBO`
- `GrupoDeTemasBO`
- `TraduccionAmbitoBO`
- `TraduccionAsociacionBO`
- `TraduccionCodigoBO`
- `TraduccionSeccionBO`
- `TraduccionTemaBO`
- `TraduccionGrupoDeTemasBO`
- `BusquedaBO`

6.3.3.6. Clases Action

Las clases Action se encargan de procesar las solicitudes del usuario. Para ello, hacen uso de los objetos BO, que son los que realmente llevan a cabo la lógica de negocio. Al contrario que con el resto de clases, no las vamos a listar, ya que su número es muy elevado.

Por último, decir que tenemos una clase no extensible llamada `Ctes` que recoge todas las constantes utilizadas en la aplicación.

6.3.4. Páginas JSP

Existen varias formas de crear páginas JSP. Nosotros las hemos creado utilizando el *framework* Tiles, un motor de plantillas dinámicas que nos permite modular el diseño de las páginas JSP. Este *framework* forma parte de las bibliotecas de etiquetas de Struts. Otras bibliotecas de etiquetas que hemos utilizado son las siguientes:

- Struts. La biblioteca de etiquetas que viene con Struts. Esta biblioteca permite manipular *beans*, controlar el flujo de ejecución dentro de una página JSP, etc.
- JSTL. La biblioteca de etiquetas estándar de JSP, que implementa funciones de uso frecuente en aplicaciones JSP.
- DisplayTag. Es una potente biblioteca de etiquetas de fuente abierta que permite visualizar listas de objetos en forma de tablas. Con ella, también podemos ordenar, paginar, exportar los datos de las tablas en distintos formatos (CSV, Excel, XML), agrupar los datos por columnas, etc. Esta biblioteca la hemos utilizado en la zona de administración, para mostrar los idiomas, ámbitos, etc. de la base de datos Codetic.
- PagerTag. Implementa la paginación de listas de objetos. La paginación nos permite mostrar los objetos de una lista poco a poco, es decir, en distintas páginas. Así, podemos mostrar los 50 primeros objetos, después los siguientes 50, etc. Esta biblioteca la hemos utilizado para paginar los resultados de las búsquedas.
- Text2Html. La biblioteca Text2Html la hemos creado para el proyecto. Permite convertir texto en HTML de una forma muy simple, separando en párrafos el texto y encerrándolos entre etiquetas `<p>` y `</p>`.

Otras tecnologías que hemos empleado han sido:

- Las hojas de estilo CSS. Con ellas hemos diseñado y maquetado las páginas web. También hemos controlado la impresión de las páginas web.
- JavaScript. Usado para confirmar la eliminación de registros de la base de datos y para mostrar/ocultar los menús desplegables del formulario de búsqueda avanzada.

Por último, decir que hemos comprobado que la aplicación se visualiza correctamente en los navegadores:

- Internet Explorer 6
- Mozilla Firefox 1
- Opera 8
- Netscape Browser 8

para las resoluciones de pantalla de 800x600 y 1024x768. Estos son los navegadores y las resoluciones de pantalla que más se utilizan en Internet, como podemos ver en diversas estadísticas de uso de navegadores [13, 72].

6.3.5. Internacionalización

Uno de los objetivos del proyecto era crear una aplicación web en español en la que poder consultar códigos deontológicos. Finalmente, decidimos internacionalizar la aplicación, posibilitando visualizarla tanto en español como en inglés. Nuestra aplicación detecta automáticamente el idioma del navegador y permite el cambio de idioma de cualquier página, excepto el de la página de resultados de las búsquedas.

Al usar UTF-8, una codificación de caracteres UNICODE que contiene los alfabetos de la mayor parte de las lenguas del mundo, podríamos ofrecer la web en árabe, chino, etc., sin importar el alfabeto del idioma. Básicamente, lo único que tendríamos que hacer para añadir un nuevo idioma sería crear un archivo de recursos (`MessageResources`) con los mensajes traducidos a dicho idioma.

La configuración de la internacionalización UTF-8 fue algo problemática debido a que existe poca información y esta es bastante dispersa. En especial, fueron dos las cosas que más trabajo nos costó encontrar para conseguir que nuestra aplicación utilizara UTF-8:

- Debemos usar el filtro `SetCharacterEncodingFilter` proporcionado por Tomcat. Este filtro permite que la aplicación acepte cualquier tipo de codificación de caracteres. Nosotros lo hemos tenido que configurar para que utilice UTF-8.
- Debemos utilizar UTF-8 como juego de caracteres de nuestra base de datos.

6.3.6. Despliegue

Para el despliegue, hemos empaquetado la aplicación web como un archivo WAR (*Web Application Resource*, *Web ARchive*). Un archivo WAR es un archivo comprimido que contiene todos los archivos de una aplicación web. De esta forma, lo único que hay que hacer para instalar la aplicación es copiar el fichero `codetic.war` del CD-ROM al directorio

`CATALINA_HOME\webapps`, donde `$CATALINA_HOME` es el directorio de instalación de Tomcat.

Capítulo 7

Manuales

Hemos creado tres manuales que documentan la aplicación web Codetic. Cada uno de ellos cubre distintos aspectos:

1. Manual de instalación. Este manual contiene las instrucciones necesarias para instalar la aplicación web Codetic.
2. Manual de usuario. Documentación de la parte pública de la aplicación web Codetic.
3. Manual de administración. Este manual cubre todos los aspectos de administración de la aplicación web Codetic.

7.1. Manual de instalación de Codetic

En esta sección se explica cómo instalar todo lo necesario para hacer funcionar nuestra aplicación web en un equipo Windows, aunque el proceso es muy similar para otras plataformas (Unix, Mac...). El *software* que hemos utilizado es de libre distribución, por lo que se ha incluido en el CD-ROM del proyecto:

- MySQL 5.0 Community Edition (Capítulo 4).
- MySQL Administrator 1.1 (Apartado 4.3.2).
- MySQL Query Browser 1.1 (Apartado 4.3.2).
- Apache Tomcat 5.0 (Apartado 3.2).

La sección se divide en los siguientes apartados, que se corresponden con los distintos pasos del proceso de instalación:

- Instalación del SGBD MySQL.
- Creación de la base de datos Codetic.

- Instalación del servidor web Tomcat.
- Instalación de la aplicación web Codetic.

7.1.1. Instalación del SGBD MySQL

Para la instalación del servidor de base de datos MySQL se debe descomprimir el fichero `mysql-5.0.15-win32.zip` que se adjunta en el CD-ROM, ejecutar el fichero `Setup.exe` y seguir todos los pasos de la instalación.

A partir de la versión 4.1.5 y posteriores del servidor MySQL, el proceso de instalación en Windows se complica un poco, puesto que se utiliza un nuevo instalador que incluye un asistente para la configuración del servidor. Por esto, vamos a mostrar cómo instalar y configurar paso por paso el servidor MySQL:

1. Una vez hemos iniciado la instalación, tras unos instantes, nos aparecerá la pantalla de bienvenida. Pulsaremos el botón *Next*.

2. Nos aparecerá una pantalla donde se debe seleccionar el tipo de instalación, en la que podremos elegir entre una instalación típica (*Typical*), completa (*Complete*) o a medida (*Custom*). Seleccionaremos la típica y pulsaremos el botón *Next*.

3. La siguiente ventana nos muestra una lista con todas las opciones de configuración. Pulsaremos en *Install*, con lo que se iniciará la instalación.

4. Una vez finalizada la instalación, nos aparecerá una ventana donde podemos registrarnos en MySQL.com para tener acceso a algunos servicios de su web (recibir el boletín electrónico mensual de MySQL...) o, si ya estamos registrados, introducir *email* de registro y contraseña. También podemos cancelar el registro.

5. A continuación veremos una pantalla que nos preguntará si queremos configurar MySQL en este momento. Dejaremos marcada la opción *Configure the MySQL Server now* y pulsaremos en *Finish*.

6. Ahora nos aparecerá un asistente para la configuración del servidor MySQL y pulsaremos en *Next*.

7. En la siguiente pantalla se debe seleccionar el tipo de configuración deseada: detallada (*Detailed Configuration*) o estándar (*Standard Configuration*). Seleccionaremos la opción de configuración detallada y pulsaremos en *Next*. De esta forma podremos configurar más opciones de MySQL utilizando el asistente. Si marcásemos la configuración estándar, el asistente nos pediría menos información pero habría que configurar algunas opciones manualmente.

8. La siguiente ventana nos da a elegir entre tres opciones. Dependiendo del uso que queramos dar al equipo en el que se instala marcaremos una de las tres:

   - *Developer Machine*: marcaremos esta opción si en el equipo donde hemos instalado el servidor MySQL se utiliza también para otras aplicaciones. El servidor MySQL utilizará la memoria mínima necesaria.
   - *Server Machine*: marcaremos esta opción si vamos a utilizar el equipo para algunas aplicaciones (no demasiadas). Con esta opción el servidor MySQL utilizará un nivel medio de memoria.
   - *Dedicated MySQL Server Machine*: marcaremos esta opción sólo si queremos utilizar el equipo como un servidor dedicado exclusivamente a MySQL. Con esta opción el servidor MySQL utilizará el máximo de memoria disponible. Se obtendrá un rendimiento elevado pero el equipo sólo servirá para MySQL.

   Una vez hayamos elegido una opción, pulsaremos en *Next*.

9. Dependiendo del uso que queramos dar a la base de datos marcaremos una de las tres opciones que se nos muestran a continuación y pulsaremos en *Next*. Normalmente se marcará *Multifunctional Database* salvo que queramos utilizar MySQL como base de datos para transacciones de otra base de datos MySQL.

10. A continuación veremos una pantalla donde seleccionaremos la unidad y la carpeta donde queremos guardar los ficheros de datos (*Tablespace*) de las tablas InnoDB, un tipo de tabla con soporte para la integridad referencial.

    Después de haber establecido la unidad y la carpeta, pulsaremos en *Next*.

11. Seleccionaremos ahora el número aproximado de conexiones concurrentes (varios clientes conectados a la vez) que tendrá nuestro servidor de MySQL. La primera opción asume unas 20, la segunda unas 500 y la tercera permite especificarlas manualmente. Este parámetro es aproximado, no tiene por qué ser exacto.

    Una vez hayamos seleccionado una opción, pulsaremos en *Next*.

12. En la siguiente ventana dejaremos marcada la opción *Enable TCP/IP Networking* si queremos que los clientes se puedan conectar mediante TCP/IP al equipo servidor de MySQL. Podremos cambiar el puerto por el que lo harán. Por defecto se suele dejar 3306. Si tenemos instalado algún cortafuegos deberemos abrir dicho puerto.

    También dejaremos seleccionada la opción *Enable Strict Mode* si queremos que el servidor se comporte como un servidor de bases de datos tradicional, que será lo normal, y pulsaremos el botón *Next*.

13. La pantalla que veremos a continuación nos permite elegir el juego de caracteres que queramos utilizar. Por defecto está marcado Latin1, válido para las lenguas de Europa Occidental. En nuestro caso seleccionaremos UTF-8, que es el juego de caracteres que mejor soporta el multilingüismo, y pulsaremos una vez más el botón *Next*.

14. El siguiente paso es importante pues nos pide que especifiquemos el tipo de arranque del servidor MySQL. Si seleccionamos la primera opción, *Install As Windows Service*, el programa de instalación nos creará un servicio que será el encargado de ejecutar el servidor MySQL. También nos permite especificar el nombre del servicio y si queremos que arranque automáticamente al iniciar el sistema (*Launch the MySQL Server automatically*). La segunda opción, *Include Bin Directory in Windows PATH*, añadirá las variables de entorno necesarias para la ejecución de los ficheros necesarios para iniciar MySQL. La opción recomendada es *Install As Windows Service*.

    Una vez hayamos elegido una opción, pulsaremos de nuevo en *Next*.

15. En la siguiente pantalla introduciremos la contraseña para el usuario administrador (`root`) y pulsaremos en *Next*. Marcaremos la opción *Enable root access from remote machines* si queremos que se pueda acceder como administrador desde otros equipos. Es importante establecer una contraseña para el usuario *root*, ya que si no lo hacemos, cualquiera podría acceder al sistema y hacer lo que quisiera con los datos.

16. Por último, pulsaremos en *Execute*, con lo que se guardarán las opciones de configuración que hemos especificado, y luego en *Finish*, para finalizar la configuración de MySQL.

7.1.1.1. Instalación de MySQL Administrator y MySQL Query Browser

Una vez instalado y configurado el servidor de la base de datos, podemos instalar los programas MySQL Administrator y MySQL Query Browser, dos herramientas gráficas que nos permiten gestionar la información y administrar MySQL a todos los niveles. El primero permite administrar visualmente y de manera sencilla servidores de bases de datos MySQL, mientras que el segundo permite crear, ejecutar y optimizar sentencias SQL en MySQL. Para instalar estos programas simplemente debemos ejecutar los ficheros `mysql-administrator-1.0.20-win` y `mysql-query-browser-1.1.9-win`, que adjuntamos en el CD-ROM, y seguir los pasos de la instalación.

7.1.2. Creación de la base de datos Codetic

Podemos crear la base de datos de varias maneras. Aquí se va a explicar cómo hacerlo utilizando el programa `mysql`, el cliente que viene por defecto con el servidor MySQL. Los pasos a seguir son los siguientes:

1. Nos conectaremos al servidor MySQL como usuario `root` a través del programa cliente que queramos, por ejemplo, MySQL Query Browser.

2. Crearemos una base de datos que se llamará `codetic` y que utilizará el juego de caracteres UTF8. La sentencia SQL a ejecutar sería:

```
CREATE DATABASE codetic
    DEFAULT CHARACTER SET utf8
    DEFAULT COLLATE utf8_general_ci;
```

3. Crearemos todas las tablas de la base de datos y cargaremos todos los datos en dichas tablas a partir del *script* de instalación `codetic-ins.sql`, que adjuntamos en el CD-ROM. Esto se puede hacer mediante las siguientes sentencias, en las que asumimos que su unidad de CD-ROM se encuentra en la unidad D:

```
USE codetic;
SOURCE D:\Install\codetic-ins.sql;
```

4. Por último, crearemos un usuario que tenga como nombre `codeticDBA` y como clave `codeticDBA`, y le daremos todos los permisos sobre la base de datos `codetic`. Este será el usuario a través del cual la aplicación web Codetic se conectará con la base de datos MySQL.

   Aunque en la versión 5.0.2 de MySQL existe una sentencia para crear usuarios, `CREATE USER`, en versiones anteriores se usa exclusivamente la sentencia `GRANT` para crearlos. En general es preferible usar `GRANT`, ya que si se crea un usuario mediante `CREATE USER`, posteriormente hay que usar una sentencia `GRANT` para concederle privilegios. Usando `GRANT` podemos crear un usuario y al mismo tiempo concederle también los privilegios que tendrá:

```
GRANT ALL ON codetic.*
  TO 'codeticDBA'@'%' IDENTIFIED BY 'codeticDBA' ;
```

Todos estos pasos también se pueden realizar sin necesidad de escribir ningún comando SQL, mediante el uso de una herramienta gráfica, como MySQL Administrator.

7.1.3. Instalación del servidor web Tomcat

La instalación de Tomcat requiere que en la máquina esté instalado previamente el JDK [1] de Java. En el caso de que no lo tengamos instalado, podremos instalarlo ejecutando el fichero `j2sdk-1_4_2_04-windows-i586-p.exe` que se adjunta en el CD-ROM.

Una vez hayamos instalado el JDK, el siguiente paso es instalar Tomcat. Para ello se debe ejecutar el archivo `jakarta-tomcat-5.0.28` que viene en el CD-ROM y seguir los pasos de la instalación. Tomcat usa el puerto 8080 como puerto HTTP predeterminado, aunque durante la instalación este número puede ser cambiado a 80. Con esto, evitamos el tener que especificar el puerto 8080 al escribir las direcciones URL de nuestra aplicación.

Finalmente, debemos crear un usuario que tenga como nombre `codeticAdmin` y como clave `codeticAdmin` en el archivo de configuración de usuarios de Tomcat, que se encuentra en `$CATALINA_HOME\conf\tomcat-users.xml`, donde `$CATALINA_HOME` es el directorio de instalación de Tomcat. A dicho archivo le tendremos que añadir las siguientes líneas:

```
<role rolename="codeticAdmin"/>
<user username="codeticAdmin" password="codeticAdmin"
  roles="codeticAdmin"/>
```

Este será el usuario que utilizará el administrador para poder entrar en la zona de administración de la aplicación.

Debido a que es un documento XML, debemos respetar el orden de las etiquetas. Dicho de otro modo, debemos la línea `<role>` debajo de las que ya están y lo mismo para la línea `<user>`.

Como indicamos en el Apartado 3.2, no es recomendable usar Tomcat para un entorno de producción. Para ese fin es mejor usar la combinación Apache-Tomcat.

7.1.4. Instalación de la aplicación web Codetic

Lo único que hay que hacer es copiar el fichero `codetic.war` del CD-ROM al directorio `CATALINA_HOME\webapps`. Un fichero WAR (*Web Archive*) no es más que un fichero comprimido que contiene todos los archivos necesarios para la aplicación web. Será Tomcat el que se encargará de descomprimir el archivo cuando se arranque el servidor web.

7.2. Manual de usuario de Codetic

Bienvenido al manual de usuario de Codetic, una aplicación web desde la que podrá consultar y realizar distintos tipos de búsqueda (por palabras, por idioma, por organización...) sobre

[1] El paquete JDK contiene el entorno de desarrollo de Java de Sun. Sirve para desarrollar programas Java y proporciona el entorno de ejecución necesario para ejecutar dichos programas.

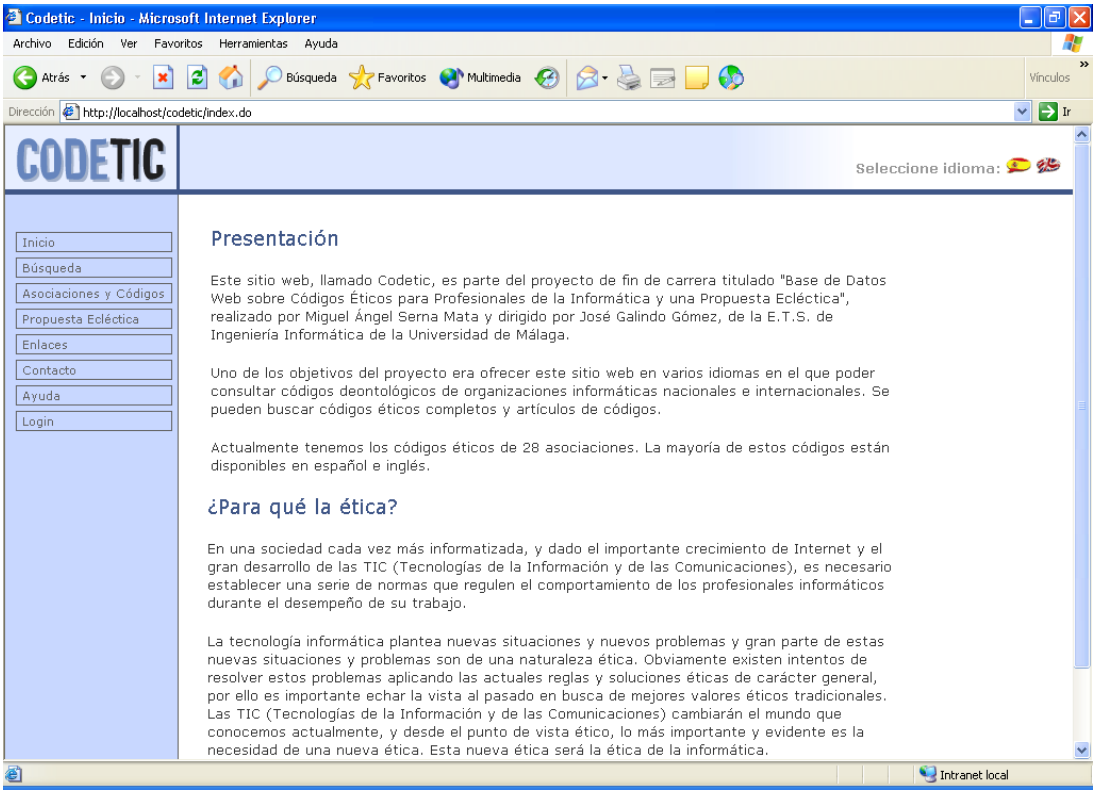

Figura 7.1: Página principal de Codetic.

una base de datos de códigos éticos para profesionales de la informática.

7.2.1. Página principal

A continuación se describen las distintas opciones a las que podemos acceder desde la página principal (Figura 7.1):

- **Cambio de idioma**. Cuando nos conectamos a Codetic, éste detecta automáticamente el idioma del navegador y muestra la página de bienvenida en dicho idioma. Si el idioma no es ni el español ni el inglés, entonces la muestra, por defecto, en inglés. Sin embargo, podemos cambiar el idioma de la aplicación cuando queramos; desde cualquier página. Para ello, deberemos hacer clic en alguna de las banderas que se encuentran en la esquina superior derecha de la página (Figura 7.1).

- **Menú de navegación**. En la parte izquierda de la pantalla (Figura 7.1), tenemos un menú que sirve para navegar a través de la aplicación. Las distintas opciones de este menú son:

  - **Inicio**. Esta es la página de bienvenida de Codetic (Figura 7.1). Desde ella podemos acceder a cualquier parte de la aplicación web.
  - **Búsqueda**. Permite realizar búsquedas de artículos de códigos éticos. Para más información, consulte el Apartado 7.2.2.

- **Asociaciones y Códigos**. Muestra la lista de asociaciones almacenadas en nuestra base de datos. Podemos consultar su información, así como sus códigos éticos. Ver el Apartado 7.2.3 para mayor información.
- **Propuesta Ecléctica**. Contiene la propuesta ecléctica del código ético que proponemos en este proyecto, intentando unificar en un único código las virtudes de los que hemos estudiado y mejorar otros aspectos que consideramos relevantes.
- **Enlaces**. Página donde se muestra una selección de enlaces en Internet relacionados con la ética y los códigos éticos informáticos. Algunos de estos recursos están disponibles en español y otros lo están en inglés.
- **Contacto**. En esta página aparece la información de contacto de los administradores de la web, por si desea hacer sugerencias, preguntas o colaborar enviando traducciones de códigos éticos.
- **Ayuda**. Contiene este manual de usuario. En él se documenta la parte pública de la aplicación web Codetic.
- **Login**. Página desde la que se accede a la zona de administración de Codetic, reservada exclusivamente para los usuarios administradores.

7.2.2. Búsqueda de artículos

Desde esta página (Figura 7.2) podemos realizar búsquedas de artículos. Para ello, lo único que tenemos que hacer es introducir las palabras que deseamos buscar y pulsar el botón «Buscar». Una vez hecho esto, nos aparecerá una página con los resultados de la búsqueda.

7.2.2.1. Interpretación de los resultados de las búsquedas

Aunque creemos que la página de resultados es fácil de interpretar, vamos a explicar los elementos que la forman. La Figura 7.3 muestra estos elementos:

A. **Palabras buscadas**. Especifica las palabras que hemos buscado.

B. **Barra de navegación**. Contiene el rango de resultados, sobre el total, que se están visualizando. También contiene la navegación de páginas, donde el enlace «Siguiente» salta a la siguiente página de resultados. En el caso de que hubiera un enlace «Anterior», éste saltaría a la anterior página de resultados.

C. **Resultado**. Representa un artículo encontrado. Está formado, a su vez, por tres partes:

   C.1. **Título del resultado**. Consta del título del código ético del artículo, seguido del nombre del mismo. Al hacer clic en este enlace, verá el artículo completo (Apartado 7.2.2.2).

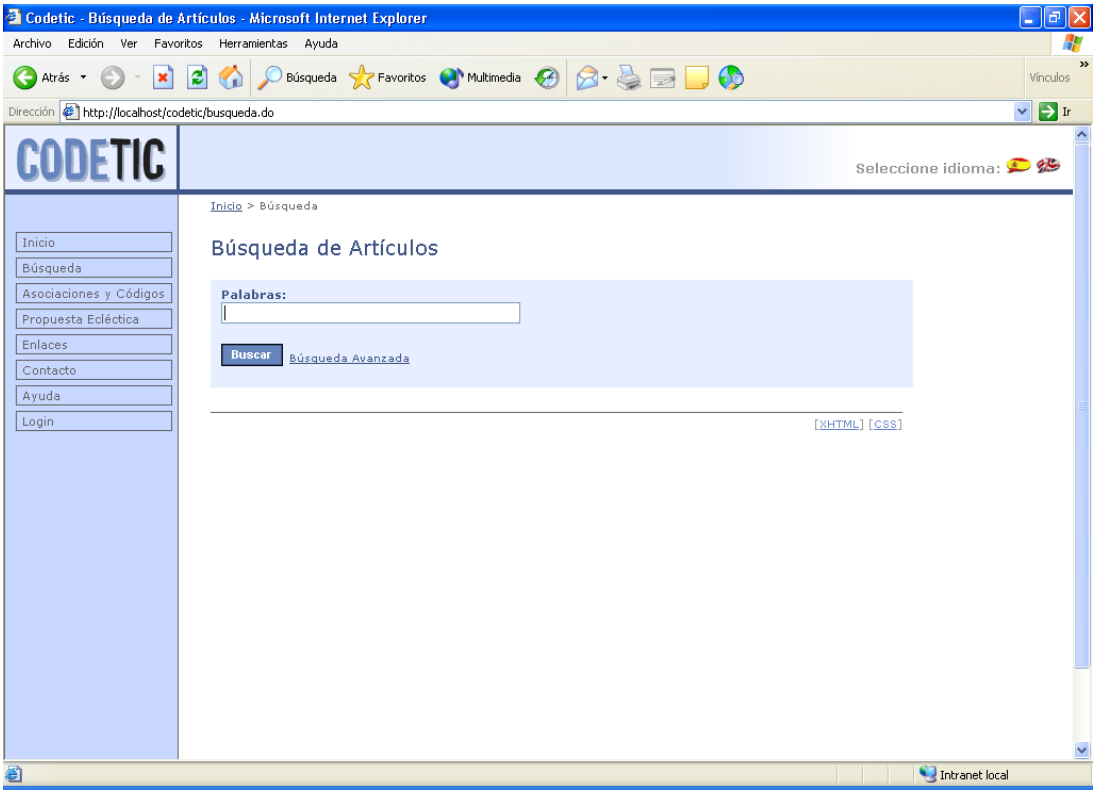

Figura 7.2: Página de búsqueda de artículos.

C.2. **Nombre de la asociación**. Es el nombre de la asociación a la que pertenece el código del artículo.

C.3. **Texto del resultado**. Contiene una parte del texto del artículo encontrado.

7.2.2.2. Visualización de un artículo completo

Al hacer clic en el título de cualquier resultado de la búsqueda, se accederá al contenido completo del artículo. En la Figura 7.4 podemos ver un artículo completo de ejemplo. Como se puede observar, en la parte derecha de la página se muestra la siguiente información sobre el código ético al que pertenece el artículo:

- **Asociación**. Nombre de la asociación a la que pertenece el código ético.
- **Aprobación**. Año en el que se aprobó el código ético. Si no se dispone de tal información, entonces aparecerá la palabra «N/D» (No Disponible) en su lugar.
- **Web**. Enlace a la página web oficial del código. En el caso de que apareciese, el significado del texto «N/D» es el mismo que para la aprobación.

Al final del artículo, existe un enlace llamado «Ver el código ético completo» que, como su propio nombre indica, nos lleva al código ético completo.

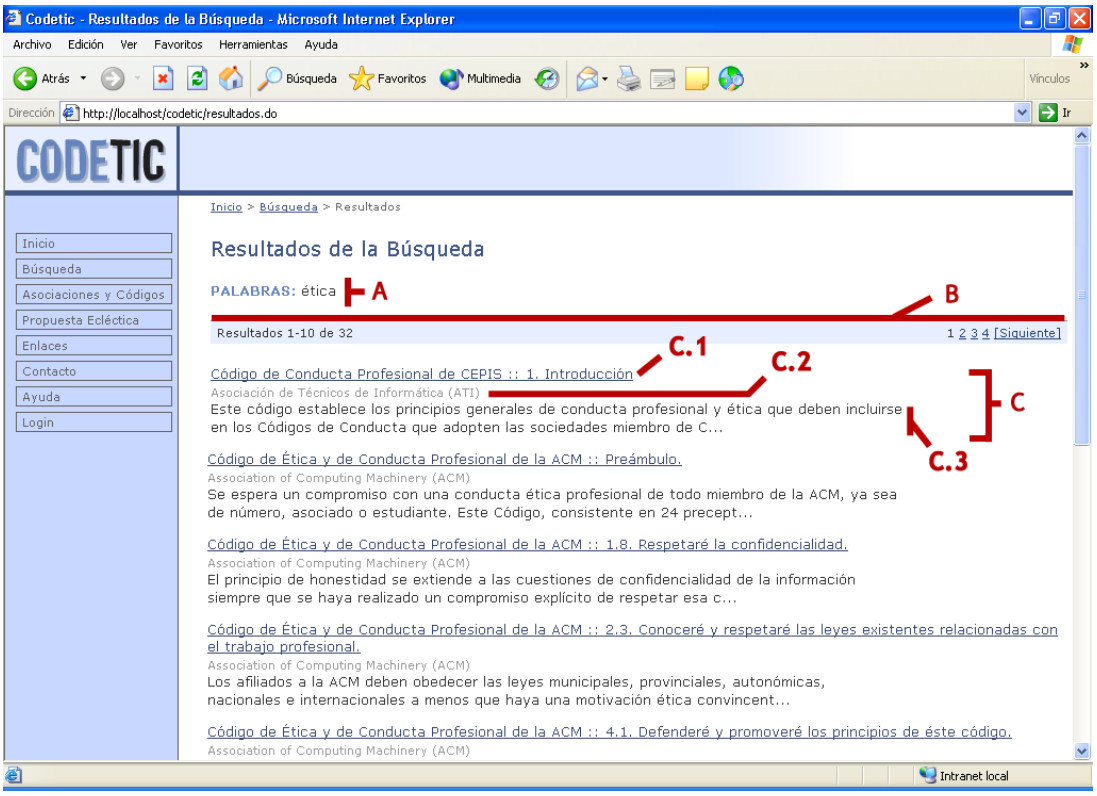

Figura 7.3: Elementos de la página de resultados.

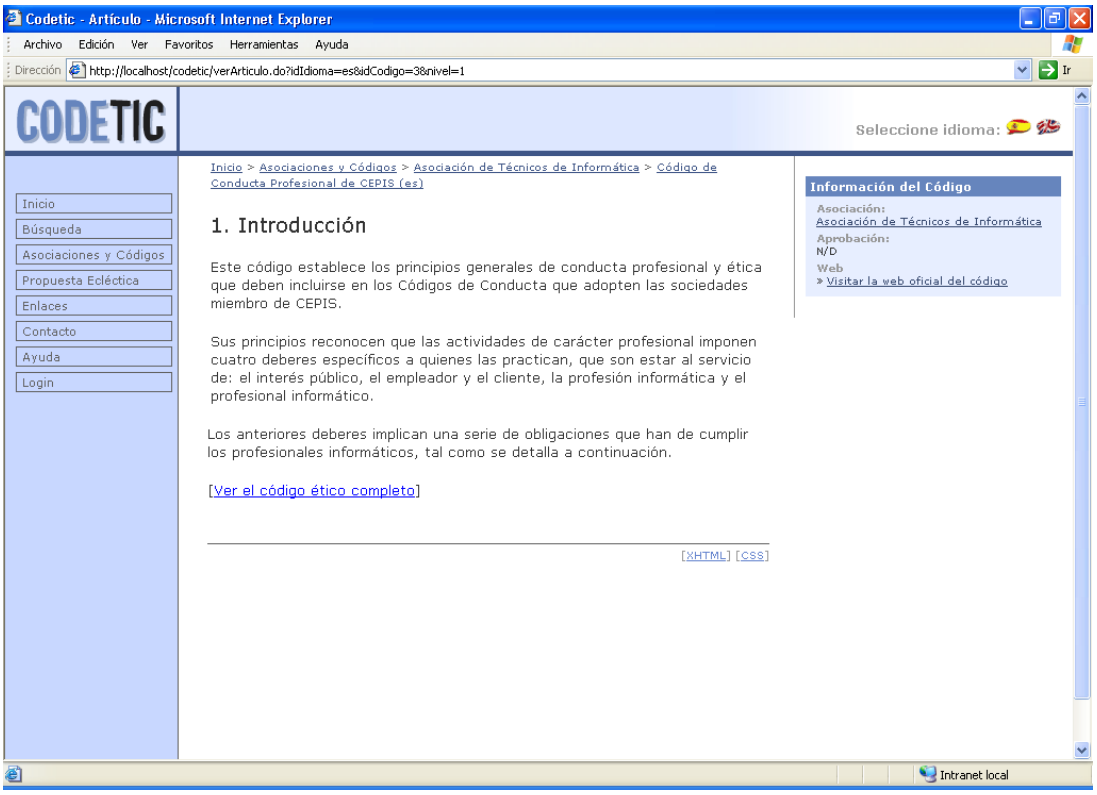

Figura 7.4: Página de visualización de un artículo completo.

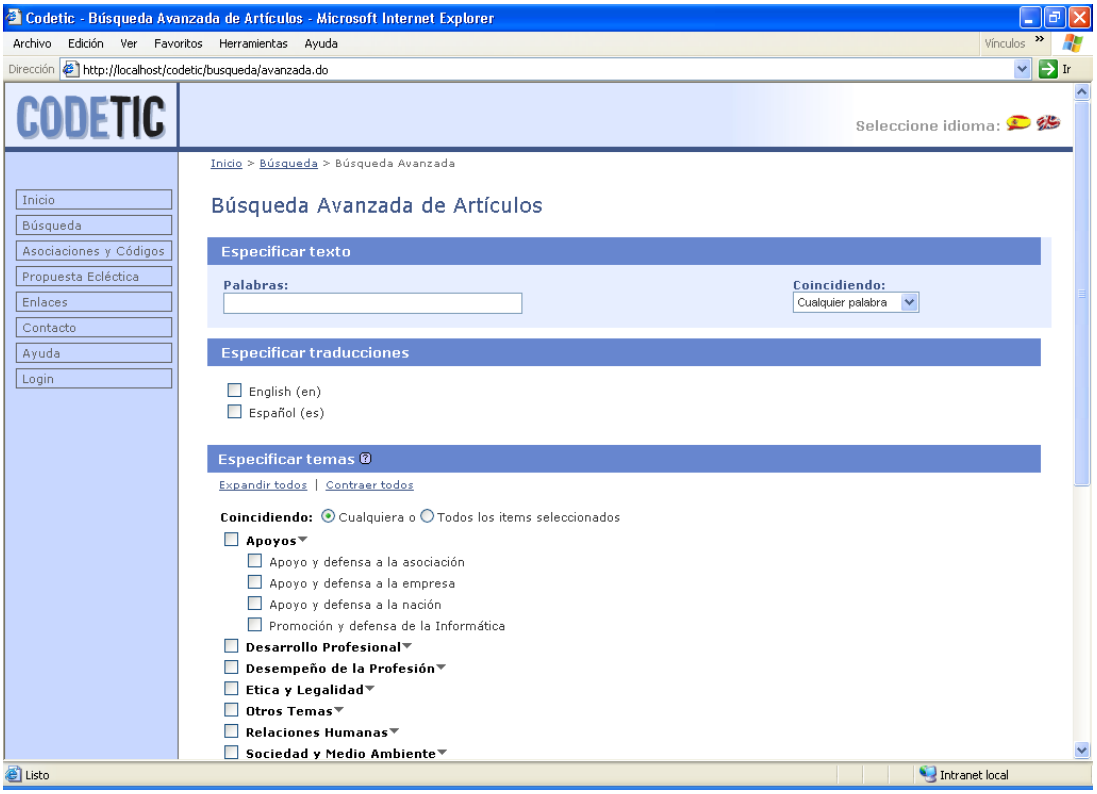

Figura 7.5: Página de búsqueda avanzada de artículos.

7.2.2.3. Búsqueda avanzada

Desde la búsqueda simple (Figura 7.2) podemos acceder a la búsqueda avanzada (Figura 7.5) con sólo hacer clic en el enlace «Búsqueda Avanzada», que se encuentra a la derecha del botón «Buscar». La búsqueda avanzada permite realizar consultas personalizadas, más precisas, que ayuden a localizar el contenido deseado.

El formulario de búsqueda avanzada consta de los siguientes elementos:

1. **Especificar texto**. Debemos introducir las palabras deseadas en el campo de texto «Palabras». En el caso de que hayamos introducido varias palabras, seleccionarremos de la lista desplegable «Coincidendo» una de las siguientes opciones:

   - **Cualquier palabra**. Devuelve todos los artículos en los que aparezcan cualquiera de las palabras que se indiquen. Por ejemplo, la búsqueda de las palabras «dilema ético» devuelve todos los artículos en los que aparezca la palabra «dilema», todos los artículos en los que aparezca la palabra «ético», y todos los artículos en los que aparezcan las dos palabras.
   - **Todas las palabras**. Devuelve todos los artículos en los que aparezcan todas la palabras que se indiquen. Por ejemplo, la búsqueda de las palabras «dilema ético» devuelve todos los artículos en los que aparezcan la palabra «dilema» y también la palabra «ético».

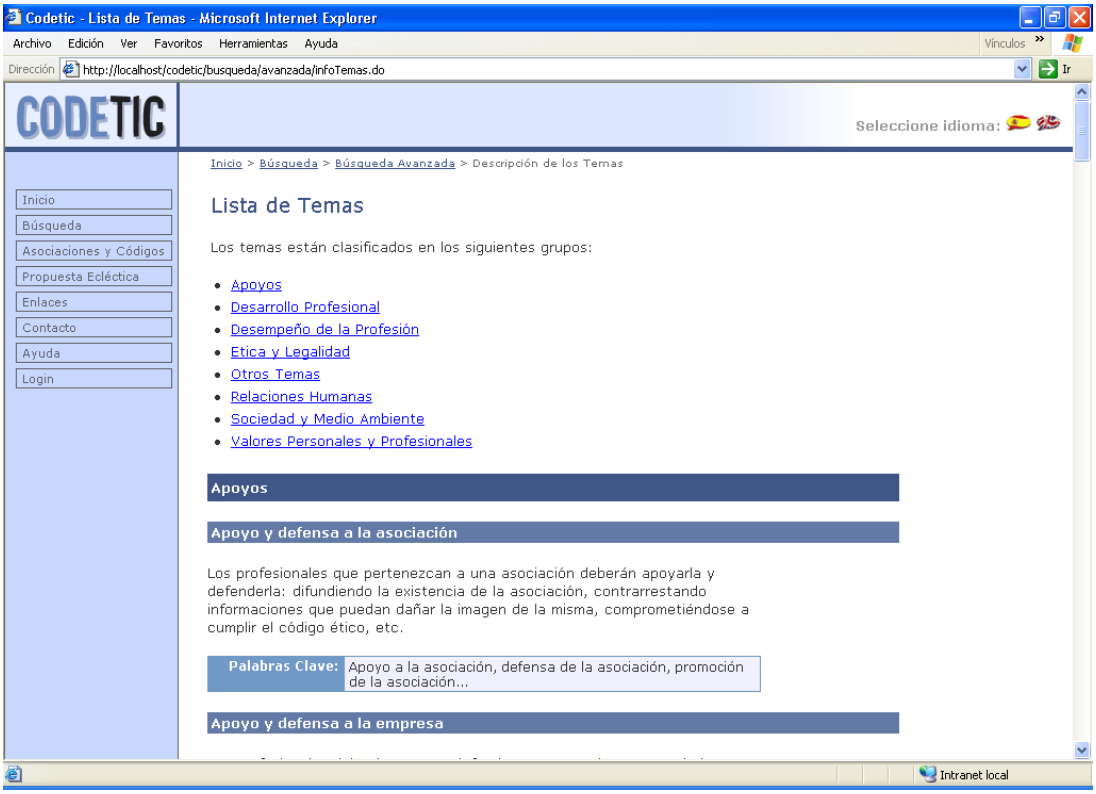

Figura 7.6: Página que contiene las descripciones de todos los temas.

- **Frase exacta**. Devuelve todos los artículos que contengan la frase introducida (todas las palabras y en el mismo orden). Por ejemplo, la búsqueda de las palabras «dilema ético» devuelve todos los artículos en los que aparezca exactamente la frase «dilema ético».

2. **Especificar traducciones**. Está formado por una serie de casillas que permiten acotar la búsqueda a los artículos escritos en un determinado idioma.

3. **Especificar temas**. Permite acotar la búsqueda a los artículos que traten sobre determinados temas. Los temas aparecen organizados por grupos. En un principio, únicamente se muestran los temas del primer grupo. Para mostrar los temas de todos los grupos deberemos hacer clic en el enlace «Expandir todos», y para ocultarlos deberemos hacerlo en «Contraer todos». Si sólo queremos ver los temas de un grupo concreto, basta con hacer clic en el nombre del propio grupo. Si hacemos clic, de nuevo, en el nombre del grupo, volveremos a ocultar los temas del mismo.

   Además de señalar los temas y/o grupos de temas que queramos, también deberemos elegir una de las siguientes opciones para el campo «Coincidiendo»:

   - **Cualquiera**. Devuelve todos los artículos que traten sobre cualquiera de los temas señalados.

- **Todos los items seleccionados**. Devuelve todos los artículos que traten sobre todos y cada uno de los temas señalados.

Por último, a la derecha del título «Especificar temas», existe un icono de ayuda. Si lo pulsamos, veremos una lista con todos los temas, organizados por grupos (Figura 7.6). Para cada tema, se muestra su nombre, junto a una descripción y un conjunto de palabras clave que ayudan a delimitar la problemática abarcada por el tema.

4. **Especificar asociaciones**. Permite acotar la búsqueda a los artículos de determinadas asociaciones. Las asociaciones aparecen organizadas por continentes. En un principio, sólo aparecen los continentes, estando ocultas todas sus asociaciones. Para mostrar las asociaciones de todos los continentes deberemos hacer clic en el enlace «Expandir todos», y para ocultarlos deberemos hacerlo en «Contraer todos». Si sólo queremos ver las asociaciones de un continente concreto, basta con hacer clic en el nombre del propio continente. Si hacemos clic, de nuevo, en el nombre del continente, volveremos a ocultar las asociaciones del mismo.

5. **Otras opciones**. Contiene un menú desplegable con el número de artículos que se mostrarán en cada página de resultados. Pueden ser intervalos de 10, 25 ó 50 artículos. Debe escoger una de estas tres opciones.

7.2.3. Asociaciones y códigos

Esta página muestra una lista con todas las asociaciones informáticas almacenadas en la base de datos (Figura 7.7). Existen dos formas de ordenarlas:

- **Alfabéticamente**. Para ello tenemos que hacer clic en el enlace «Alfabéticamente», que se encuentra inmediatamente debajo del título de la página. Este es el orden por defecto en el que aparecen las asociaciones (Figura 7.7).

- **Geográficamente**. Muestra las asociaciones organizadas por continentes (Figura 7.8). Se consigue haciendo clic en el enlace «Geográficamente», que está situado debajo del título de la página.

Los nombres de las asociaciones enlazan con una página en el que se muestra toda la información de la asociación, así como todos sus códigos.

7.2.3.1. Información de una asociación

A esta página se accede desde la página que muestra todas las asociaciones, simplemente haciendo clic en cualquiera de los nombres de las asociaciones. Como ejemplo, mostramos la información de la ACM (*Association of Computing Machinery*) (Figura 7.9). En ella podemos ver varias cosas:

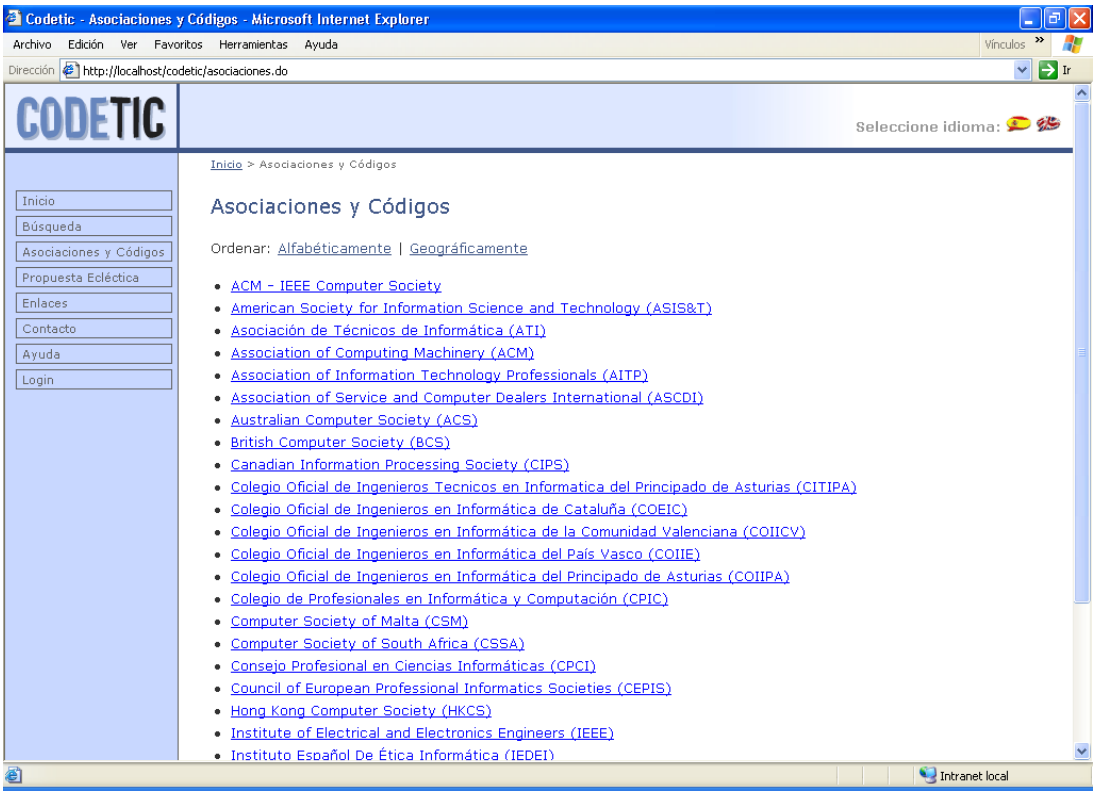

Figura 7.7: Página que muestra la lista de asociaciones informáticas.

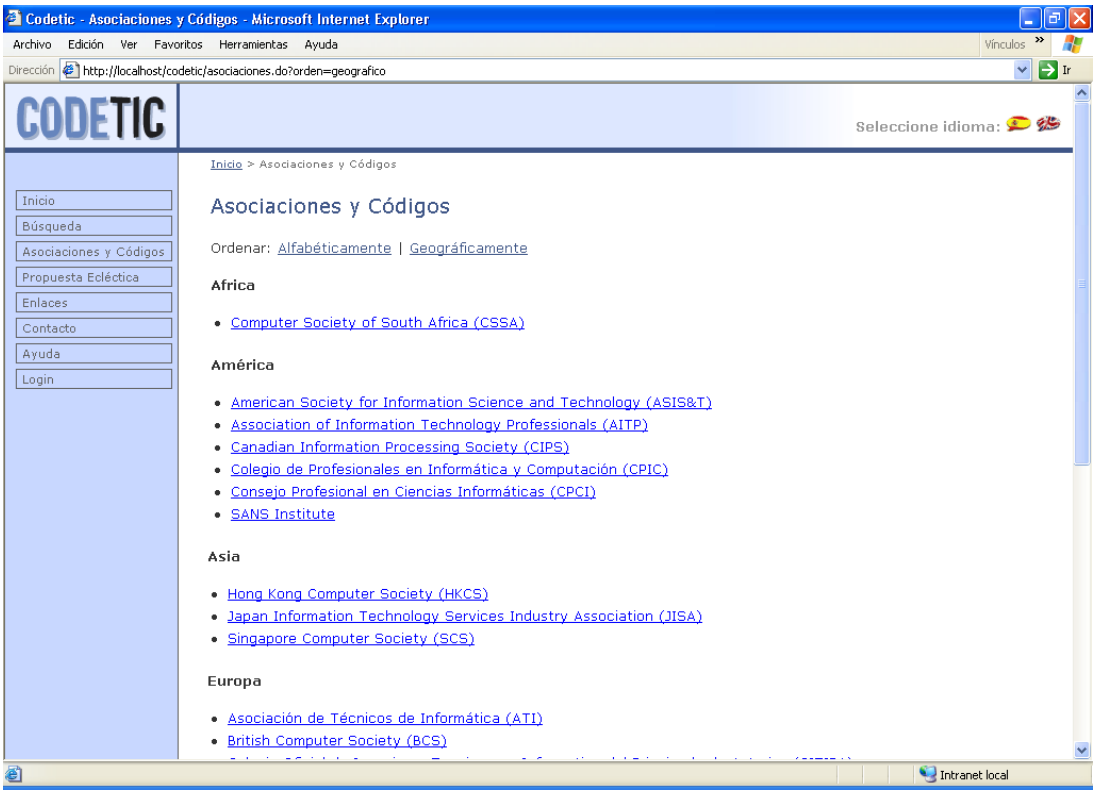

Figura 7.8: Página que muestra la lista de asociaciones informáticas ordenadas por continente.

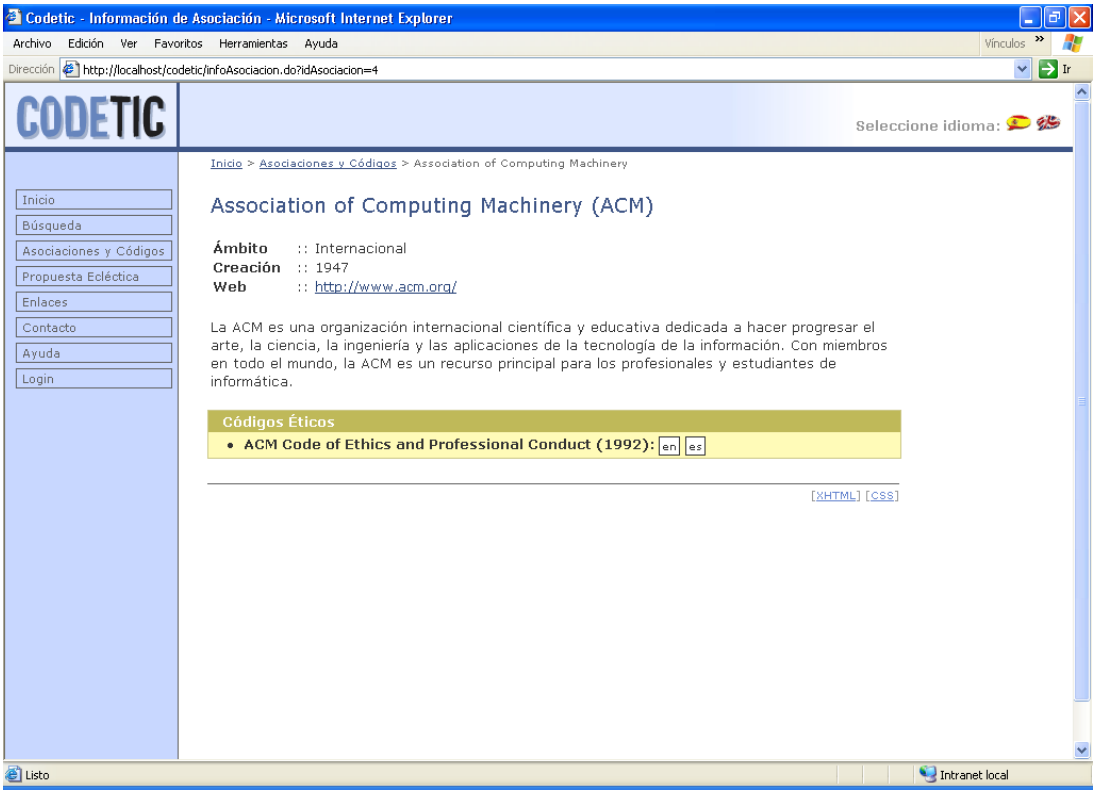

Figura 7.9: Página de información de la asociación ACM.

- **Ámbito**. Ámbito geográfico al que pertece la asociación. Por ejemplo, «Internacional», «Continental (Europa)», etc.
- **Creación**. Año de creación de la asociación.
- **Web**. Sitio web oficial de la asociación.

Además, aparece una descripción de la asociación y, en último lugar, un cuadro con todos los códigos éticos de la asociación. Al final de cada código tenemos uno o varios pequeños recuadros blancos con un código de idioma en su interior, cada uno de los cuales representa a las distintas traducciones disponibles para ese código. Los distintos códigos de idiomas que pueden aparecer son:

- **es** para el español.
- **en** para el inglés.

7.2.3.2. Visualización de un código ético

Si hacemos clic en alguna de las traducciones de un código (Figura 7.9), representadas por recuadros blancos, entonces visualizaremos dicha traducción del código ético. En la Figura 7.10 podemos ver el código ético de la ACM.

Aparte del propio código, existen otros elementos dentro de esta página:

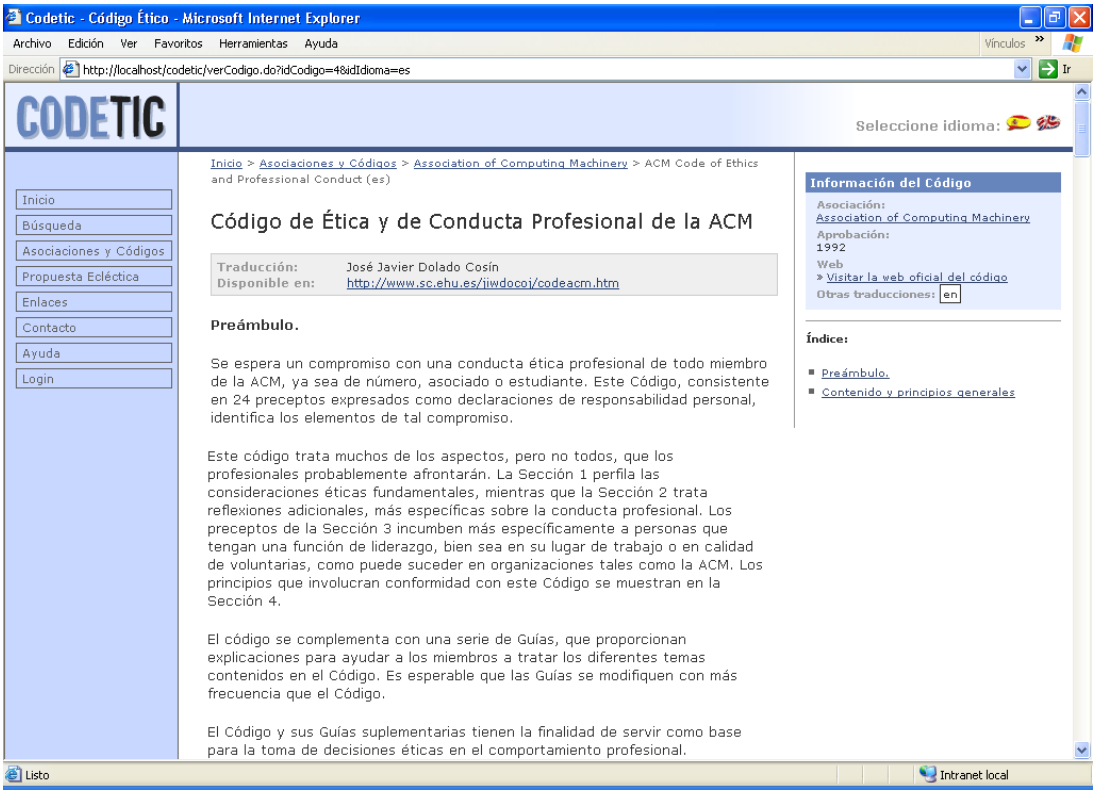

Figura 7.10: Página con el código ético en español de la asociación ACM.

- **Recuadro de traducción**. Si aparece, puede especificar el nombre de los autores de la traducción, la página web oficial de la traducción y/o comentarios de los traductores.
- **Información del código**. Muestra la siguiente información sobre el código ético al que pertenece el artículo:
  - **Asociación**. Nombre de la asociación a la que pertenece el código ético.
  - **Aprobación**. Año en el que se aprobó el código ético. Si no se dispone de tal información, entonces aparecerá la palabra «N/D» (No Disponible) en su lugar.
  - **Web**. Enlace a la página web oficial del código. En el caso de que apareciese, el significado del texto «N/D» es el mismo que para la aprobación.
  - **Otras traducciones**. Permite acceder al resto de traducciones disponibles para dicho código, en el caso de que éstas existan.
- **Índice**. Es el índice general de contenidos del código ético.

7.3. Manual de administración de Codetic

Bienvenido al manual de administración de Codetic. Este manual surge con el propósito de explicar todas aquellas tareas que se pueden realizar desde la zona de administración de Codetic

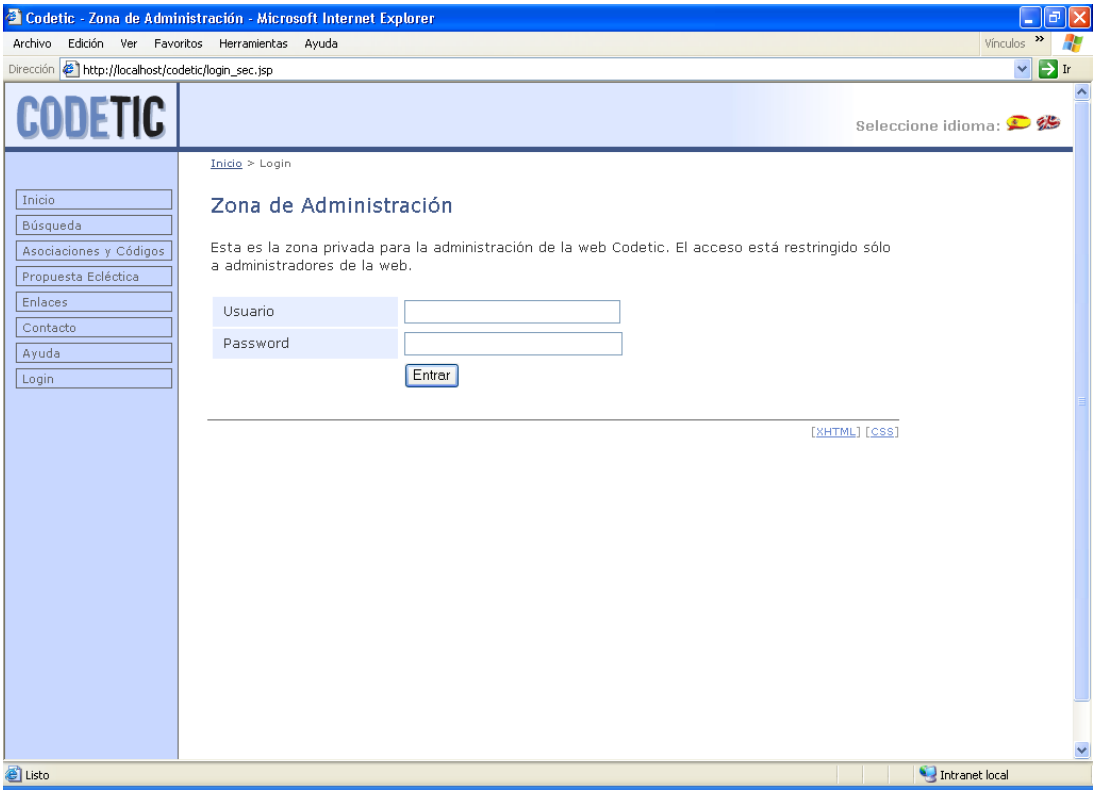

Figura 7.11: Página de identificación.

y que permiten gestionar la información contenida en la base de datos.

7.3.1. Entrar y salir del sistema

Lo primero que tenemos que hacer para poder entrar en la zona de administración es identificarnos como administradores. Para ello, haremos clic en el enlace «Login» del menú principal (Figura 7.1), con lo que accederemos a la página de identificación (Figura 7.11). Esta página contiene un formulario en el que introduciremos nuestro nombre de usuario y nuestra contraseña. Una vez introducidos, pulsaremos el botón «Entrar», con lo que se comprobará en el sistema la existencia de dicho usuario. Si el usuario existe, entonces, pasaremos a la página de administración (Figura 7.12), que veremos más adelante. Si no, se nos volverá a mostrar la página de identificación, para que volvamos a identificarnos.

7.3.2. Página de administración

La página de administración (Figura 7.12) es la página principal de la zona de administración. Si observamos la parte izquierda de la página, observaremos que hay un nuevo menú justo debajo del menú principal. Este nuevo menú es el menú de administración, que nos permite navegar por la zona de administración. Las distintas opciones de este menú, que iremos viendo a lo largo de los siguientes apartados, son:

- **Administración**. Esta opción va precedida por dos puntos (::) y encabeza el resto de opciones. Es el enlace a la página de administración.
- **Logout**. Salir del sistema.
- **Idiomas**. Permite administrar todos los idiomas de la base de datos.
- **Ámbitos**. A través de esta opción podemos administrar todos los ámbitos de la base de datos, así como sus respectivas traducciones.
- **Asociaciones y Códigos**. Permite administrar todas las asociaciones y códigos éticos de la base de datos, así como sus respectivas traducciones.
- **Temas**. Se encarga de la administración de los temas y de sus traducciones.
- **Grupos de temas**. Esta opción nos permite gestionar los grupos de temas y sus traducciones.
- **Ayuda de Administración**. Contiene este manual de administración. En él se documenta la zona de administración de la aplicación web Codetic.

Aparte de este menú, que estará disponible hasta que salgamos del sistema como administradores, en el cuerpo de la página de administración también aparece una lista de enlaces. Como podemos observar, esta lista es simplemente un subconjunto de las opciones del menú de administración vistas anteriormente.

7.3.3. Consideraciones generales

Las páginas de gestión de todas las entidades (idiomas, ámbitos, traducciones de ámbito...) siguen un mismo patrón. En todas ellas existe una tabla en la que podemos ver los distintos registros que hay almacenados para esa entidad. Por ejemplo, en la Figura 7.13 puede verse la página de administración de los idiomas. En ella vemos que existen dos idiomas almacenados en la base de datos. El primero de ellos es el inglés y tiene como identificador la palabra «en» y como nombre «English». El segundo es el español y tiene como identificador la palabra «es» y como nombre «Español».

Los registros de la tabla pueden ordenarse en orden descendente o ascendente según el contenido de determinadas columnas. Estas columnas se diferencian del resto porque tienen un icono especial situado a su derecha y porque, al pasar el cursor por encima de ellas, su nombre se subraya. Existen tres tipos de iconos asociados a estas columnas y se interpretan de la siguiente manera:

 Los registros de la tabla no están ordenados por esta columna.

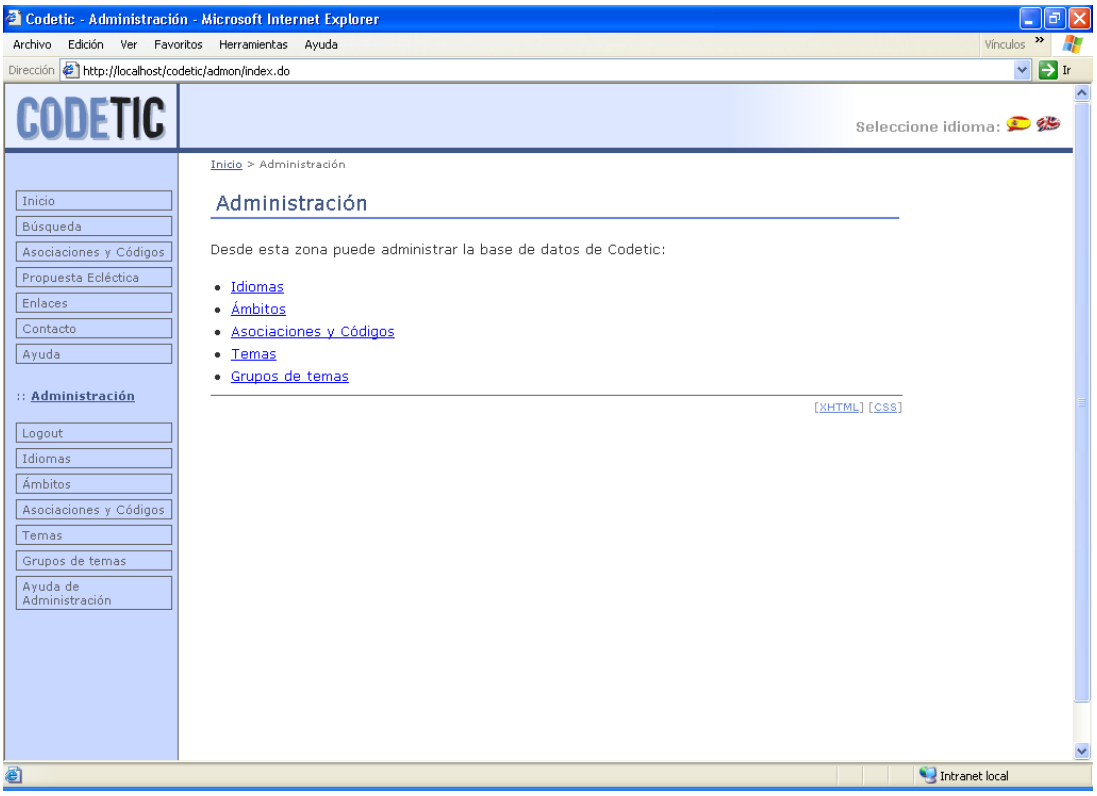

Figura 7.12: Página de administración.

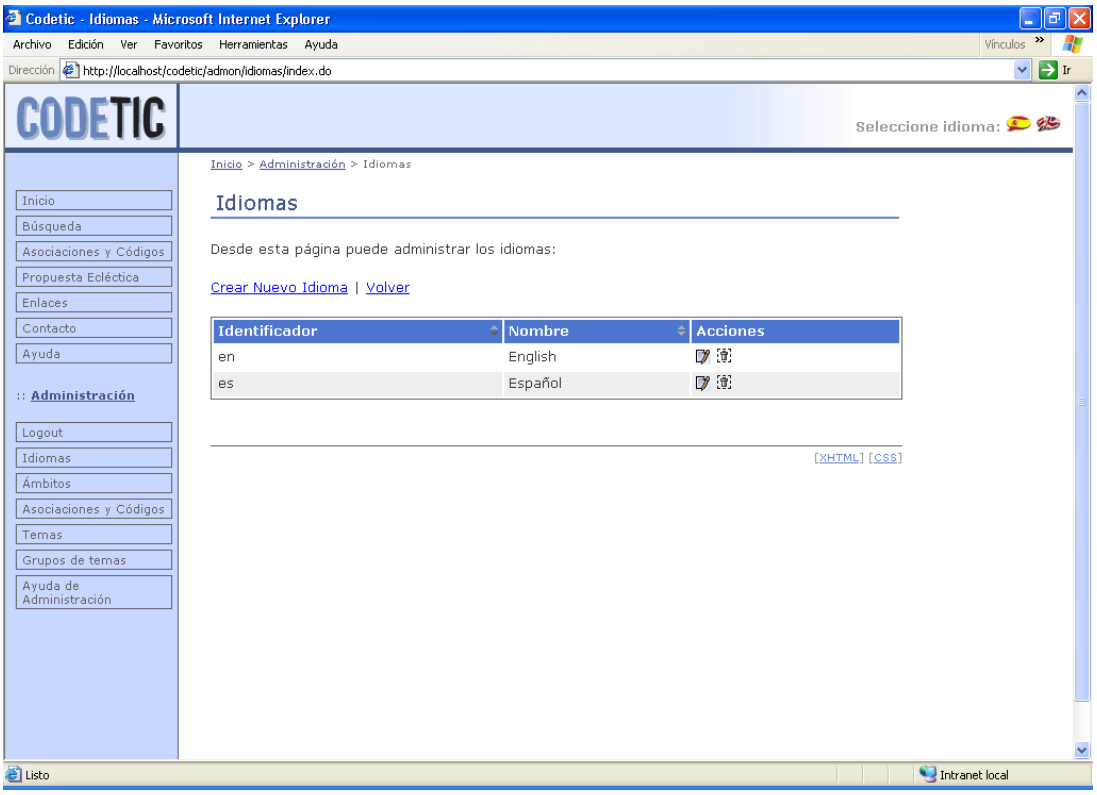

Figura 7.13: Página de administración de idiomas.

 Los registros de la tabla están ordenados ascendentemente según el contenido de esta columna.

 Los registros de la tabla están ordenados descendentemente según el contenido de esta columna.

Todas estas tablas están formadas por una serie de columnas que se corresponden con algún atributo de la entidad, excepto la última columna, denominada «Acciones». Esta columna contiene un icono por cada acción que podemos llevar a cabo para cada uno de los registros almacenados en la base de datos:

 Editar un registro (botón editar). Este icono aparece en todas las tablas.

 Eliminar un registro (botón eliminar). Este icono aparece en todas las tablas.

 Administrar las traducciones asociadas a un registro (botón traducciones). Este icono sólo aparece en las tablas de gestión de algún tipo de traducciones.

 Administrar los códigos éticos de una asociación (botón códigos). Este botón sólo aparece en la tabla de gestión de asociaciones.

 Administrar las secciones de un código ético (botón secciones). Este botón sólo aparece en la tabla de gestión de códigos éticos.

 Administrar las traducciones de las secciones asociadas a la traducción de un código ético (botón secciones traducidas). Este botón sólo aparece en la tabla de gestión de traducciones de un código ético.

Además de estas acciones, encima de las tablas existe un enlace que nos permite crear un nuevo registro en la base de datos (véase la Figura 7.13 para el caso de los idiomas). Con esta, son tres las acciones comunes a todas las tablas: crear, editar y eliminar.

En las operaciones de creación y edición de un registro es necesario rellenar o modificar los campos de los formularios asociados al registro. Los campos obligatorios de cada formulario aparecen marcados con asteriscos «*». Las figuras 7.14 y 7.15 muestran, respectivamente, los formularios de creación de un nuevo idioma y de edición del idioma inglés.

Para que la creación y la modificación de un registro se lleve a cabo los datos introducidos en el formulario deben ser válidos. Si no lo son, entonces se nos volverá a mostrar el formulario, donde se nos indicarán todos los errores de validación.

La eliminación de un registro implica la eliminación de todos los registros vinculados a éste. Por ejemplo, al eliminar un idioma, todos las traducciones escritas en ese idioma también son eliminadas. Por eso, antes de eliminar un registro se nos muestra un mensaje de confirmación.

Existe un elemento especial de navegación llamado *breadcrumb* (rastro de migas de pan) que aparece en todas las páginas y está situado debajo de la cabecera e inmediatamente antes del contenido principal de las mismas. Este elemento nos permite orientarnos dentro del sitio

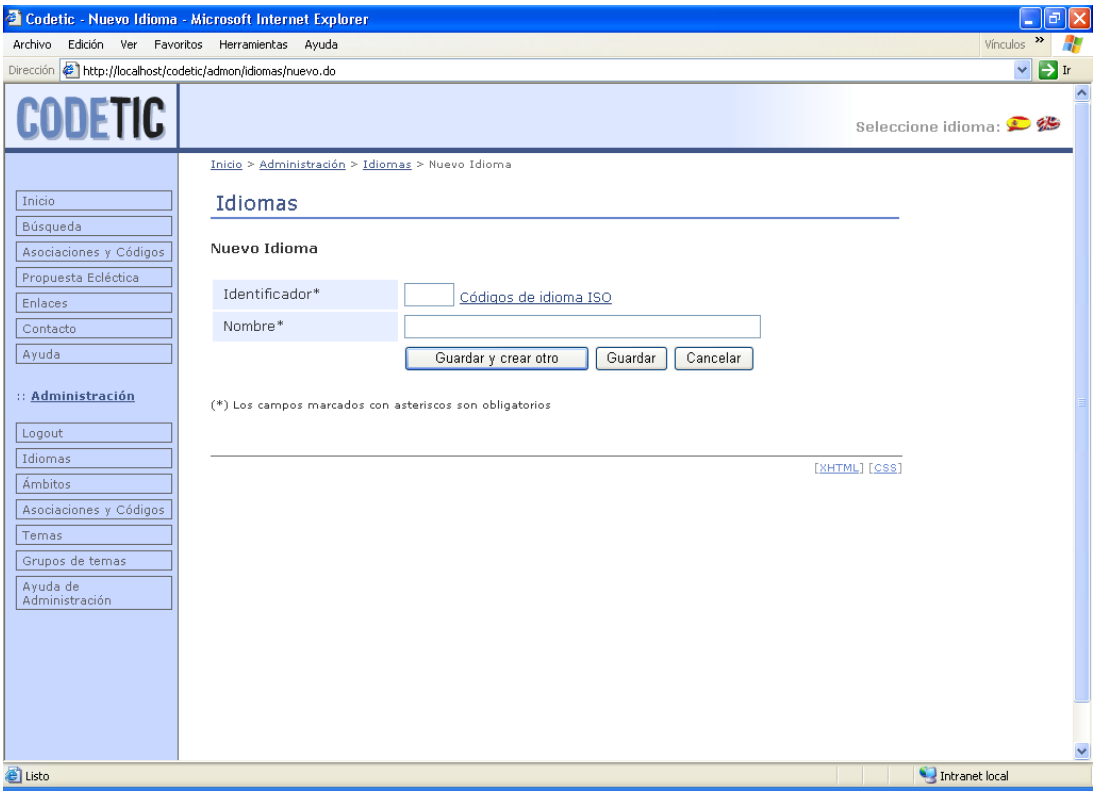

Figura 7.14: Página de creación de un nuevo idioma.

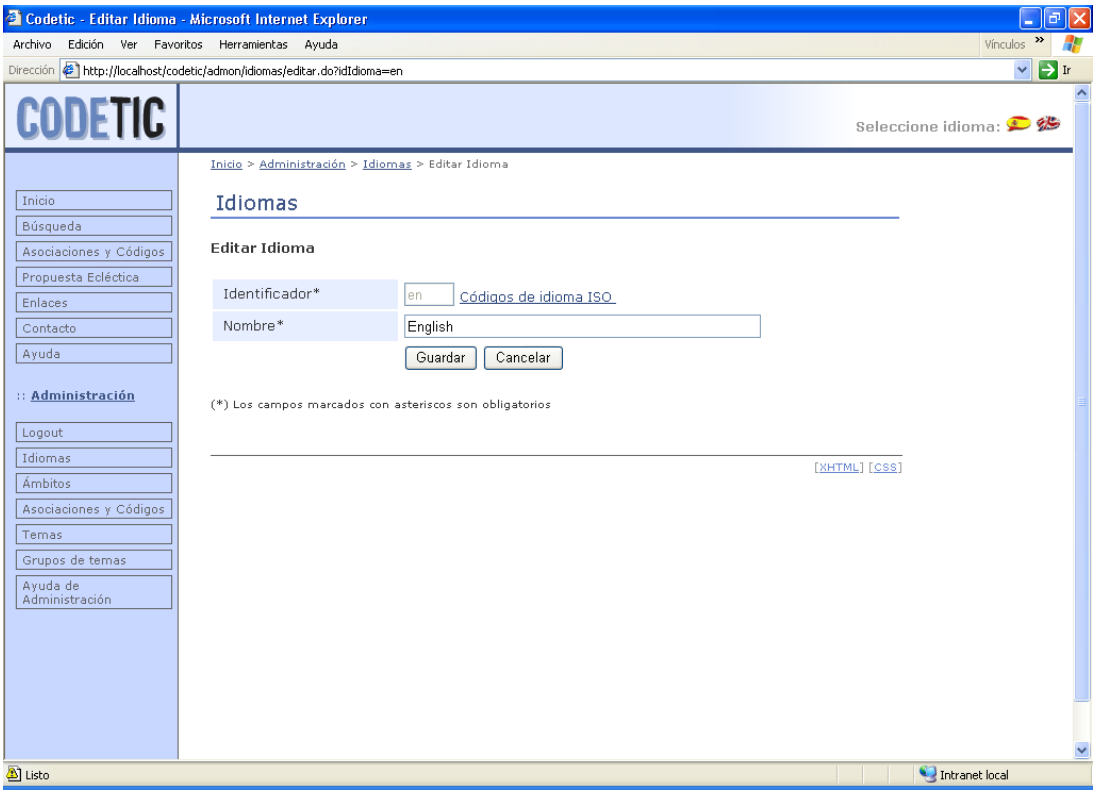

Figura 7.15: Página de edición de un idioma.

web Codetic y es especialmente útil para navegar por la zona de administración, puesto que esta zona tiene bastantes niveles de profundidad en los que es fácil perderse. Para la página de gestión de idiomas (Figura 7.13), el *breadcrumb* que podemos ver es el siguiente:

Inicio >Administración >Idiomas

Por último, comentar que todos los iconos disponen de un mensaje de ayuda emergente (*tooltiptext*) que es mostrado al poner el cursor del ratón encima de ellos.

7.3.4. Gestión de idiomas

Podemos acceder a esta página (Figura 7.13) a través de la opción «Idiomas» del menú de administración o a través del enlace «Idiomas» que aparece en el cuerpo de la página de administración.

Esta página muestra una tabla con todos los idiomas que hay almacenados en la base de datos. Las columnas de esta tabla son:

- **Identificador**. Código de dos letras que identifica al idioma.
- **Nombre**. Es el nombre del idioma escrito en su mismo idioma.
- **Acciones**. Contiene las diversas acciones que se pueden realizar sobre cada uno de los idiomas:

   Editar un idioma (botón editar).

   Eliminar un idioma (botón eliminar).

Más arriba de la tabla de idiomas tenemos dos enlaces. El primero, «Crear Nuevo Idioma», nos lleva al formulario de creación de un nuevo idioma. El segundo, «Volver», nos permite volver a la página anterior.

A continuación vamos a explicar cada una de las operaciones que podemos realizar para administrar los idiomas.

Crear un nuevo idioma

Para crear un nuevo idioma debemos hacer clic en el enlace «Crear Nuevo Idioma» de la página de administración de idiomas (Figura 7.13). Entonces, accederemos a la página que se muestra en la Figura 7.14. A continuación, introduciremos los datos del nuevo idioma en los campos del formulario:

- **idIdioma**. Representa el código de idioma ISO definido por el estándar ISO-639-1, disponible en http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt. Este estándar proporciona códigos de dos letras escritas en minúsculas para 136 lenguas del mundo. Por ejemplo, el código ISO-639-1 para el inglés es «en».

- **nombre**. Representa el nombre del idioma escrito en el mismo idioma. Por ejemplo, para el español sería «español», mientras que para el inglés sería «english».

Una vez los hayamos introducido, haremos clic en el botón «Guardar y crear otro» si deseamos insertar otro idioma. En este caso, si los datos introducidos son válidos, se crea el idioma y se nos vuelve a mostrar de nuevo el formulario vacío para que introduzcamos los datos de un nuevo idioma. Si no, haremos clic en el botón «Guardar», con lo que el idioma se creará, en el caso de que sus datos sean válidos, y regresaremos a la página de administración de idiomas. Existe otro botón llamado «Cancelar» que nos envía directamente a la página de administración de idiomas, cancelando la operación.

Editar un idioma

El administrador puede editar y modificar los idiomas de la base de datos. Para ello, deberá pulsar el botón «Editar» del idioma deseado desde la pantalla de administración de idiomas (Figura 7.13). La Figura 7.15 muestra la pantalla para gestionar un idioma.

Una vez hayamos modificado los datos, pulsaremos el botón «Guardar» para guardar los cambios. También podemos pulsar el botón «Cancelar» para cancelar la operación.

Eliminar un idioma

Para eliminar un idioma hay que pulsar en el botón «Eliminar» desde la pantalla de administración de idiomas (Figura 7.13). Entonces nos aparecerá un mensaje por pantalla para que confirmemos la eliminación. Al eliminar un idioma, todos los datos asociados a éste se eliminarán del sistema.

7.3.5. Gestión de ámbitos

Los ámbitos representan las posibles zonas geográficos a las que pueden pertenecer las distintas asociaciones. Para acceder a la pantalla de administración de ámbitos (Figura 7.16) debemos hacer clic en la opción «Ámbitos» del menú de administración o en el enlace «Ámbitos» que aparece en la página de administración.

Esta página muestra una tabla con todos los ámbitos de la base de datos. Esta tabla tiene las siguientes columnas:

- **Nombre de Administración**. Es el nombre que representa, independientemente del idioma, a un ámbito determinado.

- **Acciones**. Contiene las diversas acciones que se pueden realizar sobre cada uno de los ámbitos:

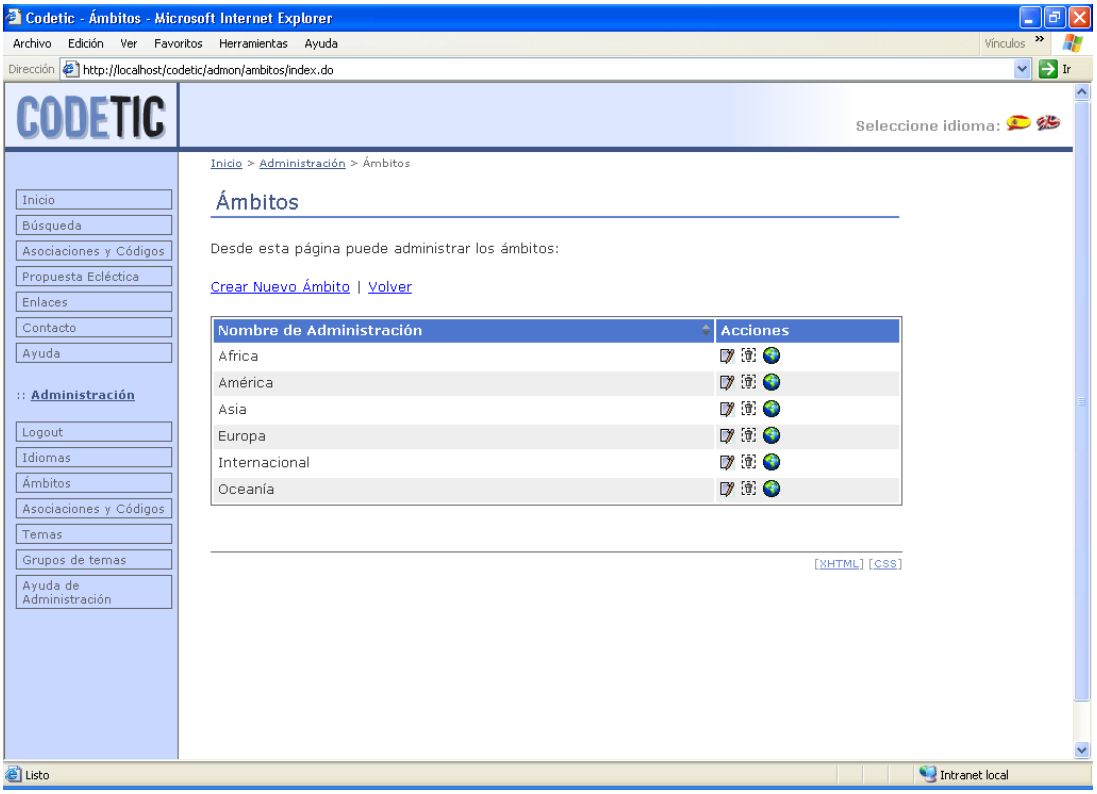

Figura 7.16: Página de administración de ámbitos.

 Editar un ámbito (botón editar).

 Eliminar un ámbito (botón eliminar).

 Administrar las traducciones del ámbito (botón traducciones).

Aparte de esta tabla, tenemos dos enlaces situados justo encima de ella. El primero, «Crear Nuevo Ámbito», nos lleva al formulario de creación de un nuevo ámbito. El segundo, «Volver», nos permite volver a la página anterior.

Expliquemos cada una de las operaciones que podemos realizar sobre los ámbitos.

Crear un nuevo ámbito

Para crear un nuevo ámbito debemos hacer clic en el enlace «Crear Nuevo Ámbito» de la página de administración de ámbitos (Figura 7.16). Entonces, accederemos a la página que se muestra en la Figura 7.17. El único dato que se pide en esta pantalla es:

- **Nombre de Administración**. Un nombre que representa, independientemente del idioma, a un ámbito determinado y es utilizado para la administración de Codetic. Puesto que el idioma nativo de los administradores es el español, este atributo estará escrito en español. Como ejemplo, un posible valor para este atributo sería «Internacional».

Tras introducir este dato, haremos clic en el botón «Guardar y crear otro» si deseamos insertar otro ámbito. En este caso, si los datos introducidos son válidos, se crea el ámbito y

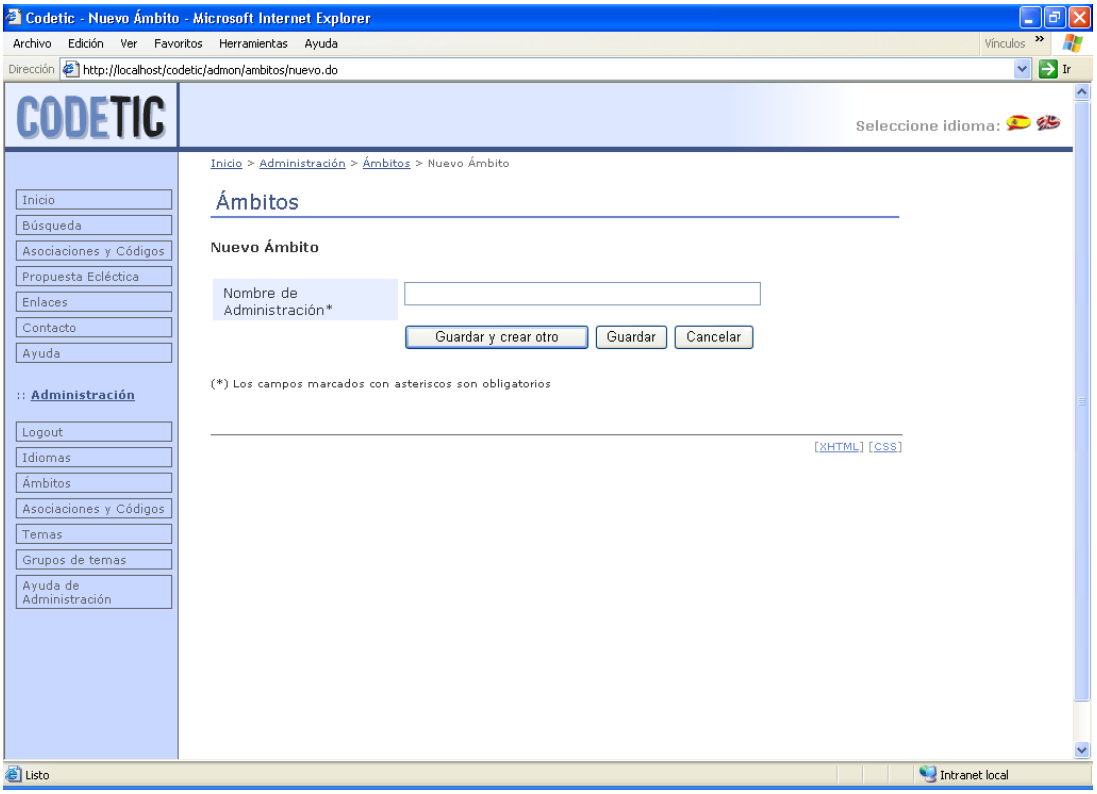

Figura 7.17: Página de creación de un nuevo ámbito.

se nos vuelve a mostrar de nuevo el formulario vacío para que introduzcamos los datos de un nuevo ámbito. Si no, haremos clic en el botón «Guardar», con lo que el ámbito se creará, en el caso de que sus datos sean válidos, y regresaremos a la página de administración de ámbitos. Existe otro botón llamado «Cancelar» que, como su propio nombre indica, cancela la operación, y nos envía a la página de administración de ámbitos.

Editar un ámbito

Para modificar los datos de un ámbito, se deberá pulsar el botón «Editar» del ámbito deseado desde la página de administración de ámbitos (Figura 7.16). La Figura 7.18 muestra la pantalla de edición de un ámbito.

Una vez hayamos modificado los datos, pulsaremos el botón «Guardar» para guardar los cambios. También podemos pulsar el botón «Cancelar» para cancelar la operación.

Eliminar un ámbito

Para eliminar un ámbito hay que pulsar en el botón «Eliminar» desde la pantalla de administración de ámbitos (Figura 7.16). Tras esto, nos aparecerá un mensaje por pantalla para que confirmemos la eliminación. Al eliminar un ámbito, todos los datos asociados a éste se eliminarán de la base de datos.

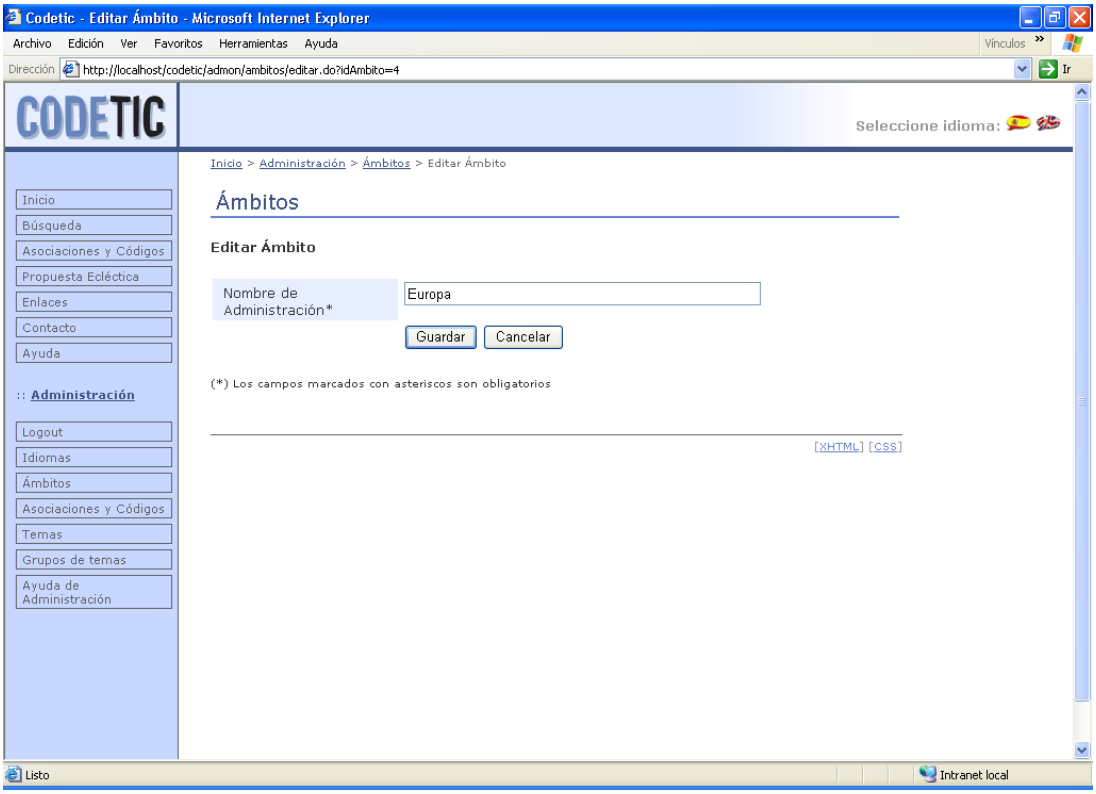

Figura 7.18: Página de edición de un ámbito.

7.3.5.1. Gestión de las traducciones de un ámbito

Es posible administrar las traducciones de un ámbito. Para ello, pulsaremos el botón «Traducciones» del ámbito deseado desde la pantalla de gestión de ámbitos (Figura 7.16).

La pantalla asociada a la gestión de las traducciones del ámbito «Europa» aparece en la Figura 7.19, en la que se muestra una tabla con las siguientes columnas:

- **Idioma**. Representa el idioma de la traducción.
- **Nombre**. Es el nombre del ámbito escrito en el idioma de la traducción.
- **Descripción**. Muestra la descripción del ámbito escrita en el idioma de la traducción.
- **Acciones**. Contiene las diversas operaciones que se pueden realizar sobre cada una de las traducciones:

  Editar una traducción (botón editar).
  Eliminar una traducción (botón eliminar).

Más arriba de la tabla tenemos dos enlaces: «Crear Nueva Traducción», para crear una nueva traducción, y «Volver», para volver a la página anterior.

A continuación vamos a explicar cada una de las operaciones que podemos realizar para administrar estas traducciones.

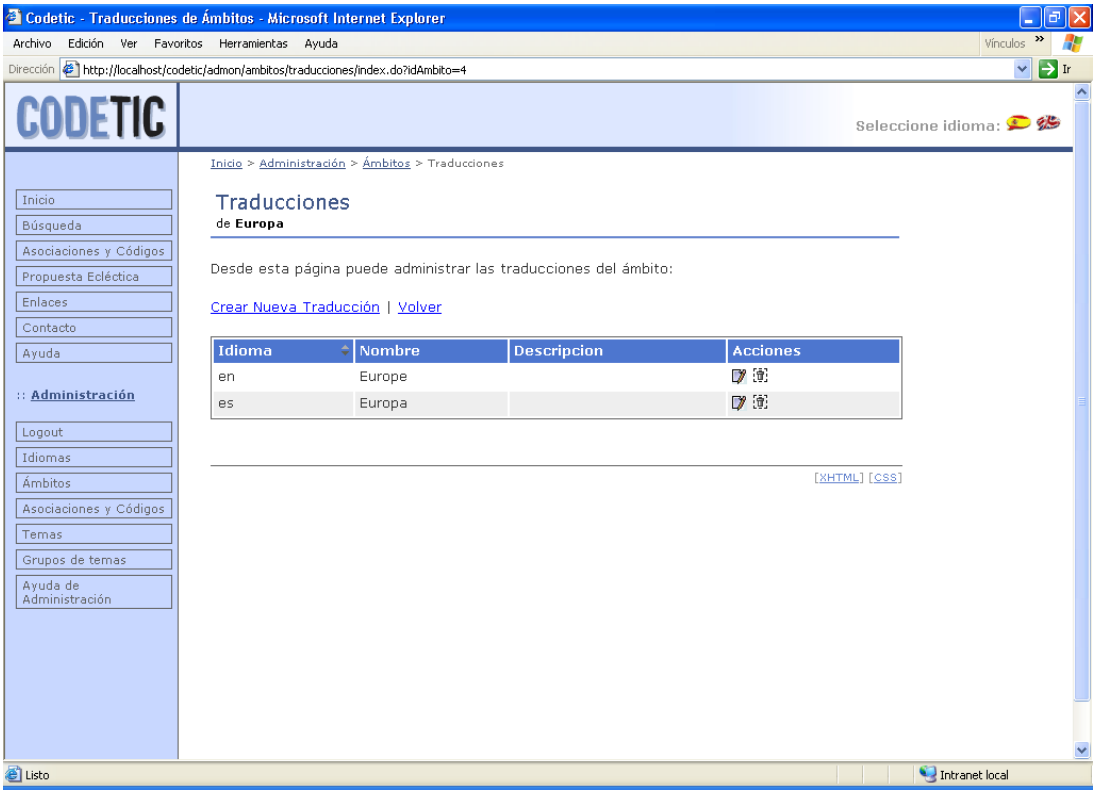

Figura 7.19: Página de administración de las traducciones de un ámbito.

Crear una nueva traducción

Lo primero que debemos hacer es pulsar el enlace «Crear Nueva Traducción» de la página de administración de traducciones del ámbito (Figura 7.19). Con ello, visualizaremos la página que se muestra en la Figura 7.20. A continuación, debemos introducir los datos de la traducción en el formulario que aparece en dicha página.

Los campos de este formulario son los siguientes:

- **Idioma**. Recoge el idioma de la traducción.
- **Nombre**. Es el nombre del ámbito escrito en el idioma de la traducción.
- **Descripción**. Muestra la descripción del ámbito escrito en el idioma de la traducción.

Una vez hayamos introducido los datos, pulsaremos en el botón «Guardar y crear otro» si deseamos seguir creando nuevas traducciones. En este caso, si los datos introducidos son válidos, se crea la traducción y se nos vuelve a mostrar de nuevo el formulario vacío para que introduzcamos los datos de una nueva traducción. Si no, haremos clic en el botón «Guardar». Con esto, se creará la traducción, siempre que sus datos sean válidos, y regresaremos a la página de administración de traducciones del ámbito. También existe un botón llamado «Cancelar» que nos envía directamente a la página de administración de traducciones del ámbito, cancelando así la operación de creación.

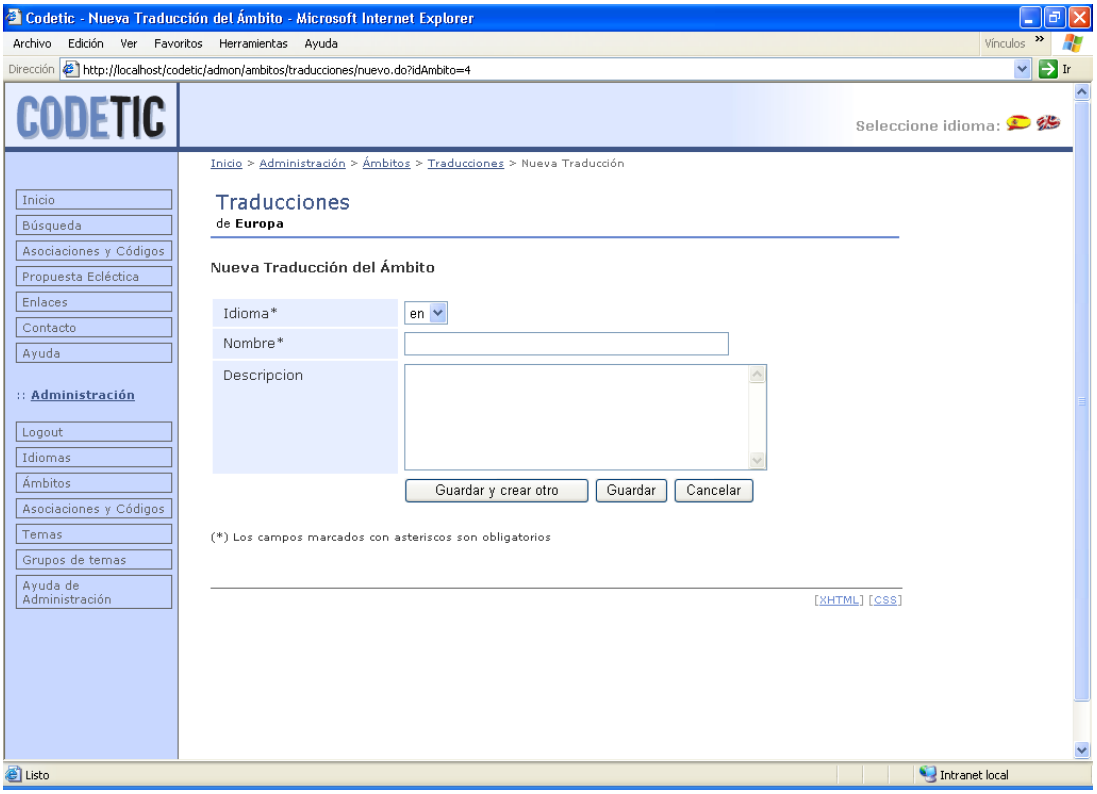

Figura 7.20: Página de creación de una nueva traducción de ámbito.

Editar una traducción

Si queremos editar la traducción de un ámbito, haremos clic en el botón «Editar» asociado a la traducción de dicho ámbito desde la página de administración de traducciones del ámbito (Figura 7.19). De esta manera, accederemos a la pantalla mostrada en la Figura 7.21.

El próximo paso consiste en modificar los datos que queramos, tras el cual pulsaremos el botón «Guardar» si deseamos que se lleven a cabo los cambios realizados, o el botón «Cancelar», para cancelar dichos cambios.

Eliminar una traducción

Para eliminar una traducción simplemente hay que pulsar en el botón «Eliminar» desde la pantalla de administración de las traducciones de ámbito (Figura 7.19). Seguidamente nos aparecerá un mensaje por pantalla para que confirmemos si realmente queremos eliminar dicha traducción.

7.3.6. Gestión de asociaciones

La Figura 7.22 muestra la pantalla de gestión de asociaciones. Para acceder a ella debemos hacer clic en la opción «Asociaciones y Códigos» del menú de administración o en el enlace del mismo nombre que aparece en la página de administración.

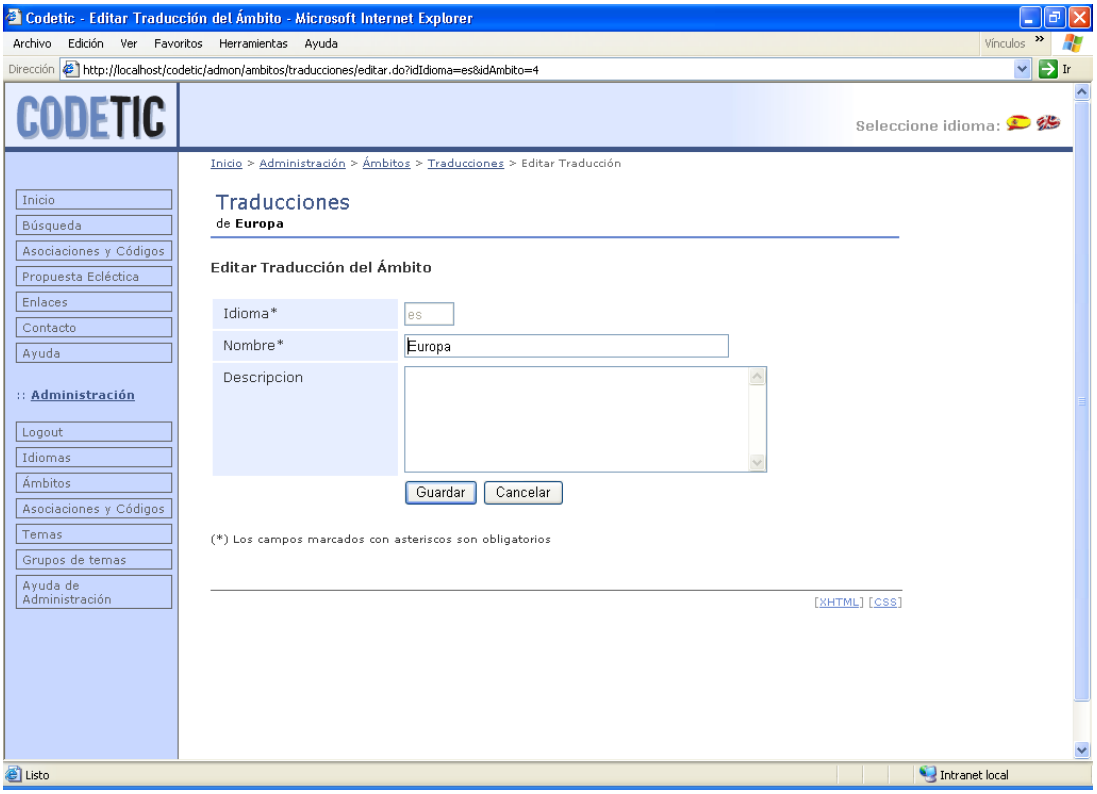

Figura 7.21: Página de edición de una traducción de ámbito.

Esta pantalla muestra una tabla con todas las asociaciones que existen en la base de datos. Las columnas de esta tabla son las siguientes:

- **Ámbito**. Representa el ámbito geográfico al que pertenece la asociación.
- **Nombre**. Es el nombre completo de la asociación escrito en el idioma «oficial» de ésta. Para las asociaciones españolas este nombre estará escrito en español, para las italianas en italiano, para las internacionales en inglés, etc.
- **Acciones**. Contiene las diversas acciones que se pueden realizar sobre cada una de las asociaciones:

   Editar una asociación (botón editar).
   Eliminar una asociación (botón eliminar).
   Administrar las traducciones de la asociación (botón traducciones).
   Administrar los códigos éticos de la asociación (botón códigos).

Además de la tabla, tenemos dos enlaces situados justo encima de ella. El primero, «Crear Nueva Asociación», nos lleva al formulario de creación de una nueva asociación. El segundo, «Volver», nos permite volver a la página anterior.

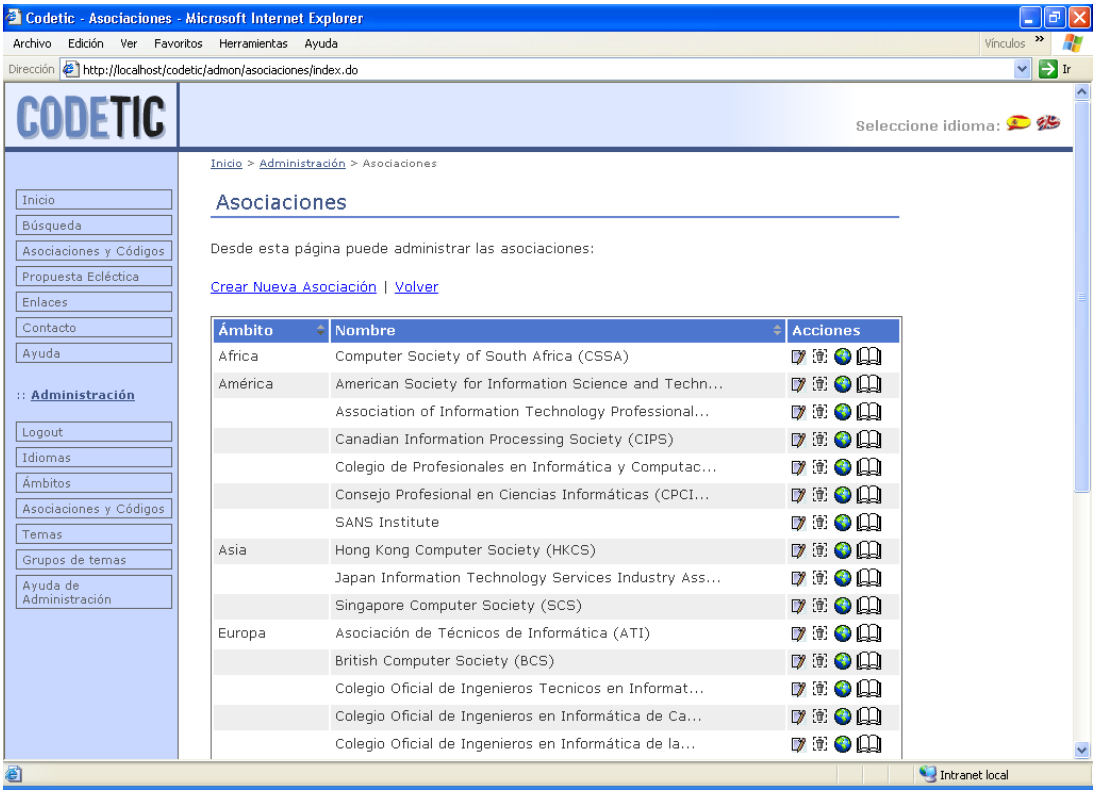

Figura 7.22: Página de administración de asociaciones.

Las operaciones que podemos realizar sobre las asociaciones son las mismas que las que podemos realizar sobre cualquier otra entidad: crear, editar y eliminar. Así, le remitimos al Apartado 7.3.5, donde se explica cómo se realizan todas estas operaciones para la entidad ámbito. Lo único que debe tener en cuenta es que los datos asociados a una asociación son distintos a los asociados a un ámbito. Veamos cada uno de los datos asociados a una asociación:

- **Ámbito**. Indica el ámbito geográfico al que pertenece la asociación.
- **Nombre**. Representa el nombre completo de la asociación escrito en el idioma «oficial» de ésta.
- **Siglas**. Contiene las siglas de la asociación.
- **Web**. Contiene la dirección URL del sitio web oficial de la asociación.
- **Año de Creacion**. Recoge el año en que se creó la asociación.

7.3.6.1. Gestión de las traducciones de una asociación

Para administrar las traducciones de una asociación, pulsaremos el botón «Traducciones» de la asociación deseada desde la pantalla de administración de asociaciones (Figura 7.22).

La pantalla asociada a la administración de las traducciones de la asociación ACM aparece en la Figura 7.23, en la que se muestra una tabla con las siguientes columnas:

- **Idioma**. Representa el idioma de la traducción.
- **Localización Geográfica**. Traducción de la zona geográfica cubierta por la asociación. Por ejemplo, «Nacional (España)» para el español, «National (Spain)» para el inglés, etc.
- **Descripción**. Muestra la descripción de la asociación escrita en el idioma de la traducción.
- **Acciones**. Contiene las distintas acciones que se pueden realizar sobre cada una de las traducciones:

   Editar una traducción (botón editar).

   Eliminar una traducción (botón eliminar).

Encima de esta tabla tenemos dos enlaces: «Crear Nueva Traducción», para crear una nueva traducción, y «Volver», para volver a la página anterior.

La forma de crear, editar y eliminar traducciones de asociaciones es la misma que para el resto de traducciones. En el Apartado 7.3.5.1 se explica cómo realizar todas estas acciones sobre las traducciones de ámbitos. Sólo debe tener en cuenta que cada tipo de traducción tiene asociados unos datos determinados. Para el caso de las traducciones de asociaciones, estos datos son los siguientes:

- **Idioma**. Representa el idioma de la traducción.
- **Localización Geográfica**. Traducción de la zona geográfica cubierta por la asociación. Por ejemplo, «Nacional (España)» para el español, «National (Spain)» para el inglés, etc.
- **Descripción**. Muestra la descripción de la asociación escrita en el idioma de la traducción.
- **Comentarios**. Sirve para añadir cualquier cosa sobre la asociación, que consideremos interesante, como puede ser el número de socios, la dirección postal, el director o presidente de la asociación, etc.

7.3.7. Gestión de códigos éticos

Para acceder a la página de gestión de los códigos éticos de una determinada asociación debemos hacer clic en el botón «Traducciones» de la asociación desde la página de gestión de asociaciones (Figura 7.22). Como ejemplo, la Figura 7.24 muestra esta página para la asociación ACM.

Esta página muestra una tabla con todos los códigos éticos de la asociación. Las columnas de esta tabla son:

- **Título Original**. Contiene el título del código escrito en su idioma original.

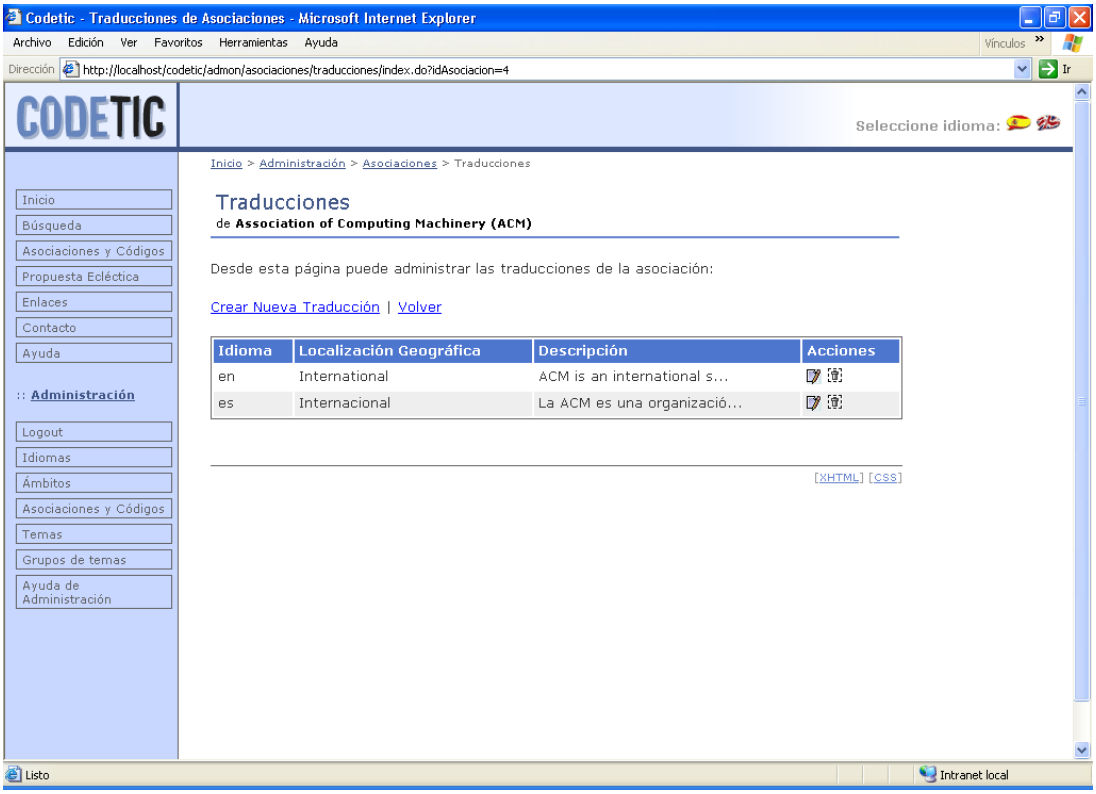

Figura 7.23: Página de administración de las traducciones de una asociación.

- **Año de Aprobación**. Contiene el año en que se creó (aprobó) el código ético por parte de la asociación.
- **Acciones**. Contiene las diversas acciones que se pueden realizar sobre cada uno de los códigos éticos:

  Editar un idioma (botón editar).

  Eliminar un idioma (botón eliminar).

  Administrar las traducciones del código (botón traducciones).

  Administrar las secciones del código (botón secciones).

Por encima de esta tabla tenemos dos enlaces. El primero, «Crear Nuevo Código», nos lleva al formulario de creación de un nuevo código. El segundo, «Volver», nos permite volver a la página anterior.

Para ver cómo crear, editar y eliminar códigos, le remitimos al Apartado 7.3.5, donde se explica cómo se realizan todas estas operaciones para los ámbitos. La forma de proceder es la misma en ambos casos. Lo único que debemos tener en cuenta es que, para los códigos, los campos de los formularios serán los siguientes:

- **Título Original**. Contiene el título del código escrito en su idioma original.
- **Web**. Recoge la dirección URL de la página web oficial del código ético.

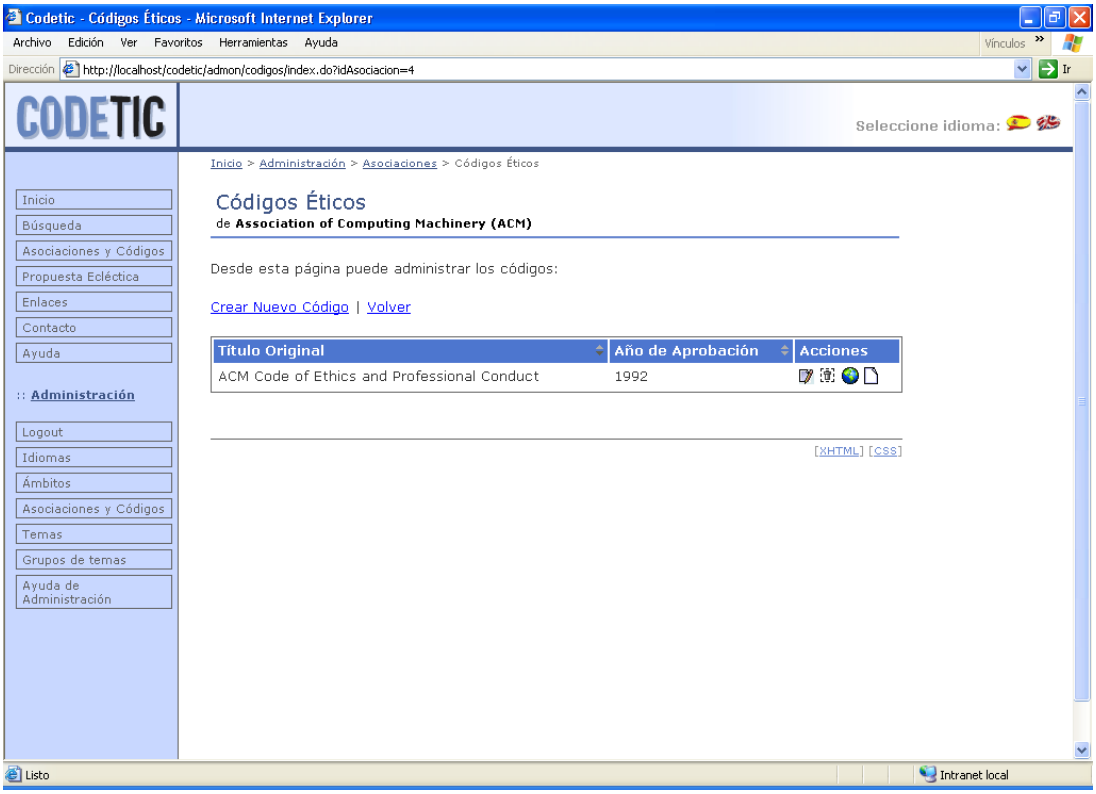

Figura 7.24: Página de administración de los códigos éticos de la asociación ACM.

- **Año de Aprobación**. Contiene el año en que se aprobó el código ético.

Si desea conocer cómo administrar las secciones que forman un código, consulte el Apartado 7.3.7.2. Si desea conocer cómo hemos organizado los códigos éticos, mire la Figura 6.18.

7.3.7.1. Gestión de las traducciones de un código

Para poder insertar traducciones de secciones de un código, antes tenemos que crear una nueva traducción del código. La forma de acceder a la página de administración de traducciones de un determinado código es pinchando en el botón «Traducciones» asociado a dicho código desde la página de administración de códigos (Figura 7.24).

La página de administración de las traducciones del código «ACM Code of Ethics and Professional Conduct», perteneciente a la asociación ACM, aparece en la Figura 7.25, en la que se muestra una tabla con las siguientes columnas:

- **Idioma**. Indica el idioma de la traducción.
- **Título**. Contiene el título traducido del código.
- **Traductores**. Autores de la traducción del código.
- **Acciones**. Contiene las diversas acciones que se pueden realizar sobre cada uno de las traducciones:

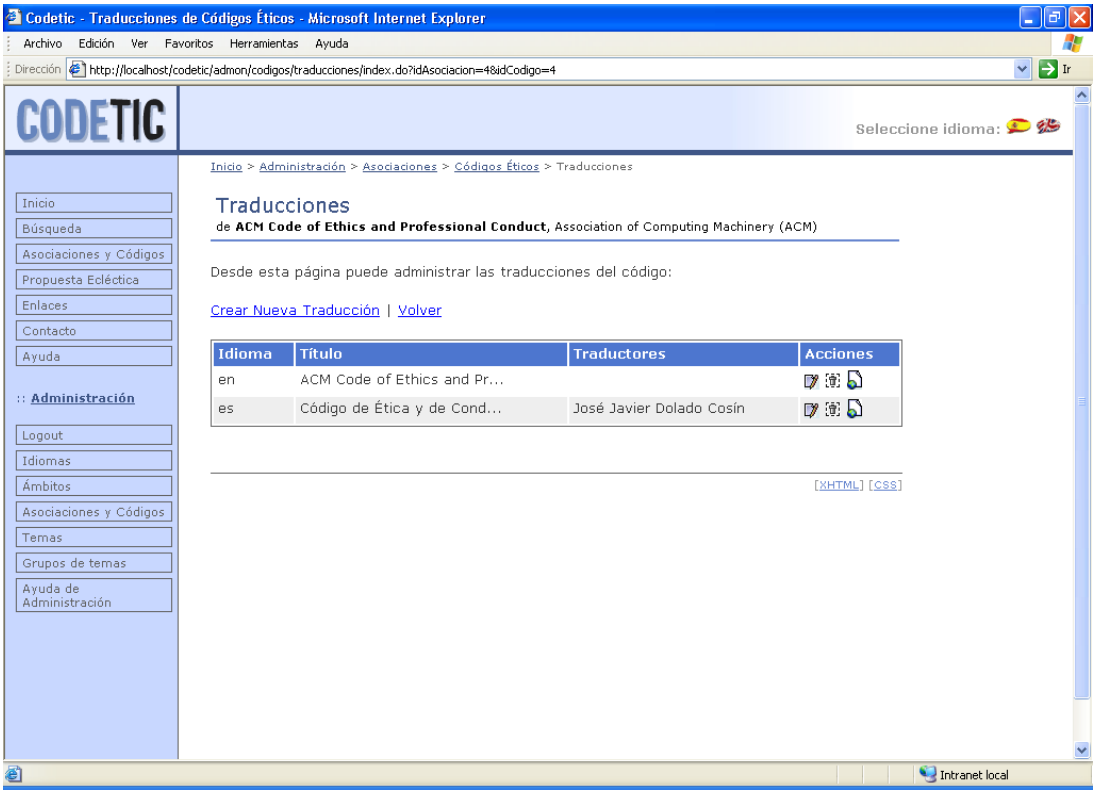

Figura 7.25: Página de administración de las traducciones del código «ACM Code of Ethics and Professional Conduct» perteneciente a la asociación ACM.

 Editar una traducción (botón editar).

 Eliminar una traducción (botón eliminar).

 Administrar las traducciones de las secciones asociadas a la traducción del código (botón secciones traducidas).

Sobre esta tabla tenemos dos enlaces: «Crear Nueva Traducción», que nos lleva al formulario de creación de una nueva traducción, y «Volver», que nos permite volver a la página anterior.

Para ver cómo crear, editar y eliminar traducciones de código, le remitimos al Apartado 7.3.5.1, donde se explica cómo se realizan todas estas operaciones para las traducciones de ámbitos. La forma de proceder es la misma en ambos casos. Lo único que debemos tener en cuenta es que, para las traducciones de código, los campos de los formularios serán los siguientes:

- **Idioma**. Es el idioma de la traducción.
- **Título**. Título del código.
- **Texto Cabecera**. Texto que está situado entre el título del código y la primera sección. Lo normal es que contenga subtítulos, dedicatorias, etc.

- **Texto Pié**. Texto que va al final del código, inmediatamente después de la última sección. Normalmente trata sobre la autoría del código, posibles reconocimientos a personas o instituciones, la fecha de adopción del código, etc.
- **Traductores**. Autores de la traducción del código.
- **Comentarios de los Traductores**. Comentarios sobre la traducción del código.
- **Web de la Traducción**. Página web de la traducción del código.

7.3.7.2. Gestión de las secciones de un código

Para administrar las secciones de un código, pulsaremos el botón «Secciones» del código deseado desde la pantalla de administración de códigos (Figura 7.24).

La pantalla asociada a la administración de las secciones del tema «ACM Code of Ethics and Professional Conduct», perteneciente a la asociación ACM, la podemos encontrar en la Figura 7.26. En ella se muestra una tabla con las siguientes columnas:

- **Nivel**. Indica todos los niveles a los que pertenece la sección. Se representa mediante una cadena formada por los niveles separados por un punto «.». De esta forma, si almacena «3.10.6», indica que se trata del artículo 6 de la subsección 10 de la sección 3. Este atributo nos permite establecer una jerarquía dentro de las secciones de un código (Figura 6.17). Así, tendremos secciones de primer nivel, con nivel=1,2..., de segundo nivel, con nivel=1.1,1.2..., etc.
- **Nombre de Administración**. Contiene un nombre que representa, independientemente del idioma, a una sección determinada y es utilizado para la administración de Codetic. Normalmente coincidirá con el título de la sección. Algunos posibles valores de este atributo son «Preámbulo», «Principio 1», etc.
- **Acciones**. Contiene las distintas acciones que se pueden realizar sobre cada una de las secciones:

   Editar una sección (botón editar).
   Eliminar una sección (botón eliminar).
   Administrar las traducciones de la sección (botón traducciones).

Encima de esta tabla tenemos un enlace que nos permite crear nuevas secciones, llamado «Crear Nueva Sección», y otro para volver a la página anterior, llamado «Volver».

Las acciones que podemos realizar sobre las secciones son las mismas que las que podemos realizar sobre cualquier otra entidad: crear, editar y eliminar. Así, le remitimos al Apartado 7.3.5, donde se explica cómo se realizan todas estas operaciones para la entidad ámbito. La

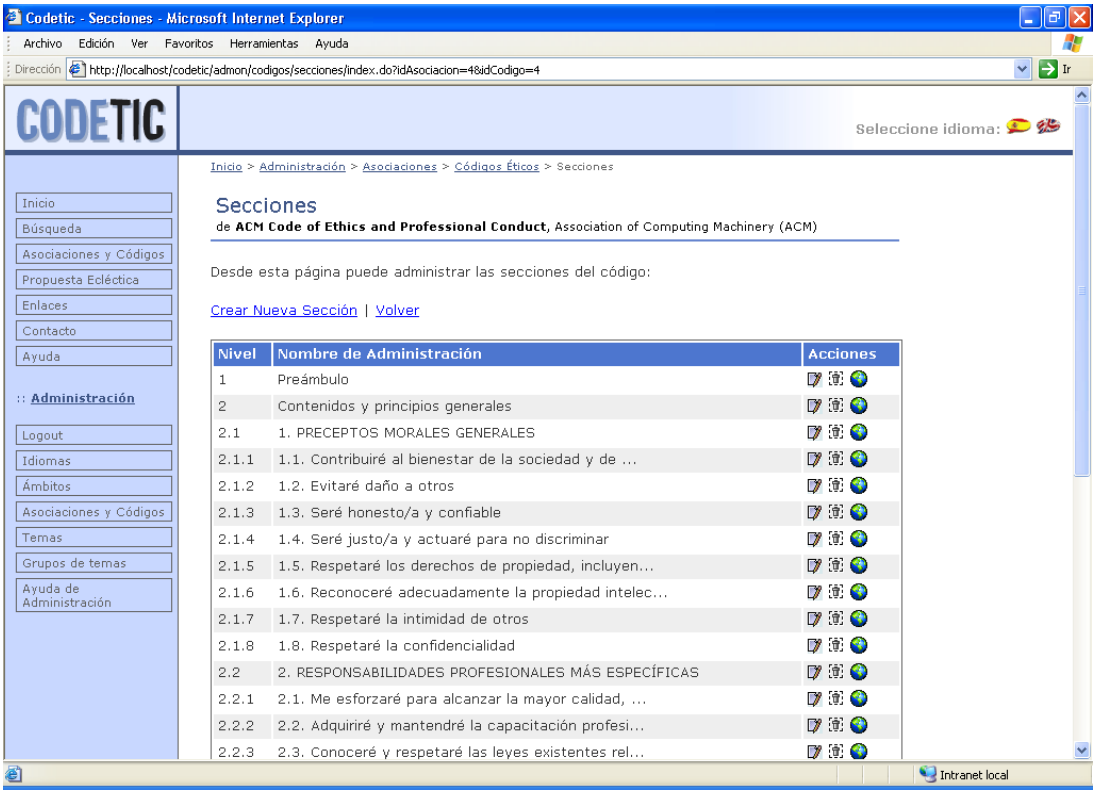

Figura 7.26: Página de administración de las secciones del código «ACM Code of Ethics and Professional Conduct», perteneciente a la asociación ACM.

única cosa que debe tener en cuenta es que, para las secciones, los formularios de creación y edición tendrán los siguientes campos:

- **Nivel**. Como hemos visto anteriormente, el valor de este campo muestra todos los niveles a los que pertenece la sección.
- **Nombre de Administación**. Es el nombre usado para la administración de la sección.
- **Temas**. Está formado por un conjunto de casillas de verificación que podemos señalar para asociar uno o más temas a la sección (véase el formulario de creación de una sección, Figura 7.27).

7.3.7.3. Gestión de las traducciones de una sección

Haciendo clic sobre el botón «Traducciones» asociado a una sección (Figura 7.26), accederemos a la página de administracción de sus traducciones (Figura 7.28).

Entonces, veremos una tabla con las siguientes columnas:

- **Idioma**. Muestra el idioma de la traducción.
- **Título**. Título «natural» de la sección. Es el título que tiene la sección en el código. Pueden existir secciones sin título.

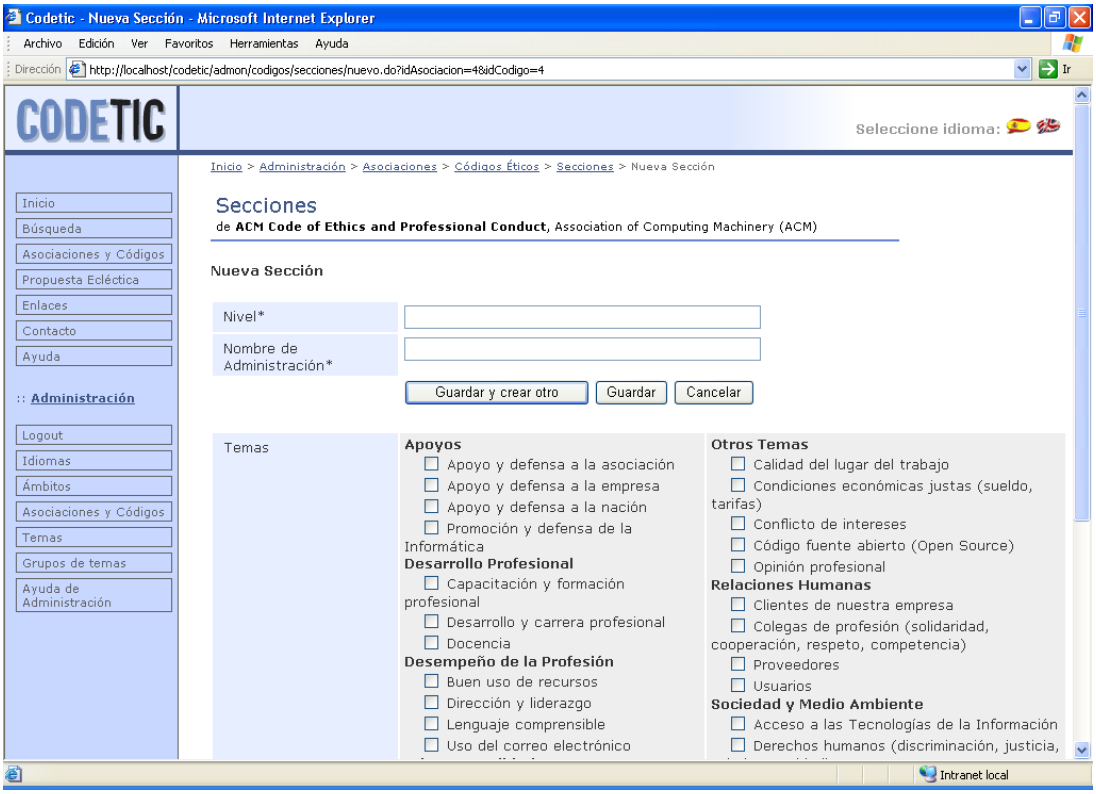

Figura 7.27: Página de creación de una nueva sección del código «ACM Code of Ethics and Professional Conduct», perteneciente a la asociación ACM.

- **Seudotítulo**. Título «artificial» de la sección. Cuando una sección no tiene título, es necesario darle uno. Este es el título que se utiliza para las secciones sin título «natural» cuando se muestran los resultados de una búsqueda.

- **Texto**. Texto de la sección.

- **Acciones**. Contiene las diversas operaciones que se pueden realizar sobre cada una de las traducciones:

   Editar una traducción (botón editar).

   Eliminar una traducción (botón eliminar).

Más arriba de la tabla tenemos dos enlaces: «Crear Nueva Traducción», para crear una nueva traducción, y «Volver», para volver a la página anterior.

Si desea conocer cómo crear, editar y eliminar traducciones de secciones, consulte el Apartado 7.3.5.1. Allí se explican cómo realizar estas operaciones sobre las traducciones de ámbitos, cuyo funcionamiento es el mismo que para cualquier tipo de traducción. Los campos de los formularios asociados a una traducción de sección, la mayoría de los cuales ya se han descrito, son los siguientes:

- **Idioma**. Es el idioma de la traducción.

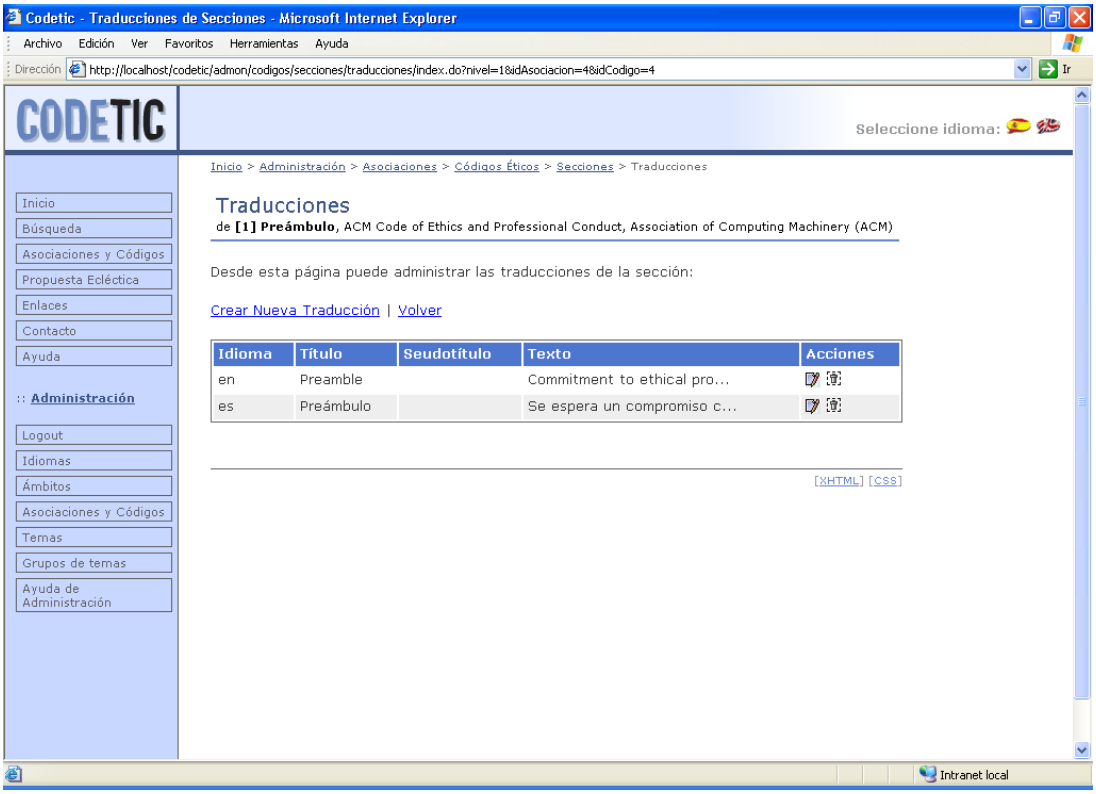

Figura 7.28: Página de administración de las traducciones de una sección.

- **Título**. Título «natural» de la sección.
- **Seudotítulo**. Título «artificial» de la sección.
- **Texto**. Texto de la sección. Puede incluir etiquetas HTML como <b>, <i>...
- **Temas**. Está formado por un conjunto de casillas de verificación que podemos señalar para asociar uno o más temas a la sección.

Aparte de esta, existe otra forma de crear, editar y eliminar traducciones de secciones que es explicada en el Apartado 7.3.7.4.

7.3.7.4. Gestión de las traducciones de secciones asociadas a una traducción de código

Codetic ofrece dos formas distintas de gestionar las traducciones de las secciones. La primera de ellas es la que hemos visto en el Apartado 7.3.7.3 y nos permite crear, modificar y eliminar las traducciones disponibles para una sección determinada. La segunda es la que vamos a tratar aquí y permite traducir de forma centralizada todas las secciones de un código.

Cada traducción de código tiene una página de administración propia desde la que podemos comprobar si existe alguna sección sin traducir a dicho idioma. Para acceder a esta página (Figura 7.29), debemos hacer clic sobre el botón «Secciones Traducidas» de la traducción correspondiente (Figura 7.25).

Con esto, veremos una tabla donde cada registro se corresponde con la traducción, en el idioma de la traducción del código, de cada una de las secciones del código. Las columnas de esta tabla son las siguientes:

- **Nivel**. Indica todos los niveles a los que pertenece la sección. Para una descripción más detallada de este campo, consulte el Apartado 7.3.7.2.
- **Título**. Título «natural» de la sección. Es el título que tiene la sección en el código. Pueden existir secciones sin título.
- **Seudotítulo**. Título «artificial» de la sección. Cuando una sección no tiene título, es necesario darle uno. Este es el título que se utiliza para las secciones sin título «natural» cuando se muestran los resultados de una búsqueda.
- **Texto**. Texto de la sección.
- **Acciones**. Contiene las diversas operaciones que se pueden realizar sobre cada una de las traducciones:

   Editar una traducción (botón editar).
   Eliminar una traducción (botón eliminar).

Esta tabla, además, tiene una cabecera o título donde se indica el número de secciones que quedan por traducir. En caso de que quede alguna, podremos traducirlas de forma individual haciendo clic sobre el enlace «Traducir» que aparece en la fila correspondiente a la sección no traducida. También tendremos la posibilidad de traducirlas todas secuencialmente gracias al enlace «Traducir las secciones no traducidas», situado encima de la tabla.

Si lo que queremos es volver a la página anterior, haremos clic en «Volver».

Para ver cómo se crean, editan y eliminan traducciones de secciones, consulte el Apartado 7.3.5.1. Allí se explican cómo realizar estas operaciones sobre las traducciones de ámbitos, cuyo funcionamiento es el mismo que para cualquier tipo de traducción. Los campos de los formularios asociados a una traducción de sección, la mayoría de los cuales ya se han descrito, son los siguientes:

- **Nivel**. Indica todos los niveles a los que pertenece la sección.
- **Título**. Título «natural» de la sección.
- **Seudotítulo**. Título «artificial» de la sección.
- **Texto**. Texto de la sección. Puede incluir etiquetas HTML como <b>, <i>...
- **Temas**. Está formado por un conjunto de casillas de verificación que podemos señalar para asociar uno o más temas a la sección.

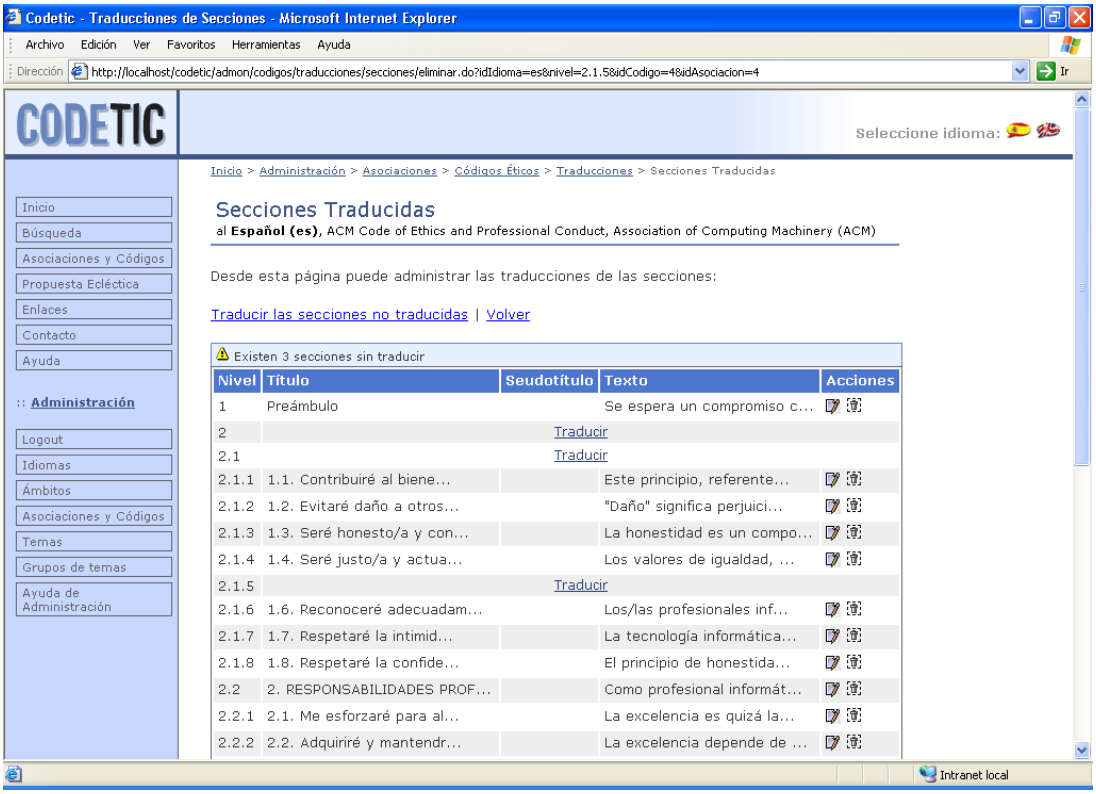

Figura 7.29: Página de administración de las traducciones de secciones asociadas a una traducción de código en la que quedan secciones por traducir.

7.3.7.5. Cómo insertar la traducción completa de un código

De forma resumida, los pasos que debemos seguir para insertar la traducción completa de un código son los siguientes:

1. Crear un nuevo código (Apartado 7.3.7).
2. Crear las secciones de dicho código (Apartado 7.3.7.2)
3. Crear la traducción del código (Apartado 7.3.7.1).
4. Traducir todas las secciones del código (Apartado 7.3.7.4).

7.3.8. Gestión de temas

Los temas son otras de las entidades que podemos gestionar como administradores. Para acceder a la página de gestión de temas (Figura 7.30), podemos hacer clic en la opción «Temas» del menú de administración o en el enlace «Temas» de la página principal de administración.

La página de administración de temas muestra una tabla con todos los temas que existen en la base de datos. Las columnas de esta tabla son las siguientes:

- **Grupo**. Indica el grupo de temas al cual pertenece el tema.

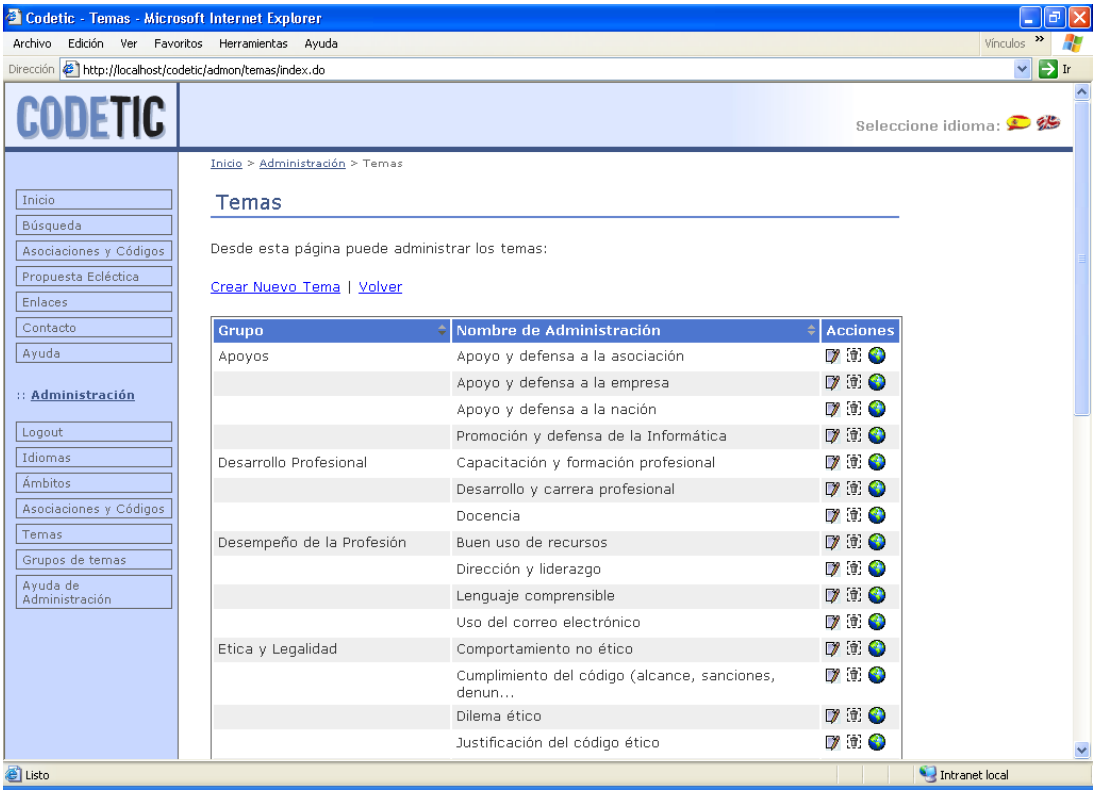

Figura 7.30: Página de administración de temas.

- **Nombre de Administración**. Es el nombre usado para la administración del tema.
- **Acciones**. Muestra un botón por cada una de las diversas acciones que se pueden realizar sobre cada uno de los temas:

   Editar un tema (botón editar).

   Eliminar un tema (botón eliminar).

   Administrar las traducciones del tema (botón traducciones).

Para ver cómo crear, editar y eliminar temas, le remitimos al Apartado 7.3.5, donde se explica cómo se realizan todas estas operaciones para los ámbitos. La forma de proceder es la misma en ambos casos. Lo único que debemos tener en cuenta es que, para los temas, los campos de los formularios serán los siguientes:

- **Grupo**. Contiene el grupo de temas al que se encuentra asociado el tema.
- **Nombre de Administración**. Este campo da nombre, independientemente del idioma, al tema y es utilizado para la administración de Codetic.

7.3.8.1. Gestión de las traducciones de los temas

Para administrar las traducciones de un tema, pulsaremos el botón «Traducciones» del tema deseado desde la pantalla de administración de temas (Figura 7.30).

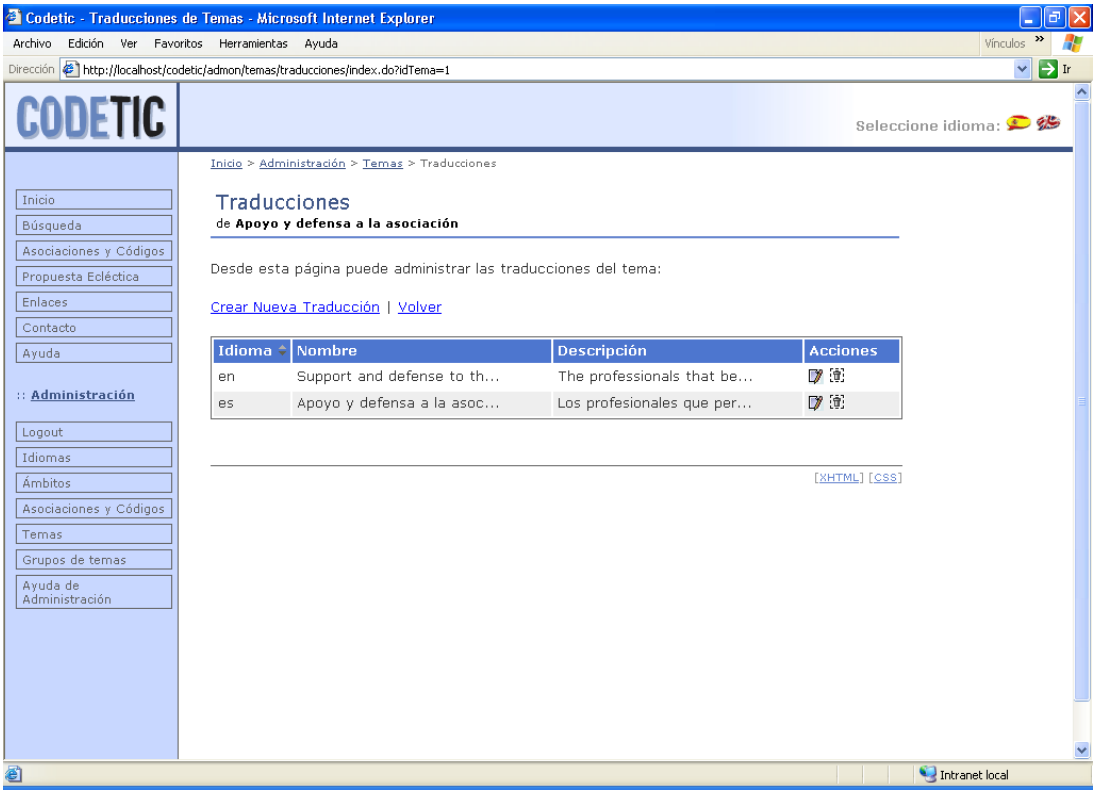

Figura 7.31: Página de administración de las traducciones de un tema.

La pantalla asociada a la administración de las traducciones del tema «Apoyo y defensa a la asociación» aparece en la Figura 7.31, en la que se muestra una tabla con las siguientes columnas:

- **Idioma**. Representa el idioma de la traducción.
- **Nombre**. Es el nombre del tema escrito en el idioma de la traducción.
- **Descripción**. Muestra la descripción del tema escrita en el idioma de la traducción.
- **Acciones**. Contiene las distintas acciones que se pueden realizar sobre cada una de las traducciones:

   Editar una traducción (botón editar).

   Eliminar una traducción (botón eliminar).

Encima de esta tabla tenemos dos enlaces: «Crear Nueva Traducción», para crear una nueva traducción, y «Volver», para volver a la página anterior.

La forma de crear, editar y eliminar traducciones de temas es la misma que para el resto de traducciones. En el Apartado 7.3.5.1 se explica cómo realizar todas estas acciones sobre las traducciones de ámbitos. La única cosa que debe tener en cuenta es que, para los temas, los formularios de creación y edición tendrán los siguientes campos:

- **Idioma**. Recoge el nombre de la traducción.
- **Nombre**. Es el nombre del tema escrito en el idioma de la traducción.
- **Descripción**. Muestra la descripción del tema escrita en el idioma de la traducción.
- **Palabras Clave**. Lista de palabras, términos, frases, etc. que ayudan a describir y definir el tema.

7.3.9. Gestión de grupos de temas

Para acceder a la pantalla de administración de grupos de temas (Figura 7.32) debemos hacer clic en la opción «Grupos de temas» del menú de administración o en el enlace «Grupos de temas» que aparece en la página de administración.

Esta página muestra una tabla con todos los grupos de temas de la base de datos. Esta tabla tiene las siguientes columnas:

- **Nombre de Administración**. Es el nombre que representa, independientemente del idioma, a un grupo determinado.
- **Acciones**. Contiene las diversas acciones que se pueden realizar sobre cada uno de los grupos de temas

   Editar un grupo de temas (botón editar).
   Eliminar un grupo de temas (botón eliminar).
   Administrar las traducciones del grupo de temas (botón traducciones).

Aparte de esta tabla, tenemos dos enlaces situados justo encima de ella. El primero, «Crear Nuevo Ámbito», nos lleva al formulario de creación de un nuevo grupo de temas. El segundo, «Volver», nos permite volver a la página anterior.

Las operaciones que podemos realizar sobre los grupos de temas son las mismas que las que podemos realizar sobre cualquier otra entidad: crear, editar y eliminar. Así, le remitimos al Apartado 7.3.5, donde se explica cómo se realizan todas estas operaciones para la entidad ámbito. Lo único que debe tener en cuenta es que los datos asociados a un grupo de temas son distintos a los asociados a un ámbito. El único dato asociado a los grupos de temas es el siguiente:

- **Nombre de Administración**. Contiene un nombre que representa, independientemente del idioma, al grupo de temas. Es utilizado para la administración de Codetic.

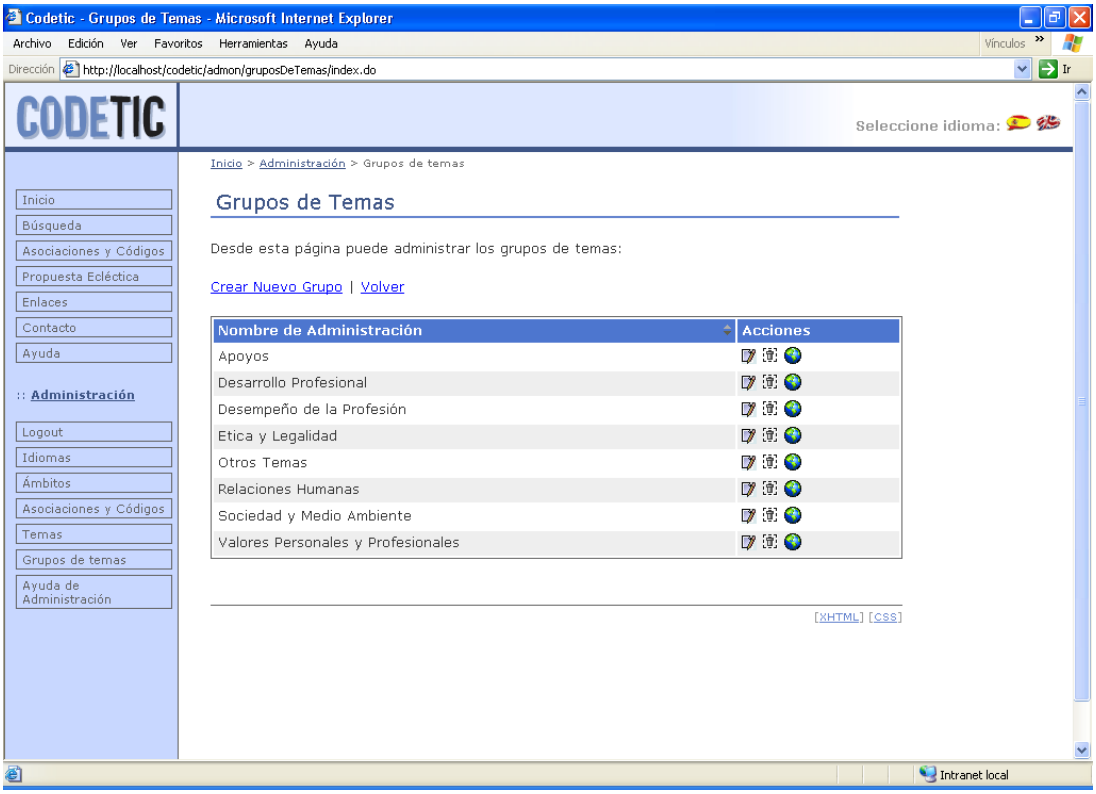

Figura 7.32: Página de administración de los grupos de temas.

7.3.9.1. Gestión de las traducciones de los grupos de temas

Es posible administrar las traducciones de un grupo de temas. Para ello, pulsaremos el botón «Traducciones» del grupo de temas cuyas traducciones queramos gestionar, desde la pantalla de gestión de grupos (Figura 7.32).

La pantalla asociada a la gestión de las traducciones del grupo de temas «Desarrollo Profesional» aparece en la Figura 7.33, en la que se muestra una tabla con las siguientes columnas:

- **Idioma**. Representa el idioma de la traducción.
- **Nombre**. Es el nombre del grupo escrito en el idioma de la traducción.
- **Descripción**. Muestra la descripción del grupo escrita en el idioma de la traducción.
- **Acciones**. Contiene las diversas operaciones que se pueden realizar sobre cada una de las traducciones:

   Editar una traducción (botón editar).

   Eliminar una traducción (botón eliminar).

Más arriba de la tabla tenemos dos enlaces: «Crear Nueva Traducción», para crear una nueva traducción, y «Volver», para volver a la página anterior.

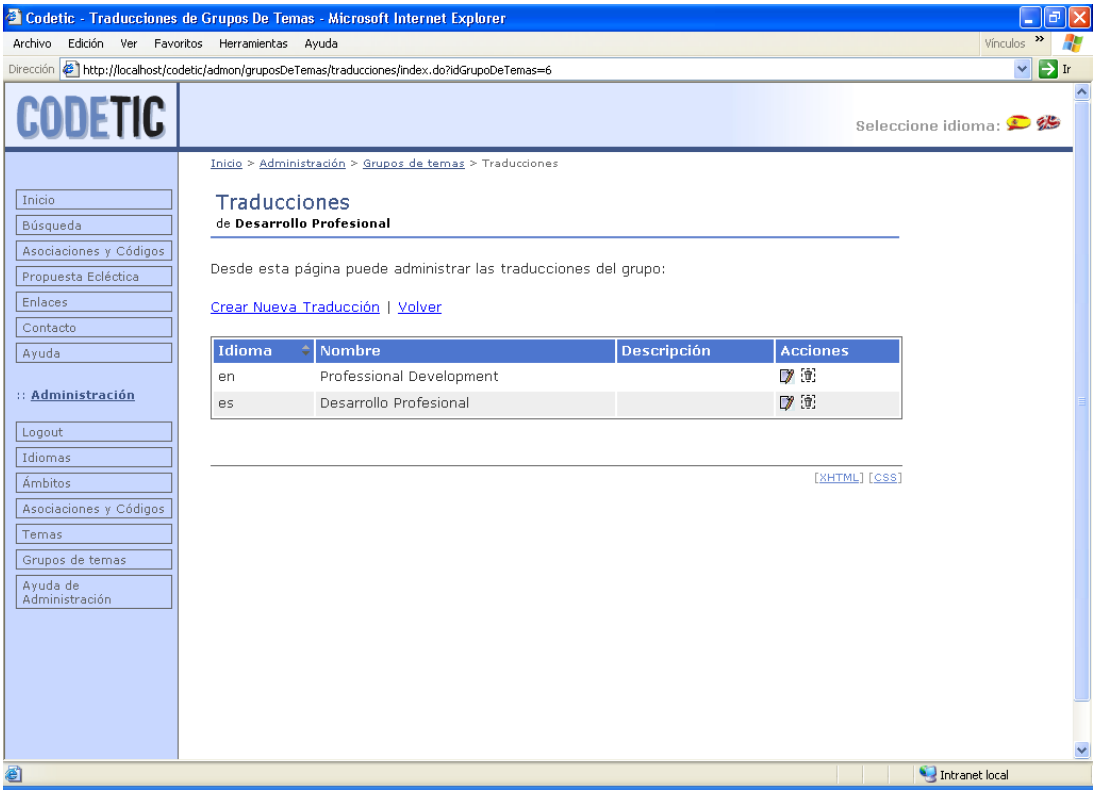

Figura 7.33: Página de administración de las traducciones de un grupo de temas.

Si desea conocer cómo crear, editar y eliminar traducciones de grupos de temas, consulte el Apartado 7.3.5.1. Allí se explican cómo realizar estas operaciones sobre las traducciones de ámbitos, pero su funcionamiento es igual que para cualquier tipo de traducción. Los campos de los formularios asociados a una traducción de grupo son los siguientes:

- **Idioma**. Recoge el idioma de la traducción.
- **Nombre**. Es el nombre del grupo escrito en el idioma de la traducción.
- **Descripción**. Muestra la descripción del grupo escrito en el idioma de la traducción.

Capítulo 8

Propuesta ecléctica

El presente código ético forma parte del Proyecto Final de Carrera titulado «Base de Datos Web sobre Códigos Éticos para Profesionales de la informática y una Propuesta Ecléctica», realizado por Miguel Ángel Serna Mata y dirigido por José Galindo Gómez, de la E.T.S. de Ingeniería informática de la Universidad de Málaga. Con este texto se ha pretendido establecer una serie de normas generales de conducta profesional y ética que pretende servir como guía de actuación para los profesionales de la informática.

Este código está creado básicamente a partir de un documento, el Código Deontológico del Colegio Oficial de Ingeniería en informática de Cataluña, aunque también recoge principios de las normativas deontológicas de otras veintisiete asociaciones informáticas, como la ACM (*Association of Computing Machinery*), el IEEE (*Institute of Electrical and Electronics Engineers*), etc. Asimismo se han consultado las pautas ambientales de la asociación PEO (*Professional Engineers Ontario*)[57].

Para que el código sea más manejable, las normas de conducta del código han sido divididas en nueve apartados. Los títulos de los apartados y su definición serían los siguientes:

1. Profesional o empresa ante la sociedad: Obligaciones de las personas físicas o jurídicas que prestan servicios en el área de la informática, cuando su actuación pueda afectar la sociedad en general o a las instituciones que la representan.

2. Profesional o empresa ante el cliente: Normas que tienen que guiar el ejercicio profesional de la informática con el fin de garantizar un buen servicio al cliente o usuario.

3. Profesional, trabajador o gestor, ante su empresa: Delimita las obligaciones del trabajador hacia su empresa, independientemente de las responsabilidades o del lugar que ocupe en su organigrama, al margen de les obligaciones que se deriven del contrato laboral.

4. Profesional ante los compañeros de profesión: Establece las pautas de comportamiento correcto entre profesionales del sector de la informática.

5. Responsabilidad personal del profesional: Esfuerzos de superación personal y mejora que debe hacer todo buen profesional de la informática.

6. Profesional ante el producto o servicio: Identifica cuál es la actitud correcta que debe tener el profesional de la informática a la hora de prestar un servicio o elaborar un producto, informático en ambos casos.

7. Empresa o gestor ante el profesional empleado: Conducta apropiada que tienen que seguir las organizaciones y personas, del sector de la informática, hacia los profesionales que se puedan ver afectados por sus decisiones, aparte de los compromisos laborales.

8. Empresa ante las otras empresas del sector: Normas de conducta que las empresas del sector de la informática han de observar para que la necesaria competencia se mantenga dentro de los parámetros correctos de lealtad y respeto mutuo.

9. Profesional o empresa ante un proveedor: Reglas que debe respetar el profesional o empresa del sector de la informática en sus relaciones con sus proveedores.

8.1. Profesional o empresa ante la sociedad

8.1.1. Hacer prevalecer los intereses generales por delante de los intereses particulares.

8.1.2. Velar porque el uso, aplicación y consecuencias de la aplicación de la informática contribuyan al bienestar de la sociedad, la humanidad y el medio ambiente.

8.1.3. Promover los principios de reducir, reutilizar y reciclar (ley de las tres erres) en referencia con la informática.

8.1.4. Dar a conocer a las autoridades públicas los impactos ambientales y las condiciones que afectan a la seguridad y el bienestar públicos.

8.1.5. Solicitar un informe de impacto ambiental cuando haya dudas de las consecuencias medioambientales del proyecto.

8.1.6. Tener en cuenta los impactos ambientales y la capacidad sustentable, así como los factores sociales, culturales y económicos.

8.1.7. Esforzarse por conservar recursos, materiales, energía, etc., y reducir al mínimo los impactos.

8.1.8. Tener en cuenta el manejo de residuos (material obsoleto o roto sin solución), incluso la prevención o reducción de su formación y su eliminación final. Por ejemplo: ordenadores viejos, impresoras, cartuchos de tinta, etc. Si el material obsoleto funciona debe donarse a organismos que le puedan sacar provecho (ONGs...). Se debe reutilizar el material fungible (cartuchos de tinta...), reciclar lo que se pueda y deshacerse del resto en los llamados «puntos limpios» o bien a través del distribuidor.

8.1.9. Trabajar con otros profesionales y organismos, incluso con otras disciplinas para mejorar la comprensión y las políticas ambientales.

8.1.10. Promover la accesibilidad de la informática por todos los niveles y estamentos de la sociedad.

8.1.11. Tratar a todo el mundo de manera justa, sin discriminar a nadie por razones de edad, discapacidad, género, orientación sexual, religión, raza, nacionalidad, u otros factores.

8.1.12. Promover este código deontológico en la sociedad, y en general, que se conozcan y respeten los códigos deontológicos que sean de aplicación o, en su defecto, los principios generales de éstos.

8.1.13. Informarse, en la medida de sus posibilidades, sobre el comportamiento deontológico de sus proveedores, clientes y de otros interlocutores, y priorizar aquellos que cumplan este código o el que les sea de aplicación.

8.1.14. Velar para que en el entorno de trabajo no se infrinja la legalidad vigente, en especial, las leyes que afecten al ejercicio de la profesión, se respeten los derechos humanos y se aplique el código deontológico que corresponda.

8.1.15. Colaborar en la identificación y conocimiento público de las buenas y malas prácticas en la informática.

8.1.16. Denunciar públicamente que un producto o servicio puede violar de modo evidente los derechos humanos, la legislación y/o este código.

8.1.17. Es un derecho y a la vez un deber guardar el secreto profesional y la información confidencial de la que se pueda disponer. Se tendrán que tomar las medidas necesarias para evitar que pueda ser revelada a terceros no autorizados, ni accidental ni voluntariamente.

8.1.18. Respetar las leyes de propiedad intelectual (derechos de autor, patentes, marcas y secretos comerciales). Una persona no debe atribuirse el mérito del trabajo o ideas de otros, incluso en los casos en los que no han sido explícitamente protegidos, por ejemplo, con derechos de autor o patente.

8.1.19. Practicar y fomentar la colaboración profesional, trabajando en equipo, ayudando y enseñando a los demás a hacer mejor su trabajo.

8.1.20. Ofrecer una crítica constructiva cuando los resultados no sean los esperados y expresar la satisfacción por el trabajo bien hecho.

8.1.21. Considerar con respeto todas las profesiones.

8.1.22. A la hora de ofrecer sus opiniones profesionales, expresarse con la claridad y lenguaje más apropiado a sus interlocutores.

8.1.23. No ofrecer opiniones profesionales en aquellos temas en los que no se sienta debidamente cualificado.

8.1.24. Ser objetivo, veraz, preciso e imparcial en las actuaciones y opiniones profesionales.

8.1.25. Informar de todos los efectos, tanto de los positivos como de los negativos, que tiene la implantación de un producto o servicio relacionado con la informática.

8.2. Profesional o empresa ante el cliente

8.2.1. Estar al corriente de las leyes relacionadas con la informática y promover su conocimiento y cumplimiento por parte del cliente.

8.2.2. Mantener confidencialidad con respecto a la información extraída de su relación, excepto si es para el mismo cliente, para uso interno o estadístico, siempre y cuando no se vulneren las voluntades expresadas por el cliente.

8.2.3. Respetar los derechos de autor de los productos o servicios residentes en los equipos informáticos del cliente.

8.2.4. Cumplir los compromisos adquiridos con el cliente, de modo diligente y aplicando las metodologías adecuadas en cada caso.

8.2.5. Hacerse responsable de las propias decisiones y de sus consecuencias.

8.2.6. No querer vender aquello que el cliente no necesita o no se adecúa a sus necesidades.

8.2.7. Aconsejar al cliente de modo objetivo sin anteponer intereses propios.

8.2.8. Informar de la necesidad de formación sobre los servicios o productos ofrecidos y facilitarla si la empresa cliente lo solicita.

8.2.9. Actuar desinteresadamente a la hora de valorar y contratar ofertas en nombre del cliente, pensando en criterios de negocio y de ética profesional y valorando también la calidad y metodología de las soluciones ofrecidas.

8.2.10. Ofrecer precios justos por productos o servicios que se presenten a la empresa cliente.

8.2.11. No excederse de presupuestos ya pactados si no es por causa justificada.

8.2.12. No infringir castigo o coacción al prestar servicios o productos presentes o futuros, necesarios para la empresa cliente, aprovechándose de posiciones dominantes.

8.2.13. Establecer contratos claros, de compra-venta o de prestación de servicios, que faciliten el entendimiento entre el cliente y el proveedor.

8.2.14. Ser objetivo y veraz sobre la experiencia profesional de los miembros del equipo de trabajo ofrecidos al cliente.

8.2.15. Facilitar un entorno de trabajo correcto a los trabajadores de la empresa cliente, al mismo nivel que los trabajadores de la propia empresa cuando el cliente se desplace a las instalaciones del proveedor.

8.2.16. Tener respeto profesional por las opiniones o actitudes expresadas por los trabajadores de la empresa cliente.

8.2.17. No discriminar a un trabajador o grupo de trabajadores de una empresa cliente.

8.2.18. Ser objetivo a la hora de evaluar el grado de dedicación de los empleados de la empresa cliente.

8.2.19. No interponerse de modo desleal en la trayectoria profesional de los trabajadores de la empresa cliente.

8.2.20. No utilizar recursos con finalidades diferentes a los indicados por la relación contractual entre proveedor y cliente, por ejemplo, para fines particulares o de terceros no relacionados con el cliente.

8.3. Profesional, trabajador o gestor, ante su empresa

8.3.1. Velar por el cumplimiento y conocimiento de les leyes relacionadas con la informática por parte de la empresa.

8.3.2. En el supuesto que la empresa no esté cumpliendo el código deontológico, denunciarlo.

8.3.3. Denunciar ante la empresa las actividades y prácticas consideradas no éticas tanto por lo que respecta a la ejecución de proyectos como en las relaciones interpersonales.

8.3.4. Escuchar y considerar las opiniones de la empresa.

8.3.5. Estar enterado de las directrices y normas de la empresa y aplicarlas cuando convenga.

8.3.6. A la hora de tomar decisiones organizativas o de proyectos considerar de modo imparcial todas las alternativas y a todas las personas teniendo en cuenta también las necesidades de la empresa.

8.3.7. Hacer prevalecer los intereses legítimos del equipo o de la empresa ante los intereses personales.

8.3.8. No utilizar recursos con finalidades diferentes a los indicados legítimamente por la empresa. Por ejemplo, no usar los recursos para fines particulares o de terceros no relacionados con la empresa.

8.3.9. Hacer buen uso de los recursos proporcionados por la empresa para el cumplimiento de las tareas. Denunciar ante la empresa las carencias de recursos de los compañeros y sugerir mejoras en el uso de los mismos.

8.3.10. Hacer un buen uso del nombre de la empresa.

8.3.11. No difundir información sensible o confidencial, relativa a la propia empresa o a los compañeros.

8.3.12. Respetar los derechos de autor. No utilizar trabajo o productos de terceros sin hacer el debido reconocimiento y obtener, en su caso, la correspondiente autorización.

8.3.13. Participar en cursos de formación ofrecidos por la empresa que puedan ser necesarios para el buen desarrollo de la actividad profesional.

8.3.14. Solicitar a la empresa la necesidad de realizar cursos de formación, en el caso de que no se hayan previsto, con arreglo a los proyectos que se desarrollan.

8.3.15. Seguir las directrices y metodologías indicadas por la empresa. En caso de que no existan, promover su creación y uso.

8.3.16. Tomarse el tiempo justo y necesario en el cumplimiento y valoración de las tareas asignadas siempre teniendo presente criterios de calidad.

8.3.17. Documentar el trabajo hecho de la forma que indique la empresa, utilizando un lenguaje de fácil comprensión.

8.3.18. Responsabilizarse de la calidad del trabajo hecho o gestionado, alertando de las problemáticas presentes o futuras que se puedan producir.

8.3.19. Dar siempre la opinión cuando se considere que se pueda ayudar a la empresa a mejorar su funcionamiento.

8.3.20. Si se dispone o se es conocedor de información lícita y útil para la empresa, difundirla libremente.

8.4. Profesional ante su compañero de profesión

8.4.1. No influir en decisiones o actuaciones de compañeros para que le beneficien, anteponiéndose al bien común.

8.4.2. Promover y facilitar el buen uso de la informática entre los compañeros, velando especialmente por el buen aprovechamiento de recursos.

8.4.3. Evitar el uso de códigos personales o lenguaje que pueda resultar confuso o incomprensible para los compañeros en todas aquellas tareas que sean o puedan ser comunes.

8.4.4. No difundir a terceros informaciones falsas o de ámbito no estrictamente profesional de un compañero que puedan perjudicar su progreso dentro de su empresa, sea la misma u otra.

8.4.5. Mantener siempre la confidencialidad de las informaciones personales sobre nuestros compañeros de las que disponemos por razón del trabajo y/o la convivencia en el trabajo, especialmente aquellas cuya divulgación pueda comportar un daño a la imagen de la persona o pueda suponer un freno para su progreso laboral.

8.4.6. No mostrar preferencias o manías con los compañeros, independientemente de las afinidades personales que pueda haber.

8.4.7. Actuar con solidaridad profesional. Valorar siempre la labor de los compañeros, con el debido reconocimiento por la tarea de cada cual.

8.4.8. Ser objetivo a la hora de opinar sobre un compañero, ya sea sobre la calidad de su trabajo o la adecuación para una determinada tarea o responsabilidad, no dejándose influir por condicionantes económicos, laborales o personales.

8.4.9. No pretender asumir una tarea si somos conscientes de que no tenemos los conocimientos o habilidades necesarios y más aún si sabemos que hay un compañero que la puede hacer correctamente.

8.4.10. Trabajar con la máxima profesionalidad y diligencia, evitando especialmente que la actividad propia pueda afectar negativamente al trabajo o a la buena imagen de los compañeros.

8.4.11. Cuando un profesional se sienta perjudicado por otro, le comunicará de forma no ofensiva (asertiva), tratando de evitar cualquier conflicto o confrontación y con el único objeto de conseguir el mejor clima de confianza y colaboración.

8.4.12. Facilitar a los compañeros de trabajo el acceso a las informaciones, metodologías y recursos necesarios para el ejercicio de su tarea.

8.4.13. Velar para que las condiciones de trabajo de los compañeros, así como las propias, sean las óptimas, evitando riesgos e incomodidades.

8.4.14. Promover que los compañeros conozcan y cumplan este código.

8.5. Responsabilidad personal del profesional

8.5.1. No hacer prevalecer intereses personales ante el bien común.

8.5.2. Formarse adecuadamente a lo largo de toda su carrera profesional.

8.5.3. Conocer, respetar y cumplir todas las leyes y normas aplicables.

8.5.4. Utilizar y promover el uso de software legal sea libre o de propiedad.

8.5.5. Buscar, utilizar y promover el uso de herramientas informáticas apropiadas.

8.5.6. Utilizar y promover el uso responsable y adecuado de las herramientas y recursos a los que se tenga alcance.

8.5.7. Trabajar con la máxima profesionalidad y diligencia.

8.5.8. Ser responsable de su trabajo y afrontar con responsabilidad las consecuencias de sus errores.

8.5.9. Seguir una metodología en el desempeño de su cometido, así como utilizar los estándares aplicables en cada momento.

8.5.10. Promover la aplicación de criterios de accesibilidad para las personas con discapacidades tanto en los proyectos desarrollados como en las instalaciones del puesto de trabajo.

8.5.11. Aplicar y velar por el cumplimiento de las recomendaciones sobre ergonomía aplicables al puesto de trabajo.

8.6. Profesional o empresa ante el producto o servicio que presta

8.6.1. Seguir una metodología adecuada para cualquier proyecto en el que se trabaje o se tenga planeado trabajar.

8.6.2. Valorar el coste del producto o servicio en relación al tiempo y recursos necesarios para su construcción y/o prestación. No ofrecer servicios o productos por debajo de este coste.

8.6.3. Ser objetivo a la hora de elegir un producto o servicio. Se seguirán indicadores conocidos o estándares para ayudar a hacer la elección.

8.6.4. Expresar la opinión profesional en relación al producto o servicio sobre todo cuando se detecte que el producto o servicio no sigue el camino previsto. Expresar la conformidad o no conformidad en el cumplimiento de los objetivos esperados del producto o servicio.

8.6.5. Procurar que el servicio o producto sea de la mejor calidad posible. Ser honesto en el ejercicio del trabajo y dedicar el tiempo justo y necesario a cada tarea.

8.6.6. Hacer un uso medido de los recursos para la construcción del producto o prestación del servicio. Procurar optimizar el servicio usando los mínimos recursos necesarios.

8.6.7. Informar de modo periódico y cuando así se solicite del estado del servicio o producto, describiendo tanto los factores favorables como desfavorables.

8.6.8. En el diseño del producto o servicio primará diseñar la solución que pueda dar más ventajas en el presente y futuro.

8.6.9. Los productos se tienen que diseñar pensando en criterios de ergonomía y accesibilidad, facilitando al máximo la utilización para cualquier tipo de usuario.

8.6.10. Proporcionar el material necesario para entender fácilmente el funcionamiento del producto o servicio. Utilizar un lenguaje de fácil comprensión, evitando el uso de tecnicismos, siglas o palabras extranjeras cuando exista una palabra traducida inteligible.

8.6.11. Promover o proponer la intervención de un tribunal de arbitraje competente si hubiese conflictos en la prestación del servicio o en el producto.

8.6.12. No utilizar ilegalmente herramientas de terceras partes en la construcción del producto o servicio.

8.6.13. Ser conocedor de las consecuencias ante un mal funcionamiento del producto o servicio y reconocer las equivocaciones en el supuesto de que se produzcan.

8.6.14. Primar la búsqueda de la solución antes que la búsqueda de los responsables del problema. Analizar el motivo del problema, resolverlo y tomar las medidas necesarias para que no se vuelva a producir.

8.6.15. No atribuirse trabajo ajeno.

8.6.16. No discriminar en la prestación del servicio a clientes o usuarios por razones diferentes de las estrictamente profesionales.

8.6.17. No hacer uso de ninguna información confidencial que pueda perjudicar a personas físicas o jurídicas, ni explotar esta información para sacar un provecho personal.

8.7. Empresa o gestor ante el profesional empleado

8.7.1. Ofrecer unos planes de formación continuada, promoviendo el desarrollo de una carrera profesional de sus empleados. Estos planes de formación han de incluir todos los temas

básicos para el desarrollo de su trabajo (leyes aplicables, seguridad, prevención de riesgos laborales, metodologías, tecnologías y conocimientos funcionales).

8.7.2. No hacer falsas promesas a los empleados ni generar falsas expectativas, y mantener la palabra de los diversos acuerdos verbales y por escrito a los que se puedan llegar.

8.7.3. Mantener una postura proactiva de cara a recolocar a sus trabajadores una vez acabado un proyecto tanto en buscarles nuevos proyectos/clientes como en darles con la antelación suficiente el material y formación que puedan necesitar.

8.7.4. Pagar un sueldo justo en función de las responsabilidades, capacidad y rendimiento de sus trabajadores.

8.7.5. Mantener la confidencialidad de los datos de sus empleados y ex-empleados, ofreciendo en todo caso sólo información veraz y lo más objetiva posible con respecto a las tareas desarrolladas por el empleado.

8.7.6. Valorar la labor de los empleados de forma imparcial, teniendo en cuenta las opiniones de los responsables y clientes que hayan trabajado con ellos, y tenerla en cuenta para la carrera profesional de sus empleados.

8.7.7. Escuchar y valorar de forma imparcial las opiniones de los empleados de la empresa.

8.7.8. Tratar a todos los empleados por igual sin discriminación de ningún tipo (sexo, edad, religión, nacionalidad, raza...).

8.7.9. No discriminar positiva ni negativamente a los empleados subcontratados con respecto a los empleados internos (formación, materiales, herramientas, espacio de trabajo, información accesible, horarios...).

8.7.10. No promover la competitividad desleal entre los empleados.

8.7.11. La asignación de personal a tareas y proyectos se adecuará en la medida de lo posible a la formación y aptitudes de los empleados.

8.7.12. A la hora de hacer las estimaciones de las tareas asignadas a los trabajadores, hacerlo de la forma más objetiva y metodológica posible, teniendo en cuenta las opiniones y valoraciones de los implicados y de los expertos en el tema.

8.7.13. Pedir responsabilidades a los empleados en función de las capacidades, funciones y tareas que tengan asignadas.

8.7.14. No hacer controles abusivos a los empleados más allá de los necesarios para valorar la calidad y resultados de su trabajo.

8.7.15. No hacer trabajar al trabajador más horas de las estipuladas en el contrato. En caso de necesidad, las horas extra serán compensadas tal y como determine el convenio aplicable.

8.7.16. No castigar ni coaccionar a los empleados por ningún motivo. Por ejemplo, para forzar a incumplir el código, la ley, un contrato legal, poner en peligro la seguridad de algún empleado o afectar negativamente la calidad del trabajo de otro compañero.

8.7.17. A la hora de dirigirse y relacionarse con los empleados ha de utilizar un lenguaje comprensible que facilite la comunicación y evite los malentendidos.

8.7.18. Dar al empleado toda la información, material, herramientas, puesto de trabajo y formación necesarias para el buen desarrollo de su trabajo.

8.7.19. Ofrecer espacio y herramientas de trabajo lo más ergonómicas, cómodas, seguras y adecuadas al trabajo que pueda conseguir, así como hacer las adaptaciones necesarias para adecuarlas a los trabajadores con algún tipo de discapacidad.

8.7.20. Fomentar el buen uso de recursos en el desarrollo del trabajo de sus empleados.

8.7.21. Ofrecer a los trabajadores en todo momento *hardware*, *software* y licencias legales, sean libres o de pago, para el desarrollo de su trabajo.

8.7.22. No apropiarse indebidamente de trabajos hechos por sus empleados con horas y recursos ajenos a la empresa.

8.8. Empresa ante las otras empresas del sector

8.8.1. Actuar con honradez y con espíritu colaborativo y profesional, en acciones de colaboración de proyectos comunes para un tercero.

8.8.2. Mantener la confidencialidad de los datos que una empresa pueda recibir traspasadas de otra, en virtud de una colaboración.

8.8.3. Utilizar un lenguaje comprensible y transparente en relaciones de colaboración, ya sean contratos, documentación o explicaciones de palabra, respecto a servicios o productos realizados conjuntamente.

8.8.4. Promover buenas prácticas de trabajo respecto a la competencia, siendo imparcial a la hora de valorar las acciones de otros.

8.8.5. En una colaboración con otra empresa, sea bajo el régimen de subcontratación o sea en proyecto compartido de igual a igual, mantenerse íntegro en la realización correcta del producto o servicio, independientemente de posibles circunstancias adversas.

8.8.6. No descalificar a los competidores, por ejemplo en la fase de venta de un producto o servicio.

8.8.7. Dar siempre una opinión profesional correcta y respetuosa de la competencia.

8.8.8. En una colaboración entre empresas, efectiva o posible, no utilizar la coacción como elemento de negociación, ni como posible castigo respecto a las personas que participen o puedan participar en la citada colaboración.

8.9. Profesional o empresa ante un proveedor

8.9.1. No utilizar prácticas abusivas en la negociación de ofertas con proveedores, tanto en cantidad como en contenido de cláusulas claramente perjudiciales para los proveedores.

8.9.2. Establecer contratos de compra-venta o de prestación de servicios claros y completos, que faciliten el entendimiento entre las partes.

8.9.3. Mantener la confidencialidad respecto a las ofertas que un proveedor pueda hacer, sin hacer uso de las mismas para presionar o informar a otros proveedores.

8.9.4. No promover la competitividad desleal entre los proveedores.

8.9.5. Pactar precios justos por productos o servicios que ofrezca la empresa proveedora.

8.9.6. Actuar objetivamente a la hora de valorar y contratar ofertas de empresas proveedoras.

8.9.7. Evaluar las ofertas valorando la calidad, la metodología y el currículum del equipo ofrecido y de la empresa.

8.9.8. No coaccionar a un proveedor valiéndose de compras de productos o servicios futuros.

8.9.9. No infringir castigo a una empresa proveedora mientras cumpla las obligaciones establecidas en el contrato.

8.9.10. Facilitar un entorno de trabajo a los empleados de la empresa proveedora equiparable al de los empleados de la propia empresa.

8.9.11. No forzar a hacer horas extra no retribuidas, ni por encima de los límites establecidos por el convenio, a los trabajadores de una empresa proveedora.

8.9.12. Facilitar la formación continua de los trabajadores de la empresa proveedora.

8.9.13. No discriminar a un trabajador sólo por el hecho de pertenecer a una empresa proveedora.

8.9.14. Poner a disposición de la empresa proveedora todo el material, *hardware*, *software* y medios necesarios para que la empresa proveedora pueda cumplir sus obligaciones contractuales.

8.9.15. No interponerse de modo desleal en la trayectoria profesional de la empresa proveedora, o de sus trabajadores.

8.9.16. Escuchar al proveedor y valorar objetivamente los consejos que éste pueda ofrecerle.

8.9.17. Tener respeto profesional por las opiniones o actitudes expresadas por los trabajadores de la empresa proveedora, respeto al producto o servicio que ofrecen.

8.9.18. Respetar los derechos de autor respecto un producto o servicio que ha ofrecido o realizado un proveedor.

8.9.19. Ofrecer al proveedor información sobre qué ha gustado y qué no ha gustado del producto o servicio ofrecido o realizado, incluyendo los motivos de rechazo en el caso de no aceptarse una oferta.

8.9.20. Hacer un tratamiento metodológico correcto de las relaciones con proveedores, desde la recepción de la oferta hasta la valoración final del producto o servicio recibido.

8.9.21. Mantener la confidencialidad y la privacidad de los datos que se tengan de la empresa proveedora.

Conclusiones y líneas futuras

A continuación se exponen las conclusiones obtenidas tras la realización del proyecto, así como se indican posibles ampliaciones o mejoras que pueden hacerse en futuras versiones.

Conclusiones

En este último capítulo vamos a repasar los principales objetivos que propusimos inicialmente en el anteproyecto de este trabajo y analizaremos si hemos logrado satisfacerlos.

- **Introducción a la ética informática**. El primer objetivo que nos marcamos fue dar una breve introducción a la ética, relacionándola con la informática, y prestando especial atención a los códigos deontológicos. De todo esto hablamos en el Capítulo 1 de la presente memoria.

- **Recopilación y traducción de códigos éticos informáticos**. Nuestra idea inicial era recopilar un número razonable de códigos éticos pertenecientes a asociaciones informáticas, y traducir algunos de ellos al español. De esta forma, buscábamos participar en la difusión de los códigos deontológicos a la comunidad hispanohablante.

  Durante la realización del proyecto, esta tarea terminó dividiéndose en tres tareas bien diferenciadas:

  - *Recopilación de códigos éticos*. Para esta tarea fue necesaria una exhaustiva investigación, fundamentalmente en Internet, aunque también se enviaron *emails* a distintas organizaciones pidiéndoles sus respectivos códigos éticos. Finalmente, logramos conseguir los códigos éticos de veintiocho organizaciones (Tabla 1.2).
  - *Traducción de los códigos éticos*. De todos los códigos conseguidos, diecinueve estaban escritos en inglés. Decidimos traducir al español la gran mayoría de estos códigos; para ser exactos, dieciséis. De esta forma, con un total de veinticinco códigos en español, nuestra recopilación se convierte en la mayor de las recopilaciones de códigos éticos informáticos en español que hemos encontrado tanto en libros y revistas técnicas como en Internet.

    Es importante destacar que la traducción de todos estos códigos ha sido una traducción humana, en la que se ha intentando mantener el espíritu y la letra del código,

evitando en lo posible traducciones libres. Aunque hemos utilizado herramientas de traducción automática, éstas sólo nos han servido como ayuda, puesto que, a día de hoy, la traducción automática no es lo suficientemente fiable como para poder sustituir la figura del traductor humano.

- *Clasificación temática de los códigos éticos*. Durante esta tarea, puesto que, en un principio, disponíamos de una lista de temas incompleta y redundante, tuvimos que clasificar dos veces cada uno de los códigos éticos. La primera clasificación nos sirvió para confeccionar una lista más completa que la que teníamos. Esta última lista es la que utilizamos en la segunda clasificación, que ya sí fue la clasificación definitiva.

- **Crear una base de datos y un sitio web para acceder a los códigos**. En la web, actualmente los principales sitios con códigos éticos informáticos sólo ofrecen algunos códigos, todos ellos escritos en inglés. De esta forma se restringe su posible difusión a causa de la lengua. Por esto, otro de los objetivos del proyecto era ofrecer una aplicación web en español en la que poder consultar códigos deontológicos de organizaciones nacionales e internacionales. Nuestra aplicación debería aprovecharse de las ventajas ofrecidas por una base de datos para así implementar distintos tipos de búsqueda: por palabras, por organización, por tema, etc.

  Al final, decidimos implementar la interfaz de la aplicación también en inglés, además de en español, para así abarcar un mayor número de posibles visitantes. En vez de crear primero la aplicación en un idioma determinado y luego traducir ésta al otro idioma, obteniendo así dos aplicaciones iguales pero con la interfaz escrita en distinto idioma, decidimos diseñar una aplicación web internacionalizada fácilmente ampliable a cualquier idioma, gracias a que utilizamos UTF8 como juego de caracteres.

  Lo primero que tuvimos que hacer antes de comenzar a construir la aplicación fue estudiar las principales tecnologías web y decidir cuales íbamos a usar para nuestro desarrollo. Todo este proceso de investigación está recogido en el Capítulo 2 de la memoria.

  Decidimos dividir el proceso de desarrollo de la aplicación en tres frases: análisis, diseño e implementación, adoptando así el modelo secuencial de desarrollo. La metodología utilizada ha sido UML, el estándar actual para desarrollos orientados a objetos. Se ha buscado maximizar la reutilización de componentes mediante el uso de bibliotecas de clases, patrones de diseño (principalmente los patrones MVC y DAO) y una arquitectura base (o *framework*), que en nuestro caso ha sido Struts, sobre la que construir nuestra aplicación.

  La metodología UML es una metodología muy general y no nos permite, por sí sola, expresar las relaciones que existen entre los distintos elementos de una aplicación web. Por ello, hemos tenido que emplear los mecanismos de extensión que UML proporciona,

creando así un perfil UML específico para aplicaciones web construidas con Struts.

El diseño de la base de datos fue una tarea problemática. Teníamos que encontrar una estructura común que englobase a todos los códigos éticos que habíamos recopilado, además de los que pudiéramos añadir en un futuro. Asimismo, cada uno de estos códigos podía tener varias traducciones. Al final, nos decantamos por el modelo que se muestra en el Apartado 6.2.1.2, que, en pocas palabras, se basa en que un código ético está formado por varias secciones, cada una de las cuales tiene un «nivel» que indica su relación con el resto de secciones.

Por último, decir que la aplicación web Codetic estará disponible de manera gratuita desde la página web del director del proyecto, junto al resto de la documentación del proyecto.

- **Proponer un código ecléctico**. Otro de los objetivos del proyecto era crear un código ético ecléctico, a partir de los códigos éticos recopilados, y como propuesta para conseguir un código ético común. Con él, intentaríamos unificar en un único código las virtudes de los que hemos estudiado y mejorar otros aspectos que consideramos relevantes. Nuestra propuesta ecléctica se encuentra recogida en el Capítulo 8 de la presente memoria.

- **Utilizar *software* de fuente abierta**. Inicialmente no era un objetivo, pero al final terminó siéndolo. Con él, hemos pretendido ofrecer una alternativa de fuente abierta para el desarrollo de aplicaciones web. Esta filosofía también se ha seguido para la redacción de la memoria, creación de los gráficos, etc.

  El hecho de sólo poder usar *software* de fuente abierta supuso que el proceso de búsqueda y elección de herramientas se complicara en determinados casos, como cuando tuvimos que elegir la herramienta de modelado UML, Poseidon for UML, o la herramienta de modelado de base de datos, DBDesigner. Esta complicación se debió a que todavía existen campos en los que no hay disponibles alternativas *open source* de calidad.

Una vez revisados los principales objetivos marcados al inicio del proyecto, podemos decir que todos ellos se han cumplido, e incluso se han visto ampliados en algunos casos.

Nos parece importante destacar que el *software* desarrollado sirve también para los códigos éticos de otros contextos (periodismo, medicina, física, química...). Más aún, sirve para clasificar cualquier tipo de texto según su temática. Lo único que debemos cambiar en cada caso es la lista de temas y sus grupos, adaptándola a la problemática del contexto.

Líneas futuras

Aunque hemos intentado que el proyecto fuera lo más completo posible, el tiempo del que disponemos es limitado y siempre quedan cosas que mejorar. A continuación destacamos algunas posibles ampliaciones futuras:

- Optimizar el rendimiento de las búsquedas. Esto se puede hacer en MySQL mediante los índices FULLTEXT, cuya función es mejorar las búsquedas de texto. El inconveniente de estos índices es que sólo se pueden definir en tablas de tipo MyISAM. Estas tablas no admiten la creación de claves foráneas. Por tanto, se tendría que reprogramar parte de la aplicación, ya que nosotros hemos hecho uso de las claves foráneas en tablas InnoDB, para así aprovechar la eliminación y la actualización en cascada que ofrecen este tipo de claves. Otra opción sería usar el módulo TSearch2 de la base de datos PostgreSQL. Este módulo permite hacer una indexación de texto completo muy eficiente. Como es evidente, esta alternativa implicaría mayores modificaciones y esfuerzos, puesto que supone cambiar la base de datos de nuestro sistema.

- Mejorar las búsquedas en diversos aspectos:

  - Implementando operadores relacionales AND y OR, así como las comillas dobles para indicar búsqueda exacta.
  - Implementando un filtro de palabras a ignorar (*stopword list*). Los ya nombrados FULLTEXT o TSearch2 ofrecen esta posibilidad.
  - Permitiendo ordenar los resultados por relevancia. La relevancia mide la similitud entre las palabras buscadas y el texto en el que se buscan dichas palabras.
  - Permitiendo que el usuario pueda buscar por palabras clave. Una forma de implementarlo podría ser mediante una lista alfabética de palabras clave, donde cada una de ella está enlazada con una búsqueda por el tema o temas que las contiene.
  - Podríamos implementar una especie de «herencia de temas», consistente en que si una sección tiene un tema se supone que éste abarca todas sus secciones inferiores.

- Cachear la información de la base de datos. Puesto que nuestra información cambia con muy poca frecuencia, podríamos cachearla en archivos de texto [46].

- Incluir la posibilidad de realizar una revisión de traducciones incompletas. Básicamente, lo que tendríamos que hacer sería crear un página web que mostrara aquellas traducciones de códigos éticos con al menos una sección sin traducir a un idioma concreto.

- Añadir en la administración la posibilidad de administrar los «Enlaces» (véase el Apartado 7.2), ya que éstos cambian con el tiempo.

- Incluir estadísticas en la aplicación web. Crear una página web que muestre estadísticas de la base de datos, como pueden ser las frecuencias absoluta y relativa de los temas, de los grupos de temas, etc. En el Apartado 1.2.4 se incluye una pequeña estadística de los datos «actuales» de la base de datos.

- Crear un glosario de términos relacionados con la ética informática e incluirlo en la aplicación web.

- Añadir la posibilidad para el usuario de descargar todos o algunos de los códigos en formato PDF, texto...

- Adaptar la aplicación web para que se ajuste a las pautas de accesibilidad WCAG 1.0 (Apartado 2.2.6).

- Mejorar la biblioteca de etiquetas Text2Html. Esta biblioteca permite convertir texto en HTML de una forma muy rudimentaria: separa en párrafos el texto y los encierra entre etiquetas `<p>` y `</p>`. Se utiliza para aplicar formato HTML al texto de los artículos justo antes de su visualización. Para mejorarla, tendríamos que ser capaces de detectar otras estructuras dentro del texto, como secciones, listas (numeradas o no), tablas, etc. Por ejemplo, podríamos tomar los símbolos -,*, etc. como listas no ordenadas (`<ul>...</ul>`). La documentación de AscToHTM [40], un conversor de texto a HTML, contiene bastantes ideas interesantes que podrían ser implementadas.

- Incluir opciones de copias de seguridad. Se trata de dotar a Codetic de facilidades para realizar copias de seguridad de la base de datos.

- Averiguar cómo se podrían indentar de una manera cómoda determinadas subsecciones sin tener que escribir código HTML en el texto de las secciones. Por ejemplo, podríamos tener un campo booleano llamado «subsecsIndentadas» en cada sección que si está a `true` indica que debemos indentar todas sus subsecciones.

- Cuando vayamos al código completo a partir de la página en la que se visualiza un único artículo, podríamos resaltar de alguna manera (recuadrar, sombrear, etc.) dicho artículo en el código. Debería haber una opción para poder desactivar/activar dicho resaltado.

- Mejorar la calidad del código de la aplicación. Para ello, podríamos realizar pruebas unitarias (*unit tests*) con JUnit[1] a las clases de nuestra aplicación. Las pruebas unitarias son una forma bastante cómoda y potente de realizar pruebas sobre el código y detectar fallos, y JUnit es un *framework* Java que permite automatizar dichas pruebas en Java.

- Aunque todos los temas que se tratan en los códigos están reflejados en la lista de temas de una u otra forma, pueden existir temas que puedan considerarse tan relevantes como para tener su tema propio. Aunque es un asunto siempre discutible, este podría ser el caso del tema «Relación entre empresas del sector», porque ese tema puede considerarse incluido en el tema «Colegas de profesión (solidaridad, cooperación, respeto, competencia)».

- Mantenimiento y puesta al día de la información almacenada en la base de datos. Es interesante ir completando la información de la base de datos, añadiendo aquellos datos que no sepamos (año de fundación de algunas asociaciones, años de aprobación de ciertos

[1] Disponible en su sitio web oficial: http://www.junit.org/

códigos, etc.), revisando las traducciones de los códigos éticos y de las descripciones de las asociaciones, y añadiendo nuevos códigos.

- Traducir al inglés los manuales de usuario y administrador, así como la propuesta ecléctica, y añadirlos a la aplicación.
- Traducir la interfaz a otros idiomas.

Apéndice A

Lista de temas para la clasificación de los artículos de los códigos éticos

Los temas están clasificados en los siguientes grupos:

- Apoyos
- Desarrollo Profesional
- Desempeño de la Profesión
- Ética y Legalidad
- Relaciones Humanas
- Sociedad y Medio Ambiente
- Valores Personales y Profesionales
- Otros Temas

A.1. Apoyos

Este grupo está formado cuatro temas en los que se incide sobre la importancia de que el profesional preste su apoyo a distintos contextos (a la asociación, a su empresa, a su nación y a la informática en general).

- *Apoyo y defensa a la asociación*. Los profesionales que pertenezcan a una asociación deberán apoyarla y defenderla: difundiendo la existencia de la asociación, contrarrestando informaciones que puedan dañar la imagen de la misma, comprometiéndose a cumplir el código ético, etc.

  Palabras clave: Apoyo a la asociación, defensa de la asociación, promoción de la asociación...

- *Apoyo y defensa a la empresa*. Los profesionales deberán apoyar, defender y respetar las normas de la empresa en la que trabajan.

  Palabras clave: Empresa pública o privada, patrones, jefes, empresarios...

- *Apoyo y defensa a la nación*. Los profesionales deben defender su nación y honrar a sus conciudadanos.

  Palabras clave: Apoyo y defensa a la nación, apoyo y defensa del país, apoyo y defensa de los conciudadanos...

- *Promoción y defensa de la Informática*. Los profesionales deberá promocionar, dar a conocer y defender los usos dignos y éticos de la informática.

  Palabras clave: Promoción de la Informática, defensa de la Informática, desarrollo tecnológico, avance de las nuevas tecnologías, sociedad de la información, intrusismo profesional...

A.2. Desarrollo profesional

En este grupo se incluyen aquellos temas que tratan sobre la evolución del profesional informático, así como aspectos propios de algunas dedicaciones concretas con especial interés ético (como la docencia en informática).

- *Capacitación y formación profesional*. Recoge aquellos asuntos relativos a la preparación, formación y puesta al día de los profesionales. También recoge aquellos asuntos relacionados con la necesidad de poseer y adquirir una capacitación profesional adecuada.

  Palabras clave: Conocimiento, capacidades, habilidades, aptitud profesional, educación, cursos, seminarios, formación continua, actualización de conocimientos...

- *Desarrollo y carrera profesional*. Recoge los asuntos relativos al desarrollo y los trabajos y puestos desempeñados por los profesionales.

  Palabras clave: Desarrollo profesional, carrera profesional...

- *Docencia*. Recoge aquellos asuntos relativos a los profesionales dedicados a la docencia de la ciencia informática.

  Palabras clave: Docencia, enseñanza, formación, profesor, alumno, universidad, escuela, colegio, academia, asignatura, evaluación del aprendizaje...

A.3. Desempeño de la profesión

Este grupo está formado por cuatro temas en los que se recogen aquellas cualidades que deberían poseer los profesionales en el cumplimiento de su trabajo.

- *Buen uso de recursos*. Los profesionales deben reconocer y apoyar los usos adecuados y autorizados de los recursos informáticos y de comunicaciones.

  Palabras clave: Uso de recursos, utilización de recursos, reducción en el consumo...

- *Dirección y liderazgo*. Recoge los asuntos relativos a las tareas de dirección y liderazgo por parte de aquellos profesionales que tienen subordinados a su cargo.

  Palabras clave: Dirección, liderazgo, organización, liderazgo organizativo, administración, gestión, jefe...

- *Lenguaje comprensible*. Los profesionales deben emplear un lenguaje comprensible para sus interlocutores.

  Palabras clave: Lenguaje comprensible, evitar la pedantería, uso de tecnicismos, siglas o palabras extranjeras...

- *Uso del correo electrónico*. Recoge aquellos asuntos relacionados con la utilización del correo electrónico por parte de los profesionales.

  Palabras clave: Correo electrónico, email, spam, envío masivo de emails, correo basura...

A.4. Etica y legalidad

Este grupo engloba a aquellos temas que tratan sobre el comportamiento ético y legal de los profesionales.

- *Comportamiento no ético*. Recoge aquellos asuntos relacionados con un comportamiento no ético general por parte de los profesionales.

  Palabras clave: Comportamiento no ético, infracciones del código ético...

- *Cumplimiento del código (alcance, sanciones, denuncias)*. Recoge los asuntos relativos al cumplimiento del código ético: alcance (a quienes afecta), sanciones, denuncias, etc.

  Palabras clave: Alcance del código, sanciones, violaciones, infracciones, denuncias, conformidad con el código...

- *Dilema ético*. Trata sobre qué hacer cuando los profesionales se encuentran ante un dilema ético.

  Palabras clave: Dilema ético, conflicto ético, problemas éticos, decisiones éticas...

- *Justificación del código ético*. Se justifica la elaboración y asunción de un código ético.

  Palabras clave: Introducción al código, prólogo del código, justificación del código...

- *Leyes, normas, estándares*. Los profesionales deberán conocer, promover y acatar aquellas leyes, normas, reglas, disposiciones, etc. establecidas por el gobierno, su asociación, su empresa, u otros organismos competentes.

  Palabras clave: Leyes, normas, estándares, prácticas abusivas, disposiciones, regulaciones, normativas, legislación, convenios, propiedad intelectual, políticas, reglamentos...

- *Promoción del comportamiento ético*. Aunque una buena forma de enseñar es dar ejemplo, este tema engloba aquellas actividades relacionadas con la educación en los valores éticos.

  Palabras clave: Educación en valores, difundir el código ético, ejemplo de comportamiento ético...

- *Propiedad intelectual*. Recoge los asuntos relativos a la propiedad intelectual, la cuál abarca cuatro tipos distintos y separados de propiedad intangible, a saber: patentes, marcas registradas, derechos de autor y secretos comerciales.

  Palabras clave: Patentes, marcas registradas, secretos comerciales, propiedad industrial, derechos de autor, licencias, copias ilegales, piratería...

A.5. Relaciones humanas

Este grupo trata sobre las relaciones que mantiene el profesional con los diversos colectivos (clientes, colegas de profesión, proveedores y usuarios) con los que debe interactuar durante el desempeño de su trabajo.

- *Clientes de nuestra empresa*. Trata sobre aquellos asuntos en los que se hace referencia a los clientes de nuestra empresa.

  Palabras clave: Cliente, estudiante, dar servicio, dar prestaciones, prestar ayuda...

- *Colegas de profesión (solidaridad, cooperación, respeto, competencia)*. Los profesionales deberán ser solidarios, cooperar y respetar a los profesionales con los que colaboren. Además, deberán reconocer el trabajo de sus colegas.

  Palabras clave: Solidaridad, cooperación, respeto, reconocimiento, competencia, conflictos entre profesionales, relaciones entre empresas...

- *Proveedores*. Este tema engloba la relación con las empresas proveedoras de bienes y servicios.

  Palabras clave: Proveedores, suministradores, la empresa como cliente...

- *Usuarios*. Trata sobre todos aquellos asuntos en los que se hace referencia a los usuarios de los sistemas informáticos desarrollados por los profesionales.

  Palabras clave: Usuarios trabajadores, usuarios clientes, usuarios de un sistema informático, consumidores...

A.6. Sociedad y medio ambiente

Los temas de este grupo priman el servicio al bien común, público y social por parte del profesional informático.

- *Acceso a las Tecnologías de la Información*. Los profesionales deben promover el acceso universal a las TI por parte de todos los miembros de la sociedad.

  Palabras clave: Acceso a las TI, TPDH (Tecnología para el Desarrollo Humano), acceso a las nuevas tecnologías, acceso a Internet, acceso a la información, desarrollo humano, brecha digital, accesibilidad...

- *Derechos humanos (discriminación, justicia, salud, seguridad)*. Los profesionales deben respetar y hacer respetar todos los derechos humanos.

  Palabras clave: Igualdad, tolerancia, respeto a los demás, justicia equitativa, salarios equitativos, dignidad, valores humanos...

- *Interés público (bien social)*. Los profesionales perseguirán el bien de la sociedad en general. Se preocuparán por el interés del público, así como se encargarán de aumentar la apreciación y el conocimiento públicos de la informática.

  Palabras clave: Interés público, bien social, sociedad, calidad de vida, derechos humanos, diversidad cultural, bienestar humano, responsabilidad social...

- *Medio Ambiente*. Los profesionales deben respetar y hacer respetar el medio ambiente.

  Palabras clave: Medio ambiente, entorno, desarrollo sostenible, consumo responsable, reciclaje, reducir el consumo, reutilizar bienes materiales, respeto a la Naturaleza, ecología...

A.7. Valores personales y profesionales

Los profesionales deben mantener una serie de valores, tanto a nivel personal como profesional, durante el ejercicio de su profesión. Los temas que forman este grupo son los siguientes:

- *Confidencialidad y privacidad*. Los profesionales deberán preocuparse por mantener la confidencialidad de la información privada.

Palabras clave: Confidencialidad, privacidad, intimidad, accesos no autorizados, seguridad en el acceso a los datos...

- *Honradez e integridad.*. Los profesionales deberán actuar personal y profesionalmente de manera honrada e íntegra.

  Palabras clave: Honradez, integridad, corrupción, exactitud, veracidad, honestidad, dignidad, confianza, lealtad, responsabilidad personal, comunicar información relevante...

- *Imparcialidad (desinterés personal)*. Los profesionales deberán actuar de manera objetiva e imparcial, sin buscar el beneficio personal.

  Palabras clave: Imparcialidad, objetividad, desinterés personal, beneficio personal...

- *Libertad profesional*. Los profesionales deben tener libertad a la hora de elegir cómo realizar su trabajo, con las limitaciones lógicas (presupuesto, material disponible...): elección de metodologías, contratación de personal, etc.

  Palabras clave: Libertad de actuación, independencia profesional, toma de decisiones sin coacciones...

- *Profesionalidad y diligencia (responsabilidad, buen servicio, calidad)*. Los profesionales deberán realizar su trabajo con profesionalidad y diligencia, asumiendo la responsabilidad de su trabajo y ofreciendo productos y servicios de calidad. Deberán cumplir las obligaciones de su trabajo profesional de acuerdo con los objetivos de plazo y presupuesto.

  Palabras clave: Responsabilidad profesional, buen servicio, trabajo profesional, excelencia, calidad, evaluación y diseño de los requisitos, validación, profesionalismo, producir software bien documentado, cumplimiento de plazos, ajustarse al presupuesto...

- *Transparencia de la información*. Los profesionales deberán comunicar a los usuarios, colegas, empresa, clientes, asociación, etc. sobre aquellos asuntos que sean de interés para los mismos.

  Palabras clave: Informar y comunicar información de interés, ocultamiento de información...

A.8. Otros temas

En este grupo se incluyen todos aquellos temas que no encajan, de manera clara, en ninguno de los otros grupos.

- *Calidad del lugar del trabajo*. Se deberá tener un lugar de trabajo que cumpla ciertos requisitos de calidad, tanto en la ergonomía de los materiales (sillas, equipos...) como en el ambiente (ventilación, productos tóxicos, extintores...).

Palabras clave: Calidad, lugar de trabajo, ergonomía persona-ordenador, luminosidad, prevención de riesgos laborales, ventilación, salud, seguridad laboral...

- *Condiciones económicas justas (sueldo, tarifas)*. Se deben establecer unas condiciones económicas justas y adecuadas a cada situación, tanto en el salario a los empleados como en las tarifas para los clientes.

  Palabras clave: Sueldo, remuneración, retribución, tarifa, precio...

- *Conflicto de intereses*. Recoge aquellos asuntos sobre los conflictos de intereses que se le pueden presentar a los profesionales.

  Palabras clave: Conflicto de interés...

- *Código fuente abierto (Open Source)*. Los profesionales, cuando les sea posible, deberán contribuir a la difusión y desarrollo de plataformas y sistemas de código abierto.

  Palabras clave: Código abierto, código fuente abierta, open source, software libre...

- *Opinión profesional*. Recoge los asuntos relativos a la opinión profesional ofrecida por los profesionales.

  Palabras clave: Opinión profesional, crítica profesional, evaluación profesional, consultoría...

Apéndice B

Código de ética y de conducta profesional de la ACM

Traducción realizada por José Javier Dolado Cosín.
Disponible en http://www.sc.ehu.es/jiwdocoj/codeacm.htm.

Preámbulo

Se espera un compromiso con una conducta ética profesional de todo miembro de la ACM, ya sea de número, asociado o estudiante. Este Código, consistente en 24 preceptos expresados como declaraciones de responsabilidad personal, identifica los elementos de tal compromiso.

Este código trata muchos de los aspectos, pero no todos, que los profesionales probablemente afrontarán. La Sección 1 perfila las consideraciones éticas fundamentales, mientras que la Sección 2 trata reflexiones adicionales, más específicas sobre la conducta profesional. Los preceptos de la Sección 3 incumben más específicamente a personas que tengan una función de liderazgo, bien sea en su lugar de trabajo o en calidad de voluntarias, como puede suceder en organizaciones tales como la ACM. Los principios que involucran conformidad con este Código se muestran en la Sección 4.

El código se complementa con una serie de Guías, que proporcionan explicaciones para ayudar a los miembros a tratar los diferentes temas contenidos en el Código. Es esperable que las Guías se modifiquen con más frecuencia que el Código.

El Código y sus Guías suplementarias tienen la finalidad de servir como base para la toma de decisiones éticas en el comportamiento profesional. Secundariamente, pueden servir como como base para juzgar las circunstancias de una queja formal relacionada con la vulneración de los estándares de ética profesional.

Debe hacerse notar que aunque no se menciona a la informática en la sección de preceptos morales, el Código se interesa en cómo estos mandatos fundamentales se aplican al comportamiento individual como profesional de la informática. Estos mandatos se expresan de una manera general para así enfatizar que los principios éticos que se aplican a la ética informática

se derivan de principios éticos más generales.

Se entiende que algunas palabras y frases en un código de ética son susceptibles de diferentes interpretaciones, y que cualquier principio ético puede entrar en conflicto con otros principios éticos en situaciones concretas. Las cuestiones relacionadas con conflictos éticos se pueden resolver mejor a través de una meditada consideración de los principios fundamentales, más que confiando en reglamentos detallados.

1. Preceptos morales generales

Como miembro de la ACM...

1.1. Contribuiré al bienestar de la sociedad y de la humanidad

Este principio, referente a la calidad de vida de todas las personas, declara una obligación para proteger los derechos humanos fundamentales y respetar la diversidad de todas las culturas. Un objetivo esencial de los profesionales de la informática es minimizar las consecuencias negativas de los sistemas informáticos, incluyendo las amenazas a la salud y a la seguridad. Cuando se diseñen o instalen sistemas, los profesionales de la informática deben intentar garantizar que los productos de sus esfuerzos se utilizarán de modos socialmente responsables, recogerán las necesidades sociales y evitarán efectos perjudiciales a la salud y al bienestar.

Además de tener un entorno socialmente seguro, el bienestar humano incluye tener un medio ambiente seguro. Por lo tanto, los profesionales informáticos que diseñan y desarrollan sistemas deben estar alerta, y hacer conscientes al resto, sobre cualquier daño potencial al medio ambiente local o global.

1.2. Evitaré daño a otros

«Daño» significa perjuicio o consecuencias negativas, tales como pérdida indeseable de información, pérdida de propiedad, daño a la propiedad, o efectos medioambientales no deseados. Este principio prohíbe el uso de la tecnología informática en maneras que resulten dañinas a usuarios, público en general, empleados y empresarios. Las acciones perjudiciales incluyen la destrucción intencional o modificación de ficheros y programas que conlleven una grave pérdida de recursos, o un gasto innecesario de recursos humanos tales como el tiempo y esfuerzo requerido para limpiar los sistemas de virus informáticos.

Las acciones bien intencionadas, incluyendo aquellas que cumplen las obligaciones asignadas, pueden ocasionar daños de manera inesperada. En tal circunstancia la persona o personas responsables están obligadas a reparar o mitigar las consecuencias negativas tanto como sea posible. Una manera de evitar daños no intencionados es considerar cuidadosamente las consecuencias potenciales en quienes vayan a verse afectados por las decisiones tomadas durante el diseño e instalación.

Para minimizar la posibilidad de dañar indirectamente a otros, los profesionales de la informática deben minimizar los funcionamientos incorrectos mediante la aplicación de los estándares aceptados para el diseño y prueba del sistema. Además, a menudo es necesario evaluar las consecuencias sociales de los sistemas para estimar la posibilidad de perjudicar seriamente a otros. Si las características del sistema aparecen mal representadas ante los usuarios, profesionales del equipo o supervisores, el individuo profesional es el responsable de cualquier perjuicio resultante.

En el entorno de trabajo el/la informático tiene la obligación añadida de informar sobre cualquier signo de peligro del sistema que pueda resultar en un grave daño personal o social. Si sus superiores no actúan para terminar o mitigar tales peligros, puede ser necesario alertar para ayudar a corregir el problema o reducir el riesgo. Sin embargo, informes arbitrarios o mal enfocados sobre irregularidades pueden ser, por sí mismos, perjudiciales. Antes de informar sobre las irregularidades, deben evaluarse cuidadosamente todos los aspectos relevantes del incidente. En particular, la valoración del riesgo y de la responsabilidad debe ser creíble. Se sugiere solicitar consejo a otros profesionales (ver principio 2.5 referente a evaluaciones minuciosas).

1.3. Seré honesto/a y confiable

La honestidad es un componente esencial de la confianza. Sin confianza una organización no puede funcionar con efectividad. El informático honesto no hará falsas o engañosas declaraciones acerca de un sistema o diseño de sistema, sino que, por el contrario, proporcionará una completa exposición de todas las limitaciones y problemas pertinentes del sistema.

Un/a profesional informático tiene la obligación de ser honesto/a acerca de sus propias cualificaciones, y acerca de cualquier otra circunstancia que pueda generar conflictos de interés.

La afiliación a organizaciones tales como la ACM puede algunas veces situar a las personas en situaciones donde sus acciones o afirmaciones pueden ser interpretadas como si fueran las de un grupo más amplio de profesionales. Un miembro de la ACM deberá tener cuidado para no desfigurar a la ACM, las políticas y posiciones de la ACM o de cualquiera de sus secciones.

1.4. Seré justo/a y actuaré para no discriminar

Los valores de igualdad, tolerancia, respeto a los demás y los principios justicia equitativa gobiernan este mandato. La discriminación basada en la raza, sexo, religión, edad, discapacidad, nacionalidad, u otros factores es una violación expresa de la política de la ACM y no se tolerará.

Las desigualdades entre diferentes grupos de personas pueden ser resultado del buen o mal uso de la información y de la tecnología. En una sociedad justa, todos los individuos tienen igual derecho a participar o beneficiarse del uso de los recursos informáticos sin distinción de raza, sexo, religión, edad, discapacidad, nacionalidad u otros factores similares. Sin embargo, estos ideales no justifican el uso no autorizado de recursos informáticos ni proporcionan una base adecuada para transgredir cualquier otro mandato ético de este código.

1.5. Respetaré los derechos de propiedad, incluyendo las patentes y derechos de autor

La vulneración de los derechos de autor, patentes, secretos comerciales y compromisos de las licencias está prohibido por la ley casi siempre. Incluso si el software no está así protegido, tales inobservancias son contrarias al comportamiento profesional. Las copias del software sólo deben realizarse con la autorización adecuada. No se debe condonar el duplicado no autorizado de materiales.

1.6. Reconoceré adecuadamente la propiedad intelectual

Los/las profesionales informáticos están obligados a proteger la integridad de la propiedad intelectual. Específicamente, una persona no debe atribuirse el mérito del trabajo o ideas de otros, incluso en los casos en los que no han sido explícitamente protegidos, por ejemplo con derechos de autor o patente.

1.7. Respetaré la intimidad de otros

La tecnología informática y de comunicaciones permite la recogida e intercambio de información personal a escala no conocida en la historia de la civilización. Existe, por tanto, un creciente potencial para la violación de la intimidad de los individuos y de los grupos. Es responsabilidad de los profesionales el mantener la confidencialidad e intimidad de los datos pertenecientes a las personas. Esto incluye tomar precauciones para garantizar la corrección de los datos, así como protegerlos de accesos no autorizados, o accidentales, a las personas no autorizadas. Además, se deben establecer procedimientos para permitir a las personas revisar sus registros y corregir las incorrecciones.

Este mandato implica que sólo debe recogerse la información necesaria para un sistema, que los periodos de mantenimiento y de disponibilidad de la información deben estar claramente definidos y obligarse a ellos estrictamente, y que la información personal recogida para un propósito específico no debe utilizarse para otros propósitos sin consentimiento de las personas. Estos principios se aplican a las comunicaciones electrónicas, incluyendo el correo electrónico, y se prohíben prácticas tales como la captura o monitorización del correo electrónico de datos de usuarios, incluyendo mensajes, sin el permiso de los usuarios o autorización legítima relacionada con la operación y mantenimiento del sistema. Los datos de usuario examinados durante las obligaciones normales de operación y mantenimiento del sistema deben tratarse con estricta confidencialidad, excepto en casos en los que exista evidencia de violación de la ley, de los códigos de la organización o de este Código. En esos casos, la naturaleza o el contenido de la información sólo debe manifestarse a las autoridades adecuadas (ver 1.9).

1.8. Respetaré la confidencialidad

El principio de honestidad se extiende a las cuestiones de confidencialidad de la información siempre que se haya realizado un compromiso explícito de respetar esa confidencialidad o, implícitamente, cuando se disponga de información privada no relacionada directamente con las obligaciones asignadas. La preocupación ética es la de respetar todas las obligaciones de confidencialidad con los empresarios, clientes y usuarios a menos que se esté libre de tales obligaciones por la ley o por otros principios de este código.

2. Responsabilidades profesionales más específicas

Como profesional informático de la ACM...

2.1. Me esforzaré para alcanzar la mayor calidad, efectividad y dignidad en los procesos y productos del trabajo profesional

La excelencia es quizá la obligación más importante de un profesional. El profesional de la informática debe esforzarse para conseguir calidad y ser consciente de las graves consecuencias negativas que pueden resultar de la pobre calidad de un sistema.

2.2. Adquiriré y mantendré la capacitación profesional

La excelencia depende de los individuos que asumen la responsabilidad de conseguir y mantener su competencia profesional. Un/a profesional debe participar en la definición de los estándares para los diferentes niveles de capacitación, y debe esforzarse para alcanzarlos. La actualización del conocimiento técnico y la aptitud profesional se pueden conseguir de diferentes maneras: mediante estudio individual, asistiendo a seminarios, conferencias o cursos, e involucrándose en organizaciones profesionales.

2.3. Conoceré y respetaré las leyes existentes relacionadas con el trabajo profesional

Los afiliados a la ACM deben obedecer las leyes municipales, provinciales, autonómicas, nacionales e internacionales a menos que haya una motivación ética convincente para no hacerlo. Se deben obedecer también las políticas y procedimientos de las organizaciones en las que uno participa. Pero este cumplimiento debe ponderarse con el hecho que muchas veces las leyes y reglas existentes pueden ser inadecuadas o inmorales y deben, por tanto, trasformarse.

El incumplimiento de una ley o reglamento puede ser ético cuando la ley o regla tiene una base moral inadecuada o cuando entra en conflicto con otra ley presuntamente más importante. Si se decide eludir una regla o ley porque parece no ética, o por cualquier otro motivo, se deben aceptar la responsabilidad por las acciones tomadas y sus consecuencias completamente.

2.4. Aceptaré y proporcionaré la adecuada revisión profesional

El trabajo profesional de calidad, especialmente en informática, depende de la crítica y revisión profesional. Siempre que sea adecuado, se debe buscar y utilizar revisiones detalladas, así como proporcionar revisiones críticas del trabajo de otros.

2.5. Proporcionaré evaluaciones completas y extensas de los sistemas informáticos y sus consecuencias, incluyendo el análisis de los posibles riesgos

Los profesionales de la informática deben esforzarse en ser perceptivos, meticulosos y objetivos cuando evalúen, recomienden y presenten descripciones de sistemas y sus alternativas. Los/las informáticos están en una posición de especial relevancia, y tienen, por tanto, la responsabilidad especial de proporcionar evaluaciones objetivas y creíbles a los superiores, clientes, usuarios y público en general. Cuando se hagan evaluaciones se deben identificar los posibles conflictos de interés, como se indica en el precepto 1.3.

Como se ha señalado en el examen del principio 1.2 sobre evitar daños, debe informarse sobre cualquier signo de peligro de los sistemas a quienes tengan posibilidad y/o responsabilidad de solventarlos. Ver las guías del precepto 1.2 para más detalles concernientes al daño, incluyendo informar sobre inobservancias profesionales.

2.6. Respetaré los contratos, acuerdos y las responsabilidades asignadas

Respetar las obligaciones contraídas es una cuestión de integridad y honestidad. Para el informático esto incluye garantizar que los elementos del sistema funcionan tal como se esperaba. También cuando se trabaja como subcontrata se está obligado a informar al contratante sobre el estado del trabajo.

Un profesional de la informática tiene la responsabilidad de solicitar un cambio en cualquier asignación que prevea no poder terminar tal como se había definido. Sólo debe aceptarse una tarea después de una cuidadosa consideración y un completo examen de los riesgos y peligros para el contratante o cliente. El primer principio subyacente en este caso es la obligación de aceptar una responsabilidad personal por un trabajo profesional. En algunas ocasiones otros principios éticos pueden tener mayor prioridad.

Puede no aceptarse un juicio de valor sobre si una tarea concreta no debe realizarse. Después de haber identificado las razones para tal opinión, pero sin poder efectuar cambios en la tarea asignada, uno puede estar obligado por contrato o por ley a continuar el trabajo como se había indicado. El criterio ético del profesional es la guía final para decidir si continuar o no. Cualquiera que sea la decisión uno debe aceptar la responsabilidad de las consecuencias. Sin embargo, realizar tareas «en contra de la opinión personal» no excusa al profesional de la responsabilidad de cualquier efecto negativo.

2.7. Mejoraré la comprensión por la comunidad de la informática y sus consecuencias

Los informáticos tienen la responsabilidad de compartir su conocimiento con la sociedad, promoviendo la comprensión de la informática, incluyendo los impactos de los sistemas informáticos y sus limitaciones. Este mandato obliga a contrarrestar cualquier opinión equivocada sobre la informática.

2.8. Accederé a los recursos de comunicación e informática sólo cuando se esté autorizado a hacerlo

El robo o la destrucción de propiedad material o electrónica se prohíbe por el mandato 1.2 «Evitaré daño a otros». Este mandato se refiere a introducirse y utilizar sin permiso un sistema informático o de comunicaciones. Introducirse incluye acceder a las redes de ordenadores y a los sistemas informáticos, o a cuentas y ficheros asociados con esos sistemas, sin autorización explícita para hacerlo. Los individuos y las organizaciones tienen el derecho de restringir el acceso a sus sistemas mientras no violen el principio de discriminación (ver 1.4).

Nadie debe introducirse o utilizar sin permiso los sistemas, software o ficheros de datos de otros. Se debe obtener siempre la aprobación correspondiente antes de utilizar los recursos del sistema, incluyendo los puertos de comunicaciones, espacio de ficheros, periféricos del sistema y tiempo de ordenador.

3. Obligaciones de liderazgo organizativo

Como miembro de la ACM y líder en la organización...

3.1. Articularé las responsabilidades sociales de los miembros de una unidad organizativa y fomentaré la completa aceptación de esas responsabilidades

Puesto que todo tipo de organizaciones tienen impactos en la comunidad, deben aceptar responsabilidades con la sociedad. Los procedimientos organizativos y las actitudes orientadas hacia la calidad y el bienestar social, reducirán el daño a los individuos, sirviendo por consiguiente al interés público y cumpliendo la responsabilidad social. Por tanto, la dirección de la organización debe promover el completo cumplimiento tanto de las responsabilidades sociales como de la calidad del rendimiento.

3.2. Gestionaré personal y recursos para diseñar y construir sistemas de información que mejoren la calidad, efectividad y dignidad de la vida laboral

La dirección de la organización es responsable de garantizar que los sistemas informáticos mejoran, no degradan, la calidad de la vida laboral. Cuando se instale un sistema informático, las organizaciones deben considerar el desarrollo personal y profesional, la seguridad física y

la dignidad humana de todos los trabajadores. En el diseño del sistema y del lugar de trabajo deben tenerse en cuenta los estándares apropiados de ergonomía persona-ordenador.

3.3. Reconoceré y apoyaré los usos adecuados y autorizados de los recursos informáticos y de comunicaciones de la organización

Puesto que los sistemas informáticos pueden convertirse en herramientas tan dañinas como beneficiosas para una organización, la dirección tiene la responsabilidad de definir claramente los usos adecuados e inadecuados de sus recursos informáticos. Aunque el número y alcance de tales reglas debieran ser mínimos, deben hacerse cumplir una vez se hayan establecido.

3.4. Garantizaré que los usuarios y aquellos que se verán afectados por el sistema informático han articulado claramente sus necesidades durante la evaluación y el diseño de los requisitos

Después el sistema debe ser validado para cumplir los requisitos Las necesidades de los usuarios actuales del sistema, los potenciales y aquellas personas cuyas vidas pueden verse afectadas por el mismo, deben ser evaluadas e incorporadas en los documentos de requisitos. La validación del sistema debe garantizar el cumplimiento de esos requisitos.

3.5. Articularé y apoyaré las políticas que protegen la dignidad de los usuarios y de quienes se vean afectados por el sistema informático

No es aceptable éticamente diseñar o instalar sistemas que inadvertida o deliberadamente desprestigien personas o grupos. Los profesionales informáticos que están en posiciones de toma de decisiones deben verificar que los sistemas se diseñan e instalan para proteger la confidencialidad individual y para mejorar la dignidad personal.

3.6. Crearé condiciones para que los miembros de la organización aprendan los principios y limitaciones de los sistemas informáticos

Esto complementa el mandato sobre la educación de la comunidad (2.7). Las oportunidades educativas son esenciales para facilitar la óptima participación de todos los miembros de la organización. Las oportunidades deben estar abiertas a todos los miembros para ayudarles a mejorar su conocimiento y capacidades en informática, incluyendo cursos que los familiaricen con las consecuencias y limitaciones de sistemas concretos. En particular, los profesionales deben ser conscientes de los peligros de la construir sistemas mediante modelos demasiado simplificados, la incertidumbre de anticipar y de diseñar para cada condición de operación posible, y otros temas relacionados con la complejidad de esta profesión.

4. Conformidad con el código

Como miembro de la ACM...

4.1. Defenderé y promoveré los principios de éste código

El futuro de la profesión informática depende de la excelencia técnica y ética. No sólo es importante para los profesionales informáticos de la ACM adherirse a los principios expresados en este Código, sino que cada miembro debe animar y apoyar la adhesión de otros miembros.

4.2. Trataré los inclumplimientos de este código como inconsecuentes con la afiliación a la ACM

La adhesión de los profesionales a un código de ética es principalmente una cuestión voluntaria. Sin embargo, si un miembro no sigue este código por involucrarse en grave mala conducta, su afiliación a la ACM puede ser cancelada.

Este código y los principios generales suplementarios han sido elaborados por el Grupo de Trabajo para la Revisión del Código de Ética y de Conducta Profesional de la ACM: Ronald E. Anderson, Chair, Gerald Engel, Donald Gotterbarn, Grace C. Hertlein, Alex Hoffman, Bruce Jawer, Deborah G. Johnson, Doris K. Lidtke, Joyce Currie Little, Dianne Martin, Donn B. Parker, Judith A. Perrolle, and Richard S. Rosenberg. El Grupo de Trabajo ha sido organizado por el ACM/SIGCAS y financiado por los fondos discrecionales del ACM SIG. Este código y los principios generales suplementarios han sido adoptados por el Consejo de la ACM el 16 de Octubre de 1992.

Publicado originalmente en: http://www.acm.org/serving/se/code_s.html

Referencias

[1] Accino, J.A., C.F. Acebal, J. Ayesa, R. Fernández, M.C. Ugarte. «La traducción técnica en revistas profesionales: el ejemplo de Novática». *II Congreso Internacional: El español, lengua de traducción*. Toledo, 20-22 mayo 2004. Disponible en Internet: <http://www.toledo2004.net/html/contribuciones/accino.htm>.

[2] Alur, D., J. Crupi, D. Malks. *Core J2EE Patterns: Best Practices and Design Strategies*, 1st edition. Prentice Hall/Sun Microsystems Press, 2001. Disponible en Internet: <http://java.sun.com/blueprints/corej2eepatterns/index.html>.

[3] Apache Software Foundation. *Sitio web oficial del servidor Apache Jakarta Tomcat*. Disponible en Internet: <http://jakarta.apache.org/tomcat/>.

[4] Barbeito, E. «DBDesigner 4 - diseño de bases de datos». *Ebarbeito*, 14 marzo 2005. Disponible en Internet: <http://blog.enrique.barbeito.org/archivos/2005/03/14/dbdesigner-4-diseno-de-bases-de-datos/>.

[5] Barroso, P. *Profesionalidad y responsabilidad ética en la Informática*. Disponible en Internet: <http://www.ehu.es/zer/zer3/2artbarr.html>.

[6] Bergsten, H. *JavaServer Pages*, Third Edition. O'Reilly, 2003.

[7] Berleur, J. «Final Remarks: Ethics, Self-Regulation and Democracy» en J. Berleur, K. Brunnstein (eds.): *Ethics of Computing: codes, spaces for discussion and law*, Chapman & Hall, London, 1996, pp. 241-256.

[8] Berleur, J., M. d'Udekem-Gevers. «Codes of Ethics Within IFIP and Other Computer Societies» en J. Berleur, K. Brunnstein (eds.): *Ethics of Computing: codes, spaces for discussion and law*, Chapman & Hall, London, 1996, pp. 3-41.

[9] Bisset, A. «Computing Professional and the 'Peace Dividend' or one bomb is as good as another» en P. Barroso (ed.): *ETHICOMP96 (Ethics and Computing) Proceedings. Volume 1*, Pontifical University of Salamanca, Madrid, 6-8 November 1996, pp. 85-93.

[10] Booch, G., J. Rumbaugh, I. Jacobson. *El lenguaje unificado de modelado*. Addison Wesley Iberoamericana, Madrid, 1999.

[11] Bourn, B. *Law as Art (with apologies to Charles Black)*. Disponible en Internet: <http://www.usinternet.com/bdbourn/black.html>.

[12] Brown, G. «Is there an Ethics of Computing?». *Journal of Applied Philosophy*, Vol. 8, No. 1, 1991, pp. 19-26.

[13] Browser News. *Stats*. Disponible en Internet: <http://www.upsdell.com/BrowserNews/stat.htm>.

[14] Bynum, T.W., *Ethics and the Information Revolution*. Ponencia en el curso de verano «Ética de la Informática», Universidad Complutense de Madrid, Madrid, 1996 [no publicado].

[15] Cavaness, C. *Programming Jakarta Struts*, Second Edition. O'Reilly, June 2004.

[16] Conallen, J. *Building Web Applications with UML*, Second Edition. Addison Wesley, 2002.

[17] Conger, S., K.D. Loch, «Ethics and Computer Use». *Communications of the ACM*, December 1995, Vol. 38, No. 12, p. 31-32.

[18] Desarrolloweb.com. *Introducción a los los lenguajes del web*. Disponible en Internet: <http://www.desarrolloweb.com/manuales/27/>.

[19] DisWeb2000. *Cómo diseñar páginas web accesibles*. Disponible en Internet: <http://usuarios.discapnet.es/disweb2000/curso/index.htm>.

[20] Dubost, K. *¡Mi sitio Web es estándar! ¿Y el tuyo?*. Traducción de Fernando Gutiérrez Ferrerías. 8 abril 2002. Disponible en Internet: <http://ferguweb.tx.com.ru/w3/Web-Quality.html>.

[21] Dyck, T. «Server Databases Clash». *eWeek*, 2002. Disponible en Internet: <http://www.eweek.com/article2/0,3959,293,00.asp>.

[22] Eclipse Plugin Central. *Directorio de plugins para eclipse*. Disponible en Internet: <http://www.eclipseplugincentral.com/>.

[23] Eclipse Plugins. *Directorio de plugins para eclipse*. Disponible en Internet: <http://eclipse-plugins.2y.net/eclipse/index.jsp>.

[24] Education and Outreach Working Group (EOWG). *Quick Tips to Make Accessible Web Sites*. W3C, 2003. Disponible en Internet: <http://www.w3c.org/WAI/References/QuickTips/>.

[25] El Rincón del Vago. *Aspectos profesionales del ingeniero informático*. Disponible en Internet: <http://apuntes.rincondelvago.com/aspectos-profesionales-del-ingeniero-informatico.html>.

[26] Esquirol, J.M. (ed.). *Tecnología, ética y futuro: Actas del I Congreso Internacional de Tecnoética*. Desclée de Brouwer, Bilbao, 2002.

[27] fabFORCE.net. *Sitio web oficial de DBDesigner 4*. Disponible en Internet: <http://www.fabforce.net/dbdesigner4/>.

[28] Fagothey, Fr.A. *Right and Reason*, Second Edition. Tan Books and Publishers, Rockford, IL, 1959.

[29] Falkner, J., B. Galbraith, R. Irani, C. Kochmer, S.N. Panduranga, K. Perrumal, J. Timney, M.M. Kunnumpurath. *Fundamentos: Desarrollo Web con JSP*. Anaya Multimedia, 2002.

[30] Ford, N. *Art of Java Web Development*. Manning, 2004.

[31] Free Software Foundation. *Categorías de software libre y no libre*. Traducción de Gerardo Santana Gómez Garrido. 1999. Disponible en Internet: <http://www.gnu.org/philosophy/categories.es.html>.

[32] Free Software Foundation. *Licencias de software libre*. Traducción de Santiago Becerra Carrillo. 2003. Disponible en Internet: <http://www.gnu.org/licenses/license-list.es.html>.

[33] Gimp-es. *Sitio web oficial del Grupo de Usuarios de GIMP en Español*. Disponible en Internet: <http://gimp.hispalinux.es/>.

[34] Gómez del Moral, D. *Comparativa de herramientas UML de libre distribución (o con pocas restricciones)*. Disponible en Internet: <http://www.diatel.upm.es/malvarez/UML/Comparativa.html>.

[35] Guibert, J.M. «¿Qué es la ética de la informática?». *Universidad de Deusto*, Bilbao, 1997. Disponible en Internet: <http://paginaspersonales.deusto.es/guibert/1etic-info.html>.

[36] Hansen, K.H. «The Power of Three: Eclipse, Tomcat, and Struts». *Java Boutique*. Disponible en Internet: <http://javaboutique.internet.com/tutorials/three/>.

[37] Haym, K., M. Roth. «Code Conventions for the JavaServer Pages Technology Version 1.x Language». *Sun Developer Network*, February 2003. Disponible en Internet: <http://java.sun.com/developer/technicalArticles/javaserverpages/code_convention/>.

[38] Holvast, J. «Codes of Ethics: Discussion Paper» en INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING (IFIP), *Ethics of Computing: Information Technology and Responsibility*, Madrid, 1992.

[39] Husted, T., C. Dumoulin, G. Franciscus, D. Winterfeldt. *Struts In Action: Building web applications with the leading Java framework*. Manning Publications, Jan 2003.

[40] JafSoft Limited. *Sitio web oficial del programa AscToHTM*. Disponible en Internet: <http://www.jafsoft.com/asctohtm/>.

[41] jGuru. «Introduction to JavaServer Pages technology». *Sun Developer Network*, September 2000. Disponible en Internet: <http://java.sun.com/developer/onlineTraining/JSPIntro/>.

[42] Johansson, R. *Developing With Web Standards: Recommendations and best practices*. 456 Berea Street, last updated on July 31, 2005. Disponible en Internet: <http://www.456bereastreet.com/lab/developing_with_web_standards/>.

[43] Johnson, D.G. *Ética Informática*. Universidad Complutense, Madrid, 1996.

[44] Johnson, D.G., J.M. Mulvey. «Accountability and Computer Decision Systems». *Communications of the ACM*, December 1995, Vol. 38, No. 12, pp. 58-64.

[45] Kizza, J.M. *Ethical and Social Issues in the Information Age*, 2nd Edition. Springer, 2003.

[46] Kurniawan, B. *Java for the Web with Servlets, JSP, and EJB: A Developer's Guide to J2EE Solutions*. New Riders Publishing, 2002.

[47] López, A. *Desarrollo de un sistema de información para la gestión de casos de enfermería en el Hospital de Antequera*. Proyecto de Fin de Carrera, Escuela Técnica Superior de Ingeniería Informática, Universidad de Málaga, 2005.

[48] Lorente, S. «Autopistas de la información y previsibles impactos en la sociedad» en J.I. Ruiz Oolabuénaga (ed.): *Vida Cotidiana y Nuevas Generaciones*, Universidad de Deusto, Bilbao, 1996.

[49] Mas, J. «El código fuente abierto: el sentido común se abre paso en la informática». *Universitat Oberta de Catalunya*, diciembre 2001. Disponible en Internet: <http://www.uoc.edu/web/esp/art/uoc/mas1201/mas1201.html>.

[50] Molpeceres, A. «Diseño de software con patrones». *Programación en Castellano*, 28 agosto 2001. Disponible en Internet: <http://www.programacion.net/java/articulo/joa_patrones1/>.

[51] Moor, J.H. «What is Computer Ethics?». *Metaphilosophy*, Vol. 16, No. 4, October 1985, pp. 265-275. Disponible en Internet: <http://www.southernct.edu/organizations/rccs/resources/teaching/teaching_mono/moor/moor_definition.html>.

[52] MySQL AB. *Benchmarks*. Disponible en Internet: <www.mysql.com/information/benchmarks.html>.

[53] MySQL AB. *Sitio web oficial de MySQL*. Disponible en Internet: <http://www.mysql.com>.

[54] Objects by Design. *UML Products by Company*. Disponible en Internet: <http://www.objectsbydesign.com/tools/umltools_byCompany.html>.

[55] Parker, D.B., S. Swope, B.N. Baker. *Ethical conflicts in information and computer science, technology, and business*. QED Information Sciences, Wellesley, MA (USA), 1990.

[56] Pecorino, P., W. Maner. «The Philosopher as teacher. A proposal for course on Computer Ethics». *Metaphilosophy*, Vol. 16, No. 4, October 1985, pp. 327-337.

[57] Professional Engineers Ontario. «Environmental Guidelines for the Practice of Professional Engineering in Ontario». *Guideline of Professional Practice*, Revisión de 1998. Disponible en Internet:

<http://www.peo.on.ca/publications/Guidelines/Professional_practice_rev.pdf>.

[58] Reino, A. *Introducción a XML en castellano*. Versión 2.0, 26 enero 2000. Disponible en Internet: <www.areino.com/alf/docs/introxml.pdf>.

[59] Robinson, M., E. Finkelstein. *Jakarta Struts For Dummies*. Wiley, April 2004.

[60] Rojo, I.I. «Open Source: los programas íntegros». *Baquía*, 16 diciembre 1999. Disponible en Internet: <http://www.baquia.com/com/legacy/8512.html>.

[61] Siggelkow, B. *Jakarta Struts Cookbook*. O'Reilly, 2005.

[62] Solomon, R. *Morality and the Good Life: An Introduction to Ethics Through Classical Sources*, Segunda Edición. McGraw-Hill, New York, 1992.

[63] Sosa, J. *Sistema estadístico de encuestas On-Line*. Proyecto de Fin de Carrera, Escuela Técnica Superior de Ingeniería Informática, Universidad de Málaga, 2003.

[64] Spielman, S. *The Struts Framework: Practical Guide for Java Programmers*. Morgan Kaufmann, 2003.

[65] Storkel, S. «Eclipse -I- Historia y Toma de Contacto». Traducción de Juan Antonio Palos. *Programación en Castellano*, 17 septiembre 2004. Disponible en Internet: <http://www.programacion.com/java/articulo/jap_eclip_1/>.

[66] Sun Microsystems. *Sitio web oficial de la tecnología Java*. Disponible en Internet: <http://java.sun.com/>.

[67] Sun Microsystems. *Sitio web oficial de la tecnología JSP*. Disponible en Internet: <http://java.sun.com/products/javabeans/>.

[68] Sun Microsystems. *Sitio web oficial de la tecnología Servlet*. Disponible en Internet: <http://java.sun.com/products/servlet/>.

[69] Sun Microsystems. *Sitio web oficial de los JavaBeans*. Disponible en Internet: <http://java.sun.com/products/javabeans/>.

[70] Sun Microsystems. «Code Conventions for the Java Programming Language». *Sun Developer Network*, revised on April 20, 1999. Disponible en Internet: <http://java.sun.com/docs/codeconv/>.

[71] Sun Microsystems. *Tag Libraries Tutorial*. Last revision on 25 July, 2000. Disponible en Internet: <http://java.sun.com/products/jsp/tutorial/TagLibrariesTOC.html>.

[72] The Counter. *Browser Stats*. Disponible en Internet: <http://www.thecounter.com/stats/>.

[73] Vázquez, J.M. «Apuntes de Deontología aplicada a la Informática». *RS Cuadernos de Realidades Sociales*, Instituto de Sociología Aplicada de Madrid, nº 41/42, enero 1993, pp. 5-68.

[74] W3C. *Guía breve de accesibilidad web*. 2005. Disponible en Internet: <http://www.w3c.es/divulgacion/guiasbreves/Accesibilidad>.

[75] W3C. *Sitio web oficial del lenguaje XML*. Last modified 29 October, 2005. Disponible en Internet: <http://www.w3.org/XML/>.

[76] Weisband, S.P., B.A. Reinig. «Managing User Perceptions of Email Privacy». *Communications of the ACM*, December 1995, Vol. 38, No. 12, pp. 40-47.

[77] Welling, L., L. Thomson. *MySQL Tutorial*. Sams Publishing, 2003.

[78] Wikimedia Foundation. *Wikipedia: La enciclopedia libre*. Disponible en Internet: <http://es.wikipedia.org/wiki/Portada>.

Índice de tablas

Índice de figuras